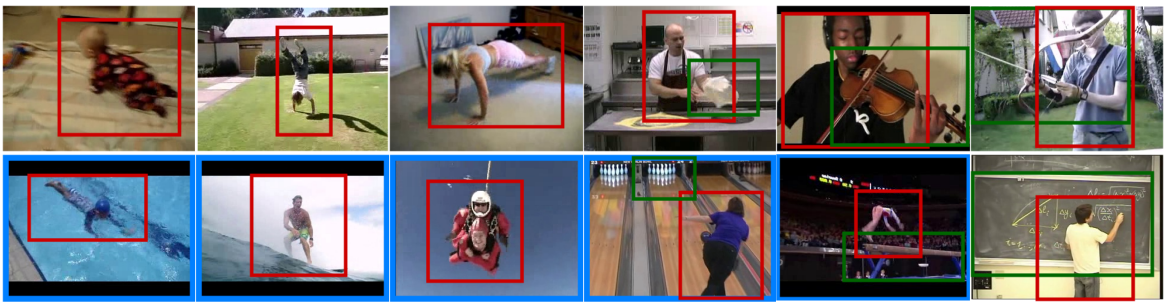
Semantic-Reginal CNNs for Action Recognition



Yifan Wang

Master Thesis
April 2016

Supervisors:
Jie Song
Dr. Limin Wang
Prof. Dr. Otmar Hilliges

ETH zürich

Advanced Interactive Technologies

Abstract

Human action is a high-level concept in computer vision research and understanding it may benefit from different semantics, such as human pose, interacting objects, and scene context.

In this thesis, we explicitly exploit semantic cues with aid of existing object detectors for action recognition in videos, and thoroughly study their effect on the recognition performance for different types of actions.

Specifically, we propose a new deep architecture by incorporating object/human detection results into the framework for action recognition.

Our proposed architecture not only shares great modeling capacity with two-stream input augmentation, but also exhibits the flexibility of leveraging semantic cues (e.g. scene, person, object) for action understanding.

We perform experiments on UCF101 dataset and demonstrate its superior performance to the original two-stream CNN. In addition, we systematically study the effect of incorporating semantic cues on the recognition performance for different types of action classes, and try to provide some insights for building more reasonable action benchmarks and robust recognition algorithms.

Ackowledgement

I would like to express my sincere gratitude to my supervisor Jie Song. His continuous guidance and boundless patience has given me immense inspiration and courage.

My sincere thanks also goes Dr.Limin Wang, whose insightful advice was crucial for the major breakthroughs in this thesis.

The completion of this thesis also owes to the tremendous love and care from my precious friends.

Especially, I would like to thank Srivathsan Murali, Seon-Wook Park for their endless encouragement and unconditioned help. Working beside you has been a great honour and your dedication to work has been a constant motivation for me.
To Pavol Vyhlidal, who taught me never cease to confront personal boundaries and take on new challenges in life.
To Federico Danieli, who has accompanied me through my ups and downs. Your valuable opinions gave rise to numerous progress; your loving support has been the greatest fuel that drove me forward.

Last but not least, I am thankful to my family. You enabled everything, EVERYTHING.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Action Recognition

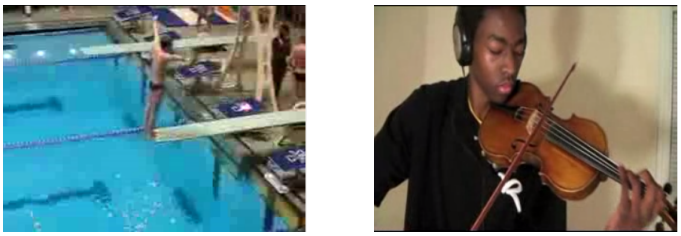
Video based action recognition finds its usage in a broad range of practical applications such as surveillance, video analysis and retrieval as well as human computer interaction. In the recent years, evidently increasing research effort has been devoted in this topic. However, despite the intense research, it remains one of the unconquered fields in computer vision.

The two major milestones are marked with the development of a series of trajectory based feature extractor [47, 48] and the success application of deep neural network in action recognition [41]. The former, typically used in the standard Bag of Visual Words (BoVW) pipeline together with advanced encoding method [35, 34], is broadly used as benchmark for later methods. Meanwhile the latter, the first Convolutional Neural Network (CNN) [28] based approach that yielded competitive performance, stimulated many valuable follow-up proposals with various focuses oriented on improvements for deployment of deep neural network in action recognition [50, 52].

However, as video datasets become more and more challenging, the performance of CNN based methods seem to have reached a bottleneck. In particular, as video clips gets more realistic and the temporal trimming more fuzzy, the definition of "action" demands a second thought.

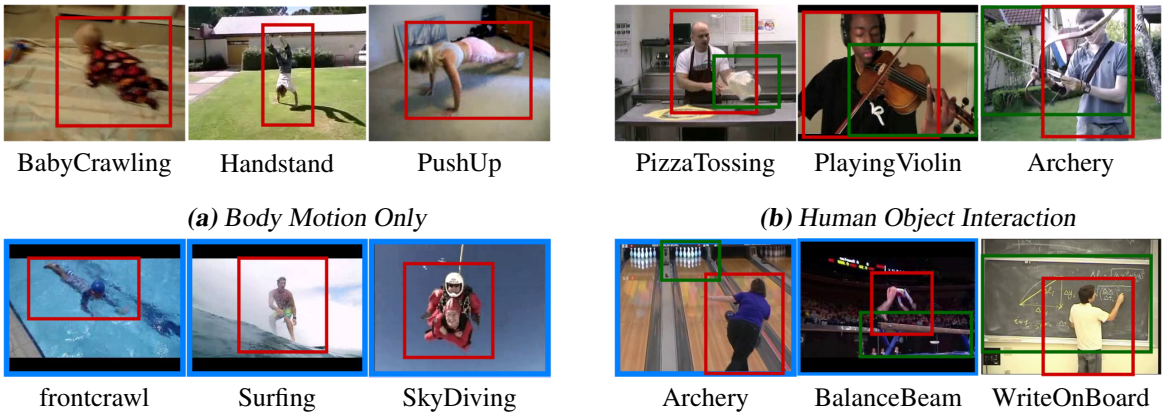
On one hand, an action is a series of events connected in time, recognizing an action often requires consideration of temporal evolution. The original CNN tackles this issue by applying a simple temporal pooling on the frame-wise classification results, which becomes an apparent weakness when dealing with long untrimmed video sequences. In order to tackle this drawback, 3D-CNN [19] as well as LSTM (Long Short Term Memory) Models [4, 54] have been opted to emphasize the evolvement in time. However, given the complexity of their architectures, the performance yielded by such approaches are harshly jeopardized by the inadequacy of available datasets.

On the other hand, the recognition of an action in video can be seen as an interplay of recognizing different components of an event. In fact, for many action categories one does not even need any temporal evolution to classify them; as is shown in Figure 1.1, one can easily classify the whole video sequence given the composition of objects or scene in a single frame. As a



(a) Scene *(b) Object*

Figure 1.1: *Sample frames from UCF101 dataset, where perceptually the appearance of scene or objects is sufficient for correct recognition of an action class.*



(a) Body Motion Only *(b) Human Object Interaction*

(c) Body Motion in Specific Scene *(d) Human Object Interaction in Specific Scene*

Figure 1.2: *Action classes can be grouped into four types: (1) body motion only, (2) human object interaction, (3) human in specific scene, and (4) human object interaction in specific. Classifying each of these categories may require a diverse emphasis on the semantic cues, like human body (red box), interacting objects (green box), and global context (blue box).*

matter of fact, this concept has already been widely implemented and intensely optimised in low-level features, primarily by applying various fusion techniques [33] on descriptors from different cues such as Histogram of Gradients (HOG) [27] for appearance cue and Motion Boundary Histogram (MBH) [48] for motion cue. However on semantic level this field still remains scarce.

1.2 Objective, Challenge and Contribution

As aforementioned, the definition of an action is manifold rather than straightforward. Often a successful recognition process involves diverse factors. Inspired by natural human perceptual procedure, we define three most relevant cues in action recognition, namely: the action performer (person) including his appearance and motion, objects in case of human-objection interaction and the actual scene where the action takes place. In most cases, person should be the most decisive cue, but under certain circumstances, the other two cues could be dominating or even essential. Current CNN-based action recognition methods mostly use the whole frame as input. Consequently, the network is expected to infer the internal relations automatically and

	P	P+O	P+S	P+O+S
class ratio with $< \%50$ acc.	$\frac{4}{21}$	$\frac{3}{25}$	$\frac{3}{28}$	$\frac{1}{27}$
mean Acc.	64.08%	81.32%	81.66%	88.40%

Table 1.1: *UCF action classes with test accuracy lower than* $< \%50$ *(split 1).*

ideally learn to shift its attention to the most informative cue.

Regardless of CNN's great performance in the task of action recognition, it is unclear:

- what the network has actually learned
- whether it's able to abstract the different semantic cues and internalize their relations with respect to the target action

In order to answer these questions, we divide the existing action classes into four sub-categories, namely

1. Body Motion Only (referred as "*P*"), where the action is defined solely by the motion and appearance of the involving person;
2. Human-Object Interaction (referred "*P+O*"), where the action involves a specific object;
3. Body Motion in specific Scene (referred as "*P+S*"), where the action takes place in a discriminative environment;
4. Human Object Interaction in specific Scene (referred "*P+O+S*"), where the action involves both a particular object and a distinct environment.

Examples of this categorization on UCF101[43] dataset is shown in Figure 1.2. (for the complete categorization results of dataset UCF101, please refer to Table A.1 in appendix). This categorization reflects the relative relevance of the three semantic cues in a given action class and will be used as a guideline for analysis through the thesis.

Using this categorization, we evaluated the per category performance on UCF101 and observed significantly inferior test accuracies in categories *P* (see Table 1.1). This indicates that under current datasets, deep neural networks are not yet able to abstract high level semantic structures and is often perturbed by ambiguous information.

This observation motivated us to explicitly incorporate semantic structure into the CNN architecture.

One possible and probably the most straightforward approach is to train separate classifiers for each cue and fuse a final result by applying various late or early fusion methods. While this may indeed yield better result in terms of accuracy, it does so at the cost of generability and scalability, thus certainly not a long-term solution. In contrary we aim to keep our system as concise, general and computational efficient as possible.

Specifically, we propose an integrated deep neural network framework, which combines various detection results (i.e. semantic cues) into a two-stream action recognition pipeline. We use the recently proposed Faster-RCNN [38] as human and common object detector and adapt them

to the video domain by leveraging temporal constraints among a sequence of detection results. These temporally coherent detection results provide semantic information about the activities portrait in the videos, such as the locations of a person and the relations of person and objects over time. We incorporate these semantic cues (detection results) into proposed action recognition pipeline, where detected bounding boxes guide the localization of salient regions in the video stream and enable CNNs to select discriminative features with ROI (Region Of Interest) pooling [10]. In order to fuse semantic channels in an optimal fashion, we propose and evaluate different fusion methods and training strategies to optimize the weights of our deep models. We empirically study the effect of different settings and make recommendations for the best choice.

In summary the contribution of this thesis includes:

1. We proposed a CNN model on weakly annotated video to encode the semantic decomposition of in action cognitive process into the three cues, namely person, interacting objects and scene. We achieve state-of-art performance on UCF101 dataset. Meanwhile by utilizing the complementary properties of the cues, we increase the robustness of our model.
2. In addition, we systematically study the effect of incorporating semantic cues on the recognition performance for different types of action classes as defined previously, and try to provide some insights for building more reasonable action benchmarks and robust recognition algorithms.

1.3 Thesis Outline

- In chapter 2 we start with an detailed overview on action recognition with particular focus on the previous approaches concerning the utilization and leverage of semantic cues.
- chapter 4 provides an overall introduction to our approach.
- The specific implementation details concerning training and testing settings will be summarized in chapter 5.
- Finally in chapter 6, we give a comprehensive analysis based on our empirical study.

2 Related Work

Video-based action recognition has been intensely researched in the recent years owing to its abundant application values. However accurate action classification for realistic video sequences remains a challenging task. First of all, actions exhibit vast variability caused by view angle, background environment, as well as the action execution itself. More importantly, the perception and cognition of an action, even for human, is genuinely involving and complex. Often it requires high level abstraction and inference from other semantic hints. For this reason, effort has been made to use augmented input as complementary information for better classification.

Single Cue For action recognition with, person has always been the most intuitive subject for action recognition. The very first attempts in action recognition used tracks of human body parts [53, 7] or shape of the human body to describe motion. Since these approaches depend on very accurate person or pose detection, they were only suited for artificial settings. As new realistic data were released, they were soon replaced by a variety of low-level local features for their robustness, among which especially noteworthy are feature descriptors such as 3D-SIFT [40], HOF (Histogram of Optical Flow) [27] and MBH (Motion Boundary Histogram) [47] as well as features extractors such as STIP (space-time interesting points) [26] to DT (dense trajectories) [47] and iDT (improved dense trajectories) [48]. These low-level features, although not specifically articulate body motion, focus on points with strong motion signal, hence they are by design intended to characterize body motion.

Cue Augmentation In terms of cue augmentation, object and scene are the most common choice. [32], one of the first attempts, jointly modeled the human and object movement in a belief network. This was deepened in the work by [13], where they inferred object reaction in a graphical model, which in turn is used to discriminate confusion action classes. In [31], the author used augmented data collected from video scripts to infer scene information, which proved to be assisting or even dominating for many action categories. [16] exploited both scene and object cues. In order to handle imperfect human and object extractions, they adopted MIL (Multiple Instance Learning) framework in their system and trained global weights for each cue before performing an early fusion. As a more extreme example, [17] classified each video frame as an object with a very large object/scene classifier trained on image data ImageNet [39].

Using the classification scores as the sole feature, this method achieved competitive results.

One drawback of these aforementioned approach lies in their dependency on reliable object and person detection (except [31] as it used external information source). A few works worked around this issue by incorporating scene and objects in a more subtle way. For example [46] and [37] integrated scene information by grouping local dense features from the whole frame in different regions obtained by applying different segmentation schemes.

Cue Augmentation in Deep Neural Network Contrary to conventional approaches, Deep Neural Network (DNN) is an End-2-End solution that trains a classifier from raw image or video input while simultaneously learning a holistic representation of the input in an unsupervised manner. As is demonstrated in some analytical works such as [55, 30]for image classification task, it is able to abstract high-level semantic features.

Cue fusion in DNN-based method for action recognition has attracted relatively less attention, partially because deep neural network is expected to encode semantic cues implicitly by itself. However, as is often the case, providing explicit information and employing clear structure to DNN model can bring about significant improvement. Indeed, the first victory of deep learning in action recognition by [41] owes to the direct supply of motion structure with optical flow input.

[51] trained 5 individual DNNs (spatial CNN, motion CNN, spatial LSTM, motion LSTM and audio CNN) and subsequently performed a late weighted fusion using externally trained weights. However as this approach trains multiple neural networks, hence very computationally expensive and not well scalable. Moreover learning fusion weights demands an extra validation split from training set, shrinking the already small training data.

Nonetheless a few of works in still-image scene classification and action recognition [52, 12]. Among them R*CNN [12] is most related to our work. In their work they also leveraged an unified neural network architecture to incorporate person and object cues. Our work differs from R*CNN mainly in two folds: (1) no ground truth bounding boxes of human and objects are provided, we use advanced object detectors and efficient post-pocessing to make our system more flexible and general; (2) action recognition in video domain is more challenging, our proposed network has an additional stream with optical flow input.

3 Preliminary

3.1 Deep Learning

Deep learning is a state-of-art machine learning algorithm that implements the concept of neural network. It trains a model designed for a specific task (hence discriminative) while simultaneously learns the feature representations in an unsupervised manner. As will be explained in detail later in this chapter, the parameters in each layer are learned using Stochastic Gradient Descent (SGD) [3], while the gradients are back-propagated from the top layer down to the input data.

CNN (Convolutional Neural Network) is a subgroup of deep learning model intended to emulate the visual cortex of animals. Compared with other neural network structures, CNN is characterized by its sparse local spatial connectivity between adjacent layers. This means each hidden unit in layer m is connected only with a subset of units in layer $m-1$ that have spatially contiguous *receptive field* [25] (illustrated in Figure 3.1). Here, receptive field is a term borrowed from biological visual perception referring to the sub-region on visual field a neuron is sensitive to.

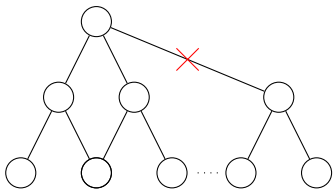


Figure 3.1: *Units of two consecutive layers are only connected if their receptive field is spatially contiguous.*

3.1.1 Fundamental Components in a CNN

A CNN typically comprises of 5 kinds of layers. These layers, equivalent to neurons in a classic neural network, perform a series of linear and non-linear operations on the input image. The output of these layers is called *feature map*.

Convolutional Layer

Convolutional layer is the core of a CNN. It applies a linear shift-invariant filter, called *kernel* on the input feature. Figure 3.2 (source [1]) illustrates a convolutional operation with a 3×3 kernel on a single input channel. This operation is repeated over the whole spatial dimensions as well as all input channels. Mathematically, we can write a 2D convolution on input $\mathbf{x}$ with kernel $\mathbf{w}$ at point (w, h) as

$$\mathbf{w} * \mathbf{x}(w, h) = \sum_{q=-\frac{1}{2}(c-1)}^{\frac{1}{2}(c-1)} \sum_{p=-\frac{1}{2}(c-1)}^{\frac{1}{2}(c-1)} \mathbf{w}(p, q)\mathbf{x}(w - p, h - q),$$

where c denotes the kernel size.

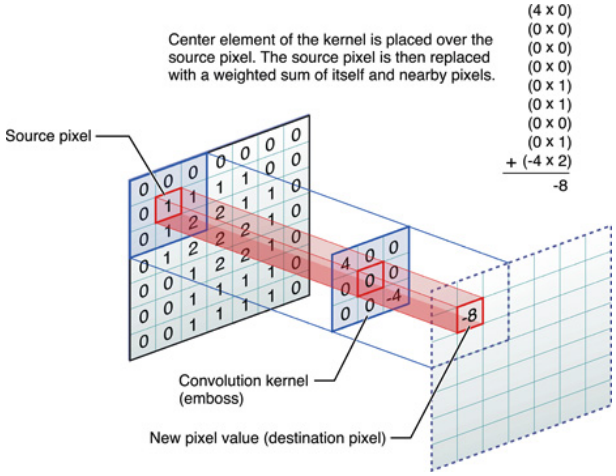


Figure 3.2: *Illustration of a 2D convolutional operation using a single kernel of size 3 on a input layer*

A convolutional kernel is associated with each input channel and each output channel. Formally, given a input feature map $\mathbf{x}$, the k'-th channel in output can be expressed as

$$h[w, h, k'] = b_{k'} + \sum_{k \in K_{in}} \mathbf{x}[w, h, k] * \mathbf{w}_{k'k}.$$

$\mathbf{b}$ and $\mathbf{w}$ together make up $K' \times K \times c \times c$ parameters for a convolutional layer.

Since the values of each kernel $\mathbf{w}_k$ stay unchanged over the whole feature map, the total number of parameters are greatly reduced. This improves the learning efficiency as well as the generalization property of the network. Meanwhile it also undertones that the local response to a pattern is independent of its spatial location.

Pooling Layer

Pooling layers are used to aggregate input feature in a region to reduce its dimension. Furthermore, since small translation or perturbation in the input within the pooling region does not affect the output, pooling also introduces certain degree of translation-invariance.

The pooling operation is parameterized by stride and kernel size. An example of a MaxPooling is illustrated in Figure 3.3 (source [21]).

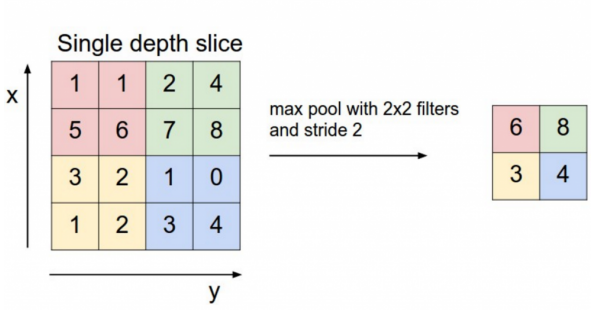


Figure 3.3: *Illustration of a MaxPooling operation with kernel size 2 and stride 2*

Rectified Linear Unit Layer

Rectified Linear Unit (ReLU) layer is one of the activation functions that simulate switches in neurons by performing a fixed non-linear operation on the input signal.

ReLu performs a simple threshold

$$f(x) = \max(x, 0).$$

Compared to other activation functions, such as $sigmoid$ and $tanh$, ReLU exhibits better qualities in accelerating the convergence rate [23] and leads to better results [29].

Fully Connected Layer

Fully connected layer, also known as inner product layer, performs an inner product between weight vector $\{w_k\}_1^{K'}$ and all channels of the input data. Given an input $\mathbf{x}$ of size $K \times H \times W$, the output is a vector of length K', whose value at position k' is

$$\mathbf{h}[k'] = b_{k'} + \sum_{w,h,k} \mathbf{x}[w, h, k]\mathbf{w}_{k'}[w, h, k].$$

As a result, fully connected layers have $K' \times K \times H \times W$ parameters.

DropOut Layer

DropOut [15] is a recent invention in deep learning. It addresses the issue of overfitting by randomly dropping, i.e. zero-setting, features together with their connections in training. Intuitively, this forces the units to be independent of each other and hence prevent units from excessive co-adapting.

During training a unit is dropped with probability q (*dropout rate*), i.e.

$$f(x) = \begin{cases} 0, & \text{if } p \leq q \\ x, & \text{otherwise} \end{cases}.$$

The resulting feature is then multiplied by a factor of $\frac{1}{q}$ to preserve the scale of output (thus also input of the upper next layer), which is particularly important in fine-tuning an pre-trained model.

Dropout proves to be very effective for tasks that inhabit dense input features and outperforms conventional methods tackling this issue, such as $L1$ or $L2$ weight penalties [44].

3.1.2 Parameter Learning via Back-propagation

Training a deep neural network, supervised or not, works by optimizing a target function.

For an supervised K-class classification problem with labeled data $\mathcal{X} = \{\mathbf{x}_i, \hat{y}_i\}_{i=0}^{N}$ and predictions $\{y_i\}_{i=0}^{N}$, this target function is often defined as the softmax loss:

$$\begin{aligned} L(\mathbf{w}; \mathcal{X}) &= \frac{1}{N} \sum_{i=0}^{N-1} L(\mathbf{w}; \mathbf{x}_i, y_i) \\ &= -\frac{1}{N} \sum_{i=0}^{N-1} \log \Pr(y_i = \hat{y}_i | \mathbf{x}_i, \mathbf{w}) \\ &= -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{K-1} \{\hat{y}_i = j\} \log p_{y|x}(j | \mathbf{x}_i, \mathbf{w}), \end{aligned} \tag{3.1}$$

where $\{\hat{y}_i = j\}$ is the indicator function that equals 1 when the predicate is true, namely:

$$\{\hat{y}_i = j\} = \begin{cases} 1 & \text{if } \hat{y}_i = j \\ 0 & \text{otherwise} \end{cases}.$$

$p_{y|x}(j | \mathbf{x}_i, \mathbf{w})$, the probability that $\mathbf{x}_i$ is classified as j, is computed with softmax function:

$$p_{y|x}(j | \mathbf{x}_i, \mathbf{w}) = \frac{\exp h_j(\mathbf{x}, \mathbf{w})}{\sum_{j=0}^{K-1} \exp h_j(\mathbf{x}, \mathbf{w})},$$

where $h_j(\mathbf{x}, \mathbf{w})$ denotes the classification score of $\mathbf{x}_i$ for class j computed by the network.

The objective of training a network is to find the parameters $\mathbf{w}$ that minimize the softmax loss, i.e.

$$\mathbf{w} = \arg\max L(\mathbf{w}; \mathcal{X}).$$

Stochastic Gradient Descent In Deep Learning, this optimization problem is solved using the idea of *gradient descent*. Gradient descent is a simple learning algorithm that updates the parameters $\mathbf{w}$ iteratively along the direction that induces the steepest descend in L, which is effectively the negative gradient of L wrt $\mathbf{w}$. The magnitude of update is determined by a hyperparameter *learning rate* η. Formally, the parameter update at step t can be written as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial L}{\partial \mathbf{w}}.$$

The gradient, computed as

$$\frac{1}{N}\sum_{i=0}^{N-1}\frac{\partial L(\mathbf{w};\mathbf{x}_i,y_i)}{\partial \mathbf{w}},$$

is very expensive to evaluate over the whole data samples, hence it is often approximated with a subset of samples (called *batch*) $\{\mathbf{x}_m, y_m\}_{m=0}^{M}$, where $M \leq N$ denotes the number of samples in a batch, also referred to as *batchsize*. In other words,

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{1}{M}\sum_{m=0}^{M}\frac{\partial L(\mathbf{w};\mathbf{x}_m,y_m)}{\partial \mathbf{w}}.$$

The approximation formulates an important variant of gradient descent, namely *stochastic gradient descent* (SGD).

Furthermore, to improve the convergence rate and stability of the algorithm, instead of using the gradient as update value only, the parameters are updated with a linear combination of the gradient and the previous update value $\mathbf{v}_t$, i.e.

$$\mathbf{v}_{t+1} = \mu\mathbf{v}_t - \eta\frac{\partial L}{\partial \mathbf{w}} \tag{3.2}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_{t+1}. \tag{3.3}$$

By doing so, each update is smoothened across previous iterations, which pushes the parameters towards the global minimum even under oscillating gradients [36]. The hyperparameter μ is called *Momentum*, it indicates the amount of influence from the previous iterations and effectively increases the magnitude of update.

SGD is the most basic yet most commonly used policy for parameter learning, while several variations exist that improve this general form with different focus (e.g. Adaptive Gradient [5] and Adaptive Momentum [22]).

Backpropagation In deep learning, the mapping from input $\mathbf{x}$ to the final output y is composed of many layers of linear and non-linear functions. In order to calculate the $\frac{\partial L}{\partial \mathbf{w}}$ in Equation 3.2, one needs to apply the chainrule in basic calculus. If $y = f(h)$, while h is a function of x, i.e. $y = f \circ h(x)$, chainrule gives the following formula to compute the derivative $\frac{\partial y}{\partial x}$:

$$\frac{\partial y}{\partial x} = \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}.$$

Essentially *Backpropagation* is nothing else but applying chainrule sequentially top-to-down from the final loss layer to compute the gradients of the loss function w.r.t. all parameters.

Assume we know the gradient of loss function L wrt to the input feature map of l-layer $\frac{\partial L}{\partial \mathbf{x}_l}$, where $\mathbf{x}_l$ is computed from layer $l-1$

$$\mathbf{x}_l = h_{l-1}(\mathbf{x}_{l-1}, \mathbf{w}_{l-1}).$$

We can easily calculate two partial derivatives

$$\frac{\partial L}{\partial \mathbf{w}_{l-1}} = \frac{\partial L}{\partial \mathbf{x}_l}\frac{\partial h_{l-1}}{\partial \mathbf{w}_{l-1}}, \quad \frac{\partial L}{\partial \mathbf{x}_{l-1}} = \frac{\partial L}{\partial \mathbf{x}_l}\frac{\partial h_{l-1}}{\partial \mathbf{x}_{l-1}}. \tag{3.4}$$

We refer the two derivatives as *param derivatives* and *data derivatives*. While the former is the desired gradient in Equation 3.2, the latter is propagated down to the lower layers, where this process will be repeated to compute all remaining param derivatives.

To summarize, during backpropagation each layer of the network takes the data derivatives from the upper layer, computes param derivatives and data derivatives using Equation 3.4 and outputs data derivatives to the lower layers.

3.1.3 Feature Map

Feature map refers to the output of a convolutional layer (typically after activation layer such as ReLu). As this naming indicates, a feature map conveys two pieces of information, namely:

1. what is the feature, encoded in the actual values or activation state of each specific channel in the feature map
2. where is the feature, encoded in the spatial location of an activation.

As is revealed in [55], each channel of a feature map represents a particular pattern. An activation in a specific channel at a specific location essentially describes the existence of a type of pattern at the corresponding location in the input image. In lower layers, the encoded patterns are usually comprised of low level features, such as edges and corners. As one proceeds deeper in the network, they become more abstract and discriminative. In an object classification model trained for ImageNet, the feature map in the last convolutional layer can be interpreted as representations of different object classes.

A very explanatory example (shown in Figure 3.4) is given by [14], where the relation between channel, pattern and location is comprehensively illustrated.

Moreover, as is pointed out by [14], while the spatial position explicitly encodes the location of a given pattern, the feature value itself also provides *finer* position information. This is concluded from the fact that from the visualization of a single activation one can extract the "outline" of the targeted pattern (see Figure 3.5 given by [55]). As a result, one can utilize the feature values across all channels to obtain refined localization information of a composed object.

To summarize, both classification and localization information are present in feature maps; while the former is expressed in the activated feature values itself, the latter is encoded in both the feature values and the spatial position of activations.

3.2 Dynamic Programming

Dynamic Programming is a powerful optimization algorithm. Applicable in both discrete and continuous, deterministic and stochastic domains, it is a tool for many complex real-world

Figure 3.4: *Left: the 66-th channel of a sample feature map in conv5 layer. Right: Other image samples that generate high response in this channel. Simply put, a high response in this channel indicates there is a "λ" shaped object at the corresponding position in input image.*

Figure 3.5: *Reconstruction from a single activation from two different channels in conv5 feature map (left) and the corresponding input pattern (right). Reconstructions from the same channel shows subtle differences in outline, implying that the activation value conveys refined locational information.*

problems. In our thesis, we use its finite discrete deterministic form to find the actual action performer from raw person detections.

Discrete deterministic Dynamic Programming constitutes the following problem setup:

- given the dynamics of a deterministic discrete time (in-)variant system defined in a finite time steps N

$$\mathbf{x}_{n+1} = f_n\left(x_n, u_n\right) \text{ with } x_n \in \mathcal{X}_n,\ u_n \in \mathcal{U}_n,\ n \in \{0, 1, \ldots N-1\},$$

  where $\mathcal{X}_n$ and $\mathcal{U}_n$ denote the set of possible state and control input at time n;

- given the cost of applying input u_n on state x_n at time step n

$$g_n\left(x_n, u_n\right)$$

- find the optimal policy π^* such that

$$\begin{aligned}\pi^* &= \arg\min J \\ &= \arg\min_{\pi} \sum_{n=0}^{N} g_n\left(x_n, u_n\right) + g_T\left(x_N, T\right) \\ &= \arg\min_{\{\mu_0, \mu_0, \ldots, \mu_{N-1}\}} \sum_{n=0}^{N} g_n\left(x_n, \mu_n\left(x_n\right)\right) + g_N\left(x_N\right)\end{aligned} \tag{3.5}$$

  where μ_n is a state dependent single step time-(in)variant strategy $u_n = \mu_n\left(x_n\right)$. Notice although system dynamics does not appear in Equation 3.5, it is integrated with the time increment.

Directly solving Equation 3.5 would require evaluation of all possible connections. For a fully connected system (every state in the current step is reachable from every state in the previous step), if the average number of states in a stage is K, the computations complexity would be $\mathcal{O}\left(K^N\right)$.

Dynamic Programming provides an efficient way to solve this problem. Instead of evaluating the total cost from all stages, Dynamic Programming considers the minimal cost from x_k to terminal node T, called *Cost-to-Go*

$$J_k(x_k) = \min_{\pi_k(\mathbf{x}_k)} \sum_{n=k}^{N} g_n\left(x_n, \mu_n\left(x_n\right)\right) + g_T, \tag{3.6}$$

where $\pi_k\left(x_k\right)$ denotes a policy from the particular node x_k at step k. It should be noticed that per definition, the optimal total cost J^* from step 0 to step N can be written as the Cost-to-Go at step 0 minimized over all state $x_0 \in \mathcal{X}_0$.

$$\begin{aligned}J^* &= \min_{\pi_0} \sum_{n=0}^{N} g_n\left(x_n, \mu_n\left(x_n\right)\right) + g_T\left(x_N, T\right) \\ &= \min_{x_0} \left(\min_{\pi_0(\mathbf{x}_0)} \sum_{n=0}^{N-1} g_n\left(x_n, \mu_n\left(x_n\right)\right) + g_T \right) \\ &= \min_{x_0} J_0(x_0).\end{aligned} \tag{3.7}$$

According to the *Principle of Optimality* ([2]), assuming we arrive at state x_k with an optimal policy $\pi^* = \{\mu_n^*\}_{n=0}^N$ and considering the subproblem from x_k to T, then the truncated policy $\pi_k = \{\mu_n^*\}_{n=k}^N$is optimal for the subproblem.

As a result, the optimal policy π_k^* for Equation 3.6 can be divided into the single-step optimal strategy $\mu^*(x_k)$ and $\pi_{k+1}^*\left(x_{k+1}^*\right)$, where $x_{k+1}^{\mu*}$ is a shorthand notation for the destination state at step $k+1$ if we apply the optimal control input $\mu^*\left(x_k\right)$ on state x_k, i.e. $f_k\left(x_k, \mu^*\left(x_k\right)\right)$. Hence

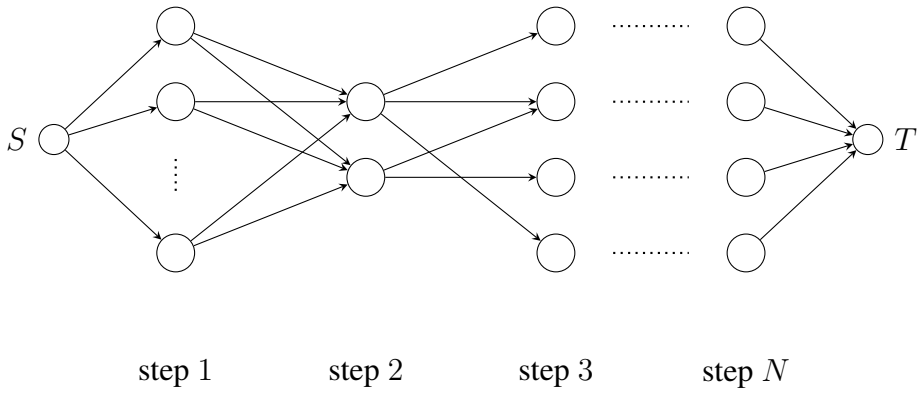


Figure 3.6: *General Shortest Path Problem*

Equation 3.6 can be reformulated in a recursive form

$$\begin{aligned} J_k(x_k) &= \min_{\pi^k(x_k)} \sum_{n=k}^{N} g_n + g_T \\ &= \min_{\pi_k} \left(g_n + \sum_{n=k+1}^{N} g_n + g_T \right) \\ &= \min_{\mu_k} \left(g_n + \min_{\pi_{k+1}\left(x_{k+1}^{\mu}\right)} \left(\sum_{n=k+1}^{N} g_n + g_T \right) \right) \\ &= \min_{\mu_k} \left(g_n + J_{k+1} \left(\mu_k(x_k) \right) \right). \end{aligned} \tag{3.8}$$

In other words, in order to compute the total optimal cost, we only need to evaluate Equation 3.8 at each state x_k of every time step from step N to step 0. Thus in a fully connected system, the total complexity is $\mathcal{O}\left(NK^2\right)$.

In the special case where the system dynamic simplifies to be the assignment of the next state, this setup degenerates to be the Shortest Path Problem as illustrated in Figure 3.6. In this case, each node represents a state in its stage and connection cost a_{ij}^nbetween node i in stage n and j in stage $n+1$ constitutes the step cost $g_n(x_n, u_n)$.

It is worth noting, however, that in practice the core advantage of Dynamic Programming in terms of computational efficiency lies in the fact, that it significantly reduces repeated computing at the moderate expense of memory storage. Since in each iteration the algorithm computes the optimal strategy at each state, once these intermediate results are cached properly, one can obtain the desired control input at any given state or time by simply looking up the computed value. This property is especially favourable when dealing with a complex problem, where it is necessary to re-evaluate certain states multiple times.

4 Methodology

We choose deep learning for our task. In contrast to conventional vision learning methods, deep learning methods not only learn a classification model but also a holistic representation of the raw image or video input, which liberates the user from exhaustive pre- and post-processing, hence opens the door to a series of online applications for this field of study. Moreover, deep neural networks are able to extract high level semantic features, which is critical in dealing with abstract concepts like action.

In order to separate semantic structures, we adopt the method in R*CNN [12], which utilizes the concept of regional pooling initially proposed by [10].

While ground truth person bounding boxes is available in R*CNN, we only have action labels for each video sequence. We employ state-of-art object detector Faster-RCNN [38] for person and object detection.

To handle detection failure due to motion blur, extreme pose variation or occlusion and additionally filter out persons irrelevant to the action, we use Dynamic Programming on frame-wise detections to estimate a smooth person trajectory through the video sequence.

Finally we train our network using settings and techniques suggested in [49] and delve into various fusion methods to maximize the complementary effect of different cues.

The rest of this chapter is organized as follows:

- in section 4.1 we give a comprehensive insight into our model architecture;
- in section 4.2 we provide an overview on Faster-RCNN as well as a recap over our post-processing on raw detection results.
- finally in section 4.3 we propose various fusion methods and describe the motivations behind each of them.

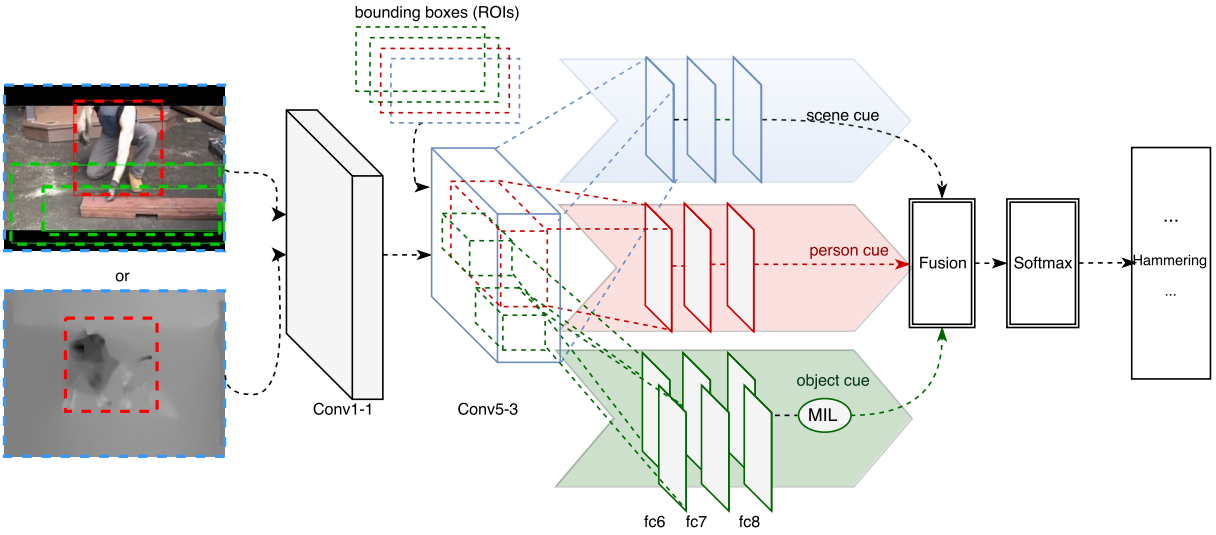


Figure 4.1: *The general architecture of our network. Semantic cues (scene, person and objects) are integrated as pooling regions to create explicit semantic structures (parallel fc layers). The classification results from each semantic channel are merged in a fusion unit as well be discussed in section 4.3.*

4.1 Model Architecture

The general setting of our network architecture is illustrated in Figure 4.1. It is composed of three parts.

The first part of our model is comprised of very deep convolutional layers. We adopt VGG16 configuration proposed by [42]. This publicly available network is trained on ImageNet data for the ILSVRC2012 [39] challenge and has been used as initialization network for various tasks. A total of 13 convolutional layers and 3 fully connected layers are implemented. For clarity, its specifications are recapped in Table 4.1.

4.1.1 Semantic decomposition using RoiPooling

The second part of the network starts with a RoiPooling (Region Of Interest Pooling) layer ([10]), which replaces the original MaxPooling after conv5-3.

Compared to regular pooling layers, which operate on the whole spatial span, the RoiPooling layer works in regions given at runtime. More concretely, it takes a series of rectangular regions as input and generates an independent feature map for each ROI by performing MaxPooling within the spatial area defined by each of these ROIs.

Since the ROIs take on various sizes, a conventional pooling operation would output feature maps of different sizes. However the consecutive fully connected layer requires fixed input size. In order to guarantee uniform output size, RoiPooling layer adopts adaptive bin size. As is illustrated in 4.2 (modified from [14]), given a ROI of size $W \times H$ and the output feature map specified to be $w \times h$ (in our case 7×7), RoiPooling first evenly divides the given ROIs into

Layer	Param	Effective Receptive Field
Conv1-1 Conv1-2	$k = 2 \times 2, \quad n = 64,$ $p = 1, \qquad s = 1$	3
MaxPool	$k = 2 \times 2, \quad s = 2$	6
Conv2-1 Conv2-2	$k = 3 \times 3, \quad n = 128,$ $p = 1, \qquad s = 1$	14
MaxPool	$k = 2 \times 2, \quad s = 2$	20
Conv3-1 Conv3-2 Conv3-3	$k = 3 \times 3, \quad n = 256,$ $p = 1, \qquad s = 1$	44
MaxPool	$k = 2 \times 2, \quad s = 2$	56
Conv4-1 Conv4-2 Conv4-3	$k = 3 \times 3, \quad n = 512,$ $p = 1, \qquad s = 1$	104
MaxPool	$k = 2 \times 2, \quad s = 2$	128
Conv5-1 Conv5-2 Conv5-3	$k = 3 \times 3, \quad n = 512,$ $p = 1, \qquad s = 1$	224
MaxPool	$k = 2 \times 2, \quad s = 2$	272
fc6 Dropout	$n = 4096, \quad r = 0.5$	-
fc7 Dropout	$n = 4096, \quad r = 0.5$	-
fc8	$n = 1000$	-
softmax		

Table 4.1: *Configuration of Conv1 - Conv5 layers in VGG16. k, n, p, s denote denote kernel size, number of output channels and padding and stride respectively. For the sake of brevity, ReLu layers, attached after every convolutional and fully connected layer, are omitted in the table.*

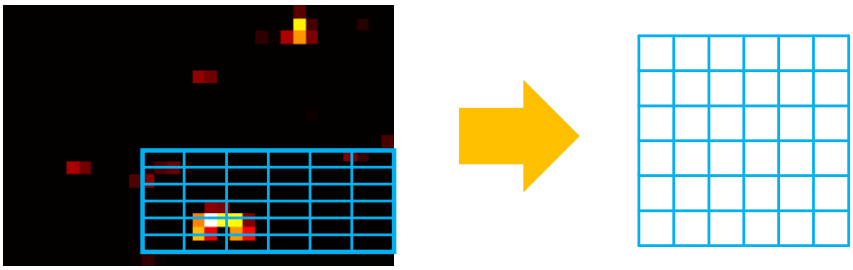


Figure 4.2: *RoiPooling layer performs MaxPooling in regions of arbitrary size to a fixed size feature map by adapting the bin sizes.*

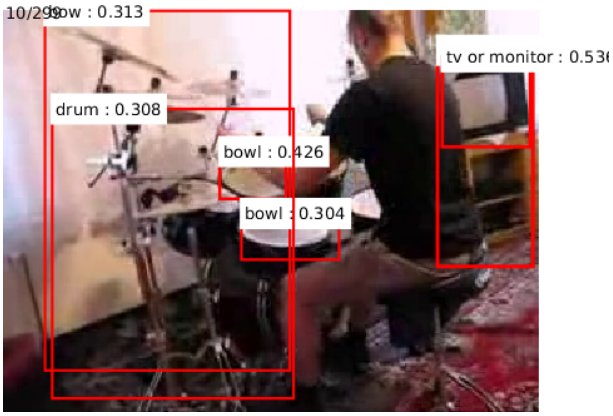


Figure 4.3: *Example where the key object is mixed among many irrelevant objects.*

bins of size $\lfloor \frac{W}{w} \rfloor \times \lfloor \frac{H}{h} \rfloor$ and then performs MaxPooling in each bin.

In accordance with our previous analysis, we group the ROIs into three categories, namely scene, person and objects. Depending on the category of ROIs, their feature maps are to be fed into one of the three fully connected branches representing each of the three cues respectively. The underlying idea is simple: since the contribution of different semantic components are distinct, they should be handled separately. By leveraging the spatial correspondence between feature maps and input images, we extract each cue on feature level, so as to train a more specialized classifier for each of them.

As for the ROI inputs, we use an external object detector (see section 4.2) to generate person and object ROIs, while scene ROI is simply set to be the full input span $(0, 0, W, H)$, in which case RoiPooling is equivalent to a regular MaxPooling.

For each input frame I, exactly one person ROI and one scene ROI are used, while a variable number of ROIs are accepted for the object cue. This is due to the fact that in current datasets the defined action classes mostly involve only a single protagonist, while the contributing objects may be, and often are, multiple.

4.1.2 Object Channel with Multiple Instance Learning

Often the decisive object, if detected at all, is mixed among many irrelevant detections (see Figure 4.3). This described problem matches basic scenario in Multiple Instance Learning (MIL).

In MIL, one is given bags of samples, called instances. Unlike in conventional supervised learning problems, where each instance is labeled individually, labels are assigned to *bags of* instances. In the standard MIL assumption, a bag (and all its containing instances) is labeled positive, as long as there exists one positive instance in the bag; conversely, a bag is labeled negative, if all containing instances are negative. The goal in MIL is to induce the underlying concept that will label individual instances correctly. In our case, bags translate to video frames, and instances are the detected objects. Considering the previous example shown in Figure 4.3, all objects are labeled as "Drumming", although only "Drum" is the conclusive instance.

MIL has its own established field of study and it is not straightforward to transfer it directly to deep learning framework. While an abundance of algorithms has been developed, which implicitly or explicitly generalize the underlying MI assumption ([9]), they commonly leverage the most representative instance within a bag to encode the similarity of the bag to a candidate concept. This similarity measurement is used to (a) find the desired concept alternately or (b) create a new feature space for bag representation, upon which a classifier can be trained to predict new bags.

In the scope of this thesis, we adopt the approach proposed by Gkioxari, which took inspiration from this aforementioned idea. In their work [12], they combine multiple context regions, called secondary regions, using a MIL layer. For an input frame and a set of objects O, the MIL layer computes the classification score for class c as:

$$s\left[c\right] = \max_{o \in O} s_o\left[c\right], \tag{4.1}$$

where $s_o\left[c\right]$ denotes the classification score of object o ROI for class c. In other words, for each class the classification score of a frame (bag) is given by the object (instance) that is most representative for target class. The back-propagation works similar to a normal MaxPooling layer. Derivatives from the upper layer will be passed down to (and only to) the instance from which the value was pooled, i.e.

$$\left[\frac{\partial L}{\partial x_i}\right]_c = \left\{i = \arg\max_n x_n\left[c\right]\right\} \left[\frac{\partial L}{\partial y}\right]_c, \tag{4.2}$$

where $\left[\frac{\partial L}{\partial x_i}\right]_c$ and $\left[\frac{\partial L}{\partial y}\right]_c$ denote the derivatives for class c and $\{i = \arg\max_n x_n\left[c\right]\}$ indicates whether the classification score for class c is pooled from this instance i. From the validation results reported in [12], their model incorporating this MIL layer empirically improved the classification performance in PASCAL VOC action recognition challenge, although in-depth analysis over the influence of MIL is not provided.

4.1.3 Fusion

Finally, an independent classification score is produced by each branch of the fully connected layers. As the final part of our model, a merging unit combines them into a single classification score, which will be passed through a Softmax layer to obtain the actual class probabilities. Whereas R*CNN simply used sum merging, we implemented and empirically evaluated various of merging methods including sum, max, multiloss and weighted sum. The detail of each variation will be given in chapter 5, while the results and analysis will be presented in chapter 6.

4.2 Human Object Detection

In this section, we describe our method for extracting reliable person and object bounding boxes.

4.2.1 Object Detector Faster-RCNN

The bounding boxes for objects and persons are extracted using the state-of-the-art object detector Faster-RCNN (Regional Convolutional Neural Network)[38]. Faster-RCNN is purely CNN-based state-of-the-art object detector that provides an end-to-end near real-time solution for object detection.

Faster-RCNN is an upgrade of Fast-RCNN. Compared to the latter, it significantly improves test-time speed by integrating regional object proposal into the system. As can be seen in the Figure 4.4, Faster-RCNN consists of two parts. The first part, *Regional Proposal Net* (*RPN*) takes an image as input and generates as output a set of rectangular regions with high objectness, called *proposals*. The second part of the system, *Detection Net* (DN), uses these proposals to compute the object location and predict the object class.

In both parts, object locations are parameterized as a 4-tuple $\mathbf{t}$, which specifies a scale-invariant translation and log-space height/width shift relative to a reference box a [11]. More specifically, the parameterization $\mathbf{t} = (t_x, t_y, t_w, t_h)$ is defined as

$$\begin{aligned} t_x &= (x - x_a)/w_a \\ t_y &= (y - y_a)/y_a \\ t_w &= \log(w/w_a) \\ t_h &= \log(h/h_a), \end{aligned} \tag{4.3}$$

where x, y, w and h denote the center coordinates, width and height of a bounding box.

While both parts can be deployed as a standalone network for its own task, Faster-RCNN proposes a unified network that further improves test-time speed. More concretely, RPN and DN share the same convolutional layers. At test-time, an input image only needs to go through the convolutional layers once. The output of the last shared convolutional layer is first used in RPN for proposal generation and then, together with RPN's output, in DN for detection.

The convolutional layers adopt the architecture in VGG16 (see Table 4.1). The total stride at the last convolutional layer adds up to 16, i.e. an input image of the size $W \times H$ will generate a $\lceil \frac{W}{16} \rceil \times \lceil \frac{H}{16} \rceil \times K$ feature map, where K denotes the number of channels and equals 512 in VGG16.

RPN The scheme of proposal generation is depicted in Figure 4.5 (slightly modified from source [38]). It takes an input image and generates a set of rectangular regional proposals and 2-tuples corresponding to their binary (foreground/background) prediction.

To generate proposals, a lower-dimensional feature vector $\mathbf{h} \in \mathbf{R}^d$ is first computed from a small $n \times n$ neighbourhood of the convolutional feature map (ref the sliding window in Figure 4.5) via inner product. Formally, for an input feature map $\mathbf{x}$ and weight $\mathbf{w}$, the d-th entry of

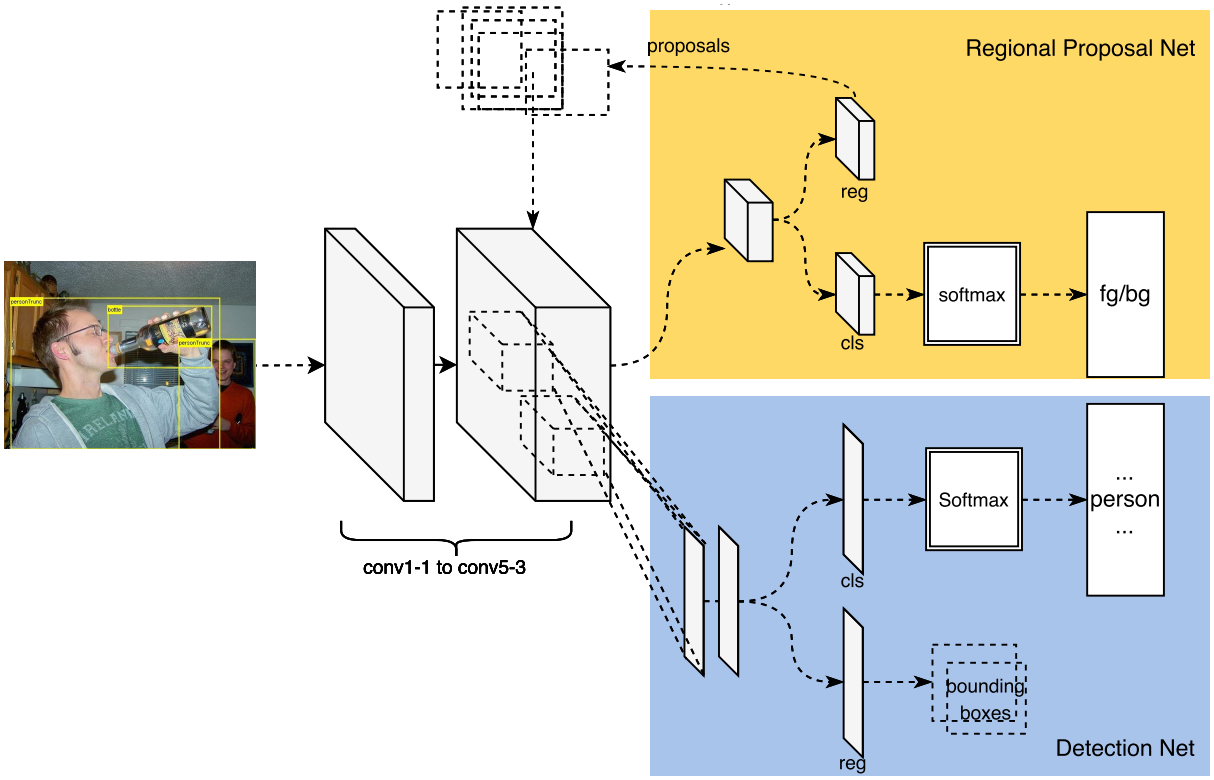


Figure 4.4: *Architecture of Faster-RCNN.*

the output lower-dimensional vector is computed as

$$\mathbf{h}\left[d\right] = \sum_{(w,h)\in W,\, k} \mathbf{x}\left[w,h,k\right] \mathbf{w}_d\left[w,h,k\right], \tag{4.4}$$

where W is the sliding window, w, h and k denote the width, height and channel index respectively.

The resultant feature vectors form representations of the corresponding image patch in the input image. These feature vectors are then fed into two branches of fully connected layers, *cls* and *reg*, for classification (foreground/background) and regression (coordinates of bounding boxes) respectively.

To be specific, at each position of the feature map, k proposal bounding boxes are generated. Those are parameterized relative to k "anchors", which serve as references for the actual proposals. Anchors are composed of rectangular regions of various scales and ratios so as to detect objects under scale and size variation. Faster-RCNN implements 3 scales and 3 ratios, yielding a total of 9 anchors at each position. In the fully connected layers, k inner product are carried out independently so that the predictions for each anchor can be computed simultaneously, i.e.

$$\mathbf{y}_{cls}[i,k] = \mathbf{h}^T \mathbf{w}_{ik}^{cls}, \tag{4.5}$$

where $i \in \{0,1\}$ denotes bg and fg labels and

$$\mathbf{y}_{reg}[i,k] = \mathbf{h}^T \mathbf{w}_{ik}^{reg}, \tag{4.6}$$

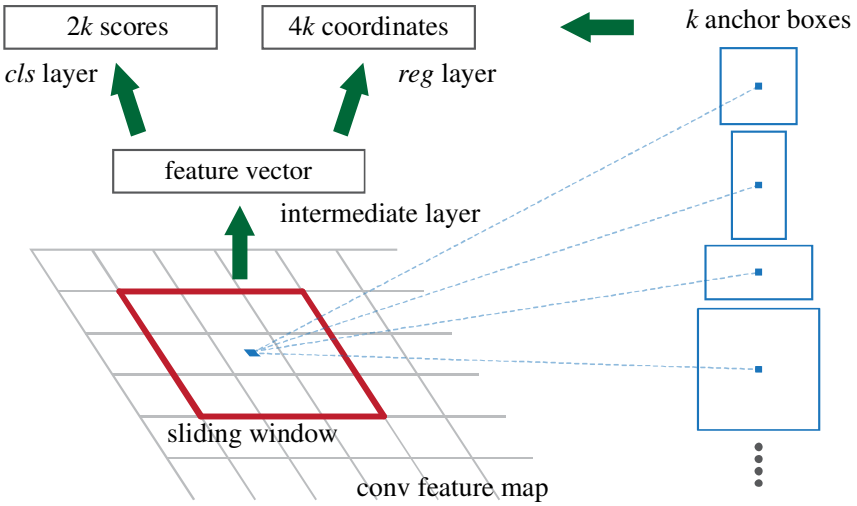


Figure 4.5: *Proposal generation in Faster-RCNN (slightly modified from source [38])*

where $i \in \{0, 1, 2, 3\}$ denotes the four parameters defining the location of bounding boxes.

On implementation level, since the weights in Equation 4.4-Equation 4.6 are shared spatially across all anchor positions, these two steps of inner product can be efficiently realized using two consecutive convolutional layers with kernel size n and 1 respectively.

The use of anchor is a novel solution to address multi-scale problem. By creating an individual reference for each scale-ratio combination, it makes it possible to train multi-scale filters simultaneously on a single feature, which serves as the key ingredient for RPN's computation efficiency.

RPN adopts multi-task training same as in DN, i.e. the loss function is a weighted sum of the classification loss L_{cls} and regression loss L_{reg}. With y_i, $\hat{y}_i$, $\mathbf{t}_i$ and $\hat{\mathbf{t}}_i$ denoting the predicted class, groundtruth class, predicted object location (defined as Equation 4.3) and the groundtruth object location of an anchor i, the loss of the network can be written as

$$L = \frac{1}{N_{cls}} \sum_i L_{cls}\left(y_i, \hat{y}_i\right) + \lambda \frac{1}{N_{reg}} \sum_i \hat{y}_i L_{reg}\left(\mathbf{t}_i, \hat{\mathbf{t}}_i\right). \tag{4.7}$$

The classification loss L_{cls} is the binary softmax loss (see Equation 3.1). The regression loss L_{reg} is the smoothed L1 loss introduced in [10]. It is defined as

$$L_{reg}\left(\mathbf{t}, \hat{\mathbf{t}}\right) = \sum_{d=0}^{4} f\left(\mathbf{t}\left[d\right] - \hat{\mathbf{t}}\left[d\right]\right), \tag{4.8}$$

where f is the SmoothL1Loss defined as

$$f\left(x\right) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}. \tag{4.9}$$

While L_{cls} is comprised of both negative and positive samples, L_{reg} is contributed only by the positive ones. These two terms are normalized with N_{cls} and N_{reg} and weighted with hyper-parameter λ to ensure balance between the two losses. In the current implementation $\lambda = 10$,

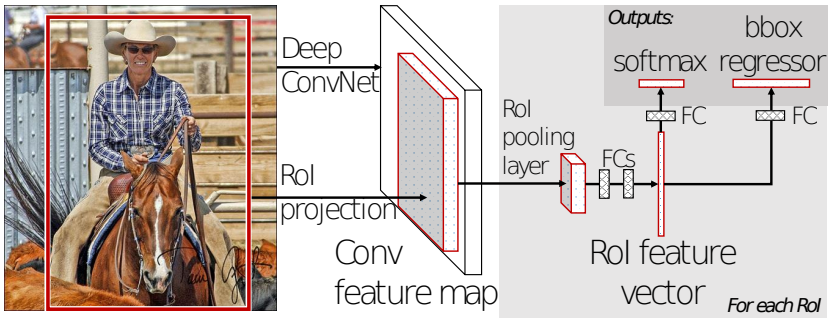


Figure 4.6: *Architecture of the Detection Network in Faster-RCNN is completely inherited from Fast-RCNN. The ROIs are generated from Regional Proposal Network. (source [10])*

N_{cls} and N_{reg} are respectively implemented as the minibatch size (total number of positive and negative samples per image) and the number of anchor positions.

For better training efficiency, samples are constructed with care. First of all, to avoid fuzzy samples, labels are assigned using the following rule:

$$\hat{y}_i = \begin{cases} & \text{if IoU} > 0.7 \\ 1 & \text{or} \\ & i \text{ has the highest IoU with a gt bb} \\ 0 & \text{if IoU} < 0.3 \end{cases} , \tag{4.10}$$

Moreover, since negative samples are dominating, in order to avoid bias towards the negative ones, the samples are assembled with a pos:neg ratio up to 1:1.

DN The structure of Detection Network (shown in Figure 4.6 [10]) remains the same as in Fast-RCNN.

After the last convolutional layer, regional proposals generated from RPN will be used as ROIs in the RoiPooling layer. As explained in section 4.1, this layer creates a feature map with fixed spatial dimension (e.g. 7×7) from a rectangular region of arbitrary spatial size. For each proposal, an feature map is generated and are passed to the successive layers as an individual sample. Two fully connected layers attached on top of the RoiPooling layer compute a large feature vector, which is fed to classification and regression branches in the same fashion as in RPN.

In the loss function, the binary softmax loss is replace by a $K+1$ softmax loss, for K object classes and a catch-all background class. Meanwhile, slight different implementation is taken for the normalizers and weight λ: while in DN $\lambda = 1$, the normalizer is unified between two terms and is implemented as the minibatch size.

$$L = \frac{1}{N} \sum_i L_{cls} \left(y_i, \hat{y}_i\right) + \lambda \frac{1}{N} \sum_i \hat{y}_i L_{reg} \left(\mathbf{t}_i, \hat{\mathbf{t}}_i\right) . \tag{4.11}$$

Training The authors of Faster-RCNN has published their trained model for PASCAL VOC detection challenge [6]. which contains 20 object classes, including: person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor.

Clearly, these 20 classes are far from adequate to serve our purpose, not to mention many of them are inapt for our target action classes (e.g. sheep, potted plant).

Hence we train our own model with expanded object categories. These are handpicked from ILSVRC2014 [39] (200 categories) and PASCAL VOC [6] (20 categories) detection challenge, with exclusion in food, most animals (except horse) and small objects. This yields 118 categories in total and the complete training data is comprised of 196,780 images.

Since detection network assumes fixed proposals during training, Faster-RCNN adopted a staged training method to achieve layer sharing. This method can be summarized into four steps:

1. Train a RPN (`RPN0`) initialized from pre-trained ImageNet VGG16 model [42].
2. Use `RPN0` to generate proposals for the same training data;
3. Train a DN (`DN0`) initialized (again) from ImageNet VGG16 using the proposals generated from the previous step.
4. Train a RPN (`RPN1`) by fine-tuning the fully connected layers of `DN0`.
5. Use `RPN1` to generate proposals from the training data again.
6. Train a DN (`DN1`) by fine-tuning the fully connected layers of `DN0` using the newly generated proposals.
7. `RPN1` and `DN1` yield the final Faster-RCNN model.

In this thesis, we adopted this staged training method. However, it is worth mentioning, by the time of writing, an update has been published that allows joint training of RPN and DN. In this method the proposals generated from RPN branch are fed immediately as pre-computed input into the RoiPooling layer of DN branch *in each iteration step*. While the proposals in fact are input dependent, this approach does not compute their gradient in back-propagation (RoiPooling function is not w.r.t proposal differentiable), hence called *Approximate joint training*. Despite of that, this method yields similar results as the staged training and at the same time reduces the training time by $25\% \sim 50\%$.

4.2.2 Human Track Extraction

Compared to ImageNet data, frame images in video data have lower resolution and large motion blur. The objects of interest are subject to greater appearance variation as well as temporary occlusion. Consequently, person detection is inevitably less reliable in video frames.

Meanwhile, in realistic video datasets videos often contain irrelevant person detections. If used crude, they will add significant amount noise to the training data and consequently increase training difficulty or even affect convergence.

Thus before deploying the detection results as our model input, we need to pre-process the raw

Figure 4.7: *An example where incorrect (advertisement) or irrelevant (jury) detections are consistently given by the detector with high confidence. In this case, we use optical flow to pick out the action performer.*

detections so as to

1. filter out incorrect and irrelevant detections;
2. recover detection failures in individual frames;
3. refine location of bounding boxes.

For these proposes, we propose a simple yet powerful algorithm using the idea of dynamic programming.

We start by characterizing "action relevant" person detections. Contrary to spurious detections, actual action performer displays high overall detectability (except sporadic detection failure) as well as high spatial and appearance consistency in sequential frames. In other words, the bounding boxes locating an action performer should form a smooth and continuous "tube" across frames. However, this criterion alone does not suffice. As is shown in Figure 4.7, wrong detections are often generated consistently with high certainty. Nonetheless, while those mostly remain still with the elapse of time, true action protagonist distinguish himself with high motion saliency.

Inspired by these observations, our task becomes finding a path via raw detections along time axis with the highest detection confidence, temporal consistency and motion saliency. These properties can be well expressed quantitatively. The detectability can be written using the classification probability given by the object detector; temporal consistency in location is reflected in form of Intersection over Union (IoU) and the appearance consistency in form of template matching metrics such as Normalized Cross Correlation (NCC); motion saliency can be described in terms of optical flow. Consider each raw detection as a node and the quantitative metrics as costs (or scores) for connections, this problem can be stated as a shortest path problem and solved efficiently with dynamic programming.

Formally, we define the following problem:

- Given a video of T frames and the person bounding boxes of this video, we divide the video into $N = \lfloor \frac{T}{T_0} \rfloor$ intervals the length of T_0. The bounding boxes within the n-th interval construct the state space

$$b_n^i \in \mathcal{B}_n \text{ with } n \in \{1, 2, \ldots, N\},$$

  $\mathcal{B}_n = \emptyset$ when no person is detected in interval n.

- the control input u_n at step n is the choice of bounding box in step $n+1$, while the input

space $\mathcal{U}_{\setminus}$ includes all raw detections and does not depend on the current state, i.e.

$$\begin{aligned} b_{n+1} &= f_k\left(u_k\right) \\ u_k &\in \{1, 2, \ldots, |\mathcal{B}|\}. \end{aligned}$$

- the score s_{ij} for connecting two bounding boxes b_n^i and b_{n+1}^j consecutive intervals n and $n+1$ is defined as

$$s_{ij}\left(b_n^i, b_{n+1}^j\right) = s_{prob}\left(b_{n+1}^j\right) + s_{motion}\left(b_{n+1}^j\right) + s_{IoU}\left(b_n^i, b_{n+1}^j\right) + s_{size}\left(b_{n+1}^j\right). \quad (4.12)$$

  - s_{prob} is the probability with which bounding box b_{n+1}^j is classified as person, describing detection certainty

$$s_{prob}\left(b_{n+1}^j\right) = \Pr\left(c^j = \text{person} | b_{n+1}^j\right). \quad (4.13)$$

  - s_{IoU} is a function of IoU of the two concerned bounding boxes, aimed to reflect the temporal consistency

$$s_{IoU}\left(b_n^i, b_{n+1}^j\right) = \begin{cases} IoU\left(b_n^i, b_{n+1}^j\right) & \text{if } IoU\left(b_n^i, b_{n+1}^j\right) > 0 \\ -c \text{ with } c > 0 & \text{otherwise.} \end{cases}. \quad (4.14)$$

  - s_{motion} is the average of the normalized magnitude of optical flow in area b_{n+1}^j, describing motion saliency

$$s_{motion}\left(b_{n+1}^j\right) = \frac{1}{|b_{n+1}^j|} \sum_{(x,y)\in b_{n+1}^j} \|o\left(x, y\right)\| \quad (4.15)$$

  - $s_{size}\left(b_{n+1}^j\right)$ puts additional penalty on small bounding boxes. This proves to be useful in complex scenes since the action performer is usually the most prominent among all detected person.

$$s_{size}\left(b_{n+1}^j\right) = f\left(\frac{|b_{n+1}^k|}{\max_k |b_{n+1}^k|}\right), \quad (4.16)$$

    where f, shown in Figure 4.8, is a sigmoid-like non-linear mapping to suppress small regions, while preserving the large ones. A sigmoid function has the form

$$f\left(x\right) = \frac{L}{1 + \exp\left(-k\left(x - x_0\right)\right)}. \quad (4.17)$$

    In our implementation, we set the parameters to

$$x_0 = 0.5 \quad k = 12 \quad L = 0.5. \quad (4.18)$$

    While x and k are set so to keep the x-span of the "S" shape within $[0, 1]$, so as to comply with the input range, L is determined from empirical test results.

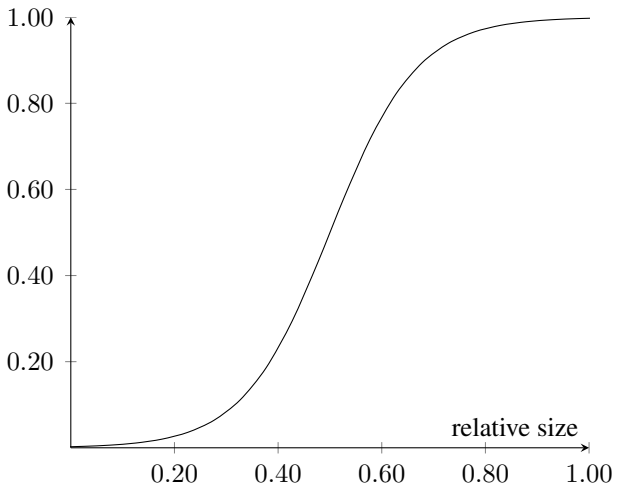


Figure 4.8: *Although the relative size of a detected person gives hint to his relevancy, our experiment suggests this relation is not linear. Hence we apply a sigmoid-like threshold function to suppress only very small bounding boxes.*

- Finally we consider the following optimization problem:

$$\max_{\pi} \left(\sum_{n=1}^{N} s_{ij} \left(b_n^i, b_{n+1}^j \right) \right), \tag{4.19}$$

  where π denotes a sequence of bounding boxes selected from each interval.

This problem can be solved efficiently using dynamic programming by applying Equation 3.8 from section 3.2 ($\max$ instead of $\min$).

The optimal policy π^* is a sequence of N bounding boxes. The frames, from which the bounding boxes are chosen, form a set of *key frames*. From these key frames, we apply a linear ex-interpolation on the four coordinates of the bounding boxes to obtain a smooth track of the person of interest in the whole video span.

There is one noteworthy design choice in our method. As an alternative to the natural choice to treat each frame as a time step, we group multiple consecutive frames to form a single step, thus implicitly increase the state space at each step $\mathcal{B}_n$. As a result, we are able to "look ahead" when no feasible solution exist at current frame. This introduced flexibility is the key to recover detection failures and correct faulty detections.

An example of resulted person track is shown in Figure 4.9. As we can see, our method successfully recognizes the action performer among multiple people detections and fix detection failures even under large temporal gap.

Although in most cases we are able to locate the action performer from the optimal track determined by our algorithm, exceptions reside especially in action classes where multiple people are involved or overlapped. In this case, we need to find another track that is independent from the previous one. We stress independent here, since in the concerned situation the interested person tracks are often intertwined. As a result, simply returning the second optimal solution from dynamic programming would have resulted in a mixed track, partially containing the already extracted one (see Figure 4.10 for a concrete example).

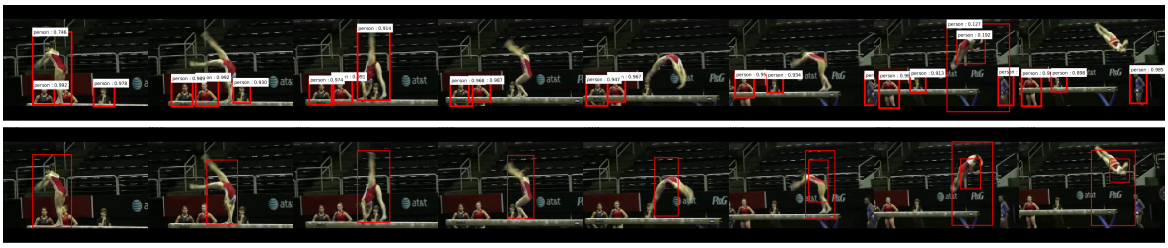


Figure 4.9: *An example of the resulted person track using our algorithm. Compared to the raw detections (upper row), our method is able to filter out noisy detections and recover missing frames, despite detection failures over a long period and confident detections of irrelevant people.*

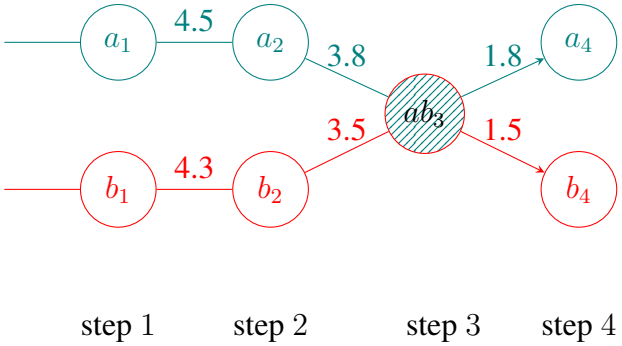


Figure 4.10: *Illustration of two person tracks that are intercepted at one frame. Red and green represent the tracks of person a and b, and the green one is apparently the optimal track. However if we want to extract person b additionally by simply backtracking from the node with the second highest score (in this case b_1) in step 1, we will end up with the track $b_1 - b_2 - a_3 - b_4$.*

We solve this problem by eliminating the first person track (and bounding boxes that highly overlap with it) from the available raw detection, while preserving the interception bounding box (the shadowed node in Figure 4.10). From this reduced state space, we repeat dynamic programming algorithm to extract the second person track. An example of the resulting person tracks is shown in Figure 4.11.

The complete algorithm is summarized in Algorithm 1. Notice that small adaptation is implemented to handle intervals void of detections ($\mathcal{B}_n = \emptyset$). Additionally, score-filtering as well as early termination condition are added for the sake of computation efficiency.

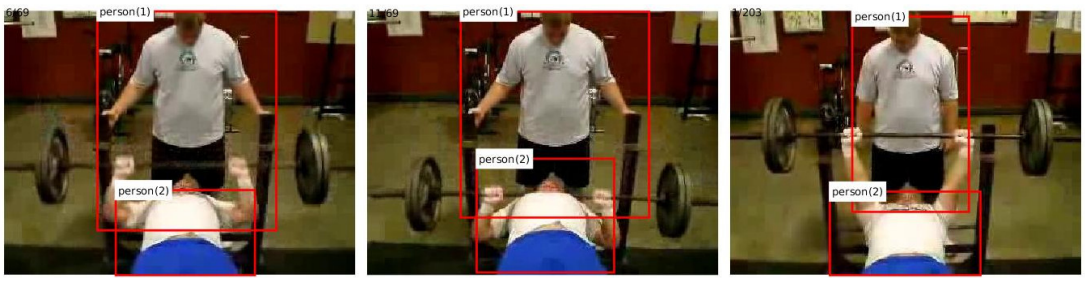


Figure 4.11: *An example where multiple individual person tracks, partially overlapping, are extracted.*

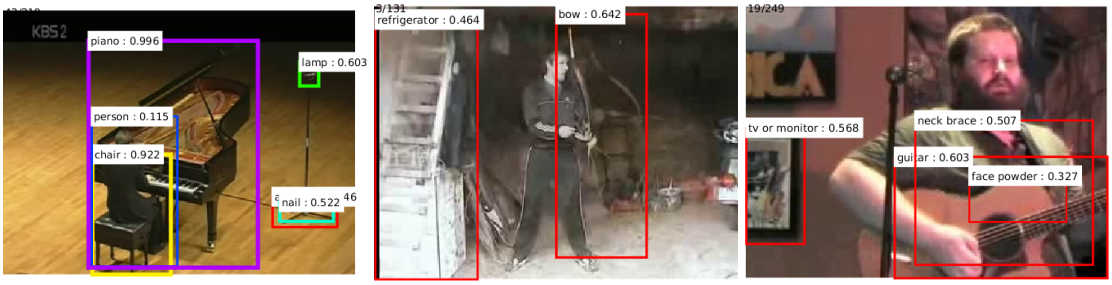


Figure 4.12: *Examples of object detection results*

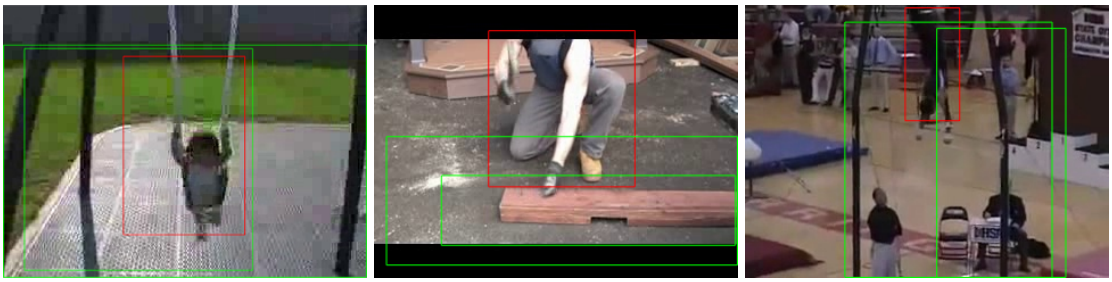


Figure 4.13: *Examples of filtered bounding boxes.*

4.2.3 Object Filtering

Due to much larger variance and complexity presented in video, meanwhile refrained by the number of target object categories, detection results for objects are inevitably less reliable than for human. On the other hand, drawing a clear line between relevant and irrelevant objects is not possible. As a matter of fact, as mentioned in [17], while the existence of certain object categories matter, the absence of them provides useful information as well. While manually matching object categories with action classes is feasible, such approach is not suited for general scenarios.

However as the examples in Figure 4.12 shows, the classification probability is usually a good indicator on how substantial the detected object is. Meanwhile the size of the object and its distance to human gives important hints to the relevance of the object.

From these observations, we first eliminate all objects whose largest prediction probability lower than 0.1 then we exclude bounding boxes with length smaller than 20 pixels. In the frames existence of any actor, we further filter out objects that do not overlap with any detected action performer. The remaining detected objects comprise ROIs for object channel.

In Figure 4.13, we show some examples of filtered object detections.

4.3 Merging Unit

Obvious, different semantic cue impact the understanding and interpretation of each individual type of actions with different weights. Hence, one main focus in this thesis is to find a sophisticated way to integrate them together.

For the purpose, we propose 6 kinds of fusion methods. Assume we have C action classes and L semantic channels, the classification score generated by channel l is denoted as $s_l \in \mathbb{R}^C$, then we have:

1. **sum**: the classification score of class c is the summation over all available cues.

$$s\left[c\right] = \sum_{l=1}^{L} s_l\left[c\right]. \tag{4.20}$$

   In back-propagation, loss gradient is equally passed to all cues. Implicitly, this assumes that all cues have equal contributions to the final decision.

2. **max**: instead of summing the classification scores, one can also apply max operation. Intuitively, this is equivalent as picking the strongest cue for each action class as the final class representation. Recall the MaxPooling operation discussed in 3.1.1, during back-propagation gradient from upper layer is passed down only to the corresponding cue, from which the maximum value is pooled. As a result, each cue will be updated according to its actual contribution. Formally,

$$s\left[c\right] = \max_{l=1}^{L} s_l\left[c\right]. \tag{4.21}$$

   It should be noted, this fusion is equivalent to the MIL merge we use for object cue (see section 4.1). In this case, results from different cues are instances in a bag.

3. **weighted** The drawback of sum fusion is that it treats all cues equally across all action classes. In reality, this is not true in most cases, e.g. person in "JumpingRope" has much higher relative importance than person in "HorseRace". To address the issue, we propose weighted fusion, namely

$$s\left[c\right] = \sum_{l=1}^{L} w_l[c] s_l\left[c\right], \tag{4.22}$$

   There are a total $L \times C$ weights and $w_l\left[c\right]$ is the weight of the l-th cue for class c.

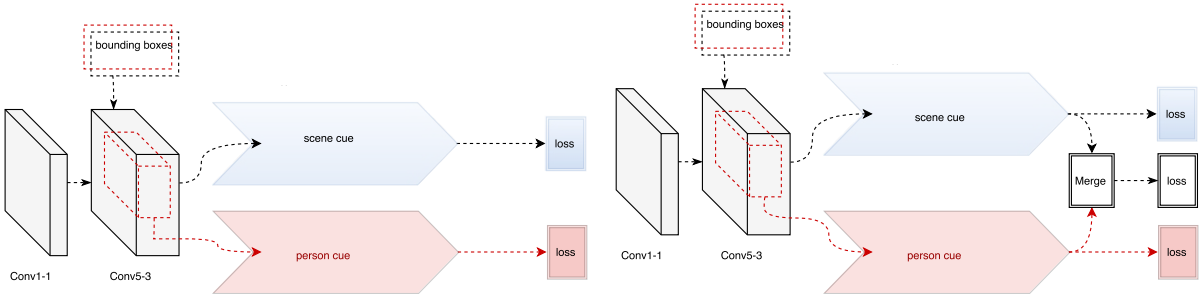
4. **cross-weighted** Moreover, we experiment with the cross-class joint weighting same as in [52]. In other words, the final classification score is determined by the classification results of all cues and all classes. This setting incorporates extra information from class confusions and exclusions. For example, if a sample frame has strong response to "Kayaking" from scene channel, then a smaller weight should be assigned to "HorseRace" in person channel. Formally, the final score can be written as

$$s\left[c\right] = \sum_{l=1}^{L} \sum_{c'=1}^{C} w_l\left[c, c'\right] s_l\left[c'\right]. \tag{4.23}$$

   There are in total $L \times C \times C$ weights.

5. **multi-loss** Besides the above fusion methods, we also explored with multi-tasking training scheme implemented in Fast-RCNN. Classification using each semantic cue is regarded as a separate task. In forward pass, each cue generates a loss independently as illustrated in Figure 4.14a, whose gradients are passed down through the corresponding fc branch and combined before the shared convolutional layer. For testing, we compute the final prediction via either summation or maximization.

6. **multi-loss+** At last, as depicted in Figure 4.14b we combine the multi-tasking with fusion unit, where the choice fusion unit is determined by the result of above comparisons. While this architecture still uses the merged result as final classification decision, separate losses are used as constraints to guarantee single-cue performance. In order to emphasis the importance of merged result, we assign a higher weight to the merged loss. As a result, the parameters will be updated more in favour of the merged result.



(a) *Multi-loss Fusion* ***(b)*** *Multi-loss+ Fusion*

Figure 4.14: *Architectures for multi-loss fusion proposals.*

Algorithm 1 Top-K Person Track Extraction

Output: $metricThreshold$
Input:
T RGB video frames,
$2T$ flow images,
T sets of bounding boxes from raw person detection $\{\mathcal{B}_t\}_1^T$
if $p\left(person|b\right) < s_{min}$ **then** ▷ Filter very uncertain detections
    remove b from $\{\mathcal{B}_t\}_1^T$
end if
Parameter: T_0, s_{min}, K
for 1 to K **do** ▷ Top-K person tracks
    group $\{\mathcal{B}_t\}_1^T$ into $\lfloor \frac{T}{T_0} \rfloor$ intervals, $\{\mathcal{B}_i\}_1^N$
    remove empty intervals and get $\{\mathcal{B}'_i\}_1^{N'}$ $s.t. \mathcal{B}'_i \neq \emptyset$
    if $N' \leq \frac{N}{2}$ **then**
        break
    end if
    procedure DYNAMIC PROGRAMMING(flow, $\{\mathcal{B}'_i\}_1^{N'}$)
        $J_{N'}\left(b_{N'}^i\right) = s_{prob}\left(b_{N'}^i\right) + s_{motion}\left(b_{N'}^i\right) + s_{size}\left(b_{N'}^i\right).$
        repeat
            Apply dynamic programming recursion:
            $J_n\left(b_n^i\right) = \max_j \left(s_{ij}\left(b_n^i, b_{n+1}^j\right) + J_{n+1}\left(b_{n+1}^j\right)\right)$
            $\mu_n^*\left(b_n^i\right) = \arg\max_j \left(s_{ij}\left(b_n^i, b_{n+1}^j\right) + J_{n+1}\left(b_{n+1}^j\right)\right)$
        until $k = 1$
        if $\max_i J_1\left(b_1^i\right) < 0$ **then** ▷ soft constraint
            break
        end if
        forward trace $\{\mu_n^*\}_1^{N'}$ to get $\pi^* = \{b_1^*, b_2^*, \ldots b_{N'}^*\}$
    end procedure
    linear ex-interpolate from π^* to get $\pi^{**} = \{b_t^*\}_1^T$
    ▷ remove found person track according to Figure 4.10
    for $t = 1$ to T **do**
        if $IoU\left(b_t^i, b_t^*\right) > 0.8$ **and** $\nexists\, j \neq k,\ s.t. \mu(b_{t-1}^j) = \mu(b_{t-1}^k) = b_t^i$ **then**
            remove b_t^i from $\mathcal{B}_t$
        end if
    end for
end for

5 Implementation Details

We use the deep learning library Caffe [20] in all of our experiments.

5.1 Data

We choose UCF101 dataset because it is a good trade off between number of action classes and the variety of actions. This dataset contains 101 action classes and there are at least 100 video clips for each class. The whole dataset contains 13,320 video clips, which are divided into 25 groups for each action category.

The person and object bounding boxes are extracted as described in section 4.2.

We retrieve frame images using OpenCV decoder. The flow inputs are obtained using TVL1 algorithm from OpenCV. Both inputs are resized to 256×340.

5.2 Training

We fix our network input to be 224×224. For spatial net, dimension is $224 \times 224 \times 3$, while for temporal the net input dimension is $224 \times 224 \times 2L$, which is comprised of the concatenation of flow images (in x and y) in L consecutive frames.

Data augmentation is a important step in CNN training as it effectively increases data size, elevates variance and prevents overfitting. Random cropping, horizontal flipping and RBG jittering are common techniques for this purpose. In this thesis, we applied horizontal flipping and cropping for data augmentation. In particular, we adopted the cropping scheme suggested in [49]. This scheme creates 5×14 kinds of croppings candidates from 5 positions (four corners and center of the original image) and 14 size variants. The width and height of the cropping window is randomly chosen from $\{168, 192, 224, 256\}$, while the most two extreme combinations (168×256 and 256×168) are rejected. The cropped image patch will be resized to 224×224, which will be used as input for our network.

When training with person or object cue, we use exactly one person bounding box and at least

one object bounding boxes for each frame. For temporal network, we use the union of bounding boxes from each L frames to determine the input bounding box for the sample sequence.

In order to guarantee sufficient area of person bounding boxes in the cropping window, we reject cropping candidates those overlapping ratios in width and height are lower than 0.5. In case no person has been detected in the frame, we use the whole span of the input image as person bounding box so as to keep the gradient scale consistent.

As for objects, we discard object bounding boxes outside cropping area. Similar as for person, in absence of valid object bounding boxes, scene bounding box will be used instead.

Since in the current implementation, if the input ROIs are smaller than the specified output dimension $w \times h$ (7×7), RoiPooling layers returns an all-zero output. Thus we enlarge ROIs whose length is smaller than 16×7, where 16 is the total stride in conv5-3.

In all experiments, we use the public model VGG16 [42] for initialization and set both *dropout rate* to be 0.8. Without particular specification, RGB stream is trained for 10000 iterations; the *learning rate* is initialized to be $1e^{-3}$ and decreased every 4000 iteration. Whereas for flow, we train 19000 iterations and set the initial *learning rate* to be $5e^{-3}$, which is reduced every 8000 iterations. A batch is assembled by randomly sampling 16 frames from 16 different videos. While so, we set the *itersize* to 16, so that the effective *batchsize* remains 256.

5.3 Testing

For fair comparison, we follow the same testing routine as in [41]. Specifically, from each input frame 10 samples generated from 5 crops and horizontal flips.

However, when using person cue, we discard croppings that have insufficient interceptions with person (same as in training) and the same procedure is repeated for every person track.

The final classification results from averaging the classification scores of all valid croppings and person tracks.

6 Evaluation

In this chapter, we pursuit to answer the following three questions

1. Does semantic structure help action classification?
2. If so, how to bring out the best complementary effect?
3. How does semantic structure affect recognition performance for each action category?

In the following analysis, baseline refers to an in-house trained two-stream CNN network that does not integrate semantic cues. In other words, baseline model utilizes single scene cue. Without special declaration, evaluations in section 6.1~section 6.4 refers to UCF101 split1, while in section 6.5 we compare the classification performance of our proposed model with state-of-art methods across all UCF101 splits.

We choose train our own network instead of using a published model [49] because we have noticed that although the average classification accuracy stays consistent, the actual class accuracies have been shifted, meaning that the local minimum has slightly changed due to actual implementation differences. As we will cast a detailed investigation concerning category-wise performance in section 6.4, we use our own baseline in order to maintain a consistent local minimum for fair comparison.

The average accuracy of our baseline model and the published model is summarized in Table 6.1.

mAcc (%)	Split1		Split2		Split3		Avg	
Model	spatial	temporal	spatial	temporal	spatial	temporal	spatial	temporal
Theirs	79.8	85.7	77.3	88.2	77.8	87.4	78.4	87.0
Ours Baseline	79.42	85.27	77.14	88.13	77.25	86.96	77.93	86.79

Table 6.1: *Comparison of our in-house trained baseline with publicly available model [49]*

6.1 Contributions of Semantic Cues

In this section we verify whether incorporating explicit semantic structure defined by "person", "scene" and "object" cues enhances action recognition.

For this purpose we evaluate scene-only network and person-only network for spatial and temporal streams to study the relative importance of individual cues for different streams respectively. (Since object is regarded as an aiding cue, we omit evaluating its individual performance here.) To acquire an idea on how well scene and person cues complement each other, we apply a late-fusion by averaging the classification scores yielded from the two networks.

As we can see from Table 6.2, person and scene exhibit unequal relative relevance in spatial and temporal nets. While for spatial net, classification based on the whole scene is much more accurate than based on person (79.42% vs 73.82%), the relation is reversed in temporal net. This is not surprising since optical flow is inherently person-centric. On contrary, spatial net is fed on RGB information of single frames, where the appearance of action performer is usually less descriptive while the global contexts often provide essential hints for correct classification.

For both nets, combining person and scene in a late-fusion manner improves the classification accuracy, indicating the reciprocal property between the two cues.

From this point, we investigate the effect when incorporating both semantic cues into the same model as proposed in section 4.1 in favor of computational efficiency. As we can see from Table 6.2, the performance boost transfers to the proposed joint model and for spatial network the classification accuracy even exceeds that from late fusion (80.46% vs 80.10%).

Lastly, we incorporated object cue in spatial net (since object motion is too ambiguous in defining action, we omitted object cue for temporal net). Unfortunately, incooperating object channel does not generate significant improvement. While generating a marginal improvement compared to scene-only net (79.71% vs 79.42%), it falls behind "person+scene" model by 0.75%. We think the cause is manifold:

1. Object appearances have great intra-class variation as well as inter-class correlations. Particularly given that the object detections are still very noisy, the discriminant power in object channel is insufficient. In order to leverage object cue, a more expressive representation might be needed.

2. Since in the current implementation, we only utilize locational information of the object to extract sub-regions of the feature maps, object channel can be considered as an augmentation of scene channel. The summation of all three channels together implicitly puts more weight on scene channel, abating the strength of person channel.

3. The current implementation of MIL layer couldn't effectively update weights for relevant and irrelevant objects distinctively. As explained in section 4.1 object i's c-th entry (and the connected weights and lower layers) will be updated, as long as this object produces the strongest signal in class c among all objects in the input frame I. From our understanding, this algorithm can only work well under two assumptions: (1) there exist at least one relevant object in the frame (2) the detected objects are not relevant to other action classes. To elaborate this issue, consider we have detected two object regions containing "bow" and "green arena floor" in a frame from "Archery" action class. Assume this frame

is correctly classified, according to Equation 3.1 a *positive* gradient will be passed to all other classes to *suppress* the corresponding classification signal. However, since "green arena floor" is the most representative region for classes such as "TennisSwing", it will be forced to decrease its response for "TennisSwing", although in reality it is positively contributive to this class.

mAcc(%)	S	P	S+P (two nets)	S+P (jointly)	S+P+O (jointly)
Spatial	79.42	73.82	80.16	**80.46**	79.71
Temporal	85.27	87.02	**87.84**	87.63	–

Table 6.2: *Integrating of explicit person cue (P) additionally to scene (S) is beneficial for both spatial and temporal streams. The contribution of object cue is marginal. Our proposed architecture (joint) is profitable in spatial net.*

6.2 Fusion Methods

From the previous section, we have learned that introducing person channel to action recognition effectively increases classification accuracy. In this section we investigate various fusion methods described in section 4.3 so as to maximize the inter-channel complementary power. Considering training effort, all experiments are conducted using spatial net on split1 of UCF101 dataset.

The evaluation results are listed in Table 6.3.

First of all, opposed to our initial expectation, max fusion behaves considerably inadequately among all fusion variants. We believe this is due to the fact that since max operation only selectively updates the stronger channel (see section 4.3), it requires both channels to remain balanced in order to train all channels equally. During training if one channel appears to be stronger than the other, max would keep reinforcing the same channel, hence tilting the balance even further.

Secondly from Table 6.3 we also see that multiloss fusion variants yield inferior results than single-loss methods (except max). While multiloss performs well in jointly train a shared model for different tasks (e.g. regression and classification in Fast-RCNN [10]) and dataset augmentation as in [41], this scheme cannot sufficiently leverage the joint contribution from person and scene cues. According to our analysis, there are two possible reasons.

1. Unlike the aforementioned examples, where each sub-task is relatively independent, action recognition via different semantic cues are much stronger interplay, therefore sharing the same loss (thus a much similar parameter update in fc layers) could be beneficial for both channels;

2. Since scene region pooling always includes person region pooling and since person bounding boxes can contain noisy inputs, combining the classification scores helps recover individual input noise thus increase the robustness of the system.

Lastly, as is shown in Table 6.3 the two weighted fusion methods do not bring substantial improvement. We think this is because since during DNN training models always overfit to training data, the inter-class confusion as well as relative cue importance cannot reflect the true distribution. Hence, weighting classification scores could not induce further performance boost. On contrary, increasing the total number of model parameters could lead to even severe overfitting.

Fusion	max	sum	weighted	cross weighted	multiloss (sum test)	multiloss (max test)	multiloss+
Avg Acc.(%)	78.95	**80.46**	79.77	80.20	79.15	79.03	78.91

Table 6.3: *Exploration of fusion methods using scene and person cues for spatial net on UCF101. Sum fusion outperforms other fusion methods.*

Based on previous empirical study, we are ready to build up our final action recognition architecture: sum-fused model with semantic channels "Scene" and "Person".

The final results on all 3 splits are summarized in Table 6.4.

mAcc (%)	Split1		Split2		Split3		Avg	
Stream	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
Spatial	79.42	**80.46**	**77.14**	76.53	77.25	**77.97**	77.93	**78.32**
Temporal (P)	85.27	**87.02**	88.13	**89.00**	86.96	**88.86**	86.79	**88.29**
Temporal (P+S)	85.27	**87.63**	88.13	**89.33**	86.96	**88.36**	86.79	**88.48**
Two Stream	90.98	**92.75**	91.45	**92.14**	91.05	**92.91**	91.15	**92.60**
Two Stream (Temp P+S)	90.98	**92.55**	91.45	**91.98**	91.05	**92.36**	91.15	**92.30**

Table 6.4: *Based on the previous empirical study, we propose using "scene" + "person" (P+S) model architecture for spatial and temporal network. We compare our proposed model with baseline over three splits on the UCF101 dataset, whereas baseline is the in-house implementation of two-stream CNNs. Two Stream results are yielded from summing spatial and temporal classification scores using weight* $1:3$.

6.3 ROI Quality

In this section we study the effect of the person detection quality on the model performance.

	Baseline	S+P (raw)	S+P (filtered)	S+P (GT)
Accuracy (%)	51.16	52.01	53.77	54.25

Table 6.5: *Accuracy on JHMDB (split 1) using person ROIs of different quality. Similar as in UCF101, baseline refers to the inhouse trained model using solely "scene" channel.*

For this purpose, we focus on the annotated video dataset, JHMDB [18], and evaluate the spatial S+P model using (1) person ROIs from raw Faster-RCNN object detector, (2) filtered actor ROIs as described in subsection 4.2.2 and (3) ground truth person ROIs.

JHMDB is created from HMDB [24]. It is a fully annotated video dataset, which consists of 928 trimmed video clips from 21 action classes. These classes are mainly single-actor action classes,

As is shown in Table 6.5 the quality of extracted person ROIs directly affects accuracy for action recognition. For JHMDB dataset (mostly single person action), even raw detection results suffices to bring in an evident improvement; futhermore On the other hand, our filtered person ROIs are able to generate near groundtruth performance. This suggest that our method is robust against suboptimal ROI extraction.

6.4 Action Categories

In this section, we return to the action class categorization (Figure 1.2) proposed in chapter 1 that gave incentive to this thesis and evaluate the effectiveness of our model with respect to these categories.

Recall that we showed in Table 1.1 the baseline model performs significantly worse in action categories with weak dependence on scene. In our analysis earlier, this could be caused by the inefficiency of conventional method in autonomously learning to abstract the most discriminant semantic information from complex context. This becomes a much severe issue when the dataset inhabits small variance, as deep neural networks easily overfit to unrelated information.

Our proposed structure tackles this issue by providing explicit semantic structures. By design, it should be particularly useful for scene-independent classes. In this regard, we evaluate our model category-wise with respect to the baseline (scene only).

Figure 6.1 compares the per-category mean classification accuracy on split1 using different combinations of cues (we fix fusion method to be sum fusion, as it is the best performing one from the discussion in section 4.3). In Figure 6.2 we show the 10 most improved and most reduced action classes using our method compared to baseline.

To begin with, it should be noticed with increasing importance of scene, classification accuracy shows a clear rising trend both in spatial and temporal nets, which justifies our hypothesis that the conventional semantic-indifferent methods is sensitive to uninformative variability in scene channel.

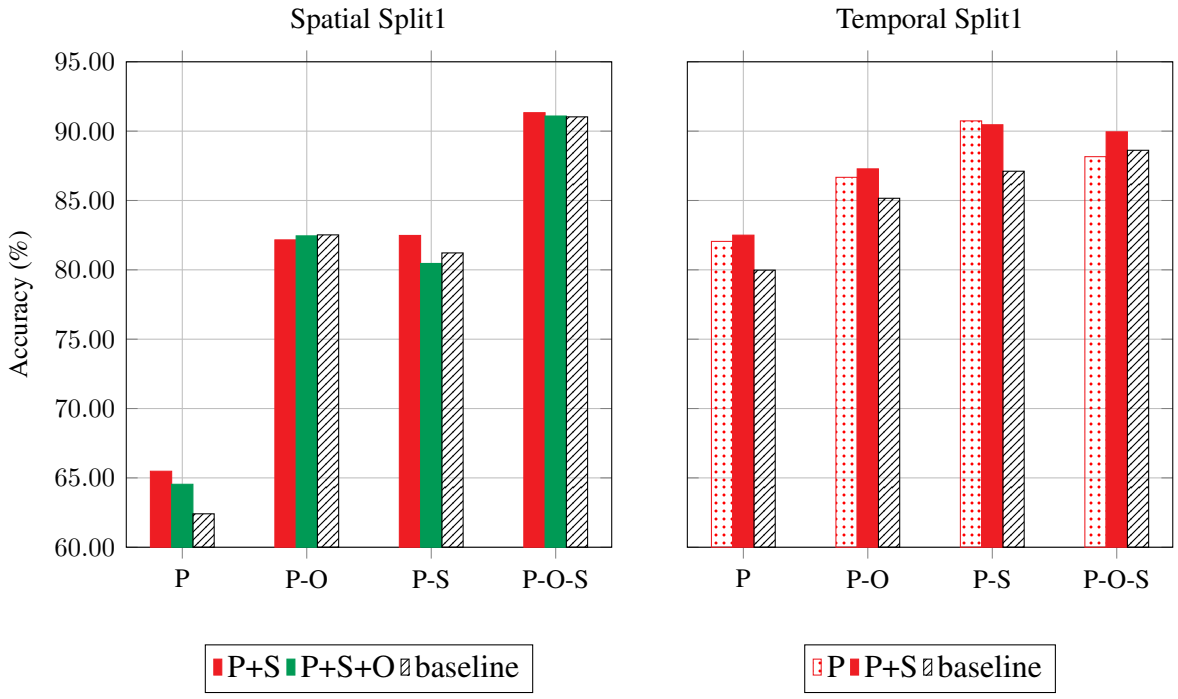


Figure 6.1: *Model performance evaluated according to different action categories on spatial stream (left) and flow stream (right). By integrating person cue, the performance of the originally weakest action category (motion only, P) is evidently enhanced.*

Person Channel We first focus on the effect of person cue. As is shown in Figure 6.1, models with separate person channel introduce significant performance boost in *P* category for both spatial and flow nets. This proves that the integration of person cue is able to reduce the interference from scene channel.

Interestingly, an equally large improvement can be observed in *P-S* category. A careful examination over change of class-accuracies suggests that this improvement is mainly induced by action classes in similar scenes, for instance "FrontCrawl" vs "BreastStroke", "CricketBowling" vs "CricketShot" (ref Figure 6.2). This indicates, the utilization of person bounding boxes provides more refined pooling regions localizing the actual actions, which enables the identification of more sophisticated differences between similar actions.

While for spatial net the performance gain induced becomes less decisive in *P-O* and *P-O-S* categories, it remains striking for temporal net, suggesting that the person cue is especially beneficial for temporal net. This agrees with the conclusion we drew in section 6.1, namely for motion domain person is the dominating cue for successful action recognition, while for appearance domain the whole scene plays a more decisive role.

Object Channel On the other hand, the effect of object channel is not definite. As we can see from Figure 6.1, although the merged result improves the *P* category by a good margin (2.06% in Figure 6.1), this is in fact induced by person cue. In the remaining categories, object channel approximately aligns with scene channel, which confirms our assertion in section 6.1.

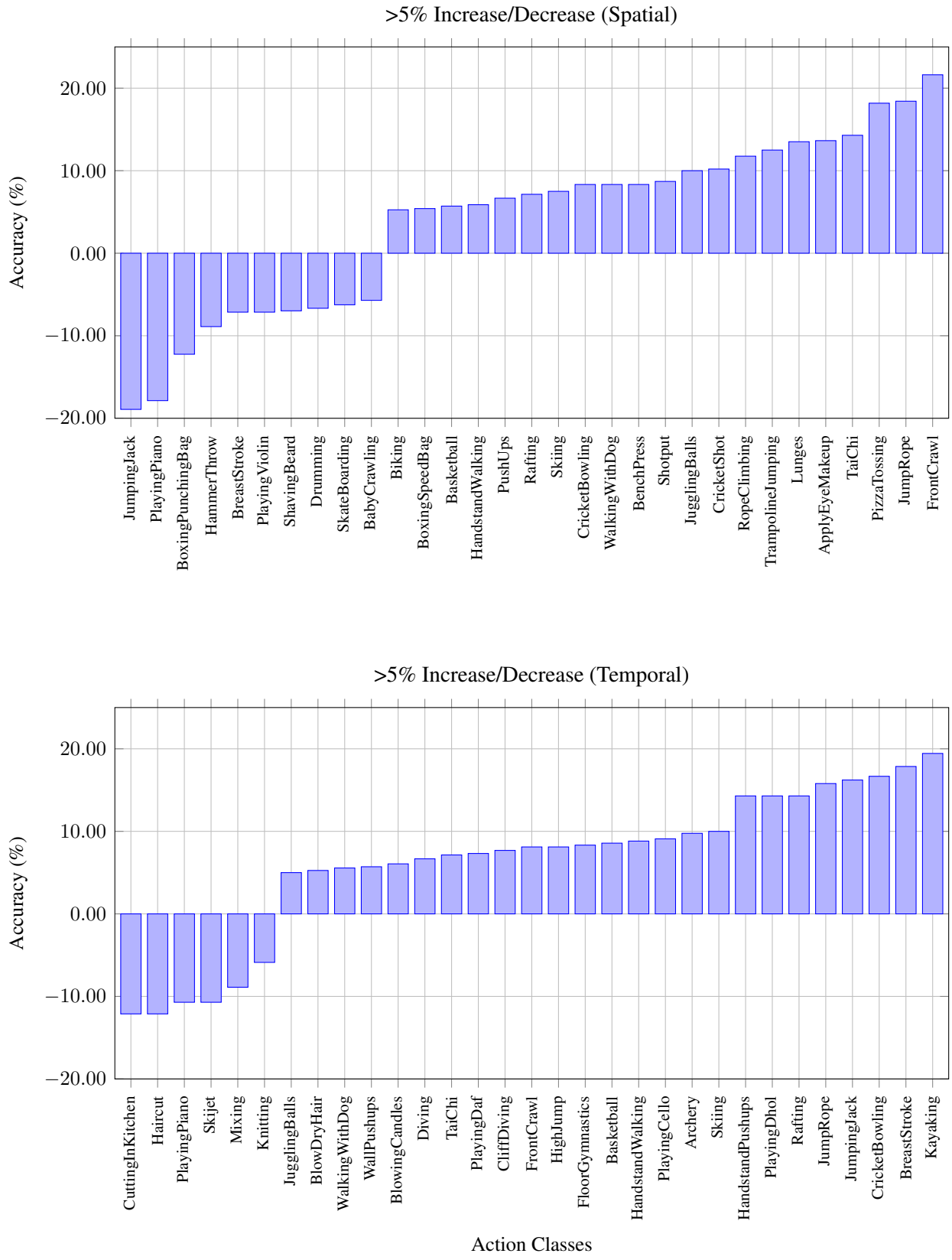


Figure 6.2: *Classes with $> 5\%$ performance gain and decline in spatial and temporal using our proposed network (P+S).*

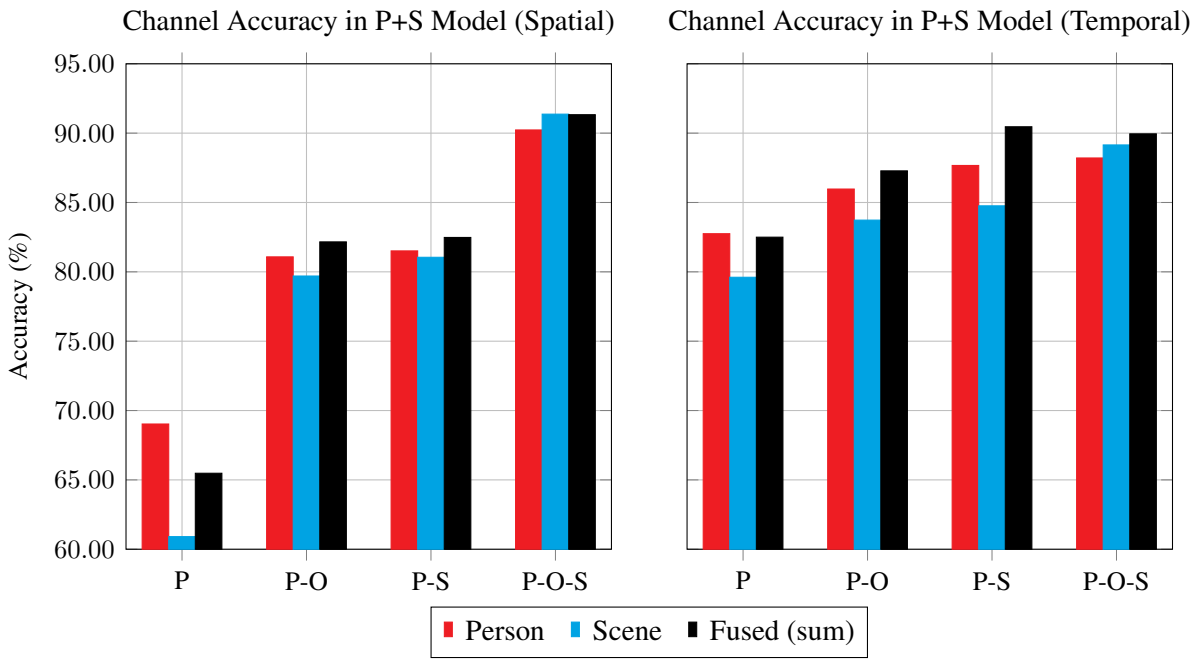


Figure 6.3: *Performance from individual channels before merging unit in spatial network with sum-fused person and scene channels. The relative performance discrepancy echoes our action class categorization. The merged aligns with the stronger channel, showing the complementary effects of our model.*

Complementarity Furthermore, Figure 6.3 depicts another noteworthy point. In this figure, we investigate the performance of individual channels *before* sum merging and their merged result in each action category. As we can see, person channel outperforms scene channel in most action categories. As the scene becomes more indicative, the advantage of person channel gradually resides until it is overtaken in *P-O-S* category, where the scene is overwhelmingly indicative. This trends confirms our action categorization. At the same time, it implies that when explicitly decomposed either channel is able to unfold its classification strength in its own "specialization".

Moreover, as is black plot in Figure 6.3 shows, the merged results either align with (in *P-O*, *P-S* and *P-O-S* categories) or evidently is boosted (in *P* category) by the stronger channel. This implies that our proposed model is able to exploit the strength from both semantic channels in an effective and complementary way.

6.5 Comparison with State-of-the-art

Finally we compare our proposed approach to other state-of-the-art methods on UCF101 dataset.

The results are summarized in Table 6.6, where we compare our result with both traditional approaches such as improved trajectories (iDTs), and deep learning representations, such as Two Stream and Deep Two Stream.

Method	iDT + FV [48]	C3D [45]	TwoStream [41]	TwoStream [41] + LSTM [54]	Deep TwoStream [49]	Ours
Avg.Acc (%)	85.9	85.2	88.0	88.6	91.4	**92.6**

Table 6.6: *Comparison with the state-of-the-art methods on the UCF101 dataset.*

7 Conclusion and Future Work

7.1 Conclusion

In this thesis, we aim to comprehend a better understanding over video-wise action recognition.

We started by diagnosing the shortcomings in current approaches, and concluded that performance of current approaches is weakened by the overfitting to ambiguous information. One of the causes lies CNN's inefficiency in identifying and separating relevant semantic cues from complex context.

To this end, we proposed a framework that explicitly integrates multiple semantic cues into the conventional CNN pipeline. Especially, we explored different fusion methods for combining the semantic cues and quantitatively investigated various training strategies better overall performance in terms of classification accuracy. Furthermore we systematically evaluated the impact of each semantic cues and their actual contribution to different types of action classes.

The proposed model is computational efficient and modular, hence exhibiting great modeling capacity and flexibility. In terms of classification performance, it exceeds the conventional two-stream CNN pipeline and yields state-of-the-art performance on challenging dataset.

7.2 Future Work

In our thesis, we have concluded that the separation of object channel does not make evident contribution to action recognition and a possible reason could be the lack of expressiveness in the representation. As a potential future work, one could explore more explicit representations to describe human object interaction and more challengingly investigate the feasibility to incorporate such representations *efficiently* in existing frameworks.

In the scope of this thesis, we adopted simple average pooling to obtain the video-wise classification result of 25 sampled frames (or sequences in temporal stream). This setting enabled fair comparisons with related works, but neglected the fact that the importance among video frames in a video sequence is unequally distributed. This issue has been addressed using recurrent neural network [4] or ranking based method [8]. It would be interesting to incorporate semantic

regional pooling with temporal pooling.

Last but not least, in our proposed system, object detections are generated during pre-processing externally using a separate CNN. It should be noticed that the detection CNN share great similarity with our action classification CNN. Hence from the perspective of computational efficiency, it is worth experimenting the possibility to combine the both of them to create a model that generates salient semantic regions *and* with the aid of the generated regions predicts action classes.

Appendix

A

A.1 Categorization of UCF101

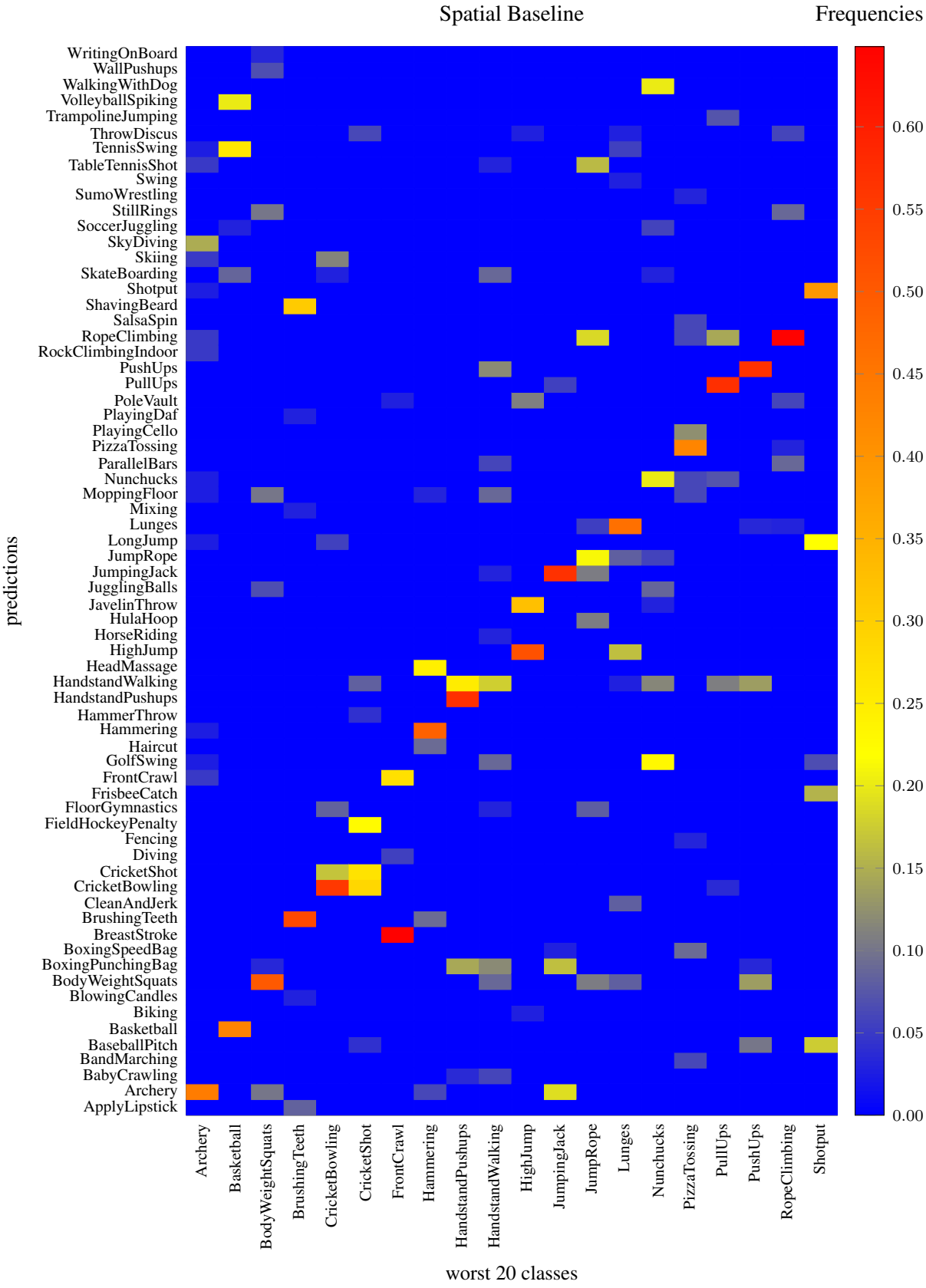
Body Motion (*P*)	ApplyEyeMakeup, ApplyLipstick, BabyCrawling, Basketball, BodyWeightSquats, BoxingSpeedBag, BrushingTeeth, Haircut, HandstandPushups, HandstandWalking, HeadMassage, HulaHoop, JugglingBalls, JumpingJack, JumpRope, Lunges, PullUps, PushUps, RopeClimbing, SalsaSpin, ShavingBeard, TaiChi, WallPushups, YoYo
Human-Object Interaction (*P-O*)	Archery, BenchPress, Biking, BlowDryHair, BlowingCandles, BoxingPunchingBag, CleanAndJerk, Drumming, Hammering, HorseRiding, Knitting, Mixing, Nunchucks, PizzaTossing, PlayingCello, PlayingDaf, PlayingDhol, PlayingFlute, PlayingGuitar, PlayingPiano, PlayingSitar, PlayingTabla, PlayingViolin, SoccerJuggling, Typing, WalkingWithDog
Body Motion in specific Scene (*P-S*)	BandMarching, BasketballDunk, BreastStroke, CliffDiving, CricketBowling, CricketShot, Diving, Fencing, FieldHockeyPenalty, FloorGymnastics, FrisbeeCatch, FrontCrawl, GolfSwing, HammerThrow, HighJump, IceDancing, JavelinThrow, LongJump, MilitaryParade, Punch, RockClimbingIndoor, Shotput, SkyDiving, SumoWrestling, Surfing, ThrowDiscus, VolleyballSpiking
Human-Object Interaction in specific Scene (*P-O-S*)	BalanceBeam, BaseballPitch, Billiards, Bowling, CuttingInKitchen, HorseRace, Kayaking, MoppingFloor, ParallelBars, PoleVault, PommelHorse, Rafting, Rowing, SkateBoarding, Skiing, Skijet, SoccerPenalty, StillRings, Swing, TableTennisShot, TennisSwing, TrampolineJumping, UnevenBars, WritingOnBoard

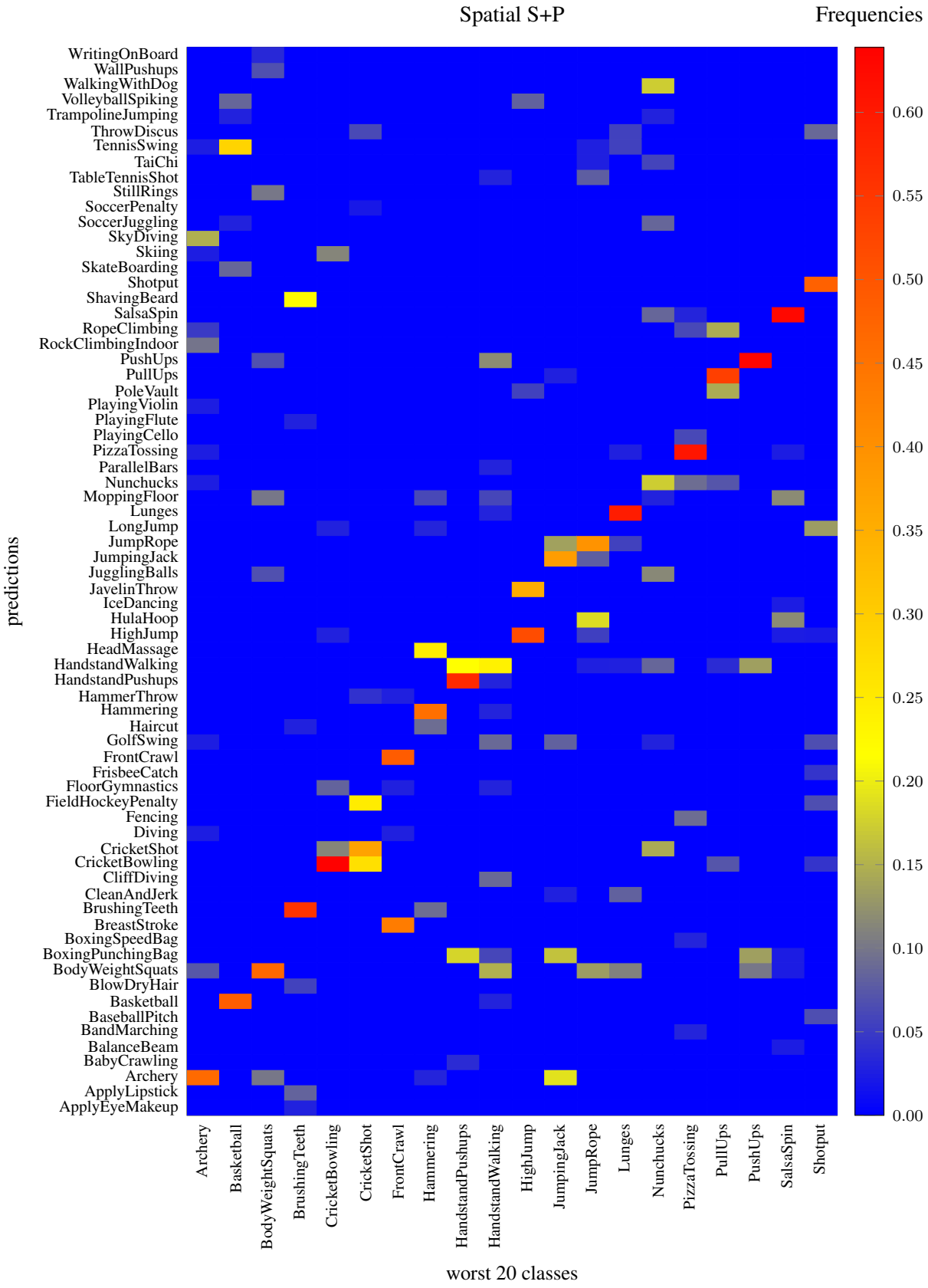
Table A.1: *Categorization of action classes in UCF101 dataset according to their semantic composition.*

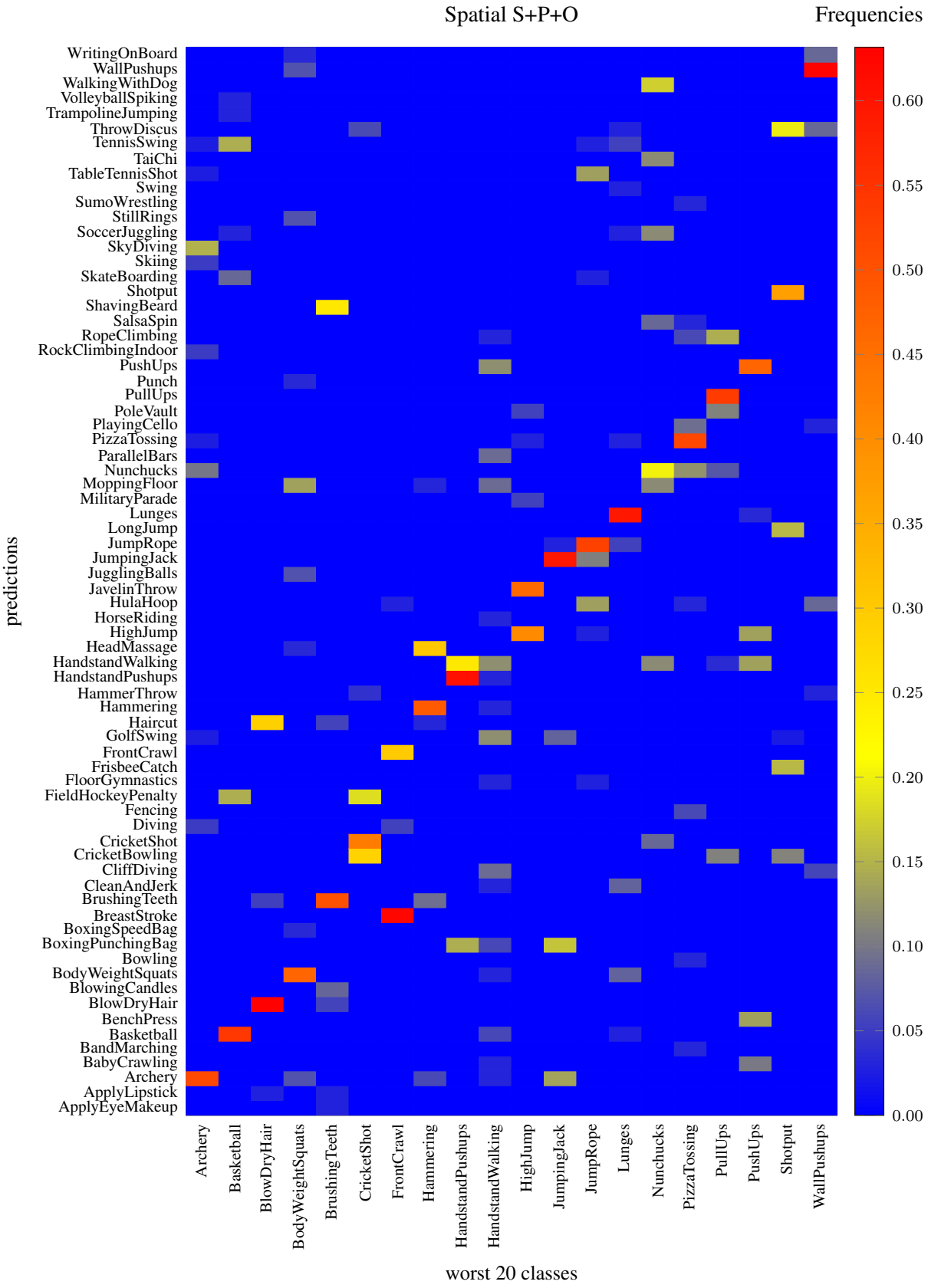
A.2 Object Categories in Faster-RCNN

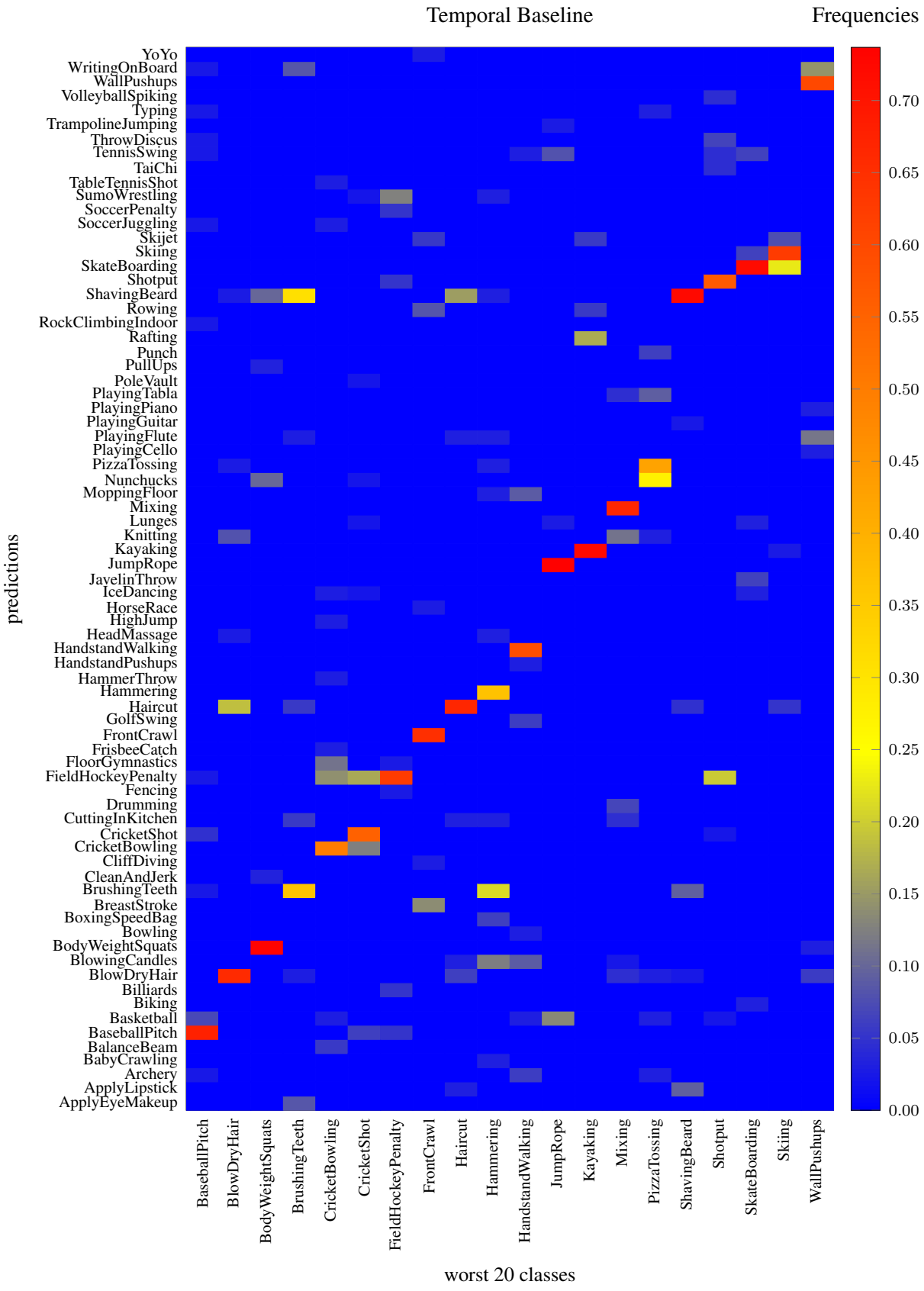
id	Name	id	Name	id	Name	id	Name
1	accordion	34	dishwasher	66	nail	98	stove
2	airplane	35	drum	67	neck brace	99	sunglasses
3	axe	36	dumbbell	68	person	100	swimming trunks
4	baby bed	37	electric fan	69	piano	101	table
5	backpack	38	face powder	70	pineapple	102	tennis ball
6	balance beam	39	flower pot	71	ping-pong ball	103	tie
7	band aid	40	frying pan	72	pizza	104	toaster
8	baseball	41	golf ball	73	plastic bag	105	traffic light
9	basketball	42	golfcart	74	pomegranate	106	train
10	bathing cap	43	guitar	75	popsicle	107	trombone
11	beaker	44	hair dryer	76	power drill	108	trumpet
12	bench	45	hair spray	77	pretzel	109	tv or monitor
13	bicycle	46	hamburger	78	printer	110	unicycle
14	bookshelf	47	hammer	79	puck	111	vacuum
15	bow	48	harmonica	80	punching bag	112	violin
16	bowl	49	harp	81	purse	113	volleyball
17	brassiere	50	hat with a wide brim	82	racket	114	washer
18	bus	51	helmet	83	refrigerator	115	water bottle
19	can opener	52	horizontal bar	84	remote control	116	watercraft
20	car	53	horse	85	rubber eraser	117	wine bottle
21	cart	54	iPod	86	rugby ball	118	bottle
22	cello	55	ladle	87	ruler		
23	chain saw	56	lamp	88	salt or pepper shaker		
24	chair	57	laptop	89	saxophone		
25	cocktail shaker	58	lipstick	90	screwdriver		
26	coffee maker	59	maillot	91	ski		
27	computer keyboard	60	microphone	92	snowmobile		
28	computer mouse	61	microwave	93	snowplow		
29	corkscrew	62	milk can	94	soap dispenser		
30	croquet ball	63	miniskirt	95	soccer ball		
31	cup or mug	64	motorcycle	96	sofa		
32	diaper	65	mushroom	97	stethoscope		
33	digital clock						

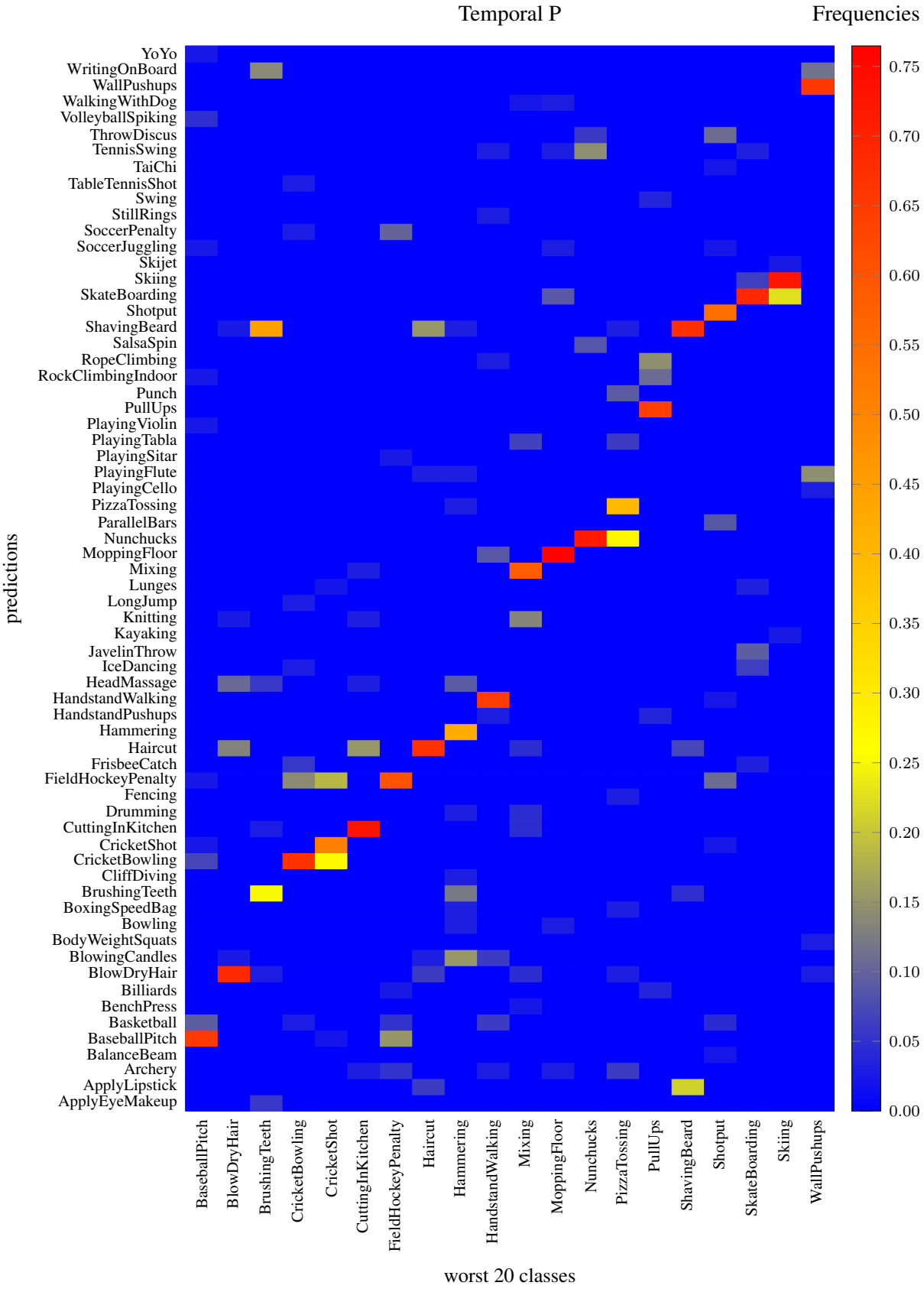
A.3 Confusion Maps

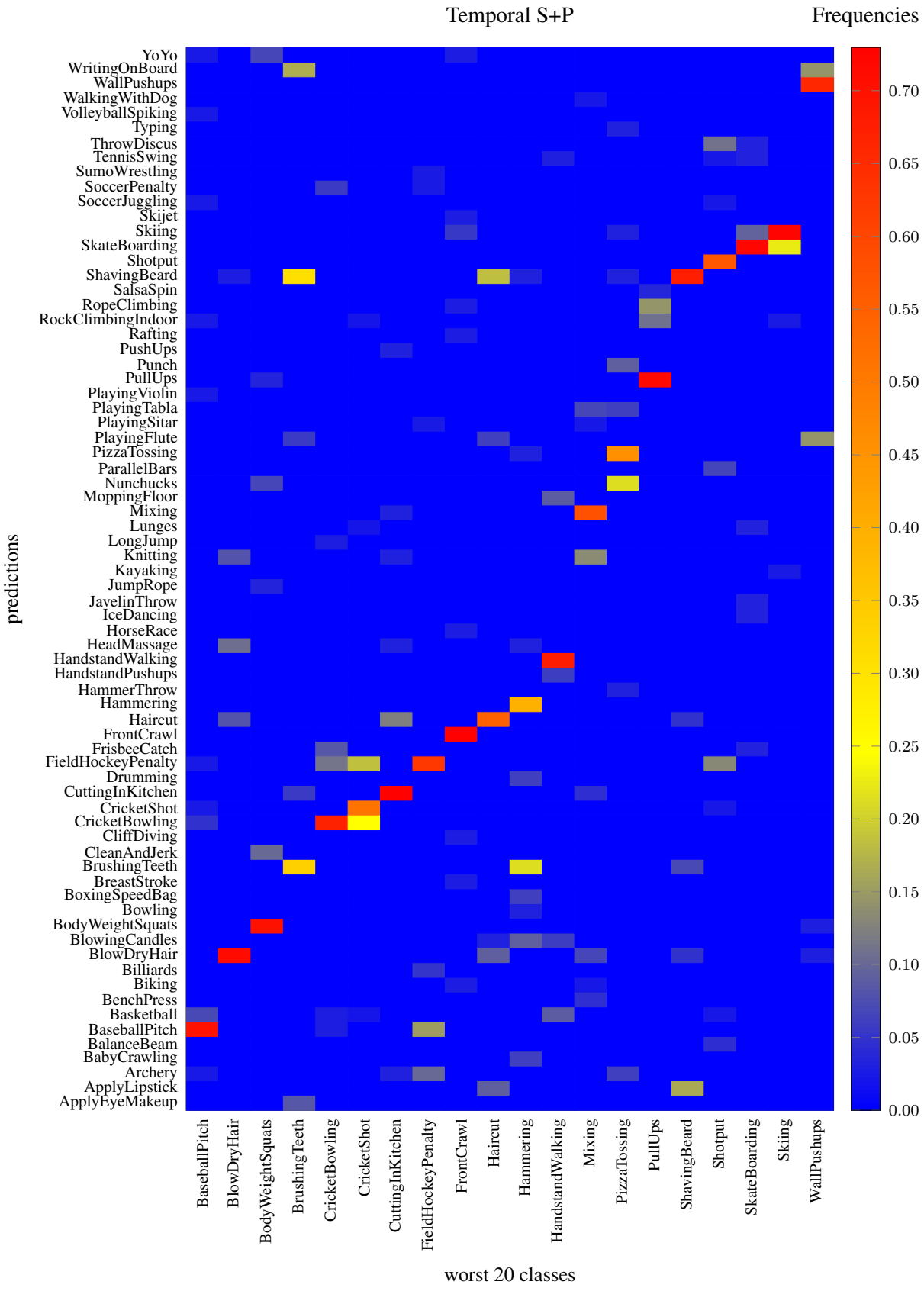



Spatial S+P
Frequencies
predictions
WritingOnBoard
WallPushups
WalkingWithDog
VolleyballSpiking
TrampolineJumping
ThrowDiscus
TennisSwing
TaiChi
TableTennisShot
StillRings
SoccerPenalty
SoccerJuggling
SkyDiving
Skiing
SkateBoarding
Shotput
ShavingBeard
SalsaSpin
RopeClimbing
RockClimbingIndoor
PushUps
PullUps
PoleVault
PlayingViolin
PlayingFlute
PlayingCello
PizzaTossing
ParallelBars
Nunchucks
MoppingFloor
Lunges
LongJump
JumpRope
JumpingJack
JugglingBalls
JavelinThrow
IceDancing
HulaHoop
HighJump
HeadMassage
HandstandWalking
HandstandPushups
HammerThrow
Hammering
Haircut
GolfSwing
FrontCrawl
FrisbeeCatch
FloorGymnastics
FieldHockeyPenalty
Fencing
Diving
CricketShot
CricketBowling
CliffDiving
CleanAndJerk
BrushingTeeth
BreastStroke
BoxingSpeedBag
BoxingPunchingBag
BodyWeightSquats
BlowDryHair
Basketball
BaseballPitch
BandMarching
BalanceBeam
BabyCrawling
Archery
ApplyLipstick
ApplyEyeMakeup
Archery
Basketball
BodyWeightSquats
BrushingTeeth
CricketBowling
CricketShot
FrontCrawl
Hammering
HandstandPushups
HandstandWalking
HighJump
JumpingJack
JumpRope
Lunges
Nunchucks
PizzaTossing
PullUps
PushUps
SalsaSpin
Shotput
worst 20 classes
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Spatial S+P+O
Frequencies
predictions
WritingOnBoard
WallPushups
WalkingWithDog
VolleyballSpiking
TrampolineJumping
ThrowDiscus
TennisSwing
TaiChi
TableTennisShot
Swing
SumoWrestling
StillRings
SoccerJuggling
SkyDiving
Skiing
SkateBoarding
Shotput
ShavingBeard
SalsaSpin
RopeClimbing
RockClimbingIndoor
PushUps
Punch
PullUps
PoleVault
PlayingCello
PizzaTossing
ParallelBars
Nunchucks
MoppingFloor
MilitaryParade
Lunges
LongJump
JumpRope
JumpingJack
JugglingBalls
JavelinThrow
HulaHoop
HorseRiding
HighJump
HeadMassage
HandstandWalking
HandstandPushups
HammerThrow
Hammering
Haircut
GolfSwing
FrontCrawl
FrisbeeCatch
FloorGymnastics
FieldHockeyPenalty
Fencing
Diving
CricketShot
CricketBowling
CliffDiving
CleanAndJerk
BrushingTeeth
BreastStroke
BoxingSpeedBag
BoxingPunchingBag
Bowling
BodyWeightSquats
BlowingCandles
BlowDryHair
BenchPress
Basketball
BandMarching
BabyCrawling
Archery
ApplyLipstick
ApplyEyeMakeup
Archery
Basketball
BlowDryHair
BodyWeightSquats
BrushingTeeth
CricketShot
FrontCrawl
Hammering
HandstandPushups
HandstandWalking
HighJump
JumpingJack
JumpRope
Lunges
Nunchucks
PizzaTossing
PullUps
PushUps
Shotput
WallPushups
worst 20 classes
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Temporal Baseline
Frequencies
predictions
YoYo
WritingOnBoard
WallPushups
VolleyballSpiking
Typing
TrampolineJumping
ThrowDiscus
TennisSwing
TaiChi
TableTennisShot
SumoWrestling
SoccerPenalty
SoccerJuggling
Skijet
Skiing
SkateBoarding
Shotput
ShavingBeard
Rowing
RockClimbingIndoor
Rafting
Punch
PullUps
PoleVault
PlayingTabla
PlayingPiano
PlayingGuitar
PlayingFlute
PlayingCello
PizzaTossing
Nunchucks
MoppingFloor
Mixing
Lunges
Knitting
Kayaking
JumpRope
JavelinThrow
IceDancing
HorseRace
HighJump
HeadMassage
HandstandWalking
HandstandPushups
HammerThrow
Hammering
Haircut
GolfSwing
FrontCrawl
FrisbeeCatch
FloorGymnastics
FieldHockeyPenalty
Fencing
Drumming
CuttingInKitchen
CricketShot
CricketBowling
CliffDiving
CleanAndJerk
BrushingTeeth
BreastStroke
BoxingSpeedBag
Bowling
BodyWeightSquats
BlowingCandles
BlowDryHair
Billiards
Biking
Basketball
BaseballPitch
BalanceBeam
BabyCrawling
Archery
ApplyLipstick
ApplyEyeMakeup
BaseballPitch
BlowDryHair
BodyWeightSquats
BrushingTeeth
CricketBowling
CricketShot
FieldHockeyPenalty
FrontCrawl
Haircut
Hammering
HandstandWalking
JumpRope
Kayaking
Mixing
PizzaTossing
ShavingBeard
Shotput
SkateBoarding
Skiing
WallPushups
worst 20 classes
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Temporal P
Frequencies
predictions
YoYo
WritingOnBoard
WallPushups
WalkingWithDog
VolleyballSpiking
ThrowDiscus
TennisSwing
TaiChi
TableTennisShot
Swing
StillRings
SoccerPenalty
SoccerJuggling
Skijet
Skiing
SkateBoarding
Shotput
ShavingBeard
SalsaSpin
RopeClimbing
RockClimbingIndoor
Punch
PullUps
PlayingViolin
PlayingTabla
PlayingSitar
PlayingFlute
PlayingCello
PizzaTossing
ParallelBars
Nunchucks
MoppingFloor
Mixing
Lunges
LongJump
Knitting
Kayaking
JavelinThrow
IceDancing
HeadMassage
HandstandWalking
HandstandPushups
Hammering
Haircut
FrisbeeCatch
FieldHockeyPenalty
Fencing
Drumming
CuttingInKitchen
CricketShot
CricketBowling
CliffDiving
BrushingTeeth
BoxingSpeedBag
Bowling
BodyWeightSquats
BlowingCandles
BlowDryHair
Billiards
BenchPress
Basketball
BaseballPitch
BalanceBeam
Archery
ApplyLipstick
ApplyEyeMakeup
BaseballPitch
BlowDryHair
BrushingTeeth
CricketBowling
CricketShot
CuttingInKitchen
FieldHockeyPenalty
Haircut
Hammering
HandstandWalking
Mixing
MoppingFloor
Nunchucks
PizzaTossing
PullUps
ShavingBeard
Shotput
SkateBoarding
Skiing
WallPushups
worst 20 classes
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Temporal S+P
Frequencies
predictions
YoYo
WritingOnBoard
WallPushups
WalkingWithDog
VolleyballSpiking
Typing
ThrowDiscus
TennisSwing
SumoWrestling
SoccerPenalty
SoccerJuggling
Skijet
Skiing
SkateBoarding
Shotput
ShavingBeard
SalsaSpin
RopeClimbing
RockClimbingIndoor
Rafting
PushUps
Punch
PullUps
PlayingViolin
PlayingTabla
PlayingSitar
PlayingFlute
PizzaTossing
ParallelBars
Nunchucks
MoppingFloor
Mixing
Lunges
LongJump
Knitting
Kayaking
JumpRope
JavelinThrow
IceDancing
HorseRace
HeadMassage
HandstandWalking
HandstandPushups
HammerThrow
Hammering
Haircut
FrontCrawl
FrisbeeCatch
FieldHockeyPenalty
Drumming
CuttingInKitchen
CricketShot
CricketBowling
CliffDiving
CleanAndJerk
BrushingTeeth
BreastStroke
BoxingSpeedBag
Bowling
BodyWeightSquats
BlowingCandles
BlowDryHair
Billiards
Biking
BenchPress
Basketball
BaseballPitch
BalanceBeam
BabyCrawling
Archery
ApplyLipstick
ApplyEyeMakeup
BaseballPitch
BlowDryHair
BodyWeightSquats
BrushingTeeth
CricketBowling
CricketShot
CuttingInKitchen
FieldHockeyPenalty
FrontCrawl
Haircut
Hammering
HandstandWalking
Mixing
PizzaTossing
PullUps
ShavingBeard
Shotput
SkateBoarding
Skiing
WallPushups
worst 20 classes
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

Bibliography

[1] Apple Inc., *Convolution operations*, https://developer.apple.com/library/ios/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html, vImage Programming Guide.

[2] Richard Bellman, *The theory of dynamic programming*, Tech. report, DTIC Document, 1954.

[3] Léon Bottou, *Large-scale machine learning with stochastic gradient descent*, Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[4] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell, *Long-term recurrent convolutional networks for visual recognition and description*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2625–2634.

[5] John Duchi, Elad Hazan, and Yoram Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, The Journal of Machine Learning Research **12** (2011), 2121–2159.

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

[7] Claudio Fanti, Lihi Zelnik-Manor, and Pietro Perona, *Hybrid models for human motion recognition*, Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, IEEE, 2005, pp. 1166–1173.

[8] Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars, *Modeling video evolution for action recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5378–5387.

[9] James Foulds and Eibe Frank, *A review of multi-instance learning assumptions*, The Knowledge Engineering Review **25** (2010), no. 01, 1–25.

[10] Ross Girshick, *Fast r-cnn*, Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

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

[12] Georgia Gkioxari, Ross Girshick, and Jitendra Malik, *Contextual action recognition with r* cnn*, Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1080–1088.

[13] Abhinav Gupta and Larry S Davis, *Objects in action: An approach for combining action understanding and object perception*, Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[14] Kaiming He, *Convolutional feature maps*, http://kaiminghe.com/iccv15tutorial/iccv2015_tutorial_convolutional_feature_maps_kaiminghe.pdf, ICCV 2015 Tutorial on Tools for Efficient Object Detection.

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

[16] Nazli Ikizler-Cinbis and Stan Sclaroff, *Object, scene and actions: Combining multiple features for human action recognition*, Computer Vision–ECCV 2010 (2010), 494–507.

[17] Mihir Jain, Jan C van Gemert, and Cees GM Snoek, *What do 15,000 object categories tell us about classifying and localizing actions?*, Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE, 2015, pp. 46–55.

[18] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael Black, *Towards understanding action recognition*, Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3192–3199.

[19] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, *3d convolutional neural networks for human action recognition*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **35** (2013), no. 1, 221–231.

[20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, *Caffe: Convolutional architecture for fast feature embedding*, Proceedings of the ACM International Conference on Multimedia, ACM, 2014, pp. 675–678.

[21] Andrey Karpathy Justin Johnson, *Convolutional neural networks for visual recognition*, http://cs231n.github.io/convolutional-networks, Standford CS online notes.

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

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

[24] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, *HMDB: a large video database for human motion recognition*, Proceedings of the International Conference on Computer Vision (ICCV), 2011.

[25] LISA Lab, *Convolutional neural networks (lenet)*, http://deeplearning.net/tutorial/lenet.html, deep learning tutorial.

[26] Ivan Laptev, *On space-time interest points*, International Journal of Computer Vision **64** (2005), no. 2-3, 107–123.

[27] Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld, *Learning realistic human actions from movies*, Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[28] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural computation **1** (1989), no. 4, 541–551.

[29] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng, *Rectifier nonlinearities improve neural network acoustic models*, Proc. ICML, vol. 30, 2013, p. 1.

[30] Aravindh Mahendran and Andrea Vedaldi, *Understanding deep image representations by inverting them*, Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE, 2015, pp. 5188–5196.

[31] Michael Marszalek, Ivan Laptev, and Cordelia Schmid, *Actions in context*, Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 2929–2936.

[32] Darnell J Moore, Irfan A Essa, and Monson H Hayes III, *Exploiting human actions and object context for recognition tasks*, Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference

on, vol. 1, IEEE, 1999, pp. 80–86.

[33] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao, *Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice*, arXiv preprint arXiv:1405.4506 (2014).

[34] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng, *Action recognition with stacked fisher vectors*, Computer Vision–ECCV 2014, Springer, 2014, pp. 581–595.

[35] Florent Perronnin, Jorge Sánchez, and Thomas Mensink, *Improving the fisher kernel for large-scale image classification*, Computer Vision–ECCV 2010, Springer, 2010, pp. 143–156.

[36] Boris T Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics **4** (1964), no. 5, 1–17.

[37] Kishore K Reddy and Mubarak Shah, *Recognizing 50 human action categories of web videos*, Machine Vision and Applications **24** (2013), no. 5, 971–981.

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

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision (IJCV) **115** (2015), no. 3, 211–252.

[40] Paul Scovanner, Saad Ali, and Mubarak Shah, *A 3-dimensional sift descriptor and its application to action recognition*, Proceedings of the 15th international conference on Multimedia, ACM, 2007, pp. 357–360.

[41] Karen Simonyan and Andrew Zisserman, *Two-stream convolutional networks for action recognition in videos*, Advances in Neural Information Processing Systems, 2014, pp. 568–576.

[42] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, CoRR **abs/1409.1556** (2014).

[43] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah, *Ucf101: A dataset of 101 human actions classes from videos in the wild*, arXiv preprint arXiv:1212.0402 (2012).

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.

[45] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri, *Learning spatiotemporal features with 3d convolutional networks*, ICCV, 2015, pp. 4489–4497.

[46] Muhammad Muneeb Ullah, Sobhan Naderi Parizi, and Ivan Laptev, *Improving bag-of-features action recognition with non-local cues.*, BMVC, vol. 10, Citeseer, 2010, pp. 95–1.

[47] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu, *Action recognition by dense trajectories*, Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 3169–3176.

[48] Heng Wang and Cordelia Schmid, *Action recognition with improved trajectories*, Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3551–3558.

[49] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao, *Towards good practices for very deep two-stream convnets*, arXiv preprint arXiv:1507.02159 (2015).

[50] Peng Wang, Yuanzhouhan Cao, Chunhua Shen, Lingqiao Liu, and Heng Tao Shen, *Temporal pyramid pooling based convolutional neural networks for action recognition*, arXiv preprint arXiv:1503.01224 (2015).

[51] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, Xiangyang Xue, and Jun Wang, *Fusing multi-stream deep networks for video classification*, CoRR **abs/1509.06086** (2015).

[52] Yuanjun Xiong, Kai Zhu, Dahua Lin, and Xiaoou Tang, *Recognize complex events from static images by*

fusing deep channels, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1600–1609.

[53] Yaser Yacoob and Michael J Black, *Parameterized modeling and recognition of activities*, Computer Vision, 1998. Sixth International Conference on, IEEE, 1998, pp. 120–127.

[54] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici, *Beyond short snippets: Deep networks for video classification*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 4694–4702.

[55] Matthew D Zeiler and Rob Fergus, *Visualizing and understanding convolutional networks*, Computer vision–ECCV 2014, Springer, 2014, pp. 818–833.