Development of New Models for Vision-Based Human Activity Recognition

Adel Saleh Ali Alraimi

UNIVERSITAT ROVIRA i VIRGILI

Development of New Models for

Vision-Based Human Activity Recognition

Adel Saleh Ali Alraimi

DOCTORAL THESIS
2019

UNIVERSITAT ROVIRA I VIRGILI
Development of New Models for Vision-Based Human Activity Recognition
Adel Saleh Ali Alraimi

Development of New Models for Vision-Based Human Activity Recognition

DOCTORAL THESIS

Author:
Adel Saleh Alraimi

Advisors:
Prof. Dr. Domènec Savi Puig Valls
Dr. Miguel Angel Garcia Garcia

Departament d'Enginyeria Informàtica i Matemàtiques

UNIVERSITAT ROVIRA I VIRGILI
Tarragona
2019

UNIVERSITAT ROVIRA I VIRGILI
Development of New Models for Vision-Based Human Activity Recognition
Adel Saleh Ali Alraimi

UNIVERSITAT ROVIRA I VIRGILI
Departament d'Enginyeria Informàtica i Matemàtiques

Av. Paisos Catalans, 27
43007 Tarragona
Tel. +34 977 55 95 95
Fax. +34 977 55 95 97

We STATE that the present study, entitled "Development of New Models for Vision-Based Human Activity Recognition", presented by Adel Saleh Alraimi for the award of the degree of Doctor, has been carried out under our supervision at the Departament d'Enginyeria Informàtica i Matemàtiques.

Tarragona, January 2019

Doctoral Thesis Supervisors,

Dr. Domènec Savi Puig Valls

Dr. Miguel Angel Garcia Garcia

UNIVERSITAT ROVIRA I VIRGILI
Development of New Models for Vision-Based Human Activity Recognition
Adel Saleh Ali Alraimi

To my sons Abdullah and Emran, my wife Mona, my parents and brothers and my best friend Mohamed Abdel-Nasser

UNIVERSITAT ROVIRA I VIRGILI
Development of New Models for Vision-Based Human Activity Recognition
Adel Saleh Ali Alraimi

Abstract

Action recognition methods enable intelligent machines to recognize human actions in daily life videos. However, many action recognition methods give noticeable misclassification rates due to the big variations within the videos of the same class, and the changes in viewpoint, scale and background. To reduce the misclassification rate, we propose a new video representation method that captures temporal evolution of the action happening in the whole video. Our experimental results using two state-of-the-art human activity recognition datasets (HMDB51 and Hollywood) demonstrate that our method is on par with the state-of-the-art methods.

In addition, we propose a new visual embedding method for detecting human actions in images. We build a bridge between images and text documents, which is applicable to images in total analogy with textual word embeddings. In an end-to-end fashion, visual embedding is applied similarly to natural language processing. In this method, we learn visual dictionaries based on filters of convolution layers of a convolutional neural network, which is used to capture the visual context of images. We employ visual embedding to convert words to real vectors. In addition, we assess different designs of dictionary building methods.

Indeed, a huge number of applications require efficient segmentation performance, such as activity recognition, navigation, and human body parsing. One of the important applications is gesture recognition, that is, the ability to understand human hand gestures by detecting and counting finger parts in a video stream or in still images. Thus, accurate finger parts segmentation yields more accurate gesture recognition. Consequently, in this thesis, we highlight two contributions: 1) we propose a data-driven deep learning pooling policy based on multi-scale feature map extraction at different scales (called FinSeg). A novel aggregation layer is introduced in this model, in which the feature maps generated at each scale are weighted using a fully connected layer. 2) With the lack of realistic labeled finger parts datasets, we propose a labeled dataset for finger parts segmentation (FingerParts dataset). To the best of our knowledge, the proposed dataset is the first attempt to build a

realistic dataset for finger parts semantic segmentation. Experimental results show that the proposed model outperforms the standard FCN network.

Keywords: Human Activity, Deep Learning, Representation Learning, Hand-Crafted Features, Semantic Segmentation.

Acknowledgements

The author was supported by a PhD grant from URV in 2015 (Martí Franquès program).

I would like to express my gratitude to my supervisors Prof. Dr. Domenec Puig and Dr. Miguel Angel Garcia for their useful guidance, insightful comments, and considerable encouragement to complete this thesis. I also want to thank my lab mates for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main goal of the action recognition field is to give computers the ability to recognize human actions in real life videos. The performance of applications, such as surveillance systems (Ben Aoun et al., 2011) and human-computer interaction (Bouchrika et al., 2014), mainly depend on the accuracy of human activity recognition systems.

Human action recognition is still an open challenging problem in the computer vision community. Several methods have been proposed to improve the performance of human action recognition in uncontrolled videos. Bag of words (BOW) based on a set of low level features, such as histograms of optical flow (HOF), histograms of oriented gradients (HOG) and motion boundary histograms (MBH) have become very common video representations for action recognition (Laptev et al., 2008). These models are insensitive to the position and orientation of the objects in the image. In addition, they have fixed length vectors irrespectively of the number of objects and number of frames in each video. Moreover, they have a poor localization of the objects and actions in the videos. The use of local information can be very useful to improve the recognition rate (Peng et al., 2014).

Constructing robust models to recognize free-form activities of the same class is still an open problem (Wang et al., 2016). One of the sources of misclassification is the wide margin of variation inside the same class. Other sources can be the

change in viewpoint, scale and background clutter. In addition, there are other high level unmeasurable factors that can play a role in human action recognition, such as human-objects interaction, human poses and scene context.

Indeed, most of the development in the action recognition area over the last years has been related to three approaches:

- The first approach is the definition of local features, such as spatio-temporal features (Laptev, 2005) and densely sampled local visual features (Laptev et al., 2008). We can also mention interest points, motion-based gradient descriptors (Jain et al., 2013) and dense trajectories (Wang et al., 2013a).
- The second approach is concentrated on the exploration of encoding methods, which has a long history of success in the object recognition field. A good example is Fisher vectors (Wang and Schmid, 2013).
- The third approach is related to the exploration of using of deep learning models.

1.1 Thesis objectives

The main objectives of this thesis are:

1. To address the analysis of human activities and develop suitable machine learning methods. We consider models usually applied in this research area to analyze streams of video data.
2. Unfortunately, the lack of widely good data sets for activity recognition limits the possibility of comparing different machine learning models in this field. On the other hand, such a comparison is very important since it would help researchers to choose the most appropriate models in the different applications of human activity recognition. To that end, we extend the analysis of the existing models with different data sources, and we exploit the knowledge resulting from other fields, such as semantic segmentation.
3. To use deep learning models to perform the body part segmentation task in

order to obtain a completely abstracted recognition system.

1.2 Contributions

The main contributions of this thesis are the following:

1. We analyze the performance of several hand crafted features to recognize human activities in videos. We present a video representation that exploits the properties of the trajectories of local descriptors in human action videos. In addition, we propose a new video representation method that captures temporal evolution of the action happening in the whole video. We show that combining the descriptors of improved dense trajectories with a multiple kernel learning technique can reduce the misclassification rate, and also aggregating the coherent frames in each video may have a different impact on the recognition results.

   The results of the previous studies have been published in the following journal:

   - **Adel Saleh**, Mohamed Abdel-Nasser, Miguel Angel Garcia, Domenec Puig, "Aggregating the temporal coherent descriptors in videos using multiple learning kernel for action recognition," *Pattern Recognition Letters*, 105: 4-12 (2018). Impact Factor: 1.952 (**Q2**).

   Preliminary work on improving the performance of human activity recognition methods that use hand-crafted features has been presented in the following conference papers:

   - **Adel Saleh**, Miguel Angel Garcia, Farhan Akram, Mohamed Abdel-Nasser, Domenec Puig, "Exploiting the Kinematic of the Trajectories of the Local Descriptors to Improve Human Action Recognition," Proceedings of the 10th International Conference on Computer Vision Theory and Applications (VISAPP2016), Vol. 1, pp. , 2016. **Core B.**
   - **Adel Saleh**, Mohamed Abdel-Nasser, Farhan Akram, Miguel Angel

Garcia, Domenec Puig, "Analysis of Temporal Coherence in Videos for Action Recognition," *International Conference on Image Analysis and Recognition*, pp. 325-332, 2016. **Core C.**

- **Adel Saleh**, Mohamed Abdel-Nasser, Miguel Angel Garcia, Domenec Puig, "The impact of coherence analysis and subsequences aggregation on representation learning for human activity recognition," *Artificial Intelligence Research and Development: Recent Advances and Applications*, vol. 288, pp. 64-70, IOS press, 2016.

2. We propose a new visual embedding method for image classification. It goes further in the analogy with textual data and allows us to read visual sentences in a certain order as in the case of text. The proposed method considers the spatial relations between visual words. It uses a very popular text analysis method called word2vec. In this method, we learn visual dictionaries based on filters of convolution layers of a convolutional neural network, which is used to capture the visual context of images. We employ visual embedding to convert words to real vectors.

   The results of the previous study have been presented at the following international conference:

   - **Adel Saleh**, Mohamed Abdel-Nasser, Md Mostafa Kamal Sarker, Vivek Kumar Singh, Saddam Abdulwahab, Nasibeh Saffari, Miguel Angel Garcia, and Domenec Puig, "Deep Visual Embedding for Image Classification," *International Conference on Innovative Trends in Computer Engineering (ITCE2018)*, p. 31-35. IEEE, 2018.

3. We propose a labeled dataset for finger parts segmentation. To the best of our knowledge, the proposed dataset is the first attempt to build a realistic dataset for finger parts semantic segmentation. We also propose a data-driven deep learning pooling policy based on multi-scale feature map extraction at different scales. A novel aggregation layer is introduced in this model.

The results of the previous study have been submitted to the following international conference:

- **Adel Saleh**, Hatem A. Rashwan, Mohamed Abdel-Nasser, Vivek Kumar Singh, Md Mostafa Kamal Sarker, Saddam Abdulwahab, Miguel Angel Garcia and Domenec Puig, "FinSeg: Finger Parts Semantic Segmentation Using Multi-Scale Feature Maps Aggregation of FCN," *International Conference on Computer Vision Theory and Applications (VISAPP2019).* **Core B.**

1.3 Thesis organization

The thesis is divided into the following six chapters:

4. Chapter 1: *Introduction*

   This chapter introduces the problem of human activity recognition. It starts with the motivation behind the thesis and the contributions related to improving human activity recognition methods.

5. Chapter 2: *Exploiting the Kinematics of the Trajectories of Local Descriptors to Improve Human Action Recognition*

   In this chapter we present a video representation that exploits the properties of the trajectories of local descriptors in human action videos. We use spatio-temporal information, which is led by trajectories to extract kinematic properties: tangent vector, normal vector, bi-normal vector and curvature.

6. Chapter 3: *Aggregating the temporal coherent descriptors in videos using multiple learning kernel for action recognition*

   In this chapter, we propose a new video representation method that captures temporal evolution of the action happening in the whole video.

7. Chapter 4: *Deep Visual Embedding for Image Classification*

   In this chapter, we build a bridge between images and text documents, which is applicable to images in total analogy with textual word embeddings. In

particular, we propose a new visual embedding method for image classification. The proposed method considers the spatial relations between visual words. In this method, we learn visual dictionaries based on filters of convolutional layers of CNNs, which are used to capture the visual context of images.

8. Chapter 5: *Finger Parts Semantic Segmentation Using Multi-Scale Feature Map Aggregation of FCN*

   In this chapter, we propose a labeled dataset for finger parts segmentation. In addition, we propose a data-driven deep learning pooling policy based on multi-scale feature map extraction at different scales.

9. Chapter 6: *Concluding remarks*

   This chapter presents the conclusions of the thesis and some lines of future research.

Chapter 2

Exploiting the Kinematics of the Trajectories of Local Descriptors to Improve Human Action Recognition

2.1 Introduction

Several works showed that the performance of human recognition methods can be significantly improved while using the trajectory of spatio-temporal interest points (Wang et al., 2009). In (Wang et al., 2011), trajectories were used as features to build a codebook of visual words. They proposed a robust method to get information of trajectory shape by tracking densely sampled points using optical flow fields. In (Wang et al., 2013b), Fisher coding is compared with other encoding methods. It provided better results for action recognition. In (Jain et al., 2013), an improved method using motion stabilization and person trajectories is demonstrated. In (Raptis and Soatto, 2010), parts from different grouping trajectories were embedded into a graphical model. In (Sekma et al., 2013), the

Delaunay triangulation method is applied to the trajectories of each video to get geometric relationships of objects. A graph is built for trajectories and then encoded (this method is also known as bag-of-graphs).

Many works were dedicated to describe the shape of the area around the trajectory, local motion and appearance pattern. The common methods in this line are: histograms of optical flow (HOF) (Laptev et al., 2008), motion boundary histograms (MBH) (Dalal et al., 2006) and histograms of oriented gradients (HOG) (Dalal and Triggs, 2005). After extracting the features and encoding them, a codebook is built using a clustering algorithm, such as k-means. In (Wang and Schmid, 2013), the authors showed that camera motion compensation and removal of the inconsistent matches generated by human motion can greatly improve the performance of dense trajectories. They used the Fisher vector (Sánchez et al., 2013) to generate a representation for each video. However, the previous approaches only consider simple trajectory information.

In this chapter, we used the kinematic features of the trajectories of local descriptors to improve the performance of the current super-vector based activity recognition methods. For each local descriptor, a trajectory is defined. Then, a set of kinematic features are calculated, such as tangent vector, normal vector, bi-normal vector and curvature. The steps of the proposed method are shown in Figure 2.1. The proposed work is inspired by (Wang et al., 2015), in which they extracted *Frenet-Serret* frames (see Section 2.3.2 for its definition) from the trajectories. Then, histograms of tangent vectors and normal, bi-normal vectors were used to overcome the dependency on the trajectory length. After that, they clustered videos using histograms. In their work, they did not discuss the performance of these features for activity recognition. In (Jain et al., 2013), the authors used kinematic features of the flow field to capture additional information about motion patterns. The proposed method is different from the technique discussed in (Wang et al., 2015), since kinematic features of the trajectory (tangent vector, binormal vector an curvature) are applied rather

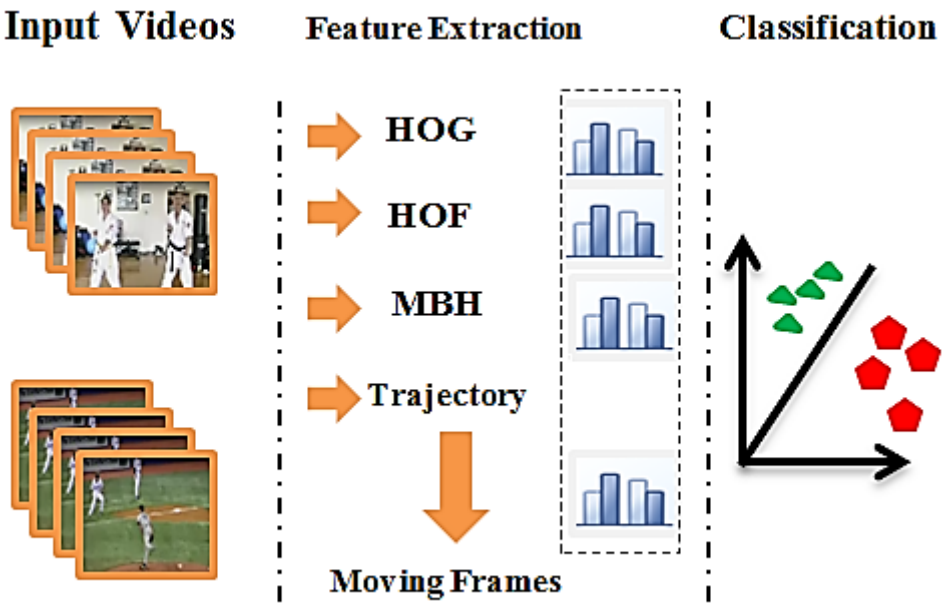

Figure 2.1: The proposed approach.

than histograms. The proposed method exploits the improved trajectories of the same length and then combines them with low level features, such as HOF, HOG and MBH. Compared to the proposed method, the CNN based method proposed in (Simonyan and Zisserman, 2014) is better in terms of accuracy but it has high time complexity because it needs a large number of training samples with supervised labels.

2.2 Methods

The main idea of the proposed approach is that we are getting complementary information like motion, acceleration and curvature, which provide a useful description of the trajectory. The improved results lead to the conclusion that statistically modeling the trajectories of low-level features enhances the recognition performance of the concepts in videos.

2.3 Hand-crafted features

Below, we explain the feature extraction methods used in this thesis: histogram of oriented gradients (HOG), histogram of optical flow (HOF), and motion boundary histogram (MBH). We also explain the steps that involve the computation of Fisher vectors (FVs).

- **HOG and HOF descriptors:** HOG captures information about appearance, whereas HOF captures local motion information. Wang et al. (2009) reported that the HOG and HOF descriptors yield good results on different activity recognition datasets compared to classical descriptors. To compute the HOG descriptor, we compute gradient magnitude responses in the horizontal and vertical directions. To compute the HOF descriptors, optical flow displacement vectors in the horizontal and vertical directions are determined (Wang and Schmid, 2013). As a result, a 2D vector field per frame is calculated. The orientations are quantized into eight bins for HOG and nine bins for HOF, as described in (Laptev et al., 2008). The length of HOG is 96 ($2 \times 2 \times 3 \times 8$), while the length of HOF was 108 ($2 \times 2 \times 3 \times 9$).
- **MBH descriptor:** MBH is a very popular descriptor for video classification and detection tasks. Dalal et al. (2006) showed the robustness of the MBH descriptor against camera and background motion. The simple idea behind MBH is computing oriented gradients over the vertical and horizontal optical flow displacements. Indeed, the horizontal and vertical displacements of the optical flow are treated as gray-level images of the motion displacements. For each optical flow component, a histogram of oriented gradients is computed using the same configuration used for still images. Indeed, information related to motion changes in the object boundaries is obtained using the flow difference, whereas information about still objects and camera motion is discarded.
- **Fisher vectors (FVs):** In computer vision, FVs have widely been used as

an image representation by pooling local image features. It is also used as a global descriptor for image classification. FV can be used for frame level pooling, subsequence level pooling or video level pooling.

FV encoding was derived from the well-known Fisher kernel, which is based on the assumption that the generation process of a local descriptor X can be modeled as a probability density function $p(X, \theta)$. Using the gradient of the G log-likelihood, it is possible to describe how the parameters in θ contribute to the generation process of X. Then, the sample can be described as follows:

$$G_\theta^X = \nabla_\theta log(p(X;\theta))/N \tag{2.1}$$

The dimensionality of this vector depends on the number of parameters in θ. A Gaussian mixture model is used to model the probability density function. Note that $\theta = \{\pi_1, \mu_1, \sigma_1 \ldots, \pi_K, \mu_K, \sigma_K\}$ contains the parameters of the model, where π, μ, σ are Gaussian mixture weights, means and diagonal covariance, respectively. An improved Fisher vector was proposed in (Perronnin et al., 2010) as follows:

$$\rho_k = \frac{1}{\sqrt{\pi_k}\gamma_k(x - \mu_k)/\sigma_k} \tag{2.2}$$

$$\tau_k = \frac{1}{\sqrt{\pi_k}((x - \mu_k)^2/\sigma_k^2 - 1)} \tag{2.3}$$

where γ_k is the weight the of the local descriptor x corresponding to the k^{th} Gaussian component, and thus:

$$m = \pi_1 N(x;\mu_1;\sigma_1) + \cdots + \pi_k N(x;\mu_k;\sigma_k). \tag{2.4}$$

Parameter γ_k can be determined as follows:

$$\gamma_k = \pi_k N(x;\mu_k;\sigma_k)/m \tag{2.5}$$

To construct the FV, the gradients of Eq. 2.2 and 2.3 are concatenated as follows:

$$FV = [\rho_1, \tau_1, \ldots, \rho_K, \tau_K] \tag{2.6}$$

2.3.1 Theoretical Background to Calculate Moving Frames

Suppose that we have a curve which describes a trajectory, as follows:

$$r(t) = (x(t), y(t), z(t)) \tag{2.7}$$

Then the tangent vector to the curve at a point is:

$$v(t) = r^{'}(t) = (x^{'}(t), y^{'}(t), z^{'}(t)) \tag{2.8}$$

Some researchers call it *velocity* vector and its length is called *speed* (Figure 2.2). The derivative vector of unitary length is especially important. It is called the unit tangent vector and is obtained by dividing the derivative vector by its length:

$$T(t) = \frac{(x'(t), y'(t), z'(t))}{r'(t)} \tag{2.9}$$

also,

$$T(t) = \frac{v(t)}{|v(t)|} \tag{2.10}$$

Since $T(t)$ is a unit vector, then

$$T(t)T(t) = 1 \tag{2.11}$$

For all values of t, if we calculate the differentiation of both sides of this equation we find that

$$2T^{'}(t)T(t) = 0 \tag{2.12}$$

Thus $T^{'}(t)$ and $T(t)$ are always orthogonal for each value of t. We define that

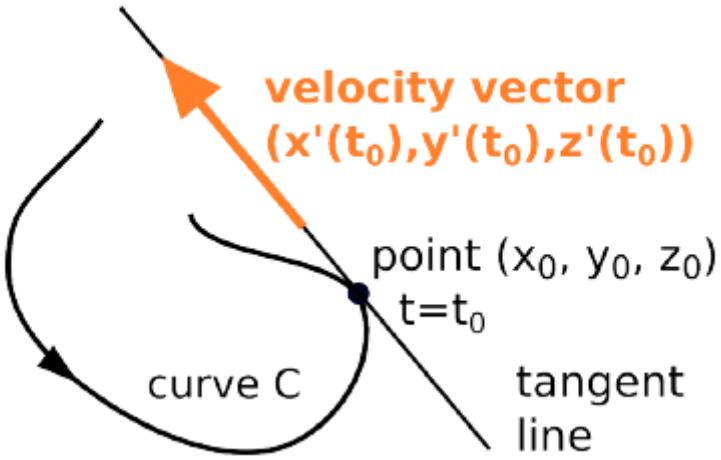

Figure 2.2: Velocity vector.

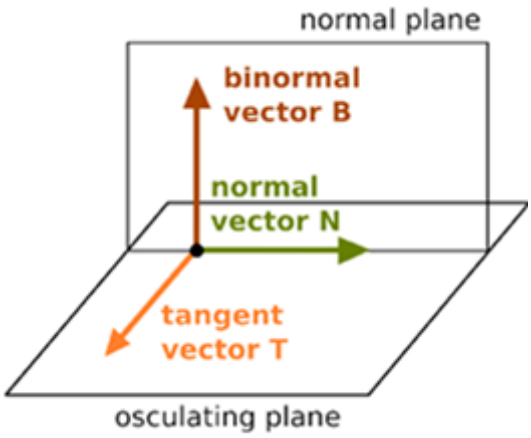

Figure 2.3: The osculating plane defined by tangent and normal vectors and the direction of the binormal vector.

unit normal as follows.

$$N(t) = \frac{T'(t)}{|T'(t)|} \tag{2.13}$$

For each point, we can span a plane using T and N. This plane is called osculating plane (see Figure 2.3). If we are dealing with 2D motion, then $z(t) = 0$. Obviously, that normal vector and osculating plane are not defined when $T^{'}(t) = 0$. The derivative of the velocity vector is called acceleration vector.

$$a(t) = a_T(t)T(t) + a_N(t)N(t) \tag{2.14}$$

This vector has the same direction as the force needed to keep the particle on the track of the curve. This force makes a particle traveling along the curve to stay on this course. Without this force, that particle would continue the motion as indicated by the velocity vector and not stay on the course of $r(t)$. The acceleration vector lies on the osculating plane too.
Suppose that $r(t)$ is a circle of radius ρ centered at $(0, 0)$.

$$r(t) = \rho(cos(\omega t), sin(\omega t)) \tag{2.15}$$

The velocity is $v(t) = p\omega(-sin(\omega t), cos(\omega t))$. The speed is $|v(t)| = p\omega = v_0$ The unit tangent vector is $T(t) = v(t)/|v(t)| = (-sin(\omega t), cos(\omega t))$ and the unit normal vector is $N = (-sin(\omega t), cos(\omega t))$. Since the speed is constant, then $a_T(t) = \frac{d}{dt}v(t) = 0$. There is no acceleration in the tangential direction. Hence, the whole acceleration should be in the normal direction, which can be described with the following two equations.

$$a = \rho\omega^2(-cos(\omega t), -sin(\omega t)) = \rho\omega^2 N \tag{2.16}$$

and

$$a = \frac{Nv_0^2}{\rho} \tag{2.17}$$

Hence, $a_N(t) = v_0^2/\rho$. One can get ρ at $t = t_0$ as follows.

$$\rho = |r^{'}(t_0)|/|T^{'}(t_0)| \tag{2.18}$$

The radius of the circle of the motion can be found from Eq. 2.18. The circle of the motion is called osculating circle. The reciprocal of the radius is called *curvature* at $t = t_0$ (see Figure 4). The curvature can be defined as follows.

$$k = |T'(t_0)|/|r'(t_0)| \tag{2.19}$$

2.3.2 Moving Frame

The bi-normal vector is defined as the cross product of T and N. Obviously, B is perpendicular to both T and N and its unit vector, since T and N have unit length. N and B determine the normal plane (see Figure 2.4). All lines in the normal plane are perpendicular to the tangent vector T.

In the literature, it is agreed to call the triple (T, N, B) moving frame (it is also known as FrenetSerret frame). The moving frame (T,N,B) is an orthonormal basis. That means that each vector is of unit length, and each pair of them is perpendicular. Therefore, every 3D vector can be represented using a linear combination of these three components. Consequently, they take over the role of the usual basis vectors $k = (0, 0, 1), i = (1, 0, 0), j = (0, 1, 0)$ at a point on the curve.

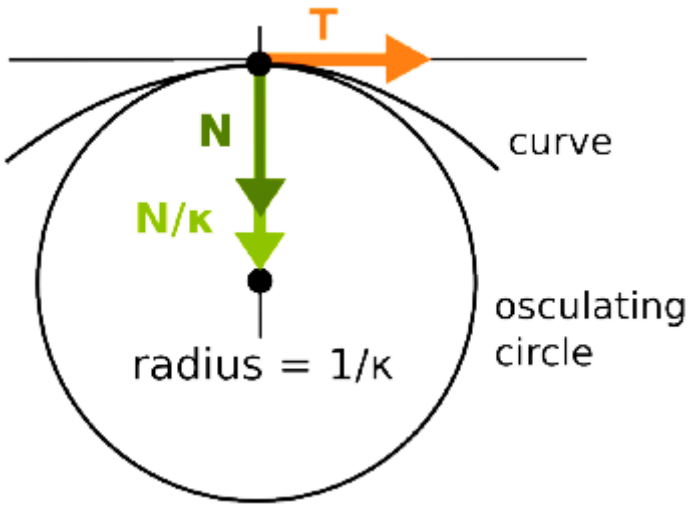

Figure 2.4: The relationship between osculating circle and curvature.

2.3.3 From Trajectories to Moving Frames

Let $r = \{r(1), \ldots, r(t), \ldots, r(n)\}$ be a trajectory in the 2D space, in which $r(t)$ represents the positions of a moving point in the t^{th} video frame, and n is the length of the trajectory. Since we exploit improved dense trajectories (Wang and Schmid, 2013), all trajectories have the same length. For each trajectory, the proposed

method calculates the moving frame (T, N, B) and the curvature k. Then, it builds a codebook for the concatenated vectors of (T, N, B, k). It does the same for HOF, HOG and MBH.

Obviously, the calculated descriptors extract meaningful information from the trajectory which does not depend on the original location of the tracked point. The good thing is that information extracted from the trajectory can cover descriptions in both 3D and 2D spaces. Furthermore, the 3D trajectory is able to accurately describe the spatial and temporal relations among the multiple trajectories. Generally, image data suffer from the motion confusion caused by the possibly different viewpoints in visual projection but it is still very useful for 2D applications. Therefore, the overall descriptor consists of the concatenation of HOF, HOG, MBH and kinematic features (moving frame + curvature).

2.3.4 Classification

In the classification step, we input the learned representations of the videos into a SVM classifier with a one-against-all strategy to classify them. A SVM classifier is a supervised learning classifier that discriminates between two classes by finding a hyperplane that separates them. Given a labeled training set of the form $(x_i, y_i), i = 1, 2, \ldots k$, where $x_i \in \mathbb{R}^n$ are the feature values, $y_i \in \{1, -1\}$ is the binary classification, n is the number of features and k is the number of instances, the SVM classifier solves the following optimization problem:

$$\begin{aligned} &\|\omega\|^2_{\omega,\xi} + C\sum_{i=1}^{k}\xi_i \\ &\qquad s.t. \quad y_i(\omega^T\phi(x_i) + b) \geq 1 - \xi_i, \\ &\qquad\qquad \xi_i \geq 0. \end{aligned} \tag{2.20}$$

In this equation, the soft margin parameter C helps the SVM optimization process to avoid misclassifying each training instance. The weight vector ω is normal to the

separating hyperplane. The parameter ξ is used to give a degree of flexibility to the algorithm when fitting the data and b represents the bias.

Each sample x_i is mapped into a higher dimensional space using a kernel function, $K(x_i, x_j) = (\phi^T(x_i).\phi(x_j))$ to make the data linearly separable in the new space. Then, the SVM classifier determines a hyperplane with a maximum margin of separation between the classes. In the case of a linear SVM classifier, ϕ refers to a dot product. In the case of non-linear SVM, the classifier function is formed by non-linearly projecting the training data of the input space to a feature space of a higher dimension by using a kernel function. In this chapter, we used a SVM with RBF-χ^2 kernel (radial basis function).

2.4 Experimental results and discussion

2.4.1 Dataset

The HMDB51 dataset, which is a generic action classification dataset (Kuehne et al., 2011), is used in this chapter. Videos in this dataset were collected from different sources: YouTube and movies. It contains around 6,670 videos, which are further grouped into 51 action classes, where each class contains around 100 videos. To measure the performance, we followed the original evaluation protocol. We used three training and testing splits and the average accuracy over the three splits was computed.

2.4.2 Experimental Setup

In our experiments, we adopted the improved dense trajectory features used in (Wang and Schmid, 2013). To implement the kinematic model, we built 256 Gaussian mixture models and used them to cluster randomly taken 250,000 samples for each type separately. The resulting GMM models were used to build a FV for each descriptor type of low level vectors. The Fisher vectors were concatenated and sent

to a classifier. We used a SVM classifier with RBF (χ^2 kernel). Since the motion is 2D, we took the first two dimensions from tangent and normal vectors and the third dimension from the bi-normal vector for each trajectory. Since the improved length of the trajectory is 15 (by default), the trajectory descriptor has 90 dimensions (30 from the tangent vector, 30 from the normal, 15 from the bi-normal and 15 from the curvature). For dimension reduction and correlation removal, an algorithm based on principle component analysis (PCA) was applied before building the FV. The PCA factor was set to 0.5.

Table 2.1: Comparison of the baseline methods with the proposed approach using the HMDB51 dataset.

Method	**Accuracy**
(Yang and Tian, 2014)	26.90%
(Wang and Schmid, 2013)	57.20%
(Hou et al., 2014)	57.88%
(Simonyan and Zisserman, 2014)	**59.50**%
Proposed approach	58.20%

In this work, a complementary descriptor is designed by using the trajectories of local descriptors. It utilizes the spatio-temporal data of the trajectories, and extracts additional information about the shape of trajectories. The proposed method enhances the recognition performance of concepts in videos. A CNN-based model has a better performance because it represents higher level semantic concepts, but it has a high time complexity and requires complicated training passes in the training stage. In turn, the proposed method is faster than (Simonyan and Zisserman, 2014). It took approximately 1 day for one temporal CNN on a system with four NVIDIA Titan cards (it took 3.1 times the aforementioned training time on single GPU). In turn, our approach took approximately 14 hours on a Core i7 2.5GHz CPU with 16 GB RAM. This certifies that our approach gives comparable results with a much smaller training time.

2.5 Conclusion

In this chapter, a new method to recognize human actions in videos is proposed. It exploits information of trajectories extracted from the motion frames. The proposed method calculates tangent, normal, bi-normal and curvature. Then, it combines them with classical low level features. The proposed approach gives a better description of the geometrical shape of the trajectories and shows comparable results with the state-of-the-art. The performance of the proposed method is evaluated by using a complex and large-scale action dataset HMDB51. Experimental results demonstrate that the proposed approach is comparable with several state-of-the-art methods as shown in Table 1.

Chapter 3

Aggregating The Temporal Coherent Descriptors in Videos Using Multiple Learning Kernel for Action Recognition

3.1 Introduction

In spite of the fact that the interest in action and event recognition has increased (Tang et al., 2012; Izadinia and Shah, 2012), there are not many works to model temporal patterns inside videos. Obviously, modeling the temporal evolution of a video is a challenging open problem due to the level of complexity mentioned before. In addition, actions in videos are performed at varying speeds, and often the speed of the same action can non-linearly change within a single video. A number of methods have been proposed to capture the temporal evolution in videos for action recognition tasks exploiting the learning-to-rank technique (Fernando et al., 2016) and hidden Markov models (Wang and Mori, 2011; Wu and Shao, 2014). In the last few years, conditional random fields based methods were also proposed (Song et al., 2013), and

recently, deep neural networks based methods were proposed to tackle the activity recognition problem (Simonyan and Zisserman, 2014; Husain et al., 2016).
One can easily realize that the impact of these works on the action recognition performance has been disappointing so far. Simpler but more powerful approaches, such as temporal pyramids, which are similar to spatially dividing the images or objects are not enough. Nevertheless, it became clear that many activities have ordered temporal patterns. In this chapter, we propose a new video representation method that captures the temporal evolution of the action in the whole video. We show that the aggregation of coherent frames may have a different impact on the results. We also propose clever combinations of different features based on multiple kernel learning.

In the literature, several researchers have studied the problem of capturing temporal patterns of videos for activity recognition. Important improvements have been seen in local motion patterns modeling. Jain et al. (2013) proposed to determine the location of actions in the video and then use them to refine recognition. Saleh et al. used the curvature of the trajectories of local descriptors and other kinematic properties to improve the performance of activity recognition methods.
To avoid designing features, it is reasonable to apply deep learning methods (Le et al., 2011; Taylor et al., 2010). The temporal pattern can thus be captured by a convolutional neural network (CNN) using two approaches. The first approach extends the connectivity of the network architecture along time. An example of this approach was proposed in (Karpathy et al., 2014). The second approach stacks optical flow frames and inputs them into a CNN (Simonyan and Zisserman, 2014). This approach is called two-stream neural network because it has two kinds of inputs: raw RGB and stacked optical flow. This network can learn visual appearance and motion information in an unsupervised way from video cubes. Indeed, CNNs have been oriented to capture different types of information. For instance, Chéron et al. (2015) represented different parts of the human body with motion and appearance

based descriptors to extract features from the human pose. These descriptors were calculated from each frame and then aggregated over time to form a video descriptor. They captured temporal patterns by considering temporal differences between frames and then concatenated the difference of vectors. In (Gkioxari and Malik, 2015), two CNNs were used to capture both appearance and motion based features in action tubes. The first network is called spatial-CNN network, which takes RGB frames as input and captures the appearance information of the actor and other visual details from the scene as well. The second network is called motion-CNN, which works on the optical flow and captures the temporal patterns of the actors. By combining the output of the hidden layers of the two networks, they extract spatio-temporal features. Srivastava et al. (2015) used the state of the long-short term memory (LSTM) encoder after observing the last input frame as a video representation. Du et al. (2015) showed that skeleton based action recognition can be managed using a hierarchical recurrent neural network. Another popular way to tackle action recognition using deep neural networks is to combine the CNN and LSTM as demonstrated in (Yue-Hei Ng et al., 2015).

Jain et al. (2015) showed how they can take advantage of objects in the videos for action classification. The aforementioned methods have a common point, which is capturing the local changes in a small temporal window, but they are not designed to capture motion patterns to model the whole video, which is an important factor in some actions.

Hidden Markov models (HMM) and discriminative conditional random fields (CRF) also have been proposed to model the dynamics of videos along time (Yamato et al., 1992; Sminchisescu et al., 2006). HMMs are used to learn a joint distribution over both observations and action targets. Thus, the observations are the visual appearance and/or local motion feature vectors acquired from input videos. HMMs can learn the appearance and/or the motion pattern along time of a given action class. The biggest challenge is to learn different variations of

one action class. Because of that, HMMs may require a lot of training samples to capture discriminative joint probability distributions. In turn, CRFs can learn to discriminate a pair of action classes by modeling the conditional distribution over their class labels. Like the HMMs, CRFs methods also need a large amount of input samples to train and capture all parameters of the models. The proposed method does not depend on the target class to extract temporal patterns from the video sequence. It relies on standard aggregation of video specific dynamic information, support vector regression (SVR) and support vector machines (SVM) to discriminate between action classes.

In the recent past, new approaches based on CRFs, HMMs and action grammars were proposed to model motion patterns for action recognition tasks (Song et al., 2013; Tang et al., 2012; Wang and Mori, 2011). Wang and Mori (2011) combined a part-based approach with large-scale template features to obtain a discriminative model based on max-margin CRFs. Song et al. (2013) defined a feature function for the aforementioned CRFs model using a series of complex heuristics. In addition, Tang et al. (2012) proposed a max-margin method to model the temporal structure in a video. They used an HMM to learn the transitions of action appearances and the duration of actions.

In the recent literature, one can find works that apply temporal ordering models for activity recognition tasks (Izadinia and Shah, 2012; Ponce-López et al., 2015; Sun and Nevatia, 2013). Their aim was to infer complex actions from predefined useful basic-level action detectors. Sun and Nevatia (2013) imposed a representation for low-level action events that includes statistical information of the action transition probabilities. Spatio-temporal primitives of sub-gestures were used to model the goal actions after detecting them through genetic algorithms. The dynamics of the target actions were modeled using HMMs or dynamic time warping. Owing to the variability of motion patterns in videos, latent sequential models are not capable to overcome this problem. Thus, temporal pyramids were introduced in (Gaidon

et al., 2012; Laptev et al., 2008) to address this problem. A method that applies Fourier analysis was also proposed in (Revaud et al., 2013). In this chapter, we model the temporal structure of videos following (Fernando et al., 2016), which models the evolution of appearance or local motion using a learning-to-rank technique. We show the effect of aggregation of coherent frames. We also demonstrate how useful is to process input videos as subsequences of related frames.

3.2 Methods

Fig. 3.1 presents the steps of the proposed method: feature extraction, coherence analysis, weighted combination, representation learning and classification. In the feature extraction step, we compute local descriptors from each frame $(F1, F2, \ldots, F_N)$, and then we encode each type of descriptors using the Fisher vector technique $(FVdesc_1, FVdesc_2, \ldots, FVdesc_k)$. In the coherence analysis step, we automatically determine the subsequences in each video. In the weighted combination step, for each subsequence we use an aggregation operator to combine the Fisher vectors of each type of descriptors extracted from each frame $(FVdesc_1, FVdesc_2, \ldots, FVdesc_k)$ into one Fisher vector $(FVdesc_c)$, and then we use the multiple kernel learning (MKL) technique to produce a weighted combination (FV_{sub}) of the Fisher vectors of all descriptors $(FV1_c, FV2_c, \ldots, FVN_c)$. In the representation learning step, we input the Fisher vectors of all subsequences into a learning-to-rank technique to produce a representation of the whole video. Finally, the representations of all videos are fed into a SVM classifier to discriminate between the actions.

3.2.1 Feature Extraction

Feature trajectories have shown to be efficient for representing videos. Wang and Schmid (2013) used dense sampled local visual features at several spatial scales.

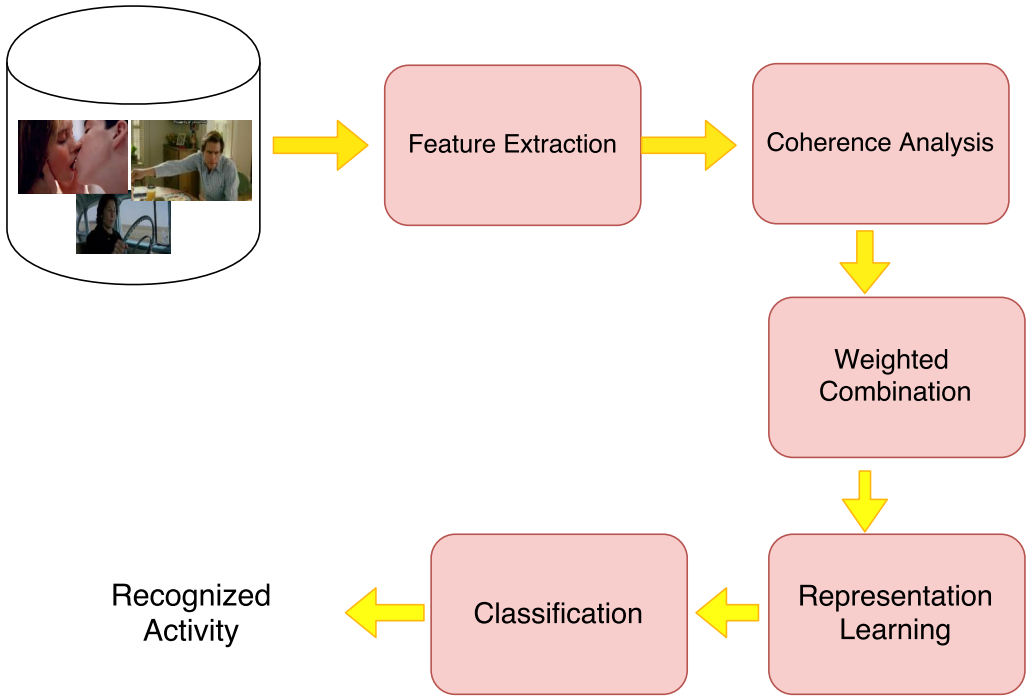

Figure 3.1: The steps of the proposed method.

They estimated a homography with RANSAC (random sample consensus) using SURF matching (speeded up robust features) between two consecutive frames. The calculated low-level descriptors were computed on the warped optical flow to capture motion patterns. Below, we explain the feature extraction methods used in this chapter: HOG, HOF, and MBH. We also explain the steps of FVs.

3.2.2 Coherence Analysis

To find the coherent frames, we calculate dense HOG descriptors for each frame in the input video $H = [h_1, h_2, ..., h_N]$, where N indicates the number of frames. Assume that h_t and h_{t+1} are HOG descriptors of two consecutive frames, we calculate the similarity between them. Thus, we determine a similarity vector for each video, $S = [s_1, s_2, ..., s_{n-1}]$. To decide if two consecutive frames are coherent, we use the following hard threshold:

$$S^{'}(t) = \begin{cases} 1 & S(t) \geq Z_{th} \\ 0 & S(t) < Z_{th} \end{cases} \tag{3.1}$$

where Z_{th} is the mean of the similarity measure vector S. Value '1' indicates that the two frames are coherent and they belong to the same subsequence. In turn, '0' indicates that the two frames do not belong to the same subsequence. In this chapter, we evaluate the impact of three similarity measures on the recognition rate (correlation, Euclidean distance and cosine measures). Fig. 3.2 shows an example of the coherence analysis in a video. Note that yellow circles include the frame number, green circles show the similarity indicator, and each red rectangle represents one subsequence (i.e., coherent frames). Frames 1, 2 and 3 are coherent (they lie in the same subsequence) while frames 3 and 4 are not coherent (they lie in the different subsequence). Although we used the HOG descriptor for coherence analysis, any other descriptor can be used to perform that task.

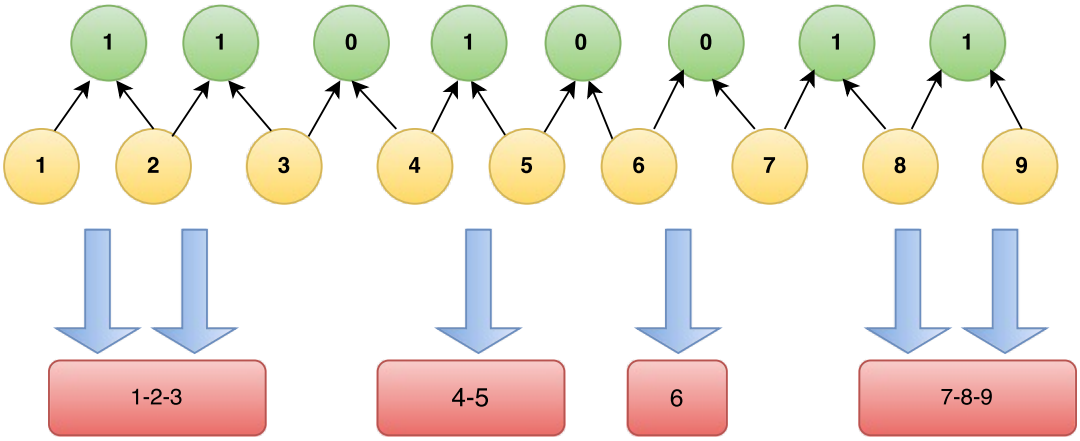

Figure 3.2: An example of coherence analysis in a video.

3.2.3 Weighted Combination

In this subsection, we explain how we use an aggregation method to combine the FVs of each type of descriptors. In addition, we explain how we use the MKL technique to produce a weighted combination of the FVs of all descriptors.

3.2.3.1 Aggregation Methods

For each subsequence, we aggregate the FVs of each type of descriptors (HOG, HOF and MBH) extracted from each frame $(FVdesc_1, FVdesc_2, \ldots, FVdesc_k)$ into one FV $(FVdesc_c)$ using an aggregation operator. In this chapter, we evaluate the performance of two aggregation operators: mean and median. For example, we can use the mean operator to aggregate the FVs of HOG descriptors $(FVHOG_1, FVHOG_2, \ldots, FVHOG_k)$ extracted from a subsequence of k frames as follows: $FVHOG_c = \frac{1}{k}\sum_{a=1}^{k} FVHOG_a$. This step reduces the complexity of the problem because it reduces the dimension of the vector that we input into the MKL algorithm.

3.2.3.2 Multiple Kernel Learning (MKL)

MKL methods are used to learn an optimal linear or non-linear combination of predefined kernels. MKL methods have been used for a long time in many applications, such as event recognition, object recognition in images and biomedical data fusion (Chen et al., 2013; Bucak et al., 2014; Yu et al., 2010). Obviously, the nature of these resources are different. Consequently, they have different notions of similarity and thus require different kernels. MKL algorithms can be used to combine kernels already established from different data sources instead of creating a new kernel.

In this chapter, we input the FVs obtained from the aggregation step of each type of descriptors ($FVHOG_c, FVHOF_c$ and $FVMBH_c$) into a MKL algorithm to produce a weighted combination (FV_{sub}) for each subsequence. Indeed, there are a lot of reasons to use MKL, such as:

- It gives us the ability to choose optimal kernel and parameters from a larger set of kernels (which is the case in practice). This automatic selection reduces the bias.
- It provides an intelligent way to combine data from different sources (in this

chapter we combine the FVs of HOG, HOF and MBH).

MKL algorithms have been developed for supervised, semi-supervised and unsupervised learning. Big part of the progress has been done on supervised learning with linear combinations of kernels. One can explain MKL algorithms as the addition of an extra parameter to the minimization problem of the learning algorithm. For instance, in case of supervised learning of a linear combination of a set of n kernels. Let us introduce a kernel $K^{'} = \sum_{i=1}^{n} \beta_i K_i$ where β is the coefficient vector for the kernels. Since the kernels are additive, the new function is still a kernel. Given a set of data X with labels Y, one can rewrite the minimization problem as follows: $\min_\beta \mathrm{E}(Y, K') + R(K)$, where R is a regularization term and E is the error function. The definition of E depends on the solving method, which can be the square loss function or the hinge loss function (for SVM algorithms), and R is an ℓ_n-norm or any combination of norms. This optimization problem can then be solved by any standard optimization method (Bach et al., 2004).

Notation: In this work, we follow Aiolli and Donini (2015) to solve a classification problem with training samples defined as $\{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$ and test samples defined $\{(x_{l+1}, y_{l+1}), (x_{l+2}, y_{l+2}), ..., (x_L, y_L)\}$, where $x \in R^m$ and $y \in \{-1, 1\}$ if we solve a binary classification problem or $y \in \{1, 2, .., C\}$ if we solve a multi-class classification problem. For simplicity, suppose that our problem is binary. Let $X \in R^{L \times m}$ denote the matrix where samples are arranged in rows and $y \in R^L$ the vector of labels. Let $K \in R^{L \times L}$ denote the kernel matrix, which contains the kernel values of each data point (training and test). Given a training set, the domain of probability distributions $\gamma \in R^l_+$ defined over the sets of positive and negative training examples can be denoted as $\hat{\Gamma}$:

$$\hat{\Gamma} = \left\{ \gamma \in R^l_+, \sum_{i \in \oplus} = 1, \sum_{i \in \ominus} = 1. \right\} \tag{3.2}$$

Note that any element γ in $\hat{\Gamma}$ corresponds to a pair of data points.

Scalable multiple kernel learning algorithm (EasyMKL): In this chapter,

we use the EasyMKL method (Aiolli and Donini, 2015) because it can easily deal with hundreds of thousands of kernels and more. EasyMKL showed robustness in comparison with other alternative methods (random, average, etc.) even for data containing noise. EasyMKL learns a vector of coefficients η that define the combined kernel according to:

$$K = \sum_{r=1}^{R} \eta_r K_r, \eta_r \geq 0 \tag{3.3}$$

It is clear that one must restrict the possible choices of K, and this can be done by a regularization term. The problem of learning the kernel combination is posed as a $min - max$ problem over variables γ and η. Aiolli and Donini (2015) proposed to maximize the distance between the positive and negative samples with a unitary norm vector η as the weak kernel combination vector:

$$\max_{\eta:\|\eta\|=1} \min_{\Gamma\in\gamma} (1-\lambda)\gamma^T \hat{Y}(\sum_{r}^{R} \eta_r \hat{K}_r)\hat{Y}\gamma + \lambda\|\gamma\|^2 \tag{3.4}$$

Consider the following equation:

$$Q(\eta,\gamma) = (1-\lambda)\gamma^T \hat{Y}(\sum_{r}^{R} \eta_r \hat{K}_r)\hat{Y}\gamma + \lambda\|\gamma\|^2 \tag{3.5}$$

If the vector of the rth entry is defined as $d_r(\gamma) = \gamma^T \hat{Y}\hat{K}_r\hat{Y}\gamma$, then

$$Q(\eta,\gamma) = (1-\lambda)\eta^T d(\gamma) + \lambda\|\gamma\|_2^2 \tag{3.6}$$

Consequently, the main problem can be rewritten as:

$$\min_{\gamma\in\Gamma} \max_{\eta:\|\eta\|=1} Q(\eta,\gamma) = \min_{\gamma\in\Gamma} \max_{\eta:\|\eta\|=1} (1-\lambda)\eta^T d(\gamma) + \lambda\|\gamma\|_2^2 \tag{3.7}$$

Finding η^*, which is a simple analytic solution that maximizes Q:

$$\eta^* = \frac{d(\gamma)}{\|\gamma\|_2} \tag{3.8}$$

and

$$\min_{\gamma\in\Gamma} Q(\eta^*,\gamma) = \min_{\gamma\in\Gamma}(1-\lambda)\|d(\gamma)\|_2 + \lambda\|\gamma\|_2^2 \tag{3.9}$$

The optimal γ is a regularized minimizer of ℓ_2-norm of the distance vector. It is possible to find the minimizer as this is a convex function. To simplify the problem further, Aiolli and Donini (2015) proposed to minimize an upper-bound instead of corresponding to the ℓ_1-norm of the distances, and thus they obtained the following equation:

$$\min_{\gamma\in\Gamma}(1-\lambda)\|d(\gamma)\|_1 + \lambda\|\gamma\|_2^2 = \min_{\gamma\in\Gamma}(1-\lambda)\gamma^T\hat{Y}(\sum_{r}^{R}\hat{K}_r) + \lambda\|\gamma\|_2^2 \tag{3.10}$$

3.2.4 Representation Learning

To learn a representation of each video, we use the learning-to-rank technique, which consists of the following three steps: frame-level representation, sequence smoothing, and learning the representation of the whole video (Fernando et al., 2016).

1. **Frame-level representation:** Each video can be represented as follows: $X = [x_1, x_2, ..., x_n]$, where x_t contains the descriptors of a frame at time step t.
2. **Sequence smoothing:** To avoid abrupt action changes in videos, we smooth the sequence X using time varying mean as follows: $m_t = \frac{1}{t}\sum_{i=1}^{t} x_i$. We then normalize m_t as follows: $v_t = \frac{m_t}{\|m_t\|}$. It is clear that v_t only contains information about the direction of m_t. Therefore, this approach guarantees a smooth output.
3. **Learning the representation of the whole video:** The learning-to-rank technique was proposed in (Fernando et al., 2016, 2015) for activity recognition. Given the smoothed vectors, the learning-to-rank technique learns their rank (i.e., $m_n \succ m_{n-1} \succ m_{n-2} \succ \cdots \succ m_1$). A linear learning-to-rank technique learns a pairwise linear function $\psi(m_t; u)$, where $u \in \mathbb{R}^D$ is a vector containing its parameters. The ranking score of m_t is computed by $\psi(m_t; u) = u^T.v_t$.

The learning-to-rank technique optimizes the parameters u of $\psi(m_t; u)$ using the objective function of Eq. 3.11 with the constraint $\forall t_i, t_j \quad v_{t_i} \succ v_{t_j} \iff v_{t_i}.u^T \succ v_{t_j}.u^T$ as follows:

$$\arg\min_{u} \quad \frac{1}{2}\|u\|^2 + C \sum_{\forall i,j, v_i \succ v_j} \epsilon_{ij}, \quad s.t. \quad u^T(v_{t_i} - v_{t_j}) \geq 1 - \epsilon_{ij},$$

$$\epsilon_{ij} \geq 0. \qquad (3.11)$$

where C is the regularization parameter and ϵ is the margin of tolerance. The parameters u are used to describe the evolution of actions in videos. In other words, we use u to represent the whole video.

3.2.5 Classification

In the classification step, we input the learned representations of the videos into a SVM classifier with a one-against-all strategy to classify them.

3.3 Experimental Results and Discussion

3.3.1 Datasets

We used two state-of-the-art datasets to evaluate the proposed method: HMDB51 and Hollywood2. Hollywood2 is one of the largest and most challenging available datasets for real world actions. There are 12 classes: answering phone, driving car, eating, fighting, getting out of car, shaking hands, hugging, kissing, running, sitting down, sitting up and standing up. The actions were collected from a set of 69 Hollywood movies. It consists of around 487k frames (about 20 hours of video). It is divided into a training set of 823 sequences and a test set of 884 sequences. There is no overlap between the 33 movies in the training set and the 36 movies in the testing set (Marszalek et al., 2009). As is standard on this dataset, performance of

our method is measured by the mean average precision (mAP) over all classes.

HMDB51 is a generic action classification dataset (Kuehne et al., 2011). The videos of this dataset were collected from YouTube and some movies. It contains 6,670 videos, which were further grouped into 51 action classes, in which each class contains around 100 videos. To measure the performance of the proposed method, we used three training and test splits and then the average accuracy over them was computed.

3.3.2 Experimental Setup

To implement FVs, we set the number of Gaussians to K= 256 and randomly sample a subset of 256,000 features from the training set to estimate the GMM. Recent evaluations (Chatfield et al., 2011; Oneata et al., 2013; Wang et al., 2013a) showed that the aforementioned values give good results with the action classification. The resulting GMM models were used to build FVs. We used the VLFeat library to implement this part. For the learning-to-rank method, we used the source code of (Fernando et al., 2015). For MKL , we used the source code of (Aiolli and Donini, 2015), setting the value of λ to 0.1 and the value of γ of the radial basis function to 0.1. In our experiments, we set C = 1.0 (regularization-loss trade off) for the SVM classifier, which gave good results when validating on a subset of training samples.

3.3.3 Results

We assess the impact of three similarity measures (correlation, Euclidean distance and cosine measures) and two aggregation operators (mean and median) on the performance of the proposed method. As shown in Table 3.1 and 3.2, the cosine similarity and the correlation measure gave the same recognition rates with Hollywood2 dataset because they can be viewed as variants of the inner product. The mean operator is sensitive to extreme observations, and thus it can be pulled to the direction of the outliers (outliers are extreme values that differ greatly from other values). In general, the median operator may provide a better performance

Table 3.1: The impact of changing similarity measures and the use of mean function for aggregation on the recognition rate with Hollywood2 dataset. Each row contains the mAP rate of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	38.8	67.2
Cosine	46.8	**69.2**
Correlation	46.8	**69.2**

Table 3.2: The impact of changing similarity measures and the use of median function for aggregation on the recognition rate with Hollywood2 dataset. Each row contains the mAP rate of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	43.1	68.1
Cosine	46.8	**69.2**
Correlation	46.8	**69.2**

Table 3.3: The impact of changing similarity measures and the use of mean function for aggregation on the recognition rate with HMDB51 dataset. Each row contains the recognition accuracy of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	31.1	53.8
Cosine	37.4	55.4
Correlation	37.4	55.4

with outliers. In the case of the Hollywood2 dataset, each subsequence consists of homogeneous frames, and this clearly explains why we have identical values in Table 3.1 and 3.2.

In Table 3.3 and 3.4, we present the recognition accuracy with the HMDB51 dataset. As we can see, the cosine similarity and the correlation measure also gave the best recognition results. We notice that in both datasets, the median function gave recognition accuracy better than the one of the mean operator with the Euclidean distance. As we see in Tables 3.1−3.4, the recognition results are improved when we combine the proposed method with the baseline of (Fernando et al., 2015). After Applying the MKL technique, we got the state-of-the-art results with the Hollywood2 dataset and comparable values with the HMDB51 dataset.

In Table 3.5 we show the performance of the proposed method with the two datasets. With the Hollywood2 dataset, we achieved an accuracy of 70.2%. The application of coherence analysis and Fisher vector aggregation of the obtained subsequence information can help beating the state-of-art methods. It also outperforms the

Table 3.4: The impact of changing similarity measures and the use of median function for aggregation on the recognition rate with HMDB51 dataset. Each row contains the recognition accuracy of each similarity metric.

Method	Proposed method	Proposed+ (Fernando et al., 2015)
Euclidean distance	34.5	54.6
Cosine	37.4	55.4
Correlation	37.4	55.4

performance of recent deep learning approaches, such as (Sharma et al., 2015) and (Srivastava et al., 2015).

It is clear from Table 3.5 that the results of the proposed method with the HMDB51 and Hollywood2 datasets are comparable to the ones of (Fernando et al., 2015), as a result of the existence of a big number of slow actions in the datasets. The CNN model proposed in (Simonyan and Zisserman, 2014) has a good performance because it represents a higher level of semantic concepts, but it has a high time complexity and requires complicated training passes. In turn, the proposed method is much faster. The model proposed in (Simonyan and Zisserman, 2014) took approximately one day for one temporal CNN on a system consisting of four *NVIDIA Titan cards* (it took 3.1 times the training time if a single GPU was used). In turn, our approach took approximately 16 hours on a Core i7 2.5GHz CPU with 16 GB RAM.

Table 3.5: Comparison between the proposed method and related works using HMDB51 and Hollywood2 datasets.

Method	HMB51	Hollywood2
Two-stream ConvNet (Simonyan and Zisserman, 2014)	59.40	-
Learning-to-rank (Fernando et al., 2015)	61.80	70.00
Composite LSTM Mode (Srivastava et al., 2015)	44.00	-
Kinematic of the Trajectories	58.20	-
Coherence Analysis + FVs aggregation	40.55	65.00
Coherence Analysis+ Learning-to-rank	61.90	**70.20**
Coherence Analysis + Learning-to-rank + MKL	**62.1**	69.73

Fernando et al. (2015) used the *time varying mean* method for smoothing the feature vectors in order to cope with abrupt changes in human action. They did not use the *moving average* (MA) method for that task because of two reasons:

1. A window size must be chosen, which is not always straightforward as actions often take place at different times.

2. The last frames of a video are ignored.

In our method we automatically divide each video into a set of subsequences. Therefore we do not need to define a window size to aggregate or smooth the feature vectors. The window size is defined by the length of each subsequence (adaptive window size). In this way, we avoid the main problem of using the MA method. Furthermore, we compute the mean or the median over an adaptive window size. Thus, we guarantee that each vector in the time series is included.

The main reason for the low recognition rates is that temporal coherence can be a weak measure of discriminability with fast actions. If the frames are quite distinct, then maximizing the temporal coherence directly leads to weak discriminative features. After coherence analysis, if we get distinct frames in each subsequence, we lose the power of using subsequences, which is the case in fast actions. Indeed, slow change across frames means slow actions. Slow actions make the aggregation more effective. As shown in Table 3.2, there is a degradation in performance of our method when we used the median operator for subsequence aggregation because the subsequences contain few frames (due to fast actions).

The proposed method also outperforms fresh deep learning approaches because the coherence analysis step suppresses the redundant information in the videos. In other words, we suppress unnecessary information via computing one FV for each subsequnce. Fig. 3.3 shows the confusion matrix of the Hollywood2 dataset. As shown in Table 3.7, with the Hollywood2 dataset our method gives better results in the case of slow actions (with less dramatic changes) as in the following cases: AnswerPhone, FightPerson, SitDown, SitUp and StandUp. The proposed method works better than the baseline of (Fernando et al., 2015) in the case of slow actions since it makes the input sequences more generalized and lightens the unnecessary details.

Table 3.6 shows the diagonal elements of the confusion matrix of the HMDB51 dataset. Note that we cannot display the complete confusion matrix because it has

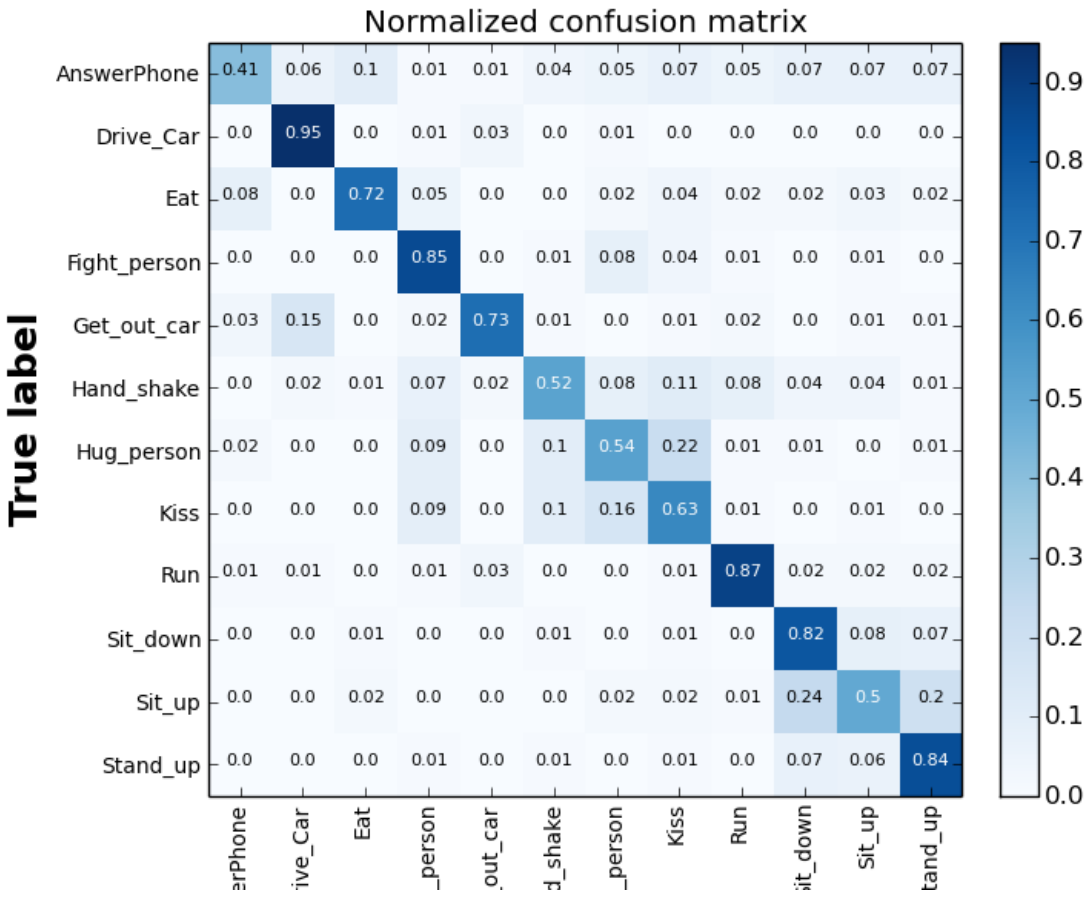

Figure 3.3: The confusion matrix for the Hollywood2 dataset.

a size of 51 × 51. Table 3.8 shows the performance of the proposed method with 12 randomly selected actions from the 51 classes of the HMDB51 dataset. We obtained better recognition with the following actions: Chew, Climb, Climbs tair, Dribble, Drink, Eat, Fall floor, and Kick. All these actions are slow. The main reason of these results is the difficulty of the recognition of some videos and the high variation within the videos of several classes. We notice that the available action recognition datasets lack diversity. Consequently, the trained models are suffering from overfitting, thus training and test datasets should be collected in the cross category style to generalize actions and avoid over-fitting.

3.4 Conclusion

In this chapter, we have modeled the evolution of human actions for activity recognition. The proposed method consists of five steps: feature extraction, coherence analysis, weighted combination, representation learning and classification. In the feature extraction step, we used HOG, HOF and MBH feature extraction methods, and then we encoded each type of descriptors using the Fisher vector technique. In the coherence analysis step, we determined the subsequences of each video. We evaluated the effect of three similarity measures (correlation, Euclidean distance and cosine measures) on the performance of the proposed method. The cosine similarity and correlation measure gave the same recognition rates. For each subsequence, we used an aggregation method to combine the FVs of each type of descriptors extracted from each frame into one FV. To do so, we used the mean and median operators. We used the MKL technique to produce a weighted combination of the FVs of all descriptors. We input the FVs of all subsequences into a learning-to-rank method to produce a representation of the whole video. The representations of all videos were fed into a SVM classifier to discriminate between the actions. We showed the effect of the aggregation of coherent frames on the recognition rate. We also demonstrated how useful is to process input videos as subsequences of related frames. With HMDB51 and Hollywood databases, we showed how different aggregation methods can change the recognition performance and combining the aggregated FVs using the MKL method can have a positive impact on the recognition results.

To further improve the results, the future work will focus on using features extracted from deep CNNs in the feature extraction step. We will also use the dynamic image technique in the representation learning step.

Table 3.6: The diagonal elements of the confusion matrix of the HMDB51 dataset.

Class	Accuracy	Class	Accuracy	Class	Accuracy
Brush hair	0.76	Hit	0.57	Shoot ball	0.89
Cartwheel	0.52	Hug	0.90	Shoot bow	1.00
Catch	1.00	Jump	0.38	Shoot gun	0.85
Chew	0.47	Kick ball	0.31	Sit	0.39
Clap	0.46	Kick	0.68	Situp	1.00
Climb stairs	0.98	Kiss	0.93	Smile	0.44
Climb	0.65	Laugh	0.41	Smoke	0.43
Dive	0.85	Pick	0.27	Somersault	0.52
Draw	0.47	Pour	1.00	Stand	0.27
Dribble	1.00	Pullup	1.00	Swing	0.16
Drink	0.51	Punch	0.51	Sword exercise	0.42
Eat	0.51	Push	0.848	Sword	0.13
Fall floor	0.35	Pushup	1.00	Talk	0.67
Fencing	0.61	Ride bike	1.00	Throw	0.16
Flic flac	0.68	Ride horse	1.00	Turn	0.222
Golf	1.00	Run	0.41	Walk	0.38
Handstand	0.73	Shake hands	0.82	Wave	0.14

Table 3.7: Analyzing the performance of the proposed method and (Fernando et al., 2015) with the Hollywood2 dataset.

Action	(Fernando et al., 2015)	Proposed method
AnswerPhone	0.39	**0.41**
DriveCar	0.96	0.95
Eat	0.74	0.73
FightPerson	0.84	**0.85**
GetOutCar	0.74	0.72
HandShake	0.55	0.52
HugPerson	0.53	0.53
Kiss	0.63	0.63
Run	0.89	0.88
SitDown	0.80	**0.81**
SitUp	0.45	**0.50**
StandUp	0.83	**0.84**

Table 3.8: Analyzing the performance of the proposed method and (Fernando et al., 2015) with the HMDB51 dataset.

Action	**(Fernando et al., 2015)**	**Proposed method**
Brush hair	0.89	0.76
Cartwheel	0.52	0.52
Chew	0.39	**0.47**
Clap	0.84	0.46
Climb stairs	0.52	**0.98**
Climb	0.62	**0.65**
Dive	0.53	0.85
Dribble	0.56	**1.00**
Drink	0.42	**0.51**
Eat	0.37	**0.51**
Fall floor	0.32	**0.35**
Kick ball	0.23	**0.31**

Chapter 4

Deep Visual Embedding for Image Classification

4.1 Introduction

Several methods have been proposed for classification, clustering and indexing text documents in the literature. Those methods are fully grown and very effective to deal with huge numbers of classes. Textual data contains words and thus it can be processed using information about words. It is always possible to learn a word representation upon the existence of a given word in different documents. This idea can also be applied to images. Indeed, any image can be considered as a document containing a set of patches. However, this analogy would not stand further because text words are discrete values while local image patches are represented using high-dimensional and real-valued descriptors. To obtain discrete visual words, we should find a way to project image patches or patch descriptors to a single integer number per patch or descriptor. Therefore, local patches or descriptor vectors can be represented in terms of the region (i.e. discrete word) that they belong to. In the text prepossessing field, vector representations of words (word embedding or *word2vec*) has been recently proposed to capture the context-based relationships between words.

Those representations achieve the state-of-the-art results in different text processing tasks. Indeed, the main obstacle to apply those methods to images is the above mentioned nature of image patches.

In this chapter, we build a bridge between images and text documents, which is applicable to images in total analogy with textual word embedding. In an end-to-end fashion, the way of visual embedding is done in the same manner as in natural language processing (NLP). The proposed method uses a very popular text analysis method called 'word2vec'. In this method, we learn visual dictionaries based on filters of convolution layers of a convolutional neural network (CNN), which is used to capture the visual context of images. We employee visual embedding to convert words to real vectors. In addition, we assess different designs of dictionary building methods.

4.2 Related Works

Word embedding is basically generating real valued vectors that capture semantic similarity. Such vectors are used as representations of words for NLP tasks, such as sentiment analysis, text classification and document clustering. These representations have been learned using neural networks (Collobert and Weston, 2008; Mikolov et al., 2013a) and then used to initialize a multi-layer network model (Collobert and Weston, 2008; Mikolov et al., 2013a). Like these approaches, we learn word embedding from image online and fine-tune them to predict the visual classes of images. Xu et al. (Xu et al., 2014) and Lazaridou et al. (Lazaridou et al., 2015) use the visual appearance to improve the word2vec representation by predicting real image representations from *word2vec*. While they focus on capturing the appearance, we focus on capturing the visual words inside the image. Other groups of researchers use visual and textual attributes to learn distributional models of the meaning of words (Silberer et al., 2013; Silberer and Lapata, 2014). In turn, our set of visual words are learned in the training step of a CNN. Other researchers

use word embedding inside larger models for complex tasks, such as image captioning (Kiros et al., 2014b; Vinyals et al., 2015) and image retrieval (Kiros et al., 2014b). These works are multi-modal (i.e., textual-visual). In turn, we only apply visual embedding.

Bags-of-visual words (BOVW) have been proposed by Sivic et al. in (Sivic and Zisserman, 2003). BOVW became very popular due to its efficiency and high performance. Other works, such as probabilistic latent semantic analysis (pLSA) (Sivic et al., 2005) and latent Dirichlet allocation (LDA) (Dance et al., 2004) use unsupervised classification. They compute latent concepts in images using the co-occurrences of visual words. These techniques obtain best results when the number of categories is known. The random forests classifier has been used in some works, e.g., (Dance et al., 2004). However, the state-of-the-art results were obtained with the support vector machines (SVM) classifier. The authors of (Bosch et al., 2007) combine local matching of the features and specific kernels based on the χ^2 to get the best results. Most BOVW methods build their visual words in a clustering step. In turn, we learn our dictionary from the data using CNN or we build them using local binary pattern (LBP) or Gabor filters.

Vision and NLP: Recently, different problems at the intersection of NLP and vision are considered to be interesting to researchers (Kottur et al., 2016; Tirilly et al., 2008). Significant breakthroughs have been achieved in several tasks, such as visual question answering (Vinyals et al., 2015; Antol et al., 2015; Gao et al., 2015), image captioning (Chen and Lawrence Zitnick, 2015; Donahue et al., 2015a; Hodosh et al., 2013), aligning text and vision (Karpathy and Fei-Fei, 2015; Kiros et al., 2014a) and video description (Donahue et al., 2015b). Unlike these works, our approach is generic, i.e., it can be also used for multiple tasks.

4.3 Word Embedding

In NLP, word embedding is a set of language modeling algorithms where words and phrases are mapped to vectors of real numbers. Mathematically, embedding is a word projection from a space with one dimension (sparse vector) to a continuous vector space with a much lower dimension. The propsal in (Mikolov et al., 2013b; Pennington et al., 2014) showed that mapping can be done using neural networks. In the literature, other word embedding models were proposed, such as probabilistic models (Mikolov et al., 2013b), dimensionality reduction based models (Lebret and Collobert, 2013), word context based (Lebret and Collobert, 2013). Experiments showed that word and phrase embeddings are able to boost the performance in NLP tasks (for example, syntactic parsing and sentiment analysis) when they were used as the underlying input representation. Taking into account this intuition, and supposing that patterns from a given (or learned) dictionary are occurring in the images, any image can be converted to an image of visual words. In this work, we follow (Pennington et al., 2014) to do the embedding step. Below, we discuss some options to build the dictionaries.

Simple pixel dictionary: It is clear that the use of values of image pixels are the simplest dictionary that can be built. It consists of gray-scale values of the pixels. The main disadvantages of this approach are:

- It does not capture any information about the context. It only describes a single pixel.
- It is sensitive to noise.

Local binary pattern (LBP): The LBP operator was proposed by Ojala et al. (Ojala et al., 2002). Indeed, the LBP descriptor is one of the most successful descriptors due to its outstanding advantages, such as robustness to illumination changes, low complexity in terms of computation and implementation. LBP characterizes the structure of a local image patch. The main idea of LBP is to

convert differences between a given pixel value at the central point and those of its neighbors. The value of the binary pattern is used to label the given pixel. The response of LBP is calculated as follows:

$$LBP = \sum_{p=0}^{P} s(g_p - g_c)2^p \tag{4.1}$$

where P is the number of neighboring pixels, g_c is the center pixel value and g_p is a neighboring pixel. As shown in Eq.1, for any pixel in the neighborhood, if the center pixel's value is greater than the neighbour's value, $s = 0$ is set to the neighbor pixel; otherwise, $s = 1$ is set to the neighbor pixel. This gives an 8-digit binary number and it is usually converted to a decimal value.

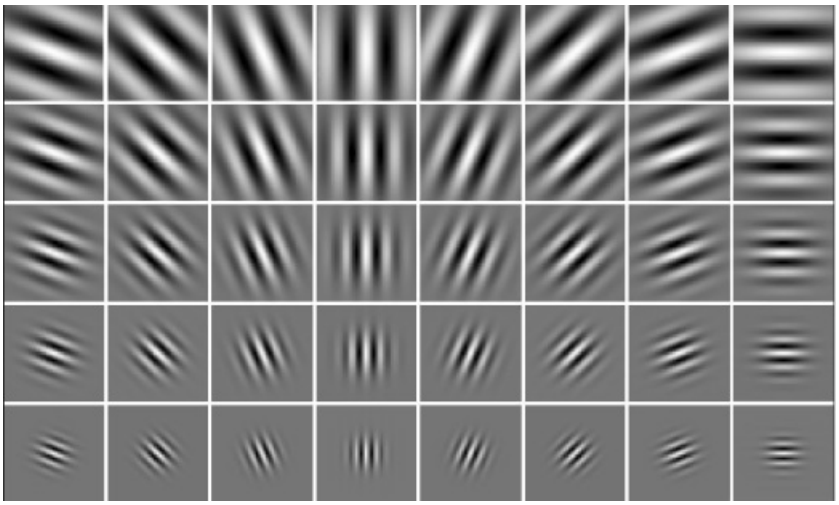

Figure 4.1: Gabor filters

Gabor filters: Fig. 4.1 sows a set of Gabor filters with various scales and rotations. Jones and Palmer explained in (Ojala et al., 2002) that the real part of the complex Gabor function is a good simulation to the receptive field weight functions found that found in simple cells in the mammals striate cortex. In other words, image analysis using Gabor filters is similar to the human visual system. The impulse response of Gabor filters is defined analytically as a Gaussian function multiplied by a sinusoidal wave or a plane wave in case of 2D Gabor filters. As shown in Eq. 2, the filters have a real and an imaginary component representing orthogonal

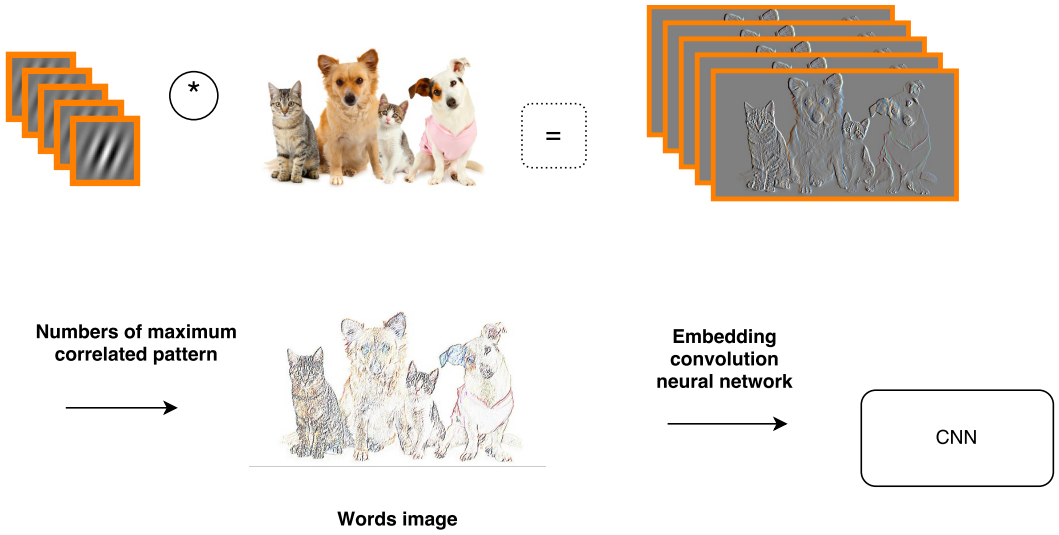

Figure 4.2: Embedding procedure for analytically given dictionary

directions. Components can be formed into a complex number or used individually. The complex form of Gabor filters can be defined as follows:

$$g(x, y, \lambda, \theta, \psi, \omega, \gamma) = exp(-\frac{x'^2 + \gamma^2 y'^2}{2\omega^2})exp(i(2\pi\frac{x'}{\lambda} + \psi)) \tag{4.2}$$

And the real part can be given as follows:

$$g(x, y, \lambda, \theta, \psi, \omega, \gamma) = exp(-\frac{x'^2 + \gamma^2 y'^2}{2\omega^2})cos(2\pi\frac{x'}{\lambda} + \psi) \tag{4.3}$$

where $x' = xcos\theta + ysin\theta$ and $y' = -xsin\theta + ycos\theta$. Convolving an image I with a bank of Gabor filters with N filters will produce N images. The resulting images are stacked horizontally as shown in Fig. 4.2 and for every x, y in the output image, the number of the filter with the highest response in the same coordinates is written. It is always possible to find the word in a given location using a similar policy (see Fig. 4.2).

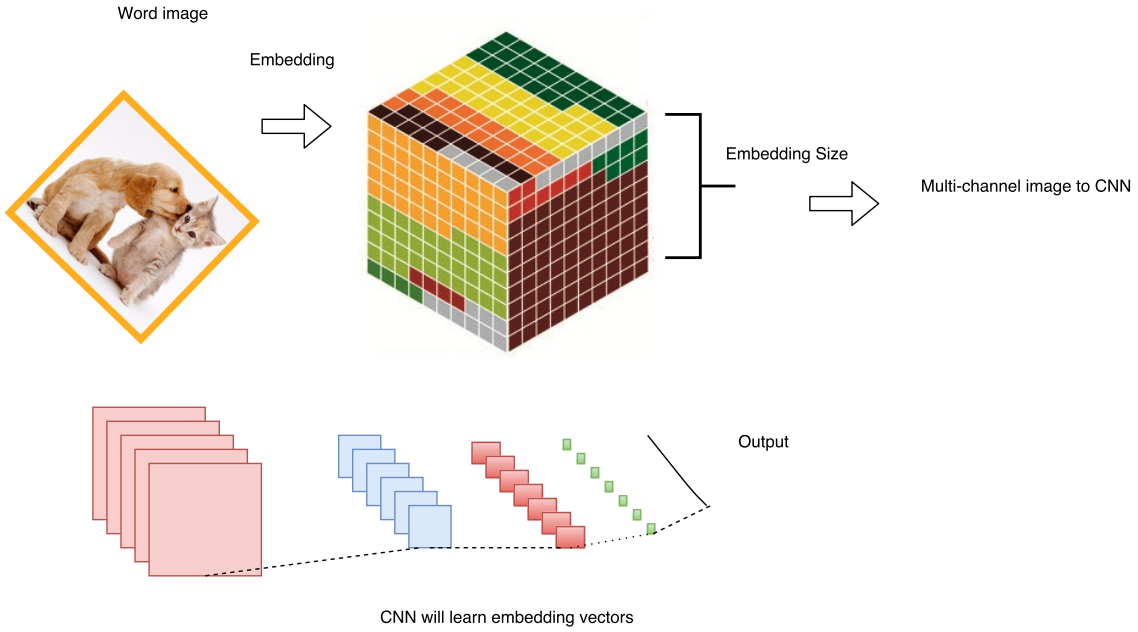

Figure 4.3: CNN architecture for a given dictionary

4.4 Proposed Method

In this section, we show how to compute the visual words and find them inside the images. The proposed method is independent of the dictionary building step, as long as the resulting dictionary has a finite number of visual words. Each patch in the image will be converted to the number of closest visual word in the used dictionary. This step produces a 'words image' (see Fig. 4.3). It is then passed to a CNN that captures the spatial context of words. After that, every word in the word image is embedded to a real valued vector with embedding size d, resulting in an image with d channels. Then, the d-channel image passes through the layers of the CNN and the process continues until we get a stable embedding vector.

As shown in Fig. 4.4, in the proposed method we use filters learned by CNN to build a dictionary. The first layer of the CNN is a convolution layer with N filters. Afters passing through this layer, we will get N feature maps. The number of filters with the highest responses are chosen, and then the resulting responses are passed

to the next layers. It is possible to learn filters (build a dictionary). The size of the dictionary equals the number of filters of the first convolutional layer. In this chapter, we adopt the learned filters to build a dictionary. We also consider that a Gabor dictionary is a special case of learned filters.

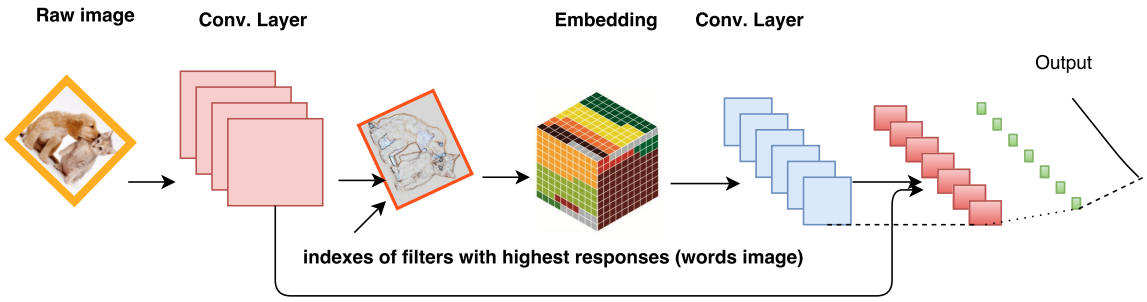

Figure 4.4: Proposed method

4.5 Experimental Results an Discussion

4.5.1 Datasets

To assess the performance of the proposed method, we use two state-of-the-art image classification datasets: MNIST and CIFAR-10.

The MNIST dataset contains 70,000 28x28 gray-scale images. The images contain hand-written digits from 0 to 9. In the dataset, there are 60,000 training images and 10,000 test images.

The CIFAR-10 dataset contains 60,000 RGB images of 32x32 pixels belonging to 10 classes. The dataset is split into 50,000 training images and 10,000 test images.

To assess the performance of the proposed model in the action recognition task, we use the **UB101** dataset (Ma et al., 2017). This dataset consists of around 23.8K action images that correspond to the 101 human actions in the UCF101 dataset (one of the state-of-the-art datasets for human activity recognition in videos). One can find at least 100 images in each class and most classes have 150-300 images.

4.5.2 Model Setup

In our proposed model, the first layer of the CNN is a convolution layer. This layer consists of 128 filters (the size of dictionary in our model is 128 words). The output of the first layer is an image with 128 channels. To convert the multi-channel image to a word image, we take the index of the channel with the highest response. The intuition behind that is that word (filter) with the highest response is the closest to the patch in a given location. Thus, we have a single-channel image which contains indices of closest words. The next embedding layer will convert each single pixel of the word image to a vector of real values. In our experiments, an embedding size of 16 is used. Thus, each word image will be converted to an image of 16 channels containing real values. The next layer is a convolutional layer, in which the number of filters is set to the visual dictionary size. Afterwards, the outputs of the first and second convolutional layers are concatenated to learn visual words, which are the filters of the first convolutional layer.

4.5.3 Results

Table 4.1 shows that the performance of CNN can be improved by adding embedding in the bottom layers of the model. Our custom CNN model gives an error of 1.0 with the MNIST dataset without embedding and 0.5 after embedding, which is a quite remarkable improvement. Fig. 4.5 shows the change of accuracy across training and test epochs of the CNN, while Fig. 4.6 presents the confustion matrix of the proposed method with the MNIST dataset. Applying the same model to the CIFAR10 dataset gives an accuracy of 0.79 without embedding and 0.81 after embedding at the bottom of the CNN. Fig. 4.7 demonstrates the change of accuracy across training and test epochs of the CNN, while Fig. 4.8 presents the confusion matrix of the proposed method with the CIFAR10 dataset. We avoid putting embedding at the bottom of state-of-the-art models, such as VGG and ResNet, in order to show the contribution

of the visual embedding itself. Table 4.2 demonstrates that the learned filters as dictionary give a much better performance compared to the LBP dictionary.

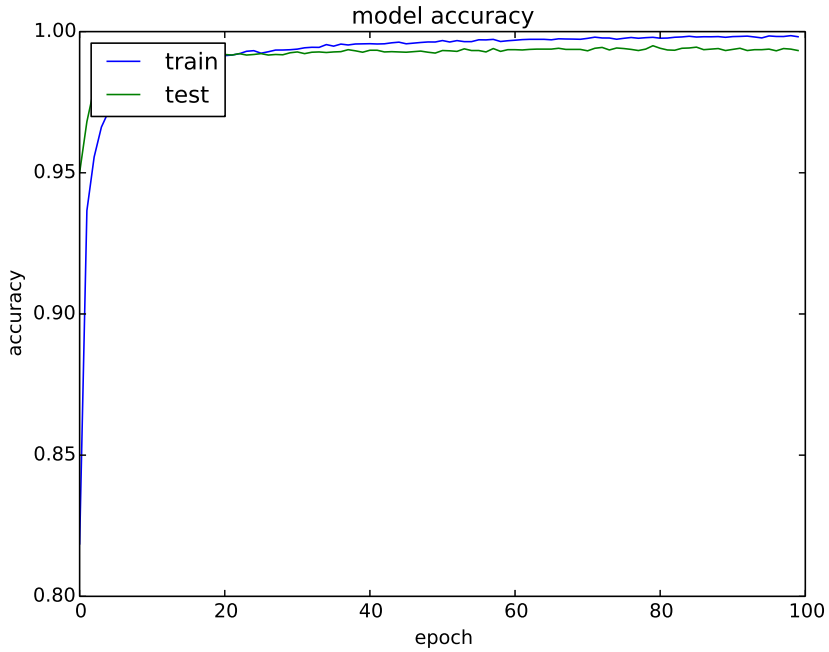

Figure 4.5: The accuracy of the proposed method with the MNIST dataset

Table 4.1: Performance of a custom convolution neural network before and after embedding application

Method	MNIST (error)	CIFAR10 (accuracy)
Before	1.0	79.01
After	**0.5**	**81.01**

Table 4.2: Performance of a LBP Dictionary and Learned Filters

Method	MNIST (error)	CIFAR10 (accuracy)
Learned Filters	**0.5**	**81.01**
LBP dictionary	0.7	61.01

Table 4.3 shows that the proposed method gives a promising performance with the MNIST and CIFAR10 datasets. Our method shows a quite good performance on the MNIST (an error of 0.5) and promising behavior on CIFAR10 (an accuracy of 0.81). The degradation of behavior in CIFAR10 is due to two factors: the difficulty

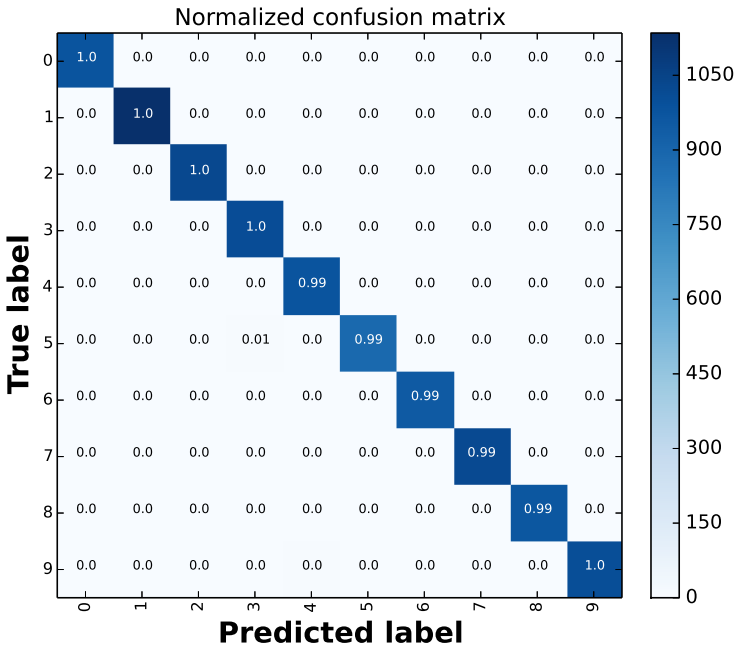

Figure 4.6: The confusion matrix of the proposed method with the MNIST dataset

of the classification problem on this dataset and the simplicity of our model. Table 4.3 also shows that the proposed method is on par with several recently published image classification methods.

Table 4.3: Comparing the proposed method with related methods

Method	MNIST (error)	CIFAR10 (accuracy)
Proposed method	0.5	81.01
Generalizing pooling (Lee et al., 2016)	0.31	92.38
ResNet	-	94.07
Deep Fried Convnets (Yang et al., 2015)	0.71	-
Adversarial Examples (Goodfellow et al.)	0.78	-
PCANet Examples (Chan et al., 2015)	0.62	78.67

4.5.4 Model Performance on Action Recognition in Still Images

To show the performance of the model in the action recognition task, we use the *UB101* dataset (Ma et al., 2017). In fact, this dataset was collected to explore the hypothesis of whether one can use action images from the web to train better CNN

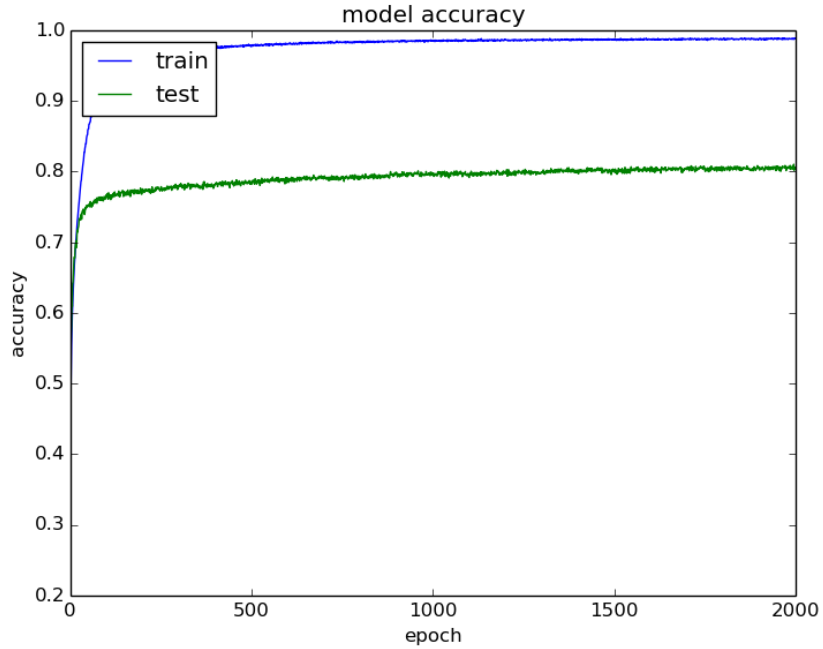

Figure 4.7: The accuracy of the proposed method with the CIFAR10 dataset

models for action recognition in videos.

Here we use different CNN models (VGG11, VGG13, VGG16 and ResNet18) as a baseline. Table 4.4 shows that models with the embedding mechanism outperform the corresponding ones without embedding. The best results are achieved with ResNet18 (an accuracy of 92 %). As shown in Figures 4.9-4.12, the proposed model takes a slightly longer time to converge. This can be explained by extra parameters of the embedding layers which need more time for training but capture more information for the decision making step.

Table 4.4: Accuracy of the proposed model with the BU101 dataset.

Model	Baseline	With Embedding
VGG11	73.409	**81.656**
VGG13	70.015	**77.793**
VGG16	56.602	**58.101**
ResNet18	91.022	**92.638**

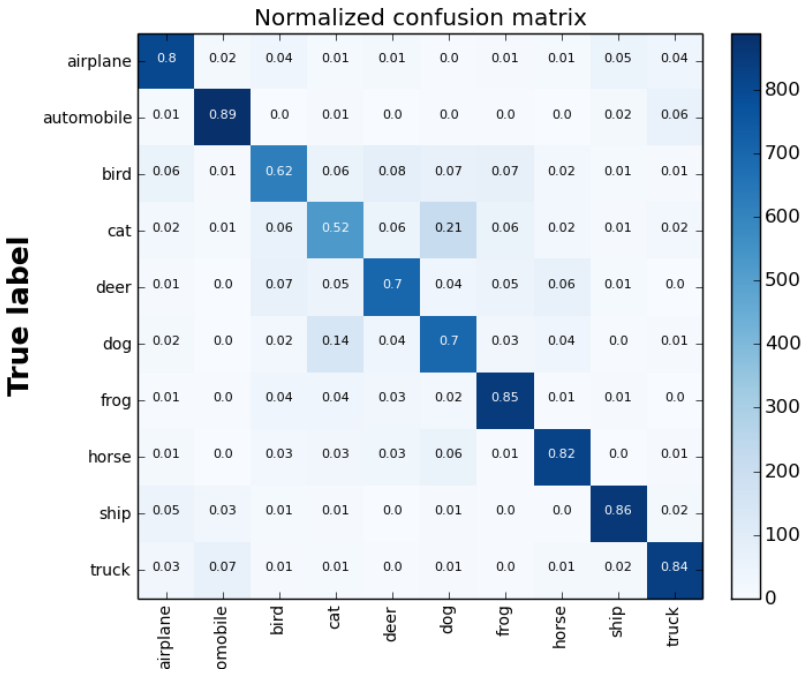

Figure 4.8: The confusion matrix of the proposed method with the CIFAR10 dataset

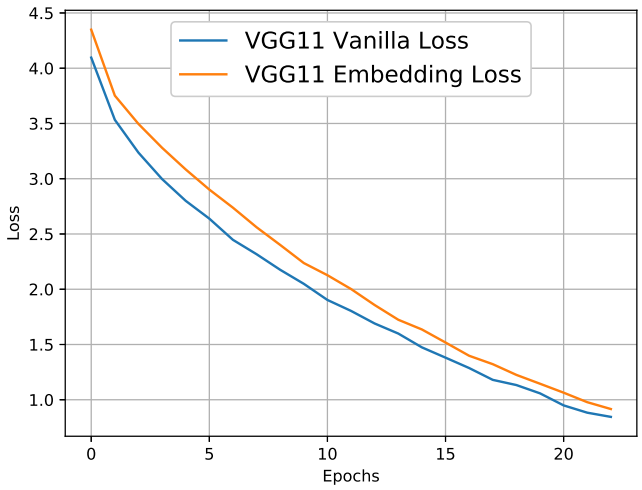

Figure 4.9: The convergence of VGG11 on the BU101 dataset.

4.6 Conclusion

In this chapter, we have presented a CNN model to embed visual words in images. We have treated images as a textual document, built visual words and embedded them to capture the spatial context surrounding them. The learned embedding performed

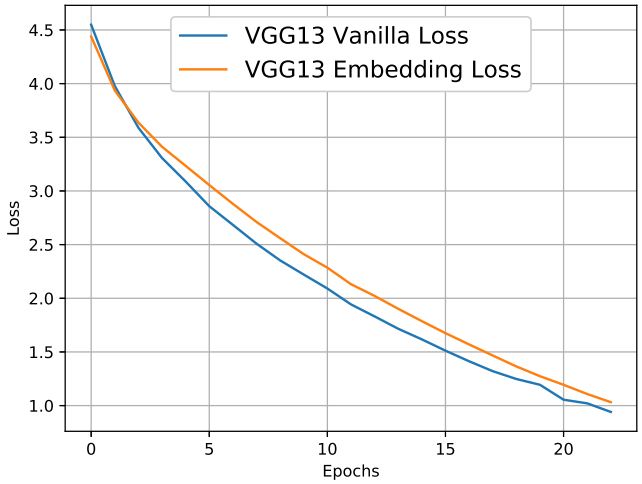

Figure 4.10: The convergence of VGG13 on the BU101 dataset.

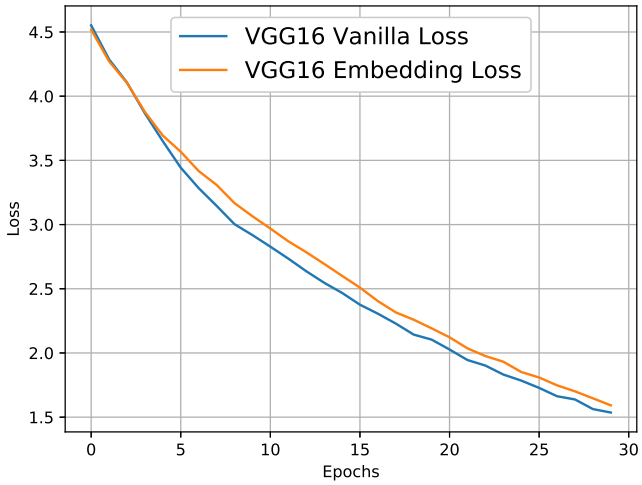

Figure 4.11: The convergence of VGG16 on the BU101 dataset.

better than the same CNN model with the original images. The main goal of our proposed method is to push a framework that shows how to build a bridge between embedding in text data and how to adapt it for visual data and a wide range of generic image and video tasks.

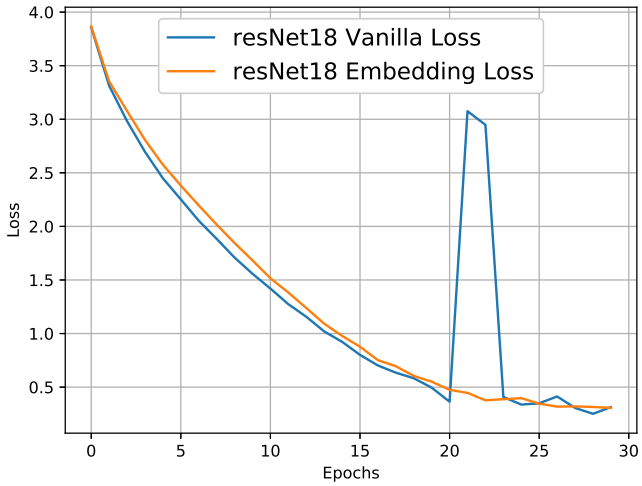

Figure 4.12: The convergence of ResNet18 on the BU101 dataset.

Chapter 5

Finger Parts Semantic Segmentation Using Multi-Scale Feature Map Aggregation of FCN

5.1 Introduction

Semantic segmentation is an important task in image recognition and understanding. It is considered as a dense classification problem. The main task in semantic segmentation is to assign a unique class to every pixel in an image. Recently, CNNs have obtained significant results in image understanding tasks. However, these approaches still exhibit obvious shortcomings when they come to dense prediction tasks, e.g., semantic segmentation. The main reason for the shortcomings is that these models include repeated steps of pooling and convolution that can cause losing much of the fine image information.

One way to overcome this shortcoming is to learn an up-sampling operation (deconvolution) to generate the feature maps of higher-resolution. Indeed, those deconvolution operations cannot recover the lost low-level visual features after the down-sampling operations. For this reason, they are unable to precisely generate

a high resolution output. Indeed, the low-level visual structure is essential for a proper prediction on the boundaries and details alike. Recently, the work proposed in (Chen et al., 2018) applied dilated convolution filters to deal with larger receptive fields without down-sampling the image. The aforementioned approach is successful, but it has two limitations. First, the dilated convolution uses a coarse sub-sampling of features, which likely causes a loss of important details. Second, it performs convolutions on a large number of detailed feature maps that have high dimensional features, which yields additional algorithmic complexity.

Several applications necessitate accurate segmentation methods, such as activity recognition, navigation, and human body parsing. One of the important applications is gesture recognition, which is the ability to understand human hand gestures by detecting and counting finger parts in a video stream or in still images. In this chapter, we attempt to deal with such small objects (i.e., finger parts). Consequently, it is essential to extract extra information from different image scales (e.g., fine to coarse features). Thus, we propose to enforce the low level layers to learn these *fine-to-coarse* features. This is achieved by feeding different resolutions of input images to the network. This will be advantageous information for solving finger parts semantic segmentation task, and it can help the model to overcome scale variations, which is considered as high-level knowledge. However, the question here is which scale will be more beneficial for extracting high-level information for an accurate segmentation of finger parts. Thus, after feeding images of different scales, our proposed model can learn to weight the generated feature maps at different scales. These feature maps are up-sampled to a unified-scale and then pooled to feed them to the next layers, as shown in Figure 5.1. The main contributions of this chapter can be summarized as follows:

- We propose a novel deep aggregation layer based on a multi-scale segmentation network which combines coarse semantic features with fine-grained low-level features in a parallel fashion to generate high-resolution semantic feature maps.

The proposed model is called *FinSeg*.

- With the lack of realistic labeled finger parts datasets, we release a dataset for finger parts semantic segmentation (called *FingerParts* dataset). As far as we know, this is the first available dataset for finger parts segmentation using a high resolution real images.

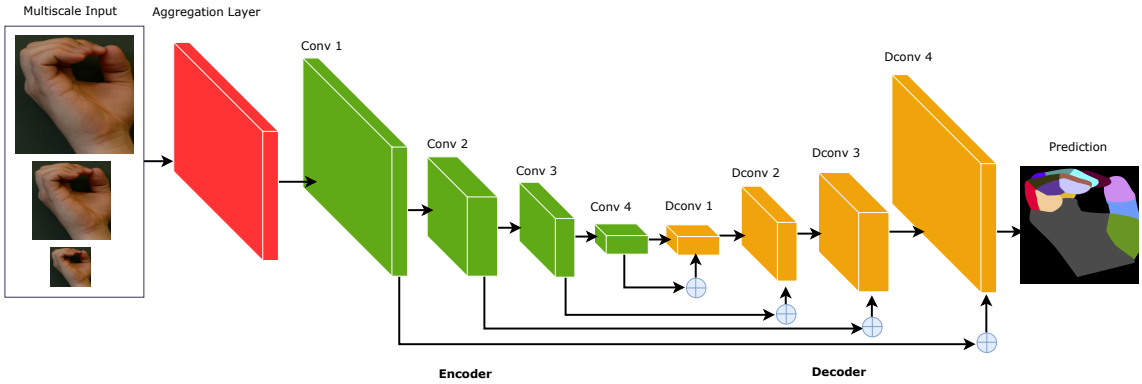

Figure 5.1: The main structure of the proposed model (FinSeg). Red block refers to the generation of the feature maps from the proposed aggregation block (shown in details in Figure 5.2).

5.2 Related Work

Recently, the most successful methods for semantic segmentation are related to deep learning models, specifically CNNs. In (Girshick et al., 2014), a region-proposal-based method has been used to estimate segmentation results. In turn, the authors of (Long et al., 2015; Chen et al., 2018) have shown the effective feature generation of CNNs and presented semantic segmentation based on fully convolutional networks (FCNs). It worth to note that FCN becomes a standard deep network for different applications, such as image restoration (Eigen et al., 2013), image super-resolution (Dong et al., 2014) and depth estimation (Eigen and Fergus, 2015; Eigen et al., 2014). However, the main limitation of networks based on the FCN architecture is the low-resolution prediction. Thus, many works proposed different techniques to tackle this limitation in order to generate high-resolution predictions. For instance, conditional random field (CRF) has been used as a post

layer for coping with this problem. This is done by generating a middle resolution score feature map and then refining boundaries using a dense CRF. In addition, an atrous convolution layer has been proposed in (Chen et al., 2014). The atrous layers are convolution filters with different rates to extract the key features of input images in different scales. In (Zheng et al., 2015), a robust end-to-end fashion parsing method is proposed by adding recurrent layers in order to improve the performance of the FCN network.

Furthermore, many deconvolution based methods have been proposed in (Badrinarayanan et al., 2015; Noh et al., 2015) to learn how to up-sample low resolution prediction by taking into account the advantage of middle layer features in the FCN network. For example, the work proposed in (Chen et al., 2014) added prediction layers to middle layers to generate prediction scores at multiple resolutions. Then the multi-resolution predictions are averaged to generate the final prediction. However, this model was trained in multi-stage style rather than end-to-end manner. In turn, other methods, such as SegNet (Badrinarayanan et al., 2015), (Sarker et al., 2018; Singh et al., 2018) and U-Net (Ronneberger et al., 2015) have used skip-connections in the decoder architecture to add information from feature maps extracted of the middle layers to the deconvolution layers.

Unlike the aforementioned methods, the proposed FinSeg model exploits the multi-scale features in the low-level layers in order to predict coarse-to-fine semantic features extracted from different resolution of an input image. In addition, unlike the standard FCN network, FinSeg uses the residual network, namely ResNet101, instead of the VGG network. In addition, we use the skip-connections of all encoder layers to add feature maps to all decoder layers as shown in Figure 5.1.

5.3 Proposed Model

We propose a deep semantic segmentation model (FinSeg) based on a new aggregation layer. FinSeg accepts an input image at different resolutions, extracts

feature maps of every scale, weights each extracted feature map, pools them and then feeds the final feature maps through long range connections to achieve a high-resolution semantic segmentation of finger parts. Below, we describe the steps of our model.

5.3.1 FinSeg Architecture

As shown in Figure 5.1, the proposed model has an encoder-decoder architecture. In general, the encoder reduces the spatial dimension through pooling layers along with summarizing the input images. In turn, the decoder recovers the object mask and spatial dimension. Following (Ronneberger et al., 2015), we use skip-connections from the encoder to the decoder in order to recover the object details in the decoder stage by transferring low level features from lower layers to the higher ones.

5.3.2 Aggregation Layer

We show the architecture of the aggregation layer in Figure 5.2. An image I is fed into the model with s scales. The input images $I_1, I_2...I_s$ are fed to a parallel sequence of convolutional layers. Shared convolution filters are applied on the images of different scales. After feeding images of different scales through the first parallel layers of the model, the resulting feature maps have different sizes. Since it is not possible to aggregate feature maps with different sizes, the multi-scale feature maps are up-sampled to the largest dimension and aggregated in one feature map. After aggregation, the resulting feature maps are then fed into the next aggregation layer and this procedure is repeated k times.

A fully connected layer (FC) of s inputs and $s \times nl$ outputs is used to learn the weights of the aggregation layer, where s is the number of scales and nl is the number of internal subsequent layers of the aggregation layer. We propose a fully automated procedure that can learn how to give a high weight for the more important scaled feature maps and suppress the others. In this study, $s = 3$ and $nl = 3$ are

the optimum values that yield the best results. The FC layer learns to weight the resulting feature maps of each scale (see Figure 5.2). A softmax function is used as an activation for each of the resulting s weights. In this work FC is initialized with an input vector $w = [1/3; 1/3; 1/3]$. It is obvious that we start by giving an equal weight to all scales.

Suppose that the final aggregated feature maps extracted at a layer l can be expressed as follows:

$$F_{l,i} = \sum_{i=1}^{s} w_{l,i} F_{l,i-1}$$

under the constraint $\sum_{i=1}^{s} w_{l,i} = 1$, where $F_{l,i-1}$ are the feature maps of the previous scale $i-1$, and $i \in 1....s$ with $l \geq 2$. The resulting $F_{l,i}$ is then fed into the convolutional layer of the next internal layer.

5.3.3 Encoder and Decoder of FinSeg

Encoder: After calculating the multi-scale aggregated feature maps, they are fed into the encoder network. The encoder consists of four convolutional layers followed by a max-pooling layer (down-sampling layers) to encode the input into feature representations at different levels, as shown in Figure 5.1. The encoder layers are adapted from the pre-trained ResNet101 network (the first four layers only).

Decoder: It consists of up-sampling and summing followed by regular convolution operations. To recover original image dimensions by up-sampling, we use bi-linear interpolation. Thus, we expand the feature map dimensions to meet the same size as the corresponding blocks of the encoder and then apply skip connections by summing the feature maps of the decoder layer with the ones generated from the corresponding encoder layers.

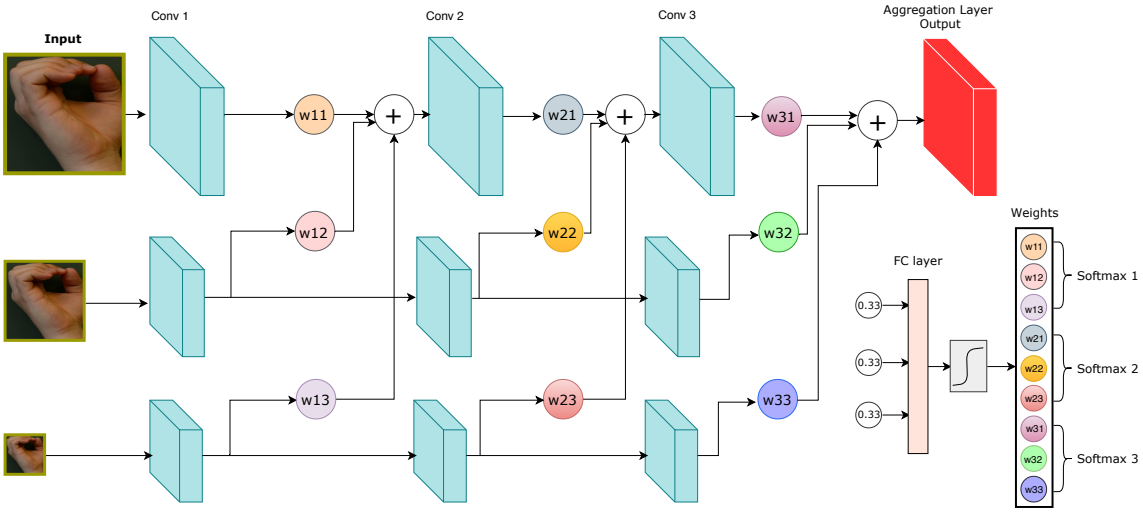

Figure 5.2: Architecture of the aggregation layer. Feature maps are aggregated at the largest scale in each internal layer.

5.4 Results and Discussion

5.4.1 FingerParts Dataset

In this chapter, we introduce a new dataset based on real hand images (called FingerParts) that can be used for the human palm and finger parts segmentation task. The FingerParts dataset contains 1,100 real images and their corresponding annotations. We have ordered human made annotations, which is in general perfect. The number of hands per image is ranging from one to three in most cases. These images can contain backside or frontal views of different hands as shown in Figure 3. Furthermore, $1,000$ images were taken from a public dataset for hand gesture recognition (Kawulok et al., 2014; Nalepa and Kawulok, 2014; Grzejszczak et al., 2016). In addition, 100 images were collected by scrapping images from *Google Image*. The results of scrapping were manually checked in order to avoid repeated and non-relevant images. The number of classes in the dataset is 17: a class for the background, 3 classes per finger ($3 \times 5 = 15$) and one for each palm. Information about key-points is also available. There are 16 key points per hand (i.e., 15 for the fingers and one for the palm). In Table 5.1, we show a comparison between

the FingerParts dataset with prior state-of-the-art datasets. It is obvious that our dataset is based on realistic images and it can be used for semantic segmentation and gesture recognition tasks.

Data Augmentation

In this study, we applied data augmentation by scaling the input images by a random value varying between 0.5 and 2.0. In addition, we applied illumination changes via a gamma correction operator with values varying from 0.5 to 3.0 with a step of 0.5. Random horizontal flipping was also applied. Furthermore, we added extra synthetic backgrounds to the input images to expose the model to more difficult tasks. In total, we have 58,380 images for training and 4935 for testing.

5.4.2 Training Procedure

In each iteration, FinSeg reads a batch of 8 images, resizes them to 512x512 and normalizes them. The normalization step consists of 3 steps: 1) the input image is divided by 255. This step makes the values of each RGB image range between 0 and 1.0, 2) centralization of image values by subtracting $[0.485, 0.456, 0.406]$ from the RGB channels is applied, and 3) the RGB channels are divided by $[0.229, 0.224, 0.225]$. Those values were used in the ImageNet dataset for the classification task and fixed (empirically) by computer vision community. An initial learning rate of 0.01 with a weight decay of 10^{-8} were used in the training procedure. SGD was chosen as an optimizer with a value of 0.99 for the momentum parameter.

Table 5.1: Quantitative comparison of our proposed dataset, FingerParts, with public datasets of the hand segmentation task.

Dataset	Number of Images	Segmentation Task	Real/Synthetic	Key Point
(Zimmermann and Brox, 2017)	41258	Yes	Synthetic	Yes
(Kawulok et al., 2014)	899	No	Real	Yes
MU HandImages (Barczak et al., 2011)	2425	No	Real	Yes
FingParts(our)	1100	Yes	Real	Yes

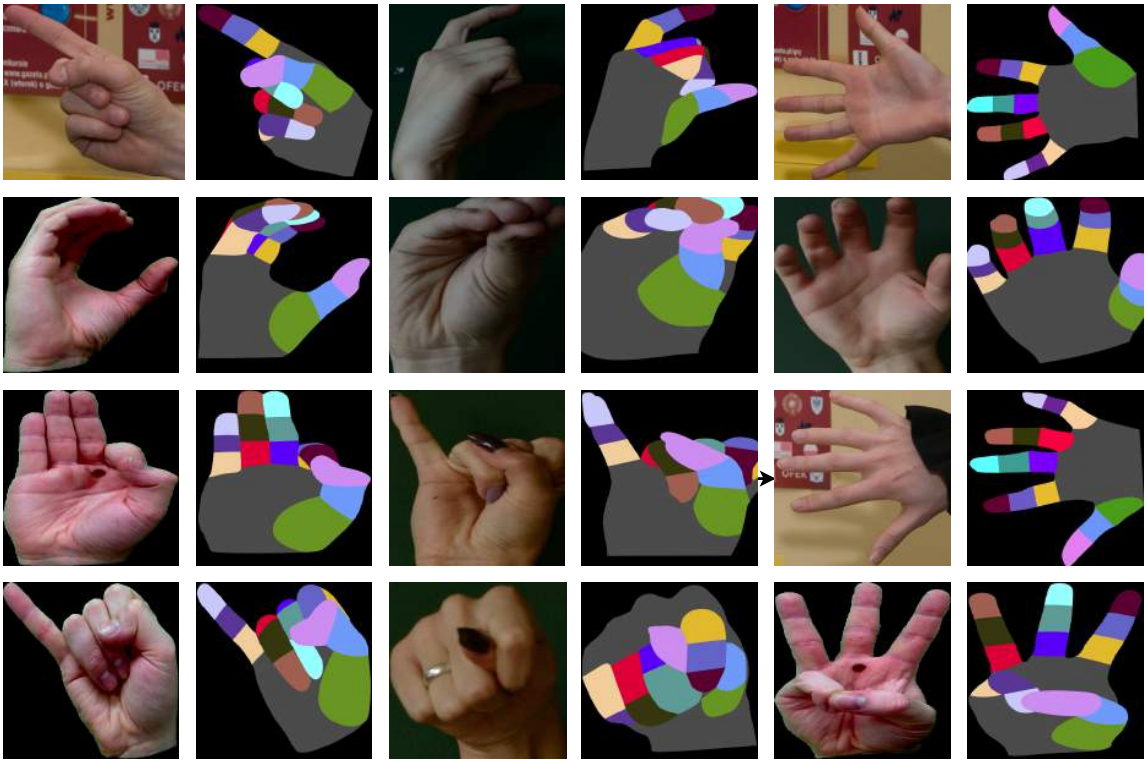

Figure 5.3: Samples of the proposed FingerParts dataset.

In this study, the cross-entropy is used as a loss function. It is defined as:

$$CE = -\sum_{i} y_i' log(y_i)$$

where y_i is the probability for predicted class i and y_i' is the true probability for that class.

Although the proposed aggregation layer adds some algorithmic complexity to the proposed model by using multi-scale layers, it converges in the same number of iterations as the standard FCN model (See Figure 5.4). However, the training process is more expensive in terms of time consumption.

5.4.3 Evaluation Metrics

In this section, we use two metrics to assess the performance of the proposed model: the *Intersection over Union* (IoU) and *pixel accuracy*. In the literature, IoU is referred to as the Jaccard index, which is basically a metric to calculate the percent

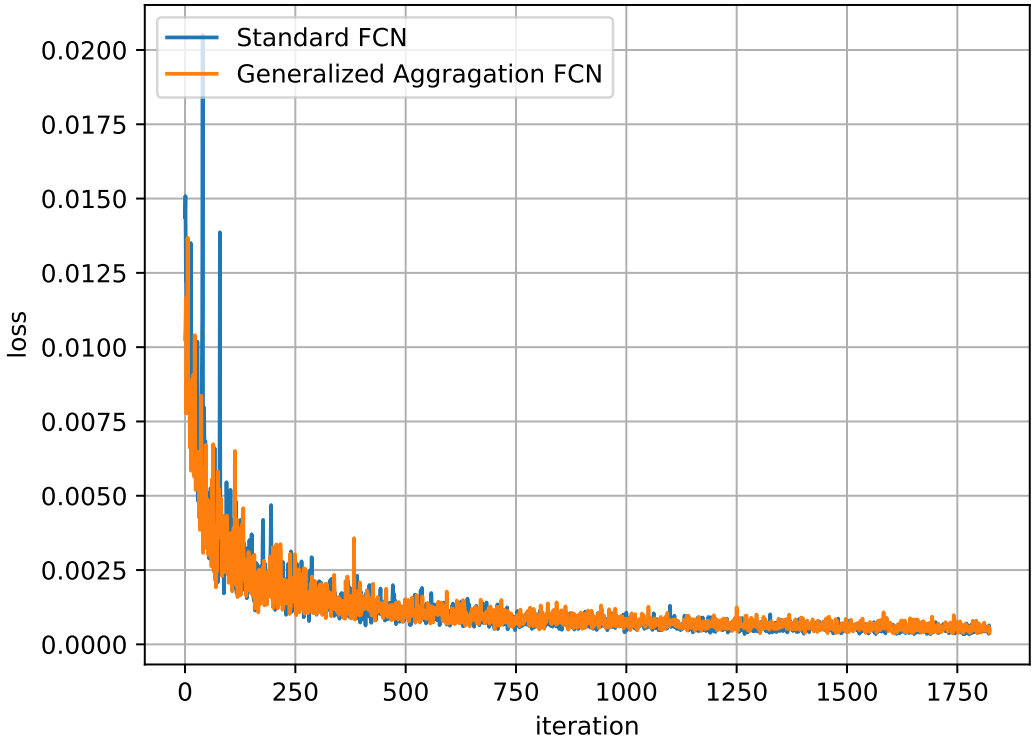

Figure 5.4: The convergence of the proposed model and the FCN model.

overlap between the target mask and the prediction output.

$$IoU = \frac{target \cap prediction}{target \cup prediction}$$

We also use the *pixel accuracy* metric. This metric reports the percent of pixels in the image which were correctly classified. The pixel accuracy is calculated for each class separately as well as globally over all classes. It can be defined as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.4.4 Results

We evaluate our approach on the proposed dataset (FingerParts). To present the usefulness of automatically selecting feature map scales, we choose the FCN model of (Chen et al., 2014) as baseline. In this section, we compare the results of three variations of the aggregation layer of the proposed model (AvrageAggr, AggrFCNSoftmax and AggrFCNRelu) with the ones of the FCN model. For the first variation of the proposed aggregation layer (AvrageAggrFCN), we apply aggregation by averaging the feature maps at different scales with the same internal layer. For the second variation (AggrFCNSoftmax), we use a softmax function as the activation function applied to the weights corresponding to the FC layer. In the third variation (AggrFCNRelu), we add a Relu after every internal convolution layer of the aggregation block.

Table 5.2 shows the experimental results of the proposed model with the proposed dataset. The baseline model, FCN, yielded an IoU of 0.58 and an accuracy of 87%. AvrageAggr gave an improvement of 4% in IoU values (only after averaging the feature maps extracted at different scales). However, for the accuracy, there was a small improvement ($< 0.5\%$).

Learning a weight for the resulting feature maps at a given scale is a generalized

form of aggregation, and it has more potential to find optimized weights. According to the results shown in Table 5.2, predicting the weights of each feature map using an FCN layer yields better results than the baseline model. An improvement of 5% with AggrFCNSoftmax was achieved. Another experiment was conducted to check the Relu function as an activation function with AggrFCNRelu. It yielded an IoU improvement of about 3%. Thus, the best results were achieved when we apply the softmax function for estimating the weight values of each scale.

Qualitative results of some of these experiments are shown in Figure 5.5. Supporting our quantitative results, the proposed model with AggrFCNSoftmax (using aggregation of FC and sofmax layers) presents visual improvements of finger parts segmentation with our dataset, compared to the FCN model and the two other variations of the proposed model (AggrFCNRelu and AvrageAggr).

Table 5.2: Performance of the three variants of the proposed model (AggrFCNSoftmax, AggrFCNRelu and AvrageAggr) and the FCN model.

Method	IoU	Accuracy
FCN (Chen et al., 2014)	0.5833	87.32
AvrageAggr	0.6231	87.64
AggrFCNSoftmax	**0.6307**	**88.13**
AggrFCNRelu	0.6151	87.91

To assess the performance of the proposed model in a particular case, we select an image randomly (see Figure 5.6) from the dataset.Then, we analyze the performance of the proposed model under different conditions: illumination changes, background changes, and image flipping. With no effects on the input image, our model achieved an IoU of 0.5515. Applying an illumination effect based on nonlinear Gamma correction with different values ($\gamma \in \{0.5, 1.0, 1.5, 2.5\}$) causes a degradation in the performance of our model (IoU drops to 0.5515). This degradation can be explained by the disappearance of small parts in Figure 5.6-(col 1-2). Another issue was investigated by changing the background and image flipping. Our experiments show that with changes in the background, IoU reduces to 0.5501 (see Figure 5.6-(cols. 3-4)), while image flipping reduces the IoU to 0.5493 (see Figure 5.6-(cols. 5-6)).

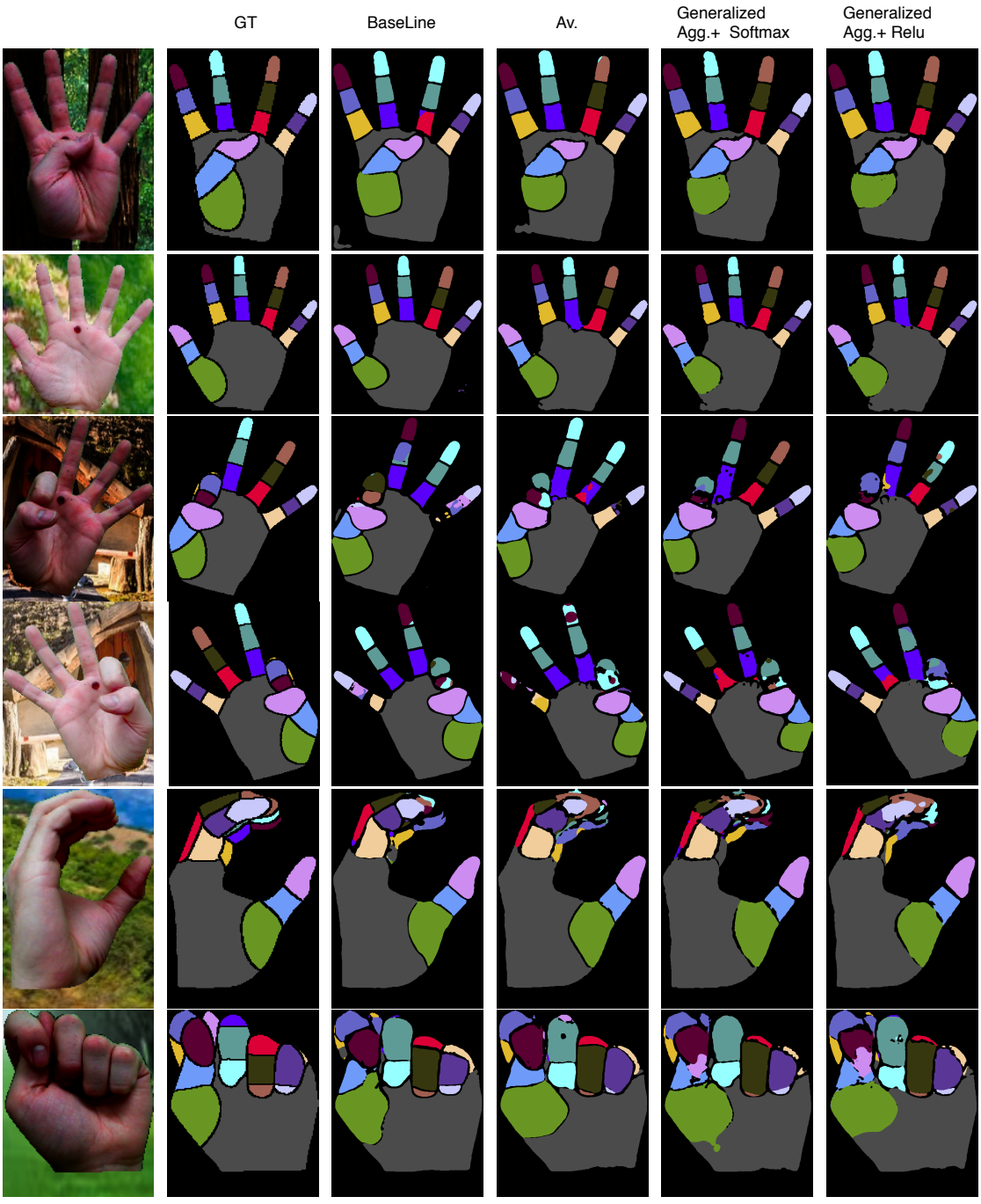

Figure 5.5: A visual comparison between the different versions of the proposed model (FinSeg) and the FCN model with the FingerParts dataset. Input images (col. 1), ground-truth (col. 2), results of the FCN model (col. 3), results of AvrageAggr (col. 4), results of AggrFCNSoftmax (col. 5), and results of AggrFCNSoftmax (col. 6).

As shown in the figures, the change of the IoU value is around 0.55 under different conditions, such as illumination changes, adding background and image flipping. Consequently, we can say that changing on the global context of the input images has an insignificant impact on the final decision of the proposed model. It is important to note that different finger parts are discriminated using their relative location to the palm more than their appearance. Thus, we can conclude that the model learns how to extract global shape information from the input images.

Figure 5.6: Analyzing the performance of the proposed model under different conditions: illumination changes (cols. 1-2), background changes (cols. 3-4), and image flipping (cols. 5-6).

5.5 Conclusions

In this chapter, we have proposed a novel deep learning based model for finger parts semantic segmentation. The proposed model is based on generating feature maps with different resolution from an input image. These feature maps are then aggregated together using automated weights estimated from a fully connected layer. The estimated weights are used to assign a high weight for the most important scaled feature maps and suppress the others. The generated feature maps are fed into an encoder-decoder network with skip-connections to predict the final segmentation mask. In addition, we have introduced a new dataset that can help solve the finger parts semantic segmentation problem. To the best of our knowledge, FingerParts is the first dataset for finger parts semantic segmentation with real high resolution images. The proposed model outperformed the standard FCN network with an improvement of 5% in terms of the IoU metric. Future work will include the use of the segmented fingers parts to improve the accuracy of gesture recognition methods and egocentric action recognition task.

Chapter 6

Concluding Remarks

6.1 Summary of Contributions

In this thesis, we have analyzed human activity recognition in videos. We have considered models usually applied in this research area to analyze streams of video data. We have extended the analysis of the existing models with different data sources, and we have exploited the knowledge resulting from other fields, such as semantic segmentation. In addition, deep learning models have been used to perform the body part segmentation task in order to obtain a completely abstracted recognition system. The thesis is divided into six chapters: introduction (chapter 1), methods based on hand-crafted features (chapter 2 and 3), methods based on deep learning (chapter 4 and 5) and summary and future work (chapter 6).

In chapter 1, we introduced the problem of human activity recognition, the motivation behind the thesis and the contributions made in this thesis.

In chapter 2, we proposed a new method to recognize human actions in videos. It uses information about trajectories extracted from the motion frames. The proposed method calculates tangent, normal, binormal and curvature vectors, and combines them with classical low-level features. The proposed approach obtains a better description of the geometrical shape of the trajectories and shows comparable results with the state-of-the-art. The performance of the proposed method is evaluated by

using the complex and large-scale action dataset HMDB51. Experimental results demonstrated that the proposed approach is comparable with several state-of-the-art methods.

In chapter 3, we have modeled the evolution of human actions for activity recognition. The proposed method consists of five steps: feature extraction, coherence analysis, weighted combination, representation learning and classification. In the feature extraction step, we used HOG, HOF and MBH feature extraction methods, and then we encoded each type of descriptors using the Fisher vector technique. In the coherence analysis step, we determined the subsequences of each video. We evaluated the effect of three similarity measures (correlation, Euclidean distance and cosine measures) on the performance of the proposed method. The cosine similarity and correlation measures gave the same recognition rates. For each subsequence, we used an aggregation method to combine the FVs of each type of descriptors extracted from each frame into one FV. To do so, we used the mean and median operators. We then used the MKL technique to produce a weighted combination of the FVs of all descriptors. We fed the FVs of all subsequences into a learning-to-rank method to produce a representation of the whole video. The representations of all videos were finally fed into a SVM classifier to discriminate between the different actions. We showed the effect of the aggregation of coherent frames on the recognition rate. We also demonstrated how useful is to process input videos as subsequences of related frames. With the HMDB51 and Hollywood databases, we showed how different methods of aggregation can change the recognition performance and how combining the aggregated FVs using the MKL method can have a positive impact on the recognition results.

In chapter 4, we have presented a CNN model to embed visual words in images. We have treated images as a textual document, built visual words and embedded them to capture the spatial context surrounding them. The learned embedding performed better than the same CNN model fed with the original images. The main goal of the

proposed method is to push a framework that shows how to build a bridge between embedding in text data and how to adapt it for visual data and a wide range of generic image and video tasks.

In chapter 5, we introduced to a new dataset that can help solve the finger parts semantic segmentation problem. To the best of our knowledge, it is the first public dataset with real high resolution images. In addition, we proposed a novel deep model for finger parts semantic segmentation. The proposed model is based on generating feature maps with different resolutions from an input image. These features maps are then aggregated together using automated weights estimated from the fully connected layer. The estimated weight is used for giving a high weight to the most important scaled feature maps and for suppressing the others. The generated feature maps are fed into an encoder-decoder network with skip-connections to predict the final segmentation. The proposed model outperformed the standard FCN network with an improvement of about 5% of the IoU metric.

6.2 Future Research Lines

The work presented in this thesis makes a contribution to the recognition of human activity in videos. We believe this is an interesting and important field of research. Several directions of future work have been identified during this work. For example:

1. The temporal coherent method can be improved by using features extracted from deep CNNs in the feature extraction step. We will also use the dynamic image technique in the representation learning step.
2. We will apply image embedding to the bottom of state-of-the-art CNN models and use it to recognize actions in videos.
3. With the increasing availability of wearable cameras, research on first-person view videos (egocentric videos) has received much attention recently. We will use deep learning models to design a system for recognizing human activity in

such videos.

4. We will use the proposed semantic segmentation model to segment the body parts, and then we will exploit the segmented image to recognize human actions in still images and videos.

References

Aiolli, F. and Donini, M. (2015). Easymkl: a scalable multiple kernel learning algorithm. *Neurocomputing*, 169:215–224.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.

Barczak, A., Reyes, N., Abastillas, M., Piccio, A., and Susnjak, T. (2011). A new 2d static hand gesture colour image dataset for asl gestures.

Ben Aoun, N., Elghazel, H., and Ben Amar, C. (2011). Graph modeling based video event detection. In *2011 International Conference on Innovations in Information Technology (IIT)*, pages 114–117. IEEE.

Bosch, A., Zisserman, A., and Munoz, X. (2007). Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Bouchrika, T., Zaied, M., Jemai, O., and Amar, C. B. (2014). Neural solutions to interact with computers by hand gesture recognition. *Multimedia Tools and Applications*, 72(3):2949–2975.

Bucak, S. S., Jin, R., and Jain, A. K. (2014). Multiple kernel learning for visual object recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1354–1369.

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. (2015). Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032.

Chatfield, K., Lempitsky, V. S., Vedaldi, A., and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, volume 2, page 8.

Chen, L., Duan, L., and Xu, D. (2013). Event recognition in videos by learning from heterogeneous web sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2666–2673.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Chen, X. and Lawrence Zitnick, C. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431.

Chéron, G., Laptev, I., and Schmid, C. (2015). P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3218–3226.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE.

Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *Computer Vision–ECCV 2006*, pages 428–441. Springer.

Dance, C., Willamowski, J., Fan, L., Bray, C., and Csurka, G. (2004). Visual categorization with bags of keypoints.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015a). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015b). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Dong, C., Loy, C. C., He, K., and Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer.

Du, Y., Wang, W., and Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118.

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658.

Eigen, D., Krishnan, D., and Fergus, R. (2013). Restoring an image taken through a window covered with dirt or rain. In *Proceedings of the IEEE international conference on computer vision*, pages 633–640.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374.

Fernando, B., Gavves, E., Oramas, J., Ghodrati, A., and Tuytelaars, T. (2016). Rank pooling for action recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Fernando, B., Gavves, E., Oramas, J. M., Ghodrati, A., and Tuytelaars, T. (2015). Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387.

Gaidon, A., Harchaoui, Z., and Schmid, C. (2012). Recognizing activities with cluster-trees of tracklets. In *BMVC 2012-British Machine Vision Conference*, pages 30–1. BMVA Press.

Gao, H., Mao, J., Zhou, J., Huang, Z., Wang, L., and Xu, W. (2015). Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in neural information processing systems*, pages 2296–2304.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies

for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Gkioxari, G. and Malik, J. (2015). Finding action tubes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. corr (2015).

Grzejszczak, T., Kawulok, M., and Galuszka, A. (2016). Hand landmarks detection and localization in color images. *Multimedia Tools and Applications*, 75(23):16363–16387.

Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Hou, R., Zamir, A. R., Sukthankar, R., and Shah, M. (2014). Damn–discriminative and mutually nearest: Exploiting pairwise category proximity for video action recognition. In *Computer Vision–ECCV 2014*, pages 721–736. Springer.

Husain, F., Dellen, B., and Torras, C. (2016). Action recognition based on efficient deep feature learning in the spatio-temporal domain. *IEEE Robotics and Automation Letters*, 1(2):984–991.

Izadinia, H. and Shah, M. (2012). Recognizing complex events using large margin joint low-level event model. In *European Conference on Computer Vision*, pages 430–444. Springer.

Jain, M., Jégou, H., and Bouthemy, P. (2013). Better exploiting motion for better action recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2555–2562. IEEE.

Jain, M., van Gemert, J. C., and Snoek, C. G. (2015). What do 15,000 object categories tell us about classifying and localizing actions? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 46–55.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Kawulok, M., Kawulok, J., Nalepa, J., and Smolka, B. (2014). Self-adaptive algorithm for segmenting skin regions. *EURASIP Journal on Advances in Signal Processing*, 2014(1):170.

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014a). Multimodal neural language models. In *International Conference on Machine Learning*, pages 595–603.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014b). Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539.

Kottur, S., Vedantam, R., Moura, J. M., and Parikh, D. (2016). Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4985–4994.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). Hmdb: a large video database for human motion recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2556–2563. IEEE.

Laptev, I. (2005). On space-time interest points. *International journal of computer vision*, 64(2-3):107–123.

Laptev, I., Marszałek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE.

Lazaridou, A., Pham, N. T., and Baroni, M. (2015). Combining language and vision with a multimodal skip-gram model. *arXiv preprint arXiv:1501.02598*.

Le, Q. V., Zou, W. Y., Yeung, S. Y., and Ng, A. Y. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE.

Lebret, R. and Collobert, R. (2013). Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*.

Lee, C.-Y., Gallagher, P. W., and Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Ma, S., Bargal, S. A., Zhang, J., Sigal, L., and Sclaroff, S. (2017). Do less and achieve more: Training cnns for action recognition utilizing action images from the web. *Pattern Recognition*, 68:334–345.

Marszalek, M., Laptev, I., and Schmid, C. (2009). Actions in context. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2srivastava2015unsupervised936. IEEE.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nalepa, J. and Kawulok, M. (2014). Fast and accurate hand shape classification. In *International Conference: Beyond Databases, Architectures and Structures*, pages 364–373. Springer.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987.

Oneata, D., Verbeek, J., and Schmid, C. (2013). Action and event recognition with fisher vectors on a compact feature set. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1817–1824.

Peng, X., Wang, L., Wang, X., and Qiao, Y. (2014). Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer.

Ponce-López, V., Escalante, H. J., Escalera, S., and Baró, X. (2015). Gesture and action recognition by evolved dynamic subgestures. In *BMVC*, pages 129–1.

Raptis, M. and Soatto, S. (2010). Tracklet descriptors for action modeling and video analysis. In *Computer Vision–ECCV 2010*, pages 577–590. Springer.

Revaud, J., Douze, M., Schmid, C., and Jégou, H. (2013). Event retrieval in large video collections with circulant temporal encoding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2459–2466.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Saleh, A., Akram, M. A. G. F., Abdel-Nasser, M., and Puig, D. Exploiting the kinematic of the trajectories of the local descriptors to improve human action recognition.

Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245.

Sarker, M., Kamal, M., Rashwan, H. A., Banu, S. F., Saleh, A., Singh, V. K., Chowdhury, F. U., Abdulwahab, S., Romani, S., Radeva, P., et al. (2018). Slsdeep: Skin lesion segmentation based on dilated residual and pyramid pooling networks. *arXiv preprint arXiv:1805.10241*.

Sekma, M., Mejdoub, M., and Amar, C. B. (2013). Human action recognition using temporal segmentation and accordion representation. In *Computer Analysis of Images and Patterns*, pages 563–570. Springer.

Sharma, S., Kiros, R., and Salakhutdinov, R. (2015). Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*.

Silberer, C., Ferrari, V., and Lapata, M. (2013). Models of semantic representation with visual attributes. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 572–582.

Silberer, C. and Lapata, M. (2014). Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 721–732.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576.

Singh, V. K., Romani, S., Rashwan, H. A., Akram, F., Pandey, N., Sarker, M., Kamal, M., Barrena, J. T., Saleh, A., Arenas, M., et al. (2018). Conditional generative adversarial and convolutional networks for x-ray breast mass segmentation and shape classification. *arXiv preprint arXiv:1805.10207*.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., and Freeman, W. T. (2005). Discovering object categories in image collections.

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE.

Sminchisescu, C., Kanaujia, A., and Metaxas, D. (2006). Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2):210–220.

Song, Y., Morency, L.-P., and Davis, R. (2013). Action recognition by hierarchical sequence summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3562–3569.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852.

Sun, C. and Nevatia, R. (2013). Active: Activity concept transitions in video event classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 913–920.

Tang, K., Fei-Fei, L., and Koller, D. (2012). Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1250–1257. IEEE.

Taylor, G. W., Fergus, R., LeCun, Y., and Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer.

Tirilly, P., Claveau, V., and Gros, P. (2008). Language modeling for bag-of-visual words image categorization. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 249–258. ACM.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2011). Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176. IEEE.

Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013a). Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79.

Wang, H. and Schmid, C. (2013). Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558. IEEE.

Wang, H., Ullah, M. M., Klaser, A., Laptev, I., and Schmid, C. (2009). Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference (BMVC)*, pages 124–1. BMVA Press.

Wang, L., Qiao, Y., and Tang, X. (2013b). Motionlets: Mid-level 3d parts for

human motion recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2674–2681. IEEE.

Wang, W.-C., Chung, P.-C., Cheng, H.-W., and Huang, C.-R. (2015). Trajectory kinematics descriptor for trajectory clustering in surveillance videos. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1198–1201. IEEE.

Wang, Y. and Mori, G. (2011). Hidden part models for human action recognition: Probabilistic versus max margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1310–1323.

Wang, Y., Song, J., Wang, L., Van Gool, L., and Hilliges, O. (2016). Two-stream sr-cnns for action recognition in videos. BMVC.

Wu, D. and Shao, L. (2014). Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–731.

Xu, R., Lu, J., Xiong, C., Yang, Z., and Corso, J. J. (2014). Improving word representations via global visual context. In *NIPS Workshop on Learning Semantics*.

Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE.

Yang, X. and Tian, Y. (2014). Action recognition using super sparse coding vector with spatio-temporal awareness. In *Computer Vision–ECCV 2014*, pages 727–741. Springer.

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., and Wang, Z.

(2015). Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483.

Yu, S., Falck, T., Daemen, A., Tranchevent, L.-C., Suykens, J. A., De Moor, B., and Moreau, Y. (2010). L 2-norm multiple kernel learning and its application to biomedical data fusion. *BMC bioinformatics*, 11(1):309.

Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537.

Zimmermann, C. and Brox, T. (2017). Learning to estimate 3d hand pose from single rgb images. Technical report, arXiv:1705.01389. https://arxiv.org/abs/1705.01389.

UNIVERSITAT
ROVIRA i VIRGILI