ABSTRACT

Title of dissertation: IMPROVING EFFICIENCY AND GENERALIZATION OF VISUAL RECOGNITION

Ruichi Yu
Doctor of Philosophy, 2018

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

Deep Neural Networks (DNNs) are "heavy" in terms of their number of parameters and computational cost. This leads to two major challenges: first, training and deployment of deep networks are expensive; second, without tremendous annotated training data, which are very costly to obtain, DNNs easily suffer over-fitting and have poor generalization. We propose approaches to these two challenges in the context of specific computer vision problems to improve their efficiency and generalization.

First, we study network pruning using neuron importance score propagation. To reduce the significant redundancy in DNNs, we formulate network pruning as a binary integer optimization problem which minimizes the reconstruction errors on the final responses produced by the network, and derive a closed-form solution to it for pruning neurons in earlier layers. Based on our theoretical analysis, we propose the Neuron Importance Score Propagation (NISP) algorithm to propagate the importance scores of final responses to every neuron in the network, then prune neurons in the entire networks jointly.

Second, we study visual relationship detection (VRD) with linguistic knowledge distillation. Since the semantic space of visual relationships is huge and training data is limited, especially for long-tail relationships that have few instances, detecting visual relationships from images is a challenging problem. To improve the predictive capability, especially generalization on unseen relationships, we utilize knowledge of linguistic statistics obtained from both training annotations (internal knowledge) and publicly available text, *e.g.*, Wikipedia (external knowledge) to regularize visual model learning.

Third, we study the role of context selection in object detection. We investigate the reasons why context in object detection has limited utility by isolating and evaluating the predictive power of different context cues under ideal conditions in which context provided by an oracle. Based on this study, we propose a region-based context re-scoring method with dynamic context selection to remove noise and emphasize informative context.

Fourth, we study the efficient relevant motion event detection for large-scale home surveillance videos. To detect motion events of objects-of-interest from large scale home surveillance videos, traditional methods based on object detection and tracking are extremely slow and require expensive GPU devices. To dramatically speedup relevant motion event detection and improve its performance, we propose a novel network for relevant motion event detection, ReMotENet, which is a unified, end-to-end data-driven method using spatial-temporal attention-based 3D ConvNets to jointly model the appearance and motion of objects-of-interest in a video.

In the last part, we address the recognition of agent-in-place actions, which are associated with *agents* who perform them and *places* where they occur, in the context of

outdoor home surveillance. We introduce a representation of the geometry and topology of scene layouts so that a network can generalize from the layouts observed in the training set to unseen layouts in the test set. This Layout-Induced Video Representation (LIVR) abstracts away low-level appearance variance and encodes geometric and topological relationships of places in a specific scene layout. LIVR partitions the semantic features of a video clip into different places to force the network to learn place-based feature descriptions; to predict the confidence of each action, LIVR aggregates features from the place associated with an action and its adjacent places on the scene layout. We introduce the Agent-in-Place Action dataset to show that our method allows neural network models to generalize significantly better to unseen scenes.

IMPROVING EFFICIENCY AND GENERALIZATION OF VISUAL RECOGNITION

by

Ruichi Yu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Professor David Jacobs
Professor Ramani Duraiswami
Professor Tom Goldstein

Acknowledgments

First, I would like to express my gratitude to my advisor, Professor Larry S. Davis. As a research advisor, Larry is extraordinary. He provides me opportunities to explore a diverse area of research directions which helps me to gradually understand the whole area of computer vision and machine learning research. He always encourages me to think and propose research ideas, and even if they are immature, Larry shows a lot of patience to help me develop and polish them. He is very knowledgable in many research areas and always inspires me and provides me relevant literatures. From working with him, I not only broaden my horizon and improve my understanding of the field, but also learn how to do creative and meticulous research. Besides being a research advisor, Larry also influenced my a lot of other ways. He has a great sense of humor and it is always relaxed and enjoyable to talk to him. It is very lucky of me to have Larry as my research advisor.

Meanwhile, I also would like to thank Vlad Morariu, who has been a great research mentor during my graduate study. Vlad provides me with consistent help in discussing and polishing ideas and writing papers. The most important thing I learnt from him is that as a researcher, we should also "shoot for the moon", which means that we should start thinking of something impactful, even if it seems very difficult at the beginning. Thanks are due to Professor Rama Chellappa, Professor Tom Goldstein, Professor Ramani Duraiswami and Professor David Jacobs for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript. I would like to thank Hongcheng Wang and Jan Neumann from Comcast Applied AI research, and Chun-Fu Chen, Jui-Hsin Lai, and Ching-Yung Lin in IBM T.J. Watson Research Center. A part

of this thesis was conducted under the collaboration with them during my internship. I would like to express my thankfulness to my colleagues Ang Li, Jingxiao Zheng, Mingfei Gao and Xintong Han at the University of Maryland. All of them have provided me the greatest research collaboration and support.

During my graduate study, I have had the opportunity to study in many great graduate classes and I would like to thank all the teachers Marine Carpuat and Philip Resnik. I am also fortunate to have had multiple brilliant research mentors from internships. I would like to thank all of them for their invaluable comments and discussions which help to frame the way I conduct and present research, including Mahmudul Hasan from Comcast Labs, Junhua Mao, Alper Ayvaci and Congcong Li from Waymo. I would like to thank all of my other colleagues in the Computer Vision Lab and the CS department who has enriched my graduate life in many ways.

I would also like to acknowledge help and support from the staff members of UMD CS, UMIACS, Bluecrab cluster, Deepthought cluster and Vulcan cluster. I would like to acknowledge financial support from the Office of NavalResearch (ONR) and Dean's Fellowship from University of Maryland CS department and Graduate Research Appreciation Travel Award. I also acknowledge the University of Maryland supercomputing resources for computation support. Finally, I owe my deepest thanks to my mother, father and grandparents who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times, my heartfelt thanks to my wife, Qinyi Xu, who gives me her unlimited support and encouragement, and makes my everyday life much more colorful and joyful. It is impossible to remember all, and I apologize to those I've inadvertently left out.

Contents

List of Tables

List of Figures

List of Abbreviations

SVM	Support Vector Machine
PCA	Principal Component Analysis
SIFT	Scale Invariant Feature Transform
CRF	Conditional Random Field
NLP	Natural Language Processing
IOU	Intersection Over Union
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GT	Ground Truth
RCNN	Region-based Convolution Neural Network
DPM	Deformable Part-based Model
RNN	Recurrent Neural Network
NLL	Negative Log Likelihood
NISP	Neuron Importance Score Propagation
FRL	Final Response Layer
VRD	Visual Relationship Detection
VG	Visual Genome
ReMotENet	Relevant Motion Event Network
mAP	mean Average Precision
LIVR	Layout-Induced Video Representation
RAL	Relative Accuracy Loss
LK	Linguistic Knowledge
SF	Spatial Feature
FUB	For Upper Bound
AUB	Against Upper Bound
CS	Context Selection
FPS	Frame Per Second
PR	Precision Recall

Chapter 1: Introduction

1.1 Proposed Work

Deep Neural Networks (DNNs) are "heavy" in terms of their number of parameters and computational cost. This leads to two major challenges: first, training and deployment of deep networks are expensive; second, without tremendous annotated training data, which are very costly to obtain, DNNs easily suffer over-fitting and have poor generalization. We propose approaches to these two challenges in the context of specific computer vision problems to improve their efficiency and generalization.

In the first work, we propose Neuron Importance Score Propagation (NISP) algorithm to guide the pruning of a deep neuron network to improve its efficiency at deployment. To reduce the significant redundancy in deep Convolutional Neural Networks (CNNs), most existing methods prune neurons by only considering statistics of an individual layer or two consecutive layers (e.g., prune one layer to minimize the reconstruction error of the next layer), ignoring the effect of error propagation in deep networks. In contrast, we argue that it is essential to prune neurons in the entire neuron network jointly based on a unified goal: minimizing the reconstruction error of important responses in the "final response layer" (FRL), which is the second-to-last layer before classification, for a pruned network to retrain its predictive power. Specifically, we apply feature ranking

techniques to measure the importance of each neuron in the FRL, and formulate network pruning as a binary integer optimization problem and derive a closed-form solution to it for pruning neurons in earlier layers. Based on our theoretical analysis, we propose the Neuron Importance Score Propagation (NISP) algorithm to propagate the importance scores of final responses to every neuron in the network. The CNN is pruned by removing neurons with least importance, and then fine-tuned to retain its predictive power. NISP is evaluated on several datasets with multiple CNN models and demonstrated to achieve significant acceleration and compression with negligible accuracy loss.

In the second work, we address the problem of poor generalization capability of visual relationship detection task by utilizing external knowledge to regularize the learning process of a deep network. Understanding the visual relationship between two objects involves identifying the *subject*, the *object*, and a predicate relating them. We leverage the strong correlations between the predicate and the $\langle subj, obj \rangle$ pair (both semantically and spatially) to predict predicates conditioned on the subjects and the objects. Modeling the three entities jointly more accurately reflects their relationships compared to modeling them independently, but it complicates learning since the semantic space of visual relationships is huge and training data is limited, especially for long-tail relationships that have few instances. To overcome this, we use knowledge of linguistic statistics to regularize visual model learning. We obtain linguistic knowledge by mining from both training annotations (internal knowledge) and publicly available text, *e.g.*, Wikipedia (external knowledge), computing the conditional probability distribution of a predicate given a $\langle subj, obj \rangle$ pair. As we train the visual model, we distill this knowledge into the deep model to achieve better generalization. Our experimental results on the Visual Relation-

ship Detection (VRD) and Visual Genome datasets suggest that with this linguistic knowledge distillation, our model outperforms the state-of-the-art methods significantly, especially when predicting unseen relationships (*e.g.*, recall improved from 8.45% to 19.17% on VRD zero-shot testing set).

In the third work, we investigate the reasons why context in object detection has limited utility by isolating and evaluating the predictive power of different context cues under ideal conditions in which context provided by an oracle. Based on this study, we propose a region-based context re-scoring method with dynamic context selection to remove noise and emphasize informative context. We introduce latent indicator variables to select (or ignore) potential contextual regions, and learn the selection strategy with latent-SVM. We conduct experiments to evaluate the performance of the proposed context selection method on the SUN RGB-D dataset. The method achieves a significant improvement in terms of mean average precision (mAP), compared with both appearance based detectors and a conventional context model without the selection scheme.

In the fourth work, we address the problem of detecting relevant motion caused by objects of interest (e.g., person and vehicles) in large scale home surveillance videos. The traditional method usually consists of two separate steps, i.e., detecting moving objects with background subtraction running on the camera, and filtering out nuisance motion events with deep learning based object detection and tracking running on cloud. The method is extremely slow, and does not fully leverage the spatial-temporal redundancies with a pre-trained off-the-shelf object detector. To dramatically speedup relevant motion event detection and improve its performance, we propose a novel network for relevant motion event detection, ReMotENet, which is a unified, end-to-end data-driven method

using spatial-temporal attention-based 3D ConvNets to jointly model the appearance and motion of objects-of-interest in a video. ReMotENet parses an entire video clip in one forward pass of a neural network to achieve significant speedup, which exploits the properties of home surveillance videos, and enhances 3D ConvNets with a spatial-temporal attention model and frame differencing to encourage the network to focus on the relevant moving objects. Experiments demonstrate that our method can achieve comparable or event better performance than the object detection based method but with three to four orders of magnitude speedup (up to 20k$\times$) on GPU devices. Our network is efficient, compact and light-weight. It can detect relevant motion on a 15s surveillance video clip within 4-8 milliseconds on a GPU and a fraction of second (0.17-0.39s) on a CPU with a model size of less than 1MB.

In the end, we address the recognition of agent-in-place actions, which are associated with *agents* who perform them and *places* where they occur, in the context of outdoor home surveillance. We introduce a representation of the geometry and topology of scene layouts so that a network can generalize from the layouts observed in the training set to unseen layouts in the test set. This Layout-Induced Video Representation (LIVR) abstracts away low-level appearance variance and encodes geometric and topological relationships of places in a specific scene layout. LIVR partitions the semantic features of a video clip into different places to force the network to learn place-based feature descriptions; to predict the confidence of each action, LIVR aggregates features from the place associated with an action and its adjacent places on the scene layout. We introduce the Agent-in-Place Action dataset to show that our method allows neural network models to generalize significantly better to unseen scenes.

1.2 Previous publication

Most of the materials in this thesis has been published at top-tier venues on computer vision, and some of them are under review. The network pruning method using importance score propagation was published in IEEE Conference on Computer Vision and Pattern Recognition in 2018 [3]. The visual relationship detection using linguistic knowledge distillation was published in International Conference on Computer Vision in 2017 [4]. The efficient relevant motion detection using spatial temporal attention model was published in IEEE Winter Conference on Applications of Computer Vision in 2018 [5]. The work for exploring the role of context in object detection was published in British Machine Vision Conference in 2016 [6]. The work of agent-in-place action recognition is under review [7]. Other publications during my graduate study includes generating holistic 3D scene abstractions for text-based image retrieval [8] (IEEE Conference on Computer Vision and Pattern Recognition 2017), dynamic zoom-in network for fast object detection in large images [9] (IEEE Conference on Computer Vision and Pattern Recognition 2018), VITON: an image-based virtual try-on network [10] (IEEE Conference on Computer Vision and Pattern Recognition 2018) and C-WSL: count-guided weakly supervised localization [11] (European Conference on Computer Vision ECCV).

1.3 Organization

The thesis is organized as follows. Chapter 2 introduces the network pruning method using importance score propagation. Chapter 3 introduces the visual relation-

ship detection using linguistic knowledge distillation. Chapter4 introduces the role of context in object detection. Chapter 5 introduces the efficient relevant motion detection using spatial temporal attention. Chapter4 introduces the agent-in-place action recognition using layout-induced video representation. The thesis is then concluded in Chapter 7.

Chapter 2: Pruning Networks using Neuron Importance Score Propagation

2.1 Introduction

CNNs require a large number of parameters and high computational cost in both training and testing phases. Recent studies have investigated the significant redundancy in deep networks [12] and reduced the number of neurons and filters [13–16] by pruning the unimportant ones. However, most current approaches that prune neurons and filters consider only the statistics of one layer (*e.g.*, prune neurons with small magnitude of weights [14, 15]), or two consecutive layers [16] to determine the "importance" of a neuron. These methods prune the "least important" neurons layer-by-layer either independently [14] or greedily [15, 16], without considering all neurons in different layers jointly.

One problem with such methods is that neurons deemed unimportant in an early layer can, in fact, contribute significantly to responses of important neurons in later layers. Our experiments (see Sec.2.7) reveal that greedy layer-by-layer pruning leads to significant reconstruction error propagation, especially in deep networks, which indicates the need for a global measurement of neuron importance across different layers of a CNN.

To address this problem, we argue that it is essential for a pruned model to retain the most important responses of the second-to-last layer before classification ("final response layer" (FRL)) to retrain its predictive power, since those responses are the direct inputs of the classification task (which is also suggested by feature selection methods, *e.g.*, [17]). We define the importance of neurons in early layers based on a **unified goal**: *minimizing the reconstruction errors of the responses produced in FRL.* We first measure the importance of responses in the FRL by treating them as features and applying some feature ranking techniques (*e.g.*, [17]), then propagate the importance of neurons backwards from the FRL to earlier layers. We prune only nodes which have low propagated importance (*i.e.*, those whose removal does not result in large propagated error). From a theoretical perspective, we formulate the network pruning problem as a binary integer programming objective that minimizes the weighted ℓ^1 distance (proportional to the importance scores) between the original final response and the one produced by a pruned network. We obtain a closed-form solution to a relaxed version of this objective to infer the importance score of every neuron in the network. Based on this solution, we derive the *Neuron Importance Score Propagation* (NISP) algorithm, which computes all importance scores recursively, using only one feature ranking of the final response layer and one backward pass through the network as illustrated in Fig. 2.1.

The network is then pruned based on the inferred neuron importance scores and fine-tuned to retain its predictive capability. We treat the pruning ratio per layer as a pre-defined hyper-parameter, which can be determined based on different needs of specific applications (*e.g.*, FLOPs, memory and accuracy constraints). The pruning algorithm is generic, since feature ranking can be applied to any layer of interest and the importance

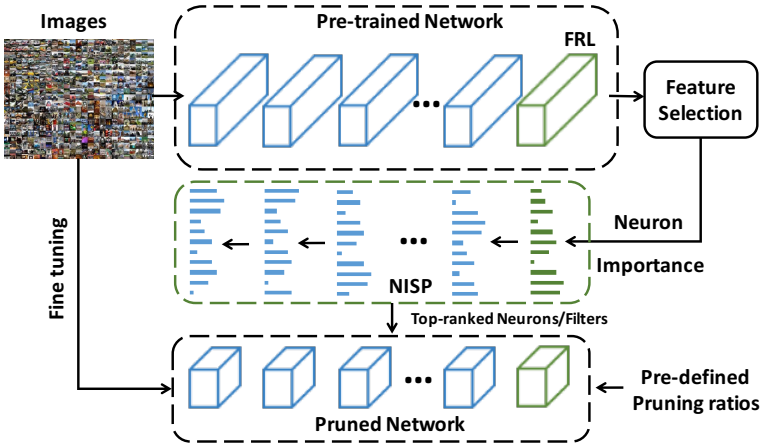


Figure 2.1: NISP: System Overview.

scores can still be propagated. In addition, NISP is not hardware specific. Given a pre-trained model, NISP outputs a smaller network of the same type, which can be deployed on the hardware devices designed for the original model.

We evaluate our approach on MNIST [18], CIFAR10 [19] and ImageNet [20] using multiple standard CNN architectures such as LeNet [18], AlexNet [21], GoogLeNet [22] and ResNet [23]. Our experiments show that CNNs pruned by our approach outperform those with the same structures but which are either trained from scratch or randomly pruned. We demonstrate that our approach outperforms magnitude-based and layer-by-layer pruning. A comparison of the theoretical reduction of FLOPs and number of parameters of different methods shows that our method achieves faster full-network acceleration and compression with lower accuracy loss, *e.g.*, our approach loses 1.43% accuracy on Alexnet and reduces FLOPs by 67.85% while Figurnov *et al.* [24] loses more (2%) and reduces FLOPs less (50%). With almost zero accuracy loss on ResNet-56, we achieve a 43.61% FLOP reduction, significantly higher than the 27.60% reduction by Li *et al.* [15].

Contribution We introduce a generic network pruning algorithm which formulates

the pruning problem as a binary integer optimization and provide a closed-form solution based on final response importance. We present NISP to efficiently propagate the importance scores from final responses to all other neurons. Experiments demonstrate that NISP leads to full-network acceleration and compression for all types of layers in a CNN with small accuracy loss.

2.2 Our Approach

An overview of NISP is illustrated in Fig. 2.1. Given a trained CNN, we first apply a feature ranking algorithm on this final response layer and obtain the importance score of each neuron. Then, the proposed NISP algorithm propagates importance scores throughout the network. Finally, the network is pruned based on the importance scores of neurons and fine-tuned to recover its accuracy.

Feature Ranking on the Final Response Layer Our intuition is that the final responses of a neural network should play key roles in full network pruning since they are the direct inputs of the classification task. So, in the first step, we apply feature ranking on the final responses.

It is worth noting that our method can work with any feature selection that scores features *w.r.t.*their classification power. We employ the recently introduced filtering method Inf-FS [17] because of its efficiency and effectiveness on CNN feature selection. Inf-FS utilizes properties of the power series of matrices to efficiently compute the importance of a feature with respect to all the other features, *i.e.*, it is able to integrate the importance of a feature over all paths in the affinity graph.

2.3 Neuron Importance Score Propagation (NISP)

Our goal is to decide which intermediate neurons to delete, given the importance scores of final responses, so that the predictive power of the network is maximally retained. We formulate this problem as a binary integer programming (optimization) and provide a closed-form approximate solution. Based on our theoretical analysis, we develop the *Neuron Importance Score Propagation* algorithm to efficiently compute the neuron importance for the whole network.

Problem Definition The goal of pruning is to remove neurons while minimizing accuracy loss. Since model accuracy is dependent on the final responses, we define our objective as minimizing the weighted distance between the original final responses and the final responses after neurons are pruned of a specific layer. In following, we use bold symbols to represent vectors and matrices.

Most neural networks can be represented as a nested function. Thus, we define a network with depth n as a function $F^{(n)} = f^{(n)} \circ f^{(n-1)} \circ \cdots \circ f^{(1)}$. The l-th layer $f^{(l)}$ is represented using the following general form,

$$f^{(l)}(\mathbf{x}) = \sigma^{(l)}(\mathbf{w}^{(l)}\mathbf{x} + \mathbf{b}^{(l)}), \tag{2.1}$$

where $\sigma^{(l)}$ is an activation function and $\mathbf{w}^{(l)}, \mathbf{b}^{(l)}$ are weight and bias, and f(n) represents the "final response layer". Networks with branch connections such as the skip connection in ResNet can be transformed to this representation by padding weights and merging layers.

We define the *neuron importance score* as a non-negative value *w.r.t.*a neuron, and

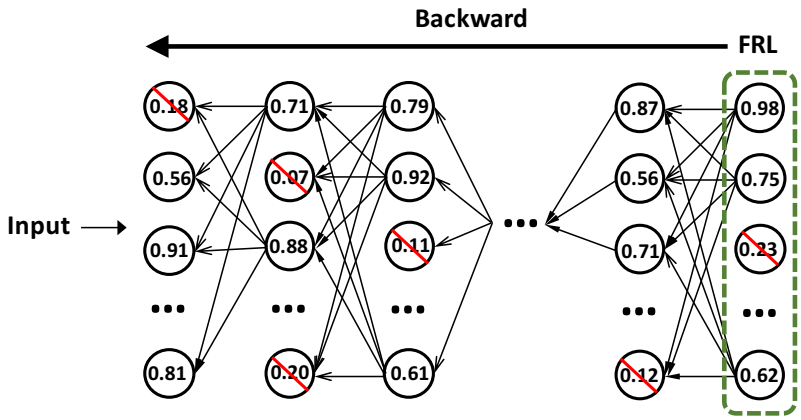


Figure 2.2: Neuron Importance Score Propagation

use $\mathbf{s}_l$ to represent the vector of neuron importance scores in the l-th layer. Suppose N_l neurons are to be kept in the l-th layer after pruning; we define the *neuron prune indicator* of the l-th layer as a binary vector $\mathbf{s}_l^*$, computed based on neuron importance scores $\mathbf{s}_l$ such that $s_{l,i}^* = 1$ if and only if $s_{l,i}$ is among top N_l values in $\mathbf{s}_l$.

Objective Function The motivation of our objective is that the difference between the responses produced by the original network and the one produced by the pruned network should be minimized w.r.t. important neurons. Let $F^{(n)}$ be a neural network with n layers. Suppose we have a dataset of M samples, and each is represented using $\mathbf{x}_0^{(m)}$. For the m-th sample, we use $\mathbf{x}_l^{(m)}$ to represent the response of the l-th layer (which is the input to the $(l+1)$-th layer). The final output of the network is $\mathbf{x}_n^{(m)}$ and its corresponding non-negative neuron importance is $\mathbf{s}_n$. We define

$$G^{(i,j)} = f^{(j)} \circ f^{(j-1)} \circ \cdots \circ f^{(i)} \tag{2.2}$$

as a sub-network of $F^{(n)}$ starting from the i-th layer to the j-th layer. Our goal is to compute for the l-th layer the neuron prune indicator $\mathbf{s}_l^*$ so that the influence of pruning

the l-th layer on the important neurons of the final response is minimized. To accomplish this, we define an optimization objective w.r.t. the l-th layer neuron prune indicator, *i.e.*,

$$\arg\min_{\mathbf{s}_l^*} \sum_{m=1}^{M} \mathcal{F}(\mathbf{s}_l^* | \mathbf{x}_l^{(m)}, \mathbf{s}_n; G^{(l+1,n)}) \ , \tag{2.3}$$

which is accumulated over all samples in the dataset. The objective function for a single sample is defined as

$$\mathcal{F}(\mathbf{s}_l^* | \mathbf{x}, \mathbf{s}_n; F) = \langle \ \mathbf{s}_n, \ |F(\mathbf{x}) - F(\mathbf{s}_l^* \odot \mathbf{x})| \ \rangle \ , \tag{2.4}$$

where $\langle \cdot, \cdot \rangle$ is dot product, $\odot$ is element-wise product and $|\cdot|$ is element-wise absolute value. The solution to Eq. 2.3 indicates which neurons should be pruned in an arbitrary layer.

Solution The network pruning problem can be formulated as a binary integer program, finding the optimal neuron prune indicator in Eq. 2.3. However, it is hard to obtain efficient analytical solutions by directly optimizing Eq. 2.3. So we derive an upper bound on this objective, and show that a sub-optimal solution can be obtained by minimizing the upper bound. Interestingly, we find a feasible and efficient formulation for the importance scores of all neurons based on this sub-optimal solution.

Recall that the k-th layer is defined as $f^{(k)}(\mathbf{x}) = \sigma^{(k)}(\mathbf{w}^{(k)}\mathbf{x} + \mathbf{b}^{(k)})$. We assume the activation function $\sigma^{(k)}$ is Lipschitz continuous since it is generally true for most of the commonly used activations in neural networks such as Identity, ReLU, sigmoid, tanh, PReLU, *etc*. Then we know for any $\mathbf{x}, \mathbf{y}$, there exists a constant $C_\sigma^{(k)}$ such that $|\sigma^{(k)}(\mathbf{x}) - \sigma^{(k)}(\mathbf{y})| \leq C_\sigma^{(k)} |\mathbf{x} - \mathbf{y}|$. Then it is easy to see

$$|f^{(k)}(\mathbf{x}) - f^{(k)}(\mathbf{y})| \leq C_\sigma^{(k)} |\mathbf{w}^{(k)}| \cdot |\mathbf{x} - \mathbf{y}| \ , \tag{2.5}$$

where $|\cdot|$ is the element-wise absolute value. From Eq. 2.2, we see that $G^{(i,j)} = f^{(j)} \circ G^{(i,j-1)}$. Therefore, we have,

$$\begin{aligned} &|G^{(i,j)}(\mathbf{x}) - G^{(i,j)}(\mathbf{y})| \\ &\quad \leq C_\sigma^{(j)} |\mathbf{w}^{(j)}| |G^{(i,j-1)}(\mathbf{x}) - G^{(i,j-1)}(\mathbf{y})| \,. \end{aligned} \tag{2.6}$$

Applying Eq. 2.5 and Eq. 2.6 repeatedly, we have, $\forall i \leq j \leq n$,

$$|G^{(i,n)}(\mathbf{x}) - G^{(i,n)}(\mathbf{y})| \leq C_\Sigma^{(i,n)} \mathbf{W}^{(i,n)} |\mathbf{x} - \mathbf{y}|, \tag{2.7}$$

where $\mathbf{W}^{(i,j)} = |\mathbf{w}^{(j)}||\mathbf{w}^{(j-1)}| \cdots |\mathbf{w}^{(i)}|$, and $C_\Sigma^{(i,j)} = \prod_{k=i}^{j} C_\sigma^{(k)}$. Substituting $\mathbf{x} = \mathbf{x}_l^{(m)}, \mathbf{y} = \mathbf{s}_l^* \odot \mathbf{x}_l^{(m)}, i = l+1$ into Eq. 2.7, we have

$$\begin{aligned} &|G^{(l+1,n)}(\mathbf{x}_l^{(m)}) - G^{(l+1,n)}(\mathbf{s}_l^* \odot \mathbf{x}_l^{(m)})| \\ &\quad \leq C_\Sigma^{(l+1,n)} \mathbf{W}^{(l+1,n)} |\mathbf{x}_l^{(m)} - \mathbf{s}_l^* \odot \mathbf{x}_l^{(m)}| \,. \end{aligned} \tag{2.8}$$

Since $\mathbf{s}_n$ is a non-negative vector,

$$\begin{aligned} &\mathcal{F}(\mathbf{s}_l^* | \mathbf{x}_l^{(m)}, \mathbf{s}_n; G^{(l+1,n)}) \\ &\quad = \langle \mathbf{s}_n, |G^{(l+1,n)}(\mathbf{x}_l^{(m)}) - G^{(l+1,n)}(\mathbf{s}_l^* \odot \mathbf{x}_l^{(m)})| \rangle && (2.9) \\ &\quad \leq \langle \mathbf{s}_n, C_\Sigma^{(l+1,n)} \mathbf{W}^{(l+1,n)} |\mathbf{x}_l^{(m)} - \mathbf{s}_l^* \odot \mathbf{x}_l^{(m)}| \rangle && (2.10) \\ &\quad = C_\Sigma^{(l+1,n)} \langle \mathbf{W}^{(l+1,n)\mathsf{T}} \mathbf{s}_n, (\mathbf{1} - \mathbf{s}_l^*) \odot |x_l^{(m)}| \rangle \,. && (2.11) \end{aligned}$$

Let us define $\mathbf{r}_l = \mathbf{W}^{(l+1,n)\mathsf{T}} \mathbf{s}_n$; then

$$\begin{aligned} &\textstyle\sum_{m=1}^{M} \mathcal{F}(\mathbf{s}_l^* | \mathbf{x}_l^{(m)}, \mathbf{s}_n; G^{(l+1,n)}) \\ &\quad \leq C_\Sigma^{(l+1,n)} \textstyle\sum_{m=1}^{M} \langle \mathbf{r}_l, (\mathbf{1} - \mathbf{s}_l^*) \odot |\mathbf{x}_l^{(m)}| \rangle && (2.12) \\ &\quad \leq C_\Sigma^{(l+1,n)} \textstyle\sum_{m=1}^{M} \sum_i r_{l,i} (1 - s_{l,i}^*) |x_{l,i}^{(m)}| && (2.13) \\ &\quad = C_\Sigma^{(l+1,n)} \textstyle\sum_i r_{l,i} (1 - s_{l,i}^*) \sum_{m=1}^{M} |x_{l,i}^{(m)}| \,. && (2.14) \end{aligned}$$

Since $|\mathbf{x}_{l,i}^{(m)}|$ is bounded, there must exist a constant C_x such that $\sum_{m=1}^{M} |x_{l,i}^{(m)}| \leq C_x, \forall i$. Thus, we have

$$\sum_{m=1}^{M} \mathcal{F}(\mathbf{s}_l^* | \mathbf{x}_l^{(m)}, \mathbf{s}_n; F^{(l+1)}) \leq C \sum_i r_{l,i}(1 - s_{l,i}^*), \tag{2.15}$$

where $C = C_{\Sigma}^{(l+1,n)} C_x$ is a constant factor.

Eq. 2.15 reveals an upper-bound of our objective in Eq. 2.3. Thus, we minimize this upper-bound, *i.e.*,

$$\arg\min_{\mathbf{s}_l^*} \sum_i r_{l,i}(1 - s_{l,i}^*) \Leftrightarrow \arg\max_{\mathbf{s}_l^*} \sum_i s_{l,i}^* r_{l,i} \ . \tag{2.16}$$

The optimal solution to Eq.2.16 is sub-optimal with respect to the original objective in Eq. 2.3, however it still captures the importance of neurons. It is easy to see that if we keep N_x neurons in the l-th layer after pruning, then the solution to Eq. 2.16 is that $s_{l,i}^* = 1$ if and only if $r_{l,i}$ is among the highest N_x values in $\mathbf{r}_l$. According to the definition of neuron prune indicator in Sec. 2.3, $\mathbf{r}_l = \mathbf{W}^{(l+1,n)\intercal} \mathbf{s}_n$ is a feasible solution to the importance scores of the l-th layer response. This conclusion can be applied to every layer in the network. Based on this result, we define the neuron importance of a network as follows.

Definition 2.3.1 (Neuron importance score)**.** *Given a neural network $F^{(n)}$ containing n layers and the importance score $\mathbf{s}^{(n)}$ of the last layer response, the importance score of the k-th layer response can be computed as*

$$\mathbf{s}_k = |\mathbf{w}^{(k+1)}|^{\intercal} |\mathbf{w}^{(k+2)}|^{\intercal} \cdots |\mathbf{w}^{(n)}|^{\intercal} \mathbf{s}_n, \tag{2.17}$$

where $\mathbf{w}^{(i)}$ is the weight matrix of the i-th layer.

An important property of neuron importance is that it can be computed recursively (or propagated) along the network.

Proposition 2.3.2 (Neuron importance score propagation)**.** *The importance score of the k^{th} layer response can be propagated from the importance score of the $(k+1)^{th}$ layer by*

$$\mathbf{s}_k = |\mathbf{w}^{(k+1)}|^\intercal \mathbf{s}_{k+1}, \tag{2.18}$$

where $\mathbf{w}^{(k+1)}$ is the weight matrix of the $(k+1)^{th}$ layer.

Algorithm We propose the *Neuron Importance Score Propagation* (NISP) algorithm (shown in Fig. 2.2) based on Proposition 2.3.2. Initially, we have the importance score of every neuron in the final response layer of the network. Definition 2.3.1 shows that the importance score of every other layer in the network is directly correlated with the importance of the final response. However, instead of computing the importance expensively using Definition 2.3.1, we see from Eq. 2.18 that the importance score of a lower layer can be propagated directly from the adjacent layer above it. An equivalent form of Eq. 2.18 is

$$s_{k,j} = \textstyle\sum_i |w_{i,j}^{(k+1)}| s_{k+1,i}, \tag{2.19}$$

where $s_{k,j}$ is the importance score of the j-th neuron in the k-th layer response.

We conclude from Eq. 2.19 that the importance of a neuron is a weighted sum of all the subsequent neurons that are directly connected to it. This conclusion also applies to normalization, pooling and branch connections in the network (*i.e.*, a layer is directly connected with multiple layers). The NISP algorithm starts with the importance in FRL and repeats the propagation (Eq. 2.19) to obtain the importance of all neurons in the network with a single backward pass (Fig. 2.1).

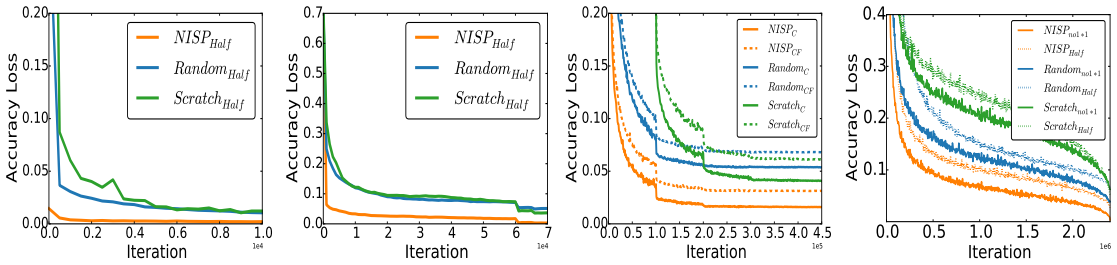
Pruning Networks Using NISP Given target pruning ratios for each layer, we propagate the importance scores, compute the prune indicator of neurons based on their importance scores and remove neurons with prune indicator value 0. The importance propagation and layer pruning happens jointly in a single backward pass, and the importance of a pruned neuron is not propagated to any further low-level layers. For fully connected layers, we prune each individual neuron. For convolution layers, we prune a whole channel of neurons together. The importance score of a channel is computed as the summation of the importance scores of all neurons within this channel.

2.4 Experiments

We evaluate our approach on standard datasets with popular CNN networks. We first compare to *random pruning* and *training-from-scratch* baselines to demonstrate the effectiveness of our method. We then compare to two other baselines, *magnitude-based pruning* and *layer-by-layer pruning* to highlight the contributions of feature ranking and neuron importance score propagation, respectively. Finally, we benchmark the pruning results and compare to existing methods such as [15, 24–26].

Experimental Setting We conduct experiments on three datasets, MNIST [18], CIFAR10 and ImageNet [20], for the image classification task. We evaluate using five commonly used CNN architectures: *LeNet* [18], *Cifar-net*, *AlexNet* [21], *GoogLeNet* [22] and *ResNet* [23].

All experiments and time benchmarks are obtained using Caffe [27]. The hyper-parameter of Inf-FS is a loading coefficient $\alpha \in [0, 1]$, which controls the influence of



(a) MNIST (b) CIFAR10 (c) ImageNet: AlexNet (d) ImageNet: GoogLeNet

Figure 2.3: Learning curves of random pruning and training from scratch baselines and NISP using different CNNs on different datasets. The pruning ratio of neurons and filters is 50%. Networks pruned by NISP (orange curves) converge the fastest with the lowest accuracy loss.

variance and correlation when measuring the importance. We conduct PCA accumulated energy analysis as suggested in [28] to guide our choice of pruning ratios.

2.5 Comparison with Random Pruning and Train-from-scratch Baselines

We compare to two baselines: (1) randomly pruning the pre-trained CNN and then fine-tuning, and (2) training a small CNN with the same number of neurons/filters per layer as our pruned model from scratch. We use the same experimental settings for our method and baselines except for the initial learning rate. For training from scratch, we set the initial learning rate to the original one, while for fine-tuning tasks (both NISP and random pruning), the initial learning rate is reduced by a factor of 10.

LeNet on MNIST: We prune half of the neurons in FC layers and half of the filters in both convolution layers in Fig. 2.3(a). Our method is denoted as $NISP_{Half}$, while the

baseline methods that prune randomly or train from scratch are denoted as $Random_{Half}$ and $Scratch_{Half}$. Our method outperforms the baselines in three aspects. First, for fine-tuning (after pruning), unlike the baselines, our method has very small accuracy loss at iteration 0; this implies that it retains the most important neurons, pruning only redundant or less discriminative ones. Second, our method converges much faster than the baselines. Third, our method has the smallest accuracy loss after fine-tuning. For LeNet on MNIST, our method only decreases 0.02% top-1 accuracy with a pruning ratio of 50% as compared to the pre-pruned network.

Cifar-net on CIFAR10: The learning curves are shown in Fig. 2.3(b). Similar to the observations from the experiment for LeNet on MNIST, our method outperforms the baselines in the same three aspects: the lowest initial loss of accuracy, the highest convergence speed and the lowest accuracy loss after fine-tuning. Our method has less than 1% top-1 accuracy loss with 50% pruning ratio for each layer.

AlexNet on ImageNet: To demonstrate that our method works on large and deep CNNs, we replicate experiments on AlexNet with a pruning ratio of 50% for all convolution layers and FC layers (denoted as $NISP_{CF}$ when we prune both conv and FC layers). Considering the importance of FC layers in AlexNet, we compare one more scenario in which our approach only prunes half of the filters but without pruning neurons in FC layers (denoted as $NISP_{C}$). We reduce the initial learning rate by a factor of 10, then fine-tune 90 epochs and report top-5 accuracy loss. Fig. 2.3(c) shows that for both cases (pruning both convolution and FC layers and pruning only convolution layers), the advantages we observed on MNIST and CIFAR10 still hold.

GoogLeNet on ImageNet: We denote the reduction layers in an inception mod-

ule as "Reduce", and the 1×1 convolution layer without reduction as "1×1". We use the quick solver from Caffe in training. We conduct experiments between our method and the baselines for 3 pruning strategies: (*Half*) pruning all convolution layers by half; (*noReduce*) pruning every convolution layer except for the reduction layers in inception modules by half; (*no1x1*) pruning every convolution layer by half except the 1×1 layers in inception modules. We show results for two of them in Fig. 2.3(d), and observe similar patterns to the experiments on other CNN networks. For all GoogLeNet experiments, we train/fine-tune for 60 epochs and report top-5 accuracy loss.

2.6 Feature Selection v.s. Magnitude of Weights

How to define neuron importance is an open problem. Besides using feature ranking to measure neuron importance, other methods [14–16] measure neuron importance by magnitude of weights. To study the effects of different criteria to determine neuron importance, we conduct experiments by fixing other parts of NISP and only comparing the pruning results with different measurements of importance: 1. using feature selection method in [17] (NISP-FS) and 2. considering only magnitude of weights (NISP-Mag). For the Magnitude-based pruning, the importance of a neuron in the final response layer equals the absolute sum of all weights connecting the neuron with its previous layer. To compare only the two metrics of importance, we rank the importance of neurons in the final response layer based on the magnitude of their weight values, and propagate their importance to the lower layers. Finally, we prune and fine-tune the model in the same way as the NISP method.

For the "NISP-Mag" baseline, we use both AlexNet and Cifar-net architectures. The learning curves of those baselines are shown in Fig. 2.4. We observe that "NISP-FS" yields much smaller accuracy loss with the same pruning ratio than "NISP-Mag", but "NISP-Mag" still outperforms the random pruning and train-from-scratch baselines, which shows the effectiveness of NISP with different measurement of importance. In the remainder of this paper, we employ the feature ranking method proposed in [17] in NISP.

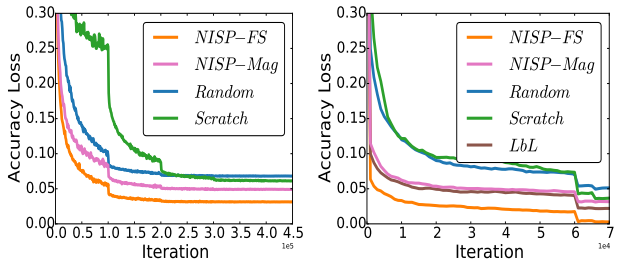
2.7 NISP v.s. Layer-by-Layer Pruning

To demonstrate the advantage of the NISP's importance propagation, we compare with a pruning method that conducts feature ranking on every layer to measure the neuron importance and prune the unimportant neurons of each layer independently. All other settings are the same as NISP. We call this method "Layer-by-Layer" (LbL) pruning.

One challenge for the "LbL" baseline is that the computational cost of measuring neuron importance on each layer is huge. So we choose a small CNN structure trained on the CIFAR10 dataset. Fig. 2.4(b) shows that although the "LbL" method outperforms the baselines, it performs much worse than NISP in terms of the final accuracy loss with the same pruning ratio, which shows the need for measuring the neuron importance across the entire network using NISP.

To further study the advantage of NISP over layer-by-layer pruning, we define the Weighted Average Reconstruction Error (WARE) to measure the change of the important neurons' responses on the final response layer after pruning (without fine-tuning) as:

$$\text{WARE} = \frac{\sum_{m=1}^{M} \sum_{i=1}^{N} s_i \cdot \frac{|\hat{y}_{i,m} - y_{i,m}|}{|y_{i,m}|}}{M \cdot N}, \tag{2.20}$$



(a) AlexNet on ImageNet (b) Cifar-net on CIFAR10

Figure 2.4: Comparison with layer-by-layer (LbL) and magnitude based (Mag) pruning baselines.

where M and N are the number of samples and number of retained neurons in the final response layer; s_i is the importance score; $y_{i,m}$ and $\hat{y}_{i,m}$ is the response on the m^{th} sample of the i^{th} neuron before/after pruning.

We design different Cifar-net-like CNNs with different numbers of Conv layers, and apply NISP and LbL pruning with different pruning ratios. We report the WARE on the retained neurons in the final response layer ("ip1" layer in Cifar-net-like CNNs) in Fig. 2.5. We observe that: 1. As network depth increases, the WARE of the LbL-pruned network dramatically increases, which indicates the error propagation problem of layer-by-layer pruning, especially when the network is deep, and suggests the need for a global pruning method such as NISP; 2. The WARE of the LbL method becomes much larger when the pruning ratio is large, but is more stable when using NISP to prune a network; 3. NISP methods always reduce WARE on the retained neurons compared to LbL. The small reconstruction errors on the important neurons in the final response layer obtained by NISP provides a better initialization for fine-tuning, which leads to much lower accuracy loss of the pruned network.

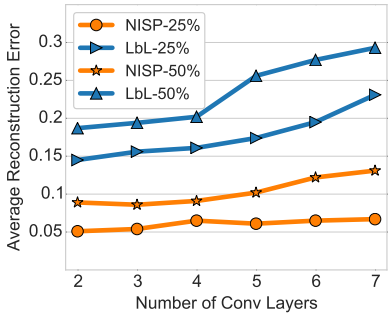


Figure 2.5: Weighted Average Reconstruction Error (WARE) on the final responses without fine-tuning.

2.8 Comparison with Existing Methods

We compare our method with existing pruning methods on AlexNet, GoogLeNet and ResNet, and show results in Table 2.1.

In Table 2.1, for AlexNet, the pruning ratio is 50%. NISP-A denotes pruning all Conv layers; NISP-B denotes pruning all Conv layers except for Conv5; NISP-C denotes pruning all Conv layers except for Conv5 and Conv4; NISP-D means pruning Conv2, Conv3 and FC6 layers. For GoogLeNet, we use the similar the pruning ratios of the 3×3 layers in [25], and we prune 20% of the reduce layers. Our method is denoted as "NISP".

To compare theoretical speedup, we report reduction in the number of multiplication and the number of parameters following [25] and [24], and denote them as [FLOPs$\downarrow\%$] and [Params.$\downarrow\%$] in the table. Pruning a CNN is a trade-off between efficiency and accuracy. We compare different methods by fixing one metric and comparing the other.

On AlexNet, by achieving smaller accuracy loss (1.43% ours vs. 2.00% [24]), our method NISP-A manages to reduce significantly more FLOPs (67.85%) than the one

[1] A negative value here indicates an improved model accuracy.

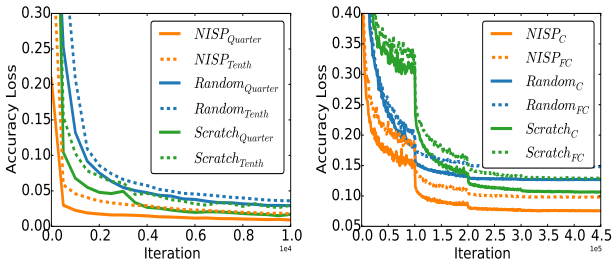
in [24] (50%), denoted as "Perforate" in the table; compare to the method in [26] (denoted as "Learning"), our method NISP-C achieves much smaller accuracy loss (0.54% ours vs. 1.20%) and prunes more FLOPs (53.70% ours vs. 48.19%). We manage to achieve 0 accuracy loss and reduce over 40% FLOPs and 47.09% parameters (NISP-D). On GoogLeNet, Our method achieves similar accuracy loss with larger FLOPs reduction (58.34% vs. 51.50%) Using ResNet on Cifar10 dataset, with top-1 accuracy loss similar to [15] (56-A, 56-B. 110-A and 110-B), our method reduces more FLOPs and parameters.

We also conduct our ResNet experiments on ImageNet [20]. We train a ResNet-34 and a ResNet-50 for 90 epochs. For both ResNet models, we prune 15% and 25% of filters for each layer (denote as "NISP-X-A" and "NISP-X-B" ("X" indicates the ResNet model) in Table 2.1), and obtain 27-44% FLOPs and parameter reduction with tiny top-1 accuracy loss, which shows superior performance when compared with the state-of-the-art methods [15, 16].

2.9 Additional Analysis

Below, we provide case studies and ablation analysis to help understand the proposed NISP pruning algorithm.

Similar Predictive Power of Networks Before/After Pruning. To check whether the pruned network retains similar predictive power with the original network, we compare the final 1000-category classification results of the original AlexNet and the pruned one with fine-tuning using the ILSVRC2012 validation set. 85.9% of the top 1 predictions of the two networks agree with each other, and 95.1% top 1 predictions of the pruned net-



(a) LeNet Prune 75% and 90%

(b) AlexNet Prune 75%

Figure 2.6: Evaluations for different pruning ratios.

work can be found in the top 5 predictions of the original network. The above experiments show that the network pruned by NISP performs similarly with the original one.

Recursive pruning. One advantage of NISP is that it needs only a one-pass pruning with one-time fine-tuning, which is very efficient and it reduces significant computational cost. To study whether recursive pruning helps, we conduct a recursive pruning experiment on AlexNet. The final goal of the pruning ratio is 50% for all layers. Instead of direct pruning, we prune 25% of neurons prune twice using NISP. We observe that the importance score ranking of neurons generated by this recursive method slightly differs from the ranking list produced by the proposed direct approach, *e.g.*, among top 50% most important neurons in both ranking lists, 84.27% neurons overlap. The final accuracy change between direct pruning (76.38% top-5 accuracy) and recursive pruning (76.71%) is also small. We conclude that a neuron's importance score may change during recursive pruning, but after fine-tuning with a small learning rate, the change becomes negligible.

Sensitivity of pruning ratios. The selection of per-layer pruning ratios given a FLOPs budget is a challenging open problem with a large search space. Due to time

limitation, we either choose a single pruning ratio for all layers or replicate the pruning ratios of baseline methods (*e.g.*, [25]), and NISP achieves smaller accuracy loss, which shows the effectiveness of NISP. In practice, if time and GPU resources permit, one can search the optimal hyper-parameters by trying different pruning ratio combinations on a validation set.

To study the effect of different pruning ratios, we experimented on pruning only a single layer or a type of layers by 50% and monitored the accuracy loss. We observed that the sensitivities of pruning per layer are similar to each other for some networks (*e.g.*, ResNet); some layers are more sensitive to a large pruning ratio (*e.g.*, the Conv1 layer of AlexNet and the "1x1" layers of GoogLeNet).

We also evaluate NISP with very large pruning ratios. We test on pruning ratios of 75% (denoted as *Quarter* in the figures) and 90% using LeNet (Fig. 2.6(a)) (denoted as *Tenth*) for both Conv and FC layers. For AlexNet (Fig. 2.6(b)), we test on pruning ratios of 75% (*Quarter*) for both convolution and FC layers, and we test two pruning strategies: (1) prune 75% of neurons in FC layers and filters in Conv layers, denoted as *FC*; and (2) only prune 75% of the convolution filters without pruning FC layers, denoted as *C*.

The above experiments show that NISP still outperforms all baselines significantly with large pruning ratios, in terms of both convergence speed and final accuracy.

2.10 Conclusion

We proposed a generic framework for network compression and acceleration based on identifying the importance levels of neurons. Neuron importance scores in the layer

of interest (usually the last layer before classification) are obtained by feature ranking. We formulated the network pruning problem as a binary integer program and obtained a closed-form solution to a relaxed version of the formulation. We presented the Neuron Importance Score Propagation algorithm that efficiently propagates the importance to every neuron in the whole network. The network is pruned by removing less important neurons and fine-tuned to retain its predicative capability. Experiments demonstrated that our method effectively reduces CNN redundancy and achieves full-network acceleration and compression.

	Model	Accu.↓%	FLOPs↓%	Params.↓%
AlexNet	NISP-A	**1.43**	**67.85**	33.77
on ImageNet	Perforated [24]	2.00	50.00	-
	NISP-B	**0.97**	**62.69**	1.96
	Tucker [25]	1.70	62.55	-
	NISP-C	**0.54**	**53.70**	2.91
	Learning [26]	1.20	48.19	-
	NISP-D	0.00	40.12	47.09
GoogLeNet	NISP	**0.21**	**58.34**	**33.76**
on ImageNet	Tucker [25]	0.24	51.50	31.88
ResNet	NISP-56	0.03	**43.61**	**42.60**
on CIFAR10	56-A [15]	**-0.06**[1]	10.40	9.40
	56-B [15]	-0.02	27.60	13.70
	NISP-110	0.18	**43.78**	**43.25**
	110-A [15]	**0.02**	15.90	2.30
	110-B [15]	0.23	38.60	32.40
ResNet	NISP-34-A	**0.28**	27.32	27.14
on ImageNet	NISP-34-B	0.92	**43.76**	**43.68**
	Res34 [15]	1.06	24.20	-
	NISP-50-A	**0.21**	27.31	27.12
	NISP-50-B	0.89	**44.01**	**43.82**
	Res50 [16]	0.84	36.79	33.67

Table 2.1: Compression Benchmark.

Chapter 3: Visual Relationship Detection with Internal and External Linguistic Knowledge Distillation

3.1 Introduction

Detecting visual relationships from images is a central problem in image understanding. Relationships are commonly defined as tuples consisting of a subject (*subj*), predicate (*pred*) and object (*obj*) [29–31]. *Visual* relationships represent the visually observable interactions between subject and object $\langle subj, obj \rangle$ pairs, such as $\langle person, ride, horse \rangle$ [1].

Recently, Lu *et al.* [1] introduce the visual relationship dataset (VRD) to study learning of a large number of visual relationships from images. Lu *et al.*predict the predicates independently from the subjects and objects, and use the product of their scores to predict relationships present in a given image using a linear model. The results in [1] suggest that predicates cannot be predicted reliably with a linear model that uses only visual cues, even when the ground truth categories and bounding boxes of the subject and object are given ([1] reports Recall@100 of only 7.11% for their visual prediction). Although the visual input analyzed by the CNN in [1] includes the subject and object, predicates are predicted without any knowledge about the object categories present in the image or their

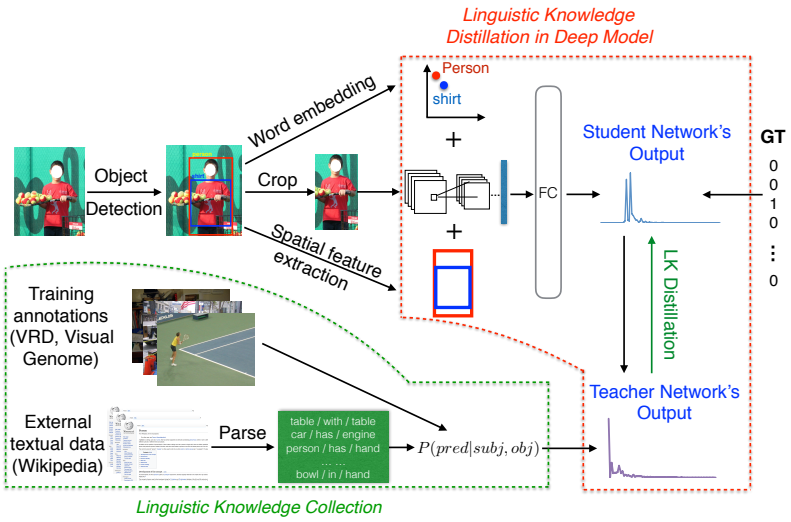


Figure 3.1: Linguistic Knowledge Distillation Framework.

relative locations. In contrast, we propose a probabilistic model to predict the predicate name jointly with the subject and object names and their relative spatial arrangement:

$$P(R|I) = P(pred|I_{\text{union}}, subj, obj)P(subj)P(obj). \tag{3.1}$$

While our method models visual relationships more accurately than [1], our model's parameter space is also enlarged because of the large variety of relationship tuples. This leads to the challenge of insufficient labeled image data. The straightforward—but very costly—solution is to collect and annotate a larger image dataset that can be used to train this model. Due to the long tail distribution of relationships, it is hard to collect enough training images for all relationships. To make the best use of available training images, we leverage linguistic knowledge (LK) to regularize the deep neural network. One way to obtain linguistic knowledge is to compute the conditional probabilities $P(pred|subj, obj)$ from the training annotations.

However, the number of $\langle subj, pred, obj \rangle$ combinations is too large for each triplet to be observed in a dataset of annotated images, so the internal statistics (*e.g.*, statistics of

the VRD dataset) only capture a small portion of the knowledge needed. To address this long tail problem, we collect external linguistic knowledge ($P(pred|subj, obj)$) from public text on the Internet (Wikipedia). This external knowledge consists of statistics about the words that humans commonly use to describe the relationship between subject and object pairs, and importantly, it includes pairs unseen in our training data. Although the external knowledge is more general, it can be very noisy (*e.g.*, due to errors in linguistic parsing).

We make use of the internal and external knowledge in a teacher-student knowledge distillation framework [32,33], shown in Figure 1, where the output of the standard vision pipeline, called the *student* network, is augmented with the output of a model that uses the linguistic knowledge to score solutions; their combination is called the *teacher* network. The objective is formulated so that the student not only learns to predict the correct one-hot ground truth labels but also to mimic the teacher's soft belief between predicates.

Our main contribution is that we exploit the role of both visual and linguistic representations in visual relationship detection and use internal and external linguistic knowledge to regularize the learning process of an end-to-end deep neural network to significantly enhance its predictive power and generalization. We evaluate our method on the VRD [1] and Visual Genome (VG) [2] datasets. Our experiments using Visual Genome show that while the improvements due to training set size are minimal, improvements due to the use of LK are large, implying that with current dataset sizes, it is more fruitful to incorporate other types knowledge (*e.g.*, LK) than to increase the visual dataset size—this is particularly promising because visual data is expensive to annotate and there exist many readily available large scale sources of knowledge that have not yet been fully leveraged

for visual tasks.

3.2 Our Approach

A straightforward way to predict relationship predicates is to train a CNN on the union of the two bounding boxes that contain the two objects of interest as the visual input, fuse semantic features (that encode the object categories) and spatial features (that encode the relative positions of the objects) with the CNN features (that encode the appearance of the objects), and feed them into a fully connected (FC) layer to yield an end-to-end prediction framework. However, the number of $\langle subj, pred, obj \rangle$ tuples is very large and the parameter space of the end-to-end CNN would be huge. While the subject, predicate, and object are not statistically independent, a CNN would require a massive amount of data to discover the dependence structure while also learning the mapping from visual features to semantic relationships. To avoid over-fitting and achieve better predictive power without increasing the amount of visual training data, additional information is needed to help regularize the training of the CNN.

Figure 3.1 summarizes our proposed model. Given an image, we extract three input components: the cropped images of the union of the two detected objects (BB-Union); the semantic object representations obtained from the object category confidence score distributions obtained from the detector; and the spatial features (SF) obtained from pairs of detected bounding boxes. We concatenate VGG features, semantic object vectors, and the spatial feature vectors, then train another FC layer using the ground truth label (GT) and the linguistic knowledge to predict the predicate. Unlike [1], which used the

VGG features to train a linear model, our training is end-to-end without fixing the VGG-net. Following [32, 33], we call the data-driven model the "student", and the linguistic knowledge regularized model the "teacher".

3.3 Linguistic Knowledge Distillation

Preliminary: Incorporating Knowledge in DNNs The idea of incorporating additional information in DNNs has been exploited recently [32–34]. We adapted Hu *et al.*'s teacher-student framework [32, 33] to distill linguistic knowledge in a data-driven model. The teacher network is constructed by optimizing the following criterion:

$$\min_{t \in T} \mathrm{KL}(t(Y)||s_\phi(Y|X)) - C\mathbb{E}_t[L(X,Y)], \tag{3.2}$$

where $t(Y)$ and $s_\phi(Y|X)$ are the prediction results of the teacher and student networks; C is a balancing term; ϕ is the parameter set of the student network; $L(X,Y)$ is a general constraint function that has high values to reward the predictions that meet the constraints and penalize the others. KL measures the KL-divergence of teacher's and student's prediction distributions. The closed-form solution of the optimization problem is:

$$t(Y) \propto s(Y|X)\exp(CL(X,Y)) \; . \tag{3.3}$$

The new objective which contains both ground truth labels and the teacher network is defined as:

$$\min_{\phi \in \Phi} \frac{1}{n} \sum_{i=1}^{n} \alpha l(s_i, y_i) + (1-\alpha) l(s_i, t_i), \tag{3.4}$$

where s_i and t_i are the student's and teacher's predictions for sample i; y_i is the ground truth label for sample i; α is a balancing term between ground truth and the teacher

network. l is the loss function. More details can be found in [32, 33].

3.4 Knowledge Distillation for Visual Relationship Detection

Linguistic knowledge is modeled by a conditional probability that encodes the strong correlation between the pair of objects $\langle subj, obj \rangle$ and the predicate that humans tend to use to describe the relationship between them:

$$L(X, Y) = \log P(pred|subj, obj), \tag{3.5}$$

where X is the input data and Y is the output distribution of the student network. $P(pred|subj, obj)$ is the conditional probability of a predicate given a fixed $\langle subj, obj \rangle$ pair in the obtained linguistic knowledge set.

By solving the optimization problem in Eq. 3.2, we construct a teacher network that is close to the student network, but penalizes a predicted predicate that is unlikely given the fixed $\langle subj, obj \rangle$ pairs. The teacher's output can be viewed as a projection of the student's output in the solution space constrained by linguistic knowledge. For example, when predicting the predicate between a "plate" and a "table", given the subject ("plate") and the object ("table"), and the conditional probability $P(pred|plate, table)$, the teacher will penalize unlikely predicates, (*e.g.*, "in") and reward likely ones (*e.g.*, "on"), helping the network avoid portions of the parameter space that lead to poor solutions.

Given the ground truth label and the teacher network's output distribution, we want the student network to not only predict the correct predicate labels but also mimic the linguistic knowledge regularized distributions. This is accomplished using a cross-entropy loss (see Eq. 3.4).

One advantage of this LK distillation framework is that it takes advantage of both knowledge-based and data-driven systems. Distillation works as a regularizer to help train the data-driven system. On the other hand, since we construct the teacher network based on the student network, the knowledge regularized predictions (teacher's output) will also be improved during training as the student's output improves. Rather than using linguistic knowledge as a post-processing step, our framework enables the data-driven model to absorb the linguistic knowledge together with the ground truth labels, allowing the deep network to learn a better visual model during training rather than only having its output modified in a post-processing step. This leads to a data-driven model (the student) that generalizes better, especially in the zero-shot scenario where we lack linguistic knowledge about a $\langle subj, obj \rangle$ pair. While [32–34] used either the student or the teacher as the final output, our experiments show that both the student and teacher in our framework have their own advantages, so we combine them to achieve the best predictive power (see section 3.7).

3.5 Linguistic Knowledge Collection

To obtain the linguistic knowledge $P(pred|subj, obj)$, a straightforward method is to count the statistics of the training annotations, which reflect the knowledge used by an annotator in choosing an appropriate predicate to describe a visual relationship. Due to the long tail distribution of relationships, a large number of combinations never occur in the training data; however, it is not reasonable to assume the probability of unseen relationships is 0. To tackle this problem, one can apply additive smoothing to assign a very

small number to all 0's [35]; however, the smoothed unseen conditional probabilities are uniform, which is still confusing at LK distillation time. To collect more useful linguistic knowledge of the long-tail unseen relationships, we exploit text data from the Internet.

One challenge of collecting linguistic knowledge online is that the probability of finding text data that specifically describes objects and their relationships is low. This requires us to obtain the knowledge from a huge corpus that covers a very large domain of knowledge. Thus we choose the Wikipedia 2014-06-16 dump containing around 4 billion words and 450 million sentences that have been parsed to text by [36] to extract knowledge.

We utilize the scene graph parser proposed in [37] to parse sentences into sets of $\langle subj, pred, obj \rangle$ triplets, and we compute the conditional probabilities of predicates based on these triplets. However, due to the possible mistakes of the parser, especially on text from a much wider domain than the visual relationship detection task, the linguistic knowledge obtained can be very noisy. Naive methods such as using only the linguistic knowledge to predict the predicates or multiplying the conditional probability with the data-driven model's output fail. Fortunately, since the teacher network of our LK-distillation framework is constructed from the student network that is also supervised by the labeled data, a well-trained student network can help correct the errors from the noisy external probability. To achieve good predictive power on the seen and unseen relationships, we obtain the linguistic knowledge from both training data and the Wikipedia text corpus by a weighted average of their conditional probabilities when we construct the teachers' network, as shown in Eq. 3.4. We conduct a two-step knowledge distillation: during the first several training epoches, we only allow the student to absorb the knowl-

edge from training annotations to first establish a good data-driven model. After that, we start distilling the external knowledge together with the knowledge extracted from training annotations weighted by the balancing term C as shown in Eq. 3.4. The balancing terms are chosen by a validation set we select randomly from the training set (*e.g.*, in VRD dataset, we select 1,000 out of 4,000 images to form the validation set) to achieve a balance between good generalization on the zero-shot and good predictive power on the entire testing set.

3.6 Semantic and Spatial Representations

In [1], Lu *et al.*used the cropped image containing the union of two objects' bounding boxes to predict the predicate describing their relationship. While the cropped image encodes the visual appearance of both objects, it is difficult to directly model the strong semantic and spatial correlations between predicates and objects, as both semantic and spatial information is buried within the pixel values of the image. Meanwhile, the semantic and spatial representations capture similarities between visual relationships, which can generalize better to unseen relationships.

We utilize word-embedding [38] to represent the semantic meaning of each object by a vector. We then extract spatial features similarly to the ones in [39]:

$$\left[\frac{x_{min}}{W}, \frac{y_{min}}{H}, \frac{x_{max}}{W}, \frac{y_{max}}{H}, \frac{A}{A_{img}}\right], \tag{3.6}$$

where W and H are the width and height of the image, A and A_{img} are the areas of the object and the image, respectively. We concatenate the above features of two objects as the spatial feature (SF) for a $\langle subj, obj\rangle$ pair.

We predict the predicate conditioned on the semantic and spatial representations of the subject and object:

$$P(R|I) = P(pred|subj, obj, B_s, B_o, I) \cdot P(subj, B_s|I)P(obj, B_o|I), \quad (3.7)$$

where $subj$ and obj are represented using the semantic object representation, B_s and B_o are the spatial features, and I is the image region of the union of the two bounding boxes. For the BB-Union input, we use the same VGG-net [40] in [1] to learn the visual feature representation. We adopt a pre-trained word2vec vectors weighted by confidence scores of each object category for the subject and the object, then concatenate the two vectors as the semantic representation of the subject and the object.

3.7 Experiments

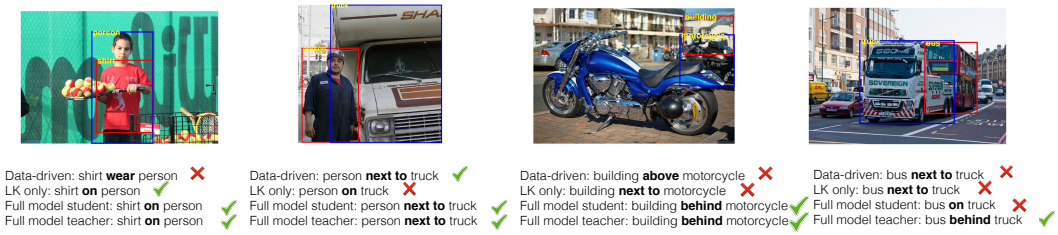
We evaluate our method on Visual Relationship Detection [1] and Visual Genome [2] datasets for three tasks: **Predicate detection**: given an input image and a set of ground truth bounding boxes with corresponding object categories, predict a set of predicates describing each pair of objects. This task evaluates the prediction of predicates without relying on object detection. **Phrase detection**: given an input image, output a phrase $\langle subj, pred, obj \rangle$ and localize the entire phrase as one bounding box. **Relationship detection**: given an input image, output a relationship $\langle subj, pred, obj \rangle$ and both the subject and the object with their bounding boxes.

Both datasets have a zero-shot testing set that contains relationships that never occur in the training data. We evaluate on the zero-shot sets to demonstrate the generalization

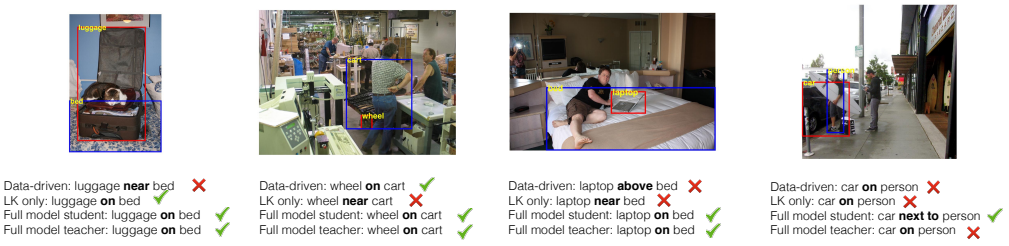
improvements brought by linguistic knowledge distillation.

Implementation Details. We use VGG-16 [40] to learn the visual representations of the BB-Union of two objects. We use a pre-trained word2vec [38] model to project the subjects and objects into vector space, and the final semantic representation is the weighted average based on the confidence scores of a detection. For the balancing terms, we choose $C = 1$ and $\alpha = 0.5$ to encourage the student network to mimic the teacher and the ground truth equally.

Evaluation Metric. We follow [1, 41] using Recall@n (R@n) as our evaluation metric (mAP metric would mistakenly penalize true positives because annotations are not exhaustive). For two detected objects, multiple predicates are predicted with different confidences. The standard R@n metric ranks all predictions for all object pairs in an image and compute the recall of top n. However, instead of computing recall based on all predictions, [1] considers only the predicate with highest confidence for each object pair. Such evaluation is more efficient and forced the diversity of object pairs. However, multiple predicates can correctly describe the same object pair and the annotator only chooses one as ground truth, *e.g.*, when describing a person "next to" another person, predicate "near" is also plausible. So we believe that a good predicted distribution should have high probabilities for all plausible predicate(s) and probabilities close to 0 for remaining ones. Evaluating only the top prediction per object pair may mistakenly penalize correct predictions since annotators have bias over several plausible predicates. So we treat the number of chosen predictions per object pair (k) as a hyper-parameter, and report R@n for different k's to compare with other methods [1,41,42]. Since the number of predicates is 70, $k = 70$ is equivalent to evaluating all predictions *w.r.t.*two detected objects.



(a) Seen relationships



(b) Zero-shot Relationships

Figure 3.2: Visualization of predicate detection results.

3.8 Evaluation on VRD Dataset

Predicate Prediction We first evaluate it on predicate prediction (as in [1]). Since [41, 44, 45] do not report results of predicate prediction, we compare our results with ones in [1, 42].

Part 1 of Table 3.1 shows the results of linguistic knowledge distillation with different sets of features in our deep neural networks. In addition to the data-driven baseline "Baseline: U only", we also compare with the baseline that only uses linguistic priors to predict a predicate, which is denoted as "Baseline: L only". The "Visual Phrases" method [42] trains deformable parts models for each relationship; "Joint CNN" [43] trains

[1]In predicate detection, R@100,k=1 and R@50,k=1 are equivalent (also observed in [1]) because there are not enough objects in ground truth to produce over 50 pairs.

[2]The recall of different k's are not reported in [1].We calculate those recall values using their code.

a 270-way CNN to predict the subject, object and predicate together. The visual only model and the full model of [1] that uses both visual input and language priors are denoted as "VRD-V only" and "VRD-Full". S denotes using the student network's output as the final prediction; T denotes using the teacher network's output. T+S denotes that for $\langle subj, obj \rangle$ pairs that occur in the training data, we use the teacher network's output as the final prediction; for $\langle subj, obj \rangle$ pairs that never occur in training, we use the student network's output.

End-to-end CNN training with semantic and spatial representations. Comparing our baseline, which uses the same visual representation (BB-Union) as [1], and the "VRD-V only" model, our huge recall improvement (R@100/50, k=1 increases from 7.11% [1] to 34.82%) reveals that the end-to-end training with soft-max prediction outperforms extracting features from a fixed CNN + linear model method in [1], highlighting the importance of fine-tuning. In addition, adding the semantic representation and the spatial features improves the predictive power and generalization of the data-driven model.

To demonstrate the effectiveness of LK-distillation, we compare the results of using different combinations of features with/without using LK-distillation. In Part 1 of Table 3.1, we train and test our model on only the VRD dataset, and use the training annotation as our linguistic knowledge. *"Linguistic knowledge only" baseline* ("Baseline: L only") itself has a strong predictive power and it outperforms the state-of-the-art method [1] by a large margin (*e.g.*, 51.34% vs. 47.87% for R@100/50, k=1 on the entire VRD test set), which implies the knowledge we distill in the data-driven model is reliable and discriminative. However, since, some $\langle subj, obj \rangle$ pairs in the zero-shot test set never occur in the linguistic knowledge extracted from the VRD train set, trusting only the

linguistic knowledge without looking at the images leads to very poor performance on the zero-shot set of VRD, which explains the poor generalization of "Baseline: L only" method and addresses the need for combining both data-driven and knowledge-based methods as the LK-distillation framework we propose does.

The benefit of LK distillation is visible across all feature settings: the data-driven neural networks that absorb linguistic knowledge ("student" with LK) outperform the data-driven models significantly (*e.g.*, R@100/50, k=1 is improved from 37.15% to 42.98% for "U+W" features on the entire VRD test set). We also observe consistent improvement of the recall on the zero-shot test set of data-driven models that absorb the linguistic knowledge. The student networks with LK-distillation yield the best generalization, and outperform the data-driven baselines and knowledge only baselines by a large margin.

Unlike [32–34], where either the student or the teacher is the final output, we achieve better predictive power by combining both: we use the teacher network to predict the predicates whose $\langle subj, obj \rangle$ pairs occur in the training data, and use the student network for the remaining. The setting "U+W+SF+LK: T+S" performs the best. Fig. 3.2(a) and 3.2(b) show a visualization of different methods.

Phrase and Relationship Detection To enable fully automatic phrase and relationship detection, we train a Fast R-CNN detector [46] using VGG-16 for object detection. Given the confidence scores of detected each detected object, we use the weighed word2vec vectors as the semantic object representation, and extract spatial features from each detected bounding box pairs. We then use the pipeline in Fig. 3.1 to obtain the predicted predicate distribution for each pair of objects. According to Eq. 3.7, we use the product of

the predicate distribution and the confidence scores of the subject and object as our final prediction results. We also adopt the triplet NMS in [44] to remove redundant detections. To compare with [1], we report R@n, k=1 for both phrase detection and relationship detection. For fair comparison with [41] (denoted as "Linguistic Cues"), we choose k=10 as they did to report recall. In addition, we report the full recall measurement k=70. Evaluation results on the entire dataset and the zero-shot setting are shown in Part 1 of Tables 3.2 and 3.3. Our method outperforms the state-of-the-art methods in [1] and [41] significantly on both entire testing set and zero-shot setting. The observations about student and teacher networks are consistent with predicate prediction evaluation. We also compare our method with the very recently introduced "VIP-CNN" in [44] and "VRL" [45] and achieve better or comparable results. For phrase detection, we achieve better results than [45] and get similar result for R@50 to [44]. One possible reason that [44] gets better result for R@100 is that they jointly model the object and predicate detection while we use an off-the-shelf detector. For relationship detection, we outperform both methods, especially on the zero-shot set.

3.9 Evaluation on Visual Genome Dataset

We also evaluate predicate detection on Visual Genome (VG) [2], the largest dataset that has visual relationship annotations. We randomly split the VG dataset into training (88,077 images) and testing set (20,000 images) and select the relationships whose predicates and objects occur in the VRD dataset. We conduct a similar evaluation on the dataset (99,864 relationship instances in training and 19,754 in testing; 2,056 relationship

test instances are never seen in training). We use the linguistic knowledge extracted from VG and report predicate prediction results in Table 3.4.

Not surprisingly, we observe similar behavior as on the VRD dataset—LK distillation regularizes the deep model and improves its generalization. We conduct another experiment in which images from Visual Genome dataset augment the training set of VRD and evaluate on the VRD test set. From the Part 2 of Tables 3.1, 3.2 and 3.3, we observe that training with more data leads to only marginal improvement over almost all baselines and proposed methods. However, for all experimental settings, our LK distillation framework still brings significant improvements, and the combination of the teacher and student networks still yields the best performance. This reveals that incorporating additional knowledge is more beneficial than collecting more data.

3.10 Distillation with External Knowledge

The above experiments show the benefits of extracting linguistic knowledge from internal training annotations and distilling them in a data-driven model. However, training annotations only represent a small portion of all possible relationships and do not necessarily represent the real world distribution, which has a long tail. For unseen long-tail relationships in the VRD dataset, we extract the linguistic knowledge from external sources: the Visual Genome annotations and Wikipedia, whose domain is much larger than any annotated dataset. In Table 3.5, we show predicate detection results on the VRD test set using our linguistic knowledge distillation framework with different sources of knowledge. From Part 2 and Part 4 of Table 3.5, we observe that using only the ex-

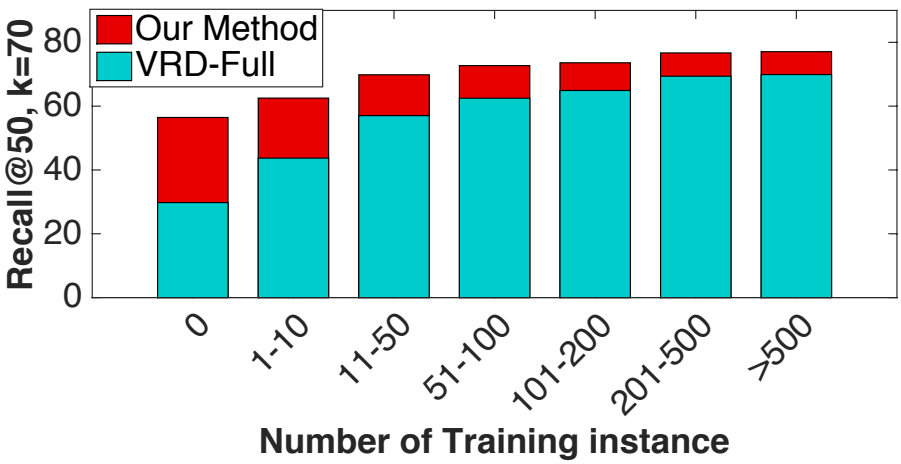


Figure 3.3: Performance with varying sizes of training examples.

ternal knowledge, especially the very noisy one obtained from Wikipedia, leads to bad performance. However, interestingly, although the external knowledge can be very noisy (Wikipedia) and has a different distribution when compared with the VRD dataset (Visual Genome), the performance of the teacher network using external knowledge is much better than using only the internal knowledge (Part 1). This suggests that by properly distilling external knowledge, our framework obtains both good predictive power on the seen relationships and better generalization on unseen ones. Evaluation results of combining both internal and external linguistic knowledge are shown in Part 3 and Part 5 of Table 3.5. We observe that by distilling external knowledge *and* the internal one, we improve generalization significantly (*e.g.*, LK from Wikipedia boosts the recall to 19.17% on the zero-shot set) while maintaining good predictive power on the entire test set.

Fig. 3.3 shows the comparison between our student network that absorbs linguistic knowledge from both VRD training annotations and the Wikipedia text (denoted as "Our Method") and the full model in [1] (denoted as "VRD-Full"). We observe that our method significantly outperforms the existing method, especially for the zero-shot

(relationships with 0 training instance) and the few-shot setting (relationships with few training instances, *e.g.*, ≤ 10). By distilling linguistic knowledge into a deep model, our data-driven model improves dramatically, which is hard to achieve by only training on limited labeled images.

3.11 Conclusion

We proposed a framework that distills linguistic knowledge into a deep neural network for visual relationship detection. We incorporated rich representations of a visual relationship in our deep model, and utilized a teacher-student distillation framework to help the data-driven model absorb internal (training annotations) and external (public text on the Internet) linguistic knowledge. Experiments on the VRD and the Visual Genome datasets show significant improvements in accuracy and generalization capability.

	Entire Set			Zero-shot		
	R@100/50[1] k=1	R@100 k=70	R@50 k=70	R@100/50 k=1	R@100 k=70	R@50 k=70
Part 1: Training images VRD only						
Visual Phrases [42]	1.91	-	-	-	-	-
Joint CNN [43]	2.03	-	-	-	-	-
VRD-V only [1]	7.11	37.20 [2]	28.36	3.52	32.34	23.95
VRD-Full [1]	47.87	84.34	70.97	8.45	50.04	29.77
Baseline: U only	34.82	83.15	70.02	12.75	69.42	47.84
Baseline: L only	51.34	85.34	80.64	3.68	18.22	8.13
U+W	37.15	83.78	70.75	13.44	69.77	49.01
U+W+L:S	42.98	84.94	71.83	13.89	72.53	51.37
U+W+L:T	52.96	88.98	83.26	7.81	40.15	32.62
U+SF	36.33	83.68	69.87	14.33	69.01	48.32
U+SF+L:S	41.06	84.81	71.27	15.14	72.72	51.62
U+SF+L:T	51.67	87.71	83.84	8.05	41.51	32.77
U+W+SF	41.33	84.89	72.29	14.13	69.41	48.13
U+W+SF+L: S	47.50	86.97	74.98	**16.98**	**74.65**	**54.20**
U+W+SF+L: T	54.13	89.41	82.54	8.80	41.53	32.81
U+W+SF+L: T+S	**55.16**	**94.65**	**85.64**	-	-	-
Part 2: Training images VRD + VG						
Baseline: U	36.97	84.49	70.19	13.31	70.56	50.34
U+W+SF	42.08	85.89	72.83	14.51	70.79	50.64
U+W+SF+L: S	48.61	87.15	75.45	17.16	75.26	55.41
U+W+SF+L: T	54.61	90.09	82.97	9.23	43.21	33.40
U+W+SF+L: T+S	55.67	95.19	86.14	-	-	-

Table 3.1: Predicate Detection on VRD Testing Set. Part 1 uses the VRD training images; Part 2 uses the training images in VRD [1] and images of Visual Genome (VG) [2] dataset.

	Phrase Detection						Relationship Detection					
	R@100, k=1	R@50, k=1	R@100, k=10	R@50, k=10	R@100, k=70	R@50, k=70	R@100, k=1	R@50, k=1	R@100, k=10	R@50, k=10	R@100, k=70	R@50, k=70
Part 1: Training images VRD only												
Visual Phrases [42]	0.07	0.04	-	-	-	-	-	-	-	-	-	
Joint CNN [43]	0.09	0.07	-	-	-	-	0.09	0.07	-	-	-	-
VRD - V only [1]	2.61	2.24	-	-	-	-	1.85	1.58	-	-	-	-
VRD - Full [1]	17.03	16.17	25.52	20.42	24.90	20.04	14.70	13.86	22.03	17.43	21.51	17.35
Linguistic Cues [41]	-	-	20.70	16.89	–	–	–	–	18.37	15.08	–	–
VIP-CNN [44]	**27.91**	22.78	-	-	–	–	20.01	17.32	-	-	–	–
VRL [45]	22.60	21.37	-	-	–	–	20.79	18.19	-	-	–	–
U+W+SF+L: S	19.98	19.15	25.16	22.95	25.54	22.59	17.69	16.57	27.98	19.92	28.94	20.12
U+W+SF+L: T	23.57	22.46	29.14	25.96	29.09	25.86	20.61	18.56	29.41	21.92	31.13	21.98
U+W+SF+L: T+S	24.03	**23.14**	**29.76**	**26.47**	**29.43**	**26.32**	**21.34**	**19.17**	**29.89**	**22.56**	**31.89**	**22.68**
Part 2: Training images VRD + VG												
U+W+SF+L: S	20.32	19.96	25.71	23.34	25.97	22.83	18.32	16.98	28.24	20.15	29.85	21.88
U+W+SF+L: T	23.89	22.92	29.82	26.34	29.97	26.15	20.94	18.93	29.95	22.62	31.78	22.65
U+W+SF+L: T+S	24.42	23.51	30.13	26.73	30.01	26.58	21.72	19.68	30.45	22.84	32.56	23.18

Table 3.2: Phrase and Relationship Detection: Distillation of Linguistic Knowledge. We use the same notations as in Table 3.1.

	Phrase Detection						Relationship Detection					
	R@100, k=1	R@50, k=1	R@100, k=10	R@50, k=10	R@100, k=70	R@50, k=70	R@100, k=1	R@50, k=1	R@100, k=10	R@50, k=10	R@100, k=70	R@50, k=70
Part 1: Training images VRD only												
VRD - V only [1]	1.12	0.95	-	-	-	-	0.78	0.67	-	-	-	-
VRD - Full [1]	3.75	3.36	12.57	7.56	12.92	7.96	3.52	3.13	11.46	7.01	11.70	7.13
Linguistic Cues [41]	-	-	15.23	10.86	-	-	-	-	13.43	9.67	-	-
VRL [45]	10.31	9.17	-	-	–	–	8.52	7.94	-	-	–	–
U+W+SF+L: S	**10.89**	**10.44**	**17.24**	**13.01**	**17.24**	**12.96**	**9.14**	**8.89**	**16.15**	**12.31**	**15.89**	**12.02**
U+W+SF+L: T	6.71	6.54	11.27	9.45	9.84	7.86	6.44	6.07	9.71	7.82	10.21	8.75
Part 2: Training images VRD + VG												
U+W+SF+L: S	11.23	10.87	17.89	13.53	17.88	13.41	9.75	9.41	16.81	12.72	16.37	12.29
U+W+SF+L: T	7.03	6.94	11.85	9.88	10.12	8.97	6.89	6.56	10.34	8.23	10.53	9.03

Table 3.3: Phrase and Relationship Detection: Distillation of Linguistic Knowledge - Zero Shot. We use the same notations as in Table 3.1.

	Entire Set			Zero-shot		
	R@100/50 k=1	R@100 k=70	R@50 k=70	R@100/50 k=1	R@100 k=70	R@50 k=70
U	37.81	82.05	81.41	7.54	81.00	65.22
U+W+SF	40.92	86.81	84.92	8.66	82.50	67.72
U+W+SF+L: S	49.88	91.25	88.14	**11.28**	**88.23**	**72.96**
U+W+SF+L: T	55.02	94.92	91.47	3.94	62.99	47.62
U+W+SF+L: T+S	**55.89**	**95.68**	**92.31**	-	-	-

Table 3.4: Predicate Detection on Visual Genome Dataset. Notations are the same as in Table 3.1.

	Entire Set			Zero-shot		
	R@100/50 k=1	R@100 k=70	R@50 k=70	R@100/50 k=1	R@100 k=70	R@50 k=70
Part 1 LK: VRD						
VRD-V only [1]	7.11	37.20	28.36	3.52	32.34	23.95
VRD-Full [1]	47.87	84.34	70.97	8.45	50.04	29.77
U+W+SF+L: S	47.50	86.97	74.98	16.98	74.65	54.20
U+W+SF+L: T	54.13	89.41	82.54	8.80	41.53	32.81
Part 2 LK: VG						
U+W+SF+L: S	45.00	81.64	74.76	16.88	72.29	52.51
U+W+SF+L: T	51.54	87.00	79.70	11.01	54.66	45.25
Part 3 LK: VRD+VG						
U+W+SF+L: S	48.21	87.76	76.51	17.21	74.89	54.65
U+W+SF+L: T	**54.82**	**90.63**	**83.97**	12.32	47.22	38.24
Part 4 LK: Wiki						
U+W+SF+L: S	36.05	77.88	68.16	11.80	64.24	49.19
U+W+SF+L: T	30.41	69.86	60.25	11.12	63.58	44.65
Part 5 LK: VRD+Wiki						
U+W+SF+L: S	48.94	87.11	77.79	**19.17**	**76.42**	**56.81**
U+W+SF+L: T	54.06	88.93	81.78	9.65	42.24	34.61

Table 3.5: Predicate Detection on VRD Testing Set: External Linguistic Knowledge.

Chapter 4: The Role of Context Selection in Object Detection

4.1 Introduction

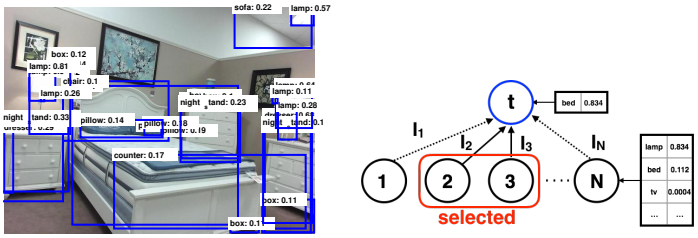
Context captures statistical and common sense properties of the real-world and plays a critical role in perceptual inference [47]. There are numerous studies that demonstrate the advantages of context in object recognition [47–51]. In contrast, other investigations have revealed situations in which context does not improve the performance of object detection [52, 53], and sometimes introducing context even decreases performance [53]. Additionally, driven by the development of deep CNNs [54, 55], the performance of object detection has been dramatically boosted [56–59]. While context has been incorporated into deep learning frameworks, the performance gain from context itself has not been significant [60,61]. This leads to the question: how important is context in object detection when we have reasonably good detectors?

To address this question, we study possible reasons why context might not improve detection. First, imperfect extraction of context information introduces errors into contextual inference. For instance, when visual context information is extracted through imperfect appearance-based detectors, as shown in Figure 4.1(a), incorrectly-detected regions can introduce noise into contextual inference, limiting the gain from context provided by correctly detected regions. Second, contextual information that is hard to extract or has

low predictive power can introduce errors into context that is easy to detect and is very predictive. For example, when predicting the presence of a pillow in an image using context provided by other objects of different categories, some object categories, such as beds and sofas, which are easier to detect and have strong relationships with respect to pillow, are informative as context. Others, such as boxes and pictures, which are either hard to detect or irrelevant to pillows, are likely to be useless or even misleading.

To further investigate these challenges, we conduct a simulation to study the role of context in isolation, without appearance-based clues. Since reliable contextual relationships between object pairs can be most reliably learned in sufficiently structured scenes, we utilize the SUN RGB-D [62] dataset, which is one of the largest indoor-scene datasets and contains a large number of annotated objects. For a given image with ground truth bounding boxes for all objects, we predict the label for each object, one at a time. The object whose label is to be predicted is referred to as the *target object* and the other objects are referred to as *contextual objects*. For the unknown target object, we remove the uncertainty of all remaining objects by assigning them to their ground-truth labels and use object-to-object contextual relationships to predict the label of the target object without access to its appearance. We observe very good prediction accuracy, which implies that, without detection noise, simple contextual relationships between objects can boost detection performance. We then study how predictive each object class is of a given target object class by ignoring one contextual object class at a time. The results suggest that different object classes vary in their ability to predict the presence of certain target object categories.

Motivated by these experiments, we propose a region-based context re-scoring



(a) Detection Results (b) Context Selection

Figure 4.1: Context Selection with Noisy Detection.

method with dynamic context selection, illustrated in figure 4.1(b), which seeks to eliminate false positive contextual regions while emphasizing likely true positive and informative ones. Specifically, we introduce a latent variable for each contextual region that determines if that region will be selected to provide context information. In practice, it is intractable to select the optimal set of contextual regions that provide the most trustworthy information when contradictory evidence exists, both *for* and *against* the target object having a certain class label. Instead, we decompose the problem by selecting informative regions that provide the strongest supporting and refuting evidence independently to compute a *For upper-bound* and an *Against upper-bound* of the confidence score, and then re-score the confidence for that object being in that class based on the difference between the two upper-bounds. The model for computing the two upper-bounds is trained by latent-SVM [63]. The proposed method is evaluated on the SUN RGB-D dataset and achieves 48.25% mean average precision (mAP), an improvement of $\sim 2.8\%$ over using object detections without context (45.47%). We also conduct experiments to study the performance of the selection model. Both the simulations on pure context and the real-world experiments using the proposed selection method demonstrate the importance of object-to-object context and the gain attributed to the context selection scheme.

4.2 The Role of Pure Context

We first conduct an experiment to analyze the utility of pure contextual relationships between objects in object detection. In this experiment, we only consider the ground-truth bounding boxes, and the label for a given box is predicted using only context information between the target box and the remaining boxes for other objects in an image. When predicting the label of the target object, we only consider its bounding box and intentionally ignore appearance information such as color, shape and texture. Moreover, the ground-truth labels and bounding boxes of all contextual objects are revealed to remove the influence of uncertainty in context. We consider three types of object-to-object context: co-occurrence, relative scale and spatial relationships.

Predicting Object Class using Pure Contextual Relationships Prediction is performed by a linear classifier. Given an image I, assume there are $N+1$ ground-truth objects, drawn from M object categories. When predicting the label of a target object t, the ground-truth bounding boxes of t and the remaining N objects are given, along with the the labels for all objects other than t. The confidence that object t is in class T is:

$$Score(o_{tT}) = \sum_{j=1}^{N}\sum_{i=1}^{M}[Co(o_{tT}, o_{ji}; \mathbf{w}_{co}) + Sc(o_{tT}, o_{ji}; \mathbf{w}_{sc}) + Sp(o_{tT}, o_{ji}; \mathbf{w}_{sp})] \cdot l_{ji} + b, \tag{4.1}$$

where o_{tT} indicates that the target object t is assigned label T, and o_{ji} indicates that the j^{th} contextual object is in class i, l_{ji} is a binary indicator variable with $l_{ji} = \mathbf{1}\{label_j = i\}$, and b is a bias term. The terms $Co(\cdot)$, $Sc(\cdot)$ and $Sp(\cdot)$ measure co-occurrence, relative

scale and spatial relationship defined in equation (4.2):

$$
\begin{aligned}
Co(o_{tT}, o_{ji}; \mathbf{w}_{co}) &= w_{co}(T, i) \cdot \log d_{co}(o_{tT}, o_{ji}) \\
Sc(o_{tT}, o_{ji}; \mathbf{w}_{co}) &= w_{sc}(T, i) \cdot \log d_{sc}(o_{tT}, o_{ji}, r) \\
Sp(o_{tT}, o_{ji}; \mathbf{w}_{sp}) &= w_{spx}(T, i) \cdot \log d_{spx}(o_{tT}, o_{ji}, x) + w_{spy}(T, i) \cdot \log d_{spy}(o_{tT}, o_{ji}, y),
\end{aligned}
\tag{4.2}
$$

where $\mathbf{w}_{co}$, $\mathbf{w}_{sc}$, $\mathbf{w}_{sp}$ are weight vectors for each set of contextual features respectively. The term $d_{co}(o_{tT}, o_{ji})$ is the likelihood that a target object t of category T and object j of category i co-occur in the same image. The term $d_{sc}(o_{tT}, o_{ji}, r)$ is the likelihood that a target object t of category T and object j of category i have relative scale ratio r. The terms $d_{spx}(o_{tT}, o_{ji}, x)$ and $d_{spy}(o_{tT}, o_{ji}, y)$ are the likelihoods that a target object t of category T and object j of category i have relative spatial distance (distance along one axis normalized by the height/width of the image) x along X-axis, and y along Y-axis, respectively. The likelihoods are learned from the statistics of the training set. For the co-occurrence context information, we use a two-bin histogram to represent the likelihood for the co-occurrence of a target-context object pair. For the relative scale and the spatial context, we categorize the relative scale ratios and relative distances into n slots and use an n-bin histogram to represent the likelihoods.

Given this linear model and the features extracted based on the likelihoods, we train a multi-class classifier using structural SVM [64]. To evaluate the performance of object detection using pure context, we compute prediction accuracies on the 19 common objects used in the SUN RGB-D dataset; the same object categories are used as contextual objects. The average accuracy is 70.68%, which is quite high considering that no appearance information is utilized.

The Role of Different Contextual Object Categories The above experiment shows the predictive potential of pure context. Do all object categories provide equally informative context when predicting the label of a target object, or are some of them more informative than others? To answer this question, we evaluate the predictive power of each object category with respect to a given target object category. For each target object category T, we measure the relative accuracy loss (RAL), defined as $RAL(T, C, i) = \frac{Accuracy_{C-i}(T) - Accuracy_C(T)}{Accuracy_C(T)}$, when removing the i^{th} object category from the set C of contextual object categories. Some RAL examples are shown in Table 4.1 and 4.2. For each target object category, we show the object categories that lead to the top five largest RALs. We observed that different object categories have significantly different predictive power depending on certain object categories.

Table 4.1: Relative Accuracy Loss (RAL): T = Pillow

	pillow	bed	sofa	lamp	night-stand
RAL	0.28	0.24	0.17	0.08	0.04

Table 4.2: Relative Accuracy Loss (RAL): T = Bookshelf

	book-shelf	chair	desk	table	box
RAL	0.35	0.27	0.17	0.11	0.03

In summary, pure contextual information between object pairs has high predictive power, but each contextual-object category, not surprisingly, predicts some target categories much better than others.

4.3 Region-based Context Re-scoring with Dynamic Context Selection

Based upon the above analysis, we propose a model to improve detection based on context, where contextual objects are detected automatically and are thus noisy prob-

abilistic detections. We utilize the appearance clues from state-of-the-art detectors for predicting a target object's label. The same contextual relationships discussed in the previous analysis between a detected target bounding box and the remaining ones are utilized, but each box is a region with an $(M+1)$ associated probability distribution over the possible labels including the background. We introduce binary latent variables for all contextual regions, indicating whether a contextual region is selected in the context re-scoring process.

Test-Time Re-scoring using For-and-Against Upper Bounds We propose a region-based context re-scoring model with context selection. The re-scored confidence for the target object t being in category T is:

$$\begin{aligned} Score(o_{tT}) =& w_0 \log A(o_{tT}) + \sum_{j=1}^{N}\sum_{i=1}^{M}[Co(o_{tT}, o_{ji}; \mathbf{w}_{co}) + Sc(o_{tT}, o_{ji}; \mathbf{w}_{sc}) + Sp(o_{tT}, o_{ji}; \mathbf{w}_{sp}) \\ &+ w_{A_c}(i) \log A_c(o_{ji})] \cdot l_j + b, \end{aligned} \tag{4.3}$$

where $A(o_{tT})$ and $A_c(o_{ji})$ represent the appearance-based confidence scores of the target and the contextual objects, w_0 and $w_{A_c}(i)$ are the corresponding weights, and l_j is the binary indicator variable for context selection. The proposed method can be viewed as a tree model where the target object (the root) collects context information from the selected contextual objects (the leaves). In contrast to traditional graphical models, the proposed method is an approximation that decomposes the re-scoring process into two independent ones due to the intractability of jointly solving the context selection with contradictory context information.

Our model, intuitively, corresponds to a courtroom where the prosecutors tries to prove that the target object does not come from a given class by providing the most compelling negative evidence (a small set of confidently detected context objects whose presence is inconsistent with that label for the target object), while the defendant's lawyer provides the most compelling evidence for the truth of the claim that the target object is from the given class. Our goal is to learn, from training data, how these "arguments" should be constructed for a given target class. That is, we seek to learn a computational model for a multi-valued logic [65] in an attempt to avoid noise in detection and ambivalence in contextual prediction (from the large number of incorrect and irrelevant potential objects in an image) from overwhelming the clear and compelling evidence concerning the identity of the target object. This type of multi-valued logic has been used before for human detection (where reasoning considered occlusion and image border effects [66]), but actually learning how to choose these evidential arguments from training data has not been done before. So, our solution involves identifying the different sources that provide supporting and refuting evidence independently, and then to combine the *degree of belief for* and the *degree of belief against* to obtain the final confidence of a target object being in class T. Specifically, we first re-score each target object t by selecting the evidence that most strongly supports it being in a certain class to compute its *For-Score*, and select the evidence that most strongly argues against it for its *Against-Score*. Both the *For-Score* and the *Against-Score* can be computed by maximizing function (4.3) over all possible indicator vectors that consist of indicator variables for all contextual regions, but with different weight vectors. The weight vector for computing the *For-Score* is learned with positive samples that are in class T, while the weight vector for

the *Against-Score* considers the objects that are not in class T as the positive samples. The *For-Score* can be viewed as a belief upper bound for a target object being in class T. In both cases we select contextual regions with high appearance-based confidence scores by forcing the weight $w_{A_c}(i)$ to be positive. The final degree of belief for the target object t being in class T is the margin between the *For-and-Against upper bounds*: $Score(o_{tT}) = Score_{For}(o_{tT}) - Score_{Against}(o_{tT})$.

Training with Latent-SVM The proposed re-scoring model with dynamic context selection can be trained using latent-SVM [63]. The processes for learning the weights for computing the *For-Score* and the *Against-Score* are the same except for the choice of positive samples. We describe the general training process. The weight vector and feature vector for an sample x are shown in equation 4.4 and 4.5.

$$\mathbf{w} = \{w_0, \mathbf{w}_{co}, \mathbf{w}_{sc}, \mathbf{w}_{sp}, \mathbf{w}_{A_c}, b\}\,, \tag{4.4}$$

$$\begin{aligned}\phi(x, \mathbf{l}) = \{&\log A(o_{tT}), \sum_{j=1}^{N} \log d_{co}(o_{tT}, o_{j1}) \cdot l_j, \\ &\cdots, \sum_{j=1}^{N} \log d_{co}(o_{tT}, o_{jM}) \cdot l_j, \sum_{j=1}^{N} \log d_{sc}(o_{tT}, o_{j1}) \cdot l_j, \\ &\cdots, \sum_{j=1}^{N} \log d_{sc}(o_{tT}, o_{jM}) \cdot l_j, \sum_{j=1}^{N} \log d_{sp}(o_{tT}, o_{j1}) \cdot l_j, \\ &\cdots, \sum_{j=1}^{N} \log d_{sp}(o_{tT}, o_{jM}) \cdot l_j, \sum_{j=1}^{N} \log A_c(o_{j1}) \cdot l_j \cdots, \sum_{j=1}^{N} \log A_c(o_{jM}) \cdot l_j, 1\},\end{aligned} \tag{4.5}$$

where l is the vector consists of indicator variables.

The re-scored confidence for a sample x is determined by a classifier using a func-

tion of form:

$$f_{\mathbf{w}}(x) = \max_{\mathbf{l} \in \mathbf{L}(x)} \mathbf{w} \cdot \phi(x, \mathbf{l}), \tag{4.6}$$

where $\mathbf{L}(x)$ consists of all possible latent vectors for sample x. The objective function to be minimized is:

$$loss(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f_{\mathbf{w}}(x_i)) + \frac{\lambda}{2} \|\mathbf{w}\|^2, \tag{4.7}$$

where we adopt the hinge loss to minimize the loss in a max-margin manner. The constant λ is used to weight the regularization term.

Although a latent-SVM leads to a non-convex optimization, we can efficiently solve it using coordinate descent by leveraging its semi-convexity property. The coordinate descent method involves two steps. Firstly, positive samples are relabeled by selecting contextual-regions that scores the target object highest by solving a linear programming problem:

$$\mathbf{l} = \text{argmax}_{\mathbf{l} \in \mathbf{L}(x)} \mathbf{w} \cdot \phi(x, \mathbf{l}), \tag{4.8}$$

and then, the weight vector $\mathbf{w}$ is optimized by solving a convex optimization problem by minimizing the loss in equation (5.1) given the relabeled positive samples.

4.4 Experiments

Dataset We use the SUN RGB-D dataset, which contains images from [67–69], to evaluate our proposed method. We consider the 19 most common classes in the dataset. The performance is evaluated through the average precision (AP) of object detection. For com-

parison, we evaluated several R-CNN based detectors, and chose as the baseline one that utilizes the depth modality by supervision transfer (ST) [59], which uses object proposals from [70] and yields state-of-the-art mAP on the SUN RGB-D dataset for 2D object detection.

4.5 Context Selection Model and Baseline Models

Besides ST, we also compare our context selection model (CS) with the baselines including the one that "selects all" (SA) contextual regions and the one that only considers either the *For upper-bound* (FUB) or the *Against upper-bound* (AUB). For each object category T, we train the model based upon appearance-based detections using latent-SVM. When predicting a target object's label, we set a precision threshold (and choose corresponding appearance-based confidence score thresholds of all contextual objects), and only consider detections with scores higher than the thresholds as potential contextual objects to ensure the context precision for the potential contextual objects of each class reaches the precision threshold. To train the FUB model, we label boxes that have ground-truth labels in class T as positive samples and select from supporting evidence to obtain the *For upper-bound*. The training process for the AUB model is obtained by simply reversing the positive and negative training labels. During the test, for each target object t, the context selection model selects supporting refuting evidence separately to compute the FUB and the AUB, and then uses the margin between them as the final confidence score for object t being in class T.

Does context selection work? We compare the context selection model with the

select-all model by measuring average precision. Table 4.3 shows the average precision of the 19 object classes when the precision threshold for contextual regions is set as 0.4. We achieve only a 0.43% mAP gain by the SA model, with some classes improving notably (counter, desk, lamp and pillow), while others (bathtub, chair, monitor, dresser, sofa, sink and toilet) degrading when considering all contextual regions. When we apply context selection, we see large improvement in AP on almost all classes compared to both the ST and SA baselines. We observe $\sim 2.8\%$ mAP gain by the context selection method.

	bathtub	bed	book-shelf	box	chair	counter	desk	door	dresser	garbage bin	lamp	monitor	night stand	pillow	sink	sofa	table	tv	toilet	mAP
ST [59]	67.70	76.44	43.45	18.02	**42.15**	32.06	24.94	22.93	40.27	52.83	49.73	47.80	56.30	48.75	19.92	50.82	42.83	45.66	81.35	45.47
SA	59.61	77.62	42.12	19.46	40.01	**35.80**	**33.86**	23.75	39.72	50.81	51.19	43.87	**61.44**	51.20	17.61	49.37	45.86	48.39	80.37	45.90
FUB	67.75	78.89	45.60	22.11	39.19	34.82	29.54	23.92	**41.19**	52.51	53.22	48.76	59.61	53.06	23.19	50.72	43.74	50.68	81.53	47.37
AUB	67.54	78.77	45.64	20.42	40.21	35.06	26.80	23.75	41.04	52.97	52.84	48.93	58.62	51.69	23.32	**52.07**	46.65	45.59	81.51	47.02
CS	**69.15**	**80.09**	**45.94**	**22.33**	42.04	35.57	29.84	**24.44**	40.85	**53.51**	**53.67**	**48.96**	60.18	**54.19**	**24.60**	48.99	**48.86**	**51.06**	**82.50**	**48.25**

Table 4.3: Average Precision (AP) on SUN RGB-D Test Set.

How does context selection compare to simple threshold-based filtering? One may wonder whether the selection method can be modeled by a simple threshold-based selection on contextual regions based on their appearance-based confidence scores. We conducted an experiment to compare the performance of the proposed context selection method with that of the select-all method augmented with the choice of different precision thresholds for contextual regions. The results are shown in Figure 4.2. We observe that with the increase of precision threshold for contextual regions, the select-all method does perform better due to reduced noise from contextual regions. The context selection method also consistently improves as the precision threshold raises is raised from 0.1 to 0.5. Performance drops when the precision threshold exceeds 0.5. At this point, too few

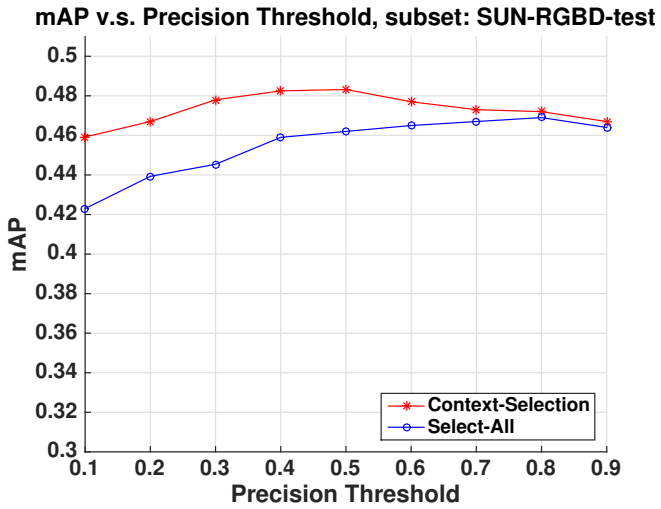


Figure 4.2: mAP v.s. Precision Threshold.

relevant regions survive the precision threshold. Generally, the context selection method outperforms the select-all method with precision-threshold-based filtering.

Does the margin between *For-and-Against upper-bounds* help? Performance of context selection methods that use only one of the *For upper-bound*, the *Against upper-bound* and the margin between them are shown in Table 4.3. By ignoring the against evidence in the FUB method, the confidence scores of true positives increase as expected, but the false positives are also boosted higher. As we subtract the AUB from the FUB, we introduce refuting evidence to balance the boosting effect, and reduce false positives. We observe a performance gain of about $\sim 1.0\%$ by combining the two upper-bounds together.

Does the selection model do more than select true positive contextual regions? Section 4.2 illustrated the differential predictive power of contextual objects for a certain target object. Ideally the context selection method should select contextual objects that exist in the image and also have strong predictive power. To test that this is indeed occur-

ring, we compare to the setting where an oracle labels the true positive contextual objects for the model to choose from. We show the APs for the SA and the CS methods with oracle in both training and testing phases in Table 4.4 and label them as SA-O and CS-O, respectively. For each target object class and a given contextual object class i, we show the ratio between the counts of selected true positive contextual objects in class i and the total number of contextual objects in that class as the selecting ratio. The top five contextual objects that have the highest selecting ratios of two target object classes are shown in Table 4.5 and 4.6. We notice that the selecting ratios vary for different contextual object categories, and by selecting a subset of informative contextual objects, the CS-O method outperforms the SA-O method. The performance of CS-O is the upper-bound of the context selection model, and the proposed selection model is a good approximation to the upper-bound.

Visualization of Selected contextual regions To visualize the performance of the context selection method, we show the selected contextual regions for four target object classes in Figure 4.3. The selection model tends to gather context information from the true positive contextual regions that can provide strong supporting or refuting evidence to predict the label of the target object.

	bathtub	bed	book-shelf	box	chair	counter	desk	door	dresser	garbage bin	lamp	monitor	night stand	pillow	sink	sofa	table	tv	toilet	mAP
CS	69.15	80.09	45.94	22.33	42.04	35.57	29.84	24.44	40.85	53.51	53.67	48.96	60.18	54.19	24.60	48.99	48.86	**51.06**	82.50	48.25
SA-O	70.07	78.95	45.48	**23.29**	42.91	36.97	**33.59**	24.37	**42.39**	52.44	52.78	49.15	**61.81**	53.38	**26.49**	**53.44**	48.71	49.16	81.73	48.79
CS-O	**73.67**	**80.56**	**50.42**	23.08	**47.51**	**37.33**	30.98	**24.98**	41.03	**54.45**	**56.49**	**50.08**	60.25	**55.29**	25.30	49.09	**49.27**	45.91	**83.55**	**49.43**

Table 4.4: Average Precision (AP) on SUN RGB-D Test Set: with Oracle.

Table 4.5: Selecting Ratio: T = Pillow

	bed	sofa	pillow	night-stand	lamp
Ratio	0.92	0.89	0.84	0.79	0.74

Table 4.6: Selecting Ratio: T = Bookshelf

	desk	table	book-shelf	chair	sofa
Ratio	0.97	0.92	0.81	0.77	0.76

Figure 4.3: Visualization of For-Against Context Selection.

4.6 Conclusion

We analyzed the predictive potential of context in an idealized case where the labels of all contextual objects are known, and only these labels and their relationships to the target objects are used to predict the target object label. Through these experiments we found that, despite ignoring the appearance of the target object, pure context is effective at predicting the target object. We also discovered that different categories vary in their ability to predict a certain target object class. Based on these experiments, we proposed a

region-based context re-scoring method with dynamic context selection to automatically improve the pool of contextual objects. Our method achieved significant performance gains when compared with the appearance-based detector and the contextual model that simply selects everything. An interesting direction of the future work is to use depth information as a contextual cue, and apply context selection in an end-to-end deep learning framework.

Chapter 5: ReMotENet: Efficient Relevant Motion Event Detection for Large-scale Home Surveillance Videos

5.1 Introduction

With the development of home security and surveillance system, more and more home surveillance cameras have been installed to monitor customers' home 24/7 for security and safety purpose. Most existing commercial solutions run motion detection on the edge (camera), and show the detected motion events (usually in shot clips of, e.g., 15s) for end users' review on the web or mobile.

Motion detection has been a challenging problem in spite of many years' development in academia and industry [71, 72]. Many nuisance alarm sources, such as tree motion, shadow, reflections, rain/snow, flags, result in many irrelevant motion events for customers.

Relevant motion detection is responsive to customers' needs. It involves pre-specified relevant objects, e.g., people, vehicles and pets, and these objects should have human recognizable location changes in the video. It not only helps to remove nuisance events, but also supports applications such as semantic video search and video summarization.

As shown in Figure 5.1, one natural method is to apply state-of-the-art object de-

tectors based on deep convolutional neural networks (CNNs) [43, 46, 73–76] to identify objects of interest. Given a video clip, background subtraction is applied to each frame to filter out stationary frames. Object detection is then applied to frames that have motion to identify the moving objects. Finally, the system generates trackers on the detection results to filter out temporally inconsistent falsely detections or stationary ones.

There are at least two problems with this object detection based method. First, it is computationally expensive, especially the object detection module. The state-of-the-art object detectors [43, 46, 73–76] need to be run on expensive GPUs devices and achieve at most 40-60 FPS [77]. Scaling to tens of thousands of motion events coming from millions of cameras becomes cost ineffective. Second, the method usually consists of several separate pre-trained methods or hand-crafted rules, and does not fully utilize the spatial-temporal information of an entire video clip. For example, moving object categories are detected mainly by object detection, which ignores motion patterns that can also be utilized to classify the categories of moving objects.

To address these problems, we propose a network for relevant motion event detection, ReMotENet, which is a unified, end-to-end data-driven method using Spatial-temporal Attention-based 3D ConvNets to jointly model the appearance and motion of objects-of-interest in a video event. As shown in Figure 5.1, ReMotENet parses an entire video clip in one forward pass of a neural network to achieve significant speedup (up to 20k$\times$) on a single GPU. This makes the system easily scalable to millions of motion events and reduces latency. Meanwhile, it exploits the properties of home surveillance videos, e.g., relevant motion is sparse both spatially and temporally, and enhances 3D ConvNets with a spatial-temporal attention model and frame differencing to encourage

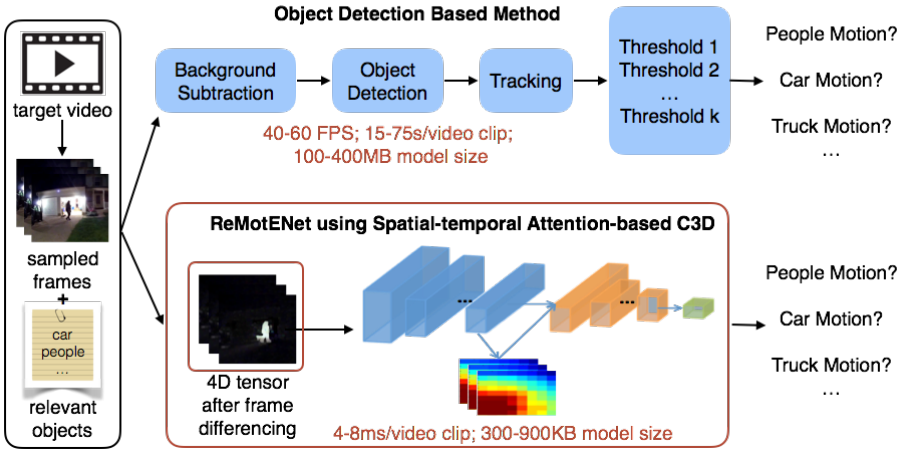


Figure 5.1: Comparison between the traditional object detection based method and ReMotENet.

the network to focus on relevant moving objects.

To train and evaluate our model, we collected a large dataset of 38,360 real home surveillance video clips of 15s from 78 cameras covering various scenes, indoor/outdoor, day/night, different lighting conditions and weather. To avoid the cost of training annotations, our method is weakly supervised by the results of the object detection based method. For evaluation, we manually annotated 9,628 video clips with binary labels of relevant motion caused by different objects. Experiments demonstrate that ReMotENet achieves comparable or even better performance, but is three to four orders of magnitude faster than the object detection based method. Our network is efficient, compact and light-weight. It can precisely detect relevant motion in a 15s video in 4-8 milliseconds on a GPU and a fraction of second on a CPU with model size of less than 1MB.

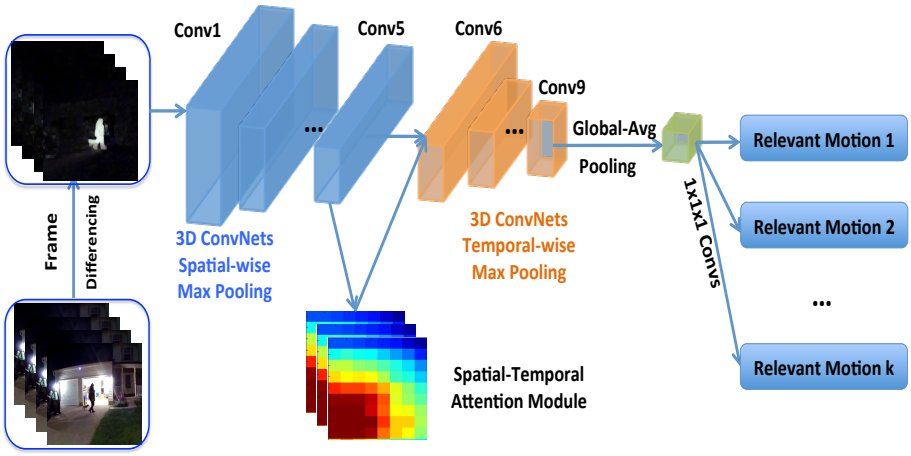


Figure 5.2: ReMotENet for Relevant Motion Event Detection.

5.2 Our Approach

To dramatically speedup relevant motion event detection and improve its performance, we propose a novel network for relevant motion event detection, ReMotENet, which is a unified, end-to-end data-driven method using spatial-temporal attention-based 3D ConvNets to jointly model the appearance and motion of objects-of-interest in a video.

Analyzing an Entire Video Clip at Once In contrast to object detection based method based on frame-by-frame processing, we propose a unified, end-to-end data-driven framework that takes an entire video clip as input to detect relevant motion using 3D ConvNets [78].

One crucial advantage of using 3D ConvNets is that they can parse an entire video clip in one forward pass of a deep network, which is extremely efficient. Meanwhile, unlike the traditional pipeline that conducts object detection and tracking independently, 3D ConvNets is an end-to-end model that jointly models the appearance of objects and their motion patterns. To fit an entire video in memory, we down-sample the video frames

spatially and temporally. We sample one FPS to uniformly sample 15 frames from a 15s video clip, and reduce the original resolution (1280×720) by a factor of 8. Since we focus on classifying whether there is a relevant motion in the video rather than fine-grained activity recognition, we believe that FPS 1 is enough to capture the motion. The input tensor of our 3D ConvNets is 15×90×160×3.

5.3 ReMotENet using Spatial-temporal Attention-based C3D

Frame Differencing Global or local context of both background or foreground objects has proven to be useful for activity recognition task [79,80] (e.g., some sports only happen on playgrounds; some collective activities have certain spatial arrangements of the objects that participant). However, since home surveillance cameras capture different scenes at different times with various weather and lighting conditions, the same relevant motion could involve different background and foreground arrangements.

So, we conduct a simple and efficient frame differencing on the 4D input tensor to suppress the influence of the background and foreground variance: for each frame sub-sampled from a video clip, we select its previous frame as a "reference-frame" and difference them. The image quality of surveillance cameras is usually low, and the moving objects are relatively small due to large field of view. As a result, it requires high resolution videos for object detector to capture fine-grained features, such as texture, to detect relevant objects. However, for ReMotENet with frame differencing, since we skip explicit object detection, it is sufficient for the network to learn coarse appearance features of objects, e.g., shape, contour and aspect-ratio with frames of low resolution, which

leads to significant speedup.

Spatial-temporal Attention-based 3D ConvNets Most video clips captured by a home surveillance camera may only contain the stationary scene with irrelevant motion such as shadow, rain and parked vehicles. Meanwhile, our network should be capable of differentiating appearance and motion pattern of different objects-of-interest.

Besides using frame differencing to suppress the background, we propose a Spatial-Temporal Attention-based (STA) model as shown in Figure 6.4 to guide the network to only pay attention to the moving objects of interest and filter out foreground motion caused by irrelevant objects. Unlike most of the attention models that are trained end-to-end with the groundtruth of final predictions, we use the detection confidence scores and bounding boxes of the moving relevant objects obtained from a general detector (e.g., Faster R-CNN) as pseudo-groundtruth to train the STA model.

The supervision from the detection results suppresses the influence of features from irrelevant regions of each frame, and the binary labels of motion adjusts the STA layer to pay attention to the spatial-temporal locations that are most informative for predicting the relevant motion. The spatial-temporal attention model can be viewed as a combination of multi-task learning and traditional attention model. In the weakly supervised framework, it helps our network to learn from noisy labels and recover some of the mistakes caused by the object detection based method. For instance, some relevant motion is missed because of bad thresholds of the tracking module, but the object detector has correctly detected the relevant objects. The STA model trained with those detection results is helpful for our network to recover the binary prediction mistake.

Different from the original C3D model that conducts max pooling both spatially and temporally [78], we separate the spatial and temporal max pooling as shown in Figure 6.4. This allows us to generate an attention mask on each input frame to capture fine-grained temporal information, and makes the network deeper to learn better representations. We first apply five layers of 3D convolutions (Conv1-Conv5) with spatial-wise max pooling on the 4D input tensor after frame differencing to extract the appearance based features. Then, we apply another 3D convolution layer (STA layer) on the output of Pool5 to obtain a binary prediction of whether our system should pay attention to each spatial-temporal location. We conduct a softmax operation on the binary predictions to compute a soft probability of attention for each spatial-temporal location. We scale the features from Pool5 by applying an element-wise multiplication between the attention mask and the extracted features. Another four layers of 3D ConvNets are used with temporal max pooling to abstract temporal features.

The network in [78] utilized several fully connected layers after the last convolution layer, which leads to a huge number of parameters and computations. Inspired by [81], we apply a spatial global average pooling (GAP) to aggregate spatial features after Pool9 and use several $1\times1\times1$ convolution layers with two filters (denoted as "Binary" layers) to predict the final binary results. The use of GAP and $1\times1\times1$ convolutions significantly reduces the number of parameters and model size of our method. The final outputs of our network are several binary predictions indicating whether there is any relevant motion of a certain object or a group of objects. The detailed network structure is shown in Table 5.1. For each Conv layer, we use ReLU as its activation.

Layer	Input Size	Kernel Size	Stride	Num of Filters
Conv1	15×90×160×3	3×3×3	1×1×1	16
Pool1	15×90×160×3	1×2×2	1×2×2	-
Conv2	15×45×80×16	3×3×3	1×1×1	16
Pool2	15×45×80×16	1×2×2	1×2×2	-
Conv3	15×23×40×16	3×3×3	1×1×1	16
Pool3	15×23×40×16	1×2×2	1×2×2	-
Conv4	15×12×20×16	3×3×3	1×1×1	16
Pool4	15×12×20×16	1×2×2	1×2×2	-
Conv5	15×6×10×16	3×3×3	1×1×1	16
Pool5	15×6×10×16	1×2×2	1×2×2	-
STA	15×3×5×16	3×3×3	1×1×1	2
Conv6	15×3×5×16	3×3×3	1×1×1	16
Pool6	15×3×5×16	2×1×1	2×1×1	-
Conv7	8×3×5×16	3×3×3	1×1×1	16
Pool7	8×3×5×16	2×1×1	2×1×1	-
Conv8	4×3×5×16	3×3×3	1×1×1	16
Pool8	4×3×5×16	2×1×1	2×1×1	-
Conv9	2×3×5×16	3×3×3	1×1×1	16
Pool9	2×3×5×16	2×1×1	2×1×1	-
GAP	1×3×5×16	1×3×5	1×1×1	-
Binary	1×1×1×16	1×1×1	1×1×1	2

Table 5.1: Network Structure of the ReMotENet using Spatial-temporal Attention-based 3D ConvNets

5.4 Network Training

Considering the large volume of home surveillance videos, it is time-consuming to annotate each training video with binary labels. So, we adopt a weakly-supervised learning framework that utilizes the pseudo-groundtruth generated from the object detection

based method (details are discussed in section 5.6). Besides binary labels, we also utilize the pseudo-groundtruth of detection confidence scores and bounding boxes of moving objects-of-interest obtained from the object detection based method to train the STA layer. The loss function of STA layer is:

$$loss = \frac{1}{N}\Bigg(C_1 \sum_n \sum_i w_{n,i} \text{CE}(g_{n,i}, y_{n,i}) + \frac{C_2}{W \cdot H \cdot T} \sum_n \sum_{w,h,t} \text{CE}(sta_{n,w,h,t}, Gsta_{n,w,h,t})\Bigg) \tag{5.1}$$

The first part is the cross-entropy loss (CE) for each relevant motion category; the second part is the CE loss between the predicted attention of each spatial-temporal location produced by "STA" layer and the pseudo-groundtruth obtained from the object detector. W, H, T are spatial and temporal size of the responses of layer "STA"; $y_{n,i}$ and $g_{n,i}$ are the predicted and groundtruth motion labels of the n^{th} sample; $sta_{n,w,h,t}$ and $Gsta_{n,w,h,t}$ are the predicted and groundtruth attention probabilities; $w_{n,i}$ is the loss weight of the n^{th} sample, which is used to balance the biased number of positive and negative training samples for the i^{th} motion category; N is the batch size; C_1 and C_2 are used to balance binary and STA loss. We choose $C_1 = 1$ and $C_2 = 0.5$ in this paper and train our network using Adam optimizer [82] with 0.001 initial learning rate. The training process converges fast with batch size 40 (¡5,000 iterations).

5.5 Experiments

5.6 Dataset

We collect 38,360 video clips from 78 home surveillance cameras. Each video is 15s and captured with FPS 10 and 1280×720 resolutions. The videos cover different times and various scenes, e.g., front door, backyard, street and indoor living room. The longest period a camera recorded is around 3 days. Those videos mostly capture only stationary background or irrelevant motion caused by shadow, lighting changes or snow/rain. Some of the videos contain relevant motion caused by people and vehicles (car, bus and truck).

The "relevant motion" in our system is defined with a list of relevant objects. In this paper, we define three kinds of relevant motion: "People motion", caused by object "people"; "Vehicle motion", caused by at least one object from {car, bus, truck}; "P+V Motion" (P+V), caused by at least one object from {people, car, bus, truck}. The detection performance of "P+V Motion" evaluates the ability of our method to detect general motion, and the detection performance of "People/Vehicle motion" evaluates the ability of differentiating motion caused by different kinds of objects.

Our network is trained using weak supervisions obtained from the object detection based method. We run Faster R-CNN based object detection with FPS 10 on the original 1280×720 video frames, and apply a state-of-the-art real-time online tracker from [83] to capture temporal consistency to obtain labels of relevant motion involving different objects-of-interest. We first run the above method on the entire dataset to detect videos

with relevant motion of P+V. Then, we randomly split the dataset into training and testing set with a 3:1 ratio. This leads to a training set with 28,732 video clips and a testing set with 9,628 video clips.

We have three annotators watching the testing videos and annotating binary labels for P+V, people and vehicle motion. If there is any human recognizable motion of the relevant objects in a video, we annotate "1" for the corresponding category. All three annotators must agree with each other, otherwise we mark the video as ambiguous and remove it from the testing set. We have 9,606 testing videos left after several rounds of annotation and cross-checking. Among the testing set, there are 973 videos with relevant motion events contain either people or vehicles (P+V), 429 videos have people motion only, 594 videos have vehicle motion only and 50 videos have both motion.

5.7 Baseline: Object Detection based Method

For the object detection based method, if one tracklet has at least two frames overlapped with the detected bounding boxes of the relevant objects with Intersection over Union (IOU) > 0.9, the average detection confidence score > 0.8, and the maximum relative location change ratio over the entire tracklet along width or height of the frame is large enough (> 0.2), we consider it as a valid tracklet. If there is at least one valid tracklet of an object in a video clip, we consider that video has valid motion of the specific object.

For background subtraction, we utilize the method proposed in [84]. We employ the start-of-the-art real-time online tracking method from [83]. The above two methods can

be efficiently run on CPU. For object detection, we utilized the Tensorflow Object Detection API [77]. Following the discussions in [77], we choose three state-of-the-art detectors: Faster R-CNN [73] with ResNet 101 [23], SSD [74] with inception V2 [85] and SSD with MobileNet [86]. According to [77], Faster R-CNN is the best detector considering detection accuracy and robustness, and are widely used in many applications [4, 6, 8, 87]. However, SSD based detector is much faster while suffers some accuracy degradation. To further reduce the computational cost of detection, MobileNet [86] is utilized as the base network in SSD framework. SSD-MobileNet is one of the most compact and fastest detectors. We do not compare to YOLO v2 due to its similar performance with SSD [76]. We do not compare to other detectors such as tinyYOLO [75] and SqueezeDet [88] due to their significant degradation of performance (e.g., ¿20% reduction in mAP from YOLO v2 to Tiny YOLO). We use F-score to jointly evaluate the detection precision and recall. The results are shown in Table 5.2.

From the tables we conclude that Faster R-CNN based detector achieves much better performance than the other two efficient detection methods. Meanwhile, image resolution and frame sample rate (FPS) have significant influence. If resolution or FPS is small, the performance of detector and tracker drops significantly. So, to achieve reasonable detection results, we need to employ robust object detection framework (Faster R-CNN) with large FPS and resolutions, which is inefficient and heavy (with model size $>$ 400MB).

We also conduct experiments without tracking. In general, tracking method improves precision but leads to lower recall. Taking the detecion of P+V motion using Faster R-CNN with FPS 10 and 1280$\times$720 resolution as an example, without tracking,

	Detector	FPS 1 1280×720	FPS 2 1280×720	FPS 5 1280×720	FPS 10 1280×720
P+V	F R-CNN	0.075	0.405	0.745	**0.785**
	SSD-Incep	0.103	0.191	0.403	0.258
	SSD-Mobile	0.048	0105	0.277	0.258
People	F R-CNN	0.116	0.519	0.766	**0.795**
	SSD-Incep	0.156	0.276	0.467	0.549
	SSD-Mobile	0.054	0.139	0.252	0.258
Vehicle	F R-CNN	0.020	0.244	0.627	**0.665**
	SSD-Incep	0.062	0.103	0.267	0.499
	SSD-Mobile	0.029	0.062	0.197	0.252

	Detector	FPS 10 320×180	FPS 10 160×90	FPS 5 320×180	FPS 5 160×90
P+V	F R-CNN	**0.667**	0.255	0.565	0.255
People	F R-CNN	**0.670**	0.251	0.574	0.256
Vehicle	F R-CNN	**0.610**	0.226	0.517	0.122

Table 5.2: F-score of relevant motion detection using different object detectors with different FPS and resolution settings.

Network structures of ReMotENet	C3D	FD-C3D	FD-D	FD-D-MT	FD-D -STA-NT	FD-D -STA-T	FD-D-STA-T-H	FD-D-STA-T-32	FD-D-STA -T-H-32
3D ConvNets?	✓	✓	✓	✓	✓	✓	✓	✓	✓
frame differencing?		✓	✓	✓	✓	✓	✓	✓	✓
deeper network?			✓	✓	✓	✓	✓	✓	✓
train STA layer with detection pseudo-GT?				✓		✓	✓	✓	✓
ST Attention?					✓	✓	✓	✓	✓
high resolution?							✓		✓
more filters?								✓	✓
AP: P+V	77.79	82.29	83.98	84.25	84.91	86.71	85.67	**87.07**	86.09
AP: People	62.25	72.21	73.69	74.41	75.82	78.95	**79.78**	77.92	77.54
AP: Vehicle	66.13	73.03	73.71	74.25	75.47	**77.84**	76.85	76.81	76.92

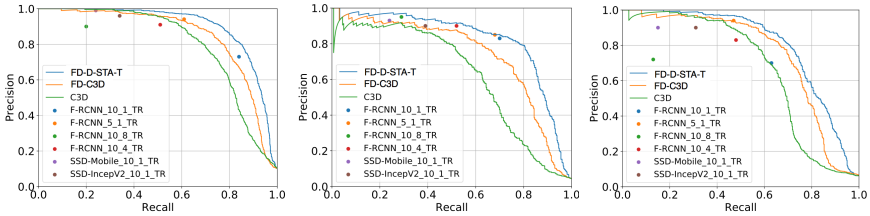
Table 5.3: The path from traditional 3D ConvNets to ReMotENet using Spatial-temporal Attention Model.

the method will have high recall (e.g., 0.8536) but very low precision (e.g., 0.2631); with tracking, it has lower recall (e.g., 0.7321) but much higher precision (e.g., 0.8467), and achieves much higher F-score.

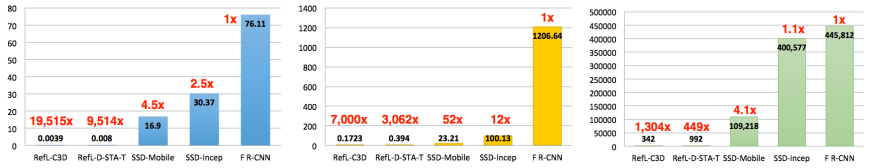
5.8 ReMotENet Performance

The outputs of ReMotENet are three binary predictions. After applying softmax on each binary prediction, we obtain probabilities of having P+V motion, people motion and vehicle motion in a video clip. We adopt Average Precision, which is a widely used evaluation metric for object detection and other detection tasks from [77] to evaluate our method. We evaluate different architectures and design choices of our methods, and report the average precision of detecting P+V motion, people motion and vehicle motion in Table 5.3. To show the improvement of our proposed method, we build a basic 3D ConvNets following [78]. We design a 3D ConvNets with 5 Conv layers followed by

Figure 5.3: Predicted Attention Mask of "FD-D-STA-NT" Method.



(a) PR Curve: P+V (b) PR Curve: People (c) PR Curve: Vehicle



(d) Run-time per 15s video on GPU (second) (e) Run-time per 15s video on CPU (second) (f) Model Size (KB)

Figure 5.4: Comparing with baselines.

spatial-temporal max pooling. Similar with the C3D in [78], we conduct 3×3×3 3D convolution with 1×1×1 stride for Conv1-Conv5, and 2×2×2 spatial-temporal max pooling with 2×2×2 stride on Pool2-Pool5. For Pool1, we conduct 1×2×2 spatial max pooling with 1×2×2 stride. Different from C3D in [78], we only have one layer of convolution in Conv1-Conv5. Meanwhile, instead of several fully connected layers, we apply a global average pooling followed by several 1×1×1 convolution layers after Conv5. The above basic architecture is called "C3D" in Table 5.3.

Evaluation of ReMotENet First, we evaluate the effect of frame differening. Column 2-3 in Table 5.3 show that by using frame differening (FD), our 3D ConvNets achieve much higher average precision for all three categories of motion, especially on people and vehicle motion detection task.

To evaluate the spatial-temporal attention model, we modify the basic C3D network architecture to separate spatial-wise and temporal-wise max pooling as shown in Table 5.1. We call the network with nine 3D ConvNets (without the STA layer) as "FD-D". "FD-D-MT" denotes using multi-task learning: we use the STA layer to predict the ST attention mask, which is trained by pseudo-groundtruth obtained from the object detection based method, but we do not multiply the attention mask with the extracted features after Pool5. Another model is "FD-D-STA-NT": we multiply the attention mask with the extracted features after Pool5 layer. However, the STA layer is trained with only binary labels of motion categories but without the detection pseudo-groundtruth. We observe that incorporating multi-task learning or end-to-end attention model only leads to small improvement, but once we combine both methods, our "FD-D-STA-T" model achieves significant improvement. Our intuition is that, adding multi-task learning alone does not directly affect the final prediction. Meanwhile, due to the sparsity of moving relevant objects in the videos, the number of positive and negative spatial-temporal location from the detection pseudo-groundtruth is extremely biased. This leads to overfitting of the model to predict the attention of all the spatial-temporal location as 0. Meanwhile, adding attention model without multi-task learning also leads to small improvement. We observe that without the weak supervision of specific objects and their locations, the attention mask predicted by "FD-D-STA-NT" may focus on motion caused by some irrelevant

objects, such as pets, trees and flags shown in Figure 5.3.

To encourage our network to pay attention to the relevant objects (in this paper, people and vehicles), we propose our full model "FD-D-STA-T", which can be viewed as a combination of multi-task learning and attention model. We use the detected bounding boxes of relevant moving objects to train the STA layer, and multiply the predicted attention mask with the extracted features from Pool5 layer. "FD-D-STA-T" achieves much higher average precision than the previous models in all three motion categories. The results of the above models are listed in column 5-8 of Table 5.3.

We also conduct experiments with other network design choices of ReMotENet. For instance, we add more filters in each convolution layer, or enlarge the input resolution from 160×90 to 320×180. As shown in Table 5.3, those design choices lead to insignificant improvements. The experiments demonstrate that ReMotENet can precisely detect relevant motion with small input FPS and resolution.

5.9 Comparing with the Object Detection based Method

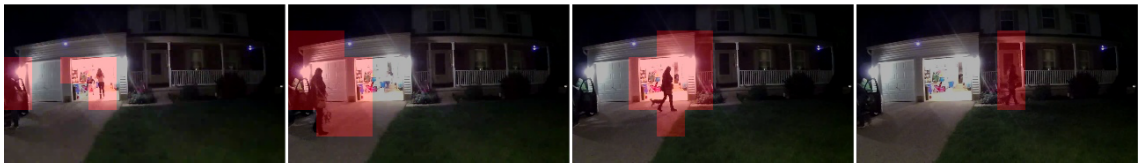
Although ReMotENet is trained with the pseudo-groundtruth obtained from the object detection based method, it outperforms the baseline in several ways:

Detection Performance. Although the training labels are noisy, ReMotENet can learn patterns of relevant motion and generalize well. We show the PR curve of our methods and the performance of the object detection based method in Figure 5.4(a), 5.4(b) and 5.4(c). From the PR curve of three models: "C3D", "FD-C3D" and "FD-D-STA-T", we demonstrate the effectiveness of frame differencing and the STA model. The PR curves of

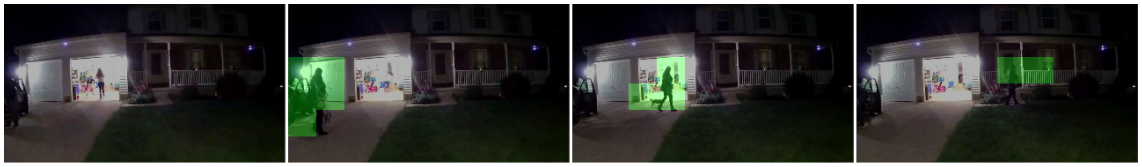
our full model "FD-D-STA-T" are higher than all PR points of the object detection based method, which shows that our method can achieve similar or better performance than the baselines by properly choosing the detection threshold. Another advantage of our method is that it is a probabilistic model, and one can tune the threshold based on the priority of precision or recall.

Run-time and Model Size. From Table 5.2 we observe that with a fixed detector, the object detection based method needs large FPS and resolutions to achieve good detection performance, especially for SSD based detectors. So, we show the time and model size benchmark results of the baselines with FPS 10 and 1280×720 resolution, which can achieve the best detection performance, in Figure 5.4(d), 5.4(e) and 5.4(f). For baselines, the run-time consists of the time for background subtraction, object detection and tracking; for ReMotENet, the run-time consists of frame differencing and the forward pass of our 3D ConvNets. We omit the time for decoding and sampling frames from the video clips for both methods. The model size is the size of Tensorflow model data file. We omit the size of meta and index file. Our method can achieve 9,514×-19,515× speedup on GPU (GeForce GTX 1080) and 3,062×-7,000× speedup on CPU (Intel Xeon E5-2650 @2.00GHz). Meanwhile, since our 3D ConvNets are fully-covolutional, and very compact (16 filters per Conv layer), we can achieve 449×-1,304× reduction on model size. Our "FD-C3D" model can analyze more than 256 15s video clips per second on GPU and around 6 videos on CPU with a 300KB model; "FD-D-STA-T" can parse 125 or 2.5 videos of 15s on GPU or CPU devices respectively with a model less than 1MB. Our method can not only efficiently analyze home surveillance videos on cloud with GPUs, but also be potentially running on the edge.

(a) Failure Cases: Object Detection based method



(b) Spatial-temporal Attention Prediction: ReMotENet (FD-D-STA-T)



(c) Spatial-temporal Attention: Pseudo-groundtruth

(d) Failure Cases: ReMotENet (FD-D-STA-T)

Figure 5.5: Visualization of results from the detection based method and ReMotENet.

5.10 Visualization

We visualize the results of the baseline and our method in Figure 5.5(a) to 5.5(d). Figure 5.5(a) shows miss/false detection of people and vehicle motion using the object detection based method. Due to the low quality of video, shadow, small object size and occlusion, object detector may fail. However, because our method is a data-driven framework trained specifically on the home surveillance video with the above properties, and

our method jointly model both object appearance and motion, we can properly handle the above cases. Figure 5.5(b) and 5.5(c) show the predicted spatial-temporal attention mask (red rectangles) of our "FD-D-STA-T" model and the pseudo-groundtruth obtained from the object detection based method. Although our "FD-D-STA-T" model can only predict a coarse attention mask, it captures the motion pattern of the people in the video. Meanwhile, although trained by the pseudo-groundtruth, our method can recover mistakes of the object detection based method (see the first figures of 5.5(b) and 5.5(c)). Figure 5.5(d) shows several failure cases of our method. For the miss detection (the second and forth figures), although our 3D ConvNets predict there is no relevant motion, it still correctly predict the coarse attention mask that cover the car and people. For the falsely detected vehicle motion (the first figure), we find that due to reflection of light, the input after frame difference is very noisy; for the false people motion detection (the third figure), our method falsely detect bird motion as people. With more training data with different objects and lighting conditions, hopefully our method can recover the above mistakes and obtain better performance.

5.11 Conclusion

We propose an end-to-end data-driven framework to detect relevant motion from large-scale home surveillance videos. Instead of parsing a video in a frame-by-frame fashion using the object detection based method, we proposed to use 3D ConvNets to parse an entire video clip at once to dramatically speedup the process. We extended the 3D ConvNets by incorporating a spatial-temporal attention model to encourage the network

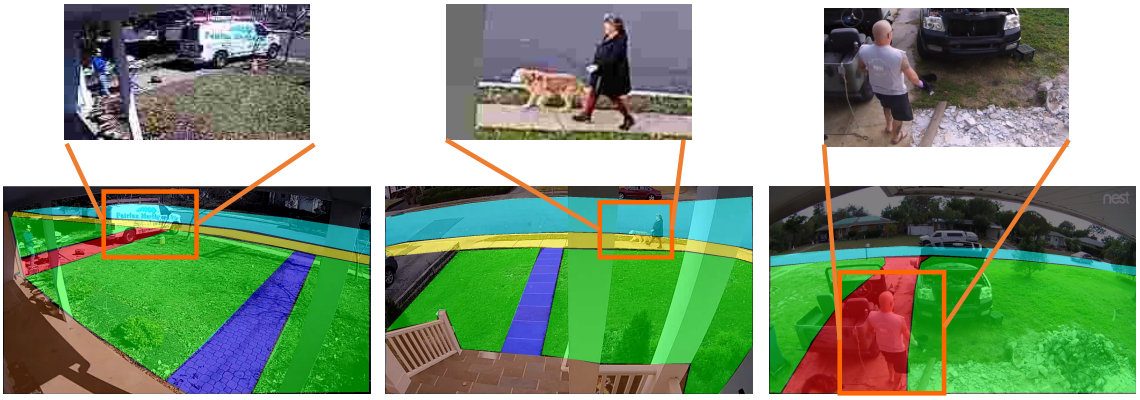
to pay more attention to the moving objects. Evaluations demonstrate that ReMotENet achieves comparable or better performance than the object detection based method while achieving three to four orders of magnitude speedup on GPU and CPU devices. ReMotENet is very efficient and compact, and therefore naturally implementable on the edge (camera).

Chapter 6: Layout-induced Video Representation for Recognizing Agent-in-Place Actions

6.1 Introduction

Recent advances in deep neural networks have brought significant improvements to many fundamental computer vision tasks, including video action recognition [89–96]. Current action recognition methods are able to detect, recognize or localize general actions and identify the agents (people, vehicles, *etc*.) performing them [78,89–91,93–101]. However, in applications such as surveillance, relevant actions often involve references to locations and directions–for example, it might be of interest to detect (*i.e.*issue an alert) a person walking towards the front door of a house, but not to detect a person walking along the sidewalk. So, what makes an action "interesting" is how it interacts with the geometry and topology of the scene in which it is performed. However, the layout of even a restricted set of scenes–cameras mounted over the entrance doors of houses monitoring the fronts and backs of the houses–vary significantly.

We describe how to represent the geometry and topology of scene layouts so that a network can generalize from the layouts observed in the training set to unseen layouts in the test set. Examples of these contextualized actions in outdoor home surveillance



<vehicle, move away (home), driveway>

<pet, move along, sidewalk> <person, move along, sidewalk>

<person, move away (home), driveway>

Figure 6.1: Example agent-in-place actions and segmentation maps. Different colors represent different places. We zoom in to the agents performing the actions for clarity. An agent-in-place action is represented as *¡agent, action, place¿*.

scenarios and the semantic segmentations of scenes from which the representations are constructed as shown in Fig.6.1. We will refer to these contextualized actions as "*agent-in-place*" actions to distinguish them from the widely studied generic action categories. By encoding layout information (class membership of places, layout geometry and layout topology) into the network architecture we eliminate the need to collect massive amounts of training data that would span the space of possible layouts. This allows the model to abstract away appearance variations and focus on how actions interact with the scene layouts. Without large-scale training data, a model that does not incorporate a representation of scene layout can easily overfit to the training scene layouts and exhibit poor generalization on new scenes.

To address the generalization problem, we propose *Layout-Induced Video Representation* (LIVR), which encodes a scene layout given the segmentation map of a static scene. The representation has three components: *1)* A semantic component represented

by the characteristic functions of the semantic labels of the layouts (a set of bitmaps used for feature aggregation in the convolutional layers of the network referred to as *"places"*); *2)* A geometric component represented by a set of coarsely quantized distance transforms of each semantic place incorporated into the convolutional layers of the network; *3)* A topological component represented through the connection structure in a dynamically gated fully connected layer of the network–essentially aggregating representations from adjacent (more generally h-connected for h hops in the adjacency graph of the semantic map) places. Fig.6.2 shows an illustration of the proposed framework.

The first two components require *semantic feature decomposition* as shown in blue in Fig.6.2. We utilize bitmaps encoded with the semantic labels of places to decompose video representations into different places and train models to learn place-based feature descriptions. This decomposition encourages the network to learn features of generic place-based motion patterns that are independent of scene layout. As part of the semantic feature decomposition, we encode scene geometry to model moving directions by dis-

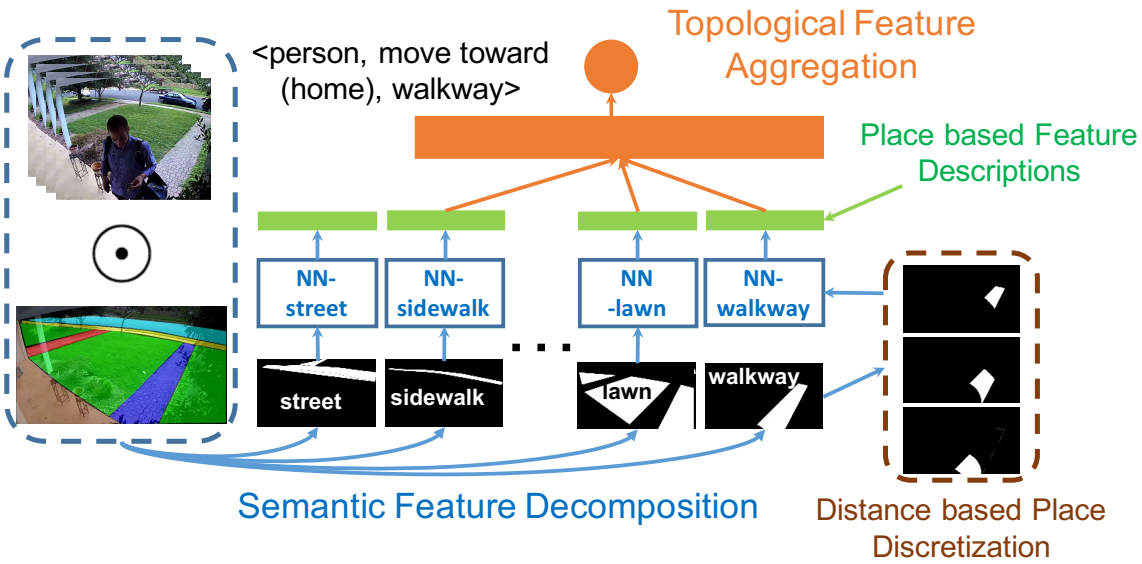
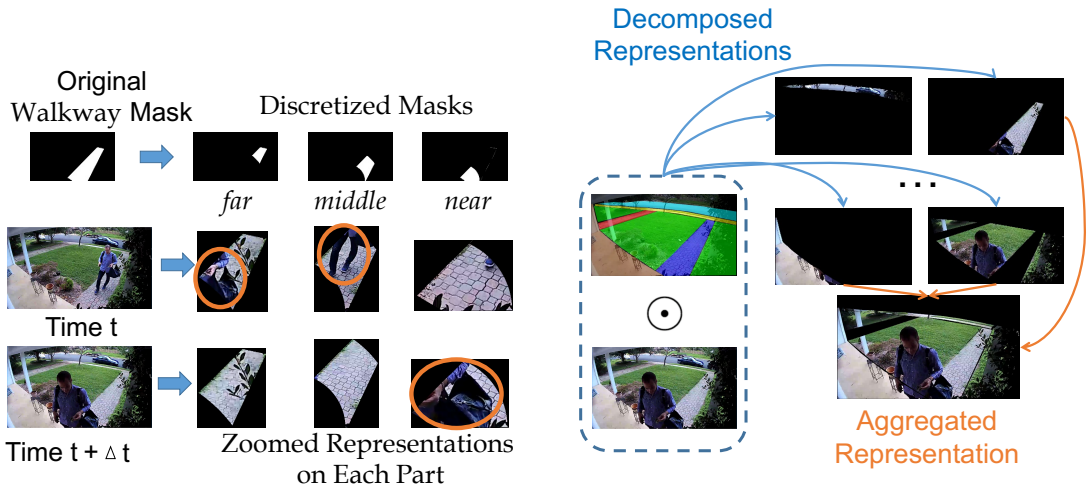


Figure 6.2: Framework of LIVR.



(a) Place Discretization (b) Topological Feature Aggregation

Figure 6.3: (a) illustrates distance-based place discretization. (b) illustrates the motivation behind topological feature aggregation.

cretizing a place into parts based on a quantized distance transform w.r.t. another place. Fig.6.2 (brown) shows the discretized bitmaps of *walkway* w.r.t. *porch*. As illustrated in Fig.6.3(a), features decomposed by those discretized bitmaps capture moving agents in spatial-temporal order, which reveals the moving direction, and can be generalized to different scene layouts. With place-based feature descriptions, we predict the confidence of an action by dynamically aggregating features within the place associated with that action and adjacent places, since the actions occurring in one place may also be projected onto adjacent places from the camera view(see Fig.6.3(b)). This *topological feature aggregation* controls the "on/off" state of neuron connections from place-based feature descriptions to action nodes at both training and testing time based on scene topological connectivity.

To evaluate LIVR, we collect a dataset called Agent-in-Place Action dataset, which

consists of over 5,000 15s videos obtained from 26 different surveillance scenes with around 7,100 actions from 15 categories. To evaluate the generalization of LIVR, we split the scenes into observed and unseen scenes. Extensive experiments show that LIVR significantly improves the generalizability of the model trained on only observed scenes and tested on unseen scenes (improving the mean average precision (mAP) from around 20% to more than 51%). Consistent improvements are observed on almost all action categories.

6.2 Related Work

Video Understanding. Our work is related to video understanding, especially video action recognition, which takes video clips as input and categorizes the actions (usually human activities) occurring in the video. Recent work in video activity recognition explores different frameworks, *e.g.*two-stream based models [89–91], RNN based models [97–99] and 3D ConvNets based methods [78, 100, 101] on different datasets [93–96]. In the context of home surveillance video understanding, prior work focuses on developing robust, efficient and accurate surveillance systems that can detect, recognize and track actions or events [102–104], or to detect abnormal events [105–107]. The most related work to ours is ReMotENet [108], which skips expensive object detection [73, 109–111] and utilizes 3D ConvNets to detect motion of an object-of-interest in outdoor home surveillance videos. We employ a similar 3D ConvNet model as proposed in [108] as a backbone architecture for extracting place-based feature descriptions for our model. Unlike previous work, our agent-in-place actions are associated with places, and we focus on improving

the generalization of a model by modeling geometrical and topological relationships in scene layouts.

Region-based Representation. Region-based representations have been widely used in computer vision tasks [112–124]. For example, SIFT [114] represents an image using features extracted from blobs; spatial pyramid pooling based methods partition the image into divisions and aggregate local features in them [115–121]; some methods utilize high-level semantic representations extracted from objects in images or actions in videos [112, 122–126]. Li *et al.* [127] represented videos by learning from weak detection bounding boxes and pooling only features related to facial regions for video face verification. Zhao *et al.* [128] proposed an image representation based on pooling semantic features of individual objects into a feature map. Our method also leverages region-based representations, but we decompose regions based on scene layout to extract place-based features and aggregate them according to scene topology. Our method abstracts complex scene layouts to learn scene-independent features to generalize to unseen scenes.

Knowledge Transfer. The biggest challenge of agent-in-place action recognition is to generalize a model trained with limited scenes to unseen scenes. Previous work on knowledge transfer in both image and video domain has been based on visual similarity, which requires a large amount of training data [129–135]. For trajectory prediction, Ballan *et al.* [129] transferred the priors of statistics from training scenes to new scenes based on scene similarity. Kitani *et al.* [136] extracted static scene features to learn scene-specific motion dynamics for predicting human activities. Instead of utilizing low-level visual similarity for knowledge transfer, our video representation abstracts away appearance

and location variance and models geometrical and topological relationships in a scene.

6.3 Layout-Induced Video Representation

6.3.1 Framework Overview

The network architecture of the layout-induced video representation is shown in Fig.6.4. For each video, we stack sampled frames of a video clip into a 4-D input tensor. Frame differences are used as the input since they suppress the influence of scene background and encourage the model to capture more abstract visual concepts [108]. Our backbone network is similar to the architecture of ReMotENet [108], which is composed of 3D Convolution (3D-conv) blocks. A key component of our framework is semantic feature decomposition, which decomposes feature maps according to region semantics obtained from given segmentation masks. This feature decomposition can be applied after any 3D-conv layer. Spatial Global Max Pooling (SGMP) is applied to extracted features within places, allowing the network to learn abstract features independent of shapes, sizes and absolute coordinates of both places and moving agents. For predicting each action label, we aggregate features from different places based on their connectivity in the segmentation map, referred to as Topological Feature Aggregation.

6.3.2 Semantic Feature Decomposition

Segmentation Maps Semantic Feature Decomposition utilizes a segmentation map of each place to decompose features and force the network to extract place-based feature descriptions individually. The segmentation maps can be manually constructed using a

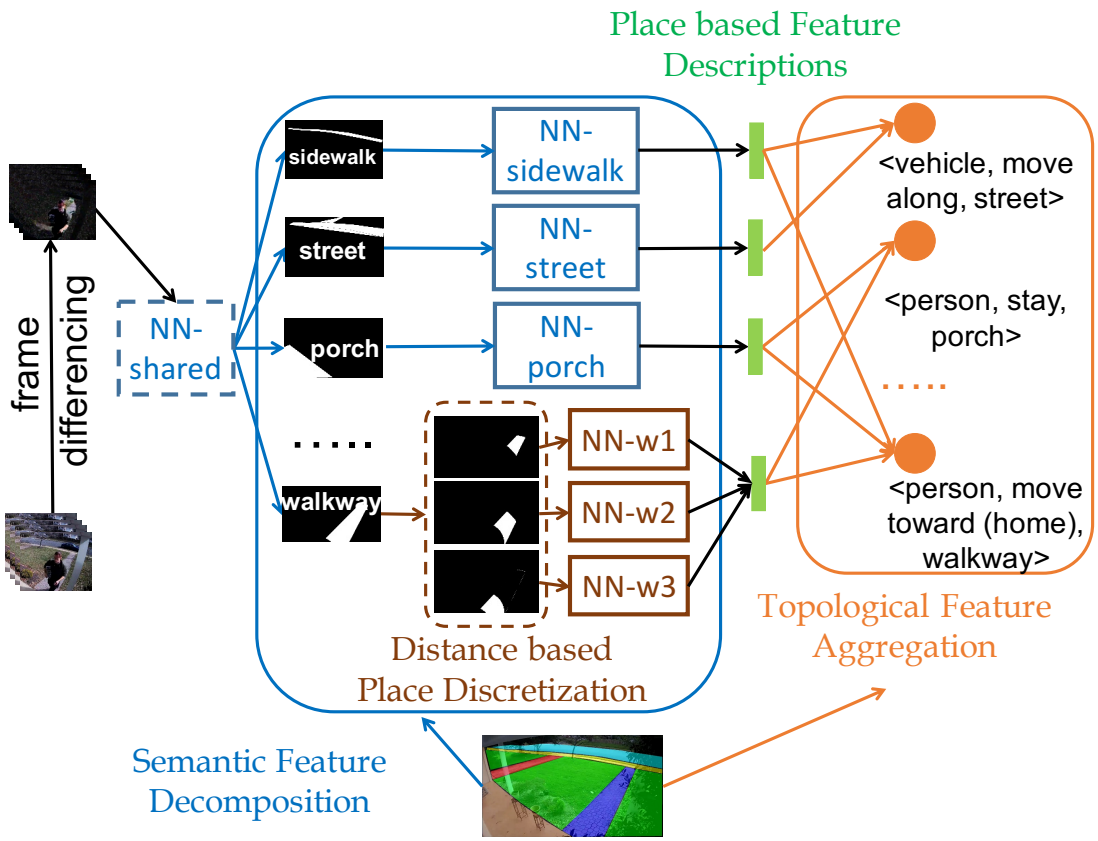


Figure 6.4: Layout-induced Video Representation Network.

mobile app we developed to annotate each place by drawing points to construct polygons. We believe that the human annotations are preferred when compared to automatic semantic segmentation methods for our task. Because the later ones segment places based on appearance (*e.g.*color, texture, *etc.*), while our task requires to differentiate places with similar appearance based on their functionality. For example, walkway, street and driveway may confuse the appearance based methods due to their similar appearance, but they have different functionalities in daily life, which can be easily and efficiently differentiated by human. Meanwhile, considering the fact that most of the time a surveillance camera should be fixed, users can annotate one map per camera very efficiently. The remaining parts of this paper will assume perfect segmentation as it is reasonably easy

to obtain. However, we will discuss the performance of our method using automatically generated maps in Sec. 6.5.4.

Place-based Feature Descriptions (PD). Given a segmentation map, we extract place-based feature descriptions as shown in the blue boxes in Fig.6.4. We first use the segmentation map represented by a set of binary masks to decompose feature maps spatially into regions, each capturing the motion occurring in a certain place. The decomposition is applied to features instead of raw inputs to retain context information[1]. Let $\mathbf{X}_L \in \mathbb{R}^{w_L \times h_L \times t_L \times c}$ be the output tensor of the L^{th} conv block, where w_L, h_L, t_L denote its width, height and temporal dimensions, and c is the number of feature maps. The place-based feature description of a place indexed with p is

$$f_{L,p}(\mathbf{X}_L) = \mathbf{X}_L \odot \mathbb{I}\left[\mathbf{M}_L = p\right] \tag{6.1}$$

where $\mathbf{M}_L \in \mathbb{I}^{w_L \times h_L \times 1}$ is the segmentation index map and $\odot$ is a tiled element-wise multiplication which tiles the tensors to match their dimensions. Place descriptions can be extracted from different levels of feature maps. $L = 0$ means the input level; $L > 0$ means after the L^{th} 3D-conv blocks. A higher L generally allows the 3D ConvNet to observe more context and abstract features. We treat L as a hyper-parameter of the framework and study its effect in Sec. 6.5.

Distance-based Place Discretization (DD). Many actions are naturally associated with moving directions *w.r.t.*some scene element (*e.g.*, the house in home surveillance). To

[1] An agent can be located at one place, but with part of its body projected onto another place in the view of the camera. If we use the binary map as a hard mask at input level, then for some places such as *sidewalk*, *driveway* and *walkway*, only a small part of the moving agents will remain after the masking operation.

learn general patterns of the motion direction in different scenes, we further discretize the place segmentation into several parts, and extract features from each part and aggregate them to construct the place-based feature description of this place. For illustration, we use *porch* as the anchor place (shown in Fig.6.5). We compute the distance between each pixel and the *porch* in a scene (distance transform), and segment a place into k parts based on their distances to *porch*. The left bottom map in 6.5 shows the porch distance transform of a scene. Let $D_L(x)$ be the distance transform of a pixel location x in the L^{th} layer. The value of a pixel x in the part indexing map $\mathbf{M}_L^{\Delta}$ is computed as

$$M_L^{\Delta}(x) = \left\lfloor \frac{D_L^{\max}(x) - D_L^{\min}(x)}{k(D_L(x) - D_L^{\min}(x))} \right\rfloor \tag{6.2}$$

where $D_L^{\max}(x) = \max\{D_L(x') | M_L(x') = M_L(x)\}$ and $D_L^{\min}(x) = \min\{D_L(x') | M_L(x') = M_L(x)\}$ are the max and min of pixel distances in the same place. They can be efficiently pre-computed. The feature description corresponding to the i^{th} part of p^{th} place in L^{th} layer is

$$f_{L,p,i}^{\Delta}(\mathbf{X}_L) = \mathbf{X}_L \odot \mathbb{I}[\mathbf{M}_L = p \wedge \mathbf{M}_L^{\Delta} = i] \tag{6.3}$$

where $\odot$ is the tiled element-wise multiplication.

Discretizing a place into parts at different distances to the anchor place and explicitly separating their spatial-temporal features allows the representation to capture moving agents in spatial-temporal order and extract direction-related abstract features. However, not all places need to be segmented since some places (such as *sidewalk*, *street*) are not associated with any direction-related action (*e.g.*moving toward or away from the house). For these places, we still extract the whole-place feature descriptors $f_{L,p}$. We will study the different choices of place discretization and the number of parts k in Sec. 6.5. To

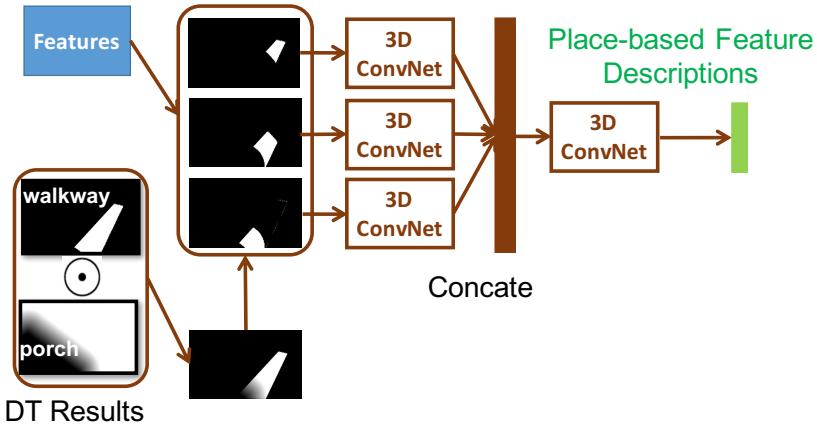


Figure 6.5: The process of distance-based place discretization.

preserve temporal ordering, we apply 3D-conv blocks with spatial-only max pooling to extract features from each discretized place, and concatenate them channel-wise. Then, we apply 3D-conv blocks with temporal-only max pooling to abstract temporal information. Finally, we obtain a 1-D place-based feature description after applying GMP (see Fig.6.5). It is worth noting that the final description obtained after distance-based place discretization has the same dimensions as non-discretized place descriptions.

6.3.3 Topological Feature Aggregation (Topo-Agg)

Semantic feature decomposition allows us to extract a feature description for each place individually. In order to predict action labels, these place features need to be aggregated. Each action is one-one mapped to a place. To predict the confidence of an action a occurring in a place p, features extracted far from place p are distractors. To reduce interference by features from irrelevant places, we structure the network to ignore these far away features using *Topological Feature Aggregation*, which utilizes the spatial connectivity between places, to guide feature aggregation.

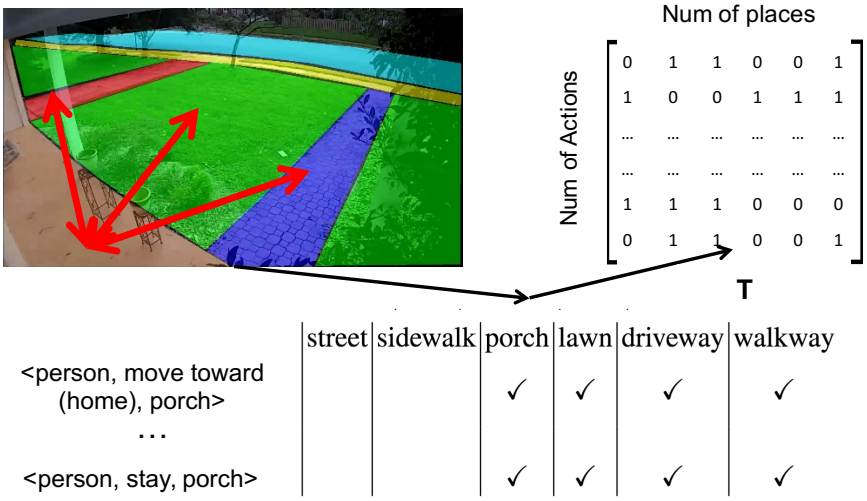


Figure 6.6: Topological feature aggregation which utilizes the connectivities between different places in a scene to guide the connections between the extracted place-based feature descriptions and the prediction labels.

Specifically, as shown in Fig.6.6, given a scene segmentation map, a source place p and a constant h, we employ a Connected Component algorithm to find the h-connected set $C_h(p)$ which contains all places connected to place p within h hops. The constant h specifies the minimum number of steps to walk from the source to a destination place. Given the h-connected place set C_h, we construct a binary action-place matrix ($\mathbf{T} \in \mathbb{R}^{n_a \times n_p}$) for the scene where n_a is the number of possible actions and n_p is the number of places. $\mathbf{T}_{i,j} = 1$ if and only if place j is in the C_h of the place corresponding to action i. Fig.6.6 shows an example segmentation map with its action-place mapping, where $C_0(porch) = \{porch\}$, $C_1(porch) = \{porch, walkway, driveway, lawn\}$, $C_2(porch)$ includes all except for *street*, and $C_3(porch)$ covers all six places.

We implement topological feature aggregation at both training and testing using a gated fully connected layer with customized connections determined by the action-place

mapping $\mathbf{T}$. Given n_p m-D features extracted from n_p places, we concatenate them to form a $(n_p \times m)$-D feature vector. We use $\mathbf{T}$ to determine the "on/off" status of each connection of a layer between the input features and the output action confidences. Let $\mathbf{T}_* = \mathbf{T} \otimes \mathbb{I}^{1\times m}$ be the actual mask applied to the weight matrix $\mathbf{W} \in \mathbb{R}^{n_a \times n_p m}$ where $\otimes$ is the matrix Kronecker product. The final output is computed by $\mathbf{y} = (\mathbf{W} \odot \mathbf{T}_*)\mathbf{f}_*$, where $\odot$ is element-wise matrix multiplication, $\mathbf{f}_*$ is the concatenated feature vector as the input of the layer. We omit bias for simplicity. Let J be the training loss function (cross-entropy loss), considering the derivative of $\mathbf{W}$, the gradient formulation is $\nabla_{\mathbf{W}} J = (\nabla_{\mathbf{y}} J \mathbf{f}_*^\intercal) \odot \mathbf{T}_*$, which is exactly the usual gradient $(\nabla_{\mathbf{s}} J \mathbf{f}^T)$ masked by $\mathbf{T}_*$. At training time, we only back-propagate the gradients to connected neurons. We also experiment with multiple fully connected layers, discussed in Sec. 6.5.

6.4 Agent-in-Place Action Dataset

We introduce a surveillance video dataset for recognizing agent-in-place actions. We collected outdoor home surveillance videos from internal donors and webcams[2] for months to obtain over $7,100$ actions from around $5,000$ 15-second video clips with 1280×720 resolution. These videos are captured from 26 different outdoor cameras which cover various layouts of typical American families' front yards and back yards.

We select 15 common agent-in-place actions to label and each is represented as a tuple containing an action, the agent performing it, and the place where it occurs. The agents, actions, and places involved in our dataset are: $Agent = \{$*person*, *vehicle*, *pet*$\}$;

[2]http://www.nestcamdirectory.com/

$Action = \{$*move along*, *stay*, *move away (home)*, *move toward (home)*, *interact with vehicle*, *move across*$\}$; $Place = \{$*street*, *sidewalk*, *lawn*, *porch*, *walkway*, *driveway*$\}$.

The duration of each video clip is 15s, so multiple actions can be observed from a single agent or multiple agents in one video. We formulate action recognition as a multi-label classification task. We split the 26 cameras into two sets: observed scenes (5) and unseen scenes (21) to balance the number of instances of each action in observed and unseen scenes and at the same time cover more scenes in the unseen set. We train and validate our model on observed scenes, and test its generalization capability on the unseen scenes. The detailed statistics of our cleaned dataset is shown in Fig. 6.7.

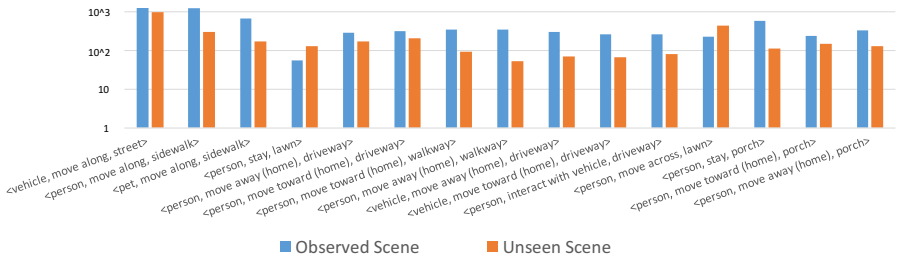


Figure 6.7: Dataset Statistics on observed and unseen scenes.

6.5 Experiments

6.5.1 Implementation Details

Network Architecture. Unlike traditional 3D ConvNets which conduct spatial-temporal max-pooling simultaneously, we found that decoupling the pooling into spatial-only and temporal-only leads to better performance. So, for each place-specific network that extracts place-based feature descriptions, we utilize nine blocks of 3D ConvNets with the first five blocks using spatial-only max pooling and the last four blocks using

temporal-only max pooling. The first two blocks have one 3D-conv layer each, and there are two convolutional (conv) layers with ReLU in between for the remaining blocks. For each place-specific network, we use 64 $3 \times 3 \times 3$ conv filters per 3D-conv layer. After conducting SGMP on features extracted by each place-specific network, the final concatenated 1-D feature dimension is 6×64 since there are 6 places in total. The inference is conducted with a gated fully connected layer, whose connections ("on/off" status) are determined by action labels and scene topology. We use the sigmoid function to obtain the predicted probability of each action. It is worth noting that if we conduct feature level decomposition ($L > 0$), we use a shared network to extract low-level features.

Anchor Place. For our dataset, the directions mentioned are all relative to the house location, and *porch* is a strong indicator of the house location. So we only conduct distance transform to *porch*[3], but the distance-based place discretization method can be easily generalized to represent moving direction w.r.t any arbitrary anchor place.

Training and Testing Details. Our action recognition task is formulated as multi-label classification without mutual exclusion. The network is trained using the Adam optimizer [82] with 0.001 initial learning rate. For input video frames, we follow [108] to use FPS 1 and down-sample each frame to 160×90 to construct a $15 \times 160 \times 90 \times 3$ tensor for each video as input. Suggested by [108], small FPS and low resolution are sufficient to model actions for home surveillance where most agents are large and the motion patterns of actions are relatively simple. We evaluate the performance of recognizing each action independently and report Average Precision (AP) for each action

[3]If there is no *porch* in a scene, we let the user to draw a line (click to generate two endpoints) to indicate its location

and mean Average Precision (mAP) over all categories.

Hyperparameter Selection. We split the 26 scenes into two sets: observed scenes and unseen scenes. We further split the videos in observed scenes into training and validation sets with a sample ratio of nearly $1:1$. We train our model on observed scenes and test it on unseen scenes. The validation set is used for tuning hyperparameters: we decompose semantics after the second conv blocks ($L = 2$); we conduct distance-based place discretization on $PL_{DT} = \{walkway, driveway, lawn\}$ and choose $k = 3$; for topological feature aggregation, we choose $h = 1$.

6.5.2 Baseline Models

We follow [108] to employ 3D ConvNets as our baseline (B/L) model. All three baseline models share the same 3D ConvNets architecture, which is very similar to the architectures of each place-specific network that extracts place-based feature descriptions, except that the last layer is fully connected instead of gated through topological feature aggregation. The difference among baseline 1, 2 and 3 is their input: B/L1 takes the raw frames as input; B/L2 applies frame difference on two consecutive frames; B/L3 incorporates the scene layout information by directly concatenating the 6 segmentation maps to the RGB channels in each frame (we call this method ConcateMap), resulting in an input of 9 channels per frame in total. We train the baseline models using the same setting as in the proposed model, and the performance of the baselines are shown in column 2-5 in Table 6.1. We observe that: 1) adding frame differencing leads to significant performance improvements; 2) marginal improvements are obtained by incorporating scene layout information using ConcateMap; 3) the testing performance gap between observed

and unseen scenes is large, which reveals the poor generalization of the baseline models. In addition, we also train a B/L3 model with $6\times$ more filters per layer to evaluate whether model size is the key factor for the performance improvement. The result of this enlarged B/L3 model is shown in column 5 of Table 6.1. Overall, the baseline models which directly extract features jointly from the entire video suffer from overfitting, and simply enlarging the model size or directly using the segmentation maps as features does not improve their generalization.

Network Architecture	B/L1	B/L2	B/L3	B/L3 +MF	Ours-V1	Ours-V2	Ours-V3	Ours-V4	Ours-V4+H	Ours-V4+MF	Ours-V4+FPS2	TSN [137]	ReMotENet [108]
3D ConvNet?	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-
frame differencing?		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-
ConcateMap?			✓	✓								-	-
place-based feature description?					✓	✓	✓	✓	✓	✓	✓	-	-
distance-based place discretization?							✓	✓	✓	✓	✓	-	-
topological feature aggregation?						✓		✓	✓	✓	✓	-	-
higher resolutions?									✓			-	-
more filters?				✓						✓		-	-
higher FPS?											✓	-	-
Observed scenes mAP	43.34	51.79	53.54	52.02	60.17	**60.97**	57.64	56.71	56.26	56.01	58.93	57.41	53.17
Unseen scenes mAP	14.62	18.46	20.96	16.12	41.54	44.73	46.57	51.41	49.13	**51.87**	50.31	23.26	21.13

Table 6.1: The path from traditional 3D ConvNets to our methods.

6.5.3 Evaluation on the Proposed Method

The path from traditional 3D ConvNets to our method. We show the path from the baselines to our method in Table 6.1. In column 6-9, we report the mAP of our models on observed scene validation set and unseen scene testing set. We observe three significant performance gaps, especially on unseen scenes: 1) from B/L3 to Ours-V1, we obtain over 20% mAP improvement by applying the proposed semantic feature decom-

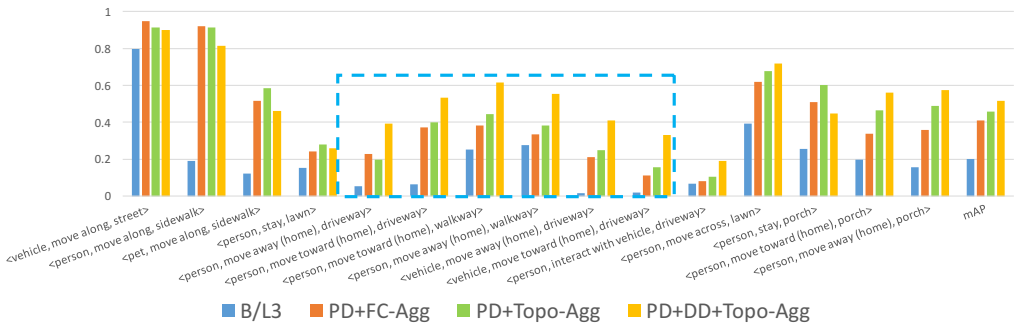


Figure 6.8: Per-category average precision of the baseline 3 and our methods on unseen scenes.

position to extract place feature descriptions; 2) from Ours-V1 to Ours-V3, our model is further improved by explicitly modeling moving directions by place discretization; 3) when compared to using a fully connected layer for feature aggregation (V1 and V3), our topological method (V2 and V4) leads to another significant improvement, which shows the efficacy of feature aggregation based on scene layout connectivity. We also evaluate the effect of resolutions, FPS and number of filters using our best model (Ours-V4). Doubling the resolution (320×180), FPS (2) and number of filters (128) only results in a slight change of the model's accuracy (columns 10-12 in Table 6.1). Besides our baselines, we also apply other state-of-the-art video action recognition methods (TSN [137] and ReMotENet [108]) on our dataset. LIVR outperforms them by a large margin, especially on the unseen scenes.

Per-category Performance. Fig.6.8 shows the average precision for each action on unseen scenes. Our method outperforms the baseline methods by a large margin on almost all action categories. When comparing the orange and green bars in Fig.6.8, we observe that the proposed topological feature aggregation (Topo-Agg) leads to consistently better generalization for almost all actions. The blue dashed box highlights the actions that

include moving directions, and consistent improvements are brought by distance-based place discretization (DD). For some actions, especially the ones occurring on street and sidewalk, since they are relatively easy to recognize, adding DD or Topo-Agg upon the place-based feature descriptions (PD) does not help much. Overall, our layout-induced video representation improves the generalization capability of the network, especially for actions that are more challenging, and are associated with moving directions.

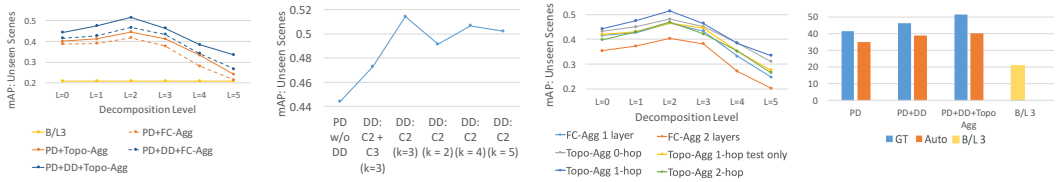
Qualitative Results. Some example actions are visualized using three frames in temporal order and the predicted probabilities of the groundtruth actions using different methods are reported in Fig.6.9. It is observed that for relatively easy actions such as *¡vehicle, move along, street¿*, performance is similar across approaches. However, for challenging actions, especially ones requiring modeling moving directions such as *¡person, move toward (home), walkway¿*, our method outperforms baselines significantly.

6.5.4 Ablation Analysis on Unseen Scenes

Place-based Feature Description. The hyper-parameter for PD is the level L, controlling when to decompose semantics in different places. Fig.6.10(a) and 6.10(c) show that the generalization capability of our model is improved when we allow the network observe the entire video at input level, and decompose semantics at feature level (after the 2nd conv blocks). However, if we extract place descriptions after a very late block (*e.g.*level 4 or 5), it fails to improve the model generalizability.

Distance-based Place Discretization. We study different strategies for determining PL_{DT} and the number of parts to discretize (k) per place. From our observations, including the

Figure 6.9: Qualitative examples: The predicted confidences of groundtruth actions using different methods. We use 3 frames to visualize a motion and orange ellipses to highlight moving agents.



(a) Decomposition Level L (b) Place Discretization (c) Topological Aggregation (d) Segmentation: GT v.s. Auto

Figure 6.10: Ablation Study of LIVR.

anchor place–*porch*, the six places in our dataset can be clustered into three categories with regard to the distance to camera: C1 includes only *porch*, which is usually the closest place to camera; C2 includes *lawn*, *walkway*, *driveway*, and actions occurring in those places usually require modeling the moving direction directly; C3 includes *sidewalk* and

street, which are usually far away from the house, and actions on them are not sensitive to directions (*e.g.*"*move along*"). We evaluate our method with two strategies to apply DD on: 1) all places belong to C2 and C3; 2) only places in C2. The results are shown in Fig.6.10(b). We observe that applying DD on C3 dose not help much, but if we only apply DD on places in C2, our method achieves the best performance. In terms of the number of discretized parts k, we evaluate k from 2 to 5 and observe from Fig.6.10(b) that the performance is robust when $k \geqslant 3$.

Topological Feature Aggregation. We evaluate different h values to determine the h-connected set and different strategies to construct and utilize the action-place mapping $\mathbf{T}$. The results are shown in Fig.6.10(c). We observe that Topo-Agg achieves its best performance when $h = 1$, *i.e.*for an action occurring in place P, we aggregate features extracted from place P and its directly connected places. In addition, we compare Topo-Agg to the naive fully connected inference layer (FC-Agg: 1 layer) and two fully-connected layers with 384 neurons each and a ReLU layer in between (FC-Agg: 2 layers). Unsurprisingly, we observe that the generalizability drops significantly with an extra fully-connected layer, which reflects overfitting. Our Topo-Agg outperforms both methods. We also conduct an experiment where we train a fully connected inference layer and only aggregate features based on topology at testing time ("Topo-Agg: 1-hop test only") and it shows worse performance.

LIVR with Automatically Generated Segmentation Maps. Although we believe it is desired to use human annotations to obtain segmentation maps, we developed an algorithm to automatically generate the segmentation maps to evaluate LIVR. As shown in

Fig.6.11, we first apply normalized cut [138] on the camera images to obtain super pixels (Fig.6.11 (b))[4]. Then, to further differentiate different places with similar appearance (*e.g.*walkway and street), we developed an algorithm to utilize the historical statistics obtained from previous videos (Fig.6.11 (d)) of a scene to generate heatmaps of some specific places[5] (Fig.6.11 (e)). Then, the two results are combined to obtain final segmentation maps ((Fig.6.11 (c))). Our method can generate reasonably good segmentation maps when compared to the groundtruth maps obtained manually (6.11 (f))). We evaluate LIVR using the imperfect maps and observe some performance degradation (around 10%), but LIVR still outperforms the baselines by a large margin (around 20%), which demonstrate the effectiveness of our method even if the segmentation maps are imperfect. We will leave the problem of automatically generating high-quality segmentation maps in home surveillance as a future work.

6.6 Conclusions and Future Directions

To improve the generalization of a deep network that learns from limited training scenes, we propose a layout-induced video representation which abstracts away low-level appearance variance but encodes the semantics, geometry and topology of scene layout. There are two possible directions of interesting future work: first, in this paper we focus

[4]We also tried deep learning based methods trained on segmentation datasets, but they perform poorly on our camera images.

[5]We utilize the patterns of moving objects to differentiate places. For example, street is a place where vehicles move along on it with limited scale changes (from the camera perspective), and walkway is a place where people with notably large scale changes walk along.

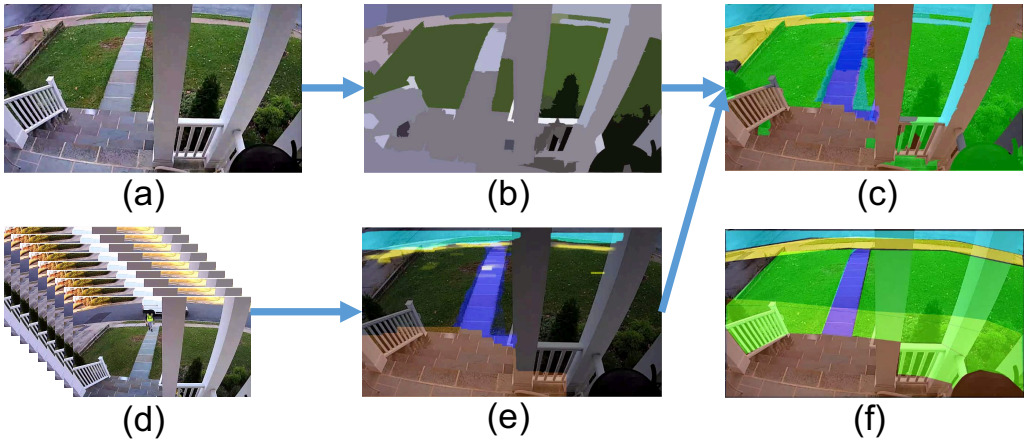


Figure 6.11: Process of Automatically Generating Segmentation Maps.

on manually annotated maps, but we may integrate the estimation of the semantic maps into the network architecture, which may require collecting more scenes for training. Second, we may extend the task from action classification with a closed action set to a more flexible compositional and structural prediction of arbitrary agents, places and actions and associate them to form agent-in-place actions, including zero-shot actions consisting of observed agents, places and actions.

Chapter 7: Conclusion

We investigated ways to improve the efficiency and generalization of visual recognition in five different works. These works are motivated by real world applications such as image classification, visual relationships detection, object detection, motion detection and action recognition. The works also study the fundamental redundancy of deep neuron networks, the way to incorporate statistical prior in an end-to-end learning framework, selective context modeling for object detection, spatial-temporal attention in an efficient motion detection pipeline and how to incorporate semantic segmentation into action recognition task respectively.

We show in our works that (1) to better reduce the redundancy of a deep neural network, we can estimate the neuron importance globally in a back propagation fashion. (2) to improve model generalization, besides collecting more data, we can utilize common sense and statistical prior of knowledge to help regularize the learning process of deep network. (3) Context is helpful in visual recognition task, but not all of them are informative. (4) Efficient 3D ConvNets can be used to speed up motion or action recognition in videos, and to improve the model's generalization capabilities on unseen scenes, we could incorporate scene layout information into an end-to-end neuron network.

Although we have witnessed the great improvement brought by deep network in

computer vision tasks, there are still many problems remaining unsolved for building efficient and general enough solutions. Interesting future direction may include understanding the structural properties of neuron network to further develop more efficient but still powerful network architectures; improving generalization of network by incorporating more knowledge or common sense into network training and inference; reduce the amount of annotated training data by weakly/self-supervised learning methods.

Bibliography

[1] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016.

[2] 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.

[3] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. Nisp: Pruning networks using neuron importance score propagation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. *CoRR*, abs/1707.09423, 2017.

[5] Ruichi Yu, Hongcheng Wang, and Larry S. Davis. Remotenet: Efficient relevant motion event detection for large-scale home surveillance videos. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[6] Ruichi Yu, Xi Chen, Vlad I. Morariu, and Larry S. Davis. The role of context selection in object detection. In *British Machine Vision Conference (BMVC)*, 2016.

[7] Ruichi Yu, Hongcheng Wang, Ang Li, Jingxiao Zheng, Vlad I. Morariu, and Larry S. Davis. Representing videos based on scene layouts for recognizing agent-in-place actions. *arXiv preprint arXiv:1711.05282*, 2018.

[8] Ang Li, Jin Sun, Joe Yue-Hei Ng, Ruichi Yu, Vlad I. Morariu, and Larry S. Davis. Generating holistic 3d scene abstractions for text-based image retrieval. *CoRR*, abs/1611.09392, 2016.

[9] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Dynamic zoom-in network for fast object detection in large images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S. Davis. Viton: An image-based virtual try-on network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Mingfei Gao, Ang Li, Ruichi Yu, Vlad I. Morariu, and Larry S. Davis. C-wsl: Count-guided weakly supervised localization. *European Conference on Computer Vision*, 2018.

[12] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'aurelio Ranzato, and Nando D. Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2148–2156. Curran Associates, Inc., 2013.

[13] Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI'11, pages 1237–1242, 2011.

[14] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[15] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

[16] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[17] Giorgio Roffo, Simone Melzi, and Marco Cristani. Infinite feature selection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4202–4210, 2015.

[18] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Intelligent signal processing*, pages 306–351. IEEE Press, 2001.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[20] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, June 2009.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1097–1105. Curran Associates, Inc., 2012.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[24] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 947–955. 2016.

[25] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shi. Compression of deep convolutional neural networks for fast and low power mobile applications. In *International Conference on Learning Representations (ICLR)*, 2016.

[26] Suraj Srinivas and R. Venkatesh Babu. Learning the architecture of deep neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 104.1–104.11, September 2016.

[27] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, MM'14, pages 675–678, New York, NY, USA, 2014. ACM.

[28] Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[29] Guodong Zhou, Min Zhang, Dong Hong, and Ji Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information.

[30] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *ACL*, pages 427–434, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[31] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *ACL*, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[32] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. Harnessing deep neural networks with logic rules. In *ACL, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[33] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P. Xing. Deep neural networks with massive learned knowledge. In *EMNLP, Austin, Texas, USA, November 1-4, 2016*, pages 1670–1679, 2016.

[34] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[35] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[36] Máté Pataki, Miklós Vajna, and Attila Csaba Marosi. Wikipedia as text. *ECRIM News*, Special theme: Big Data:48 – 48, 04/2012 2012.

[37] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *ACL Workshop on Vision and Language (VL15)*, Lisbon, Portugal, September 2015.

[38] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[39] Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. Modeling context between objects for referring expression understanding. In *European Conference on Computer Vision (ECCV)*, 2016.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[41] Bryan A. Plummer, Arun Mallya, Christopher M. Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. Phrase localization and visual relationship detection with comprehensive linguistic cues. *CoRR*, abs/1611.06641, 2016.

[42] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. 2011.

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[44] Yikang Li, Wanli Ouyang, and Xiaogang Wang. ViP-CNN: A Visual Phrase Reasoning Convolutional Neural Network for Visual Relationship Detection, February 2017.

[45] Xiaodan Liang, Lisa Lee, and Eric P. Xing. Deep Variation-structured Reinforcement Learning for Visual Relationship and Attribute Detection, March 2017.

[46] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[47] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[48] Santosh Kumar Divvala, Derek Hoiem, James Hays, Alexei A. Efros, and Martial Hebert. An empirical study of context in object detection. In *CVPR*, pages 1271–1278. IEEE Computer Society, 2009.

[49] Kevin P Murphy, Antonio Torralba, and William T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 1499–1506. MIT Press, 2004.

[50] Carolina Galleguillos, Andrew Rabinovich, and Serge Belongie. Object categorization using co-occurrence, location and appearance. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2008.

[51] Xi Chen, He He, and Larry S Davis. Object detection in 20 questions. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, 2016.

[52] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[53] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 702–709, 2012.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[55] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[56] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[57] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[59] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. *CoRR*, abs/1507.00448, 2015.

[60] Wenqing Chu and Deng Cai. Deep feature based contextual model for object detectionn. *CoRR*, abs/1604.04048, 2016.

[61] Jianan Li, Yunchao Wei, Xiaodan Liang, Jian Dong, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Attentive contexts for object detection. *CoRR*, abs/1603.07415, 2016.

[62] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[63] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, September 2010.

[64] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, New York, NY, USA, 2004. ACM.

[65] Matthew Ginsberg. Multivalued logics: A uniform approach to inference in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[66] V.D. Shet, J. Neumann, V. Ramesh, and L.S. Davis. Bilattice-based logical reasoning for human detection. *Computer Vision and Pattern Recognition, 2007*, pages 1–8, June 2007.

[67] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[68] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *1st Workshop on Consumer Depth Cameras for Computer Vision (ICCV workshop)*, November 2011.

[69] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013.

[70] Saurabh Gupta, Ross B. Girshick, Pablo Arbelaez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736, 2014.

[71] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In *Proc. IEEE Workshop on Change Detection (CDW-2014) at CVPR-2014*, pages 387–394, 2014.

[72] Andrews Sobral and Thierry Bouwmans. Bgs library: A library framework for algorithm?s evaluation in foreground/background segmentation. In *Background Modeling and Foreground Detection for Video Surveillance*. CRC Press, Taylor and Francis Group., 2014.

[73] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[75] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[76] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[77] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[78] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[79] Mahmudul Hasan and Amit K Roy-Chowdhury. Context aware active learning of activity recognition models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4543–4551, 2015.

[80] Minsi Wang, Bingbing Ni, and Xiaokang Yang. Recurrent modeling of interaction context for collective activity recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[81] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[82] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[83] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[84] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. ICPR '04, pages 28–31, Washington, DC, USA, 2004. IEEE Computer Society.

[85] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[86] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[87] Cewu Lu, Ranjay Krishna, Michael S. Bernstein, and Fei-Fei Li. Visual relationship detection with language priors. *CoRR*, abs/1608.00187, 2016.

[88] Bichen Wu, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[89] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1933–1941, 2016.

[90] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 568–576, Cambridge, MA, USA, 2014. MIT Press.

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

[92] C. Xu and J. J. Corso. Actor-action semantic segmentation with grouping-process models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[93] H. Kuhne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[94] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[95] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[96] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[97] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634. IEEE Computer Society, 2015.

[98] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702. IEEE Computer Society, 2015.

[99] Chih-Yao Ma, Min-Hung Chen, Zsolt Kira, and Ghassan AlRegib. TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *CoRR*, abs/1703.10667, 2017.

[100] Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 140–153, Berlin, Heidelberg, 2010. Springer-Verlag.

[101] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):221–231, January 2013.

[102] Ismail Haritaoglu, David Harwood, and Larry S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:809–830, 2000.

[103] Florent Fusier, Valéry Valentin, François Brémond, Monique Thonnat, Mark Borg, David Thirde, and James Ferryman. Video understanding for complex activity recognition. *Machine Vision and Applications*, 18(3):167–188, Aug 2007.

[104] Robert Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, and Osamu Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Carnegie Mellon University, Pittsburgh, PA, May 2000.

[105] Shifu Zhou, Wei Shen, Dan Zeng, Mei Fang, Yuanwang Wei, and Zhijiang Zhang. Spatial-temporal convolutional neural networks for anomaly detection and localization in crowded scenes. *Image Commun.*, 47(C):358–368, September 2016.

[106] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry S. Davis. Learning temporal regularity in video sequences. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[107] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. In Mark W. Jones Xianghua Xie and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 8.1–8.12. BMVA Press, September 2015.

[108] Ruichi Yu, Hongcheng Wang, and Larry Davis. Remotenet: Efficient relevant motion event detection for large-scale home surveillance videos. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2018.

[109] Ross Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[110] Mingfei Gao, Ang Li, Ruichi Yu, Vlad I. Morariu, and Larry S. Davis. C-wsl: Count-guided weakly supervised localization. *arXiv preprint arXiv:1711.05282*, 2017.

[111] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Dynamic zoom-in network for fast object detection in large images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[112] Li-Jia Li, Hao Su, Yongwhan Lim, and Li Fei-Fei. Object bank: An object-level image representation for high-level visual recognition. *Int. J. Comput. Vision*, 107(1):20–39, March 2014.

[113] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[114] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[115] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *IN ICCV*, pages 1458–1465, 2005.

[116] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[117] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition(CVPR*, 2009.

[118] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *IN: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN CLASSIFICATOIN*, 2010.

[119] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 143–156, Berlin, Heidelberg, 2010. Springer-Verlag.

[120] K E A van de Sande, J.R.R. Uijlings, T Gevers, and A.W.M. Smeulders. Segmentation as Selective Search for Object Recognition. In *ICCV*, 2011.

[121] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 346–361, Cham, 2014. Springer International Publishing.

[122] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 73–86, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[123] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Bejing, China, 22–24 Jun 2014. PMLR.

[124] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[125] Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[126] Ruichi Yu, Xi Chen, Vlad I. Morariu, and Larry S. Davis. The role of context selection in object detection. In *British Machine Vision Conference (BMVC)*, 2016.

[127] Ang Li, Vlad I. Morariu, and Larry S. Davis. Selective encoding for recognizing unreliably localized faces. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 3613–3621, 2015.

[128] Rui-Wei Zhao, Zuxuan Wu, Jianguo Li, and Yu-Gang Jiang. Learning semantic feature map for visual content recognition. In *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, pages 1291–1299, 2017.

[129] Lamberto Ballan, Francesco Castaldo, Alexandre Alahi, Francesco Palmieri, and Silvio Savarese. Knowledge transfer for scene-specific motion prediction. In *European Conference on Computer Vision (ECCV)*, 2016.

[130] James Hays and Alexei A. Efros. Scene completion using millions of photographs. *ACM Trans. Graph.*, 26(3), July 2007.

[131] Ce Liu, Jenny Yuen, and Antonio Torralba. *Nonparametric Scene Parsing via Label Transfer*, pages 207–236. Springer International Publishing, Cham, 2016.

[132] Joseph Tighe and Svetlana Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 352–365, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[133] Lamberto Ballan, Marco Bertini, Giuseppe Serra, and Alberto Del Bimbo. A data-driven approach for tag refinement and localization in web videos. *Computer Vision and Image Understanding*, 140:58–67, Nov. 2015.

[134] Xun Xu, Timothy M Hospedales, and Shaogang Gong. Discovery of shared semantic spaces for multiscene video query and summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1353–1367, 2017.

[135] Jenny Yuen and Antonio Torralba. A data-driven approach for event prediction. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 707–720, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[136] Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, ECCV'12, pages 201–214, Berlin, Heidelberg, 2012. Springer-Verlag.

[137] 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. *CoRR*, abs/1608.00859, 2016.

[138] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.