IET Computer Vision

Research Article

IET Journals
The Institution of Engineering and Technology

Back-dropout transfer learning for action recognition

ISSN 1751-9632
Received on 31st August 2016
Revised 1st December 2017
Accepted on 21st December 2017
E-First on 13th February 2018
doi: 10.1049/iet-cvi.2016.0309
www.ietdl.org

Huamin Ren[1] ✉, Nattiya Kanhabua[2], Andreas Møgelmose[1], Weifeng Liu[3], Kaustubh Kulkarni[4,5], Sergio Escalera[1,4,6], Xavier Baró[4,7], Thomas B. Moeslund[1]

[1]*Department of Architecture, Design and Media Technology, Aalborg University, Aalborg, Denmark*
[2]*NTENT, Barcelona, Spain*
[3]*Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway*
[4]*Computer Vision Center, Barcelona, Spain*
[5]*University of Autonoma Barcelona, Barcelona, Spain*
[6]*Universitat Autonoma de Barcelona, Barcelona, Spain*
[7]*IT, Multimedia and Telecommunications department, Universitat Oberta de Catalunya, Barcelona, Spain*
✉ *E-mail: Huamin.Ren@gmail.com*

Abstract: Transfer learning aims at adapting a model learned from source dataset to target dataset. It is a beneficial approach especially when annotating on the target dataset is expensive or infeasible. Transfer learning has demonstrated its powerful learning capabilities in various vision tasks. Despite transfer learning being a promising approach, it is still an open question how to adapt the model learned from the source dataset to the target dataset. One big challenge is to prevent the impact of category bias on classification performance. Dataset bias exists when two images from the same category, but from different datasets, are not classified as the same. To address this problem, a transfer learning algorithm has been proposed, called negative back-dropout transfer learning (NB-TL), which utilizes images that have been misclassified and further performs back-dropout strategy on them to penalize errors. Experimental results demonstrate the effectiveness of the proposed algorithm. In particular, the authors evaluate the performance of the proposed NB-TL algorithm on UCF 101 action recognition dataset, achieving 88.9% recognition rate.

1 Introduction

During the past decade, widespread use of social media has generated very large data volumes of photos and videos of everyday activities, in all situations, anytime, and at any place. YouTube, the most popular video sharing platform, is visited by nearly one-third of all Internet users (more than a billion), who watch hundreds of millions of hours of video and generate billions of views every day. The rise in online consumption of ever-growing video content has brought an urgent need for advanced video analysis techniques that can enable various end-user applications in order to improve users' experience. In this paper, we address the problem of human action recognition in videos, which is a challenging task and has received a large share of attention, e.g. [1, 2].

Recent works show that convolution neural networks (CNNs) have become state-of-the-art methods for image processing and related research areas [3–6]. The advantages of CNNs are manifold. CNNs can alleviate manual feature engineering, as well as other steps, such as image feature representation, dimensionality reduction, model training, and classifier design. In other words, we only need to provide images as input and expect their categories as output of the deep learning model, which works as a black learning box. For instance, Krizhevsky *et al.* [7] applied CNN models to the ILSVRC-2012 classification challenge and achieved 15.3% top-5 test error, compared to scale invariant feature transform (SIFT) combining Fisher vectors proposed by Deng *et al.* [8] that only achieved 26.2% top-5 test error on the same dataset.

The benefits of CNNs are beyond the convenience they bring: the huge amount of training data collected from a source dataset can be utilised to help in a related task with a different dataset. The source dataset is denoted as $\boldsymbol{S}$ where labelled data is available; the dataset in the target task is denoted as $\boldsymbol{T}$ where labelled data is infeasible or too expensive to obtain. Making use of a model learned from a source dataset and adapted to a target dataset is called *transfer learning*. Transfer learning has demonstrated its powerful learning capabilities in various vision tasks, such as object recognition [9], semantic segmentation [6], and action detection [10].

In this work, we aim at exploiting a CNN model and its potential use of transfer learning for human action recognition in videos. Fig. 1 shows an illustration of transfer learning. Our source dataset $\boldsymbol{S}$ is an image collection aimed at object recognition. Transfer learning aims at fine-tuning the model from the original image collection and adapt it to the video domain in order to detect actions in videos. In contrast to video annotation, the image-level annotation is comparatively easy to acquire. The prevalence of tagged image allows search engines to quickly produce a set of images that have some correspondence to any particular category. Therefore, we may take advantage of the huge amount of image annotations to learn actions in videos.

While transfer learning is a promising approach, it is still an open question how to adapt the model learned from the source dataset to the target dataset. One big challenge is to avoid category bias while transferring a source model to a new target. It has been pointed out in [11] that dataset bias exists even though the same category appears in various recognition datasets. This bias is more serious when transferring a model from one dataset to another. On ImageNet, intra-class variations can exist in most categories. For example, in the category 'zebra' images can represent the animal zebra or zebra crossings. This intra-class variation may lead to failure to distinguish an action containing a zebra from human crossing a road. Transferred models can be even worse if the source dataset and the target dataset contain the same or similar categories, but with slightly different meanings, for instance, the category 'Horse Riding' on ImageNet. This can be misleading in classifying actions on UCF101 dataset since there are two types of actions: 'Horse Race' and 'Horse Riding'. The training image from

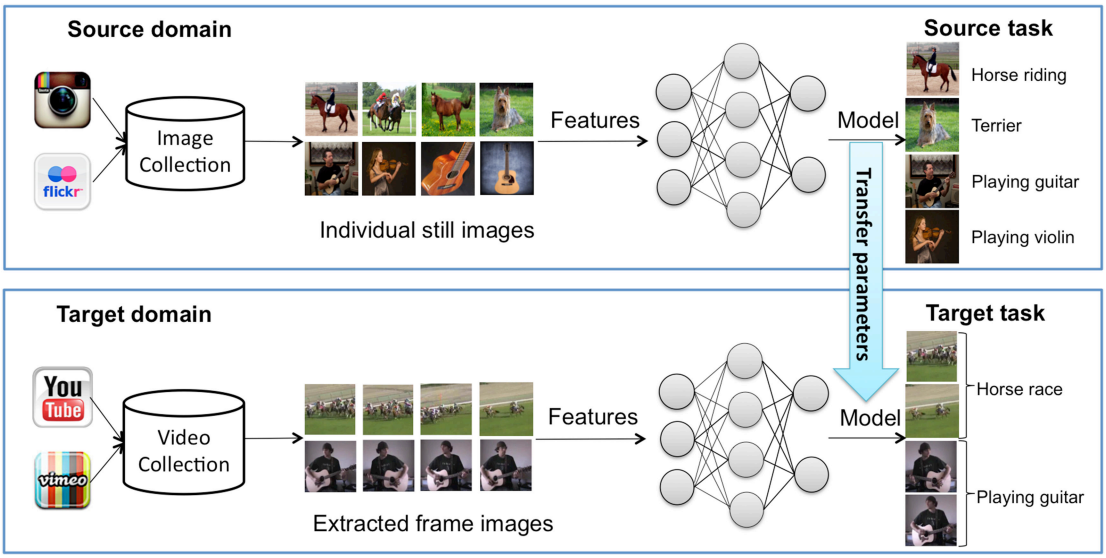

Fig. 1 *Transfer learning learns a model from a source dataset, and adapts the learned model to a target dataset. The figure illustrates transfer learning from ImageNet to UCF101, where the source dataset is for image object recognition, while the target dataset is for video action recognition*

the source dataset has a 'wrong' label while adapting to the target dataset.

In this paper, we address the aforementioned issues. Our main contributions can be summarised as follows:

i. We propose a transfer learning algorithm, namely negative back-dropout transfer learning (NB-TL) algorithm, which adapts a learned model from a source dataset $\boldsymbol{S}$ to a target dataset $\boldsymbol{T}$ by co-adapting weights from the negative samples;
ii. We experimentally show the effectiveness of our NB-TL algorithm in transferring the knowledge from image object recognition to video action recognition.
iii. Our model achieves state-of-the-art performance on the challenging UCF101 action recognition dataset by fine-tuning a CNN_M_2048 model trained on ImageNet2012 and adopting NB-TL algorithm to re-learn the weights.

In the following, we discuss related work in Section 2. In Section 3, we present our transfer learning algorithm, i.e. NB-TL. Experimental results are demonstrated in Section 4. Finally, we conclude our work and outline a future direction in Section 5.

2 Related work

In this section, we discuss related work to our method. We start by detailing the evolution of action recognition methods from hand-crafted features to deep learned features. Then we discuss the transfer learning methods in general and then list transfer learning as applied to deep learned models.

Inspired by the success of the bag-of-features pipeline [12] for object recognition, several approaches have been proposed for action recognition. The bag-of-features pipeline consists of three parts interest point detection, computing descriptors by aggregating local statistics around the interest points and encoding the descriptors computed from the video in a vector representation. Several approaches extend the two-dimensional (2D) interest point detectors to include time so the points can be computed on a video. In [13], the 2D Harris detector is extended by adding time gradients into the second moment matrix. The points can also be densely sampled in the frames of the video [14]. Typically, dense sampling gives better recognition accuracy [14] as compared to detecting interest points. Since these points fail to capture long-range motion information over several frames, Wang and Schmid [15] propose to track the densely sampled points in video. Then around each point or a trajectory descriptors are computed. Appearance information is typically captured with histogram of oriented gradients and motion information with histogram of optical flow [14]. These descriptors are quantised using unsupervised clustering such as *kmeans* or Gaussian mixture models, or alternatively represented sparsely through optimisation [16]. Histogram encoding [17] and Fisher vectors [18] are commonly applied encoding methods. Each video can be encoded into a single vector for a classifier like support vector machine (SVM) [17] or a per-frame time series can be computed for a classfier like hidden Markov model (HMM) [19]. As opposed to local descriptors described until now, global descriptors such as pose [20] can also be utilised. Pose is generally used to estimate in controlled scenarios such as videos collected by Kinect [21]. It is difficult to reliably estimate pose in more realistic videos.

As we can see, above listed methods require several parameters in each of the steps to be hand tuned. Deep learned architechtures learn several million parameters from the training data itself. The recognition accuracy is several fold better than hand crafted features. After the success of CNNs for object recognition [4, 7, 22], several CNN architechtures, for example VGG16 [22], have been applied to the problem of action recognition. In [23], the authors propose a way to model long-term temporal structure of actions and also propose good practices which can maximise the recognition accuracy of a CNN over action datasets. In [24, 25], a two-stream CNN is used; one stream takes video frames as input and the other stream takes the optical flow. The architecture proposed in [26] uses the residual connection between the two streams. Typically, the motion information captured by a CNN is limited to that provided in the optical flow. To capture more long-term temporal information between frames, Ji *et al.* [27] uses a 3D CNN. The long-term temporal information can also be modelled with a recurrent neural nets [28]. In [29], the authors combine the dense trajectory and a CNN to capture temporal information. Most the above approaches fine-tune the CNNs learned on an object recognition task to action recognition. This requires a large amount of annotated action recognition data.

Most deep learned approaches require to estimates millions of parameters. This entails they require a lot of training data. Training data can be difficult or expensive to obtain. In many cases, however, we may have useful training data in a different feature space or data distribution. Transfer learning has emerged as a new framework to improve the performance of learning by utilising labelled data from domains different to the target tasks and/or distributions.

Transfer learning was initially proposed at a conference on neural information processing systems (NIPS) workshop in 1995 and the motivation was original to retain and reuse previously learned knowledge. The definition has later changed to extract the knowledge from one or more *source tasks* and apply the knowledge to a *a target task* [30]. It is worth noting that transfer learning differs from multi-task learning which learns all source and target tasks simultaneously. Instead, transfer learning exclusively takes

knowledge from source tasks – the target task cannot be trained on directly.

There is a multitude of use cases for transfer learning, ranging across a wide spectrum of fields related to machine learning and data mining. Examples include document classification, natural language processing, and computer vision – basically any machine learning domain where annotated data can be difficult to obtain. According to the 2010 survey by Pan and Yang [30], transfer learning was initially used mostly in text labelling, spam filtering, wifi-localisation, and sentiment analysis. All of these, except wifi-localisation, pertain to text analysis. A more recent survey [31] mentions computer vision, along with fields as diverse as natural language processing (related to the earlier text analysis tasks), biology, finance, and business management.

Within computer vision, transfer learning is gaining popularity in various applications, e.g. face verification [32], tracking [33], segmentation [6], and person re-identification [34]. Of greatest interest to this paper is the work done in action recognition [35], and notable recent contributions including [36–39].

It turns out that many deep neural networks trained on natural images exhibit a curious phenomenon: on the first layer they learn features similar to Gabor filters, colour blobs, and other rather generic feature types. Such first-layer features appear not to be specific to a particular dataset or task, but general and applicable to many uses [40]. Recent studies have taken advantage of this fact to obtain state-of-the-art results when transferring from higher layers [41–43], collectively suggesting that these layers of neural networks do indeed compute features that are fairly general. These results further emphasise the importance of studying the exact nature and extent of this generality.

There are already some proposed works that use transfer learning to adapt the model trained on image net to a smaller PASCAL VOC dataset [44]. In [45], transfer learning is applied to adapt the model from object recognition to action recognition. Our work can be regarded as transferring from higher layers with regularisations. Unlike most of the existing algorithms that adopt a dropout [46] to replicate the regularisation, our algorithm regularises weights that are related to misclassified video frames. This regularisation is replicated using our proposed NB-TL method.

3 Negative back-dropout transfer learning

It has been pointed out in [47] that one of the major challenges in developing transfer methods is to produce a positive transfer between appropriately related tasks while avoiding a negative transfer between tasks that are less related. However, unlike the majority of transfer learning method which aim at finding out the most 'positive' information to be transferred, we propose an algorithm to discover the most negative neurons in deep convolutional network, and attempt to promote the performance by suppressing the negative knowledge that is wrongly transferred from the origin domain to the target domain.

In this section, we explain our proposed method. We first define the notations, then we setup the training CNN as a transfer learning problem and then explain our proposed method in terms of the transfer learning problem.

3.1 Transfer learning for classification

Consider a classification problem where we observe a dataset: $S = \{(x_1^{(s)}, y_1^{(s)}), (x_2^{(s)}, y_2^{(s)}), \ldots, (x_i^{(s)}, y_i^{(s)})\}_{i=1}^{m}$ of m labelled training samples. Each sample contains an image $x_i^{(s)}$ which belongs to one of C classes (e.g. 1000 object classes), and each class $c \in \{1, 2, \ldots, C\}$ contains a set of n_c labelled images. We let $x_i^{(s)} \in R^D$ denote the input feature vector of length D for the ith training image, and $y_i^{(s)}$ be its corresponding class label. Similarly, we also have a target dataset: $T = \{(x_1^{(t)}, y_1^{(t)}), (x_2^{(t)}, y_2^{(t)}), \ldots, (x_i^{(t)}, y_i^{(t)})\}_{i=1}^{n}$ of n labelled training samples. Each sample in the target dataset contains an image $x_i^{(t)}$ which belongs to one of K classes (e.g. 101 action classes), and each class $k \in \{1, 2, \ldots, K\}$ contains a set of n_k labelled images. We let $x_i^{(t)} \in R^D$ denote the input feature vector of length D for the ith training image, and $y_i^{(t)}$ be its corresponding class label. Note that:

i. The source and the target dataset are related datasets, and the tasks are also related. For example, if the source dataset is an image object recognition dataset, and the target dataset is a video action recognition dataset, the input of the system is the same: images, despite that they come from different data sources. The task is also similar for these two datasets: both of them aim at recognition and can be considered as a classification problem.
ii. Due to the above property, we may adopt the same feature extraction method for both source and target datasets, therefore, the source input $x_i^{(s)}$ has the same feature dimensionality D as the target input x_i^{t}.

3.2 Problem formulation

First, we formulate our objective in terms of how to minimise the loss function of the target neural networks with respect to the training samples from both the source and target datasets, as well as a convolutional neural model which is trained on the source dataset $h^{(s)}$: $h_\theta^{(s)}(x) = g(\boldsymbol{W}^{(s)}x + \boldsymbol{b}^{(s)})$. Its parameter set is $\theta^{(s)} = \{\boldsymbol{W}(s), \boldsymbol{b}(s)\}$.

In this work, we use the risk function $R(k_i, x_i^{(t)})$ as defined in [48] to measure the risk of classifying $x_i^{(t)}$ to the category k_i. Therefore, to predict the label for an image $x_i^{(t)}$, we need to find the class label k_i which minimises the risk function $R(k_i, x_i^{(t)})$, for $i = 1, \ldots, n$ so that

$$h^*(x^{(t)}) = \arg\min_{k_i \in \{1, \ldots, K\}} \frac{1}{n} \sum_{i=1}^{n} R(k_i, x_i^{(t)} | h(\theta)), \tag{1}$$

where $h(\theta)$ is a learning model and h^* is the objective classification model, which is dependent on the target dataset and also related to the source dataset.

The risk function $R(k_i, x_i^{(t)})$ can be formulated as the loss function of the CNN when k_i and x_i are relevant, formally

$$\begin{aligned} R(k_i, x_i^{(t)}) &= l(r = 1 | k_i, x_i^{(t)}) \\ &= \int_{\Theta_k} \int_{\Theta_{x^{(t)}}} l(\theta_k, \theta_{x^{(t)}}, r = 1) p(\theta_k | k) p(\theta_{x^{(t)}} | x^{(t)}) \, \mathrm{d}_{x^{(t)}} \, \mathrm{d}k \end{aligned} \tag{2}$$

where $r = 1$ represents that the label of $x_i^{(t)}$ is k_i; θ_k and $\theta_{x^{(t)}}$ are the models with respect to classes k and target images $x^{(t)}$, respectively. All possible models regarding categories and input images are represented by Θ_k and $\Theta_{x^{(t)}}$.

3.3 Negative back-dropout transfer learning

Suppose we obtain a CNN that has been trained on the source dataset, therefore, we adopt learned parameter set $\theta^{(s)}$ and use fine-tuning methods to optimise the risk function defined in (1). This is a commonly used transfer learning method to train a target dataset which is very related to the source dataset by adapting very last few layers. The last few layers are called 'specific' layers because they tend to represent features being co-adapted with the new target training data; in contrast, the first few layers are called 'generic' layers because they represent some general features such as edge filters in image processing. After fine-tuning, the parameter set for the target dataset is achieved and denoted as $\theta^{(s)}$. See Fig. 2 for the transfer learning from the source dataset to the target dataset. In this example, the source dataset is for recognising image objects, while the target dataset is for classifying actions in videos.

Replacing $\Theta^{(x^{(t)})}$ with $\theta^{(s)}$, (2) can be rewritten by

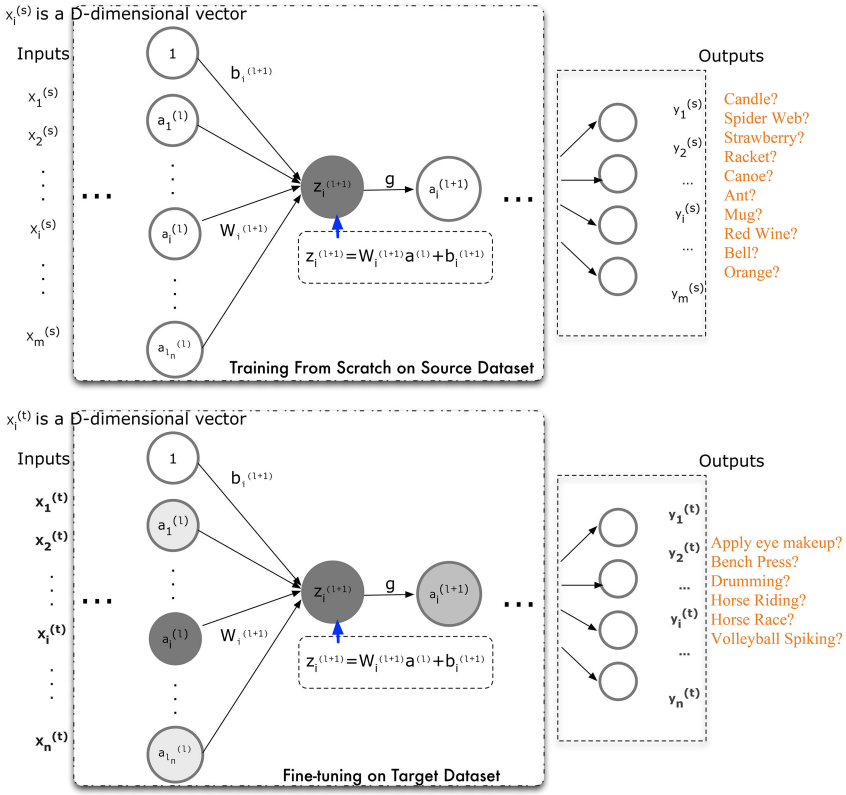

Fig. 2 *Transfer learning from image object recognition to video action recognition. The top figure shows the training on the image object recognition dataset from scratch, the bottom figure shows the fine-tuning procedure on the target dataset. Neurons with dark colours illustrate that weights will be changed during the adapting to the target dataset. Note that the feature dimension for learning from scratch and transfer learning is the same; while the output (category) is different in number and type*

$$l(k_i, x_i^{(t)}) \simeq l(r = 1 | k_i, x_i^{(t)}; \theta^{(s)}) = \int_{\Theta_k} l(\theta_k, r = 1; \theta^{(s)}) p(\theta_k | k)\, \mathrm{d}k + \Omega(\theta^{(s)}) \tag{3}$$

Here $\Omega(\theta^{(s)})$ is some regularisation to prevent overfitting. This regularisation is generally replicated using dropout [46]. In [46], it is shown that dropout works better in preventing overfitting as compared to adding a regularisation term. While transferring from the source to the target database, the loss function becomes:

$$l(k_i, x_i^{(t)}) \propto \int_{\Theta_k} l(\theta_k, r = 1; \theta^{(t)}) p(\theta_k | k)\, \mathrm{d}k + \Omega(\theta^{(t)}) \tag{4}$$

Combining (4) and (1), we obtain the final equation to minimise; this minimisation is done with the backpropogation algorithm and the regularisation $\Omega(\theta^{\mathrm{nb}})$ is replicated with negative back-dropout

$$h^*(x^{(t)}) \propto \underset{k_i \in \{1, \ldots, K\}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \int_{\Theta_k} l(\theta_k, r = 1; \theta^{(\mathrm{nb})}) p(\theta_k | k)\mathrm{d}k + \Omega(\theta^{(\mathrm{nb})}), \tag{5}$$

Negative back-dropout: Now, instead of minimising the transfer learning error directly through fine-tuning and normal dropout [46], we generate a penalty set $x^{(t')}$, which consists of training images from the target dataset that fail to be classified correctly, i.e. $x^{(t')} = \{(x_i^{(t')}, y_i^{(t')})\}$, where $i = 1, \ldots, T$ and $h(x_i^{(t')}) \neq y_i^{(t')}$. We wish to encode some preference for a certain set of weights $W^{(t)}$ over others to reduce the transfer learning error. Therefore, we propose a negative back-dropout to replicate the regularisation term $\Omega(\theta^{\mathrm{nb}})$. In order to achieve this, we search for the neurons with high-value weights for the penalty set, and then discourage large weights through an elementwise penalty by adopting a dropout strategy. This detail is illustrated further in Fig. 3. Different from an ordinary neural network which typically would use either regularisation or standard dropout, negative back-dropout neural network re-learn the network by dropping out neurons that lead to misclassification of images in the penalty set.

For any selected image and its ground truth label $\{(x_i^{(t')}, y_i^{(t')})\}$, the estimated classification based on the transferred learning model $h_{\theta'}(x^{(t)})$ turns out to be a wrong category, i.e. $\hat{y}_i^{(t')} = h_{\theta'}(x_i^{(t')}) \neq y_i^{(t')}$. For any layer $l+1$, we denote the output of the ith neuron of the $(l+1)$th layer as $\hat{y}_i^{(l+1)}$

$$\hat{y}_i^{(l+1)} = g(Z_i^{(l+1)}) \tag{6}$$

$$Z_i^{(l+1)} = \boldsymbol{W}_i^{(l+1)} \hat{y}_i^{(l)} + b_i^{(l+1)} \tag{7}$$

$$\hat{y}_i^{(l)} = g(Z_i^{(l)}) \tag{8}$$

$$\ldots \tag{9}$$

The last few layers are considered to represent specific features adapting the target dataset. Therefore, the weights that have high value in the last few layers give important impacts to the determination of the category labels. The overall penalty, as a result, is a penalty put on the neurons conveying the most wrong information, which is taken through a back-dropout fashion. In other words, the neurons from the second-to-last layer are dropped out, then the neurons from the second last layer are dropout, and so on. For each sample from the penalty set, neurons are dropped in the lth layer together with all the neurons connected to the dropped neurons in the preceding layers.

Finally, dropout is a method to replicate regularisation to prevent overfitting [46]. Typically, it is implemented by randomly setting a certain number of neuron weights and removing their connections to prevent overfitting. In our paper, the negative back-dropout sets the weights of the neurons to zero which contribute the most to misclassification. This is implemented with a penalty image set.

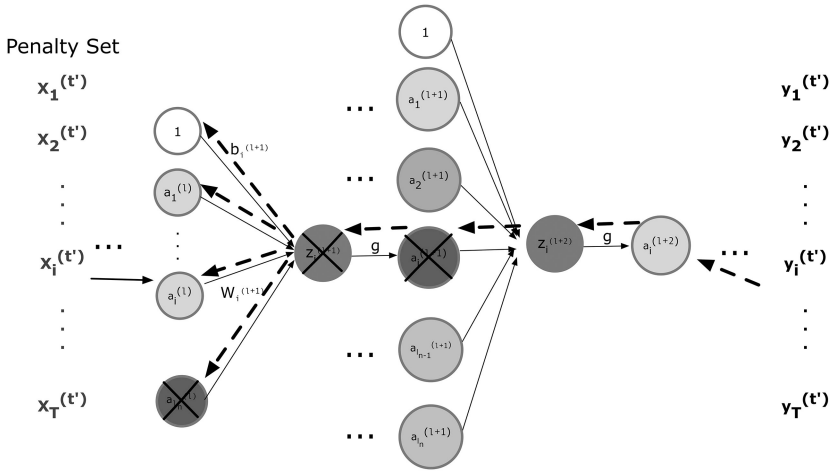

Fig. 3 *Negative back-dropout neural network. Dot lines indicate searching for high weights that contribute to the misclassification of the image i in the penalty set; neurons marked with crossing lines are dropped out*

Our negative back-dropout approach has two main benefits:

i. Dropout has a verified performance in preventing overfitting, especially when the target dataset is significantly smaller than the source dataset [41–43]. Due to the same reason, it also avoids potential overfitting when the target dataset is smaller than the source dataset.
ii. Our back-dropout is based on the classification performance of the penalty set, which consists of misclassified images. Thus, the dropout can be treated as a re-learning of the weights in the neural network but with very little computation.

4 Experiments

We train a model from scratch on the source dataset ***S***: ImageNet ILSVRC-2012, and then transfer the learned model to the target dataset ***T***: UCF101 Action dataset [49]. We give insights of our NB-TL algorithm by demonstrating quantised performance on the target dataset, and then compare our algorithm with other transfer learning methods. Finally, we show comparative results with state-of-the-art methods on two applications.

4.1 Datasets

We adopt the UCF101 dataset as the target dataset for evaluating our method. UCF101 is currently the largest action recognition dataset of realistic action videos, all collected from YouTube. It contains 101 action categories, each of which is divided into 25 groups consisting of 4–7 video clips. It has 13,320 video clips in total. UCF101 gives the largest diversity in terms of actions and has large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, and so on, making the dataset suitable to validate our proposed algorithm.

4.2 Settings for NB-TL on UCF101

We use a pre-trained CNN_M_2048 [50] model on ImageNet ILSVRC-2012 and transfer the learned model from ImageNet ILSVRC-2012 to our UCF101 dataset. The ImageNet dataset is an object recognition dataset with huge amounts of annotated images. Some categories contain similar objects or background as in UCF101 action dataset. For example, in the 'race horse' category, visual information from images containing a horse and a rider may also be used to recognise the action type of 'horse riding' from UCF101. It is worth noticing that we only use UCF 101 dataset while fine-tuning without adding additional datasets as in [2], therefore, the algorithm is more applicable when the target dataset is relatively small compared to source dataset, or hard to be extended. In terms of the model, we use CNN_M_2048 due to its demonstrated effectiveness on ILSVRC2012.

We extract frames from video clips, then down-sample the images to a fixed resolution of 224×224. Given a rectangular image, we first re-scale the image such that the shorter side is of length 224, and then crop out the central 224×224 patch from the resulting image. Videos from the same group may share some common features, such as similar backgrounds or viewpoints. Therefore, if training and test sets are generated through random sampling, it may occur that the training images and the test images come from the same group. This leads to a high accuracy by learning the test to some extent. To avoid this, we follow the same data splitting recommended in [49], and use the training set to learn the model, then report the average accuracy over three splits to compare with the-state-of-the-art methods. Take one split for example, there are 1,785,348 images for training, and 696,976 images for the test; for training, we use 90% to learn the fine-tuned model, and use 10% to create penalty set so that the model can be improved.

The network weights are learnt using the mini-batch stochastic gradient descent with momentum (set to 0.9). At each iteration, a mini-batch of 256 single frames which are randomly selected from the training set is selected. The learning rate is decreased to 0.001 from originally 0.01 in the first few layers, and then boosted on the last two fully connected layer so that the adaptive layers can learn fast. The training was regularised by weight decay (the L2 penalty multiplier set to 0.0005).

4.3 Impacts of back-dropout layers

As has been pointed out by several researchers, the first few layers of CNNs tend to generate general features, which function as filters. Therefore, our penalty set helps to re-learn the model from the errors that have been made, especially aimed at adjusting weights of the last few layers. More specifically, we search for neurons that contribute most to the classification result, and regularise the weights of these neurons to zero. Neurons with large absolute weights are considered as negative contributing neurons, and setting weights to zeros is equivalent to dropping out these function units. We then vary the ratio of dropout neurons from the last, the second-to-last and the last two fully connected layers to demonstrate the impacts of back-dropout layers on the overall performance. Finally, 260 misclassified video clips are selected, and their extracted frames are used to construct the penalty set. The accuracy of our methods is reported in Table 1.

Through the experiment, one penalty image can only determine one neuron to punish. Therefore, once the ratio of the dropout neurons is determined, the number of penalty images that need to be loaded to regularise neurons is also decided. The dropout of the last two layers obtain the best performance, hence we dropout neurons in the last two fully connected layers in the following experiments.

Table 1 Impact of negative dropout strategies on the performance of UCF101 dataset

Negative dropout strategies	Dropped out ratios, %	Accuracy, %
ND_NN + last FC layer	10	68.32
ND_NN + last FC layer	20	67.93
ND_NN + the second last FC layer	10	66.43
ND_NN + the second last FC layer	20	66.25
ND_NN + the last two FC layers	10	69.17
ND_NN + the last two FC layers	20	68.49

Table 2 Impact of the penalty set on the performance of UCF101 dataset. We compare our proposed NB-TL models with different ratio of penalty frames in the validation set and a fine-tuned model of CNN_M_2048, which is also a model built on transfer learning. The negative back-dropout algorithm is applied to the last two fully connected layers. Average precision and recall over all actions are shown

Methods	Number of penalty set	Accuracy, %
fine-tuning of CNN_M_2048	none	66.47
NB-TL + 1/3 penalty video clips in validation set	805 clips	72.24
NB-TL + 2/3 penalty video clips in validation set	1589 clips	79.93
NB-TL + full penalty video clips in validation set	2648 clips	88.20

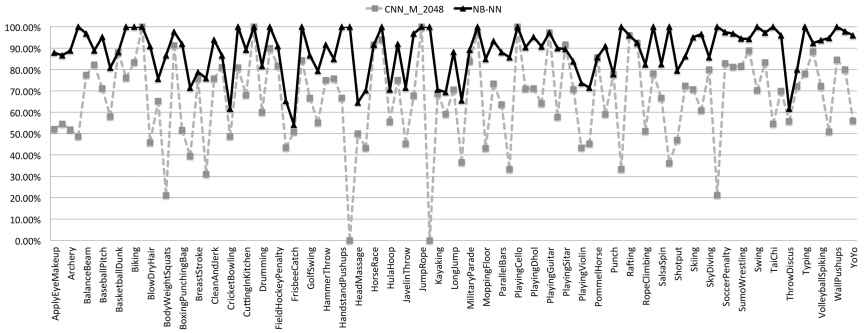

Fig. 4 *Comparative results of per-class precision on the first split of UCF101 dataset: NB-TL versus fine-tuned CNN_N_M_2048 model. X-axis is shown with one category interval*

4.4 Impacts of penalty sets

We classify each image in the validation set based on the model $\theta^{(t)}$, and then treat the image as a penalty image if it is misclassified. Note that through the training–test splitting, a separate group in each action is left out and put into the test set, which means that the visual content of the group has never been learned. Through the training–validation splitting, randomly selected frames from all action categories are loaded to train the model. Therefore, the validation set may share some visual content with the training set, for example, the same object or background. A neuron that fails to recognise a training video clip from an action 'Blowing Candles', may also fail to recognise a validation video clip from the same action. Therefore, by suppressing the neurons with high weights but wrong judgements, the model is refined through correcting 'mistakes'.

We vary the size of the penalty set based on the validation set, and compare their average accuracy over all action classes in Table 2. During the negative back-dropout procedure, weights are ordered in descending order, and the first 10% neurons in the last two fully connected layers are dropped. Not surprisingly, larger penalty set leads to a better performance, which verifies the effectiveness of the penalty sets in improving the model. Compared to the transferred CNN_M_2048 model, our algorithm with frames from 2648 video clips as penalty set achieve an ∼22% increase of accuracy.

4.5 Comparison of NB-TL with transferred CNN_M_2048 model

We compare our NB-TL model with fine-tuned model of CNN_M_2048, and demonstrate average per-class precision and recall in Figs. 4 and 5. For the majority of action categories, these transferred models tend to have the same tendency in classifying actions. This is because NB-TL is still based on the image inputs (the same for the transferred CNN_M_2048 model), moreover, the adaption is mostly taken at the last two layers, similarly to the transferred CNN_M_2048, which is adapting in the last layer. On average, our NB-TL algorithm outperforms the transferred CNN_M_2048 model. For actions such as 'Head Massage', 'Lunges', and 'Throw Discus', both models have relatively low scores; yet NB-TL performs better, which prove the effectiveness of our proposed algorithm.

4.6 Comparison with state-of-the-art methods

We report average accuracy on three training–test splits of UCF101 dataset, and compare the results with state-of-the-art methods in Table 3. Our algorithm has achieved competitive performance on the UCF101 dataset. It is worth noting that our algorithm outperforms the two-stream model only slightly. While looking into per-class recall, our NB-TL model outperforms the two-stream model in action of 'Hammering'. There might be two reasons related to this: temporal information is not taken into consideration, therefore, the action of 'Hammering' can be distinguished by image only information; also, the negative frames from 'Hammering' remove some neurons from the network which help the final classification. There are some actions in which NB-TL is not as good as two-stream model, for example, 'drumming', which is confused by 'Hammering'. Trajectory-pooled deep-convolutional descriptors [29] outperforms our algorithm by 1.4%, however, it combines long-term temporal information by pooling CNN feature maps along dense trajectories. Fisher vectors are computed on these pool features to get a single vector representation for each video. A SVM is learned for classification. Therefore, the higher accuracy comes at a cost of hand tuning the parameters of the Fisher vectors and the SVM. Our NB-TL algorithm fine-tunes on a learned model with no hand tuned

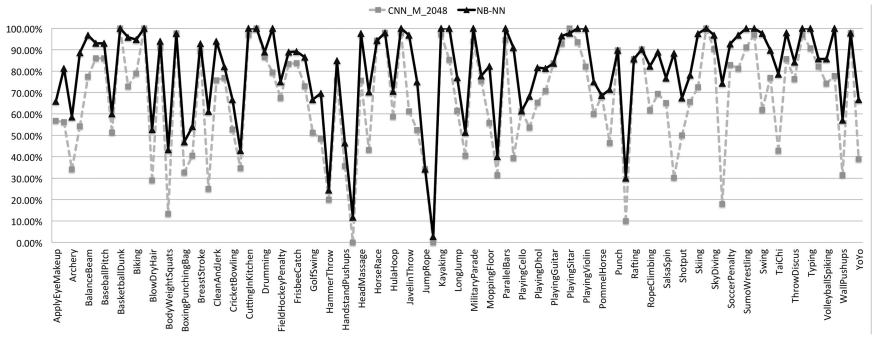

Fig. 5 *Comparative results of per-class recall on the first split of UCF101 dataset: NB-TL versus fine-tuned CNN_N_M_2048 model. X-axis is shown with one category interval*

Table 3 Comparison with state-of-the-art methods on UCF101 dataset

Methods	mAP, %	Year
Soomro *et al.* [49]	43.90	2012
CNN_M_2048 from scratch + 90% dropout [24]	52.3	2014
CNN_M_2048 from scratch + 50% dropout [24]	42.5	2014
improved dense trajectories [15, 51]	85.9	2014
Peng *et al.* [52]	87.9	2014
Cai *et al.* [53]	83.5	2014
Wu *et al.* [54]	84.2	2014
'Slow fusion' spatio-temporal ConvNet [2]	65.4	2014
two-stream model (fusion by averaging) [24]	86.9	2014
two-stream model (fusion by SVM) [24]	88.0	2014
TDD [29]	90.3	2015
NB-TL model	88.9	2016

features which not only saves training time, but also is easy to be applied to other tasks where the target dataset has weak relation to the original one.

5 Conclusion and future work

In summary, we propose a transfer learning algorithm, NB-TL, based on taking advantage of misclassified images and applying a dropout strategy. Experiments demonstrate results on varying the layer to dropout and the size of penalty set to re-train the model. Results show that NB-TL algorithm outperforms the transferred CNN_M_2048 model, and even perform slightly better than the two-stream transferred model. It is also worth noticing that we do not add additional datasets into fine-tuning, which makes our algorithm more applicable when the target dataset is limited or hard to be extended.

Transfer learning is becoming more and more popular. It has shown its potential overwhelming adaptive ability to a target dataset which may be strongly or even weakly related to the source dataset. Regarding video action recognition task, we only use spatial domain knowledge to fine-tune the model, and neglect temporal information totally. Stacked frames which provide motion in videos could also be helpful in understanding actions. Therefore, it would be considered as our future work to help improving the overall recognition performance.

6 Acknowledgments

This work was partially supported by the Spanish projects TIN2016-74946-P and TIN2015-66951-C2-2-R (MINECO/FEDER, UE) and CERCA Programme/Generalitat de Catalunya. This work also has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 752321 and 6655919.

7 References

[1] Jhuang, H., Serre, T., Wolf, L., *et al.*: 'A biologically inspired system for action recognition'. Proc. Int. Conf. on Computer Vision, Rio de Janeiro, Brazil, October 2007, pp. 1–8

[2] Karpathy, A., Toderici, G., Shetty, S., *et al.*: 'Large-scale video classification with convolutional neural networks'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Columbus, Ohio, June 2014, pp. 1725–1732

[3] Xie, J., Xu, L., Chen, E.: 'Image denoising and inpainting with deep neural networks', *Adv. Neural Inf. Process. Syst.*, 2012, **25**, pp. 341–349

[4] Szegedy, C., Liu, W., Jia, Y., *et al.*: 'Going deeper with convolutions'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Boston, Massachusetts, June 2015

[5] Cheng, G., Han, J., Guo, L., *et al.*: 'Learning coarse-to-fine sparselets for efficient object detection and scene classification'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Boston, Massachusetts, June 2015

[6] Long, J., Shelhamer, E., Darrell, T.: 'Fully convolutional networks for semantic segmentation'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Boston, Massachusetts, June 2015, pp. 3431–3440

[7] Krizhevsky, A., Sutskever, I., Hinton, G.E.: 'Imagenet classification with deep convolutional neural networks'. Proc. Annual Conf. on Neural Information Processing Systems, Lake Tahoe, USA, December 2012, pp. 1106–1114

[8] Deng, J., Berg, A., Satheesh, S., *et al.*: 'Ilsvrc-2012', available at http://www.image-net.org/challenges/LSVRC/2012/, 2012

[9] Hoffman, J., Guadarrama, S., Tzeng, E., *et al.*: 'Lsda: large scale detection through adaptation'. Proc. Annual Conf. on Neural Information Processing Systems, Montreal, Canada, December 2014

[10] Gkioxari, G., Hariharan, B., Girshick, R., *et al.*: 'R-CNNs for pose estimation and action detection'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Columbus, Ohio, June 2014

[11] Torralba, A., Efros, A.A.: 'Unbiased look at dataset bias'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Colorado Springs, June 2011, pp. 1521–1528

[12] Csurka, G., Dance, C.R., Lixin, F., *et al.*: 'Visual categorization with bags of keypoints'. Workshop on Statistical Learning in Computer Vision, ECCV, 2004

[13] Laptev, I., Lindeberg, T.: 'Space-time interest points'. 9th Int. Conf. on Computer Vision, Nice, France, 2003

[14] Wang, H., Ullah, M.M., Klaser, A., *et al.*: 'Evaluation of local spatio-temporal features for action recognition'. British Machine Vision Conf., 2009

[15] Wang, H., Schmid, C.: 'Action recognition with improved trajectories'. Proc. Int. Conf. on Computer Vision, Sydney, Australia, December 2013, pp. 3551–3558

[16] Ren, H., Liu, W., Olsen, S., *et al.*: 'Unsupervised behavior-specific dictionary learning for abnormal event detection'. British Machine Vision Conf., 2015

[17] Laptev, I., Marszalek, M., Schmid, C., *et al.*: 'Learning realistic human actions from movies'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Anchorage, Alaska, June 2008, pp. 1–8

[18] Oneata, D., Verbeek, J., Schmid, C.: 'Action and event recognition with fisher vectors on a compact feature set'. IEEE Int. Conf. on Computer Vision, 2013

[19] Kuehne, H., Gall, J., Serre, T.: 'An end-to-end generative framework for video segmentation and recognition'. 2016 IEEE Winter Conf. on Applications of Computer Vision (WACV), 2016

[20] Jhuang, H., Gall, J., Zuffi, S., *et al.*: 'Towards understanding action recognition'. Int. Conf. on Computer Vision (ICCV), 2013

[21] Xia, L., Chen, C.C., Aggarwal, J.K: 'View invariant human action recognition using histograms of 3D joints'. Computer Vision and Pattern Recognition Workshops (CVPRW), 2012

[22] Simonyan, K., Zisserman, A.: 'Very deep convolutional networks for large-scale image recognition', arXiv preprint arXiv:1409.1556, 2014

[23] Wang, L., Yuanjun, X., Zhe, W., *et al.*: 'Temporal segment networks: towards good practices for deep action recognition'. Proc. of the European Conf. on Computer Vision, 2016

[24] Simonyan, K., Zisserman, A.: 'Two-stream convolutional networks for action recognition in videos'. Proc. Annual Conf. on Neural Information Processing Systems, Montreal, Quebec, Canada, December 2014, pp. 568–576

[25] Feichtenhofer, C., Pinz, A., Zisserman, A.: 'Convolutional two-stream network fusion for video action recognition'. Conf. on Computer Vision and Pattern Recognition, 2016

[26] Feichtenhofer, C., Pinz, A., Wildes, R.: 'Spatiotemporal residual networks for video action recognition'. Advances in Neural Information Processing Systems, 2016
[27] Ji, S., Xu, W., Yang, M., *et al.*: '3D convolutional neural networks for human action recognition', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013
[28] Donahue, J., Anne Hendricks, L., Guadarrama, S., *et al.*: 'Long-term recurrent convolutional networks for visual recognition and description'. IEEE Conf. on Computer Vision and Pattern Recognition, 2014
[29] Wang, L., Qiao, Y., Tang, X.: 'Action recognition with trajectory-pooled deep-convolutional descriptors'. IEEE Conf. on Computer Vision and Pattern Recognition, 2015
[30] Pan, S.J., Yang, Q.: 'A survey on transfer learning', *IEEE Trans. Knowl. Data Eng.*, 2010, **22**, pp. 1345–1359
[31] Lu, J., Behbood, V., Hao, P., *et al.*: 'Transfer learning using computational intelligence: a survey', *Knowl.-Based Syst.*, 2015, **80**, pp. 14–23
[32] Cao, X., Wipf, D., Wen, F., *et al.*: 'A practical transfer learning algorithm for face verification'. Proc. IEEE Int. Conf. on Computer Vision, Portland, Oregon, June 2013, pp. 3208–3215
[33] Gao, J., Ling, H., Hu, W., *et al.*: 'Transfer learning based visual tracking with Gaussian processes regression'. Proc. European Conf. on Computer Vision, Zurich, Switzerland, September 2014, pp. 188–203
[34] Layne, R., Hospedales, T.M., Gong, S.: 'Domain transfer for person re-identification'. Proc. the 4th ACM/IEEE Int. Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Stream, ACM Multimedia Conf., Barcelona, Spain, October 2013, pp. 25–32
[35] Cook, D., Feuz, K.D., Krishnan, N.C.: 'Transfer learning for activity recognition: a survey', *Knowl. Inf. Syst.*, 2013, **36**, pp. 537–556
[36] Xu, T., Zhu, F., Wong, E.K., *et al.*: 'Dual many-to-one-encoder-based transfer learning for cross-dataset human action recognition', *Image Vis. Comput.*, 2016
[37] Duan, L., Xu, D., Tsang, I.H., *et al.*: 'Visual event recognition in videos by learning from web data', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012, **34**, pp. 1667–1680
[38] Liu, J., Shah, M., Kuipers, B., *et al.*: 'Cross-view action recognition via view knowledge transfer'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Boston, Massachusetts, June 2015, pp. 3209–3216
[39] Rahmani, H., Mian, A.: 'Learning a non-linear knowledge transfer model for cross-view action recognition'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Boston, Massachusetts, June 2015, pp. 2458–2466
[40] Yosinski, J., Clune, J., Bengio, Y., *et al.*: 'How transferable are features in deep neural networks?'. Proc. Annual Conf. on Neural Information Processing Systems, Montreal, Canada, December 2014, pp. 3320–3328
[41] Zeiler, M.D., Fergus, R.: 'Visualizing and understanding convolutional networks'. Proc. European Conf. on Computer Vision, Zurich, Switzerland, September 2014, pp. 818–833
[42] Donahue, J., Jia, Y., Vinyals, O., *et al.*: 'Decaf: A deep convolutional activation feature for generic visual recognition'. Proc. 31th Int. Conf. on Machine Learning, Beijing, China, June 2014, pp. 647–655
[43] Sermanet, P., Eigen, D., Zhang, X., *et al.*: 'Overfeat: integrated recognition, localization and detection using convolutional networks'. Proc. Int. Conf. on Learning Representations, April 2014
[44] Oquab, M., Bottou, L., Laptev, I., *et al.*: 'Learning and transferring mid-level image representations using convolutional neural networks'. Proc. of the IEEE conf. on Computer Vision and Pattern Recognition, 2014
[45] Jain, M., van Gemert, J.C., Snoek, C.: 'What do 15,000 object categories tell us about classifying and localizing actions?'. Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, 2014
[46] Srivastava, N., Hinton, G.E., Krizhevsky, A., *et al.*: 'Dropout: a simple way to prevent neural networks from overfitting', *J. Mach. Learn. Res.*, 2014
[47] Olivas, E.S., Guerrero, J.D.M., Sober, M.M., *et al.*: 'Handbook of research on machine learning applications and trends: algorithms, methods and techniques', '*Information science reference*' (IGI Publishing, Hershey PA, 2009), no. 2
[48] Dai, W., Chen, Y., Xue, G.R., *et al.*: 'Translated learning: transfer learning across different feature spaces'. Proc. Annual Conf. on Neural Information Processing Systems, Vancouver, B.C., Canada, December 2008, pp. 353–360
[49] Soomro, K., Zamir, A.R., Shah, M.: 'UCF101: a dataset of 101 human actions classes from videos in the wild'. CRCV-TR-12-01, 2012
[50] Chatfield, K., Simonyan, K., Vedaldi, A., *et al.*: 'Return of the devil in the details: delving deep into convolutional nets'. Proc. British Machine Vision Conf., Nottingham, UK, September 2014
[51] Oneata, D., Verbeek, J., Schmid, C.: 'The LEAR submission at Thumos 2014'. ECCV THUMOS Workshop, 2014
[52] Peng, X., Wang, L., Wang, X., *et al.*: 'Bag of visual words and fusion methods for action recognition: comprehensive study and good practice', *Comput. Vis. Image Underst.*, 2016, **150**, pp. 109–125
[53] Cai, Z., Wang, L., Qiao, X.P.Y.: 'Multi-view super vector for action recognition'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Columbus, Ohio, June 2014, pp. 596–603
[54] Wu, J., Zhang, Y., Lin, W.: 'Towards good practices for action video encoding'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Columbus, Ohio, June 2014, pp. 2577–2584