ABSTRACT

Title of dissertation: MODELING DEEP CONTEXT IN SPATIAL AND TEMPORAL DOMAIN

Xiyang Dai, Doctor of Philosophy, 2018

Dissertation directed by: Professor Larry S. Davis
Institute for Advanced Computer Studies
Department of Computer Science

Context has been one of the most important aspects in computer vision researches because it provides useful guidance to solve variant tasks in both spatial and temporal domain. As the recent rise of deep learning methods, deep networks have shown impressive performances on many computer vision tasks. Model deep context explicitly and implicitly in deep networks can further boost the effectiveness and efficiency of deep models.

In spatial domain, implicitly model context can be useful to learn discriminative texture representations. We present an effective deep fusion architecture to capture both the second order and first older statistics of texture features; Meanwhile, explicitly model context can also be important to challenging task such as fine-grained classification. We then present a deep multi-task network that explicitly captures geometry constraints by simultaneously conducting fine-grained classification and key-point localization.

In temporal domain, explicitly model context can be crucial to activity recognition and localization. We present a temporal context network to explicitly capture relative context around a proposal, which samples two temporal scales pair-wisely for precise temporal localization of human activities; Meanwhile, implicitly model context can lead to better

network architecture for video applications. We then present a temporal aggregation network that learns a deep hierarchical representation for capturing temporal consistency.

Finally, we conduct research on jointly modeling context in both spatial and temporal domain for human action understanding, which requires to predict where, when and what a human action happens in a crowd scene. We present a decoupled framework that has dedicated branches for spatial localization and temporal recognition. Contexts in spatial and temporal branches are modeled explicitly and fused together later to generate final predictions.

MODELING DEEP CONTEXT IN SPATIAL AND TEMPORAL DOMAIN

by

Xiyang Dai

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 Ramani Duraiswami
Professor Hector Corrada Bravo
Professor Tom Goldstein

Acknowledgments

I have been a sci-fi fan since I was a little kid, dreaming about the future of AI, robotics and spaceships, imaging how I could help change the world. This dream inspired me to do study in computer science and encouraged me to do research in computer vision. Step by step, I have reached the short conclusion of my PhD study, which is an important monument of my progress. Looking back, dream has been teared down piece by piece and realities taught me how to be mature. But one thing, never has been changed, is — I believe what I am doning, could help change the world, no matter how minor it is. This faith helps me pass the most difficult times of my PhD study.

I am not a person good at saying thanks, but I deeply appreciate all the people helping me, in my heart. Thanks to my parents, who have the foresights to let me study abroad, silently supporting me. Thanks to my wife, who has been untiringly accompanying and bearing me. Thanks to my friends, who provide a relaxing relief during work break. Thanks to all my lab-mates, who continuously surprise me and inspire me with new ideas. Thanks to all my co-authors, who help us learn from each other. Thanks to all my reviewers, who whip me to profession. Thanks to all my mentors during internships, who kindly share me with industry experiences, which I barely touch during academic study. Thanks to all my committee members, who provide professional suggestions and precious time to me.

Finally, my most sincere thanks to my advisor, Prof. Larry S. Davis, who enlightens my path to explore unknown. It has been my greatest honor to be one of your students.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Deep networks have shown impressive performance on many computer vision tasks. However, training such networks requires large amount of data and computing resources due to the data-driven nature. Context can be used as a guidance to accelerate such learning process. Our proposed works show that model deep context can further boost the effectiveness and efficiency of deep models. In this dissertation, we develop approaches to model deep context explicitly and implicitly in spatial (Chapter 3 and Chapter 4), temporal (Chapter 5 and Chapter 6) domain. Finally, we conduct study on modeling context in both spatial and temporal domain jointly (Chapter 7).

In the first work, we present a deep fusion model that implicitly capture deep context in spatial domain for texture recognition. Recently, deep convolutional neural networks (CNNs) have been used to learn discriminative texture representations. One of the most successful approaches is Bilinear CNN model that explicitly captures the second order statistics within deep features. However, these networks cut off the first order information flow in the deep network and make gradient back-propagation difficult. We propose an effective fusion architecture - FASON that combines second order information flow and first order information flow. Our method allows gradients to back-propagate through both flows freely and can be trained effectively. We then build a multi-level deep architecture to exploit the first and second order information within different convolutional layers. Experiments show that our method achieves improvements over state-of-the-art methods on several benchmark datasets.

In the second work, we present a deep multi-task model that explicitly capture deep context in spatial domain for fine-grained classification. Fine-grained classification of objects such as vehicles, natural objects and other classes is an important problem in visual recognition. It is a challenging task because small and localized differences between similar looking objects indicate the specific fine-grained label. At the same time, accurate classification needs to discount spurious changes in appearance caused by occlusions, partial views and proximity to other clutter objects in scenes. A key contributor to fine-grained recognition are discriminative parts and regions of objects. Past work has often attempted to solve the problems of classification and part localization separately resulting in complex models and ad-hoc algorithms, leading to low performance in accuracy and processing time. We propose a novel multi-task deep network architecture that jointly optimizes both localization of parts and fine-grained class labels by learning from training data. The localization and classification sub-networks share most of the weights, yet have dedicated convolutional layers to capture finer level class specific information. We design our model as memory and computational efficient so that can be easily embedded in mobile applications. We demonstrate the effectiveness of our approach through experiments that achieve a new state-of-the-art 93.1% performance on the Stanford Cars-196 dataset, with a significantly smaller multi-task network (30M parameters) and significantly faster testing speed (78 FPS) compared to recent published results.

In the third work, we present a deep model that explicitly capture deep context in temporal domain for action detection. We present a Temporal Context Network (TCN) for precise temporal localization of human activities. Similar to the Faster-RCNN architecture, proposals are placed at equal intervals in a video which span multiple temporal scales. We propose a novel representation for ranking these proposals. Since pooling features only inside a segment is not sufficient to predict activity boundaries, we construct a representation which explicitly captures context around a proposal for ranking it. For each temporal

segment inside a proposal, features are uniformly sampled at a pair of scales and are input to a temporal convolutional neural network for classification. After ranking proposals, non-maximum suppression is applied and classification is performed to obtain final detections. TCN outperforms state-of-the-art methods on the ActivityNet dataset and the THUMOS14 dataset.

In the fourth work, we present a novel deep architecture that implicitly capture deep context in temporal domain for video applications. We present TAN, a temporal aggregation network which decomposes 3D convolutions into spatial and temporal blocks of convolutions. By stacking spatial and temporal convolutions repeatedly, TAN forms a deep hierarchical representation for capturing spatio-temporal information in videos. Since we do not apply 3D convolutions in each layer but only apply temporal convolutions once after each downsampling layer in the network, we significantly reduce the model complexity. The use of dilated convolutions at different resolutions of the network helps in aggregating multi-scale spatio-temporal information efficiently. Experiments show that our model is well suited for dense multi-label action recognition, which is a challenging sub-topic of action recognition that requires predicting multiple action labels in each frame. We outperform state-of-the-art methods by 6% and 3% on the Charades and Multi-THUMOS dataset respectively.

In the fifth work, we propose approach on modeling spatial temporal human action understanding. It has been a hot research topic due to its vast application in video surveillance. Such task requires to accurately predict where, when and what a human action happens simultaneously, which makes it most challenging. We present a decoupled framework that has dedicated branches for spatial localization and temporal recognition. Contexts in spatial and temporal branches are modeled explicitly and fused together later to generate final predictions.

The remaining of this dissertation is organized as follows. We introduce the related

works in Chapter 2. In Chapter 3 to Chapter 7, we detailed describe all our works to model deep context in spatial and temporal domain. Chapter 8 concludes the whole dissertation.

Chapter 2: Related Work

2.1 Spatial Context Modeling

2.1.1 Texture Representations.

Texture descriptors have been well studied for decades. Classic methods include region co-variance [99], local binary patterns (LBP) [63] and other hand-craft descriptors. Robust texture representations, such as VLAD [40] and Fisher vector [79] combining with SIFT features [57] have been utilized to further improve performance on texture recognition tasks. Donahue et al.showed that deep networks can learn generic features from images that can be usefully applied to many computer vision tasks including texture recognition [20]. Zhou et al.fine-tuned such pre-trained models on specific texture and scene datasets end-to-end to achieve better performance [122]. Later, Cimpoi et al.combined improved Fisher vector encoding [67] with deep features, which further improves performance on various texture recognition datasets [12]. They further collected a large dataset of texture images, which is now considered as the state-of-the-art benchmark for texture recognition. Most recently, Lin et al.modeled second order pooling in a deep learning framework and proposed the bilinear network [98]. Later, they applied their framework to texture recognition [54]. Gao et al.proposed a compact bilinear network that utilizes random Maclaurin and tensor sketching to reduce the dimensionality of bilinear representations but preserves the discriminative power at the same time [25].

Researchers have realized the importance of fusing both first order and second order statistics in feature learning. Hong et al.proposed a second order statistics based region descriptor, named "sigma set" [36]. It was first constructed through Cholesky decomposition on the covariance matrix and then fused with the first order mean vector. Later, Doretto et al.fused central moments, central moment invariants, radial moments and region covariance together in a compact representation [21]. This representation is invariant to scale, translation, rotation and illumination changes. Recently, Li et al.proposed a feature encoding method called locality-constrained affine subspace coding (LASC) that captures both first order and second order information [52]. They showed improvement when fusing first order information into their framework.

2.1.2 Fine-grained Classification

Fine-grained classification, using discriminative parts, has been the subject of active research. [19] built a human-in-loop classification game that revealed the importance of discriminative parts for the fine-grained task. [1] first demonstrated the usefulness of part based one-versus-one discriminative features. [47] tried to detect parts using several unsupervised part detectors. Later, [48] improved such work by proposing a co-segmentation based method that can generate discriminative parts without using part annotations. Most recently, [109] showed promising improvements in the use of parts by including geometric constraints between triplets of discriminative parts.

Meanwhile, fine-grained classification performance has seen improvements owing to developments in deep network architectures. [75] showed that simply using off-the-self CNN features could lead to significant improvement over traditional hand-crafted features. [20] combined deep convolutional activation features with a deformable parts model to further improve fine-grained performance. More recently, part information has been used

more directly in the training process for deep networks. [3] applied deep convolutional networks to batches of image patches that were located and normalized by pose. Similarly, [119] borrowed the idea of region-based CNN and fine-tuned it on object parts. These two methods showed the great potential of merging part information into deep network models. However, they required ground truth part annotations during testing for good performance. To solve the part localization problem, [82] proposed a multi-proposal consensus network to predict part locations. Recently, there is a trend toward trying to solve part localization and fine-grained classification simultaneously. [53] proposed a valve linkage function that connected part localization, alignment and classification together. However, this approach attempted only to discriminate the heads of birds from their bodies. [55] proposed a network containing two streams of appearance models combined with a bilinear pooling layer, arguing that manually defined parts were sub-optimal and the bilinear pooling approach could explore optimal parts implicitly. Although the approach benefited from data-driven end-to-end learning of the deep network's parameters, the degree to which parts were discovered by the network was hard to interpret. [49], demonstrated the benefits of training with larger amounts of data (and a larger number of classes) to a fine-grained recognition task, even in the presence of noisy training labels. We consider our work separate from that. The major purpose of this paper is to illustrate the benefits of a compact network that can efficiently perform fine-grained classification and part localization simultaneously. Most recently, [38] proposed a framework that solved localized network and fine-grained classification together, which is most similar to this paper. However, this approach solved part localization and fine-grained classification separately and used a region based feature pooling around the parts to merge the discriminative information from parts. Although it showed the effectiveness of localizing and utilizing of the parts, the framework does not support end-to-end learning.

2.2 Temporal Context Modeling

2.2.1 Action recognition.

Action recognition is one of the core components of video understanding. In early works, 3D motion templates [2], or features such as SIFT-3D [80], Action MACH [77] were used for representing temporal information for action recognition. Later, the introduction of dense trajectories [102] and improved dense trajectories [103] provided a significant boost in performance for feature based pipelines.

After the early success of deep learning [51, 89, 94] on image classification, early deep learning based video recognition methods focused on utilizing deep features. Karpathy et al. [46] evaluated different fusion methods for deep features. Wang et al. [105] pooled deep-learned features based on trajectory constraints. A two stream network which learns temporal dynamics on stacked optical flow was proposed by Simonyan et al. [90] and an analysis of different ways of fusing information from these two networks was provided by Feichtenhofer et al. [23]. Since these two stream networks were trained on very short snippets, Wang et al. [107] further improved the two-stream network by introducing a sparse temporal sampling strategy and video-level supervision to enable efficient learning on whole videos.

Meanwhile, researchers explored novel designs of deep architectures specifically for video classification tasks. Tran et al. [97] proposed a deep 3-dimensional convolutional neural network (C3D) to learn spatial and temporal features simultaneously. More recently, Carreira et al. [8] collected a large scale dataset of actions, which enabled them to train a 3D convolution version of inception architecture. This network showed very impressive performance on the UCF-101 dataset.

We describe a novel deep architecture specifically for video applications. However, un-

like previous work, we do not simply apply 3D convolutions on existing ImageNet based models. Instead, we decompose 3D convolutions into separate spatial and temporal convolutions. By stacking these two types of convolutions repeatedly, we effectively capture appearance and spatio-temporal motion patterns. Since we have a single layer of temporal convolutions after each downsampling layer, we significantly reduce the filters needed for performing these tasks.

2.2.2 Multi-label prediction.

Multi-label dense action recognition is a considerably harder task than action recognition as it requires labeling frames with all the actions occurring in them. Yeung [112] first proposed this task by densely labeling every frame within a popular sports action dataset [42]. They further applied standard techniques such as LSTM, two-steam networks and created a very strong baseline. Sigurdsson et al. [88] collected another multi-label dataset by crowd-souring everyday activities at home. For recognizing these actions, Girdhar et al. [28] proposed ActionVLAD, that aggregates appearance and motion features from two-stream networks. Sigurdsson et al. [86] used a fully-connected temporal CRF model for reasoning over various aspects of activities. Dave et al. [18] employed RNNs to sequentially make top-down predictions and later then corrected them by bottom-up observations. Most recently, Sigurdsson et al. [87] performed a detailed analysis on what kinds of information are needed to achieve substantial gains for activity understanding among objects, verbs, intent, and sequential reasoning.

2.2.3 Temporal action localization.

Unlike action recognition where we predict a label per video, action detection requires predicting temporal boundaries of an action in an untrimmed video. [115] constructed a

pyramid of score distribution features (PSDF) as a representation for ranking segments of a video in a dense trajectories based pipeline. However, for large datasets, these methods require significant computational resources to extract features and build the feature representation after features are extracted. Because deep learning based methods provide better accuracy with much less computation, hand-crafted features have become less popular.

For object detection in images, proposals are a critical elements for obtaining efficient and accurate detections [76, 78]. Motivated by this approach, Jain et al. [39] introduced action proposals which extends object proposals to videos. For spatio-temporal localization of actions, multiple methods use spatio-temporal region proposals [27, 30, 64, 114]. However, these methods are typically applied to datasets containing short videos, and hence the major focus has been on spatial localization rather than temporal localization. Moreover, spatio-temporal localization requires training data containing frame level bounding box annotations. For many applications, simply labeling the action boundaries in the video is sufficient, which is a significantly less cumbersome annotation task.

Very recently, studies focusing on temporal segments which contain human actions have been introduced [5, 58, 60, 84, 92]. Similar to grouping techniques for retrieving object proposals, Heilbron et al. [5] used a sparse dictionary to encode discriminative information for a set of action classes. Mettes et al. [60] introduced a fragment hierarchy based on semantic visual similarity of contiguous frames by hierarchical clustering, which was later used to efficiently encode temporal segments in unseen videos. In [92], a multi-stream RNN was employed along with tracking to generate frame level predictions to which simple grouping was applied at multiple detection thresholds for obtaining detections.

Methods using category-independent classifiers to obtain many segments in a long video are more closely related to our approach. For example, Shou et al. [84] exploit three segment-based 3D ConvNets: a proposal network for identifying candidate clips that may contain actions, a classification network for learning a classification model and a lo-

calization network for fine-tuning the learned classification network to localize each action instance. Escorcia et al. [22] introduce Deep Action Proposals (DAPs) and use a LSTM to encode information in a fixed clip (512 frames) of a video. After encoding information in the video clip, the LSTM scores K (64) predefined start and end positions in that clip. The start and end positions are selected based on statistics of the video dataset. We show that our method performs better than global representations like LSTMs which create a single feature representation for all scales in a video for localization of activities. In contemporary work, Shou et al. [83] proposed a convolutional-de-convolutional (CDC) network by combing temporal upsampling and spatial downsampling for activity detection. Such an architecture helps in precise localization of activity boundaries. We show that the activity proposals generated by our method can further improve CDC's performance. Zhao et al. [121] decomposed their model into separate classifiers for classifying actions and determining completeness. Meanwhile, due to success in object detection for localizing objects in images, recent works started applying similar ideas to videos. Xu et al. [111] present a R-CNN like action detection framework from C3D features. Dai et al. [16] argued for the importance of sampling at two temporal scales to capture temporal context in a faster R-CNN architecture.

Chapter 3: FASON: First and Second Order Information Fusion Network for Texture Recognition

3.1 Introduction

Features from pre-trained deep models have been successfully utilized for texture recognition [12, 13, 20, 31]. By combining off-the-shelf deep features with second order encoding methods such as Locally Aggregated Descriptors (VLAD) [40] and Fisher Vectors [79], such approaches significantly improve texture recognition performance. However, these approaches require multiple stages of processing including feature extraction, encoding and SVM training, which do not take advantage of end-to-end optimization in deep learning.

Recently, researchers have designed network architectures specifically for texture recognition. One of the most successful approaches is the deep bilinear model proposed by Lin et al.[54, 98]. They model the second order statistics of convolutional features within a deep network with a bilinear pooling layer that enables end-to-end training and achieved state-of-the-art performance on benchmark datasets. However, it discards the first order information within the convolutional features, which is known to be useful to capture spatial characteristic of texture [35] and illumination [71]. The first order information is also known to be essential for back-propagation based training [61]. Hence, previous deep bilinear models neglect the potential of first order statistics in convolutional features and make training difficult.

We propose a novel deep network architecture that fuses first order and second order information. We first extend the bilinear network to combine first order information into the learning process by designing a leaking shortcut which enables the first order statistics to pass through and combine with bilinear features. Our architecture enables more effective learning by end-to-end training compared to the original deep bilinear model. This allows us to extend our fusion architecture to combine features from multiple convolution layers, which captures different style and content information. This multiple level fusion architecture further improves recognition performance. Our experiments show the proposed fusion architecture achieves consistent improvements over state-of-the-art methods on several benchmark datasets across different tasks such as texture recognition, indoor scene recognition and fine-grained object classification.

The contribution of our work is two fold:

- We design a deep fusion architecture that effectively combines second order information (calculated from a bilinear model) and first order information (preserved through our leaking shortcut) in an end-to-end deep network. To the best of our knowledge, our architecture is the first proposed method to fuse such information directly in a deep network.

- We extend our fusion architecture to take advantage of the multiple features from different convolution layers.

3.2 Deep Bilinear model

Deep bilinear models have shown promising results on several computer vision tasks including fine-grained image classification and texture classification [25, 54, 98]. Although bilinear models have only recently been integrated in a deep network with end-to-end training, the basic formulation of bilinear models has a long history in texture description. Given

an input image I, we extract its deep convolutional features $F \in \mathbb{R}^{w \times h \times ch}$ from a CNN and calculate the bilinear feature $B \in \mathbb{R}^{ch \times ch}$ as:

$$B(F) = \sum_{i=1}^{w} \sum_{j=1}^{h} F_{i,j} F_{i,j}^{T} \tag{3.1}$$

This formulation is related to orderless texture descriptors such as VLAD , Fisher Vectors and region covariance [98]. It has been shown to be effective to capture the second order statistics of texture features. The diagonal entries of the output bilinear matrix represent the variances within each feature channel, while the off-diagonal entries represent the correlations between different feature channels. The descriptor lies in a Riemannian Manifold which makes quantization difficult and distance measurement between feature vectors non-trivial [66]. Therefore, the bilinear feature is usually passed through a mapping function with signed square root and l_2 normalization that projects it to Euclidean space [67]:

$$\phi(x) = \frac{sign(x)\sqrt{|x|}}{\|sign(x)\sqrt{|x|}\|_2} \tag{3.2}$$

3.2.1 Dimension reduction

The high dimensionality of bilinear features makes end-to-end training difficult in a neural network. Following [25], we use a technique called tensor sketching to reduce the $d = ch \times ch$ bilinear output to a feature in a lower dimension c. Tensor sketching [9, 70], which is known to preserve pairwise inner products, estimates the frequency of all elements in a vector. It is a random projection technique to reduce feature dimensionality using multiple random mapping vectors defined by simple independent hash functions.

Given two randomly sampled mapping vectors $h \in \mathbb{N}^d$ where each entry is uniformly drawn from $\{1, 2, \cdots, c\}$, and $s \in \{+1, -1\}^d$ where each entry is filled with either $+1$ or

-1 with equal probability, the sketch function is defined as:

$$\Psi(x,s,h) = [C_1, C_2, \cdots, C_c] \tag{3.3}$$

where

$$C_j = \sum_{i:h(i)=j} s(i) \cdot x(i) \tag{3.4}$$

To reduce the dimensionality of bilinear features, the $ch \times ch$ size bilinear feature is first vectorized to $x \in \mathbb{R}^d$ where $d = ch \times ch$ and further projected to a lower c-dimensional vector $\mathscr{E} \in \mathbb{R}^c$ by:

$$\mathscr{E}(x) = \mathscr{F}^{-1}(\mathscr{F}(\Psi(x,s,h)) \circ \mathscr{F}(\Psi(x,s',h'))) \tag{3.5}$$

where s' and h' are drawn similarly to s and h, $\circ$ operator represents element-wise multiplication, and $\mathscr{F}$ represents the Fast Fourier Transformation. We reduce the bilinear representation to $c = 4096$ dimensions in all experiments.

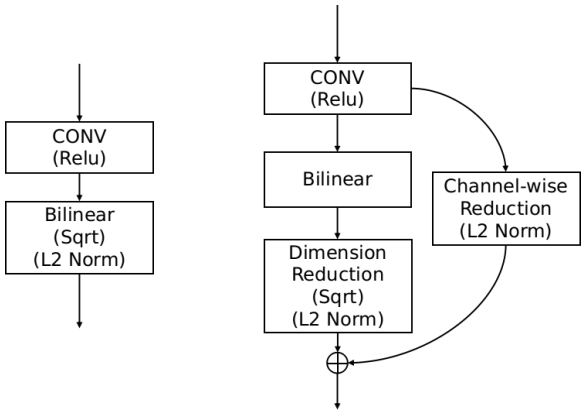


Figure 3.1: The core building block of our first and second order information fusion architecture compared to original bilinear model.

3.2.2 First order information fusion by gradient leaking

Now we introduce our core building block of the first order and second order information fusion. Although bilinear models exploit second order information from deep features well, they often suffer from the problem of vanishing gradients when gradient flow back-propagates through them, which makes it difficult to learn such models in an end-to-end training process. Therefore, recent work usually places the bilinear layer after the last convolutional layer to minimize this problem.

Inspired by the recent success of deep residual networks [34], we design a shortcut connection that passes through the first order information and combines with the second order information generated from the bilinear layer, as shown in Figure 3.1. Assuming we generate the deep feature F from the previous convolutional layer, instead of using the bilinear feature $B(F)$ directly, we combine it with a leaking function $M(F)$ that encodes first order information. Since the bilinear layer essentially captures the covariance between each feature channel, we define our leaking function as:

$$\mathscr{M}(F) = \frac{1}{wh} \sum_{i=1}^{w} \sum_{j=1}^{h} F_{i,j} \tag{3.6}$$

to provide the mean of each feature channel. This is analogous to global average pooling for a convolutional feature map.

The first order information is then combined with the second order information as follows:

$$\hat{\mathscr{B}}(F) = \mathscr{E}(\mathscr{B}(F)) \oplus \mathscr{M}(F) \tag{3.7}$$

where $\oplus$ represents the vector concatenation operation.

With the proposed formulation, the first order information can be exploited for classification, and the training can be stabilized as the architecture provides a direct pathway for

the gradients to lower layers from the leak.

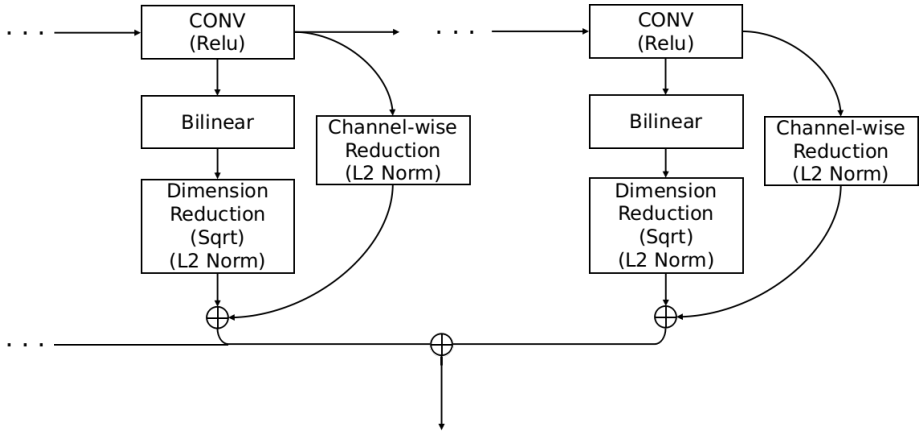


Figure 3.2: The illustration of how we accumulate first and second order information from multiple convolutional layers.

3.2.3 First and second order fusion with multiple levels of convolutional features

One benefit of our fusion framework is that we can fuse more convolutional features into bilinear layers and conduct an effective end-to-end training as shown in Figure 3.2. Given arbitrary convolutional feature maps $F_1, F_2, \cdots F_i$, we can fuse them together simply by:

$$\hat{\mathscr{B}}(F) = \biguplus_i \mathscr{E}(\mathscr{B}(F_i)) \oplus \mathscr{M}(F_i) \tag{3.8}$$

where $\biguplus_i$ indicates concatenating features generated from different convolutional layers. In this way, we force the network to utilize the first and second order information across multiple convolutional feature maps, which generally captures different style and content information.

We investigate two major network architectures: a single fusion at `conv5` level (equiv-

alent to features generated from conv5_4 of VGG-19 network) and a multiple fusion at conv4, conv5 layers (equivalent to features generated from conv4_4 and conv5_4 of VGG-19 network). For fair comparison, we also conduct experiments using typical bilinear networks without fusion on these same two setups. The detailed configurations of our architectures are shown in Figure 3.3.

3.3 Experiments

We evaluate the effectiveness and performance of our architecture and compare with state-of-the-art methods on several datasets. We also adopt the artistic style transfer technique of [26] to visualize the qualitative improvements of our architecture.

Datasets. We evaluate our architecture on four benchmark datasets: DTD (Describable Texture) dataset [12], KTH-T2b (KTH-TISP2-b) dataset [7], MIT-Indoor (MIT indoor scene) dataset [73] and Stanford Car196 [50] dataset.

The DTD dataset is considered the most widely used benchmark for texture recognition. It contains 47 texture categories with a total of 5640 images. All images are ”in the wild”, from the web image rather than collected in a controlled setting. This dataset is challenging due to its realistic nature and large intra-category variation. We report the 10-fold average accuracy as in [54].

The KTH-T2b dataset contains 11 material categories with 4752 images total. Images in each category are captured from 4 physical, planar samples under controlled scale, pose and illumination. In our experiments, we follow the evaluation setup in [54] and report the 4-fold average accuracy.

The MIT-Indoor dataset contains 67 indoor scene categories with at least 100 images per category and 15620 images total. The images can be considered as weakly structured

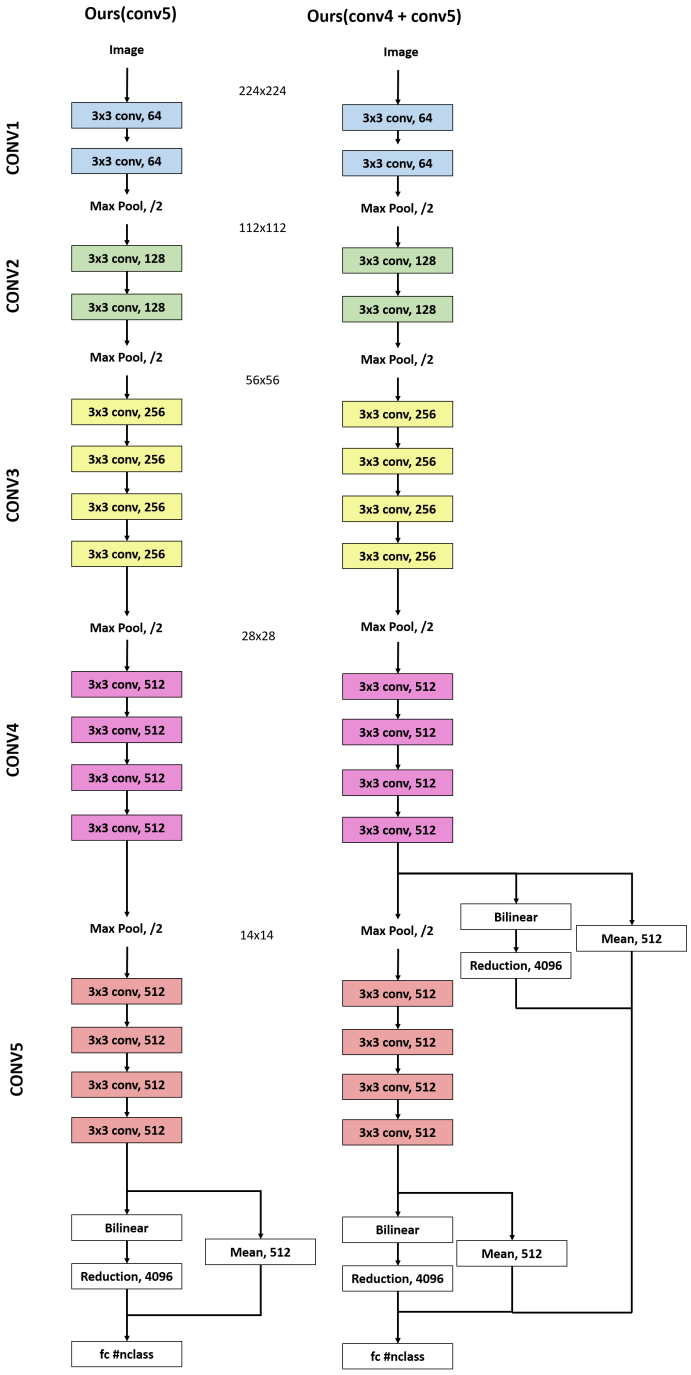


Figure 3.3: The detailed configurations of our first and second order information fusion architectures. Colored boxes represent convolutional layers with "3×3" filter size and number of output channels. Notice that we group convolution layers by their spatial dimensions, such as `conv4` and `conv5`. This allows us to maintain the same notation across different network architectures. For all convolutional layers, we use padding size 1 and use ReLU as activation layer. For all the max pooling layers, we use kernel size 2 and stride size 2 without padding.

and orderless textures, which provide a reasonable evaluation of the generalization power of our texture models. We use the training and testing split provided with the dataset, for a total of 5360 images for training and 1340 images for testing.

The Stanford Car196 dataset contains 196 different fine-grained car classes at the level of make, model and year with 16185 images total. The images are further split into a 50-50 split with 8144 training images and 8041 testing images. This dataset is considered as one the most challenging fine-grained classification dataset. We evaluate on this dataset to further test the generalization of our model.

Implementation details. We implement our architecture in a customized version of Caffe [41]. We adopt a two-stage training process to speed up training. We first fix all layers except the last fully connected layer for classification in the network to form a convex learning problem, and then relax the network to fine-tune all the layers with a constant small learning rate and high momentum. In detail, in the first training step, we use a fixed learning policy with learning rate starting at 1, weight decay at 5×10^{-6} and run for up to 50 epochs. In the second training stage, we fine-tune all the layers using a fixed learning policy with learning rate of 0.001 and weight decay at 5×10^{-4} for another 100 epochs. We did not use data augmentation and dropout for fair comparison to previous work. Incorporating these techniques may further improve the results. For experiments conducted on DTD, KTH-T2b and Stanford Car196 datasets, we use a 224×224 input size, while for the MIT-Indoor dataset we evaluate two setups with 224×224 and 448×448 input size.

3.3.1 Effectiveness of fusion

We first evaluate the effectiveness of our fusion architecture by comparing two networks with and without first order information fusion on single (conv5) and multiple (conv4+conv5) convolutional layers. For fair comparison, we use the same learning hyperparameters and

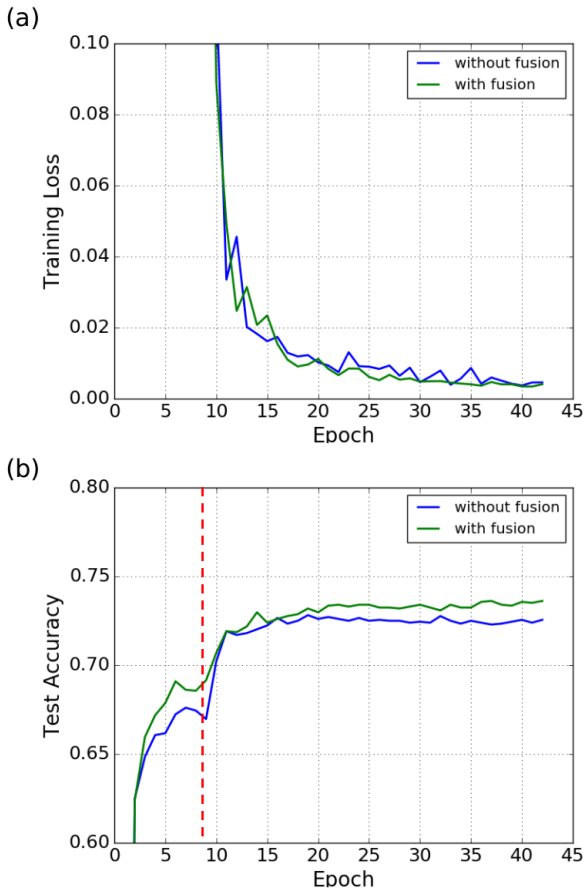


Figure 3.4: Comparison of learning curves on bilinear model using single level of convolutional feature (conv5) with and without our first order information fusion on DTD dataset.

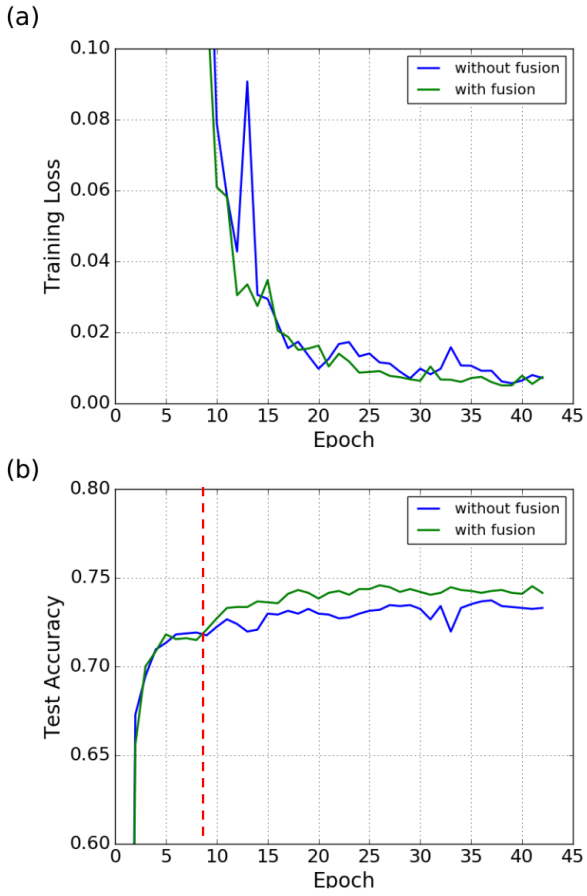


Figure 3.5: Comparison of learning curves on bilinear model using two levels of convolutional features (conv4+conv5) with and without our first order information fusion on DTD dataset.

training/testing split on the DTD dataset. For the first 8 epochs, we only train the last layer with a fixed learning rate 1. We then fine-tune all the layers with a fixed learning of 0.001. Figure 3.4 and 3.5 show both the training loss and testing accuracy. The red vertical lines highlight the point at which we switch from learning only the last layer to learning all layers. As shown by the plots of testing accuracy (Figure 3.4b and Figure3.5b), our architecture with first order information fusion clearly outperforms the bilinear network without fusion in both training stages. The training loss plots (Figure 3.4a and Figure3.5a) show that our architecture can be trained more smoothly. This is more observable in the multiple convolutional layer setup (Figure3.5a). In both setups, our experiments demonstrate the effectiveness of our approach.

We also evaluate the effectiveness of our fusion architecture on different deep networks such as VGG-16 and VGG-19. Table 3.1 shows the performance of our architecture applied to different models on the DTD dataset. Our fusion architectures gives consistent improvements over a baseline bilinear network. The performance further boosts with our multiple layer fusion when two level of convolutional layers conv4 and conv5 are combined. Further combining lower layers in the network did not improve performance. Combining the improvements from first and second order information fusion with multi-layer feature fusion, we obtain a 2% improvement from a strong bilinear CNN baseline for both VGG-16 and VGG-19.

3.3.2 Comparison with state-of-the-arts

Texture recognition. We compare the performance of our fusion architecture with several state-of-the-art methods such as [12, 13, 54] on the DTD dataset. All the results are reported based on an input size 224×224 for fair comparisons. The methods annotated with * in the table indicates that multiple scales of inputs are used instead of a single input

	VGG-16	VGG-19
conv5	72.45	72.82
conv5+fusion	73.09	73.62
Improvements	**+0.64**	**+0.80**
conv5+conv4	72.87	73.31
conv5+conv4+fusion	74.47	74.57
Improvements	**+1.60**	**+1.26**

Table 3.1: Effectiveness comparison across different network architecture on one training and testing split on DTD dataset. Our models gives consistent improvements from standard Bilinear CNN.

Method	Accuracy
DeCAF + IFV[12]	66.7 ± 0.9
FV-CNN [54]	67.8 ± 0.9
B-CNN [54]	69.6 ± 0.7
FASON (conv5)	$\mathbf{72.3 \pm 0.6}$
FASON (conv4+conv5)	$\mathbf{72.9 \pm 0.7}$
FC-VGG* [13]	62.9 ± 0.8
FV-VGG* [13]	72.3 ± 1.0
FC+FV-VGG* [13]	74.7 ± 1.0
FC-SIFT FC+FV-VGG* [13]	75.5 ± 0.8

Table 3.2: Comparison with state-of-the-art methods on the DTD dataset with 224×224 input size.

size. Table 3.2 shows that our method achieves the best performance among all single-feature methods. In particular, our best model gives 3% improvement over the B-CNN baseline. Our method also outperforms FC-VGG and FV-VGG which use multiple input scales. We also report the fusion results from previous work that use multiple features. Our method is competitive with such methods but requires only one single network.

We also evaluate the performance of our fusion architecture on KTH-T2b dataset with several state-of-the-art methods such as [12, 13, 54, 95]. Similar to the DTD dataset, all reported results are based on input size 224×224, and * in the table also represents multiple

Method	Accuracy
TREE [95]	66.3
DeCAF [12]	70.7 ± 1.6
DeCAF + IFV [12]	76.2 ± 3.1
FV-CNN [54]	74.8 ± 2.6
B-CNN [54]	75.1 ± 2.8
FASON (conv5)	$\mathbf{76.5 \pm 2.3}$
FASON (conv4+conv5)	$\mathbf{76.4 \pm 1.5}$
FC-VGG* [13]	75.4 ± 1.5
FV-VGG* [13]	81.8 ± 2.5
FC+FV-VGG* [13]	81.1 ± 2.4
FC-SIFT FC+FV-VGG* [13]	81.5 ± 2.0

Table 3.3: Comparison with state-of-the-art methods on the KTH-T2b dataset with 224×224 input size. * denotes results obtained from multiple scales.

scales are used. As shown in Table 3.3, our method also gains a 1.4% boost compared to the B-CNN baseline. Our best model is only behind FV-VGG which uses deep features calculated at three different scales. Our method with multiple convolutional layers FASON (conv4+conv5) performances slightly worse than a single layer model FASON (conv5) in this dataset. We believe this is caused by the smaller amount of training data provided in the KTH-T2b dataset, which leads to over-fitting. Our method is again competitive to previous approaches.

Indoor scene classification. In addition to the pure texture datasets, we evaluate our models on the MIT-Indoor dataset and compare with state-of-the-art methods such as [13, 52, 54, 122]. We evaluate our model with both 224×224 input size and 448×448 input size. Table 3.4 shows that our method is superior to previous methods and our best performance is 0.7% better than previous state of the art method FV-CNN, which used multiple scales. Again, our method largely outperforms the B-CNN baseline with about a 4% improvement consistently over input size 224 and 448. Meanwhile, our models also take

Method	Input Size		
	224	448	ms
LASC [52]	63.4	–	–
PLACE [122]	70.8	–	–
FC-VGG [13]	–	–	67.6
FV-VGG [13]	–	–	81.0
FV-CNN [54]	70.1	78.2	78.5
B-CNN [54]	72.8	77.6	79.0
FASON (conv5)	**76.0**	**80.8**	–
FASON (conv4+conv5)	**76.8**	**81.7**	–

Table 3.4: Comparison with state-of-the-art methods on the MIT-Indoor dataset with different input sizes. *ms* denotes results obtained from multiple scales.

advantage of larger input size, which is consistent with previous results. Our method with multiple convolutional layers FASON(conv4+conv5) further improves performance over a single convolution layer FASON(conv5).

Fine-grained classification. We further evaluate our models on the Stanford Car196 dataset and compare with popular state-of-the-art methods. Following standard evaluation protocol, we use the provided bounding box during training and testing. All images are cropped around the bounding boxes and then resized to 224×224. We compare with state-of-the-art methods [47, 48, 109, 110] with the same input size. As shown in Table 3.5, our models improve significantly over the bilinear model [98] when using the same VGG-19 architecture. Meanwhile, our models result in a 1.2% and 1.5% improvement over the best bilinear models mixing different architectures reported in [98]. Our best model is comparable to [48], which utilizes part information of cars to boost performance. Overall, our fusion models have shown promising generalization ability in fine-grained classification task and are competitive to state-of-the-art methods on the Stanford Car dataset.

Method	Accuracy
CNN [47]	70.5
ELLF [47]	73.9
CNN Finetuned [110]	83.1
FT-HAR-CNN [110]	86.3
BoT [109]	92.5
Parts [48]	92.8
FV-CNN [98]	85.7
B-CNN [98]	90.6
B-CNN (VGG16 + VGG19) [98]	91.3
FASON (conv5)	**92.5**
FASON (conv4+conv5)	**92.8**

Table 3.5: Comparison with state-of-the-art methods on the Stanford Car196 dataset using 224×224 input size.

3.3.3 Artistic Style Transfer

Artistic style transfer is a popular technique introduced in [26] that transfers the style from one artistic image to another image. Because this technique utilizes a bilinear representation to compute the style loss, performing style transfer provides an intuitive way to visualize and understand what is learned in the networks. To generate visually plausible style transfer results, the networks need to learn a good representation for both content and style. Since the style of an image is closely related to its texture, we apply our learned networks, which have learned good texture representations, to the task of artistic style transfer.

We follow the suggested settings described in [26] and use `conv4_2` layer for content loss with weight 1 and use `conv1_1`, `conv2_1`, `conv3_1`, `conv4_1` and `conv5_1` layers for style loss with weight 0.2. We run L-BFGS for 512 iterations in all experiments.

We compare our fusion architectures with the standard VGG network (using the weights learned form the DTD dataset for classification task) on different combinations of content and style images in Figure 3.6. The red boxes highlights the major differences in style. With

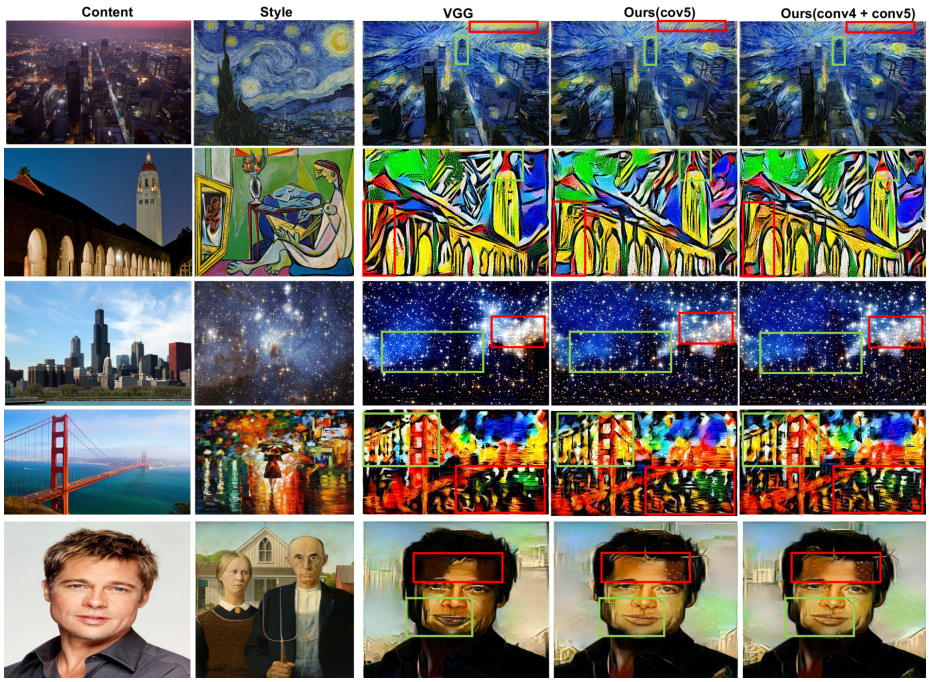


Figure 3.6: Comparison of style transfer results using different models. The green box highlights the content differences between images. The red box highlight the style difference between images.

the side-by-side comparison, our model shows richer styles (the cloud and wall appear more stylish in the images) and more accurate content (the contours of building and tower appear to be better preserved) in the generated images. These qualitative results suggest that with our architecture can preserve style and content information more effectively. Furthermore, our multi-layer fusion architecture is even better than our single-layer fusion architecture.

3.4 Conclusion

We presented a novel architecture that aggregates first and second order information within a deep network. Experiments show that our fusion architecture consistently improves over the standard bilinear networks. We additionally propose an architecture combining information from the multiple levels of convolutional layers, which further improves

overall performance. Our network can be trained end-to-end effectively. We achieve state-of-the-art performance for a single network on several benchmark datasets. In addition, the better learned texture representation from our network is shown qualitatively by the improved artistic style transfer results.

Chapter 4: Efficient Fine-grained Classification and Part Localization Using One Compact Network

4.1 Introduction

Fine-grained classification amongst various classes of man-made and natural objects is currently an active area of research because of numerous practical applications. For instance, recognizing make-models of vehicles can improve tracking of vehicles across non-overlapping camera views, or searching for a given vehicle in a forensic investigation using captured video from multiple locales. Similarly, recognizing types of birds can enable their counting in an area or understanding patterns of migration. While related to generic image classification, fine-grained classification is a significantly distinct problem because it must focus on small, localized intra-class differences (e.g., make-models of vehicles, birds species) instead of inter-class differences that are often easier to account for (e.g., vehicles vs. birds). Accounting for differences between similar looking objects within a class to recognize specific instances, while being robust to spurious changes caused by occlusions and overlap with surrounding objects makes this task very challenging.

An important aspect of the solution to the fine-grained classification is locating discriminative parts of objects. If a learning network can be designed to focus on parts such as the position of car headlights, then it will learn representations that distinguish makes and model of cars based upon different shapes of headlights. Also, it may enable us to solve more challenging tasks, such as ”locate a Honda Civic 2006 with a dent on the left

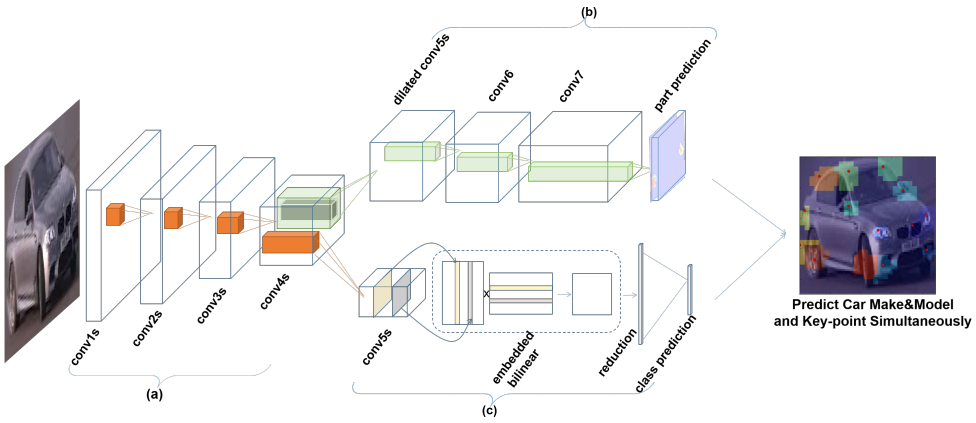


Figure 4.1: Overview of our deep multi-task learning architecture applied for joint part localization and fine-grained classification. The localization network (b) and fine-grained classification network (c) share the first four levels of convolution (a) and also have task-specific dedicated convolution layers. The whole architecture is trained end-to-end. Our multi-task network can simultaneously predict fine-grained class and part locations.

front door", since we have an understanding of parts semantics and their relative geometry. However, it is challenging to solve part localization and fine-grained classification simultaneously because the former is geometric in nature while the latter is a labeling problem. Previous work either solved these two problems separately or fused them together in complicated frameworks.

We describe a multi-task deep learning approach to simultaneously solve part localization and fine-grained classification, and demonstrate the mutual benefits of the multi-task approach. Our network architecture (see Figure 4.1) and training procedure are less complex than competing methods while achieving better results with significant speedup and with smaller memory footprint. In order to reduce the difference in the nature of localization and classification problems, We model part localization as a multi-class classification problem by representing parts as a label mask that annotates part locations, as opposed to the traditional regression of geometric coordinates of parts. This narrows the gap between part localization and fine-grained classification. It allows us to share a significant num-

ber of network parameters between the fine-grained classification and the part localization tasks and enables us to take advantage of pre-trained models as a starting point for optimization. A set of fully-convolutional layers produce the part label mask while a mixture of convolutional and embedded bilinear pooling layers are used for fine-grained classification. The part localization loss and fine-grained classification loss are combined, enabling an end-to-end multi-task data-driven training of all network parameters. Our contributions are three-fold:

- We propose a novel multi-task deep learning and fusion architecture that have both shared and dedicated convolutional layers for simultaneous part labeling and make-model classification.
- The accuracy of our approach is competitive to state-of-the-art methods on both car and bird domains.
- Our network architecture is more compact (30 M parameters) and runs much faster (78 FPS) than competitors, enabling real-time, mobile applications.

4.2 Representation Learning with Deep Multi-task Architecture

Multi-task learning has proven to be effective in several computer vision tasks. Deep networks with end-to-end training are well-suited for multiple tasks because they learn generic representations in early layers prior to specialization in later stages of the network. Recent work [56, 74, 119] has shown promising results by attaching multiple shallow and task-specific output layers to the final fully connected layer, and training the network to minimize combined losses for each task output. These approaches mainly change the last fully connected layers dedicated to specific tasks while not learning any low or mid-level representations that could influence the accuracy of multiple tasks. Our approach differs

from previous vanilla multi-task models in: (i) its design of common low and mid-level representation layers for multiple tasks, (ii) application of a careful analysis to select layers that are suitable for sharing between the two tasks, and (iii) designing task-specific deep sub-networks to capture task-specific representations. The end-to-end training for multiple tasks ensures that the task-specific representations benefit from task-specific tuning, while the shared representations are jointly influenced by the needs of both tasks. Such architecture enables joint learning of part localization and fine-grained classification effectively and efficiently.

Choosing a Base Model. Our architecture is based upon a VGG-16 network pre-trained on ImageNet [89]. We choose VGG due to several reasons. First, VGG mainly utilizes 3 by 3 convolutional kernels that can be efficiently computed. Meanwhile, the the layers of a VGG network captured low and mid-level representations are easily interpreted. Since we explicitly rely on learning common representations, VGG serves as an appropriate based model for our fusion architecture. With portability in mind, we skip state-of-the-art Inception and ResNet models on purpose, although such models may further improve the performance.

Determining Multi-task Structure. Previous approaches [56, 74, 119] implement multi-task learning by sharing all network parameters up until the final fully-connected (FC) layers; new FC layers are added for each task. Given the representational power of a single FC layer, this strategy has limited effectiveness. Our multi-task architecture uses deep sub-networks for different tasks, sharing earlier convolutional layers. Specifically, our part localization network and our fine-grained classification network architectures (described above) use the same *architecture* (VGG-16) in their first five convolutional layers. We aim to fuse the two networks together such that they use the same *weights* for a number of shared layers; if we share only the earliest layers, we may render ineffective gradient flows from task-specific sub-networks to the fused layers. If we share too many layers,

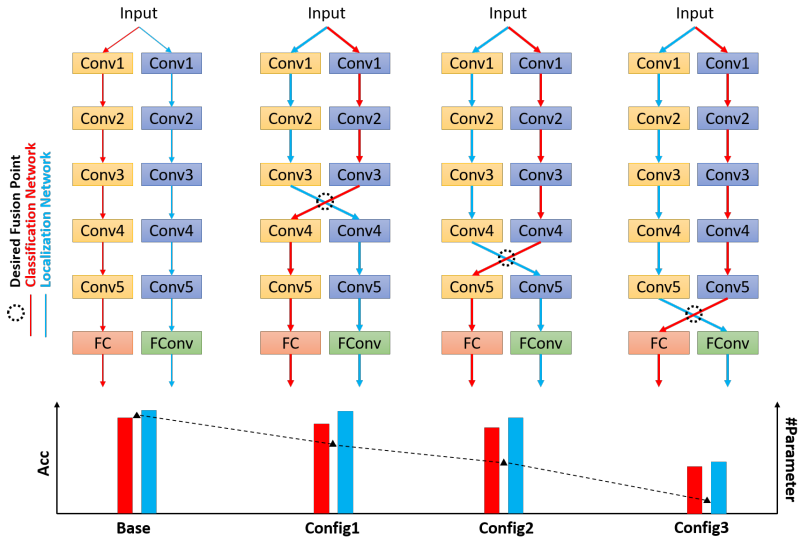


Figure 4.2: The experimental setups to determine which layers can be shared between tasks. Blue means layers are for localization. Red are trained for classification. We switch different weights between two network in different configurations, and retrain the later layers. This leads us to check the probability of fusion at desired point

we may degrade performance of the later task-specific sub-networks. To find the appropriate number of layers to share, we conducted a series of experiments. We first of all trained our localization network and classification networks separately to serve as a baseline (shown in Figure 4.2 as "Base") for measuring performance on the two tasks. Next, we switch the weights of the first three, four and five convolutional layers (shown in Figure 4.2 as "Config1, Config2, Config3"), and retrain the latter stages of the networks (this experiment is possible owing to the common *architecture* of the first layers of the two task networks) . Swapping weights like this, and then measuring task performance, allows us to establish which weights can be shared across tasks. For example, we found the performance of both task-specific networks drops significantly when we aggressively switch all convolution layers ("Config3"). The performance of both tasks with weights switched as in "Config2" indicated that features learned by the first four convolutional layers in both task-specific networks can be applied effectively to the other task, so our final architecture

(a) Shared layers

	1	2	3	4	5	6	7
layer	2 × conv	max	2 × conv	max	3 × conv	max	3 × conv
filter-stride-padding	3-1-1	2-2-0	3-1-1	2-2-0	3-1-1	2-2-0	3-1-1
#channel	64	64	128	128	256	256	512
activation	relu	idn	relu	idn	relu	idn	relu
size	224×224	112×112	112×112	56×56	56×56	28×28	28×28

(b) Dedicated localization layers

	8a	9a	10a	11a	12a	13a	14a	15a
layer	max	3 × dconv	max	avg	dconv	conv	conv	fc
filter-stride-padding	3-1-1	3-2-1	3-1-1	3-1-1	3-12-1	1-1-0	1-1-0	-
#channel	512	512	512	512	512	1024	#part	1
dilation	-	2	-	-	12	-	-	-
activation	idn	relu	idn	idn	relu	relu	relu	softmax
size	28×28	28×28	28×28	28×28	28×28	28×28	28×28	28×28

(c) Dedicated classification layers

	8b	9b	10b	11b
layer	max	3 × conv	embedding	fc
filter-stride-padding	2-2-0	3-1-1	-	-
#channel	512	512	1	1
activation	idn	relu	idn	softmax
size	14×14	14×14	8192	#class

Table 4.1: The detail configuration of our final multi-task architecture. The attributes of the column "conv", "max", "avg", "dconv", "embedding", "fc" represent "convolution", "max pooling", "average pooling", "dilated convolution", "feature embedding" and "fully connected layer".

shares these weights. The whole parameter setup of our deep multi-task architecture is described in Table 4.1.

Multi-task training. We jointly train our multi-task architecture in an end-to-end fashion using a typical multi-task fusion loss:

$$\mathscr{L}_{fuse} = \mathscr{L}_{loc} + \lambda \mathscr{L}_{cls}\,, \tag{4.1}$$

where λ is a weight factor to control the influence of each task during joint training. We determine an appropriate value in our experiments below. Our architecture (figure 4.1) shares layers between the two tasks, the detailed parameter setup is shown in Table 4.1. We describe the detailed designs of dedicated layers for each tasks in the following sections.

4.2.1 Localize Parts and Key-points with Pseudo-masking

Key-point (and, equivalently, part) localization has been widely studied for the purpose of pose estimation [68, 69, 96] and has largely been solved by learning a regression layer at the end of a deep network to extract the x, y coordinates of trained key-points. These papers have demonstrated that the regression task is sufficiently different from the image classification task that networks must be trained from scratch, rather than fine-tuned from a pre-trained network; this not only increases the amount of training data required, but also extends training time. Motivated by our desire to share layers between the localization task and the fine-grained classification task, and inspired by recent successes in semantic segmentation, rather than model part localization as a regression problem, we instead model it as a multi-class part classification problem. This decision allows us to fine-tune from a pre-trained classification network and enables weight-sharing with the fine-grained classification network. Our experiments show that this design decision also provides excellent part localization performance.

Architecture. Our parts localization approach is based upon mask generation; to support this we modify the VGG-16 architecture to be fully convolutional in the latter stages such that our output mask has dimensions 28×28. Specifically, we keep the first four convolution layers unchanged, except for dropping spatial pooling after conv4. We modify conv5 to use dilated (or atrous) convolutions, also without downsampling. In addition, we change the fully connected layers into fully convolutional layers with kernels in spatial size of 1×1. These modification allow us to reuse the same pretrained VGG-16 network weights, but output a 28×28 spatial part localization mask instead - similar to semantic segmentation, this mask is a labeled image, with pixels marked as background, or as containing a particular model part, as appropriate. The detailed configuration can be viewed in Table 4.1(b).

Learning. Given a specific part $p_c \in P = \{p_1, p_2, \cdots, p_K\}$, where c is the part class (such as "Front left light" of car) of total K classes ($K+1$ is the background class label), with normalized spatial coordinate $x \in [0,1], y \in [0,1]$, we want our localization network to generate a $m \times m$ spatial map S that predicts $S_{u,v} = c$, with $u = \lfloor x \cdot m \rfloor$ and $v = \lfloor y \cdot m \rfloor$, where $\lfloor x \rfloor$ is the truncating operator to an integer from a real number. However, considering the the area of part locations on this spatial map will be significant smaller $k \ll m^2$, which causes the learning process highly imbalanced (background vs. key-points ratio will be $\frac{m^2-k}{k} \to 1$), we apply a pseudo-masking around ground-truth part locations to make the learning easier. We define our pseudo masking strategy as

$$M_{i,j} = \begin{cases} K+1, \text{if} \quad \min dist((i,j),(x_c,y_c)) \geq t \\ \underset{c}{\text{argmin}}\, dist((i,j),(x_c,y_c)), \quad \text{otherwise} \end{cases}, \tag{4.2}$$

where $dist(\cdot)$ is a function that measures the distance; t is a trade-off to control the background/key-point ratio, we use $t = 0.1m$.

Our loss function is:

$$\mathscr{L}_{loc} = -\sum_{i=1}^{m}\sum_{j=1}^{m} \log \frac{f(i,j,M_{i,j})}{\sum_{c=1}^{K+1} f(i,j,c)}, \tag{4.3}$$

where $f(\cdot)$ represents the network. The loss includes the background class $K+1$.

Inference. Although our localization network predicts a pseudo part map, we can still recover the accurate key-point coordinate by exploring the probability maps underneath. Differing from the approach of [38], we don't need to set up a threshold to decide the existence of specific part location,since it is already handled by our pseudo masking strategy. Given a $m \times m$ prediction map S and a $m \times m \times (K+1)$ probability map *Prob* extracted from last fully convolutional layer of the part-localization network, the coordinate i_c, j_c of

a part c can be inferred by:

$$(i_c, j_c) = \underset{i,j}{\operatorname{argmax}} \left\{ Prob_{i,j,c} \cdot \mathbf{1}_{S_{i,j}=c} \right\}, \tag{4.4}$$

where $\mathbf{1}_a$ is the indicator function which is 1 if condition a is true and 0 otherwise. Our localization network design has multiple advantages:

- Our localization network shares the same amount of weights as VGG-16, hence can be fine-tuned using existing models
- Our network requires a small 224×224 input size , but yet generates a large part prediction mask
- Most importantly, we model the localization task as a classification problem, which enables straightforward fusion with the fine-grained classification task

4.2.2 Fine-grained Classification with Feature Embedding

We now describe the detailed implementations for our dedicated fine-grained classification layers.

Embedded bilinear pooling. Bilinear pooling has been proven effective to represent the feature variations from multiple streams of features. In a departure from previous work [55] that utilizes two different networks to conduct bilinear pooling, we demonstrate the effectiveness of using embedded bilinear pooling within a single network. Given a $w \times h \times ch$ shaped feature map F generated from a network, embedded bilinear pooling can be calculated as:

$$\mathsf{E} = \sum_{i=1}^{w} \sum_{j=1}^{h} \mathbf{F}_{i,j} \cdot \mathbf{F}_{i,j}^{\top}, \tag{4.5}$$

where $\mathbf{F}_{i,j}$ is a ch-dimensional column vector. This form is closely related to region covariance [99], that captures second order statistics between features, and can improve the

classification performance.

Dimensionality reduction. Embedded bilinear pooling reduces the training parameters from $w \times h \times ch \times l$ to $ch \times ch \times l$, where l is the number of hidden units of fully connected layer for prediction. However, this is still a large number that can overwhelm the whole training process, and may also lead to inefficient utilization of learned weights. We use compact bilinear pooling [24, 25], a projection method that further reduces the number of dimensions in our feature vector while preserving desirable properties of the feature. Given mapping vectors $h \in \mathbb{N}^d$ where each entry is uniformly sampled from $\{1, 2, \cdots, c\}$, and $s \in \{+1, -1\}^d$ where each entry is sampled with either $+1$ or -1 with equal probability, the sketch function is defined as:

$$\Psi(x, s, h) = [C_1, C_2, \cdots, C_c] \tag{4.6}$$

where

$$C_j = \sum_{i:h(i)=j} s(i) \cdot x(i) \tag{4.7}$$

To reduce the dimensionality of bilinear features, the $ch \times ch$ size bilinear feature E is first vectorized to $x \in \mathbb{R}^d$ where $d = ch \times ch$ and further projected to a lower c-dimensional vector $\hat{\mathsf{E}} \in \mathbb{R}^c$ by:

$$\hat{\mathsf{E}} = \mathscr{F}^{-1}(\mathscr{F}(\Psi(x, s, h)) \circ \mathscr{F}(\Psi(x, s', h'))) \tag{4.8}$$

where s' and h' are drawn similarly to s and h, $\circ$ operator represents element-wise multiplication, and $\mathscr{F}$ represents the Fast Fourier Transformation. The result of the tensor sketching process is a lower-dimensional version of E, $\hat{\mathsf{E}}$; the number of dimensions in E and $\hat{\mathsf{E}}$ used in our experiments are detailed in Table 4.1(c).

Classification loss. Finally, the reduced features $\hat{\mathsf{E}}$ can be mapped to our C fine-grained

Method	Setup	Input Resolution	#Parameters	Speed	Accuracy
Deep-LAC[53]	Parts	227	–	–	80.3
Part-RCNN[120]	BBox + Parts	227	60M	<1 FPS	76.4
Parts[48]	BBox	224	135M	<1 FPS	82.8
Bilinear[D,M][55]	BBox	448	70M	8 FPS	85.1
Stacked-CNN[38]	BBox + Parts	448	115M	20 FPS	76.7
Ours	**BBox + Parts**	**224**	**30M**	**78 FPS**	**84.3**

Table 4.2: Comparison with state-of-art methods on efficiency. The number of parameters estimation is calculated as Caffe weight size.

Parts	throat	beak	crown	forehead	right eye	nape	left eye	back
Stacked-CNN[38]	0.908	0.894	0.894	0.885	0.861	0.857	0.850	0.807
Ours	**0.963**	**0.952**	**0.950**	**0.960**	**0.939**	**0.937**	**0.943**	**0.867**
Parts	breast	belly	right leg	tail	left leg	right wing	left wing	Overall
Stacked-CNN[38]	0.799	0.794	0.775	0.760	0.750	0.678	0.670	0.866
Ours	**0.877**	**0.858**	**0.752**	**0.841**	**0.740**	**0.775**	**0.755**	**0.874**

Table 4.3: APK comparison with state-of-art methods on the CUB_200_2011.

classes using a small fully connected layer $fc(\cdot)$, trained using multinomial logistic loss:

$$\mathscr{L}_{cls} = -\sum_{i=1}^{C} \log \frac{fc(\hat{\mathsf{E}}, i)}{\sum_{c=1}^{C} fc(\hat{\mathsf{E}}, c)} \,. \tag{4.9}$$

Our classification network design has multiple advantages:

- We greatly reduce the number of parameters by replacing fully connected layers with an embedded bilinear layer.
- We are able to explore the second order information between feature maps through our embedded bilinear layer.
- We further reduce number of parameters required by introducing a random mapping technique.

4.3 Experiments

We first compare the performance of our method on the fine-grained classification task versus other leading methods considering both the performance and efficiency (speed and memory usage). Next, we demonstrate the effectiveness of part localization. We do an ablation study to illustrate that multi-task architecture improves classification performance at the end.

Dataset. We evaluate our approach on two standard fine-grained benchmarks. The Stanford Cars-196 [50] dataset contains 196 classes of car categories described by make, model and year, and has a total of 16185 images. This dataset is challenging due to: the large variation of car model, pose, and color; and often minor differences between models. We use the provided car bounding boxes during training and testing. This dataset does not provide information about parts, hence we manually annotated 30 images per class with 18 parts, such as "front right light", "rear left bumper" and so on. Our second dataset is Caltech-UCSD Birds (CUB-200-2011) [101], which contains 200 bird spices with 11788 images captured in the wild. Each image is annotated with a bounding box and 15 body parts such as "head", "breast", etc. This dataset is used as a cross-domain reference that shows our approach is easily applied to tasks from quite different domains.

Implementation. We implement our proposed multi-task network using a customized Caffe [41] package. Each input image is cropped to the object's bounding box and then resized to 224×224. We adopt a 3-step process to speed up the training process: 1) we freeze the weights of the part localization sub-network and fine-tune the classification network (including layers shared with the localization task); 2) we freeze the classification network weights and fine-tune the localization network (including the shared weights); 3) We fine-tune the whole network together with small learning rate and high momentum. The training approach is a gradual specialization of the base VGG-16 weights by incre-

mentally adapting the network to new tasks. Step 1 adapts the classification network to the new image domain. Step 2, assisted by adapation of the shared layers in step 1, adapts the part-localization sub-network to domain-specific parts. Step 3 tunes the entire network via the multi-task loss, enabling task-to-task feedback via the shared layers.

4.3.1 Efficiency and Performance

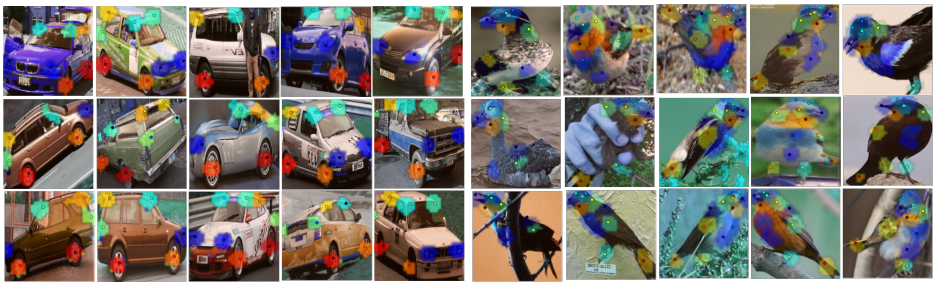
We compare our method with several recent alternative methods [38, 48, 53, 55, 120] that aimed to improve fine-grained classification problem by solving part localization. Each method is evaluated with speed and memory usage on a few key factors: input resolution, setup, number of training parameters, testing speed and accuracy, as shown in Table 4.2. All the experiments were conducted on a single Titan X (Maxwell). Our implementation runs significantly faster than all other methods, achieving about four times speed up compared to the previous fastest method. The efficiency gain mainly comes from the benefits of effective network fusion. Meanwhile, our classification performance is still competitive to state-of-the-art methods.

We further evaluate our method on Stanford Car196 dataset with the state-of-art methods, such as [47, 48, 50, 55, 104, 109, 110] on the Cars-196 dataset. Table 4.4 shows results; our multi-task approach achieves 93.1% top-1 accuracy and is better compared to the previous best [48] with a 0.3% improvement. Considering the minor improvement published recently, we think our improvement is noticeable. In addition, our approach is much more computationally efficient and solve part localization at the same time.

We also evaluate our part localization performance using the APK (average precision of key points) metric. This metric considers a predicted key-point (part location) to be correct if it lies in a radius of $\alpha\times$(image size) around the ground-truth location. We use $\alpha = 0.1$ following [38] to compare the results on CUB_200_2011. As shown in Table 4.3,

Method	Accuracy
BB-3D-G[50]	67.6
LLC[104]	69.5
ELLF[47]	73.9
CNN Finetuned[110]	83.1
FT-HAR-CNN[110]	86.3
Bilinear[D,M] [55]	91.7
BoT[109]	92.5
Parts[48]	92.8
Ours	**93.1**

Table 4.4: Comparison with state-of-art methods on the Stanford Car-196.



(a) (b)

Figure 4.3: The visualization selected results of key-point localizations on both Stanford Car-196 (a) and CUB-200-2011 (b). Solid dots represent the key-point location we predict. The transparent mask presents the part map we predict.

the localization performance is significantly better than the previous approach [38] in all part categories except ”right leg” and ”left leg”. Qualitative part localization results are shown in Figure 4.3. Our approach is capable of precisely pinpointing parts across a range of pose and aspect variations for both birds and cars.

4.3.2 Ablation Study

We first analyze the influence of the parameter λ that controls the balance between part localization and classification loss during training, and then illustrate the positive influence of multi-task training on the classification task. Figure 4.4 shows the evolution of test loss during training for both classification and part localization, with $\lambda = \{0.1, 0.2, 0.5, 1\}$

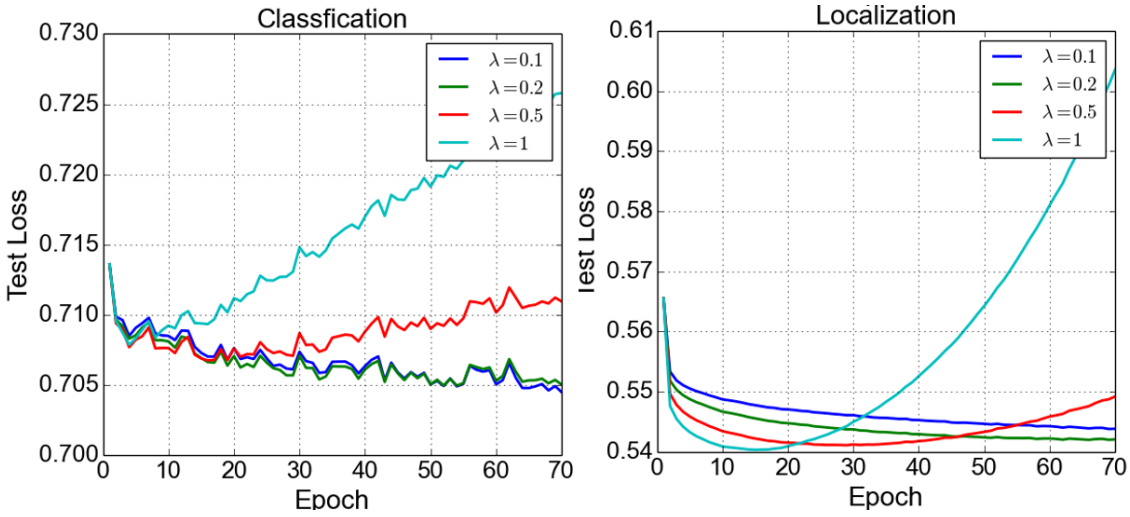


Figure 4.4: The influence of different λ value chosen on the multi-task training.

and using a small fixed learning rate over 70 epochs. Our experiments indicate that as λ increases, the gradient flow from the part localization sub-network overwhelms the training of the classification sub-network. Our best performance is with $\lambda = 0.2$.

We now evaluate the effectiveness of our multi-task architecture. We use a fine-tuned VGG-16/ImageNet as a baseline model, which archives a reasonable performance on both the CUB_200_2011 and Stanford Cars-196 datasets. In order to compare the performance before and after multi-task training, we disable the localization sub-network during training to form a classification standalone result. As shown in Table 4.5, our multi-task architecture performs better than the baseline and standalone-trained model on both datasets. This demonstrates the effectiveness of our multi-task training.

Base	Class	Part	Stanford Car196	CUB_200_2011
✓			89.1	80.0
✓	✓		92.7	83.9
✓	✓	✓	93.1	84.3

Table 4.5: Evaluating the effectiveness of the multi-task training on datasets .

4.4 Conclusion

In this paper, we propose a compact multi-task architecture that simultaneously performs part localization and fine-grained classification. We are able to fuse the localization network and classification network effectively and efficiently. Experiments demonstrate that our approach is general, being competitive on both the Stanford Cars-196 and the Cub-200-2011 birds datasets. Furthermore, our proposed network is both significantly smaller and faster than previous state-of-the-art methods, which makes real-time mobile applications possible.

With the success of such compact multi-task architecture, our future work will mainly focus on applying this architecture on car specific tasks, such as car re-identification and searching. Enhancement of the techniques presented in this paper will help solve multiple problems such as crowd-sourcing car re-identification problem with wearable devices and specific instance recognition problems using features such as a ”dent” in a specific location.

Chapter 5: Temporal Context Network for Activity Localization in Videos

5.1 Introduction

Recognizing actions and activities in videos is a long studied problem in computer vision [2, 4, 33]. An action is defined as a short duration movement such as jumping, throwing, kicking. In contrast, activities are more complex. An activity has a beginning, which is triggered by an action or an event, which involves multiple actions, and an end, which involves another action or an event. For example, an activity like "assembling a furniture" could start with unpacking boxes, continue by putting different parts together and end when the furniture is ready. Since videos can be arbitrarily long, they may contain multiple activities and therefore, temporal localization is needed. Detecting human activities in videos has several applications in content based video retrieval for web search engines, reducing the effort required to browse through lengthy videos, monitoring suspicious activity in video surveillance etc. While localizing objects in images is an extensively studied problem, localizing activities has received less attention. This is primarily because performing localization in videos is computationally expensive [22] and well annotated large datasets [6] were unavailable until recently.

Current object detection pipelines have three major components - proposal generation, object classification and bounding box refinement [76]. In [22, 84] this pipeline was adopted for deep learning based action detection as well. LSTM is used to embed a long video into a single feature vector which is then used to score different segment proposals

in the video [22]. While a LSTM is effective for capturing local context in a video [92], learning to predict the start and end positions for all activity segments using the hidden state of a LSTM is challenging. In fact, in our experiments we show that even a pre-defined set of proposals at multiple scales obtains better recall than the temporal segments predicted by a LSTM on the ActivityNet dataset.

In [84], a ranker was learned on multiple segments of a video based on overlap with ground truth segments. However, a feature representation which does not integrate information from a larger temporal scale than a proposal lacks sufficient information to predict whether a proposal is a good candidate or not. For example, in Figure 5.1, the red and green solid segments are two proposals which are both completely included within an activity. While the red segment is a good candidate, the green is not. So, although a single scale representation for a segment captures sufficient information for recognition, it is inadequate for detection. To capture information for predicting activity boundaries, we propose to explicitly sample features both at the scale of the proposal and also at a higher scale while ranking proposals. We experimentally demonstrate that this has significant impact on performance when ranking temporal activity proposals.

By placing proposals at equal intervals in a video which span multiple temporal scales, we construct a set of proposals which are then ranked using features sampled from a pair of scales. A temporal convolution network is applied over these features to learn background and foreground probabilities. The top ranked proposals are then input to a classification network which assigns individual class probabilities to each segment proposal.

5.2 Approach

Given a video $\mathscr{V}$, consisting of T frames, TCN generates a ranked list of segments $s_1, s_2, ..., s_N$, each associated with a score. Each segment s_j is a tuple t_b, t_e, where t_b and t_e

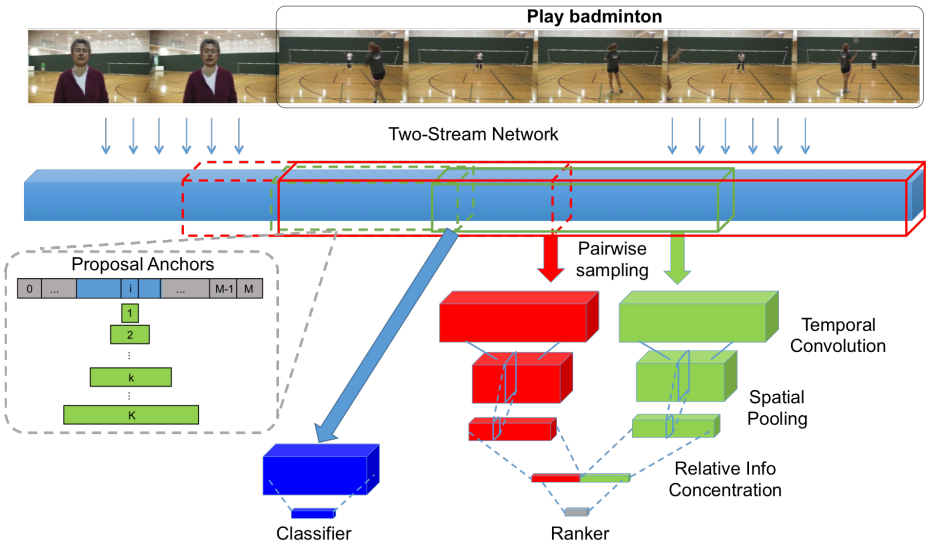


Figure 5.1: Given a video, a two stream network is used to extract features. A pair-wise sampling layer samples features at two different resolutions to construct the feature representation for a proposal. This pairwise sampling helps to obtain a better proposal ranking. A typical sliding window approach (Green line box) can miss the context boundary information when it lies inside the activity. However, the proposed pairwise sampling with a larger context window (Red line box) will capture such information and yield better proposal ranking. These pair-wise features are then input to a ranker which selects proposals for classification. The green boxes on the left represent K different proposals which are placed uniformly in a video.

denote the beginning and end of a segment. For each frame, we compute a D dimensional feature vector representation which is generated using a deep neural network. An overview of our method is shown in Figure 5.2.

5.2.1 Proposal Generation

Our goal in this step is to use a small number of proposals to obtain high recall. First, we employ a temporal sliding window of a fixed length of L frames with 50% overlap. Suppose each video $\mathscr{V}$ has M window positions. For each window at position i ($i \in [0, M]$), its duration is specified as a tuple (b_i, e_i), where b_i and e_i denote the beginning and end of a segment. We then, generate K proposal segments (at K different scales) at each position i. For $k \in [1, K]$, the segments are denoted by (b_i^k, e_i^k). Also, the duration of each segment, L_k, increases as a power of two, i.e $L_{k+1} = 2L_k$. This allows us to cover all candidate activity locations that are likely to contain activities of interests, and we refer them as activity proposals, $P = \{(b_i^k, e_i^k)\}_{i=0,k=1}^{M,K}$. Figure 5.1 illustrates temporal proposal generation. When a proposal segment meets the boundary of a video, we use zero-padding.

5.2.2 Context Feature Representation

We next construct a feature representation for ranking proposals. We use all the features $\mathscr{F} = \{f_1, f_2, ..., f_m\}$ of the untrimmed video as a feature representation for the video. For the k^{th} proposal at window position i ($P_{i,k}$), we uniformly sample from $\mathscr{F}$ to obtain a D dimensional feature representation $Z_{i,k} = \{z_1, z_2, ..., z_n\}$. Here, n is the number of features which are sampled from each segment. To capture temporal context, we again uniformly sample features from $\mathscr{F}$, but this time, from $P_{i,k+1}$ — the proposal at the next scale and centered at the same scale. Note that we do not perform average or max-pooling but instead sample a fixed number of frames regardless of the duration of $P_{i,k}$.

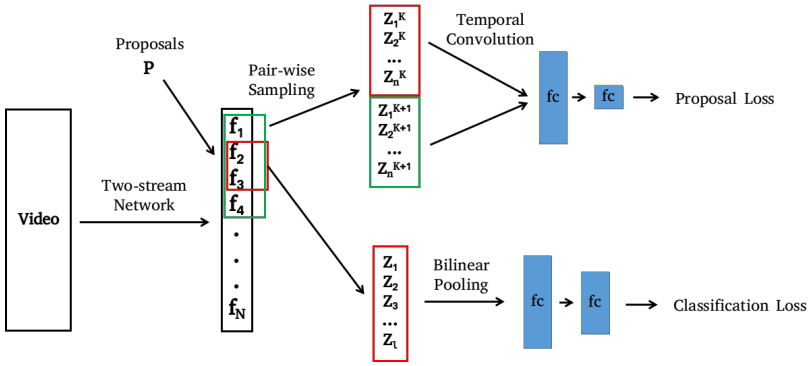


Figure 5.2: Temporal Context Network applies a two stream CNN on a video for obtaining an intermediate feature representation.

Logically, a proposal can fall into one of four categories:

- It is disjoint from a ground-truth interval and therefore, the next scale's (larger) label is irrelevant
- It includes a ground-truth interval and the next-scale has partial overlap with that ground truth interval.
- It is included in a ground-truth interval and the next level has significant overlap with the background (i.e., it is larger than the ground truth interval).
- It is included in a ground-truth interval and so is the next level.

A representation which only considers features inside a proposal would not consider the last two cases. Hence, whenever a proposal is inside an activity interval, it would not be possible to determine where the activity ends by only considering the features inside the proposal. Therefore, using a context based representation is critical for temporal localization of activities. Additionally, based on how much background the current and next scales cover, it becomes possible to determine if a proposal is a good candidate.

5.2.3 Sampling and Temporal Convolution

To train the proposal network, we assign labels to proposals based on their overlap with ground truth, as follows,

$$Label(S_j) = \begin{cases} 1, & iou(S_j, GT) > 0.7 \\ 0, & iou(S_j, GT) < 0.3 \end{cases} \tag{5.1}$$

where $iou(\cdot)$ is intersection over union overlap and GT is a ground truth interval. During training, we construct a mini batch with 1024 proposals with a positive to negative ratio of 1:1.

Given a pair of features $Z_{i,k}$, $Z_{i,k+1}$, from two consecutive scales, we apply temporal convolution to features sampled from each temporal scale separately to capture context information between scales, as shown in Figure 5.2. A temporal Convolutional Neural Network [45] enforces temporal consistency and obtains consistent performance improvements over still-image detections. To aggregate information across scales, we concatenate the two features to obtain a fixed dimensional representation. Finally, two fully connected layers are used to capture context information across scales. A two-way Softmax layer followed by cross-entropy loss is used at the end to map the predictions to labels (proposal or not).

5.2.4 Classification

Given a proposal with a high score, we need to predict its action class. We use bilinear pooling by computing the outer product of each segment feature, and average pool them to obtain the bilinear matrix $bilinear(\cdot)$. Given features $\hat{Z} = [z_1, z_2, ...z_l]$ within a proposal, we

conduct bilinear pooling as follows:

$$bilinear(\hat{Z}) = \sum_{i=1}^{l} \hat{Z}_i^T \hat{Z}_i \tag{5.2}$$

For classification, we pool all the features l which are inside the segment and do not perform any temporal sampling. We pass this vectorized bilinear feature $x = bilinear(\hat{Z})$ through a mapping function with signed square root and l^2 normalization [67]:

$$\phi(x) = \frac{sign(x)\sqrt{x}}{||sign(x)\sqrt{x}||_2} \tag{5.3}$$

We finally apply a fully connected layer and use a 201-way (200 action classes plus background) Softmax layer at the end to predict class labels. We again use the cross entropy loss function for training. During training, we sample 1024 proposals to construct a mini batch. To balance training, 64 samples are selected as background in each mini-batch. For assigning labels to video segments, we use the same function which is used for generating proposals,

$$Label(S_j) = \begin{cases} lb, & iou(S_j, GT) > 0.7 \\ 0, & iou(S_j, GT) < 0.3 \end{cases} \tag{5.4}$$

where $iou(\cdot)$ is intersection over union overlap, GT is ground truth and lb is the most dominant class with in proposal S_j. We use this classifier for the ActivityNet dataset but this can be replaced with other classifiers as well.

5.3 Experiments

In this section, we provide analysis of our proposed temporal context network. We perform experiments on the ActivityNet and THUMOS14 datasets.

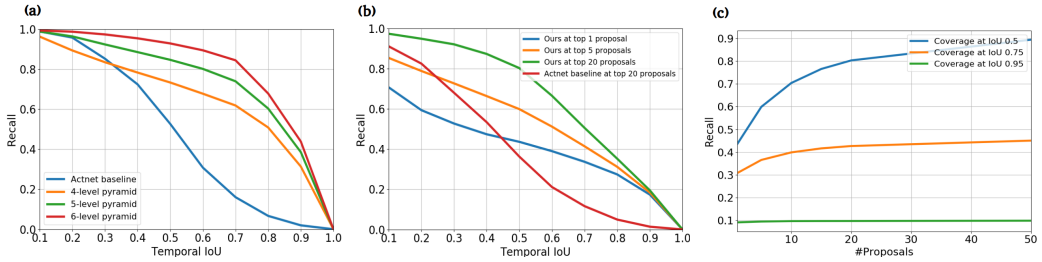


Figure 5.3: Performance of our proposal ranker on ActivityNet validation set. (a) The Recall vs IoU for pyramid proposal anchors; (b) The Recall vs IoU for our ranker at 1, 5, 20 proposals; (c) Recall vs number of proposals for our ranker at IoU 0.5, 0.75 and 0.95

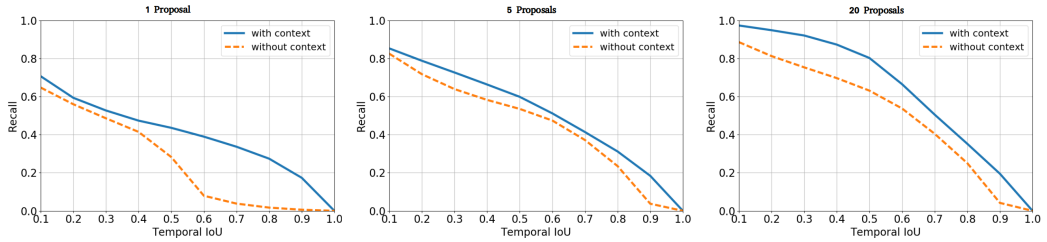


Figure 5.4: The effectiveness of context-based proposal ranker is shown in these plots. The Recall vs IoU plots show ranker performance at 1, 5, 20 proposals with and without context on ActivityNet validation set

5.3.1 Implementation details

We implement the network based on a customized Caffe repository with Python interface. All evaluation experiments are performed on a workstation with a Titan X (Maxwell) GPU. We initialize our network with pre-trained TSN models [107] and fine-tune them on both action labels and foreground/background labels to capture “actionness” and ”backgroundness”. Later, we concatenate these together as high-level features input to our proposal ranker and classifier. For the proposal ranker, we use temporal convolution with a kernel size of 5 and a stride of 1, followed by ReLU activation and average pooling with size 3 and stride 1. The temporal convolution responses are then concatenated and mapped to a fully connected layer with 500 hidden units, which is used to predict the proposal score. To evaluate our method on the detection task, we generate top K proposals (K is set to 20,

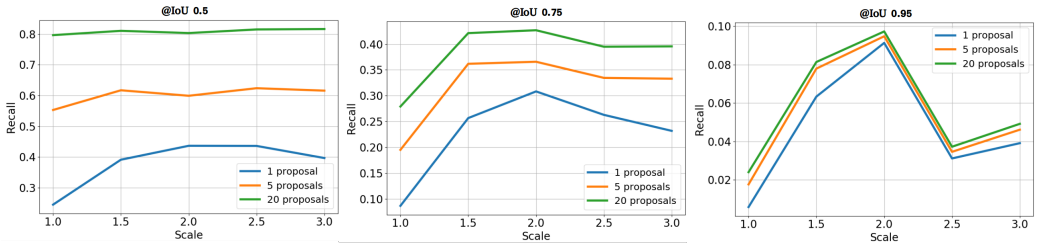


Figure 5.5: Comparing the ranker performance using different relative scale for context based proposals on ActivityNet validation set

we apply non-maximum suppression to filter out similar proposals, using an NMS threshold set as 0.45) and classify them separately. While classifying proposals, we also fuse two global video level priors using ImageNet shuffle features [59] and "actionness" features to further improve classification performance, as shown in [93]. We also perform an ablation study for different components of classification. For training the proposal network, we use a learning rate 0.1. For the classification network, we set learning the rate to 0.001. For both cases, we use a momentum of 0.9 and 5e-5 weight decay.

5.3.2 ActivityNet Dataset

ActivityNet [6] is a recently released dataset which contains 203 distinct action classes and a total of 849 hours of videos collected from YouTube. It consists of both trimmed and untrimmed videos. Each trimmed video contains a specific action with annotated segments. Untrimmed videos contain one or more activities with background involved. On average, each activity category has 137 untrimmed videos. Each video on average has 1.41 activities which are annotated with beginning and end points. This benchmark is designed for three applications: untrimmed video classification, trimmed activity classification, and untrimmed activity detection. Here, we evaluate our performance on the detection task in untrimmed videos. We use the mean average precision (mAP) averaged over multiple overlap thresholds to evaluate detection performance. Since test labels of ActivityNet are

	mAP@.5	mAP@.75	mAP@.95
without context	15.91	3.11	0.13
with context	36.17	21.12	3.89

Table 5.1: Evaluation on the influence with and without context on ActivityNet validation set

not released, we perform ablation studies on the validation data and test our full model on the evaluation server.

Proposal anchors We sample pair-wise proposals within a temporal pyramid. In Figure 5.3(a), we present the recall for the pyramid proposal anchors on ActivityNet validation set with three different levels. This figure shows the theoretical best recall one can obtain using such a pyramid. Notice that even with a 4-level pyramid with 64 proposals in total, the coverage is already better than the baseline provided in the challenge, which uses 90 proposals. This ensures our proposal ranker's performance is high with a low number of proposals.

Performance of our ranker We evaluate our ranker with different numbers of proposals. Figure 5.3(b) shows the average recall at various overlap thresholds with top 1, top 5 and top 20 proposals. Even when using *one* proposal, our ranker outperforms the ActivityNet proposal baseline by a significant margin when the overlap threshold is greater than 0.5. With top 20 proposals, our ranker can squeeze out most of the performance from pyramid proposal anchors. We also evaluate the performance of our ranker by measuring recall as the number of proposals varies (shown in Figure 5.3(c)). Recall at IoU 0.5 increases to 90% with just 20 proposals. At higher IoU, increasing the number of proposals does not increase recall significantly.

Effectiveness of temporal context We contend that temporal context for ranking proposals is critical for localization. To evaluate this claim, we conduct several experiments. In Figure 5.4, we compare the performance of the ranker with and without temporal con-

text. Using only the best proposal, without context, the recall drops significantly at high IoU (IoU > 0.5). This shows that for precise localization of boundaries, temporal context is critical. Using top 5 and top 20 proposals, without context, the recall is marginally worse. This is expected because as the number of proposals increases, there is a higher likelihood of one having a good overlap with a ground-truth. Therefore, recall results using a single proposal are most informative. We also compute detection metrics on the ActivityNet validation set to evaluate the influence of context. Table 5.1 also shows that detection mAP is much higher when using the ranker with context based proposals. These experiments demonstrate the effectiveness of our method.

Varying context window for ranking proposals Another important component for ranking proposals is the scale of context features which are associated with the proposal. Consider a case in which a proposal is contained within the ground truth interval. If the context scale is large, the ranker may not be able to distinguish between good and bad proposals, as it always see a significant amount of background . If the scale is small, there may be not enough context to determine if the proposal is contained within the ground truth or not. Therefore, we conduct an experimental study by varying the scale of context features while ranking proposals. In Figure 5.5, we observe that the performance improves up to a scale of 2. We evaluate the performance of the ranker at different scales on the ActivityNet validation set. In Table 5.2 we show the impact of varying temporal context at different overlap thresholds, which validates our claim that adding more temporal context would hurt performance, but not using context at all would reduce performance by a much larger margin. For example, changing the scale from 2 to 3 only drops the performance by 3% but changing it from 1.5 to 1 decreases mAP by 15% and 12% respectively.

Influence of number of proposals We also evaluate the influence of the number of proposals on detection performance. Table 5.3, shows that our method doesn't requires a large number of proposals to improve its highest mAP. This demonstrates the advantages

Context Scale	mAP@.5	mAP@.75	mAP@.95
1	15.91	3.11	0.13
1.5	30.51	15.56	2.23
2	36.17	21.12	3.89
2.5	36.04	17.08	0.92
3	33.29	14.35	1.03

Table 5.2: Impact of varying temporal context at different overlap thresholds on ActivityNet validation set

#Proposal/Video	mAP@.5	mAP@.75	mAP@.95
1	25.70	16.08	2.80
5	34.13	20.72	3.89
10	35.52	21.02	3.89
20	36.17	21.12	3.89
50	36.44	21.15	3.90

Table 5.3: Impact of number proposals on mAP on ActivityNet validation set

Components			mAP@.5	mAP@.75	mAP@.95
B.	F.	G.			
✓	✓	✓	36.17	21.12	3.89
✓	✓	×	33.83	20.05	3.77
✓	×	×	30.31	17.80	2.82
×	×	×	26.35	15.27	2.66

Table 5.4: Ablation study for detection performance using top 20 proposals on the ActivityNet validation set. B - Bilinear, F - Flow, G - Global prior

Evaluation Server				
Method	mAP@.5	mAP@.75	mAP@.95	Average
QCIS[108]	42.48	2.88	0.06	14.62
UPC[62]	22.37	14.88	4.45	14.81
UMD[92]	28.67	17.78	2.88	17.68
Oxford[93]	36.40	11.05	0.14	17.83
Ours	**37.49**	**23.47**	**4.47**	**23.58**

Table 5.5: Comparison with state-of-the-art methods on the ActivityNet evaluation sever using top 20 proposals

of both our proposal ranker and classifier.

Ablation study We conduct a series of ablation studies to evaluate the importance of each component used in our classification model. Table 5.4 considers three components: ”B” stands for “using bilinear pooling”; “F” stands for “using flow” and “G” stands for “using global priors”. We can see from the table that each component plays a significant role in improving performance.

Comparison with state-of-the-art We compare our method with state-of-the-art methods [62, 93, 93, 108] submitted during the CVPR 2016 challenge. We submit our results on the evaluation server to measure performance on the test set. At 0.5 overlap, our method is only worse than [108]. However, this approach was optimized for 0.5 overlap and its performance degrades significantly (to 2%) when mAP at 0.75 or 0.95 overlap is measured. Even though frame level predictions using a Bi-directional LSTM are used in [92], our performance is better when mAP is measured at 0.75 overlap. This is because [92] only performs simple grouping of contiguous segments which are obtained at multiple detection thresholds, instead of a proposal based approach. Hence, it is likely to perform worse on longer action segments.

5.3.3 The THUMOS14 Dataset

We also evaluate our framework on the THUMOS14 dataset[42], which contains 20 action categories from sports. The validation set contains 1010 untrimmed videos with 200 videos as containing positive samples. The testing set contains 1574 untrimmed videos, where only 213 of them have action instances. We exclude the remaining background videos from our experiments.

Note that solutions for action and activity detection could be different in general, as activities could be very long (minutes) while actions last just a few seconds. Due to their long duration, evaluation at high overlap (0.8 e.g.) makes sense for activities, but not for actions. Nevertheless, we also train our proposed framework on the validation set of THUMOS14 and test on the testing set. Our model also outperforms state-of-the-art methods on proposal metrics by a significant margin, which shows the good generalization ability of our approach.

Performance of our ranker Our proposal ranker outperforms existing algorithms like SCNN[85] and DAPs[22]. We show proposal performance on both average recall calculated using IoU thresholds from 0.5 to 1 at a step 0.05 (shown in Table 5.6) and recall at IoU 0.5 (shown in Table 5.7) using 10, 50, 100, 500 proposals. Our proposal ranker performs consistently better than previous methods, especially using small number of proposals.

In Table 5.8, it is clear that, the proposal ranker performance improves significantly when using a pair of context windows as input. Hence, it is important to use context features for localization in videos, which has been largely ignored in previous state-of-the-art activity detection methods.

Comparison with state-of-the-art Using off the shelf classifiers and our proposals, we also demonstrate noticeable improvement in detection performance on THUMOS14. Here, we compare our temporal context network with DAPs[22], PSDF[43], FG[81] SCNN[85]

Method	Avg.Recall [0.5:0.05:1]			
	@10	@50	@100	@500
DAPs	3.0	11.7	20.1	46.7
SCNN	5.5	16.6	24.8	48.3
Ours	7.7	20.5	29.6	49.2

Table 5.6: Average Recall from IoU 0.5 to 1 with step size 0.05 for our proposals and other methods on the THUMOS14 testing set

Method	Recall(IoU=0.5)			
	@10	@50	@100	@500
DAPs	8.4	29.2	46.9	85.5
SCNN	13.0	35.2	49.6	84.1
Ours	17.1	42.8	59.8	88.7

Table 5.7: Recall evaluation at IoU 0.5 between our proposals and state-of-the-art methods on THUMOS14 testing set

and CDC[83]. We replace the S-CNN proposals originally used in CDC with our proposals. For scoring the detections in CDC, we multiply our proposal scores with CDC's classification score. We show that our proposals further benefit CDC and improve detection performance consistently at different overlap thresholds.

5.3.4 Qualitative Results

We show some qualitative results for TCN, with and without context. Note that only top 5 proposals are shown. The ground truth is shown in blue while predictions are shown in green. It is evident that when context is not used, multiple proposals are present inside or just at the boundary of ground truth intervals. Therefore, although the location is near the actual interval (when context is not used), the boundaries are inaccurate. Hence, when de-

Method	Avg.Recall@100	mAP@0.5
Ours w/o Context	22.5	20.5
Ours w/ Context	29.6	25.6

Table 5.8: Evaluation on the influence with and without context on THUMOS14 testing set

Method	mAP@.4	mAP@.5	mAP@.6	mAP@.7
DAPs[22]	—-	13.9	—-	—-
FG[81]	26.4	17.1	—-	—-
PSDF[43]	26.1	18.8	—-	—-
SCNN[85]	28.7	19.0	—-	—-
SCNN+CDC[83]	29.4	23.3	13.1	7.9
Ours+CDC	**33.3**	**25.6**	**15.9**	**9.0**

Table 5.9: Performance of state-of-the-art detectors on the THUMOS14 testing set

tection metrics are computed, these nearby detections get marked as false positives leading to a drop in average precision. However, when context is used, the proposals boundaries are significantly more accurate compared to the case when context is not used.

5.4 Conclusion

We demonstrated that temporal context is helpful for performing localization of activities in videos. Analysis was performed to study the impact of temporal proposals in videos by studying precision recall characteristics at multiple overlap thresholds. We also vary the context window to study the importance of temporal context for localization. Finally, we demonstrated state-of-the-art performance on two challenging public datasets.

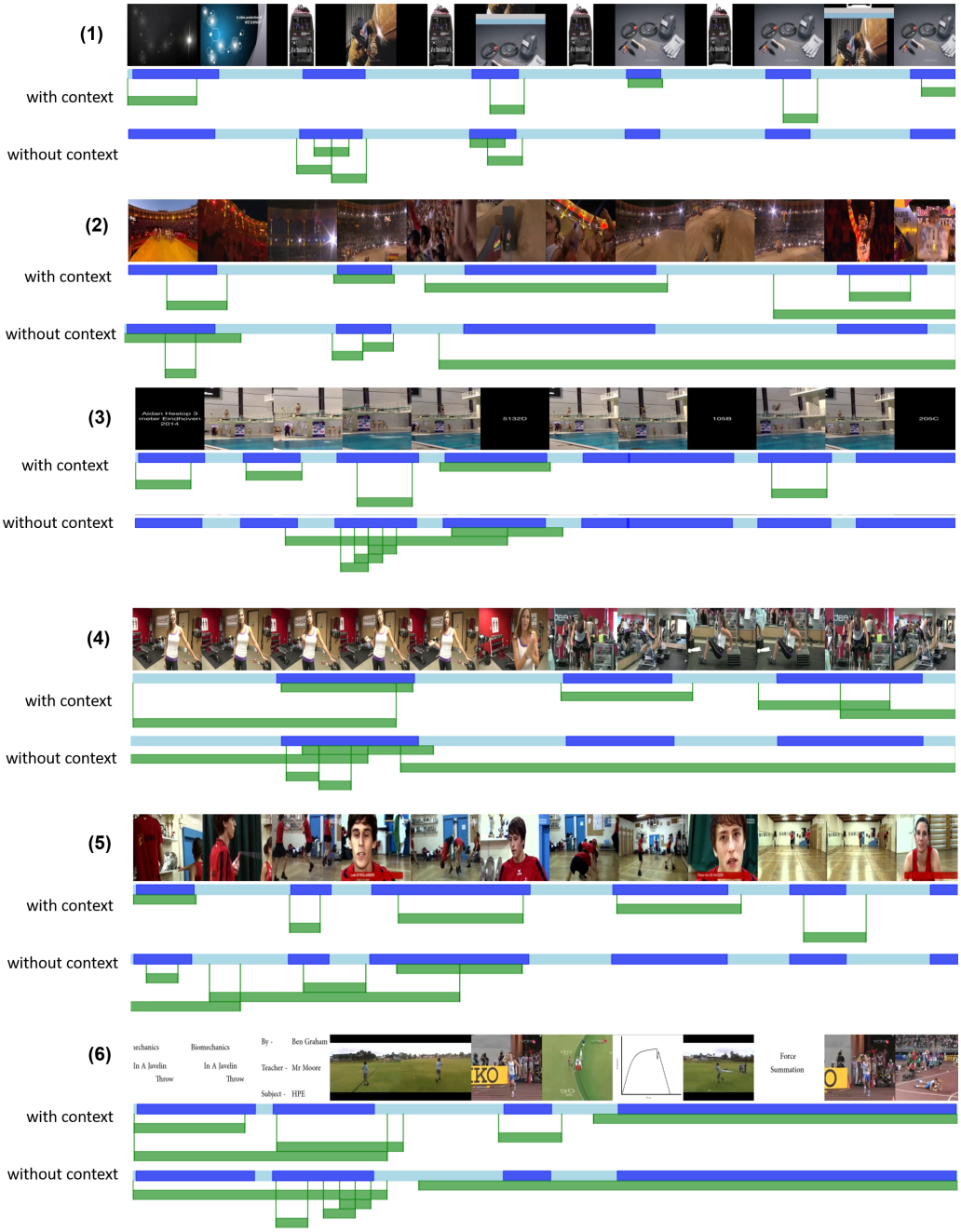


Figure 5.6: Visualization of top 5 ranking results, the blue bar denotes the ground-truth while the green one represents proposals.

Chapter 6: Temporal Aggregation Network for Dense Multi-label Action Recognition

6.1 Introduction

Convolutional Neural Networks (CNNs) have seen tremendous success across different problems including image classification [17, 34, 51], object detection [14, 29, 76, 91], style transfer [10, 15, 26], action recognition [8, 72, 90, 97] and action localization [16, 117, 118]. In each of these problems, several application specific changes to network design have been proposed. In particular, for action recognition, a two-stream network comprising two parallel CNNs, one trained on RGB images and another trained on stacked optical flow fields, showed that incorporating temporal information into the network architecture provides a significant benefit in performance [90]. Since optical flow computation is an additional overhead, network architectures like C3D [97] operate only on a sequence of images and perform 3D convolutions in each layer of the network. However, 3D convolutions in each layer increase the model complexity and with just 3x3x3 convolutions, it is hard to capture larger temporal context. Therefore, we need to design a network architecture which can learn semantic representations of actions efficiently.

While it is important to design efficient network architectures, they should also be capable of learning all the variations which appear in videos during training. In a video, multiple frames aggregated together represent a semantic label, so the amount of computation needed per semantic label is an order-of-magnitude larger compared to recognition tasks on

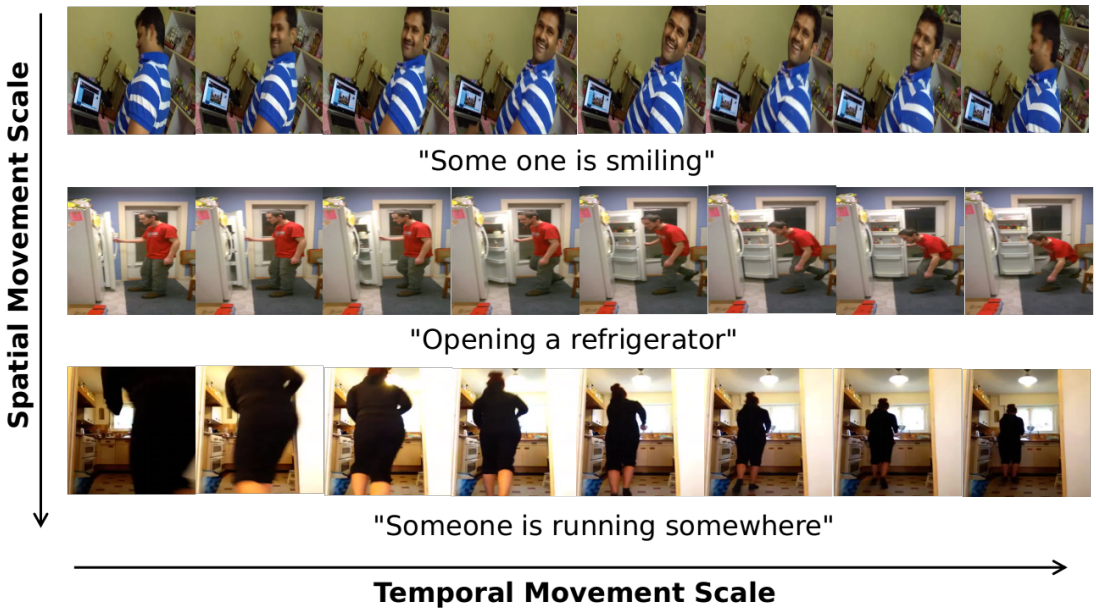


Figure 6.1: An illustration of the spatial and temporal variances of human activities

images. The difficulty is further amplified because actions can span multiple temporal and spatial scales in videos, shown in Figure 6.1. Short actions like micro-expressions (raising an eyebrow) require high-resolution spatio-temporal reasoning, while other longer actions like running or dancing involve large spatial movements, which can be identified with features with lower spatial resolution. Therefore, it is essential to aggregate multi-resolution spatio-temporal information without blowing up network complexity.

For image recognition tasks like object detection and semantic segmentation, dilated convolutions [11, 76] have been widely adopted to increase receptive field sizes without increasing model complexity. By applying dilated convolutions with different filter sizes, multi-scale context can be efficiently captured. Multi-scale *spatio-temporal* context is important for video recognition tasks. To this end, we propose TAN, which applies multi-scale dilated temporal convolutions after every spatial downsampling layer in the network. Since the spatial receptive field of the network doubles after each downsampling layer, our network has the capacity to learn fine-grained motion patterns in the feature-maps gen-

erated closer to the input. Large scale spatio-temporal patterns are captured in deeper layers which have coarse resolution. By only applying temporal convolutions after each downsampling layer, the computational burden and model complexity are reduced when compared to methods which apply 3D convolutions in each layer of the network. The use of dilated convolutions also facilitates capturing larger temporal context efficiently. We conducted extensive ablation studies to verify the effectiveness of our approach.

Our architecture is especially suitable for the dense action labeling task as it offers a good balance between temporal context and bottom up visual features computed at a particular time instant. By modeling short-term context efficiently with convolutions, TAN obtains state-of-the-art results on two benchmark datasets, Charades and Multi-THUMOS for the dense action prediction task, outperforming existing methods by 5% and 3% respectively. Although TAN is designed for dense action prediction, we also applied it to action detection where it obtains state-of-the-art results, showing the effectiveness of TAN in a variety of tasks.

6.2 Architecture

In this section, we introduce our Temporal Aggregation Network (TAN) in detail. We summarize existing methods and describe the difference. A detailed analysis and intuition of our key building blocks is then provided.

6.2.1 Temporal Modeling

Current approaches to temporal modeling generally fall in two categories:

2D Convolution + Late fusion. Many previous works treat video as a collection of frames. A common approach is to extract frame-by-frame features from deep networks pre-trained on the ImageNet classification task. Subsequently, these frame-wise features

are fused using variants of pooling, temporal convolutions or LSTMs. To compensate for missing motion cues from static image features, a second network extracts motion features from flow or improved dense trajectories (IDT) separately - commonly known as two-stream approaches.

3D Convolution. Alternatively, 3D convolutions can be used to model video directly. 3D convolutions are a natural extension to traditional 2D convolutions with an extra dimension spanning the temporal domain. 3D convolution networks form a hierarchical representation of spatio-temporal data. However, because of an additional dimension in the convolution kernels, 3D convolutional networks have significantly more parameters compared to 2D networks, which leads to difficulty in optimization and over-fitting.

Temporal Aggregation Network. Our approach differs from both previous approaches. Our goal is to model spatio-temporal information by decomposing 3D convolutions into spatial and temporal dilated convolutions. By stacking both types of convolutions repeatedly, we form a temporal aggregation network (TAN) that not only captures spatio-temporal information but also creates hierarchical representations.

6.2.2 Proposed Temporal Aggregation Module

A Temporal Aggregation module combines multiple temporal convolutions with different dilation factors (in the temporal domain) and stacks them across the entire network. This supports the capture of context information across multiple spatio-temporal scales. Temporal convolution is a simple 1D convolution. A dilated convolution is a convolution whose filter has been dilated in space by a specific factor. They have been widely used in semantic segmentation and object detection tasks [113] to capture spatial context across scales. Here, we apply 1D dilated convolutions in the temporal dimension, see Fig. 6.2. They enable capturing larger context without reducing the resolution of the feature-map.

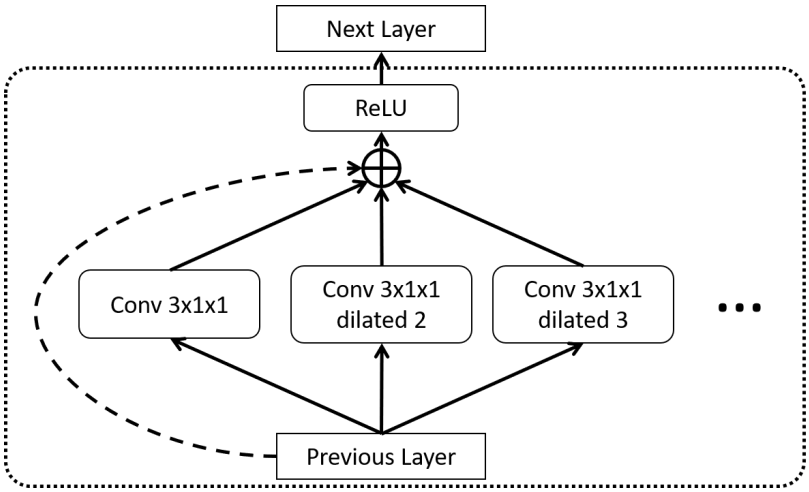


Figure 6.2: The proposed temporal aggregation module. It consists of one temporal convolution and several temporal dilated convolutions to capture motions at multiple temporal resolutions.

Multiple temporal dilated convolutions not only help to model context across temporal scales but also form an internal attention mechanism to match a large range of motion frequencies.

Implementation. Given any feature inputs from previous layers, we apply one temporal convolution with a filter size of 3 and two dilated temporal convolutions with filter size 3 and dilation factors 2 and 3 respectively. All temporal convolutions are applied across all spatial positions and channels, which help to learn temporal patterns among different filter activations. Inspired by the recent success of residual networks [34], we also add a residual identity connection from previous layers. The responses from all convolutions are further accumulated by conducting element-wise sum followed by a ReLU non-linear activation, which together behave as a soft attention based weighted selection among different temporal resolutions. The proposed module is illustrated in Figure 6.2.

Relation to LSTM. Long short term memory (LSTM) units are normally used in sequence modeling. By using a gated connection, LSTM allows a network to ”remember”

important context clues over time. Our temporal aggregation network behaves similarly by modeling temporal information using temporal convolutions. Additionally, our module is capable of distinguishing motion frequency differences among temporal resolutions, which is particularly useful when applied to action recognition tasks where actions having varying lengths.

Relation to Wavenet. Our temporal aggregation module is partially inspired by the Wavenet [100] model. Our module also uses dilated convolutions to model different temporal context at many time-scales. But there are significant differences. The major one is that our module is used in conjunction with spatial convolutions to capture both low-level and high-level temporal consistency among spatial positions, which is critical for modeling spatio-temporal relationships. Additionally, since we apply our module across different channels of spatial filter responses, it also behaves as a object instance trajectory learner similar to [105].

Relation to stack of convolutions with different filter sizes. It is possible to model multiple temporal resolutions by just stacking convolutions with different filter sizes. However, one of the main reasons we choose dilated convolution over convolution is computing cost. Multi-scale convolutions such as those used in models like inception [94], have to largely increase the size of learnable filters. To accomplish effective learning, one usually needs to resort to dimensionality reduction as the response channels increase in size rapidly. By using dilated convolution, we are able to achieve the same goal without reducing temporal resolution.

6.2.3 Spatial and Temporal Convolution Stacking

As discussed previously, we repeatedly stack spatial and temporal convolutions to model spatio-temporal information and create a hierarchical spatio-temporal representation. We

need a deep architecture that can be effectively trained. Multi-entity actions, like "person picking a book", typically involve larger and semantically richer spatio-temporal motion patterns which are better represented in deeper layers. On the other hand, other actions may be specific to an object instance, like "person smiling" which require high resolution spatio-temporal features (for smiling) while still understanding deeper semantic concepts like "person". We seek to capture high-level semantic concepts while preserving high-resolution features generated in the early layers of the network. Residual networks are a good candidate since their linear structure with residual connections preserves information learned across different convolution levels.

To model spatial information, we borrow the bottleneck structure from residual networks. A bottleneck block is a stack of 3 convolution layers with filter size 1x1, 3x3 and 1x1. The 1x1 convolution layers are used to adjust dimensions and the 3x3 convolution layer behaves as a bottleneck with smaller input/output dimensions. A residual identity connection is also attached to accelerate learning. Each bottleneck block is followed by a ReLU non-linear activation.

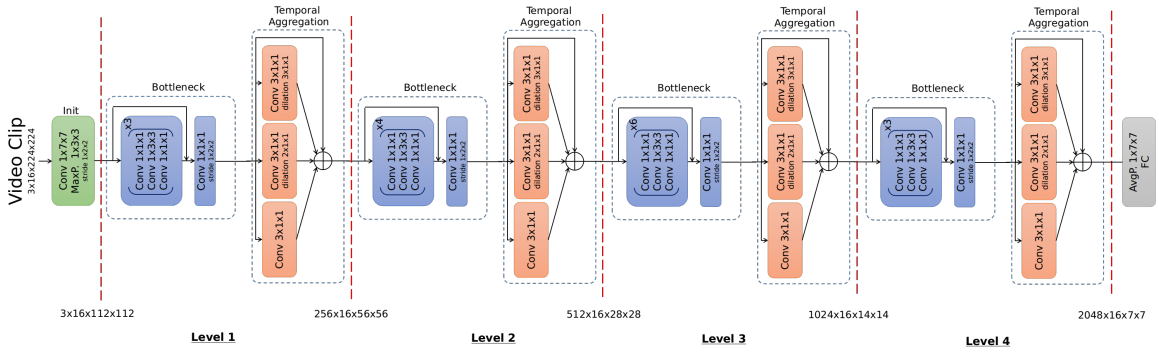


Figure 6.3: A closer look of our network architecture. Our model contains four levels. Each level consists of several bottleneck blocks and one temporal aggregation module at the end. The spatial resolution is reduced by two after each level, while the temporal resolution is not reduced.

The intuition behind repeatedly stacking spatial and temporal blocks among different

levels is straight-forward. In the early stage of the network, we capture small spatio-temporal patterns using temporal convolutions as the spatial receptive field is small. In the deeper stages of the network, we progressively capture larger spatio-temporal patterns using temporal convolutions as the spatial receptive field increases. Meanwhile, our temporal aggregation blocks support capturing both fast and slow actions across temporal scales and across spatial levels. In this way, we form hierarchical spatio-temporal representations.

Stacking filters in the same block was first proposed in the VGG network [89] in which a single large convolution of size 7x7 was broken down into three successive kernels. Such a stacking of convolution kernels increased the depth of the network and also reduced the model complexity (as three 3x3 kernels have fewer parameters than one 7x7 kernel). The receptive field of the network was also the same as the original network which used 7x7 convolutions. Since neural networks are prone to over-fitting, by reducing the model complexity while preserving the desirable characteristics of the network (Zeiler and Fergus [116]), the VGG network reduced over-fitting and hence was able to improve its performance significantly on the image classification task. Our approach of stacking temporal convolutions after spatial convolutions employs a similar strategy for reducing model complexity - albeit for modelling spatio-temporal patterns, where over-fitting is a serious concern.

6.2.4 Full Model

The final architecture consists of four levels of bottleneck blocks for spatial modeling and temporal aggregation blocks for temporal modeling. In each level, there are multiple bottleneck blocks following by one temporal aggregation block similar to a residual network.

The weights of bottleneck blocks can be initialized using pre-trained ImageNet models.

The spatial resolution is reduced after the initial convolution and pooling layers and after every level with max pooling. Notice that there is no temporal resolution reduction due to dilated convolutions in our temporal aggregation blocks. After all four levels of feature learning, the final feature outputs are pooled by average pooling and mapped to semantic space by a fully connected layer. Depending on the specific task, our network can both output an aggregated prediction or dense predictions with no extra cost. Our full model is illustrated in Figure 6.3.

6.3 Experiments

We conducted a series of ablation studies to illustrate the efficiency and effectiveness of our model. We then compare to state-of-the-art methods on dense multi-label action recognition tasks. Lastly, we apply the model to the action detection task.

Datasets. Charades [88] contains 157 action classes from common everyday activities collected by 267 people at home. It has 7986 untrimmed videos for training and 1864 untrimmed videos for validation with an average length of 30 seconds. It is considered as one of the most challenging multi-label action recognition dataset. MultiTHUMOS [112] contains 65 action classes and another 413 videos apart from the original THUMOS datatset [42] and extends it to a dense multi-label task with 1.5 labels per frame and 10.5 distinct action categories per video. It is suitable for an in-depth study of simultaneous human actions in video. THUMOS14 [42] contains 20 sport action classes from over 24 hours of videos. The detection task contains 2765 trimmed videos for training, 1,010 untrimmed videos for validation, but only 200 untrimmed videos in validation and 213 untrimmed videos in testing set have foreground labels. This dataset is quite challenging because it contains long videos of multiple short action instances.

Method	Frame mAP	Video mAP
Res50-2D	11.7	21.1
Res50-3D	10.3	19.0
Res50-TAN (Ours)	14.2	25.5

Table 6.1: Comparison of different temporal modeling architectures on Charades.

Components					Frame mAP	Video mAP
Lv. 4	Lv. 3	Lv. 2	Lv. 1	Dilation		
✓	×	×	×	✓	12.2	22.1
✓	✓	×	×	✓	13.1	23.6
✓	✓	✓	×	✓	13.7	24.9
✓	✓	✓	✓	✓	**14.2**	**25.5**
✓	✓	✓	✓	×	12.2	22.4

Table 6.2: Ablation study for placing temporal aggregation module at different levels and the effectiveness of dilation

6.3.1 Ablation Study

Analysis of Temporal Modeling Methods. We first compare our model with two baselines. The three models are:

- **Res50-2D**: A model based on ResNet-50, trained frame-by-frame and with an LSTM at the end to generate predictions.
- **Res50-3D**: A model using 3D convolutions to replace all 2D convolutions in ResNet-50. For example, we use 3x3x3 convolutions instead of 3x3 convolutions and use a stride size of 2x2x2 instead of 2x2 when needed.
- **Res50-TAN**: Our proposed method with spatial convolution layers initialized from ResNet-50.

All models take a 16 frame clip with 224x224 resolution as input in 4 FPS, which corresponds to about 4 seconds. As 3D convolutions require large number of training data, to

Sampling rates	1 FPS	2 FPS	4 FPS	8 FPS
Video mAP	21.3	25.3	25.5	24.1

Table 6.3: Comparison of performance using different temporal stride on Charades

maintain fairness in experiments, all models are pre-trained on the recently released large scale Kinetics dataset [8] and then fine-tuned and tested on the Charades dataset [88]. The results are shown in Table 6.1. We evaluated on both frame-level mAP and video-level mAP. Our proposed TAN clearly outperforms the baseline methods, while the 3D convolution version performs worst due to the reduction of temporal resolution.

Effectiveness of the Temporal Aggregation Module. We evaluate the effectiveness of our network architecture by conducting an ablation study by varying the position of the temporal aggregation module in the network, such as only at the final level 4, only at level 3,4 and at level 2,3,4. We also describe an experiment by replacing our temporal aggregation module with a simple 3x1x1 temporal convolution. Table 6.2 shows the comparative results. As more temporal aggregation modules are added, performance improves. This shows that our model can learn hierarchical spatio-temporal patterns. Meanwhile, we also evaluate the effectiveness of temporal dilated convolutions in the temporal aggregation module. When we replace our temporal aggregation module with a simple temporal convolution, performance drops significantly by 2% on Frame mAP and 3.1% on Video mAP. This shows that our temporal aggregation module is more effective at modeling temporal information from multiple temporal scales.

Impact of Sampling Rate in Videos. We train and test our model with four different video sampling rates of 1,2,4,8 FPS. As seen in Table 6.3, the performance doesn't drop significantly until we aggressively drop the video sampling rate to 1FPS. Our model shows reasonable immunity to variances of temporal resolution.

Impact of Networks Layers. We also conduct an ablation study to investigate the

Number of layers	$50_s + 12_t$	$101_s + 12_t$	$152_s + 12_t$
Frame mAP	14.2	15.5	17.6
Video mAP	25.5	27.5	31.0

Table 6.4: Comparison of performance using different depth of spatial convolutions on Charades

Method	Frame mAP	Video mAP
Ours w/o pretraining	14.0	25.3
Ours w/ pretraining	17.6	31.0

Table 6.5: Comparison of performance with and without Kinetics pretraining on Charades

influence of depth of networks. Here, we only vary the number of bottleneck blocks used in level 3 and level 4, similar to ResNet, leaving the temporal blocks unchanged at the end of each level. Table 6.4 shows that our model can further benefit from deeper networks.

Impact of Pretraining. Finally, we conduct an ablation study to investigate the influence of pretraining. Table 6.5 shows that our model can further benefit from larger dataset, meanwhile even without pretraining, our method still achieves state-of-the-art performance compared to previous approaches.

6.3.2 Multi-label Action Recognition

We conduct experiments on two popular multi-label action recognition datasets and compare our results to state-of-the-art methods.

Implementation details. For all datasets, we use a temporal aggregation network based on ResNet-152 with a 16 frame temporal resolution. It is pre-trained on the Kinetics dataset and fine-tuned later. For the Charades dataset, we use a sampling rate of 4 FPS. We optimized our model using the Adam optimizer with a learning rate of 10^{-4} for the first 10 epochs and 10^{-5} in the following 10 epochs. It is evaluated using two metrics following [88]: one is a video level mAP for video based multi-label classification and the other is a

Method	Frame mAP	Video mAP
Random [88]	–	5.9
C3D [97]	–	10.9
IDT [103]	–	17.2
Two-stream [23]	8.9	14.3
Asynchronous Temporal Field [86]	12.8	22.4
ActionVLAD [28]	–	21.0
Predictive-corrective [18]	8.9	–
R-C3D [111]	12.7	–
Ours	**17.6**	**31.0**

Table 6.6: Comparison with state-of-the-art methods on Charades

Method	Frame mAP
IDT [103]	13.3
Single-frame CNN [89]	25.4
Two-Stream [23]	27.6
Multi-LSTM [112]	29.6
Predictive-corrective [18]	29.7
Ours	**33.3**

Table 6.7: Comparison with state-of-the-art methods on MultiTHUMOS using frame level mAP

frame level mAP uniformly extracted at 25 frames per video to approximate the multi-label action detection task. For video level classification, average pooling is used to map dense frame level labels to video level labels. For MultiTHUMOS, we use a sampling rate of 10 FPS following [112]. We also optimized our model using the Adam optimizer with a learning rate of 10^{-4} for the first 10 epochs and 10^{-5} in the following 5 epochs. Finally, MultiTHUMOS is also evaluated using the standard frame level mAP metric.

Comparison with state-of-the-art. On Charades, we compare with several state-of-the-art methods such as [18, 23, 28, 86, 88, 97, 103]. Our results are shown in Table 6.6. We first show results of our model without Kinetics pre-training, and it still outperforms state-of-the-art methods by 1.2% on frame level mAP and 2.9% on video level mAP. Our model with Kinetics pre-training obtains a significant improvement over other methods and

Method	Frame mAP
Single-frame CNN [89]	34.7
Two-Stream [23]	36.2
Multi-LSTM [112]	41.3
Predictive-corrective [18]	38.9
CDC [83]	44.4
Ours	**46.8**

Table 6.8: Comparison with state-of-the-art methods on THUMOS14 using frame level mAP

Figure 6.4: Visualization of network filter responses from multiple levels. From top to bottom row, each row shows filter responses of a layer from lower to higher layer.

outperforms them by 4.8% on frame level mAP and 8.6% on video level mAP respectively. On MultiTHMOS, we compare with [18, 23, 89, 112] in Table 6.7. Our model performs 3.6% better than the previous state-of-the-art method [18]. We also compare our model with several recent network on THUMOS14, such as [18, 23, 83, 89, 112]. As seen in Table 6.8, our model again outperforms previous state-of-the-art methods [83] by 2.4% on frame-level evaluation.

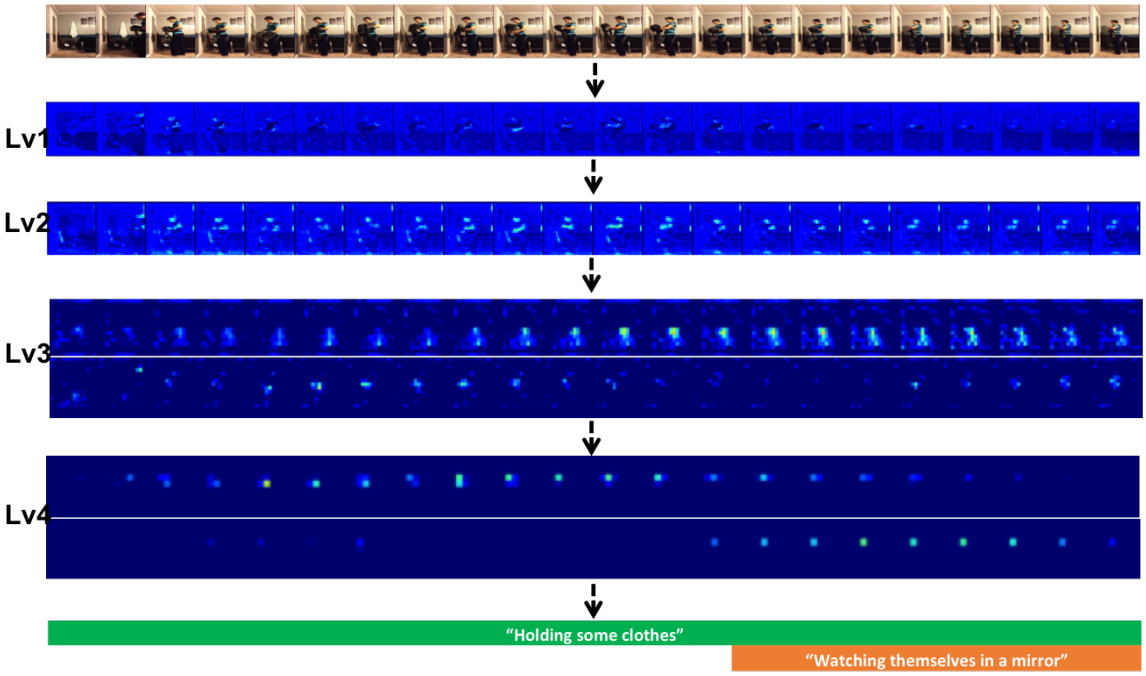


Figure 6.5: Visualization of representative filter responses after each level of temporal aggregation block.

6.3.3 Visualization

We visualize the filter responses of our network to better understand what is learned in different levels in Figure 6.4. In the example, we observe that the early layer filters have responses spread over a larger part of the image, while the later layers learn to focus where the action is present. To better understand how the network learns spatial-temporal consistency, we also visualize representative filter responses after each level of temporal aggregation block, shown in Figure 6.5. In spatial domain, the sallow layers tend to highlight fine-grained movement (such as facial, hand movement) and the deep layers tend to highlight large movement (such as body movement), due to the spatial effective receptive window increases as network goes deep. In temporal domain, the network is capable of capturing action sequence of variable length, thanks to our temporal aggregation blocks.

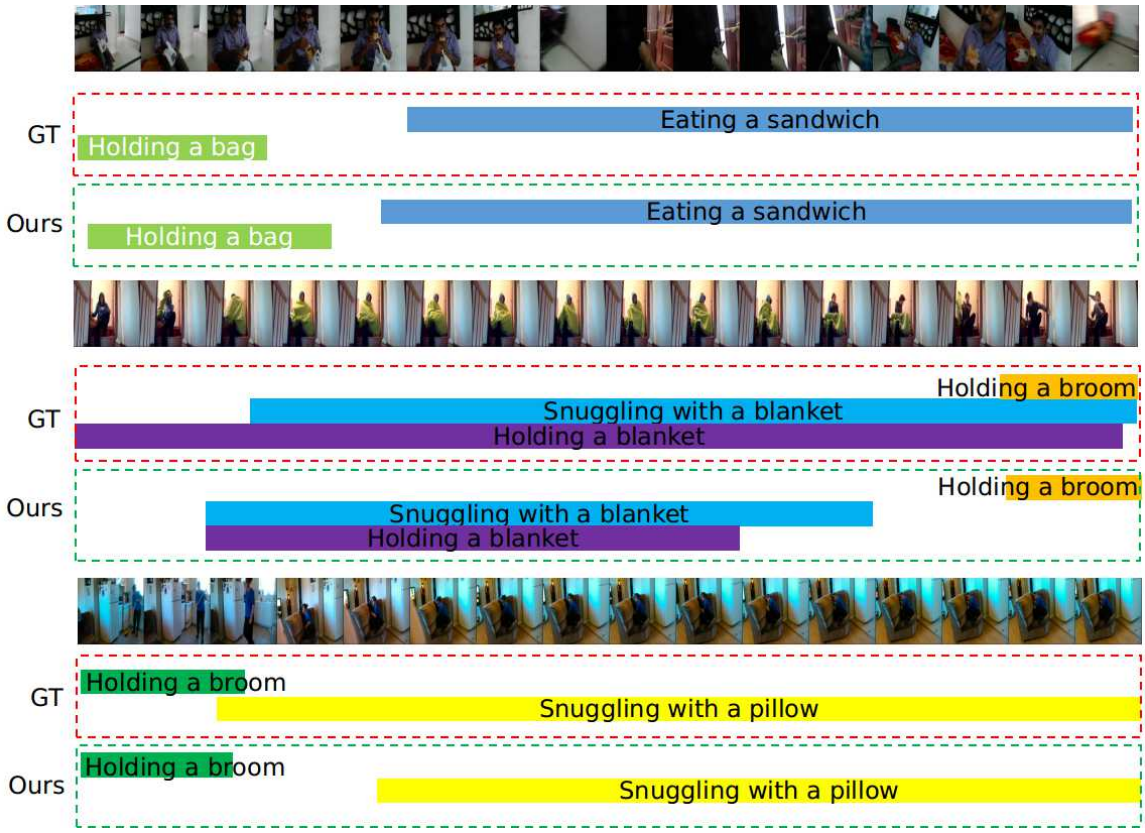


Figure 6.6: Qualitative results on the Charades dataset. Red and green dashed boxes show the ground-truth and our predictions respectively. Our network is able to correctly localize multiple actions of variable durations simultaneously.

Meanwhile, it is interesting to see that how spatial-temporal responses group up as the network advances. This demonstrates that our model progressively constructs hierarchical spatio-temporal representations for recognizing human actions.

We also present qualitative results for different samples in Figure 6.6. These results show that our network is effective in localizing both long and short duration actions simultaneously. Notice that it also predicts labels of multiple actions happening at the same time instance.

6.4 Conclusion

We presented a temporal aggregation module that combines temporal convolution with varying levels of dilated temporal convolutions to capture spatio-temporal information across multiple scales. We compare our design with multiple temporal modeling methods. Based on this, we designed a deep network architecture for video applications. Our model stacks spatial convolutions and the new temporal aggregation module repeatedly to learn a hierarchical spatio-temporal representation. It is both efficient and effective compared to existing methods such as 3D convolution networks and two-stream networks. We conduct ablation studies to analyze our model and experiments on multi-label action recognition and pre-frame action recognition datasets prove the effectiveness of our method.

Chapter 7: Spatial Temporal Human Action Understanding

7.1 Introduction

Video application has been one of long-lasting interests in computer vision communities. Thanks to recent progress of deep learning methods, there have been significant improvement over traditional video tasks such as action recognition [8, 97, 106] and activity localization [16, 22, 111]. However, spatial temporal action understanding still haven't seen large breakthrough. Although there have been several early attempts, the problem is still under well-studied and has been recognized as one of the most challenging task in video analysis research. It is hard due to the requirement of instance-level awareness that needs a system to predict where, when and what action happens for each person in a crown scene, which has multiple potential applications in video surveillance. It can be understood as a task combining both object detection and action localization.

There have been significant progress in object detection. State-of-the-art methods often shares a two-stage structure that first generate object proposals and then refine predictions. Although these models archive promising results in still image, the performance will be largely deteriorated when applied to moving object. In order to accurately localize a object, the deep networks are usually trained to highlight specific appearance features, which can be easily disturbed by any motion blurs, video refocuses, changes of explosion and nonlinear view projections from camera movement. Even worse, when apply object detection to human body, since human beings are not a rigid object, it often results as a bounding

box cutting off the moving legs and hands, which part may ne crucial to help recognize an action.

Most recently, there has been a widely practice to re-integrate motion clues into deep networks for moving object detection and action recognition. Several works have successfully adopted optical flow to smooth and improve feature quality to assist moving object detection. Others have tried to utilized optical flow to improve action recognition. In fact, the success of such methods indicate the lacking of motion learning in current image-level ImageNet pre-trained deep networks. Meanwhile, it is interesting to see that, Optical Flow, although has been used for decades in video applications, has not been throughly studied for possible formulations in spatial temporal action understanding.

In this work, we first investigate the context difference between localizing human in spatial domain and recognizing action in temporal domain. We then propose a decoupled framework built based on existing two-stage object detector that has both dedicated spatial branch and temporal branch to model the spatial and temporal context difference. Such approach has been proved effectively when evaluated on state-of-the-art benchmarks.

7.2 Decoupling Spatial and Temporal Context in a Network

We first present a decoupled framework for spatio-temporal action detection in which action localization in the spatial domain and action classification in temporal domain are decoupled, shown in Figure 7.1. To obtain the final action probability for an RoI, we multiply the actionness score (which is class-agnostic) with the fine-grained classification score. Our approach uses the Faster-RCNN architecture which is trained with 2 classes (action/no action). For fine-grained classification, a separate network is employed. Our idea is based on the assumption that spatial action localization and action recognition require different cues. Trying to solve these two problems jointly may require more data, which is

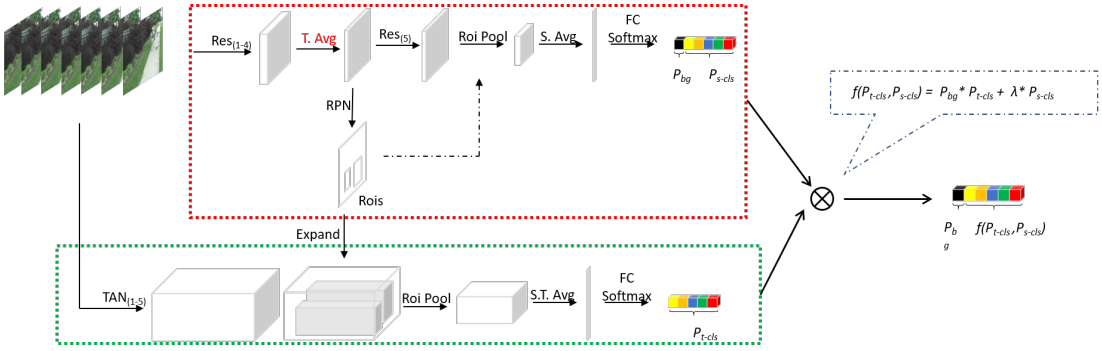


Figure 7.1: A overview of our decoupled framework, which has dedicated branches for spatial localization and temporal recognition. Contexts in spatial and temporal branches are modeled explicitly and fused together at last.

hard to collect in this setting.

7.2.1 Two-stream Faster-RCNN for Human Detection

We use a two-stream Faster-RCNN which accepts a short chunk of video (K frames) as an input and predicts the bounding box position of the middle frame of the chunk. The first stream is RGB which contains the middle frame of the chunk. The second stream uses a stack of optical-flow frames as an input. The features are combined after conv4 for RPN and after conv5 for the classification head. The classification head has 2 fully connected layers and generates action/not-action probability for each proposal. For recognizing the action in these RoIs, we use a separate network, described in the next section. Features for the network are generated like R-CNN, i.e. by cropping and re-sizing these RoIs to 224x224. The recognition scores and actionness scores are finally multiplied to obtain the final score of an action.

7.2.2 Multi-span Temporal Aggregation Network for Action Recognition

We present Temporal Aggregation Network (TAN) which decomposes 3D convolutions into spatial and temporal aggregation blocks. By stacking spatial and temporal convolutions repeatedly, TAN forms a deep hierarchical representation for capturing spatio-temporal information in videos. Since we do not apply 3D convolutions in each layer but only apply temporal aggregation blocks once after each spatial downsampling layer in the network, we significantly reduce the model complexity. The use of dilated convolutions at different resolutions of the network helps in aggregating multi-scale spatio-temporal information efficiently. Experiments show that our model is well suited for dense multi-label action recognition, which is a challenging sub-topic of action recognition that requires predicting multiple action labels in each frame.

A Temporal Aggregation module combines multiple temporal convolutions with different dilation factors (in the temporal domain) and stacks them across the entire network. This supports the capture of context information across multiple spatio-temporal scales. Temporal convolution is a simple 1D convolution. A dilated convolution is a convolution whose filter has been dilated in space by a specific factor. They have been widely used in semantic segmentation and object detection tasks to capture spatial context across scales. Here, we apply 1D dilated convolutions in the temporal dimension. They enable capturing larger context without reducing the resolution of the feature-map. Multiple temporal dilated convolutions not only help to model context across temporal scales but also form an internal attention mechanism to match a large range of motion frequencies.

We repeatedly stack spatial and temporal convolutions to model spatio-temporal information and create a hierarchical spatio-temporal representation. We need a deep architecture that can be effectively trained. Multi-entity actions, like “person picking a book”, typically involve larger and semantically richer spatio-temporal motion patterns which are

better represented in deeper layers. On the other hand, other actions may be specific to an object instance, like "person smiling" which require high resolution spatio-temporal features (for smiling) while still understanding deeper semantic concepts like "person". We seek to capture high-level semantic concepts while preserving high-resolution features generated in the early layers of the network.

The intuition behind repeatedly stacking spatial and temporal blocks among different levels is straight-forward. In the early stage of the network, we capture small spatio-temporal patterns using temporal convolutions as the spatial receptive field is small. In the deeper stages of the network, we progressively capture larger spatio-temporal patterns using temporal convolutions as the spatial receptive field increases. Meanwhile, our temporal aggregation blocks support capturing both fast and slow actions across temporal scales and across spatial levels. In this way, we form hierarchical spatio-temporal representations.

Meanwhile, the dilated convolution in our temporal aggregation block enables our network to learn from arbitrary length of input frames. Hence, a multi-span input approach can be utilized to further capture temporal context among different length and frequency.

7.3 Experiments

We conduct experiments on two challenging spatial temporal action detection datasets, UCF101-Detection and AVA to evaluate the performance of our methods. We first evaluate the performances of spatial and temporal branches separately to demonstrate the effectiveness of each component. Later we compare with state-of-the-art methods to show advantage of our system

7.3.1 Datasets

UCF101-Detection is a widely used action recognition dataset. When evaluated on spatial temporal detection task, a subset of 24 sport classes with 3207 videos is used. Similar to previous works, we evaluate our method on the first train/test split only. We used frame-mAP as the standard metric.

AVA is a newly released dataset. This dataset is challenging due to its multi-label recognition setup. It contains video 15 minutes clips from 430 movies, where 235 are used for training, 64 for validation and 131 for testing. It densely annotates 80 atomic actions (only 60 are used in evaluation) in 57.6K video segments at an one second interval. Since the test set is only evaluated on official evaluation sever during challenge. We run experiments only on validation set and use frame-mAP as a metric.

7.3.2 Ablation Study

We conduct several ablation studies to demonstrate the effectiveness of each component in our framework.

Performance of human detection

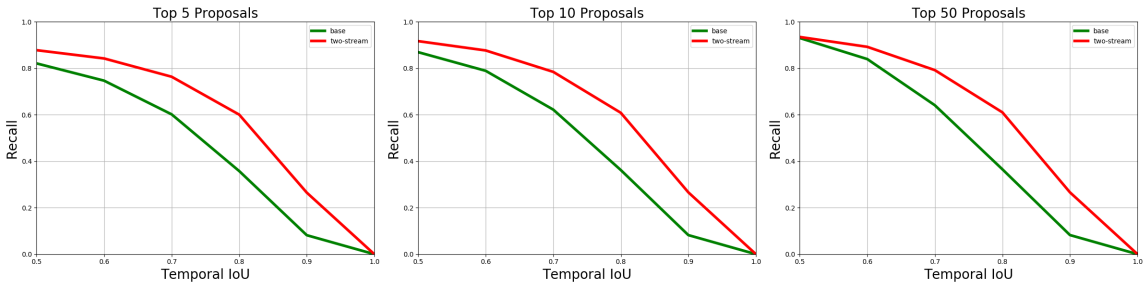


Figure 7.2: Recall Comparison between two-stream human detector and standard faster-rcnn detector

We first evaluate the performance of our two-stream faster-rcnn detector. As discussed

Span	mAP
f7s7	14.3
f7s7 + f7s3	14.9
f7s7 + f7s3 + f7s14	15.3
f7s7 + f7s3 + f7s14 + f16s4 + f16s8 + f16s16	16.5

Table 7.1: Influence of multiple spans on performance on AVA validation set. "f" means frame, "s" means stride.

in introduction, object detection trained on still image often suffers significant performance drop when applied to moving object. In this ablation study, we compare our proposed two-stream framework compared to the standard RGB approach. Shown in Figure 7.2, our two-stream human detector out-performs baseline method at a significant margin at all IoUs. This demonstrates that by integrating optical flow in our framework, we can better enhance the feature quality for spatial detection.

Performance of multi-span action recognition

Another important component is the our multi-span action recognition method. Since we don't do human tracking or linking, the performance of action recognition is solely depended on the length of input clip, which faster movement requires fewer frames at a dense stride and slower movement requires more frames at a sparse stride. Shown in table 7.1, by combing input clips with different length of frames at multiple stride, we can further improve the action recognition performance (2.2%). The influence of multiple spans on performance is obvious.

7.3.3 Compared to State-of-the-art

We compare our final framework with multiple state-of-the-art methods [32, 37, 44, 65]. On UCF101-Detection dataset, we out-perform previous best method [44] by 1.5%, shown in Table 7.3. On AVA, we also archive state-of-the-art performance and lead a 0.9%

Method	Frame mAP@0.5
Faster RCNN [76]	11.2
Google [32]	15.6
Ours	**16.5**

Table 7.2: Comparison with state-of-the-art methods on AVA validation set

Method	Frame mAP@0.5
Two-stream RCNN [65]	65.7
ACT-detector [44]	67.1
T-CNN [37]	41.4
Ours	**68.6**

Table 7.3: Comparison with state-of-the-art methods on UCF101-24

improvement over [32], shown in Table 7.2. Considering [32] has used rgb and optical flow from a clip of 40 frames, we think our improvement is significant by just using 16 frames. Overall, our framework is competitive to other state-of-the-art methods, which proves its effectiveness.

7.4 Conclusion

In this work, we present a decoupled framework for spatial temporal human action understanding, which is a hot research topic due to its vast application in video surveillance. The proposed framework has both dedicated branches for human detection and instance action recognition. It can predict where, when and what a human action happens simultaneously. Further experiments demonstrate the effectiveness of our framework.

Chapter 8: Conclusion and Open Problem

The success of large scale image classification has largely accelerated deep architectures to learn better spatial representations for image applications. However, in temporal domain, people were unable to achieve similar performance due to the lack of large-scale datasets. Thanks to the recent released datasets for large scale action classifications, researchers can now train novel architectures to learn better temporal representations for video applications. Meanwhile, such trend raises a new question: Is it possible to find a unified deep architecture that works for both spatial and temporal domain? More formally, given a deep model, the architecture should be invariant to the temporal lengths of videos. When provided with a single image, the network should effectively capture spatial context; When provided a clip of continuous images, the network should capture both spatial and temporal context simultaneously.

So, one interesting open problem is to find a deep architecture for both spatial and temporal domain. People have long-time observed that using image pre-trained deep networks would assist the temporal learning process, as it provided meaningful spatial representations. It is interesting to determine whether networks pre-trained on large scale video classification datasets can further accelerate spatial learning process, as the network may be aware of the potential 3D spatial information by inferencing from temporal information. Exploring such direction can further help find the common connections of spatial and temporal context, thus, benefit both domains.

Bibliography

[1] Thomas Berg and Peter N. Belhumeur. POOF: Part-Based One-vs-One Features for fine-grained categorization, face verification, and attribute estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[2] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001.

[3] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. In *British Machine Vision Conference (BMVC)*, Nottingham, 2014.

[4] Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 568–574. IEEE, 1997.

[5] Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1914–1923, 2016.

[6] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[7] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *ICCV*, 2005.

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3 – 15, 2004.

[10] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *Proc. CVPR*, volume 1, page 4, 2017.

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[12] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[13] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015.

[14] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[15] Xiyang Dai, Joe Yue-Hei Ng, and Larry S Davis. Fason: First and second order information fusion network for texture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7352–7360, 2017.

[16] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S. Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[17] Xiyang Dai, Ben Southall, Nhon Trinh, and Bogdan Matei. Efficient fine-grained classification and part localization using one compact network. In *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*, pages 996–1004. IEEE, 2017.

[18] Achal Dave, Olga Russakovsky, and Deva Ramanan. Predictive-corrective networks for action detection. In *CVPR*, 2017.

[19] Jia Deng, Jonathan Krause, and Li Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[20] 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 *ICML*, 2014.

[21] G. Doretto and Y. Yao. Region moments: Fast invariant descriptors for detecting small image structures. In *CVPR*, 2010.

[22] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016.

[23] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *CoRR*, abs/1606.01847, 2016.

[25] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[27] JC Gemert, Mihir Jain, Ella Gati, Cees GM Snoek, et al. *Apt: Action localization proposals from dense trajectories*. BMVA Press, 2015.

[28] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[29] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[30] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 759–768, 2015.

[31] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[32] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A. Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. 2018.

[33] Ismail Haritaoglu, David Harwood, and Larry S. Davis. W/sup 4: real-time surveillance of people and their activities. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):809–830, 2000.

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

[35] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 229–238. ACM, 1995.

[36] Xiaopeng Hong, Hong Chang, Shiguang Shan, Xilin Chen, and Wen Gao. Sigma set: A small second order statistical region descriptor. In *CVPR*, 2009.

[37] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *ICCV*, 2017.

[38] Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked CNN for fine-grained visual categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[39] Mihir Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. Action localization with tubelets from motion. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 740–747, 2014.

[40] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[41] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[42] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes, 2014.

[43] Xiaokang Yang Ashraf A.Kassim Jun Yuan, Bingbing Ni. Temporal action localization with pyramid of score distribution features. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2016.

[44] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017.

[45] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 817–825, 2016.

[46] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[47] Jonathan Krause, Timnit Gebru, Jia Deng, Li-Jia Li, and Li Fei-Fei. Learning features and parts for fine-grained recognition. In *ICPR*, 2014.

[48] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *CVPR*, 2015.

[49] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Fei-Fei Li. The unreasonable effectiveness of noisy data for fine-grained recognition. *CoRR*, abs/1511.06789, 2015.

[50] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[52] Peihua Li, Xiaoxiao Lu, and Qilong Wang. From dictionary of visual words to subspaces: Locality-constrained affine subspace coding. In *CVPR*, 2015.

[53] D. Lin, X. Shen, C. Lu, and J. Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1666–1674, June 2015.

[54] Tsung-Yu Lin and Subhransu Maji. Visualizing and understanding deep texture representations. In *CVPR*, 2016.

[55] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnns for fine-grained visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015.

[56] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[57] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[58] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1942–1950, 2016.

[59] P. S. M. Mettes, D. C. Koelma, and C. G. M. Snoek. The imagenet shuffle: Reorganized pre-training for video event detection. In *ACM International Conference on Multimedia Retrieval*, 2016.

[60] Pascal Mettes, Jan C van Gemert, Spencer Cappallo, Thomas Mensink, and Cees GM Snoek. Bag-of-fragments: Selecting and encoding video fragments for event detection and recounting. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 427–434. ACM, 2015.

[61] Shakir Mohamed. A statistical view of deep learning, 2015.

[62] Alberto Montes, Amaia Salvador, Santiago Pascual, and Xavier Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. In *1st NIPS Workshop on Large Scale Computer Vision Systems*, December 2016.

[63] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[64] Dan Oneata, Jérôme Revaud, Jakob Verbeek, and Cordelia Schmid. Spatio-temporal object detection proposals. In *European conference on computer vision*, pages 737–752. Springer, 2014.

[65] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *European Conference on Computer Vision*, pages 744–759. Springer, 2016.

[66] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[67] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[68] T. Pfister. *Advancing Human Pose and Gesture Recognition*. PhD thesis, University of Oxford, 2015.

[69] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In *Asian Conference on Computer Vision (ACCV)*, 2014.

[70] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[71] Tania Pouli, Erik Reinhard, and Douglas W. Cunningham. *Image Statistics in Visual Computing*. A. K. Peters, Ltd., Natick, MA, USA, 1st edition, 2013.

[72] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[73] A. Quattoni and A. Torralba. Recognizing indoor scenes. *CVPR Workshop*, 2009.

[74] Rajeev Ranjan, Vishal M. Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *CoRR*, abs/1603.01249, 2016.

[75] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.

[76] 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*, pages 91–99, 2015.

[77] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[78] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[79] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

[80] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 357–360, New York, NY, USA, 2007. ACM.

[81] Greg Mori Li Fei-Fei Serena Yeung, Olga Russakovsky. End-to-end learning of action detection from frame glimpses in videos. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2016.

[82] Kevin J Shih, Arun Mallya, Saurabh Singh, and Derek Hoiem. Part localization using multi-proposal consensus for fine-grained categorization. In *British Machine Vision Conference (BMVC)*, 2015.

[83] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2017.

[84] Zheng Shou, Dongang Wang, and S Chang. Action temporal localization in untrimmed videos via multi-stage cnns. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2016.

[85] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2016.

[86] Gunnar A. Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[87] Gunnar A. Sigurdsson, Olga Russakovsky, and Abhinav Gupta. What actions are needed for understanding human actions in videos? In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[88] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016.

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

[90] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[91] Bharat Singh and Larry S. Davis. An analysis of scale invariance in object detection snip. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[92] Bharat Singh, Tim K. Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[93] Gurkirt Singh and Fabio Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *CoRR*, abs/1607.01979, 2016.

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

[95] Radu Timofte and Luc Van Gool. A training-free classification framework for textures, writers, and materials. In *BMVC*, 2012.

[96] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1653–1660, 2014.

[97] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks, 12 2015.

[98] Aruni RoyChowdhury Tsung-Yu Lin and Subhransu Maji. Bilinear cnns for fine-grained visual recognition. In *ICCV*, 2015.

[99] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV*, 2006.

[100] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[101] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[102] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, May 2013.

[103] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *ICCV - IEEE International Conference on Computer Vision*, pages 3551–3558, Sydney, Australia, December 2013. IEEE.

[104] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[105] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015.

[106] 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.

[107] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[108] R. Wang and D. Tao. Uts at activitynet 2016. *AcitivityNet Large Scale Activity Recognition Challenge*, 2016.

[109] Yaming Wang, Jonghyun Choi, Vlad I. Morariu, and Larry S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *CVPR*, 2016.

[110] S. Xie, T. Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *CVPR*, 2015.

[111] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[112] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, 2015.

[113] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[114] Gang Yu and Junsong Yuan. Fast action proposals for human action detection and search. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1302–1311, 2015.

[115] Jun Yuan, Bingbing Ni, Xiaokang Yang, and Ashraf A Kassim. Temporal action localization with pyramid of score distribution features. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3093–3102, 2016.

[116] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[117] Da Zhang, Xiyang Dai, Xin Wang, and Yuan-Fang Wang. S3d: Single shot multi-span detector via fully 3d convolutional network. In *British Machine Vision Conference (BMVC)*, 2018.

[118] Da Zhang, Xiyang Dai, and Yuan-Fang Wang. Dynamic temporal pyramid network: A closer look at multi-scale modeling for activity detection. *arXiv preprint arXiv:1808.02536*, 2018.

[119] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based rcnn for fine-grained detection. In *13th European Conference on Computer Vision (ECCV)*, 2014.

[120] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *13th European Conference on Computer Vision (ECCV)*, 2014.

[121] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017.

[122] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*. 2014.