Image-Based Action Recognition Using Hint-Enhanced Deep Neural Networks[☆]

Tangquan Qi[a], Yong Xu[a,*], Yuhui Quan[a], Yaodong Wang[a], Haibin Ling[a,b]

[a]*the School of Computer Science & Engineering, South China University of Technology, Guangzhou 510006, China*
[b]*the Center for Information Science & Technology Computer & Information Science Department Temple University, Philadelphia, PA, USA*

Abstract

While human action recognition from still images finds wide applications in computer vision, it remains a very challenging problem. Compared with video-based ones, image-based action representation and recognition are impossible to access the motion cues of action, which largely increases the difficulties in dealing with pose variances and cluttered backgrounds. Motivated by the recent success of convolutional neural networks (CNN) in learning discriminative features from objects in the presence of variations and backgrounds, in this paper, we investigate the potentials of CNN in image-based action recognition. A new action recognition method is proposed by implicitly integrating pose hints into the CNN framework, i.e., we use a CNN originally learned for object recognition as a base network and then transfer it to action recognition by training the base network jointly with inference of poses. Such a joint training scheme can guide the network towards pose inference and meanwhile prevent the unrelated knowledge inherited from the base network. For further performance improvement, the training data is augmented by enriching the pose-related samples. The experimental results on three benchmark datasets have demonstrated the effectiveness of our method.

Keywords: Action Recognition, Pose Hints, Convolutional Neural Networks

[☆]Yuhui Quan would like to thank the support by National Natural Science Foundation of China (Grant No. 61602184), Science and Technology Program of Guangzhou (Grant No. 201707010147), and Educational Reform Project of South China University of Technology (Grant No. j2jwY9160960). Yong Xu would like to thank the support by National Natural Science Foundations of China (Grant No. U1611461-61672241-61602184 and 61528204), Cultivation Project of Major Basic Research of NSF-Guangdong Province (Grant No. 2016A030308013), and Science and Technology Program of Guangzhou (Grant No. 201707010147).

[*]Corresponding author

Email addresses: qi.tangquan@mail.scut.edu.cn (Tangquan Qi), yxu@scut.edu.cn (Yong Xu), yuhui.quan@scut.edu.cn (Yuhui Quan), w.yaodong@mail.scut.edu.cn (Yaodong Wang), haibin.ling@gmail.com (Haibin Ling)

1. Introduction

Human action recognition aims at recognizing human actions in videos or still images, which is an active topic in computer vision and has a wide range of applications, such as surveillance and human computer interaction [1, 2, 3, 4, 5]. Despite of the efforts made in the past decades, action recognition remains a very challenging task, where the difficulties arise from the cluttered backgrounds, human pose variations, occlusions, illumination changes, and appearance changes in videos. Such difficulties are aggravated for still images, as the motion cues, which play important roles in expressing human actions in videos [6, 7, 8, 9, 10], are completely lost in the images. See Figure 1 for an illustration of the difficulties in image-based action recognition.

1.1. Motivation

To address the aforementioned challenges, we use the-state-of-art deep learning model, convolutional neural network (CNN), to deal with action recognition. Our motivation is that CNN has shown its success in learning discriminative features from objects, even in the presence of cluttered backgrounds or large variations in the appearances and poses of objects. However, traditional CNNs cannot be directly applied to action recognition due to two obstacles:

- Data insufficiency. It is well known that CNN need to be trained on a huge number of images for satisfactory performance. Nevertheless, unlike object recognition, most existing action datasets like Stanford-40 contain a limited number of training images.
- Overfitting. A simple CNN used for action recognition is likely to overfit the appearance of objects as it is not equipped with any prior on human action. For instance, an overfitting CNN might distinguish the action of playing volleyball only via detecting the volleyball.

To deal with the problem of data insufficiency, in this paper, we investigate the transfer of CNN from object recognition to action recognition and design an effective data augmentation scheme. This work is inspired by the fact that the training dataset for object recognition is significantly more than that of action recognition. However, the CNN learned from objects emphasizes the appearance of objects and thus using such a CNN as the base network in transfer is likely to aggravate the aforementioned overfitting problem. To alleviate the overfitting, we resort to the hints given by poses, which are very important for recognizing actions, and then we develop a hint-enhanced CNN that can simultaneously and effectively utilize the hints from both the appearance and pose for action recognition from still images.

1.2. Contribution

In this paper, we develop a new CNN for utilizing pose hints in action recognition from still images, which incorporates a task of pose inference into the base CNN that originates from object recognition. By exploiting the pose hints,

Figure 1: Examples of action images in the Stanford-40 dataset. It can be observed that human performing the same action may look very different, and cluttered backgrounds, human pose variations, occlusions, illumination changes and appearance changes are often presented in the images. Our task is to recognize the actions of the people in the images.

the proposed CNN can encode pose cues for action recognition and reduce the unrelated knowledge inherited from the base network. To improve the performance of the transferred network, we augment the data with a pose-sensitive sampling strategy, where image patches are cropped within or around the human bounding box and then used as samples. We evaluated the performance of the proposed method on three widely-used benchmark datasets, including the Stanford-40 Actions dataset [11], the PPMI dataset [12] and the VOC 2012 Actions Dataset [13]. The results show the effectiveness of the proposed method as well as its superior performance to the base network.

1.3. Organization

The rest of this paper is organized as follows. The related work is described in Section 2.1. In Section 3, we present the details of the proposed method. The experimental results are discussed in Section 5, and Section 6 concludes the paper.

2. Preliminaries

2.1. Related Work

Many existing methods for action recognition extract high-level action representations by exploiting the cues from human-object interaction or human poses. The methods based on human-object interaction usually describe the

relative position, the relative size, as well as the overlap between the person and object. See [12, 14] for the examples of such kind of methods.

Human pose can be viewed as the spatial configuration of body parts, which is discriminative to a broad spectrum of actions. For instance, using computer and climbing can be distinguished by the hand poses. The part-based methods (e.g. [15, 16, 17, 18, 19]) are one of most effective methods built upon human pose, which describe a human pose by the corresponding parts of human body. Yang et al. [17] build up a graphical model to represent the relations between different body parts, including the upper-body, legs, left arm and right arm. In the similar spirit, Yao et al. [18] proposed a 2.5D graph for the representation of action, where the nodes corresponding to the key-points of the human body are represented based on their 3D positions and 2D appearances. Then a minimum set of dominating images is selected to cover all possible cases for each action class. A recent popular part-based method is the so-called *poselets* [20] method. Briefly speaking, a poselet is a detector for some specific body part, which in fact is a linear SVM trained on the clustered image patches that share a salient pattern of local pose from a viewpoint. Based on the poselets, Maji et al. [16] proposed the poselet activation vector (PAV) for representing human actions, which calculates the distribution over the poselets.

In recent years, inspired by the success of deep learning in a wide range of applications, many researchers have started to investigate the application of convolutional neural network (CNN) to action recognition. Oquab et al. [21] used an 8-layer CNN for action classification. Hoai [22] proposed an effective pooling method for CNN in action recognition based on a geometrical distribution of regions placed in bounding boxes of images. Gkioxarie et al. [23] trained body part detectors on 'Pool5' features in a sliding-window manner and combined them with the bounding boxes to jointly train a CNN. They also applied contextual cues to build an action recognition system [24]. Simonyan and Zisserman [25] combined a 16-layer CNN with a 19-layer CNN and trained multiple linear SVMs on 'FC7' features from images with bounding boxes.

Different from the above methods [21, 22, 25], we introduce poselets into CNN by integrating an auxiliary pose-inference task into the the training of CNN, which is for learning features related to human poses. One closely-related work to ours is the [23] which used deep version of poselets to capture parts of the human body under a distinct set of poses. The difference between our method and [23] is that, poses are directly used as features in [23], while they are indirectly utilized as hints in our method to regularize the network.

2.2. Convolutional Neural Network

The framework of convolutional neural network (CNN) is first introduced by LeCun et al. [26] with impressive performance in digit recognition, and soon, its variants (e.g. VGG [25], AlexNet [27]) have emerged in multitude. In general, a CNN is forward network with multiple layers, each of which can be a convolutional layer, a nonlinear activation layer, a pooling layer, or a fully-connected layer. The fully-connected layer is a classic layer in neural network which extracts the global information from its input. It is often used as the final layer

to be the classifier in the task. The convolutional layer can be viewed as the weight-sharing version of the fully-connected layer, in which multiple convolution units are used to encode the local structures of input. The filters used for convolution in this layer can be learned to adapt to the input. The simplification from fully-connected layer to convolutional layer not only helps to localize input features but also can reduce the model complexity and make the network easier to train. The activation layer following the convolutional layer is to introduce nonlinearities to the network for better representational power. The pooling layer is usually put between two convolutional layers, which is for reducing the network size and achieving spatial invariance to some degree by down-sampling. It can also introduce nonlinearities to the network.

The representational power of CNN comes from its stacking of layers. The deeper a CNN is, the more complex knowledge it can express. However, training a deep CNN, e.g. using back propagation, requires a large number of labeled data which is often lacked in real scenarios. To overcome the problem, one can resort to regularization of network, transferring networks from other task, and data augmentation.

2.3. Multi-Task Learning

Multi-task learning (MTL) refers to a technique where a main learning task is solved jointly with auxiliary related tasks using a shared input representation. MTL has been successfully applied to various applications of machine learning, from natural language processing [28] to computer vision [29] and drug discovery [30]. In CNN, one feasible way to introduce MTL is that, allowing different task to share the convolutional layers while learning task-specific fully-connected layers for improving the performance. The benefit of using MTL in CNN is that it help CNN to learn the features which is is difficult to be learned in the main task. In this paper, we design an auxiliary task which is to predict the features as an auxiliary task, and then we integrate the auxiliary task into the CNN.

3. Method

3.1. Overview of the proposed method

The method proposed in this paper is built upon the VGG network. The flow chart of the proposed method is summarized in Figure 2. In the training phase, we train a new deep neural network with action images, using a CNN pre-trained from the ImageNet dataset as the initial point. For clearness, we present the details of the proposed method with the following four parts:

- Network architecture (Section 3.2). We choose the CNN architecture developed in [25], because it has demonstrated impressive performance in various image classification tasks, especially in object recognition. We will show how to transfer it from the source task (image classification) to our main task (action recognition).

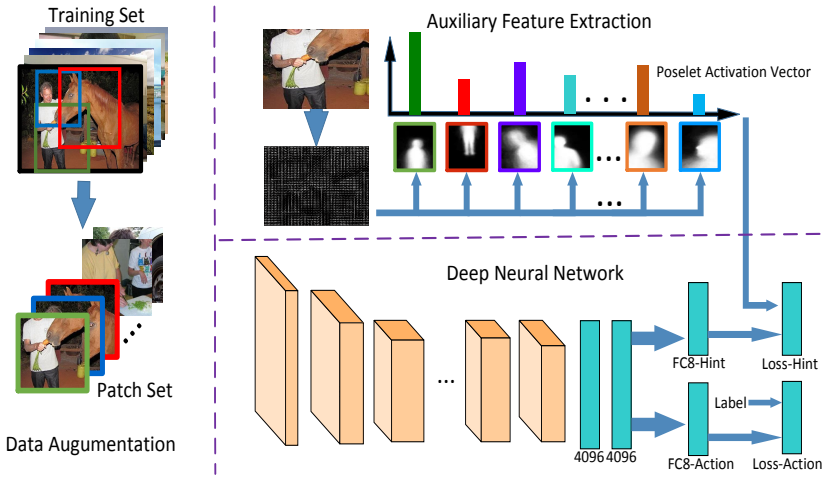

Figure 2: Overview of our proposed method. Left: data augmentation strategy introduced in Section 3.4. Top-right; auxiliary feature extraction for the pose hint task. Bottom-Right: the proposed deep neural network.

- Learning with hints (Section 3.3). We employ pose estimation as the auxiliary task in our framework, which is denoted by "FC8_Hint" in Fig. 2. The task is jointly learned with the action recognition task denoted by "FC8_Action", in order to guide the training process towards capturing more pose information. In particular, we define a joint loss over the two tasks for the guidance.
- Data preparation (Section 3.4). To enrich the training set, we crop image patches from original training images around or within the ground-truth bounding box in the image, which guarantees the new generated training images are closely related to the human poses. Moreover, we prepare data for the auxiliary task with an efficient scheme.
- Action recognition (Section 3.5). Two action recognition strategies are applied to the proposed method: (1) predicting action labels using the softmax scores; (2) training a multi-class linear SVM on the stacked 4096-dimensional CNN features (denoted as '4096D feature') to predict action labels.

3.2. Network Architecture

Inspired by recent studies on transferring CNN-based visual recognition tasks (e.g. [21, 22, 23]), we used the CNN model trained on ILSVRC2015 as the source task and transfered it into our main task (i.e. action recognition). The reason that we chose ILSVRC2015 is that many of its object categories including human are involved in the often-seen object-human interaction in human actions.

These objects, if recognized correctly, are very useful for recognizing human actions. For example, the actions of holding an umbrella, throwing an arrow, and walking the dog can be easily distinguished when umbrella, arrow, and dog are recognized first.

For our action recognition task, we designed a deep neural network architecture in the spirit of VGG [25], which is a popular CNN that has achieved the state-of-the-art performance in the ILSVRC2015 image classification task and shown its effective representations in other visual tasks. The designed neural network is similar to the 16-layer VGG network (denoted by VGG-16), which stacks 13 convolutional layers and 3 fully-connected layers. Such a deep architecture enables the VGG to learn powerful image representations. The implementation details of VGG-16 and our network will be described in 4.

To transfer the knowledge learned from object categories, we firstly pre-train the network weights of VGG-16 from scratch on ILSVRC2015, then we replace the 'FC8' layer with 'FC8_Action' and 'FC8_Hint' as illustrated in Figure 2. Moreover, our proposed neural network is trained with action images using the pre-trained VGG-16 model as the initial point. In the next, we give the details of our transfer method.

3.3. Learning with Hints

In our proposal, we jointly conduct two tasks in the neural network for regularizing the training process and enhancing the representation power, that is, the main task for image representation and the auxiliary task for pose-hint enhancement. In this section, we first give the details of how to design the auxiliary task and then present how to combine the main task with the auxiliary task.

3.3.1. Pose-hint Enhanced Module

We borrow the idea of *hint learning* [31] to address the inconsistency between the original source network (for object recognition) and the target one (for action recognition). The idea is to integrate a special hint to regularize the network so that the network can learn action-related representations and avoid overfitting regarding object appearance. As pose plays an important role in action expression, we use pose hint in our network. In details, we introduce an auxiliary pose-inference task to the original network by sharing hidden layers with the main task, by which the hint information can be encoded into the representation of the main task.

Constructing an ideal module for the auxiliary task requires accurate pose annotation of training data. This is however impractical since it costs too much effort on annotation. Alternatively, we resort to a tool on pose analysis, which is called *poselet activation vector* (PAV) [16] to build the auxiliary task. Roughly speaking, given an image, the PAV measures the distribution of poselet [20] (i.e. various types of poses) in the image. Using PAV as a soft label of pose, we define the auxiliary task as a regression problem that needs to approximate the PAV of images. More specifically, for an input image I, we denote $\phi_{\text{pav}}(I)$ to

be its PAV, and $\phi_{\text{hint}}(I)$ to be the output value regarding pose hint. Then the loss function is defined as follows,

$$\ell_{\text{hint}}(I) = \|\phi_{\text{hint}}(I) - \phi_{\text{pav}}(I)\|^2. \tag{1}$$

In the training stage, the PAV features are extracted from all training images and used as the hint features used by the auxiliary task.

3.3.2. Joint Learning

As illustrated in Figure 2, we juxtapose the auxiliary task along with the main task. The auxiliary task and the main task share all previous layers. To jointly learn the weights of these layers, we use a joint loss function defined as the weighted combination of loss functions for the two tasks. More specifically, let $\mathcal{M}$ be the network model and $\mathcal{D} = \{(I_i, y_i)\}_{i=1}^N$ be the training set of N sample images $\{I_i\}_{i=1}^N$ with associated class labels $\{y_i\}_{i=1}^N$, then the joint loss function is defined as

$$\mathcal{L}(\mathcal{M}, \mathcal{D}) = \mathcal{L}_{\text{main}}(\mathcal{M}, \mathcal{D}) + \alpha \mathcal{L}_{\text{hint}}(\mathcal{M}, \mathcal{D}) \tag{2}$$

where $\alpha \in [0, 1]$ a weight for balancing two loss terms[1], $\mathcal{L}_{\text{main}}$ and $\mathcal{L}_{\text{hint}}$ are the empirical losses for the main task and the auxiliary tasks respectively, which are defined as below:

$$\begin{aligned}\mathcal{L}_{\text{main}}(\mathcal{M}, \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \Big(& y_i \log \mathcal{M}_{\text{main}}(I_i) + \\ & (1 - y_i) \log \left(1 - \mathcal{M}_{\text{main}}(I_i)\right)\Big),\end{aligned} \tag{3}$$

$$\mathcal{L}_{\text{hint}}(\mathcal{M}, \mathcal{D}) = \frac{1}{2N} \sum_{i=1}^{N} \|\mathcal{M}_{\text{hint}}(I_i) - \phi_{\text{pav}}(I_i)\|^2, \tag{4}$$

where $\mathcal{M}_{\text{main}}(\cdot)$ and $\mathcal{M}_{\text{hint}}(\cdot)$ denote the outputs of the main task (FC8_Action) and auxiliary task (FC8_Hint) respectively. The model parameters above are trained or finely tuned using the stochastic gradient descend method. We use stochastic gradient descent algorithm to optimize $\mathcal{L}(\mathcal{M}, \mathcal{D})$. After the computation of the gradient $\partial\mathcal{L}/\partial w$, we use the following rule to update the weights w.

$$w_{t+1} \leftarrow w_t - \lambda \frac{\partial \mathcal{L}}{\partial w_t} \tag{5}$$

[1] In practice, α is determined by changing its value with some stride and evaluating the performance on validation data. See also Section 5.2.1

In (2), there are two loss functions corresponding to two tasks. One is the main task for action recognition, and the other is the auxiliary task for pose-hint enhancement. Such a pipeline belongs to MTL method discussed in Section 2.3. Therefore, our proposed loss function can be viewed as a multi-task objective function ($\mathcal{L}_{MTL} = \mathcal{L} + \alpha\mathcal{L}_{aux}$).

3.4. Data Preparation

3.4.1. Data Augmentation

A deep neural network requires a huge number of training data for satisfactory performance. As the number of action images are limited, we augment the data by enriching the action images that contain significant pose information. Firstly, for each training image I of size $w \times h$, we rescale it to $w_n \times h_n$ so that the shortest side of the image equals 256, and then we randomly crop image patches with a varying-size window of size $s \times s$, where $s = \min(w_n, h_n)/\lambda$ for $\lambda = 1, 1.3, 1.6, 2$. Such a multi-scale sampling scheme helps us to generate images that contain humans of different scales. Secondly, we label the cropped patches according to the size of overlap area between them and the ground-truth bounding box in the image. More specifically, let B be the bounding box of human performing action in I, a sample patch P is treated as positive if it overlaps significantly with B such that

$$|P \cap B| \geq 0.3|P| \quad \text{and} \quad |P \cap B| \geq 0.6|B|,$$

where $|\cdot|$ indicates the area of a box. The sample patches violating the above criterion are treated as negative. A positive patch is assigned to the same action class as I while a negative one is labeled as background.

By collecting all negative samples over the whole training set, we build up a special class named 'background', in which the sample patches have little discriminative information for action recognition. Note that the number of background patches is much larger than other positive action patches, which may cause an unbalance problem during training. To deal with this problem, we only sample 10% of background patches in training. It is also noted that all samples are resized to 224×224. Finally, our augmentation scheme boosts the number of "action" samples by 15 times on average.

3.4.2. Pose Hint Organization

We use PAV [16] for the auxiliary task. The PAV method is about the distribution of poselets, and a poselet is a body part detector trained from annotated data of joint locations of people in images. In details, a poselet is an SVM classifier that is trained to detect certain kind of patches. In implementation, we chose 150 poselets to obtain the auxiliary PAV features.

The computation of PAV for an image in our method is detailed as follows. To eliminate the effect of cluttered background, each image is cropped into multi-scale patches so that each of the patches has sufficiently large intersection area with a given bounding box of human body. In concrete, the intersection area should be larger than 15% of the area of human bounding box, so that

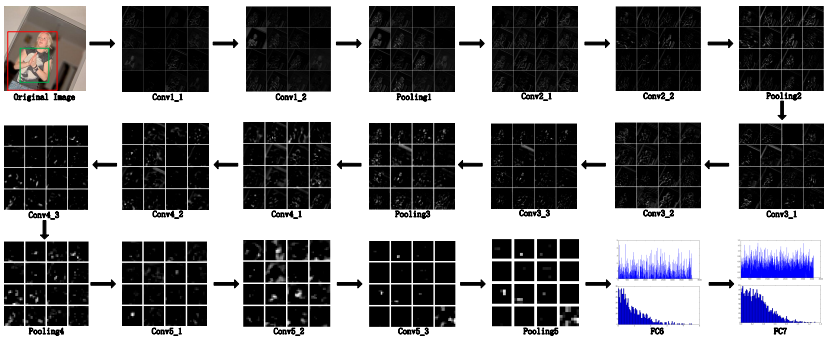

Figure 3: Illustration of the process of extracting the 4096D feature. Each small square figure denotes a feature map. Due to the limitation of space, for each convolutional layer and pooling layer, we randomly choose 16 feature maps for the illustration. For the fully-connected layers ("FC7" and "FC8"), we show two types of feature maps. One is the output of fully-connected layer, and the other is the histogram of the output value.

every patch contains a part of human body. Then the PAV is extracted from each patch, and the final PAV of the image is the average of all PAVs of the cropped patches.

3.5. Action Recognition

After the proposed neural network is trained, we use two strategies for recognizing actions from new images: (1) predicting action labels using the softmax scores, i.e. "FC8_Action"; (2) predicting action labels using a multi-class linear SVM trained on the stacked 4096D image features. The first strategy treats the network as a classifier unified with a feature extractor, while the second strategy considers the network as a pure feature extractor.

The details of the second strategy is as follows. The last fully-connected layers ("FC8_Action" and "FC8_Hint") are removed, and the 4096D image features are extracted from the activations of the penultimate layer "FC7". Given a training image I, image patches are cropped by the scheme described in Section 3.4. Then we feed the image patches into our feature extractor to extract 4096D features and then aggregate them with L_2-normalization to obtain the final 4096D features. Such a process of feature extraction is illustrated in Fig. 3. After that we train a multi-class linear SVM on the extracted features. During test, given a test image I_{test}, we first obtain its 4096D features and make a prediction by the trained SVM.

4. Implementation Details

4.1. Network Configuration

We use the VGG-16 model as the basic model to design our neural network. The network configuration of VGG-16 is shown in Table 1. The size of convolutional kernel is set to 3×3 for all convolutional layers. All hidden layers

Table 1: Network configurations. The size of kernel is 3×3 in all convolutional layers. For brevity, the ReLu activation function is not shown in this table.

Layer	Type	Channel/Size	MaxPooling	Dropout
input	input	224×224	$\times$	$\times$
1_1	conv	64	$\times$	$\times$
1_2	conv	64	$\surd$	$\times$
2_1	conv	128	$\times$	$\times$
2_2	conv	128	$\surd$	$\times$
3_1	conv	256	$\times$	$\times$
3_2	conv	256	$\times$	$\times$
3_3	conv	256	$\surd$	$\times$
4_1	conv	512	$\times$	$\times$
4_2	conv	512	$\times$	$\times$
4_3	conv	512	$\surd$	$\times$
5_1	conv	512	$\times$	$\times$
5_2	conv	512	$\times$	$\times$
5_3	conv	512	$\surd$	$\times$
FC6	full-connected	4096	$\times$	$\surd$
FC7	full-connected	4096	$\times$	$\surd$
FC8	full-connected	Num. of Classes	$\times$	$\times$

use the rectification (ReLU) activation function. The max pooling is performed over 3×3 spatial windows with stride 2. The training is regularized by using dropout for the first two fully-connected layers with dropout ratio 50%. The number of neuron units of "FC8_Action" is set the same as the number of action classes, and the dimension in "FC8_Hint" is 150.

4.2. Training

The network weights are learned via the mini-batch stochastic gradient descent with momentum. The batch size is set to 256 and the momentum is set to 0.9. The weight decay and dropout are used to regularize the training, and the L_2 multiplier is set to 5×10^{-4}. The training procedure of VGG-16 follows the standard procedure [25]. In details, the learning rate is initially set to 10^{-2}, and then decreased according to a factor of 10 when the accuracy on validation set do not show further improvement. In our experiment, the learning rate is decreased 3 times, and the learning is stopped after $370K$ iterations (74 epochs). In the training of both the model and our neural network, the learning rate decreased to 10^{-4} after 3000 iterations, and the training stopped after $10K$ iterations. Meanwhile, other hyper parameters are set to the same values as VGG-16 [25].

When training the VGG-16, a sub-image with fixed size 224×224 is randomly cropped from the selected 256×256 image. Then it undergoes random horizontal flipping and RGB jittering. In training, all sampled patches are resized into

224×224 before they are fed into network (as described in 3.4). Our method is implemented based on the Caffe toolbox [32] with the NVIDIA Tian X GPU.

4.3. Test

During, given a test image I_t, we use the data augmentation described in Section 3.4 to sample test image patches. Subsequently, these patches undergo random horizontal flipping and RGB jittering before they are fed into network. As mentioned in 3.5,two recognition methods are used to make predictions. In the first method, the class scores for I_t are then obtained by averaging the scores (from "FC8_Action") across the sampled patches. In the second method, we first extract 4096D image features for the test image patches. Then, the global average pooling is carried out on the resulting features, and this produces a 4096D feature. Finally, we make a prediction by running the trained SVM on the normalized 4096D features.

5. Experiments

We evaluated the proposed method by applying it to action classification and comparing the results with several state-of-the-art methods. The evaluation is carried out on three public benchmark datasets, including Stanford-40 Actions [11], PPMI [33] and PASCAL VOC 2012 Actions [13]. For fair comparison, the results of all the compared methods are from the standard evaluation protocols. We also evaluated the effectiveness of different components in our method, including the auxiliary task, the augmentation data, and the balancing parameter. Then we tested the influence of the size of data to our method.

In order to check the contribution of different modules in our method, we generate two variants from the proposed method as the baseline methods. Throughout the paper, we use the notations to denote different versions of our method as follows.

- Base: the baseline model which is trained without the auxiliary task.
- Base + Hint: the proposed method with the auxiliary task and softmax prediction.
- Ours: the complete version of the proposed method.

5.1. Classification on Real Datasets

5.1.1. Stanford-40 Dataset

The Stanford-40 dataset [11] contains 9532 images with 40 diverse daily human actions, such as *applauding*, *brushing teeth*, *jumping*, *holding umbrella*, *riding horse*, *using computer*, etc. The images in each class have large variations regarding human pose, appearance, and background clutters, as shown in Figure 4. We built up 10 splits from the dataset. In each split, we randomly selected 100 images for training and used the remaining images for test. Then the mean average precision (mAP) is calculated as the criterion to measure the performance of the compared methods.

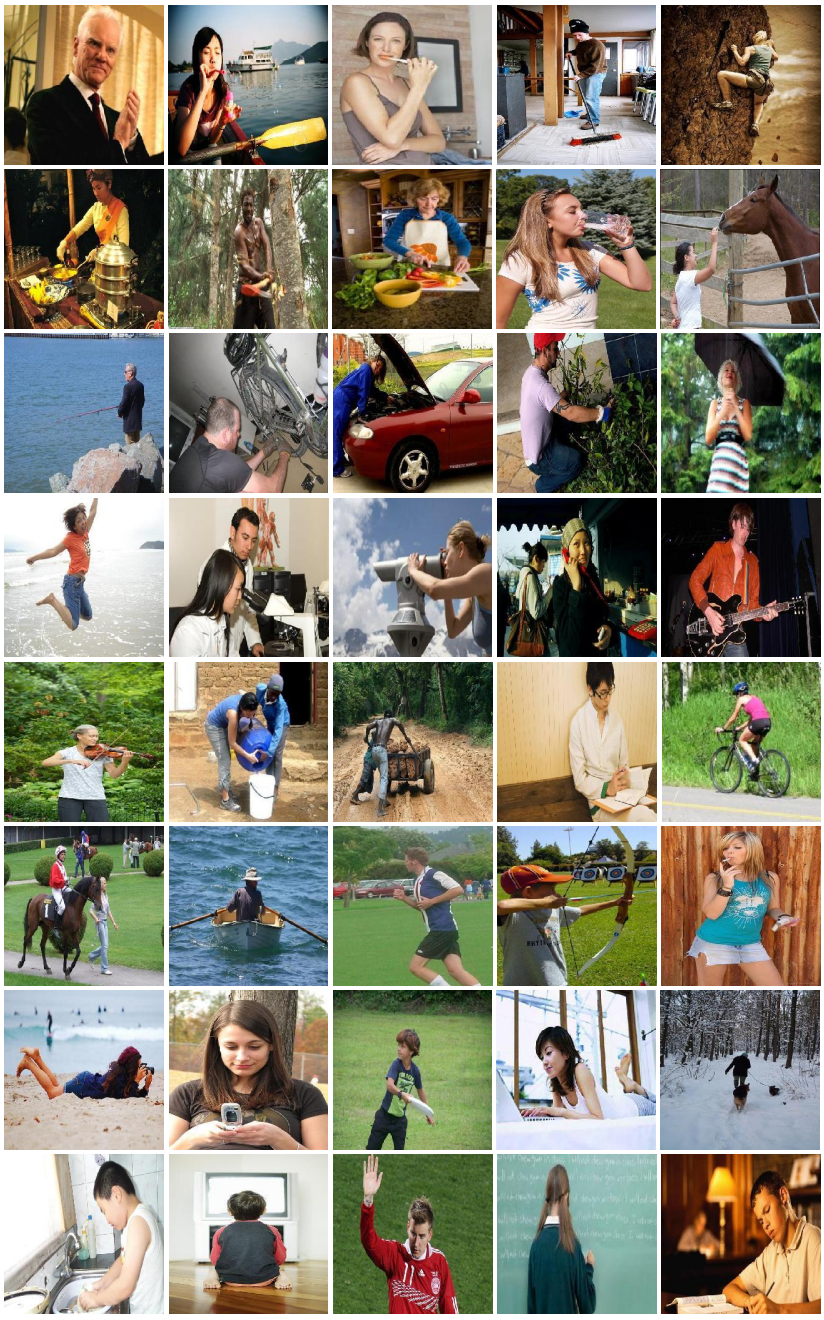

Figure 4: Example images from the Stanford-40 dataset, with one image per action class.

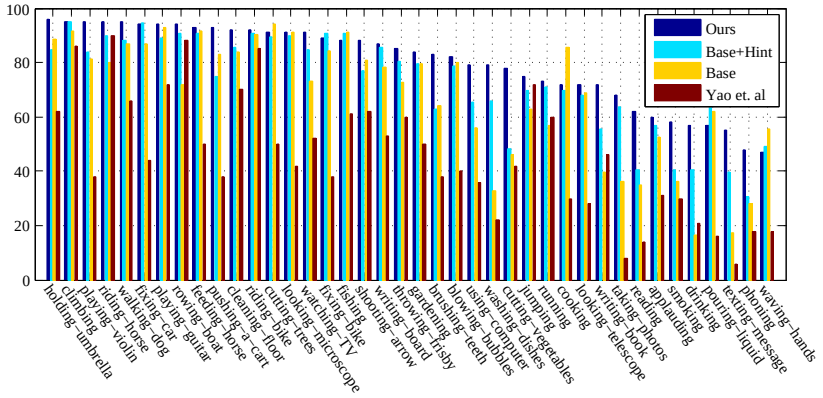

Figure 5: Average precision at each class on the Stanford-40 dataset.

Table 2: Comparison of the proposed method with Poselets [16], Yao et.al. [11], Khanet.al. [19], and Simonyan [25] regarding mAP. The bold font indicates the best performance.

Methods	Poselets [16]	Yao [11]	Khan [19]	Simonyan [25]	Base	Base+Hint	Ours
mAP (%)	11.25	45.57	51.90	72.40	71.57	76.73	**80.69**

We compare the results of our method with [16, 11, 19, 25]. The details of these compared methods are as follows:

- Poselets [16]: a part-based method using the PAV features.
- Yao et al. [11]: a method learning a set of sparse bases of action attributes and parts.
- Khan et al. [19]: a method combining color and shape descriptors.
- Simonyan and Zisserman [25]: a deep learning method which combines a 16-layer network with a 19-layer network and trains a linear SVM on the FC7 features.

The mAP results are shown in Table 2, in which our method achieves the best performance among all the compared methods. Compared with [16, 11, 19] which are built upon the hand-designed features such as HOG and color descriptors, our network can learn the deep features which are much more adaptive to data. Compared to the baseline method (without hints), the proposed method has more than 5% accuracy improvement, which demonstrates the effectiveness of the auxiliary task. Such effectiveness is further demonstrated by the superior performance of our method to the deep learning method [25].

To investigate the performance of our methods in different action classes, we show the average precision on each class in Figure 5 and compared our results with [11]. It can be seen that our method achieves superior performance

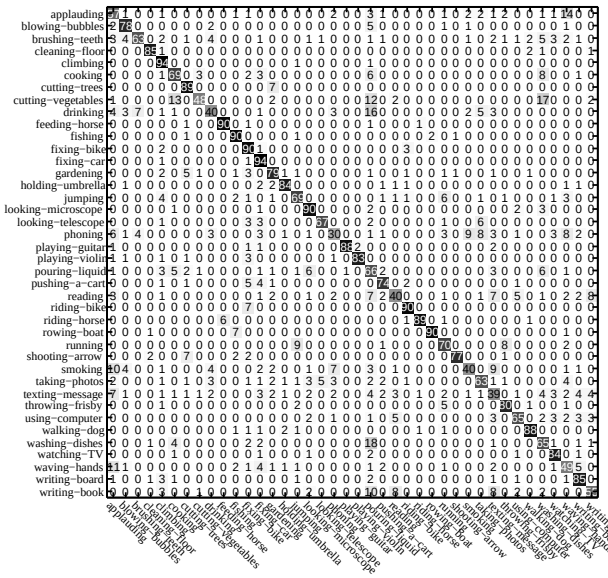

Figure 6: Confusion matrix (%) on the Stanford-40 dataset.

on all the 40 action classes. The performance of our method varies a lot on different classes, ranging from 96% on *"holding an umbrella"* to 47% on *"waving hands"*. In the recognition of *"fixing a bike"*, *"fixing a car"*, *"playing violin"*, *"pushing a cart"*, *"taking photos"*, *"blowing bubbles"*, *"brushing teeth"*, *"cutting trees"*, *"cooking"*, *"looking through a microscope"*, *"looking through a telescope"*, and *"pouring liquid"*, our method works very well. The reason is the human pose changes little in these actions and the objects are very big (e.g. umbrella and bike), which make it easy to utilize hints from poses as well as objects to distinguish human actions.

The objects interacted with human are very small on some action classes, e.g. *"drinking"*, *"taking photos"*, *"texting message"*, *"washing dishes"*, *"reading"* and *"phoning"*. In these cases, our method still performs better than [11], as our hint-enhanced neural network can well utilize the pose cues for action recognition. When action classes share similar human poses and the objects are very small, our method may not work well, as the poses are somehow confusing in such scenarios. See the confusion matrix in Figure 6 as well as some confusing pairs in Figure 7, which includes *"blowing bubble"* and *"brushing teeth"*, *"running"* and *"jumping"*, as well as *"cooking"* and *"cutting vegetables"*. To overcome this weakness, we can resort to context information to distinguish fine-gained poses. We will discuss this issue in Section 5.1.3 and Section 6.

To further study the behavior of our method, we visualize the score maps in Figure 8, which shows the representative high-score images and the corresponding response maps, in which for each action class we compute a response

Figure 7: Examples of the images misclassified on the Stanford-40 dataset. In each misclassified pair, the left image is mislabeled as the action class of the right image. We can see that each misclassified image has a similar pose with its correspondence.

map by averaging the scores of all sampled patches covering a given pixel of the test image. The top-rank false score maps are shown in Figure 9. We randomly selected 8 classes due to space limit. From the response maps we can find that the high-score response of regions locate the discriminative human poses when the involved object is small or the actions do not involve objects. For example, *armed-with-a-mop* is the discriminative pose of class *"cleaning the floor"*, and covered by high-score regions (as shown in the third row of Figure 8). In addition, high-score regions focus on the involved object which is easy to be recognized, such as *"guitar"* (*"playing guitar"*) and *"TV"* (*"watching TV"*).

To further demonstrate the correlation between response maps and recognition accuracy, we manually count the correct locations of discriminative human poses and we call the counting as location accuracy. Figure 10 illustrates the curves of location accuracy and mAP, where the dash lines divide action classes into three groups. We can find that the high location accuracy of action classes ensures the high recognition accuracy in the first two groups. In the last group (from *"cutting trees"* to *"writing on board"*), the same phenomenon is also observed except for the classes *"fishing"*, *"fixing a car"*, *"pushing a cart"* and *"watching TV"*. This is because the involved objects in these classes are easy to recognize and they play important roles in recognition. Therefore, the location accuracy of discriminative poses or objects has positive correlation with the recognition accuracy, as shown in Figure 10. In conclusion, the visualization of score maps has demonstrated that our method can provide an estimate of the action location in the image and encode pose information.

Figure 8: High-score images and the corresponding response maps on the Stanford-40 dataset. The people of interest are marked with red box. Each row denotes a different class form other rows.

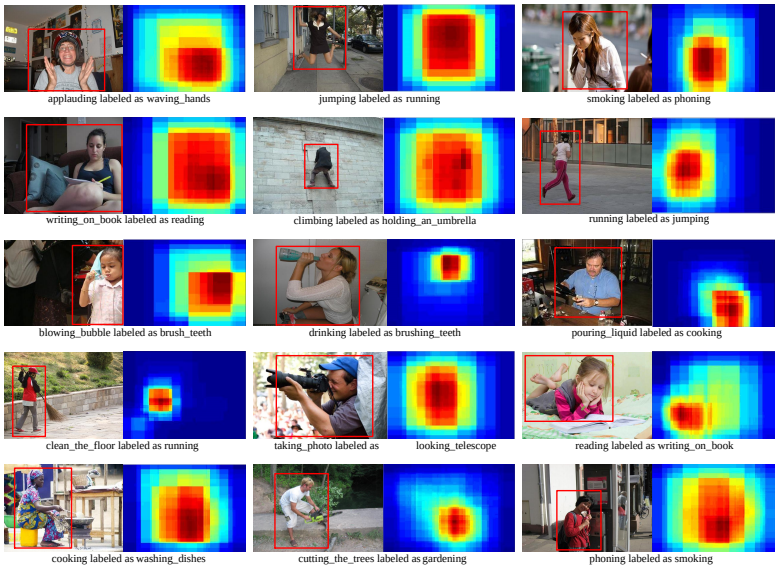

Figure 9: Top-rank false score maps on the Stanford-40 dataset. The people of interest are marked with red box.

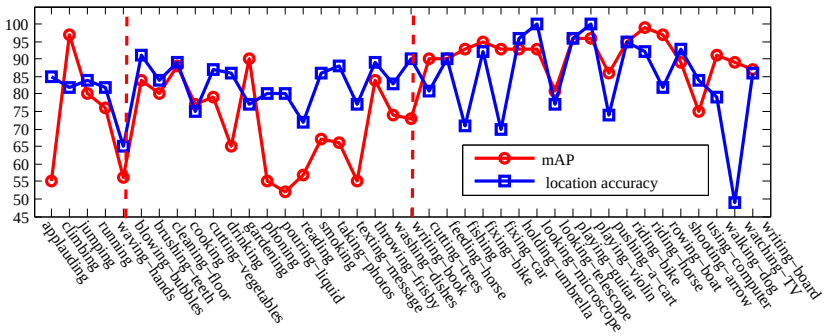

Figure 10: The correlation between mAP and location accuracy, which is calculated on the Stanford-40 dataset. The dash lines divide action classes into three groups: actions without objects (from *"applauding"* to *"waving hands"*), actions with small objects (from *"blowing bubbles"* to *"writing on a book"*), and actions with big objects (from *"cutting trees"* to *"writing on board"*).

Table 3: mAP(%) of compared methods on the PPMI dataset. The bold font indicates the best performance.

Method	Poselets [16]	SPM [34]	Grouplet [33]	Yao [11]	Simonyan [25]	Base	Base+Hint	Ours
mAP (%)	23.40	40.00	42.00	48.00	74.35	72.50	77.80	**82.2**

5.1.2. PPMI Dataset

The People Playing Musical Instruments (PPMI) dataset [12] contains 4800 images of humans interacting with 12 musical instruments, including *bassoon*, *cello*, *clarinet*, *erhu*, *flute*, *French horn*, *guitar*, *harp*, *recorder*, *saxophone*, *trumpet*, and *violin*. The images in the dataset are already cropped and centered to mainly contain the people of interest. A very important property of this dataset is that there are images of people playing the musical instruments as well as images of people holding the instruments without playing. Therefore, there are actually 24 classes in the dataset. Some images from the dataset are shown in Figure 11. It can be seen that the images in PPMI are highly diverse and cluttered.

The protocol on PPMI is similar to that of the Stanford-40 dataset. That is, we built up 10 splits for training and testing our method. In each split, we randomly selected 100 normalized images from each class for training, and the remaining 100 images are used for test. Besides the Poselet method [16] and the deep learning method (VGG) [25] that have been used on the Stanford-40 dataset in last subsection, we select additional three existing methods for comparison, including

- SPM [34]: the classic spatial pyramid matching method.
- Grouplet [33]: a method utilizing the structured information of an image by encoding a number of discriminative visual features and their spatial configurations.
- Yao et al. [12] a random field method which encodes the mutual context of the objects and human poses in human-object interaction.

The results are shown in Table 3. It can be seen that the proposed method outperforms the traditional methods by a significant margin. Again, the proposed method achieves significant accuracy improvement over the baseline methods and the deep learning method [25]. All these performance improvements come from the power of the auxiliary task that boosts the network to encode the cues of human pose.

We also show the mAP at each class in 4, and the confusion matrix is shown in Figure 12. It can be seen that most classes of holding instrument are misclassified as the corresponding classes of playing instrument, such as *"with bassoon"* and *"play bassoon"*. This is because images of each such class pair share the same musical instrument, which implies cues from objects are useless. It is also interesting to calculate from Table 4 the accuracy in recognizing the action of

Table 4: Average precision (%) of compared methods at each class on the PPMI dataset. The bold font denotes the best recognition performance.

Method	SPM [34]	Grouplet [33]	Yao [12]	Base	Base+Hint	Ours
Playing Bassoon	37	30	42	84	86	92
Playing Cello	41	41	50	95	74	81
Playing Clarinet	39	43	45	78	73	85
Playing Erhu	48	43	54	89	88	87
Playing Flute	41	47	53	55	80	79
Playing F. horn	44	38	52	77	90	90
Playing Guitar	40	50	52	89	79	88
Playing Harp	44	36	45	85	84	84
Playing Recorder	45	49	36	71	64	82
Playing saxophone	42	49	47	91	94	93
Playing Trumpet	39	53	47	90	80	86
Playing Violin	43	50	51	87	90	94
With Bassoon	38	41	47	46	62	64
With Cello	42	32	54	35	66	78
With Clarinet	39	39	48	75	82	83
With Erhu	35	35	41	59	63	77
With Flute	48	49	45	51	60	76
With F. horn	36	53	43	82	83	91
With Guitar	34	41	42	52	79	69
With Harp	38	33	47	89	85	88
With Recorder	42	52	52	61	69	68
With saxophone	36	36	50	68	77	82
With Trumpet	39	43	48	68	76	74
With Violin	35	30	45	64	82	81
Total	40	42	48	73	78	82

playing (holding) instrument, which is calculated by the average precision of all action classes related to playing (holding) instrument. The accuracy regarding playing instrument is 81.8%, while the accuracy regarding holding instrument is much lower, i.e. 73.7%. Such a performance gap is due to the fact that the human poses vary a lot within each class of holding instrument.

We also study the score maps by visualization. Figure 13 shows the representative high-score images and the corresponding response maps, and the top-rank false score response maps are shown in Figure 14. Similar conclusion to that on Stanford-40 can be made from these results.

5.1.3. PASCAL VOC Action Dataset

The PASCAL VOC 2012 dataset, which contains about 10000 images from 10 action classes, including *jumping*, *phoning*, *playing instrument*, *reading*, *riding*

Figure 11: Examples images from the PPMI dataset, with one image per action class.

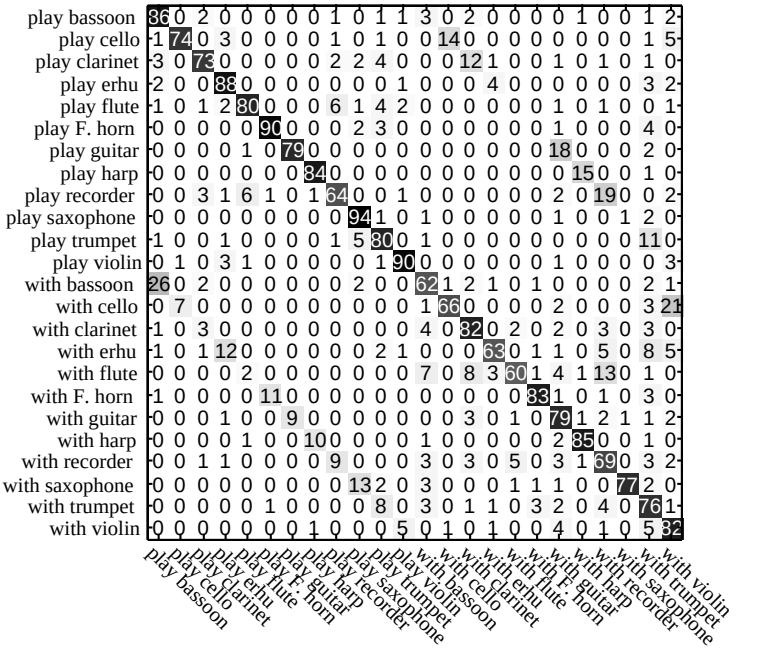

Figure 12: Confusion matrix (%) on the PPMI dataset.

Figure 13: High-score images and the corresponding response maps for the PPMI dataset. The people of interest are marked with red box.

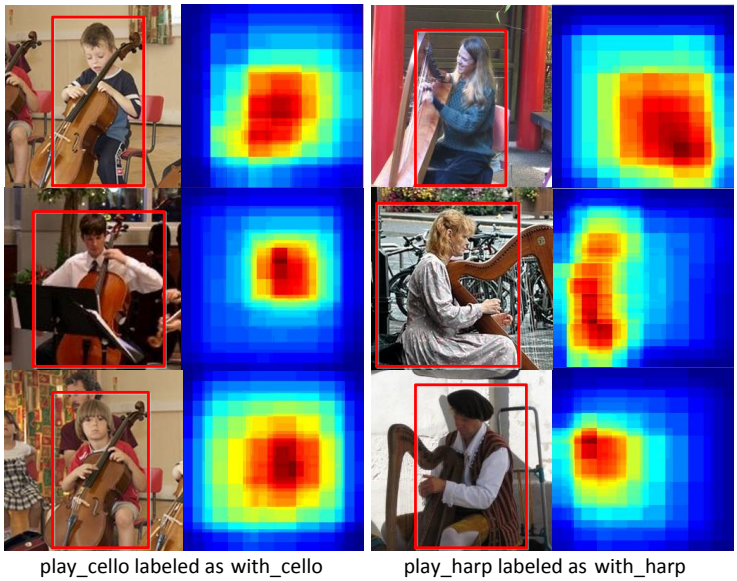

Figure 14: Top-rank false score maps on the PPMI dataset. The people of interest are marked with red box.

bike, *riding horse*, *running*, *taking photo*, *using computer*, and *walking*. It also contains a class called *other* which excludes the above 10 types of actions and it is used to test the robustness to clutters. Some images from this dataset are shown in Figure 15. The dataset provides a training set, a validation set, and a test set. In our evaluation, we combine the training set and validation set for training, and the test set is used for testing.

We compare our method with six approaches from the PASCAL challenge. Besides the Poselet method[16] and the deep learning method [25] which have been used in last subsections, the other four methods for comparison include

- Gkioxiari [23]: a part-based method which trains body part detectors on Pooling5 features with sliding window and combines them with the bounding boxes to jointly train a CNN.
- Oquab [21]: a deep learning method which trains an 8-layer network on ground-truth boxes.
- Hoai [22]: a deep learning method which uses an 8-layer network to extract FC7 features from regions at multiple locations and scales.
- R*CNN [35]: region-based CNN which combines the bounding boxes and its context to investigate the problem of action recognition.

The results are shown in Table 5. Compared with the part-based approaches [16, 23], the proposed method achieves the superior recognition performance on all the classes. Note that [23] use the same base network as ours. The difference is that it directly apply pose inference (i.e., the whole human pose is divided

Figure 15: Example images from the PASCAL VOC 2012 dataset. Here we show one image per action class.

into three parts, and features are extracted from each part and merged as the final features), while we incorporate pose inference into the base network in an implicit way. Such an implicit way help to the base network to learn more powerful features.

Compared with the baseline method (without hints), the proposed method has 2.5% accuracy improvement. From the results, we observed that the improvement mainly comes from the classes *"phoning"*, *"reading"*, *"taking photo"* and *"walking"*. This demonstrates the necessity of introducing the auxiliary task. Compared with [21, 22, 25], our method also achieved the better recognition performance on all classes. This implies our learned pose-related features are discriminative to different types of action. Compared with R*CNN, our approach show superior performance for the classes with action-specific poses, such as *"riding bike"*, *"riding horse"* and *"running"*. However, in many types of VOC 2012, the human poses of the same type of action have very big variations but similar contexts. Regarding such types of action, our method is inferior to R*CNN. One reason is that R*CNN is able to encode the context information which our method ignores. In order to demonstrate that our method can perform as well as R*CNN when combing context information, we modified our method to include the context features which are the same from R*CNN [35], and then we tested the performance of the modified version (denoted by"Ours + Context") of our method on the VOC-2012 dataset. As shown in Table 5, the improved version of our method can achieve better performance than R*CNN.

Table 5: mAP (%) of compared methods on the PASCAL VOC 2012 dataset. The bold font denotes the best performance.

Method	CNN Layers	Jumping	Phoning	Playing Instrument	Reading	Riding Bike
Poselets [16]	-	59.3	32.4	45.4	27.5	84.5
Gkioxari [23]	16	83.7	63.3	87.8	64.2	96.0
Oquab [21]	8	74.8	46.0	75.6	45.3	93.5
Hoai [22]	8	82.3	52.9	84.3	53.6	95.6
Simonyan [25]	16&19	89.3	71.3	94.7	71.3	97.1
R*CNN [35]	16	91.1	83.8	92.2	81.2	96.9
Base	16	88.2	69.0	92.5	66.7	97.1
Base+Hint	16	89.2	72.9	92.3	73.1	97.3
Ours	16	89.7	76.3	94.0	75.3	97.9
Ours+Context	16	**92.5**	**84.3**	**94.3**	**82.7**	**98.0**

Method	Riding Horse	Running	Taking Photo	Using Computer	Walking	mAP(%)
Poselets [16]	88.3	77.2	31.2	47.4	58.2	55.1
Gkioxari [23]	96.7	88.9	75.2	80.0	71.5	80.7
Oquab [21]	95.0	86.5	49.3	66.7	69.5	70.2
Hoai [22]	96.1	89.7	60.4	76.0	72.9	76.3
Simonyan [25]	98.2	90.2	73.3	88.5	66.4	84.0
R*CNN [35]	98.4	93.1	**84.3**	90.9	**77.9**	89.0
Base	97.5	91.8	75.2	86.0	69.0	83.3
Base+Hint	98.2	92.5	80.0	87.6	75.4	85.9
Ours	**98.4**	94.1	82.3	89.5	77.6	87.5
Ours+Context	98.3	**94.8**	84.1	**91.5**	77.8	**89.8**

5.2. Influence Analysis

5.2.1. Influence of the parameter α

A critical parameter of our method is the parameter α defined in the Equation (2), which is used to balance the penalty of the main task and the auxiliary task. We investigate the influence of the parameter α by changing the value from 0 to 1 with stride 0.001 on each dataset, and then plot the resultant mAP in Figure 16. As shown in the Figure 16, our method is not sensitive to small changes of α and the optimal α on each dataset is in similar scale.

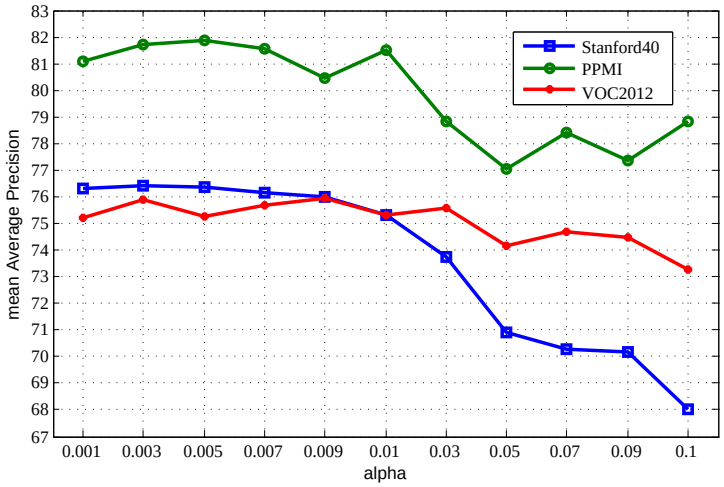

Figure 16: Influence of the parameter α to mAP (%).

5.2.2. Influence of the Data Augmentation and Auxiliary Task

To demonstrate that both the proposed hint-enhanced network and the data augmentation scheme really work, we evaluated the influence of the auxiliary task and data augmentation on the Stanford-40 dataset, by using configurations of modules of our method. The results are shown in Table 6.

It can be observed from Table 6 that the data augmentation strategy can noticeably improve the recognition performance by about 2%, and the auxiliary task can also boost the generalization ability of the proposed method. It can be also seen that the "Com" strategy almost has no benefit over the "Deep" strategy. For example, in "AugCS" outperforms "AugDS" by 0.17% . However, when using the auxiliary task with PAVs, "AugHS" has better accuracy of 3% over "AugCS". In summary, the proposed data augmentation scheme and auxiliary task can steadily boost the performance.

It is worth noting that SVM achieves better performance compared with softmax, as shown in Table 6. This is not surprising, as the SVM classifier is more local than the softmax classifier. Think of the case where two examples

Table 6: The influence of the auxiliary task on the Stanford-40 dataset. The bold font indicates the best performance. "AugData" means the deep model is trained using augmentation data. "Hint" means we use hint task to boost deep model. "Deep" denotes the 4096D deep features, and "Com" is the combined features which are obtained by concatenating "Deep" (4096D features) with PAVs. Meanwhile, "SVM" is a multi-class linear classifier, as well as "Softmax".

Method	AugData	Hint	Deep	Com	SVM	Softmax	mAP
NAug	×	×	×	×	×	√	71.57
NAugDS	×	×	√	×	√	×	75.27
NAugDS	×	×	×	√	√	×	75.31
Aug	√	×	×	×	×	√	76.45
AugDS	√	×	√	×	√	×	77.40
AugCS	√	×	×	√	√	×	77.57
AugH	√	√	×	×	×	√	78.32
AugHS	√	√	×	×	√	×	80.69

x_i and x_j from the 1st class achieves the scores $[20, -10, -10]$ and $[20, 19, 19]$ respectively. The loss of these examples in SVM is closer than that in softmax. This is often useful. Think of the example that a "walking" classifier is likely to spend most of its "effort" on the difficult problem of separating "walking" from "running", which should not be influenced by the examples of "fixing_a_car".

5.2.3. Influence of Size of Data

It is well known that CNN may overfit small data sets. Therefore, we evaluate the influence of the size of the data set to the performance of the proposed method by varying the size of the Stanford-40 dataset. We randomly chose N images within each class for training and 20 images for validation, where N is set to $[1, 5, 10, 20, 30, 40, 50, 60, 70, 80]$ respectively. The results are shown in Figure 17, from which we can observe that the generalization ability of the proposed model increases fast as the the size of data set increases.

6. Conclusion

In this paper, we proposed a new method for action recognition in still images via incorporating human pose hints into the convolutional neural network. The incorporation is implemented by jointly conducting the image recognition task and the pose hint enhancement task by using a weighted loss function. We applied high-level pose descriptions as the auxiliary features to designing the auxiliary task for hint enhancement. We evaluated our method on three challenging benchmark datasets, and our method achieved the state-of-the-art results. We also verified the effectiveness of the components in the proposed network, such as the influences of the auxiliary task, the augmentation data, and the balancing parameter.

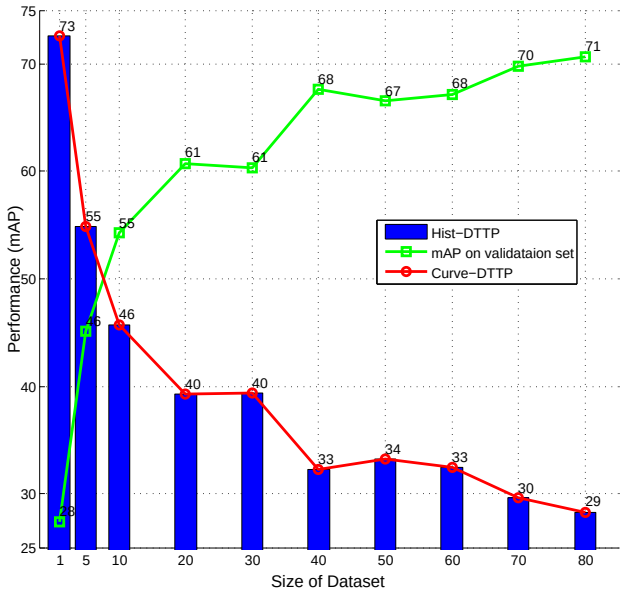

Figure 17: The influence of the data set size for the validation set on the Stanford-40 dataset, which contains 9532 action images. The horizontal axis is the data set size, and the vertical axis denotes the mAP. Hist-DTTP means the histogram of the difference between training and test performance.

In the future, we would like to integrate other useful hints into our network, such as hints from context and human-object-interaction, to achieve further improvement. In addition, we would like to incorporate the pose estimation into our method, and design an end-to-end method.

References

[1] P. Turaga, A. Veeraraghavan, A. Srivastava, R. Chellappa, Statistical computations on grassmann and 1 stiefel manifolds for image and video-based recognition, IEEE Transactions on Software Engineering 33 (11) (2011) 2273–86.

[2] L. Bourdev, S. Maji, T. Brox, J. Malik, Detecting People Using Mutually Consistent Poselet Activations, Springer Berlin Heidelberg, 2010.

[3] M. S. Ryoo, J. K. Aggarwal, Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities, in: IEEE International Conference on Computer Vision, 2009, pp. 1593–1600.

[4] J. Zhu, B. Wang, X. Yang, W. Zhang, Z. Tu, Action recognition with actons, in: International Conference on Computer Vision, 2013, pp. 3559–3566.

[5] C. Yuan, X. Li, W. Hu, H. Ling, S. J. Maybank, Modeling geometric-temporal context with directional pyramid co-occurrence for action recognition., IEEE Transactions on Image Processing 23 (2) (2014) 658–672.

[6] X. Peng, L. Wang, X. Wang, Y. Qiao, Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice, Computer Vision and Image Understanding 150 (2016) 109–125.

[7] A. A. Liu, Y. T. Su, W. Z. Nie, M. Kankanhalli, Hierarchical clustering multi-task learning for joint human action grouping and recognition., IEEE Transactions on Pattern Analysis and Machine Intelligence (2016) 1–1.

[8] A. B. Ben, J. Su, A. Srivastava, Action recognition using rate-invariant analysis of skeletal shape trajectories., IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (1) (2016) 1–13.

[9] C. Wang, Y. Wang, A. L. Yuille, An approach to pose-based action recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[10] Y. Kong, Y. Jia, Y. Fu, Interactive phrases: Semantic descriptionsfor human interaction recognition, IEEE transactions on pattern analysis and machine intelligence 36 (9) (2014) 1775–1788.

[11] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, L. Fei-Fei, Human action recognition by learning bases of action attributes and parts, in: International Conference on Computer Vision, 2011, pp. 1331–1338.

[12] B. Yao, F. F. Li, Modeling mutual context of object and human pose in human-object interaction activities, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 17–24.

[13] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: A retrospective, International Journal of Computer Vision 111 (1) (2015) 98–136.

[14] C. Schmid, A. Prest, V. Ferrari, Weakly supervised learning of interactions between humans and objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 601–14.

[15] C. Thurau, V. Hlavac, Pose primitive based human action recognition in videos or still images, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[16] S. Maji, L. Bourdev, J. Malik, Action recognition from a distributed representation of pose and appearance, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3177–3184.

[17] W. Yang, Y. Wang, G. Mori, Recognizing human actions from still images with latent poses, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 2030–2037.

[18] B. Yao, F. F. Li, Action recognition with exemplar based 2.5d graph matching, in: European Conference on Computer Vision, 2012, pp. 173–186.

[19] F. S. Khan, M. A. Rao, J. V. D. Weijer, A. D. Bagdanov, A. M. Lopez, M. Felsberg, Coloring action recognition in still images, International Journal of Computer Vision 105 (3) (2013) 205–221.

[20] L. Bourdev, J. Malik, Poselets: Body part detectors trained using 3d human pose annotations, in: IEEE International Conference on Computer Vision, 2009, pp. 1365–1372.

[21] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks (2014) 1717–1724.

[22] M. Hoai, Regularized max pooling for image categorization, in: British Machine Vision Conference, 2014.

[23] G. Gkioxari, R. Girshick, J. Malik, Actions and attributes from wholes and parts, in: IEEE International Conference on Computer Vision, 2015, pp. 2470–2478.

[24] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, Computer Science (2014) 580–587.

[25] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.

[26] Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, D. Henderson, Handwritten digit recognition with a back-propagation network, in: Advances in Neural Information Processing Systems, 1990, p. 465.

[27] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems 25 (2) (2012) 2012.

[28] R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, Journal of Parallel and Distributed Computing (2008) 160–167.

[29] R. Girshick, Fast r-cnn, Computer Science.

[30] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, V. Pande, Massively multitask networks for drug discovery, Computer Science.

[31] S. C. Suddarth, Y. L. Kergosien, Rule-injection hints as a means of improving network performance and learning time, Springer Berlin Heidelberg, 1990.

[32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, Eprint Arxiv (2014) 675–678.

[33] B. Yao, F. F. Li, Grouplet: A structured image representation for recognizing human and object interactions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 9–16.

[34] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 2169–2178.

[35] G. Gkioxari, R. Girshick, J. Malik, Contextual action recognition with r*cnn, in: IEEE International Conference on Computer Vision, 2015, pp. 1080–1088.

Tangquan Qi received the B.S. degree in Computer Science and Technology from Binzhou University, Binzhou, China and the M.S. degree in Computer Application Technology from Southwest University, Chongqing, China, in 2008 and 2011, respectively. He is a phD student in the School of Computer Science and Engineering at South China University of Technology, Guangzhou, China. His current research focus on the problem of action recognition, deep learning. His research interests include computer vision, pattern recognition and machine learning.

Yong Xu (M96CSM99) received the B.S., M.S., and Ph.D. degrees in mathematics from Nanjing University, Nanjing, China, in 1993, 1996, and 1999, respectively. He was a Post-Doctoral Research Fellow of computer science with South China University of Technology, Guangzhou, China, from 1999 to 2001, where he became a Faculty Member and where he is currently a Professor with the School of Computer Science and Engineering. His current research interests include image analysis, image and video recognition, and image quality assessment. Dr. Xu is a member of the IEEE Computer Society and the ACM.

Yuhui Quan received his Ph.D. degree in Computer Science from South China University of Technology in 2013. He worked as the postdoc research fellow in Mathematics at National University of Singapore from 2013 to 2016. He is currently the associate professor at School of Computer Science and Engineering in South China University of Technology. His research interests include computational vision, sparse representation, and machine learning.

Yaodong Wang received the B.S. degree from the School of Electronic and Information Technology in South China University of Technology, Guangzhou. He is a Master student in the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His research interests locate in deep learning, image and video processing and action recognition.

Haibin Ling received the B.S. degree in mathematics and the MS degree in computer science from Peking University, China, in 1997 and 2000, respectively, and the PhD degree from the University of Maryland, College Park, in Computer Science in 2006. From 2000 to 2001, he was an assistant researcher at Microsoft Research Asia. From 2006 to 2007, he worked as a postdoctoral scientist at the University of California Los Angeles. After that, he joined Siemens Corporate Research as a research scientist. Since fall 2008, he has been with Temple University where he is now an Associate Professor. Dr. Ling's research interests include computer vision, medical image analysis, human computer interaction, and machine learning. He received the Best Student Paper Award of the ACM Symposium on User Interface Software and Technology (UIST) in 2003, and the NSF CAREER Award in 2014. He has served as a Guest Co-Editor for Pattern Recognition, and as Area Chairs for CVPR 2014 and CVPR 2016.