Deep Quantization: Encoding Convolutional Activations with Deep Generative Model *

Zhaofan Qiu, Ting Yao, and Tao Mei
University of Science and Technology of China, Hefei, China
Microsoft Research, Beijing, China
zhaofanqiu@gmail.com, {tiyao, tmei}@microsoft.com

Abstract

Deep convolutional neural networks (CNNs) have proven highly effective for visual recognition, where learning a universal representation from activations of convolutional layer plays a fundamental problem. In this paper, we present Fisher Vector encoding with Variational Auto-Encoder (FV-VAE), a novel deep architecture that quantizes the local activations of convolutional layer in a deep generative model, by training them in an end-to-end manner. To incorporate FV encoding strategy into deep generative models, we introduce Variational Auto-Encoder model, which steers a variational inference and learning in a neural network which can be straightforwardly optimized using standard stochastic gradient method. Different from the FV characterized by conventional generative models (e.g., Gaussian Mixture Model) which parsimoniously fit a discrete mixture model to data distribution, the proposed FV-VAE is more flexible to represent the natural property of data for better generalization. Extensive experiments are conducted on three public datasets, i.e., UCF101, ActivityNet, and CUB-200-2011 in the context of video action recognition and fine-grained image classification, respectively. Superior results are reported when compared to state-of-the-art representations. Most remarkably, our proposed FV-VAE achieves to-date the best published accuracy of 94.2% on UCF101.

1. Introduction

The recent advances in deep convolutional neural networks (CNNs) have demonstrated high capability in visual recognition. For instance, an ensemble of residual nets [7] achieves 3.57% in terms of top-5 error on the ImageNet dataset [26]. More importantly, when utilizing the activations of either a fully-connected layer or a convolutional layer in a pre-trained CNN as a universal visual representation and applying this representation to other visual recognition tasks (e.g., scene understanding and semantic segmentation), CNNs also manifest impressive performances. The improvements are expected when CNNs are further fine-tuned with only amount of task-specific data.

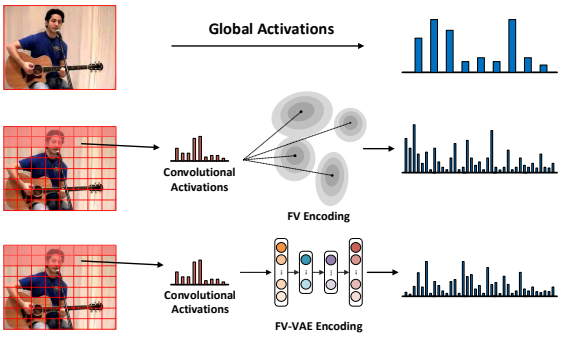


Figure 1. Visual representations derived from activations of different layers in CNNs (upper row: global activations of the fully-connected layer; middle row: convolutional activations with Fisher Vector encoding; bottom row: convolutional activations with our FV-VAE encoding).

The activations of different layers in CNNs are generally grouped into two dimensions: global activations and convolutional activations. The former directly takes the activations of the fully-connected layer as visual representations, which are holistic over the entire image, as shown in the upper row of Figure 1. The latter, in contrast, creates visual representations by encoding a set of regional and local activations from a convolutional layer to a vectorial representation using quantization strategies. For example, Fisher Vector (FV) [23] is one of the most successful quantization approaches, as shown in the middle row of Figure 1. While superior results by aggregating convolutional activations are reported in most recent studies [3, 44], convolutional activations are first extracted as local descriptors followed by another separate quantization step. Thus such descriptors may not be optimally compatible with the encoding process, making the quantization sub-optimal. Furthermore, as

*This work was performed when Zhaofan Qiu was visiting Microsoft Research as a research intern.

discussed in [13], the generative model behind FV, i.e., the Gaussian Mixture Model (GMM), cannot always represent the natural clustering of the descriptors and its inflexible Gaussian observation model limits its generalization ability.

We show in this paper that these two limitations can be mitigated by designing a deep architecture for representation learning that combines convolutional activations extraction and quantization into a one-stage learning. Specifically, we present a novel Fisher Vector encoding with Variational Auto-Encoder (FV-VAE) framework to encode convolutional activations with deep generative model (i.e., VAE), as shown in the bottom row of Figure 1. The pipeline of the proposed deep architecture generally consists of two components: a sub-network with a stack of convolution layers to produce convolutional activations followed by a VAE structure aggregating the regional convolutional descriptors to a FV. VAE consists of hierarchies of conditional stochastic variables and is a highly expressive model by optimizing a variational approximation (an inference/recognition model) to the intractable posterior for the generative distribution. Compared to traditional GMM model which has the form of a mixture of fixed Gaussian components, the inference model here can be regarded as an alternative to predict specific Gaussian components to different inputs by a single neural network, making it more flexible. It is also worth noting that a classification loss is additionally considered to preserve the semantic information in the training stage. The entire architecture is trainable in an end-to-end fashion. Furthermore, in the feature extraction stage, we theoretically prove that the FV of input descriptors can be directly computed by accumulating the gradient vector of reconstruction loss in the VAE through back-propagation.

The main contribution of this work is the proposal of FV-VAE architecture to encode convolutional descriptors with Variational Auto-Encoder. We theoretically formulate the computation of FV in VAE and substantiate an implementation of FV-VAE for visual representation learning.

2. Related Work

In the literature, visual representation generation from a pre-trained CNN model has proceeded along two dimensions: global activations and convolutional activations. The first is to extract visual representation from global activations in a CNN directly, e.g., the outputs from fully-connected layer in VGG [30] or pool5 layer in ResNet [7]. In practice, this scheme often starts by pre-training CNN model on a large dataset (e.g., ImageNet) and then fine-tuning the CNN architecture with a small amount of task-specific data to better characterize the intrinsic information in target scenario. The visual representation learnt in this direction has been exploited in a broad range of computer vision tasks including fine-grained image classification [1, 17], video action recognition [18, 19, 24, 29] and visual captioning [36, 38].

Another alternative scheme is to utilize the activations from convolutional layers in CNN as regional and local descriptors. Compared to global activations, convolutional activations from CNN are embedded with rich spatial information, making them more transferable to different domains and more robust to translation and rotation, which have shown the effectiveness in several technological advances, e.g., Spatial Pyramid Pooling (SPP) [6], Fast R-CNN [5] and Fully Convolutional Networks (FCNs) [22]. Recently, many works attempt to produce visual representation by encoding convolutional activations with different quantization strategies. For example, Fisher Vector [23] is computed on the output of the last convolutional layer of VGG networks for describing texture in [3]. Similar in spirit, Xu *et al.* utilize VLAD [11] to encode convolutional descriptors of video frame for multimedia event detection [44]. In [28] and [43], Sharma *et al.* and Xu *et al.* dynamically pool convolutional descriptors with attention models for action recognition and image captioning, respectively. Furthermore, convolutional descriptors of one convolutional layer are pooled with the guidance of the activations of the successive convolutional layer in [21]. In [20], convolutional descriptors from two CNNs are multiplied using the outer product and pooled to obtain the bilinear vector.

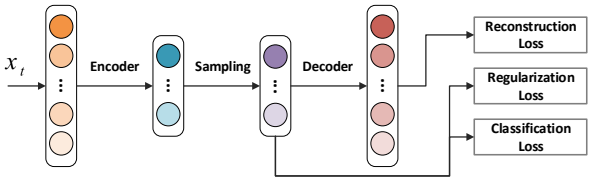
In summary, our work belongs to the second dimension and aims to compute FV on convolutional activations with deep generative models. We exploit Variational Auto-Encoder for this purpose, which optimizes an inference model to the intractable posterior. The high flexibility of the inference model and efficiency of the structure optimization makes VAE more advanced than traditional GMM. Our work in this paper contributes by studying not only encoding convolutional activations in a deep architecture, but also theoretically figuring out the computation of FV based on VAE architecture.

3. Fisher Vector Meets VAE

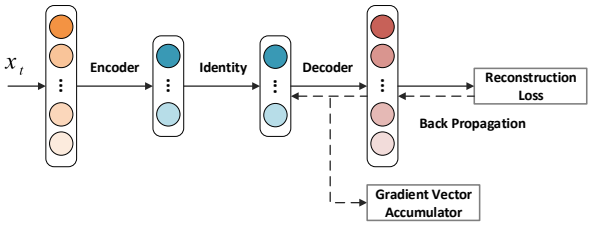
In this section, we first recall the Fisher Vector theory, followed by presenting how to estimate the probability density function in FV through VAE. The optimization of VAE is then elaborated and how to compute the FV of the input descriptors will be introduced finally.

3.1. Fisher Vector Theory

Suppose we have two sets of local descriptors $X = \{\mathbf{x}_t\}_{t=1}^{T_x}$ and $Y = \{\mathbf{y}_t\}_{t=1}^{T_y}$ with T_x and T_y descriptors, respectively. Let $\mathbf{x}_t, \mathbf{y}_t \in \mathbb{R}^d$ denote the d-dimensional features of each descriptor. In order to measure the similarity between the two sets, kernel method is employed by mapping them into a hyperspace. Specifically, assuming that the generation process of descriptors in $\mathbb{R}^d$ can be modeled by a probability density function $u_{\boldsymbol{\theta}}$ with M param-



(a) VAE Training



(b) FV Extraction

Figure 2. The overview of FV learning with VAE: (a) The training process of VAE, (b) FV extraction based on VAE.

eters $\boldsymbol{\theta} = [\theta_1, ..., \theta_M]'$, Fisher Kernel (FK) [9] between the two sets X and Y is given by

$$K(X,Y) = G_{\boldsymbol{\theta}}^{X\prime} F_{\boldsymbol{\theta}}^{-1} G_{\boldsymbol{\theta}}^{Y} \ , \tag{1}$$

where $G_{\boldsymbol{\theta}}^{X} = \nabla_{\boldsymbol{\theta}} \log u_{\boldsymbol{\theta}}(X)$ is defined as fisher score function by computing the gradient of the log-likelihood of the set based on the generative model, and $F_{\boldsymbol{\theta}} = \mathbb{E}_{X\sim u_{\boldsymbol{\theta}}}[G_{\boldsymbol{\theta}}^{X} G_{\boldsymbol{\theta}}^{X\prime}]$ is the Fisher Information Matrix (FIM) of $u_{\boldsymbol{\theta}}$ which is regarded as statistical feature normalization. Since $F_{\boldsymbol{\theta}}$ is positive semi-definite, the FK in Eq.(1) can be re-written explicitly as inner product in hyperspace:

$$K(X,Y) = \mathscr{G}_{\boldsymbol{\theta}}^{X\prime} \mathscr{G}_{\boldsymbol{\theta}}^{Y} \ , \tag{2}$$

where

$$\mathscr{G}_{\boldsymbol{\theta}}^{X} = F_{\boldsymbol{\theta}}^{-\frac{1}{2}} G_{\boldsymbol{\theta}}^{X} = F_{\boldsymbol{\theta}}^{-\frac{1}{2}} \nabla_{\boldsymbol{\theta}} \log u_{\boldsymbol{\theta}}(X) \ . \tag{3}$$

Formally, $\mathscr{G}_{\boldsymbol{\theta}}^{X}$ is well-known as Fisher Vector (FV). The dimension of FV is equal to the number of generative parameters $\boldsymbol{\theta}$, which is often much higher than that of the descriptor, making FV of higher descriptive capability.

3.2. Probability Estimation through VAE

Next, we will discuss how to estimate the probability density function $u_{\boldsymbol{\theta}}$ in FV. In general, $u_{\boldsymbol{\theta}}$ is chosen to be Gaussian Mixture Model (GMM) [27, 40] as one can approximate any distribution with arbitrary precision by GMM, in which $\boldsymbol{\theta}$ is composed of mixture weight, mean and covariance of Gaussian components. The need of a large number of mixture components and inefficient optimization of Expectation Maximization algorithm, however, makes the parameter learning computationally expensive and difficult to be applied to large-scale complex data. Inspired by the idea of deep generative models [16, 25] which enable the flexible and efficient inference learning in a neural network,

Algorithm 1 Variational Auto-Encoding (VAE) Optimization

Input: training set $X = \{\mathbf{x}_t\}_{t=1}^{T_x}$, corresponding labels $L = \{l_t\}_{t=1}^{T_x}$, loss weights $\lambda_1, \lambda_2, \lambda_3$.
Initialization: random initialized $\boldsymbol{\theta}_0, \boldsymbol{\phi}_0$.
Output: VAE parameters $\boldsymbol{\theta}^*, \boldsymbol{\phi}^*$.
repeat
  Sample $\mathbf{x}_t$ in the minibatch.
  Encoder: $\boldsymbol{\mu}_{\mathbf{z}_t} \leftarrow f_{\boldsymbol{\phi}}(\mathbf{x}_t)$.
  Sampling: $\mathbf{z}_t \leftarrow \boldsymbol{\mu}_{\mathbf{z}_t} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_{\mathbf{z}}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.
  Decoder: $\boldsymbol{\mu}_{\mathbf{x}_t} \leftarrow f_{\boldsymbol{\theta}}(\mathbf{z}_t)$.
  Compute reconstruction loss:
    $\mathcal{L}_{rec} = -\log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{z}_t) = -\log \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\sigma}_{\mathbf{x}}^2 \mathbf{I})$.
  Compute regularization loss:
    $\mathcal{L}_{reg} = \frac{1}{2} \left\|\boldsymbol{\mu}_{\mathbf{z}_t}\right\| + \frac{1}{2} \left\|\boldsymbol{\sigma}_{\mathbf{z}}\right\| - \frac{1}{2} \sum_{k=1}^{d} (1 + \log \sigma_{\mathbf{z}(k)}^2)$.
  Compute classification loss:
    $\mathcal{L}_{cls} = softmax\_loss(\mathbf{z}_t, l_t)$.
  Fuse the three loss:
    $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \lambda_1 \mathcal{L}_{rec}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda_2 \mathcal{L}_{reg}(\boldsymbol{\phi}) + \lambda_3 \mathcal{L}_{cls}(\boldsymbol{\phi})$.
  Back-propagate the gradients.
until maximum iteration reached.

we develop a Variational Auto-Encoder (VAE) to generate the probability function $u_{\boldsymbol{\theta}}$.

Following the notations in Section 3.1 and assuming that all the descriptors in the set are independent, the log-likelihood of the set can be calculated by the sum over log-likelihoods of individual descriptor and written as

$$\log u_{\boldsymbol{\theta}}(X) = \sum_{t=1}^{T_x} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) \ . \tag{4}$$

To model the probability of $\mathbf{x}_t$ generated from parameters $\boldsymbol{\theta}$, an unobserved continuous random variable $\mathbf{z}_t$ is involved with prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ and each $\mathbf{x}_t$ is generated from the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. As such, each log-likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{x}_t)$ can be measured using Kullback-Leibler divergence (D_{KL}) as

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t) &= D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_t)||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_t)) + \mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t) \\ &\geqslant \mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t) \end{aligned} \ , \tag{5}$$

where $\mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t)$ is the variational lower bound on the likelihood of descriptor $\mathbf{x}_t$ and can be written as

$$\mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_t)||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_t)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{z})], \tag{6}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a recognition model which is an approximation to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. In our proposed FV-VAE method, we use this lower bound $\mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t)$ as an approximation of the log-likelihood. Through this approximation, the generative model can be divided into two parts: encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, predicting hidden and visible probability, respectively.

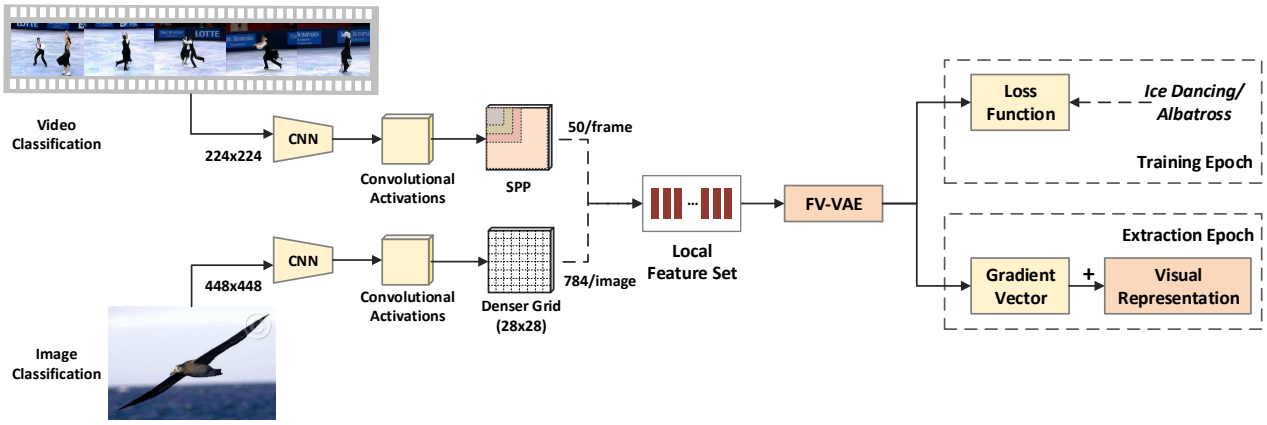


Figure 3. Visual representation learning framework for image and video recognition including our FV-VAE. Spatial Pyramid Pooling (SPP) is performed on the last pooling layer of CNN to aggregate the local descriptors of video frame, which applies four different max pooling operations and obtain (6×6), (3×3), (2×2) and (1×1) outputs for each convolutional filter, resulting a total of 50 descriptors. For image, an input with a higher resolution of (448×448) is fed into the CNN and the activations of the last convolutional layer conv_{5_4}+relu in VGG_19 are extracted, leading to dense local descriptors of 28×28. In training stage, FV-VAE architecture is learnt by minimizing the overall loss. In extraction epoch, the learnt FV-VAE is to encode the set of local descriptors into a vectorial FV representation.

3.3. Optimization of VAE

The inference model parameter $\boldsymbol{\phi}$ and generative model parameter $\boldsymbol{\theta}$ are straightforward to be optimized using stochastic gradient descend method. More specifically, let the prior distribution be the standard normal distribution $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$, and both the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and posterior approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ be multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\sigma}^2_{\mathbf{x}_t}\mathbf{I})$ and $\mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\mathbf{z}_t}, \boldsymbol{\sigma}^2_{\mathbf{z}_t}\mathbf{I})$, respectively. The one-step Monte Carlo is exploited to estimate the latent variable $\mathbf{z}_t$. Hence, the lower bound in Eq. (6) can be rewritten as

$$\begin{aligned}\mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t) \simeq \log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{z}_t) + \frac{1}{2}\sum_{k=1}^{d}(1 + \log \sigma^2_{\mathbf{z}_t(k)}) \\ - \frac{1}{2}\left\|\boldsymbol{\mu}_{\mathbf{z}_t}\right\| - \frac{1}{2}\left\|\boldsymbol{\sigma}_{\mathbf{z}_t}\right\|\end{aligned}, \quad (7)$$

where $\mathbf{z}_t$ is generated from $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}_t}, \boldsymbol{\sigma}^2_{\mathbf{z}_t}\mathbf{I})$ and it is equivalent to $\mathbf{z}_t = \boldsymbol{\mu}_{\mathbf{z}_t} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_{\mathbf{z}_t}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Figure 2(a) illustrates an overview of our VAE training process and Algorithm 1 further details the optimization steps. It is also worth noting that different from the training of standard VAE method which estimates $\boldsymbol{\sigma}_{\mathbf{x}}$ and $\boldsymbol{\sigma}_{\mathbf{z}}$ in another parallel encoder-decoder structure, we simply learn the two covariance by gradient descent technique and share them across all the descriptors, making the number of parameters learnt in VAE significantly reduced in our case. In addition to the basic reconstruction loss and regularization loss, we further take classification loss into account in our VAE training to incorporate semantic information, which has been shown effective in semi-supervised generative model learning [15]. The overall loss function is then given by

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \lambda_1 \mathcal{L}_{rec}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda_2 \mathcal{L}_{reg}(\boldsymbol{\phi}) + \lambda_3 \mathcal{L}_{cls}(\boldsymbol{\phi}) \ . \quad (8)$$

We fix $\lambda_1 = \lambda_2 = 1$ in Eq. (8) and will investigate the effect of tradeoff parameter λ_3 in our experiments. During the training, the gradients are calculated and back-propagate to the lower layers so that lower layers can adjust their parameters to minimize the loss.

3.4. FV Extraction

After the optimization of model parameters $[\boldsymbol{\theta}^*, \boldsymbol{\phi}^*]$, Figure 2(b) demonstrates how to extract Fisher Vector based on the learnt VAE architecture.

By replacing the log-likelihood with its approximation, i.e., lower bound $\mathcal{LB}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_t)$, we can obtain FV in Eq. (3):

$$\begin{aligned}\mathcal{G}^X_{\boldsymbol{\theta}^*} &= F^{-\frac{1}{2}}_{\boldsymbol{\theta}^*} \nabla_{\boldsymbol{\theta}} \log u_{\boldsymbol{\theta}^*}(X) \\ &= -F^{-\frac{1}{2}}_{\boldsymbol{\theta}^*} \sum_{t=1}^{T_x} [\nabla_{\boldsymbol{\theta}} \mathcal{L}_{rec}(\mathbf{x}_t; \boldsymbol{\theta}^*, \boldsymbol{\phi}^*)]\end{aligned}, \quad (9)$$

which is the normalized gradient vector of reconstruction loss, and can be computed directly though the back propagation operation. It is worth noticing that when extracting FV representation, we withdraw the sampling operation and use $\boldsymbol{\mu}_{\mathbf{z}_t}$ as $\mathbf{z}_t$ directly to avoid stochastic factors.

4. Visual Representation Learning

By utilizing FV-VAE as a deep architecture for quantization, a general visual representation learning framework is devised for image and video recognition, respectively, as illustrated in Figure 3. The basic idea is to construct a set of convolutional descriptors for image or video frames, followed by encoding them into a vectorial FV representation using FV-VAE architecture. Both the training epoch and FV extraction epoch are shown in Figure 3 and the entire framework is trainable in an end-to-end manner.

We exploit different strategies of aggregation to construct the set of convolutional descriptors for video frames and image, respectively, due to the different property in between. A video consists of a sequence of frames with large intra-class variations caused by, e.g., camera motion, illumination conditions and so on, making the scale of an identical object varying in different frames. Following [44], we employ Spatial Pyramid Pooling (SPP) [6] on the last pooling layer to extract scale-invariant local descriptors for video frames. Instead, we feed a higher resolution (e.g., 448×448) input into the CNN to fully utilize image information and extract the activations of the last convolutional layer (e.g., conv_{5_4}+relu in VGG_19), resulting in dense local descriptors (e.g., 28×28) for image as in [20].

In our implementation, Multi-Layer Perceptron (MLP) is employed as encoder and decoder in FV-VAE and one layer decoder is developed to reduce the dimension of FV representation. As such, the functions in Algorithm 1 can be specified as

$$\begin{aligned} &\textbf{Encoder}: \boldsymbol{\mu}_{\mathbf{z}_t} \leftarrow MLP_{\boldsymbol{\phi}}(\mathbf{x}_t) \\ &\textbf{Decoder}: \boldsymbol{\mu}_{\mathbf{x}_t} \leftarrow ReLU(W'_{\boldsymbol{\theta}}\mathbf{z}_t + \mathbf{b}_{\boldsymbol{\theta}}) \end{aligned} , \tag{10}$$

where $\{W_{\boldsymbol{\theta}}, \mathbf{b}_{\boldsymbol{\theta}}\}$ are the encoder parameters $\boldsymbol{\theta}$. The gradient vector of $\mathcal{L}_{rec}$ is calculated as

$$\begin{aligned} \nabla_{\boldsymbol{\theta}}\mathcal{L}_{rec}(\mathbf{x}_t;\boldsymbol{\theta}^*,\boldsymbol{\phi}^*) &= flatten\left\{[\frac{\partial\mathcal{L}_{rec}}{\partial W_{\boldsymbol{\phi}}}, \frac{\partial\mathcal{L}_{rec}}{\partial \mathbf{b}_{\boldsymbol{\theta}}}]\right\} \\ &= flatten\left\{[\frac{\partial\mathcal{L}_{rec}}{\partial \boldsymbol{\mu}_{\mathbf{x}_t}}\cdot\mathbf{z}'_t, \frac{\partial\mathcal{L}_{rec}}{\partial \boldsymbol{\mu}_{\mathbf{x}_t}}]\right\} \\ &= flatten\left\{\frac{\partial\mathcal{L}_{rec}}{\partial \boldsymbol{\mu}_{\mathbf{x}_t}}\cdot[\mathbf{z}'_t, 1]\right\} \\ &= flatten\left\{\frac{\boldsymbol{\mu}_{\mathbf{x}_t}-\mathbf{x}_t}{\boldsymbol{\sigma}^2_{\mathbf{x}}}\odot(\boldsymbol{\mu}_{\mathbf{x}_t}>0)\cdot[\mathbf{z}'_t, 1]\right\} \end{aligned} , \tag{11}$$

where "$flatten$" represents to flatten a matrix to a vector, and $\odot$ denotes element-wise multiplication to filter the activated elements. Considering it is difficult to obtain an analytic solution of FIM in this case, we make an approximation by replacing the expectation with the average on the whole training set:

$$F_{\boldsymbol{\theta}^*} = \mathbb{E}_{X\sim u_{\boldsymbol{\theta}}}[G^X_{\boldsymbol{\theta}} G^{X\prime}_{\boldsymbol{\theta}}] \approx \underset{X}{mean}[G^X_{\boldsymbol{\theta}} G^{X\prime}_{\boldsymbol{\theta}}] , \tag{12}$$

and

$$\mathscr{G}^X_{\boldsymbol{\theta}^*} = flatten\left\{-F^{-\frac{1}{2}}_{\boldsymbol{\theta}^*}\cdot\sum_{t=1}^{T_x}(\frac{\boldsymbol{\mu}_{\mathbf{x}_t}-\mathbf{x}_t}{\boldsymbol{\sigma}^2_{\mathbf{x}}}\odot(\boldsymbol{\mu}_{\mathbf{x}_t}>0)\cdot[\mathbf{z}'_t, 1])\right\} , \tag{13}$$

which is the output FV representation in our framework.

To improve the convergence speed and better regularize the visual representation learning for video, we train this framework by inputting one single video frame rather than multiple ones, which is randomly sampled from videos. In the FV extraction stage, the video-level representation can be easily obtained by averaging FVs of all the frames sampled from the video since FV in Eq. (13) is linear additive.

Table 1. Methodology comparison of different quantization.

Quantization	indicator	descriptor
FV [23]	Gaussian observation model	gradient with respect to GMM parameters
VLAD [11]	clustering center	difference to the assigned center
BP [20]	local feature	coordinate representation
FV-VAE	VAE hidden variable	gradient of reconstruction loss

5. Experiments

We evaluate the learnt visual representation by FV-VAE architecture on three popular datasets, i.e., UCF101 [31], ActivityNet [2] and CUB-200-2011 [39]. The UCF101 dataset is one of the most popular video action recognition benchmarks. It consists of 13,320 videos from 101 action categories. The action categories are divided into five groups: human-object interaction, body-motion only, human-human interaction, playing musical instruments and sports. Three training/test splits are provided by the dataset organisers and each split in UCF101 includes about 9.5K training and 3.7K test videos. The ActivityNet dataset is a large-scale video benchmark for human activity understanding. The latest released version of the dataset (v1.3) is exploited, which contains 19,994 videos from 200 activity categories. The 19,994 videos are divided into 10,024, 4,926, 5,044 videos for training, validation and test set, respectively. Note that the labels of test set are not publicly available and the performances on ActivityNet dataset are all reported on validation set. Furthermore, we also validate the representation on CUB-200-2011 dataset, which is widely adopted for fine-grained image classification and consists of 11,788 images from 200 bird species. We follow the official split on this dataset with 5,994 training and 5,794 test images.

5.1. Compared Approaches

To empirically verify the merit of visual representation learnt by FV-VAE, we compare the following quantization methods: **Global Activations (GA)** directly utilizes the outputs of fully-connected/pooling layer as visual representation. **Fisher Vector (FV)** [23] produces the visual representation by concatenating the gradients with respect to the parameters of GMM, which is trained on local descriptors. **Vector of Locally Aggregated Descriptors (VLAD)** [11] is to accumulate, for each clustering center learnt with K-means, the differences between the clustering center and the descriptors assigned to it, and then concatenates the accumulated vector of each center as quantized representation.

Bilinear Pooling (BP) [20] pools local descriptors in a pairwise manner by outer product. In our case, one local descriptor pairs with itself. To better illustrate the difference between the compared approaches, we details the methodology in Table 1. In particular, we decouple the quantization process into two parts: indicator and descriptor. Indicator refers to observations/distributions estimated on the whole set of local descriptors and descriptor is to represent the set with respect to the indicator.

5.2. Experimental Settings

Convolutional activations. On video action recognition task, we extract two widely adopted convolutional activations, i.e., activations of pool5 layer in VGG_19 [30] and res5c layer in ResNet_152 [7]. Given a 224×224 video frame as input, the outputs of the two layers are both 7×7 and the dimension of each activation is 512 and 2,048, respectively. For each video, 25 frames are uniformly sampled for representation extraction. On image classification problem, we feed 448×448 image into VGG_19 and the activations of conv_{5_4}+relu layer are exploited, which produce 28×28 convolutional descriptors.

VAE optimization. To make the training process of VAE stable, we first exploit L2 normalization on each convolutional activation to make the input to VAE in a common scale. Following [8, 37], dropout is then employed to randomly drop out units input to the encoder but the auto-encoder is optimized to reconstruct a complete "repaired" input. The dropout rate is fixed to 0.5. Furthermore, we utilize AdaDelta [46] optimization method implemented in Caffe [12] to normalize the gradient of each parameters for balancing their converge speed. The base learning rate is set to 1 and the size of mini-batch is 128 images/frames. The optimization will be complete after 5,000 batches.

Quantization settings. For our FV-VAE, given the local descriptor with dimension C ($C \in \{512, 2048\}$), we design a two-layer encoder ($C \rightarrow C \rightarrow 255$) to reduce the dimension to 255, coupled with a single layer decoder ($255 \rightarrow C$). The dimension of the final quantized representation is $256 \times C$. For FV and VLAD, we follow the settings in [3] and [44]. Specifically, 128 Gaussian components for FV and 256 clustering centers for VLAD are exploited. As such, the dimension of representations encoded by FV and VLAD will also be $256 \times C$. The two quantization approaches are implemented by VLFeat [35].

Classifier training. After representation learning by all the methods in our experiments, we apply signed square-root step ($sign(x)\sqrt{|x|}$) and L2 normalization ($x/\|x\|_2$) as in [3, 20, 23, 44], and then train a one-vs-all linear SVM with a fixed hyperparameter $C_{svm} = 100$.

Table 2. Performance comparisons of different quantization methods on UCF101 split1 with default VGG_19 network.

Feature	Dimension	Accuracy
GA	4096	74.91%
Concatenation	25088	75.89%
AVE	512	73.25%
FV	131072	78.85%
VLAD	131072	80.67%
BP	262144	81.39%
FV-VAE$^-$	131072	81.91%
FV-VAE	131072	83.45%

5.3. Performance

Comparison with different quantization methods. We first examine our FV-VAE and compare with other quantization methods. In addition to the four mentioned quantization methods, we also include three runs: Concatenation, AVE and FV-VAE$^-$. Concatenation is to flatten the activations of pool5 layer and concatenate into a super vector, whose dimension is 25088 ($7 \times 7 \times 512$). The representation in AVE is produced by average fusing the 49 512-dimensional convolutional activations in pool5 layer. A slightly different setting of our FV-VAE is named as FV-VAE$^-$, in which the classification loss in Eq.(8) is excluded or λ_3 is set to 0.

The performances and comparisons with default VGG_19 network on UCF101 (split 1) are summarized in Table 2. Overall, the results indicate that our FV-VAE leads to a performance boost against others. In particular, the accuracy of FV-VAE can achieve 83.45%, which makes the relative improvement over the best competitor BP by 2.5%. Meanwhile, the dimension of representation learnt by FV-VAE is only half of that of BP. There is a performance gap among three runs GA, Concatenation and AVE. Though three runs all directly originate from pool5 layer, they are fundamentally different in the way of generating frame representation. The representation of GA is as a result of flatting all kernel maps in pool5 to the neurons in a fully-connected layer, while Concatenation and AVE is by directly concatenating convolutional descriptors or average fusing them in pool5 layer. As indicated by our results, Concatenation can lead to better performance than GA and AVE. VLAD outperforms FV on UCF101, but the performance is still lower than BP. Compared to FV which produces representation with respect to a number of Gaussian mixture components, FV-VAE will learn which Gaussian distribution is needed for the input specific descriptor by an inference neural network, making FV-VAE more flexible. Therefore, FV-VAE performs significantly better than FV. More importantly, FV-VAE is trainable in an end-to-end fashion. By additionally incorporating semantic information, FV-VAE leads to apparent improvement against FV-VAE$^-$. Furthermore, by reducing the dimension of latent variable to 7, the visual representations produced by FV-

Table 3. Performance comparisons of FV-VAE with local activations from different networks on UCF101 split1.

Network	GA	FV-VAE$^-$	FV-VAE
pool5	74.91%	81.91%	83.45%
pool5 fine-tuned	79.06%	82.05%	82.13%
res5c	81.57%	85.05%	86.33%

Table 4. Performance comparisons with the state-of-the-art methods on UCF101 (3 splits, $\times$10 augmentation). C3D: Convolutional 3D [33]; TSN: Temporal Segment Networks; TDD: Trajectory-pooled Deep-convolutional Descriptor [41]; IDT: Improved Dense Trajectory [40].

Method	Accuracy
Two-stream ConvNet [29]	88.1%
C3D (3 nets) [33]	85.2%
Factorized ST-ConvNet [32]	88.1%
Two-stream + LSTM [45]	88.6%
Two-stream fusion [4]	92.5%
Long-term temporal ConvNet [34]	91.7%
Key-volume mining CNN [49]	93.1%
TSN (3 modalities) [42]	94.2%
IDT [40]	85.9%
C3D + IDT [33]	90.4%
TDD + IDT [41]	91.5%
Long-term temporal ConvNet + IDT [34]	92.7%
FV-VAE-pool5	83.9%
FV-VAE-pool5 optical flow	89.5%
FV-VAE-res5c	86.6%
FV-VAE-(pool5 + pool5 optical flow)	93.7%
FV-VAE-(res5c + pool5 optical flow)	94.2%
FV-VAE-(res5c + pool5 optical flow) + IDT	95.2%

VAE and GA are then with the same dimension of 4,096. In this case, the accuracy of FV-VAE can still achieve 78.37% which is higher than 74.91% by GA, again demonstrating the effectiveness of our FV-VAE. In addition, similar performance trends are observed at CUB-200-2011 dataset, as shown in the upper rows of Table 6, in two protocols of where the object bounding boxes are provided or not.

Comparison with different networks. Next, we turn to measure the performances on UCF101 split1 of our FV-VAE with local activations from different networks, including pool5 layer in VGG_19 and fine-tuned VGG_19 using video frames respectively, and res5c layer in ResNet_152. As shown in Table 3, compared to pool5 in VGG_19, FV-VAE on the outputs of res5c layer in ResNet_152 with a deeper CNN exhibits better performance. An interesting observation is that GA and FV-VAE$^-$ perform better on the outputs of pool5 layer in fine-tuned VGG_19 than that in VGG_19, while reverse trend is indicated by using FV-VAE. We speculate that this may be the result of overfitting in fine-tuning with UCF101, which in particular affects the descriptive ability of convolutional layers. This result also indicates the advantage of exploring semantic information in FV-VAE training based on the outputs of a general network than a fine-tuned one.

Table 5. Performance comparisons in terms of Top-1&Top-3 classification accuracy, and mean AP on ActivityNet validation set.

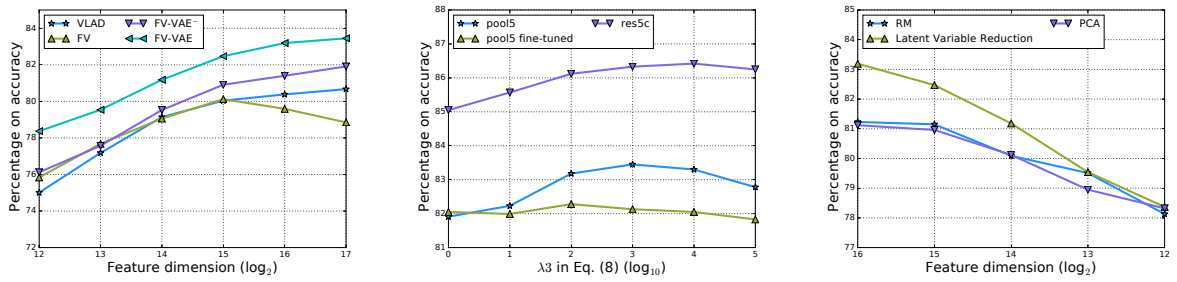
Methods	Top-1	Top-3	MAP
VGG_19-GA [30]	66.59%	82.70%	70.22%
ResNet_152-GA [7]	71.43%	86.45%	76.56%
C3D-GA [33]	65.80%	81.16%	67.68%
IDT [40]	64.70%	77.98%	68.69%
FV-VAE-pool5	72.51%	85.68%	77.25%
FV-VAE-res5c	78.55%	91.16%	84.09%

Table 6. Performance comparisons on CUB-200-2011 in two scenarios: where the object bounding boxes are provided at training and test time or not. ft: fine-tuning.

Methods	Dim	w/o ft	w/ ft	+box w/o ft	+box w/ ft
GA	4k	61.0%	70.4%	65.3%	76.4%
FV	128k	70.8%	74.0%	73.6%	77.1%
VLAD	128k	73.5%	76.5%	75.1%	79.8%
BP	256k	75.2%	78.0%	76.9%	80.8%
FV-VAE	128k	79.3%	82.4%	79.5%	83.6%
Previous works		84.5%[48] 84.1%[10] 75.7%[1]	84.1%[20] 82.0%[17] 73.9%[47]	85.1%[20] 76.4%[47]	82.8%[17] 73.0%[3]

Comparison with the state-of-the-art. We compare with several state-of-the-art techniques on three splits of UCF101, ActivityNet validation set and CUB-200-2011. The performance comparisons are summarized in Table 4, 5 and 6, respectively. It is worth noting that most recent works on UCF101 employ and fuse two or multiple modalities. For fair comparison, two basic and widely adopted modalities, i.e., video frame and optical flow "image," are considered as inputs to our visual representation framework and late fusion is used to combine classifier scores on the two modalities. As shown in Table 4, FV-VAE on activations from pool5 layer in VGG_19 with image and optical flow inputs can achieve 93.7%, which makes the relative improvement over two-stream networks [29], [45] and [4] by 6.3%, 5.7% and 1.3%, respectively. When exploiting the outputs of res5c layer in ResNet_152 on image inputs as instead, the accuracy will be further improved to 94.2%. By combining with IDT which are hand-crafted features, our final performance will boost up to 95.2%, which is to-date the best published performance on UCF101.

The results across different evaluation metrics consistently indicate that visual representation produced by our FV-VAE leads to a performance boost against baselines on ActivityNet validation set, as shown in Table 5. More specifically, FV-VAE on the outputs of pool5 in VGG_19 and res5c in ResNet_152 outperforms GA from VGG_19 and ResNet_152 relatively by 10.0% and 9.8% in terms of mAP, respectively. Furthermore, the representation learnt by FV-VAE only on visual appearance of video frame also exhibits better performance than GA representation from C3D and IDT motion features which additionally explore temporal information in videos.



(a) (b) (c)

Figure 4. Experimental analysis: (a) The accuracy of visual representation with different dimensions learnt by different quantization methods. (b) The accuracy curve of FV-VAE on activations from different networks with different λ_3 in Eq.(8). (c) The accuracy of different feature compression methods on representation learnt by FV-VAE. Note that all the performances reported in this figure are on UCF101 split1 and similar performance trends are observed at the other two datasets.

Fine-tuning VGG_19 on CUB-200-2011 for FV-VAE generally performs better than original VGG_19 on both protocols of where the object bounding boxes are given or not, as shown in Table 6. Overall, the representation learnt by FV-VAE leads to a performance boost against some baselines, e.g., [17] which extracts representation on local regions learnt by co-segmentation and [1] which combines the representations from three networks fed by warped bird head, warped body and entire image, respectively. It is not surprise that FV-VAE yields inferior performance to the other baselines, as the representation learnt by our FV-VAE is for general purpose while contributions of different regions in particular for fine-grained classification are taken into account in these methods. For instance, a saliency weight is learnt and assigned to each local region in [48], and a spatial transformer is trained to reduce the effect of translation and rotation as preprocess in [10]. More importantly, the importance estimation of each local region can be easily integrated into our framework as spatial attention.

5.4. Analysis

The effect of representation dimension. Figure 4(a) compares the accuracy of learnt representations with different dimensions by changing the number of latent variable in FV-VAE, the number of centroids in VLAD and mixture components in FV. Overall, visual representation learnt by FV-VAE consistently outperforms others at each dimension from 2^{12} to 2^{17}. In general, higher dimensional representations provide better accuracy, except that the accuracy of representation learnt by FV will decrease when the dimension is higher than 2^{15}, which may caused by overfitting. The result basically indicates the advantage of predicting Gaussian parameters by a neural network in our FV-VAE.

The effect of tradeoff parameter λ_3. A common problem with combination of multiple loss is the need to set the tradeoff parameters in between. Figure 4(b) shows the accuracy of FV-VAE with respect to different λ_3 in Eq.(8), which reflects the contribution of leveraging semantic information. As expected, the accuracy curves are all like the "$\wedge$" shapes when λ_3 varies from 10^0 to 10^5.

Feature compression. Figure 4(c) compares the performance obtained by applying different representation compression methods: (1) Random Maclaurin (RM) [14], (2) PCA dimension reduction and (3) reducing the number of latent variable in VAE. Compared to RM and PCA which separately learn a transformation for feature compression, we can reduce the dimension of the learnt FV in VAE framework by decreasing the number of latent variable. As indicated by our results, reducing the number of latent variable always achieves the best accuracy, which again confirms the high flexibility of VAE.

6. Conclusion

We have presented Fisher Vector with Variational Auto-Encoder (FV-VAE) architecture which aims to quantize the convolutional activations in a deep generative model. Particularly, we theoretically formulate the computation of FV in VAE architecture. To verify our claim, a general visual representation learning framework is devised by integrating our FV-VAE architecture and an implementation of FV-VAE is also substantiated for image and video recognition. Experiments conducted on on three public datasets, i.e., UCF101, ActivityNet, and CUB-200-2011 in the context of video action recognition and fine-grained image classification validate our proposal and analysis. Performance improvements are clearly observed when comparing to other quantization techniques.

Our future works are as follows. First, a deeper auto-encoder architecture will be explored in our FV-VAE architecture. Second, attention mechanism will be explicitly incorporated into our FV-VAE for further enhancing visual recognition. Third, Generative Adversarial Networks (GAN) will be investigated to better learn a generative model and integrated into representation learning.

References

[1] S. Branson, G. Van Horn, S. Belongie, and P. Perona. Bird species categorization using pose normalized deep convolutional nets. In *BMVC*, 2014.

[2] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[3] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *IJCV*, 118(1):65–94, 2016.

[4] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[5] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[8] D. J. Im, S. Ahn, R. Memisevic, and Y. Bengio. Denoising criterion for variational framework. In *AAAI*, 2017.

[9] T. S. Jaakkola, D. Haussler, et al. Exploiting generative models in discriminative classifiers. In *NIPS*, 1998.

[10] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[11] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[13] M. J. Johnson, D. Duvenaud, A. B. Wiltschko, S. R. Datta, and R. P. Adams. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2016.

[14] P. Kar and H. Karnick. Random feature maps for dot product kernels. In *AISTATS*, 2012.

[15] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2013.

[17] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *CVPR*, 2015.

[18] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo. Action recognition by learning deep multi-granular spatio-temporal video representation. In *ICMR*, 2016.

[19] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo. Learning hierarchical video representation for action recognition. *IJMIR*, pages 1–14, 2017.

[20] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[21] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *CVPR*, 2015.

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[23] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[24] Z. Qiu, Q. Li, T. Yao, T. Mei, and Y. Rui. MRA Asia MSM at Thumos Challenge 2015. In *CVPR workshop*, 2015.

[25] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[27] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[28] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *ICLR Workshop*, 2016.

[29] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[31] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01*, 2012.

[32] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.

[33] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[34] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*, 2016.

[35] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[36] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *ICCV*, 2015.

[37] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[38] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

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

[40] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[41] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[42] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*, 2016.

[43] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[44] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015.

[45] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[46] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[47] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014.

[48] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian. Picking deep filter responses for fine-grained image recognition. In *CVPR*, pages 1134–1142, 2016.

[49] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *CVPR*, 2016.