Agglomerative Clustering and Residual-VLAD Encoding for Human Action Recognition

Ammar Mohsin Butt 1, Muhammad Haroon Yousaf 1,2,*, Fiza Murtaza 1,3,*, Saima Nazir 1,4,*, Serestina Viriri 5,*, and Sergio A. Velastin 6,7,8

1 Department of Computer Engineering, University of Engineering and Technology, Taxila 47050, Pakistan; ammarmohsin10@live.com (A.M.B.); fizamurtaza@yahoo.com (F.M.); drsaima.nazir@fju.edu.pk (S.N.)
2 Swarm Robotics Lab, National Centre for Robotics and Automation, Taxila 47050, Pakistan
3 Department of Computer Science, Women University Swabi, Swabi 23430, Pakistan
4 Department of Software Engineering, Fatima Jinnah Women University, Rawalpindi 46000, Pakistan
5 Department of Computer Science, University of KwaZulu-Natal, Durban 4000, South Africa
6 Zebra Technologies Corporation, London SE1 9LQ, UK; sergio.velastin@ieee.org
7 School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK
8 Applied Artificial Intelligence Research Group, Department of Computer Science, University Carlos III de Madrid, 28270 Madrid, Spain
* Correspondence: haroon.yousaf@uettaxila.edu.pk (M.H.Y.); viriris@ukzn.ac.za (S.V.)

Received: 19 May 2020; Accepted: 15 June 2020; Published: 26 June 2020

Abstract: Human action recognition has gathered significant attention in recent years due to its high demand in various application domains. In this work, we propose a novel codebook generation and hybrid encoding scheme for classification of action videos. The proposed scheme develops a discriminative codebook and a hybrid feature vector by encoding the features extracted from CNNs (convolutional neural networks). We explore different CNN architectures for extracting spatio-temporal features. We employ an agglomerative clustering approach for codebook generation, which intends to combine the advantages of global and class-specific codebooks. We propose a Residual Vector of Locally Aggregated Descriptors (R-VLAD) and fuse it with locality-based coding to form a hybrid feature vector. It provides a compact representation along with high order statistics. We evaluated our work on two publicly available standard benchmark datasets HMDB-51 and UCF-101. The proposed method achieves 72.6% and 96.2% on HMDB51 and UCF101, respectively. We conclude that the proposed scheme is able to boost recognition accuracy for human action recognition.

Keywords: action recognition; bag-of-words; deep residual networks; clustering; feature encoding; classification

1. Introduction

Human action recognition [1] is one of the active areas of research in computer vision. Action recognition systems can be used in many applications such as surveillance, human–computer interactions, content-based retrieval systems and video indexing. It involves the recognition of human actions from video sequences. This task can be challenging when posed with problems such as background clutter, partial occlusion, variation of scales, appearance and lighting. An important aspect of action recognition is to find meaningful information from videos in the form of feature vectors. These feature vectors provide representations that help in discriminating different human actions. Most of the earlier research in action recognition focused on hand crafted features such as local space-time features [2], spatio-temporal features [3] and Motion Boundary Histograms (MBH) [4].
These local spatio-temporal features have shown promising results and gave direction to future research. Moreover, spatio-temporal feature detector 3D Harris [5] and 3-D scale invariant feature transform [6] developed from their 2D counterparts provide interest point based video descriptors. These techniques usually do not require human detection algorithms. However, Heng et al. [7] have stated that dense sampling methods outperform the interest point based methods based on their evaluation of several interest point based descriptors and detectors. These spatio-temporal points extract information from short temporal durations, which can result in distortion of long-term temporal actions. Hence, techniques such as feature trajectories [8,9] are used to track these interest points in order to get a robust and accurate representation. These methods have made significant progress and have shown the ability of handling some recognition challenges such as noise, lighting changes and background clutter.

Recently, deep learning algorithms [10,11] have been widely used for feature learning and classification tasks. These deep architectures build the high-level features on top of low-level features to provide an optimal representation and have proved to be quite successful in image classifications. Inspired from their success on images, researchers have proposed a variety of architectures for video classifications. Some of these architectures include spatio-temporal convolutions [10], recurrent neural networks [12] and two-stream networks [13], which are used for training two individual streams of RGB data and motion-based optical flow data. These architectures, however, often fail to register or model the motion patterns and temporal information concurrently because both streams are trained independently with no link inbetween the streams. To overcome this limitation, 3D convolutional neural network (CNN) architectures [14,15] have been proposed. These networks perform 3D convolutions and pooling, which enables the flow of temporal information across all layers. These architectures provide simultaneous feature learning for motion and appearance data, but have not proven to be effective due to their large number of parameters. More recently, Choutas et al. [16] have presented state-of-the-art results using 3D CNNs and encoding of human pose motion but requires the extraction of joint heatmaps for key points localization.

The action recognition approaches discussed earlier, provide action representations in the form of embedding or feature vectors that are either classified directly using a SoftMax layer or rely on a Bag-of-Words (BoW) pipeline. The BoW framework [17–19] has been widely used for action recognition, as it provides a simplified representation for classification tasks. It consists of the following steps: (i) feature extraction, (ii) vocabulary generation, (iii) feature encoding and classification. First, the feature vectors are extracted using a feature descriptor, for example, 3D-SIFT [6] or a ConvNet. The obtained features are quantized using a clustering algorithm [20] such as k-means or gaussian mixture models (GMMs) to form a visual vocabulary. The entries in the vocabulary are known as visual words. These visual words form the basis for feature encoding and play an important role in recognition performance. Therefore, an optimal codebook generation algorithm that can provide discriminative visual words is highly desirable. The third step involves the encoding of features for each individual video. Many researchers have worked in feature encodings [21–23] to improve the final representation. This encoded vector is classified in a final stage, which is most commonly performed by support vector machines.

In this paper, the focus is to develop a discriminative codebook and a features encoding method, which provide an all-encompassing representation for action videos. The codebook generation and encoding are the two most crucial steps for a BoW pipeline. The overall framework of the proposed method can be seen in Figure 1. In the first step, spatio-temporal features are extracted from ConvNets. The saliency of these features greatly impacts the overall recognition performance. Although hand-crafted features have had some success, such features do not generalize well to large scale video datasets [1]. Hence, we use the spatio-temporal two-stream ResNet [24] and a single stream 3D ResNet [25] architecture to extract feature descriptors. In the second step, these features are used for learning an agglomerative codebook. In the third step, the proposed agglomerative codebook
is used for feature encoding. Finally, a support vector machine classifier is used for classification. The contribution of this work is three-fold:

- First, we propose an agglomerative codebook. The aim of this codebook is to agglomerate the benefits of global codebook representation with class-specific codebook representations. The final agglomerative codebook provides discriminative codewords for the feature encoding step.
- Second, we have proposed a modified Vector of Locally Aggregated Descriptors (VLAD) vector, known as Residual-VLAD (R-VLAD), which computes the higher order statistics by finding a difference vector between the feature descriptor and mean of the nearest codeword. It requires less computational resources than VLAD due to a reduced size of the encoded vector.
- Third, to further enhance the capability of the R-VLAD vector, a hybrid feature vector is formed by fusing the locality-based descriptor with R-VLAD. The final encoded vectors are L2 normalized, which serve as the input to the classifier.

![Figure 1. Framework of proposed methodology.](image)

The rest of the paper is organized as follows: In Section 2, we explain the proposed methodology; Section 3 describes the detailed experimentation and comparisons with the state-of-the-art. Finally, the paper is concluded in Section 4.

2. Proposed Methodology

The aim of our proposed framework is to generate a discriminative codebook and provide a better representation that results in improved recognition performance. The proposed scheme for codebook generation and feature encoding is given in Figure 1. As discussed earlier, ConvNets have outperformed the traditional hand-crafted features, therefore, we extract features using ConvNets. Details of feature extraction are given in Section 2.1. These deep features are used for codebook generation and feature encoding. Sections 2.2 and 2.3 describe the codebook learning and the proposed feature encoding methodology, respectively.

2.1. Deep Feature Extraction

Deep CNN architectures when trained from scratch tend to overfit on small datasets like HMDB51 and UCF101. Therefore, we have used architectures that are pre-trained using a larger dataset, for example, ImageNet, Kinetics, etc. The video data contains the information related to appearance and flow of appearance, hence, we have used two architectures: a spatio-temporal 2D residual network [24] and a 3D architecture ResNeXt-101 [25]. The ResNet architectures have been successfully
used in image classifications [24,25]. These architectures offer shortcut connections for bypassing the layers in the network. This allows the gradients to flow from later layers to earlier ones, which can be beneficial in training very deep architectures.

2.1.1. Spatio-Temporal ResNet (ST-ResNet)

The ST-ResNet [24] is a two-stream convolutional Resnet with a spatial and a flow stream. Both motion and appearance streams use a ResNet-50 model that is pre-trained on the ImageNet dataset. We fine-tuned the model for video datasets, i.e., HMDB51 and UCF101. One of the drawbacks of two-stream architectures is the registration of motion and appearance information. To overcome this issue, spatio-temporal cues are learned by the network at different scales. This is achieved by introducing residual connections inbetween the motion and appearance streams. There are several ways of connecting these streams just like various shortcut connections. The 2048-dimensional features for both streams are extracted after the pooling layer of the last convolutional layer.

2.1.2. ResNeXt-101

Resnext-101 is a 3D convolutional neural network with 101 layers. It was originally proposed by Xie et al. [26], and Hara et al. [25] have further extended the work to 3D convolutions. These networks introduce the concept of cardinality, which denotes the size of the set of transformations. This cardinality proved to be effective compared to wider and deeper networks. The convolutional groups are introduced in ResNeXt block, these groups convert the feature maps into smaller groups. The network used in the study has a cardinality of 32. We have used a Kinetics pre-trained network, provided by Hara et al. [25].

2.2. Agglomerative Codebook Generation

The codebook formation is an essential step in any BoW pipeline as discussed in Section 1. The codebook is a compact representation of the dominant features from the training data. This codebook is used for encoding of both training and testing data. Figure 2 depicts the proposed agglomerative clustering approach (see Algorithm 1).

![Figure 2. Agglomerative clustering for codebook generation.](image-url)
Algorithm 1: Agglomerative Clustering

**Input:** A feature set \( F \) for \( N \) number of classes

**Output:** Agglomerative codebook \( C \)

**Step 1:** Build global codebook \( C_G \)

\[ C_G \leftarrow \text{Cluster}(F, \text{number of clusters}) \]

**Step 2:** Build \( C_{cs} \) for class \( i = 1; i \leq N; i++ \) do

\[ C_i \leftarrow \text{Cluster}(F_i, \text{number of clusters}) \]

\[ C_{cs} \leftarrow C_{cs} \cup C_i \]

end

**Step 3:** Clustering of \( C_G \) and \( C_{cs} \) to get agglomerative codebook

\[ C \leftarrow \text{Cluster}(C_{cs} \cup C_G, \text{number of clusters}) \]

A feature set \( F \) is extracted using a CNN architecture \([24,25]\) for \( N \) number of action classes, i.e., \( F = [F_1 + F_2 + \ldots, F_N] \). A global codebook \( C_G \) is generated by clustering all these features. Features having similar patterns will be grouped into the same clusters. The k-means++ algorithm \([27]\) is used as a cluster initialization method. Cluster optimization is performed by Lloyd’s method \([28]\). These methods provide good results as compared to the simpler k-means. The same setting is used in class-specific and final codebook generation. The global codebook with \( M \) number of clusters is represented as: \( C_G = c_m | m = 1, \ldots, M \). The global codebook can cause discriminative features to be grouped into the same cluster as unlabelled data is used. This codebook provides a global simplified representation of the entire training data. The second phase of codebook generation is learning a class-specific codebook. In a class-specific codebook, the features from each class are clustered separately. For a given class \( i \), features \( F_i = [f_1 + f_2 + \ldots, f_n] \) having \( n \) number of features are extracted, the class-specific codebook is generated given by \( c_i = c_k | k = 1, \ldots, K \) where \( K \) represents the total number of clusters for class \( i \). The final class-specific codebook is formed by appending the class-specific codebooks for all classes given as \( C_{cs} = c_j | j = 1, \ldots, (K_i \times N) \), where \( N \) represents the total number of classes. The codewords in this codebook are class-specific, meaning that equal contributions are made in the codebook from each individual class of features. However, this codebook lacks the global representation, which is present in the global codebook. The final phase of clustering involves the clustering of codewords. To combine the advantage of global representation along with class-specific representation, we introduce the idea of clustering these global and class-specific codewords. k-means clustering is applied on centroids of the global \( C_G \) and the class-specific \( C_{cs} \) codebooks. This step helps in finding linkages between the codewords having similar information in the global and the class-specific codebooks. The nearest codewords are combined, which results in a better representation containing information of global as well as class-specific visual words. The combination of two different codebooks proves to be beneficial for encoding of features. The size of the codebooks is chosen very cautiously, as it affects the overall performance of the BoW approach. Large sizes can cause processing overheads, redundant codewords and less generalizability. A small size can cause distinct features to be clustered under the same codeword, which can reduce the discriminative power of the codebook. Different codebook sizes have been tested experimentally with details given in Section 3.3.1.

2.3. Feature Encoding

The proposed feature encoding provides a simplified R-VLAD encoding and fusion-based representation for deep features, discussed in Sections 2.3.4 and 2.3.5, respectively. First, the features are extracted using a ConvNet, discussed in Section 2.1. These features are then used to construct the agglomerative codebook. The codebook thus formed is utilized for further encoding stage. The sections below first review the most commonly used encoding schemes: Vector Quantization (VQ), VLAD and Locality based coding, then the proposed method is discussed in Section 2.3.4.
2.3.1. Vector Quantization (VQ)

Let $X$ be a video descriptor with dimension $D$ with a total of $N$ number of descriptors extracted from a video, i.e., $X = [x_1, x_2, \ldots x_N] \in \mathbb{R}^{D \times N}$ and a codebook is given with $M$ number of entries having the same dimensions as the descriptor, i.e., $C = [c_1, c_2, \ldots, c_M] \in \mathbb{R}^{D \times M}$. The vector quantization forms a histogram of frequency counts of nearest codewords. The size of the final encoded vector is equal to $M$. The final encoded vector $v$ is given by:

$$v(i) = \arg\min_{v} \sum_{i=1}^{N} \|x_i - Cv_i\|^2 \quad \text{s.t.} \|v\|_0 = 1 \quad (1)$$

where $v = [v_1, v_2, \ldots v_N]$ contains the set of codes and $\|v\|_0 = 1$ means each code $v_i$ contains only one non-zero element. Only one codeword is voted for each descriptor $x$. The vector quantization is also known as hard quantization as it causes information loss due to large quantization errors.

2.3.2. VLAD Super Vector

VLAD [23] is among the most widely used super vector methods, which has shown promising results in a number of tasks. It can be regarded as the simplified version of fisher vector encoding. For a given video descriptor $X$, i.e., $X = [x_1, x_2, \ldots x_N] \in \mathbb{R}^{D \times N}$ where $N$ is the number of descriptions with $D$ dimension and a codebook $C$ of $M$ entries, i.e., $C = [c_1, c_2, \ldots, c_M] \in \mathbb{R}^{D \times M}$. The final encoded vector is given by:

$$V = [0, \ldots, (x - c_1), \ldots 0], \quad \text{s.t.} \ i = \arg\min_{j} \|x - c_i\|^2 \quad (2)$$

The final representation is a higher dimensional super vector, which contains zeros and the difference vector computed between nearest centroid and descriptor. The final representation has dimension of $M \times D$.

2.3.3. Locality Constrained Coding

For the success of many recognition applications, data locality proves to be an important factor. In [29], the authors have shown that sparse information is not enough to handle large occlusions. Local information is useful in the reconstruction of samples for query data by using nearest codewords, whereas in sparse coding the farthest entries from codebook are used for reconstruction, which is undesirable. Contrary to sparse coding, locality-based coding ensures that codewords are similar for a given class of query data. This inspires us to use a locality-based features encoding approach to achieve good representation. Locality constrained linear coding is a fast implementation for local coordinate coding. The query features are encoded by projecting features into a local coordinate system. Consider a video descriptor $X$, i.e., $X = [x_1, x_2, \ldots x_N] \in \mathbb{R}^{D \times N}$ where $N$ represents the number of descriptions having $D$ dimension and a k-means clustered codebook $C$ of $M$ entries, i.e., $C = [c_1, c_2, \ldots, c_M] \in \mathbb{R}^{D \times M}$. The final descriptor for each local descriptor is an $M$-dimensional codeword. The final descriptor is given by:

$$\psi(s) = \|s \odot s\|^2_{2} \quad \text{s.t.} \ 1^T s = 1 \quad (3)$$

where $\epsilon = \exp\left(\frac{\text{dist}(s, D)}{\sigma}\right) \sigma$ is a constant that regulates weight decay speed for locality adaptor. $1^T s = 1$ is a constraint that is a requirement for the final encoded vector.

2.3.4. Proposed Residual-VLAD (R-VLAD)

This section describes the proposed encoding method inspired by VQ and VLAD encoding schemes. The goal is to find a simple yet meaningful representation for given spatio-temporal descriptors of a video. The first step is to find the nearest codeword $c$ for each $f_i$ in $F = [f_1, f_2, \ldots f_N] \in \mathbb{R}^{D \times N}$ of a video. These nearest codeword assignments are similar to VQ
but instead of hard assignment into a histogram, a residual vector similar to VLAD is obtained by finding the difference between the descriptor \( f_i \) and the mean of the assigned codeword. This helps in describing the distribution of feature vectors with respect to the center. The mean of each assigned codeword is a unique value which is an important statistical information. The final encoded vector is given by:

\[
V_i,j = f_j - \text{mean}(c)
\]  

(4)

where \( c \) represents the nearest codeword and \( f_j \) represents the \( j \)th component of the feature descriptor \( F \). This difference vector brings complementary information related to the assigned codewords, which proves to be beneficial for the classifier. The final dimension of the encoded vector is the same as that of the feature descriptor represented by \( D \) (features dimensionality). Due to this reduced size we call it the Residual VLAD or R-VLAD feature vector. In comparison, VLAD has a dimension of \( M \times D \) (number of clusters \( \times \) the features dimensionality).

### 2.3.5. Proposed Hybrid Feature Vector

For a fusion based representation, the LLC [29] codeword is used along with the modified VLAD vector. The final form (see Algorithm 2) of the hybrid feature vector (HFV) is the concatenation of the LLC and the modified VLAD represented by:

\[
HFV = \text{Concat}[\psi(s), f_j - \text{mean}(c)]
\]

(5)

where \( \psi(s) \) is the LLC descriptor, \( c \) represents the nearest codeword and \( f_j \) represents the \( j \)th component of the feature descriptor \( F \). This residual vector and locality-based code is the final vector with a dimension of \( D + M \) (Feature Dimensionality + Number of Clusters). This vector contains the aggregation of higher order statistics while maintaining a lower order dimension unlike VLAD which generates a \( k \times D \) dimension super vector. To suppress the large values within this vector we employ intra-normalization and L2 normalize each residual vector.

### Algorithm 2: Hybrid Feature Vector (HFV) Encoding

**Input:** Features \( F = [f_1, f_2, \ldots, f_N] \in \mathbb{R}^{D \times N} \) of a video and Agglomerative codebook \( C \)

**Output:** Encoded vector \( HFV \)

**Step 1:** Computing R-VLAD

\[
V_i,j = f_j - \text{mean}(c)
\]

where \( c \) represents the nearest codeword.

**Step 2:** Computing LLC

\[
\psi(s) = \|s \odot s\|_2^2
\]

**Step 3:** Fusing R-VLAD and LLC descriptors

\[
HFV = \text{Concat}[\psi(s), f_j - \text{mean}(c)]
\]

### 3. Experimental Results and Discussion

In this section, experimental evaluation of the proposed methodology is presented. First, the publicly available datasets that have been used for experimentation are discussed in Section 3.1. The implementation details for the proposed scheme are summarized in Section 3.2. Next, the performance evaluation is given in Section 3.3 and finally a comparison with the state-of-the-art is provided in Section 3.5.

#### 3.1. Datasets

Experimentation has been carried out on two widely used action recognition datasets: HMDB51 and UCF101. HMDB51 was developed by Serre Lab of Brown University [30]. The videos are taken from YouTube and movies. It contains 51 different categories with 6800 videos approximately.
Each action category contains at least 100 videos with reasonable discrepancy. The actions in the dataset can be grouped into five categories: facial movements (chew, laugh, talk, smile), body movements (clap hands, walk, wave, jump, pull up, cartwheel, etc.), facial movement with objects (smoke, eat, drink), body movements with objects (brush hair, sword, pour, catch, ride bike, etc.) and human interactions (fencing, kick, shake hands, etc.). Figure 3 shows some images from these action categories. The HMDB51 is considered as one of the most challenging datasets due to poor quality videos and substantial camera motions.

Figure 3. HMDB51 dataset from left to right Push-up, Chew, Cartwheel, Pour, Sword-Exercise.

UCF101 is an extension of the UCF50 dataset [31], developed by the University of Central Florida. The videos are taken from YouTube. The dataset contains 101 realistic human action categories. With more than 13,000 videos in the dataset, the UCF101 offers a large diversity in terms of actions. The action classes in the dataset are divided into five types: body motion, sports, human to object interaction, human–human interaction and playing of musical instruments. Each action category is divided into 25 groups each having four to seven clips. The mean clip duration is around 7.21 s. The resolution of videos is $320 \times 240$ with a frame rate of 25 fps. Some sample frames from these videos are given in Figure 4.

Figure 4. Sample frames from the UCF101 dataset.

For these two datasets three training and testing splits are provided, which are used as a standard evaluation scheme. The average accuracy of these training-testing splits is used for performance evaluation.

3.2. Implementation Details

The experiments were performed on a workstation with an Intel Xeon 2687, 64GB RAM and an Nvidia P5000 GPU. For feature extraction two ConvNets have been used, as discussed in a previous section. The spatio-temporal ResNet architecture was taken from Feichtenhofer et al. [24]. It was pre-trained on ImageNet and fine-tuned on our datasets. This has residual connections in between motion and appearance streams to incorporate the registration of streams. RGB and flow features are extracted from the pooling layer after the conv_5 block from both streams. The ResNeXt model provided by Hara et al. [25] was pre-trained on the large video dataset Kinetics. The top layers of network conv_5 and FC layer were fine-tuned for HMDB51 and UCF101 datasets. The model used for fine-tuning was trained on 64 frame input. The input size for ResNeXt-101 is $3 \times 64 \times 112 \times 112$. The videos are first converted into frames using the ffmpeg-python library. The frames are reshaped to $112 \times 112$ pixels. Videos having less than 64 frames are looped to complete frames as necessary. Details regarding training parameters and network architecture can be seen in [25]. The training data used in training these networks were taken from the standard training splits provided with the datasets [30,31]. For the HMDB51 dataset, each training split includes around 70 videos from each
action class and for the UCF-101 dataset 18 groups out of 25 and used in training and the rest are used as testing data. The networks were fine-tuned for each of the three splits for both datasets.

A discriminative codebook is learned using a three-step clustering approach, as discussed earlier. The k-means++ cluster initialization and Lloyd’s optimization scheme were set with default parameters as they performed best in preliminary experimentation. First, a global codebook is learned considering features from all the classes. The number of clusters is varied to achieve the best results. Secondly, a class-specific codebook is formed using the features from individual classes. Finally, clusters from both codebooks are clustered together to form a final discriminative codebook. The final step in the pipeline is feature encoding. The features from each video are encoded using the proposed hybrid feature vector. These encoded vectors are L2 normalized. The final normalized vectors act as the training data for classifier, using a one-vs-all SVM classifier.

3.3. Performance Evaluation

This section discusses the parameters that are tuned to achieve optimal performance. The codebook size is a notable parameter for optimal codebook generation. In the encoding part, we have varied the number of neighbors and beta parameters for locality constraint coding. Finally, we explain the impact of normalization.

3.3.1. Codebook Size

The codebook size affects the performance of the BoW pipeline. In the proposed scheme, three codebooks are formed. The global codebook using features from all the classes, a class-specific codebook using data from individual classes and the final codebook by combining the prior codebooks. Different combinations for both codebooks have been tried, for example, varying the size of the class-specific codebook while keeping the global codebook size fixed and vice versa. The size of the final codebook was generally kept as half of the combined codebooks. The number of clusters for the global codebook were varied from 50 to 500, while fixing the number of clusters for the class-specific codebook to two clusters per class as suggested by empirical evaluations. Figure 5 shows the results for HMDB51 split1 using features from ST-ResNet and ResNeXt-101. The best results are produced when the global codebook size is 200 for ResNeXt and 150 for ST-ResNet.

3.3.2. The Encoding Parameters and Normalization Scheme

In the proposed hybrid feature vector, the number of nearest neighbors and value of beta($\beta$) (used for regularization) parameter for locality constraint coding have been evaluated empirically while fixing the size of the codebook as mentioned previously. The number of neighbors were varied between 1 to 10 and the best results were achieved at five nearest neighbors and the value of $\beta$ was set to 0.2. Normalization influences the recognition performance of the proposed scheme as learned during the experimentation. Five different normalization schemes have been evaluated, namely: L1, L2, Power Normalization (PN), PN+L1, PN+L2. The $\alpha$ parameter for Power Normalization has been set to 0.5. The results for these normalization methods are given in Figure 6.

The results are computed on HMDB51-split 1, where L1 and PN performed poorly. The best results are computed using L2 normalization. The combination of PN+L2 provides almost similar results but we rely only on L2 to avoid extra processing.
3.4. Comparison with Other Encoding Schemes

The comparison of the proposed R-VLAD vector and Hybrid Feature Vector (HFV) encoding method with other encoding schemes is presented in Table 1. The parameters for the proposed encoding scheme are fixed as discussed previously. The vector quantization follows the hard assignment and a histogram of codeword occurrences is formed. The results show that these assignments can cause loss of information, which ultimately affects recognition performance. The soft assignment (SA) [32] is another encoding method, which considers all the codewords for final voting. The $\beta$ parameter is used for controlling of softness. We set $\beta$ as 1 in evaluation. The improvement is seen as compared to VQ as more information is encoded related to other codewords. VLAD and fisher vectors are used in standard form with a codebook size of 256. R-VLAD vector outperforms the standard VLAD representation despite being a lower dimensional (more compact) representation. However, the R-VLAD residual
vector contains high order statistics, as discussed earlier. Fisher vector encoding performed poorly, which shows super vectors do not necessarily improve accuracies. Best accuracy is obtained on both datasets with the proposed HFV, which combines the R-VLAD with the LLC descriptor.

| Encoding Schemes | ST-ResNet | ResNeXt-101 | ST-ResNet | ResNeXt-101 |
|------------------|-----------|-------------|-----------|-------------|
| VQ               | 90.3      | 92.5        | 59.6      | 65.2        |
| SA [32]          | 90.2      | 91.3        | 62.4      | 67.4        |
| VLAD [23]        | 92.9      | 93.8        | 64.5      | 68.1        |
| Fisher Vector [33]| 91.0      | 90.1        | 64.2      | 64.3        |
| Proposed R-VLAD  | 93.7      | 94.0        | 65.0      | 69.8        |
| Proposed HFV     | 94.3      | 96.2        | 67.2      | 72.6        |

### 3.5. Comparison with State-of-the-Art

A comparison with relevant state-of-the-art is given in Table 2. The results are computed by calculating average accuracy of three splits of UCF101 and HMDB51 datasets. First, the proposed scheme is compared with two-stream CNNs. Most of these methods combine the encoding schemes with improved dense trajectories. The proposed scheme performs better compared to these methods except in the case of iDT + VLMPF [34]. These results are computed by fusion of five different features, namely, SCN, TCN, C3D, HMG and iDT. However, the computational complexity involved in extracting these deep features and dense trajectories makes this method ineffective. Subsequently, the results for 3D ConvNets are compared. The results suggest that 3D CNNs trained on large video datasets outperform the ones trained on large image datasets. The proposed scheme improves the performance of existing 3D ConvNets as opposed to the addition of more expensive dense trajectories. Furthermore, the proposed approach can be extended to two-stream 3D Resnets architectures given that computational complexity is not a matter of concern. These two streams consist of RGB and flow data, which proves to be effective in terms of higher accuracies. As the results suggest, the results can be improved with a traditional BoW pipeline, but at the cost of additional computational overheads.

| Features         | Method                               | UCF101 | HMDB51 |
|------------------|--------------------------------------|--------|--------|
| Two-Stream CNN   | iDT + ST-VLAD [35]                   | 91.5   | 67.6   |
|                  | LTC [11]                             | 92.7   | 67.2   |
|                  | iDT + ActionVLAD [36]                | 93.6   | 69.8   |
|                  | iDT + VLMPF [34]                     | 94.3   | 73.1   |
|                  | Two-Stream [10]                      | 88.0   | 59.4   |
|                  | ST-ResNet [24]                       | 93.4   | 66.4   |
|                  | Proposed HFV-Backbone-ResNet         | 94.3   | 67.2   |
| Single-Stream 3D CNN | C3D [15]                           | 82.3   | 51.6   |
|                   | Res3D [37]                           | 85.8   | 54.9   |
|                   | $F_{STCN} (SC1 fusion) [38]$         | 88.1   | 59.1   |
|                   | P3D [39]                             | 88.6   | -      |
|                   | iDT + C3D AdaScan [40]               | 93.2   | 66.9   |
|                   | TSN [41]                             | 94.2   | 69.4   |
|                   | ResNeXt-101 [25]                     | 94.5   | 70.2   |
|                   | Proposed HFV-ResNeXt-101             | 96.2   | 72.6   |
| Two-Stream 3D CNN | I3D + PoTion [16]                    | 98.2   | 80.9   |
|                   | I3D + IDT Hallucination [42]         | -      | 82.4   |
4. Conclusions

In this work, we worked on two important aspects of Bag-of-Words framework: optimal codebook generation and features encoding. Two deep ConvNet architectures were explored for features extraction, a two-stream 2D Resnet and a 3D Resnet. We have proposed an effective agglomerative clustering approach for codebook formation. This approach provided the most discriminative codewords as many empirical tests were conducted on global and class-specific codebooks. The limitation of this approach is that it takes three steps for generating the final codebook. The R-VLAD feature encoding is proposed which offers compact representation. The R-VLAD is combined with locality-based descriptor to form a hybrid feature vector, it offers an inclusive spatio-temporal representation. The 3D Resnets pre-trained on large video datasets have shown competitive performance compared to the 2D Resnets.

While this study was focused on 2D Resnet and a single stream 3D Resnet, in the future, two-stream 3D architectures will be explored for learning better video representations. In addition, determining the fusion weights for individual streams and improvement of computational efficiency by embedding the encoding in end-to-end trainable ConvNet.

Author Contributions: Conceptualization, A.M.B., M.H.Y., S.N.; methodology, A.M.B., F.M.; software, A.M.B.; validation, A.M.B., F.M.; investigation, A.M.B., F.M., S.N.; resources, M.H.Y.; writing—original draft preparation, A.M.B.; writing—review and editing, F.M., S.N., M.H.Y., S.V., S.A.V.; visualization, S.N., S.V.; supervision, F.M., M.H.Y., S.A.V.; project administration, M.H.Y., S.V.; funding acquisition, M.H.Y., S.V. All authors have read and agreed to the published version of the manuscript.

Funding: The research work is funded by Higher Education Commission, Pakistan for Swarm Robotics Lab (Sub-Lab: Computer Vision) under National Centre for Robotics and Automation (NCRA).

Acknowledgments: Authors acknowledge continuous support from Centre of Computer Vision Research (C²VR) and Directorate of Advanced Studies, Research and Technological Development (ASRTD), UET Taxila.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- **BOW**: Bag-of-Words
- **C3D**: Convolutional 3D Network
- **ConvNet**: Convolutional Neural Network
- **HFV**: Hybrid Feature Vector
- **LLC**: Locality Constrained Coding
- **MBH**: Motion Boundary Histograms
- **R-VLAD**: Residual-Vector of Locally Aggregated Descriptors
- **SCN**: Spatial Convolutional Network
- **TCN**: Temporal Convolutional Network
- **VQ**: Vector Quantization

References

1. Kong, Y.; Fu, Y. Human Action Recognition and Prediction: A Survey. *arXiv* **2018**, *arXiv*:1806.11230.
2. Laptev, I.; Marszałek, M.; Schmid, C.; Rozenfeld, B. Learning realistic human actions from movies. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
3. Dollár, P.; Rabaud, V.; Cottrell, G.; Belongie, S. Behavior recognition via sparse spatio-temporal features. In Proceedings of the 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Beijing, China, 15–16 October 2005; pp. 65–72.
4. Dalal, N.; Triggs, B.; Schmid, C. Human detection using oriented histograms of flow and appearance. In Proceedings of the European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; pp. 428–441.
5. Laptev, I. On space-time interest points. *Int. J. Comput. Vis.* **2005**, *64*, 107–123. [CrossRef]
6. Scovanner, P.; Ali, S.; Shah, M. A 3-dimensional sift descriptor and its application to action recognition. In Proceedings of the 15th ACM international conference on Multimedia, Bavaria, Germany, 24–29 September 2007; pp. 357–360.

7. Wang, H.; Ullah, M.M.; Klaser, A.; Laptev, I.; Schmid, C. Evaluation of local spatio-temporal features for action recognition. In Proceedings of the British Machine Learning Conference, London, UK, 7–10 September 2009.

8. Wang, H.; Schmid, C. Action recognition with improved trajectories. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 3–6 December 2013; pp. 3551–3558.

9. Raptis, M.; Soatto, S. Tracklet descriptors for action modeling and video analysis. In Proceedings of the European Conference on Computer Vision, Crete, Greece, 5–11 September 2010; pp. 577–590.

10. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Li, F.F. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.

11. Varol, G.; Laptev, I.; Schmid, C. Long-term temporal convolutions for action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 2017, 40, 1510–1517. [CrossRef] [PubMed]

12. Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2625–2634.

13. Simonyan, K.; Zisserman, A. Two-stream convolutional networks for action recognition in videos. *Adv. Neural Inf. Process. Syst.* 2014, 27, 568–576.

14. Ji, S.; Xu, W.; Yang, M.; Yu, K. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 2012, 35, 221–231. [CrossRef] [PubMed]

15. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Las Condes, Chile, 11–18 December 2015; pp. 4489–4497.

16. Choutas, V.; Weinzaepfel, P.; Revaud, J.; Schmid, C. Potion: Pose motion representation for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7024–7033.

17. Karaman, S.; Seidenari, L.; Bagdanov, A.; Del Bimbo, A. L1-regularized logistic regression stacking and transductive crf smoothing for action recognition in video. In Proceedings of the ICCV Workshop on Action Recognition With a Large Number of Classes, Sydney, Australia, 7 December 2013; Volume 13, pp. 14–20.

18. Peng, X.; Wang, L.; Cai, Z.; Qiao, Y.; Peng, Q. Hybrid super vector with improved dense trajectories for action recognition. *ICCV Workshops 2013*, 13, 109–125.

19. Uijlings, J.R.; Duta, I.C.; Rostamzadeh, N.; Sebe, N. Realtime video classification using dense hof/hog. In Proceedings of the International Conference on Multimedia Retrieval, Glasgow, Scotland, 1–4 April 2014; pp. 145–152.

20. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin, Germany, 2006.

21. Zhou, X.; Yu, K.; Zhang, T.; Huang, T.S. Image classification using super-vector coding of local image descriptors. In Proceedings of the European Conference on Computer Vision, Crete, Greece, 5–11 September 2010; pp. 141–154.

22. Huang, Y.; Huang, K.; Yu, Y.; Tan, T. Salient coding for image classification. *CVPR 2011*, 1753–1760.

23. Jégou, H.; Douze, M.; Schmid, C.; Pérez, P. Aggregating local descriptors into a compact image representation. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3304–3311.

24. Feichtenhofer, C.; Pinz, A.; Wildes, R.P. Spatiotemporal multiplier networks for video action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4768–4777.

25. Hara, K.; Kataoka, H.; Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6546–6555.

26. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500.
27. Arthur, D.; Vassilvitskii, S. k-means++: the advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, New Orleans, LA, USA, 7–9 January 2007; pp. 1027–1035.

28. Lloyd, S. Least squares quantization in PCM. IEEE Trans. Inf. Theory 1982, 28, 129–137. [CrossRef]

29. Wang, J.; Yang, J.; Yu, K.; Lv, F.; Huang, T.; Gong, Y. Locality-constrained linear coding for image classification. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3360–3367.

30. Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. HMDB: A large video database for human motion recognition. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 7 November 2011; pp. 2556–2563.

31. Soomro, K.; Zamir, A.R.; Shah, M. UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv 2012, arXiv:1212.0402.

32. Smeulders, A.; Gemert, J.; Veenman, C.; Geusebroek, J. Visual word ambiguity. IEEE Trans. Pattern Anal. Mach. Intell. 2010, 32, 1271–1283.

33. Perronnin, F.; Sánchez, J.; Mensink, T. Improving the Fisher kernel for large-scale image classification. In Proceedings of the European Conference on Computer Vision, Crete, Greece, 5–11 September 2010; pp. 143–156.

34. Duta, I.C.; Ionescu, B.; Aizawa, K.; Sebe, N. Spatio-temporal vector of locally max pooled features for action recognition in videos. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3205–3214.

35. Duta, I.C.; Ionescu, B.; Aizawa, K.; Sebe, N. Spatio-temporal vlod encoding for human action recognition in videos. In Proceedings of the International Conference on Multimedia Modeling, Reykjavik, Iceland, 4–6 January 2017; Volume 64, pp. 365–378.

36. Girdhar, R.; Ramanan, D.; Gupta, A.; Sivic, J.; Russell, B. Actionvlod: Learning spatio-temporal aggregation for action classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 971–980.

37. Tran, D.; Ray, J.; Shou, Z.; Chang, S.; Paluri, M. Convnet architecture search for spatiotemporal feature learning. arXiv 2017, arXiv:1708.05038.

38. Sun, L.; Jia, K.; Yeung, D.; Shi, B.E. Human action recognition using factorized spatio-temporal convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4597–4605.

39. Laptev, I. Learning spatio-temporal representation with pseudo-3d residual networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5533–5541.

40. Kar, A.; Rai, N.; Sikka, K.; Sharma, G. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3376–3385.

41. Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 20–36.

42. Wang, L.; Koniusz, P.; Huynh, D.Q. Hallucinating iidt descriptors and i3d optical flow features for action recognition with cnns. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 8698–8708.