Learning Representations for Image and Video Understanding

Bedanta Kumar Das

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Indian Institute of Technology Hyderabad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY HYDERABAD

June 2019

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

[illegible]

(Signature)

(Bedanta Kumar Das)

CS17MTECH11009

(Roll No.)

Approval Sheet

This Thesis entitled Learning Representations for Image and Video Understanding by Bedanta Kumar Das is approved for the degree of Master of Technology from IIT Hyderabad

[illegible]

(Maunendra Sankar Desarkar) Examiner
Dept. of Computer Science and Engineering
IITH

[illegible]

(Dr. Manish Singh) Examiner
Dept. Computer Science and Engineering
IITH

C. Krishna Mohan

(Dr. C. Krishna Mohan) Adviser
Dept. of Computer Science and Engineering
IITH

[illegible]

(Dr. Bheemarjuna Reddy T) Chairman
Dept. of Computer Science and Engineering
IITH

Acknowledgements

First of all, I would like to extend my gratitude towards Prof. C. Krishna Mohan for his continuous guidance, encouragement, and strong support during my M.Tech.

I would also like to thank Dinesh for his valuable advice and Vishnu, Sanjay, Rajat, Nandan, Nazil, and Jaya for their continuous support and contributions towards my master's research.

I am thankful to all my teachers in IIT Hyderabad for their support and valuable guidance.

I would also like to thank my seniors, juniors, and batch mates for their support.

Dedication

To my parents

Abstract

Data representation is the core of all machine learning algorithms, and their performance depends mostly on the features or representations of the input on which any machine learning algorithms can be applied. Hence, to deploy a machine learning model, a considerable amount of time is invested in designing data preprocessing pipelines and data transformations that help in efficient representation of the data so that machine learning algorithms can be applied on them. Such feature engineering is costly yet essential and accentuates the shortcomings and pitfalls of machine learning algorithms, i.e., their lack of ability to extract abstract information from the input data. Feature engineering is a way to leverage human ingenuity and prior knowledge to compensate for the shortcomings of the machine learning algorithms. Hence, to make the machine learning models easily deployable and application ready, it is highly desirable to curtail the dependence of learning algorithms on engineered features so that the construction of novel algorithms can be much faster. This thesis proposes a novel approach to represent a video as a graph for action recognition and localization using only class label information.

In addition to that, this thesis also proposes a novel subspace attention mechanism to learn to capture long-range inter-dependencies in visual data. This attention mechanism is implemented as a block which can be incorporated into any backbone convolution neural network.

Contents

Chapter 1

Introduction

The performance of machine learning algorithms relies mostly on how the data is represented. These representations capture different explanatory features or variations in the data. Although a prior domain knowledge can be used to extract informative features in the data, learning using a generic prior can also be used. By learning to represent the data, a machine learning algorithm can discover discriminative information from the input data automatically. Rerepresentation learning replaces the need for manual feature engineering and enables the machine learning algorithms to learn representations explicitly required for a particular task. Learning to represent input data is mainly motivated by the fact that, for tasks such as classification, machine learning algorithms require input that can be mathematically represented and are computationally convenient. However, for input data such as images and videos, hand-crafted features are more susceptible to errors. A potential solution is to learn to represent data, which allows the machine learning algorithm to discover abstract information automatically from the input data without relying on explicit algorithms.

1.1 Feature Extraction Techniques

In recent research, a significant amount of literature is dedicated to develop methods for better representations of the input data for various cognitive tasks. Features can be extracted based on local relationships in the data or based on global relations. The techniques involved in extracting features can be broadly classified into two categories:

- Hand Crafted Feature Extraction.
- Learned Representation.

1.1.1 Hand Crafted Feature Extraction

Most of the computer vision algorithms depend on the extraction of local features from images and videos. Therefore, much of the research in visual computing focuses on developing algorithms to discover, characterize, and improve features that can be extracted from images or videos. By hand-crafted features, we usually refer to the set of manually designed algorithms based on prior knowledge. As a result, the features extracted using hand-crafted feature extraction techniques are static, and depending on the task, we need to choose the best feature extraction algorithm appropriately. Some of the local feature extraction techniques are listed as follows:

- **Harris detector** is a widely used corner detector, which is used to detect corners in an image. A corner in an image can be considered as an intersection of two edges in two different directions. Corners are the important features in an image, and they are generally termed as interest points which are translation, rotation and illumination invariant. Although corners are scant in an image, they contain the most important features in restoring image information, and they can be used to minimize the amount of processed data for motion tracking, image stitching, image representation and other related computer vision areas.

- **Scale invariant feature transform(SIFT)** is also a highly used hand-crafted feature descriptors, and are usually used in combination with other detectors. From a set of reference images, SIFT keypoints of objects are first extracted and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors.

- **Speeded Up Robust Features (SURF)** scale invariant blob-like features. It uses the determinant of Hessian to select location of the features, as well as, to determine the characteristic scale. The calculation of the determinant requires the Gaussian second order partial derivatives L_{xx}, L_{xy}, and L_{yy} which is approximated with box filters.

1.1.2 Learned Representation

Unlike hand-crafted feature descriptors, learned representations allow the machine learning algorithms to extract discriminative feature from data automatically. One of the most popular representation learning techniques is deep learning. Deep learning models learn to extract features automatically from images or videos. Deep learning facilitates new insight into solving problems involving visual data, and many attempts have been made to use deep learning methods to extract features from raw RGB, depth, and skeleton data. Many new network architectures have been proposed for various tasks such as image classification and segmentation, object detection, human action recognition, etc. Deep learning models, especially, convolution neural networks(CNNs), have been very successful in the area of image and video understanding.

Although deep CNNs have shown exceptional performance in various image and video understanding tasks, however, they have their limitations as well. Many new network architectures have been proposed to increase learning capacity, and alleviate certain optimization issues which lead to an increase in performance for various vision tasks. Recent research in deep representation learning enables machine learning algorithms to learn local as well as global relations in visual data by learning to capture long-range inter-dependencies in visual data. This automatic learning of features from data is what makes deep learning powerful.

Chapter 2

Related Work

The main objective of better data representation is to improve the performance of the machine learning algorithms for various cognition tasks such as image classification, object detection, action recognition and localization, image segmentation, etc. In this section, we will discuss the previous representation techniques and methods proposed for two specific tasks:

- Action recognition and localization in videos
- Image classification.

2.1 Action Recognition and Localization

In recent research, the problem of action localization and action recognition have been mostly dealt with traditional computer vision approaches [1] as well as deep learning approaches [2, 3]. The existing techniques for action localization can be classified into four categories based on the level of supervision, namely, fully supervised, unsupervised, semi-supervised, and weakly supervised. We review the most recent approaches from each type in this section.

The fully supervised action localization approaches show a high performance but require two levels of annotations: 1) video-level i.e. class labels of the action in the video, and 2) frame-level i.e. bounding-box for each action instance over all the frames in a video. These approaches generate bounding box proposals in each frame individually and then classify them into an action category. Gkioxari *et al.* [4] and Weinzapfel *et al.* [5], generate bounding box proposals using the object detection method [6], and then classify them into action classes using a two-stream CNN. In [2, 7, 8, 3], recent CNN based object detectors like SSD [9] and Faster-RCNN [10] are used for bounding box generation. However, they do not consider temporal information across the frame as they apply object detection on each video frame as individually. Recently, Zhu et al. [11] propose a spatio-temporal convolutional regression network for the generation of the bounding box for action localization in each frame. All these approaches, first detect the bounding boxes in each frame and later linked them together to generate the action tube proposals in the videos using dynamic programming based on the Viterbi alogrithm [4].

The unsupervised action localization approaches do not require any ground-truth information but perform low in comparison to supervised approaches. The traditional sliding-window sampling is an unsupervised approach but computationally very slow as it needs to search an exponentially large space. Gemert *et al.* [12] and Chen *et al.* [13] perform action localization using clustering of motion trajectories effectively. More efficient methods [14, 15, 16] generate proposals for action localization by sampling bounding boxes based on the super-voxels.

The semi-supervised action localization approaches do not require ground-truth information for all the bounding box in video frames. Instead, they work with mixed samples of with and without ground-truth information. Yu and Yuan [17] use a person detector and motion scores to generate bounding boxes and compute their action score then linked them using maximum set convergence problem. Klaser *et al.* [18] uses an upper-body detector per frame and tracks them by optical flow feature points to generate spatio-temporal action tubes.

The semi-supervised approaches leverage the benefit of both the approaches but the performance still depends on the ratio of labeled vs unlabeled information used. In our proposed approach for action localization, we use a weakly supervised approach. The weakly supervised action localization approaches require video-level annotations but do not need bounding box or pixel-wise annotations in the video frames [19, 17].

2.2 Image Classification

Image classification is one of the most studied cognition tasks in computer vision. Several approaches have been proposed in the past to represent images for better classification. Convolution neural networks(CNNs) have been successfully applied in several image classification tasks. However, in some cases such as fine-grained classification, vanilla CNNs fail to perform well. Lin et al.[20] proposed a bilinear model by learning to capture second order statistics in the data. In recent research, attention mechanism is gaining popularity to capture global relationships in visual data. [21] proposes a novel double attention block to gather and distribute features to capture global relationships in the data.

In guided cognition tasks, attention is the ability to focus more on some particular parts of the input data and ignore the rest of the data. In other words, attention is a way to assign different importance to different parts of the input data, and this enables the networks to pick the salient information from the noisy data. Attention can be broadly categorized into two types viz. implicit attention and explicit attention. During the training, CNNs naturally learn a form of implicit attention where neurons in CNNs respond differently to different parts of input data, i.e., neurons respond strongly to some part of input data than others [22, 23, 24]. Explicit attention has been employed in neural networks to achieve various goals such as to facilitate the interpretation and visualization of models, to enable scalability and allow variable size input (e.g., fixed-size glimpse for any input size of the input image), to enable the computational efficiency, i.e., to reduce the FLOPs, etc.

Several recent works have incorporated (explicit) attention into neural networks for vision related tasks. Xu et al. [25] and Chen et al. [26] use attention mechanism to generate captions from the images. Wang et al. [27] propose residual attention network by stacking multiple attention modules to generate attention-aware features. Recently, several attempts have been made to incorporate (explicit) attention mechanism to improve the computational efficiency of CNNs and bolster the performance of CNNs in various vision tasks. Wang et al. [28] proposed non-local operation, a generalized form of self-attention [29], to circumvent the limitation of convolution operators, which occurs due to its locality, and boost the performance in video recognition tasks. Chen et al. [21] introduced double attention block, which captures the long-range dependencies by gathering and distributing features in the entire feature space. Woo et al. [30] introduced "convolution block attention module" (CBAM) which exploits both spatial and channel-wise feature correlation using attention mechanism to improve the representational power of CNNs. SE-Net [31] re-calibrates the feature maps using squeeze and excitation operations.

Chapter 3

Weakly Supervised Spatiotemporal Action Localization in Videos

We propose an efficient way of representing an action in a video as a graph for spatiotemporal action recognition and localization in untrimmed videos using class label information only. We construct a graph by extracting key interest points from a video clip using Harris 3-D corner detector and assign class label same as that of the video. The local points extracted from multiple classes have significant intra-class variability and inter-class similarity. To curb the intra-class variability and inter-class similarity, we apply a deep multiple instance ranking framework on the local action descriptors. To classify a graph of local actions into one of the action classes, we use a support vector machine along with a graph kernel. The graph can then be considered as a 3-D volume, which represents the localized action in a video. The experimental results show that the proposed approach outperforms the state-of-the-art methods on the three benchmark datasets, namely, UCF101-24, UCF-Sports, and JHMDB-21.

3.1 Background

In computer vision research, action recognition and localization is an interesting problem due to its numerous real-world applications such as video surveillance [32], video captioning [33], video-based human-computer interaction [3], etc. These applications require the model to recognize and localize an action in an untrimmed video correctly. However, most of the existing methods for action recognition and localization work offline and perform well on trimmed videos [1, 3]. In addition to that, existing models highly depend on guided supervision and requires and bounding box information for a particular action in a video. However, obtaining datasets with high-level annotation is very costly. Apart from that, action recognition and localization suffer from challenges such as occlusion, low video quality, camera angle, etc.

In recent research, deep learning methods like convolution neural networks (CNNs) have shown remarkable performance in a variety of tasks such as scene understanding, action recognition, and localization, and object detection [34, 2, 4, 35, 8, 3, 5, 10]. Deep learning has been successfully applied to action recognition and localization tasks because of their ability to learn discriminative features automatically from images/videos. Since the success of these methods depends on the

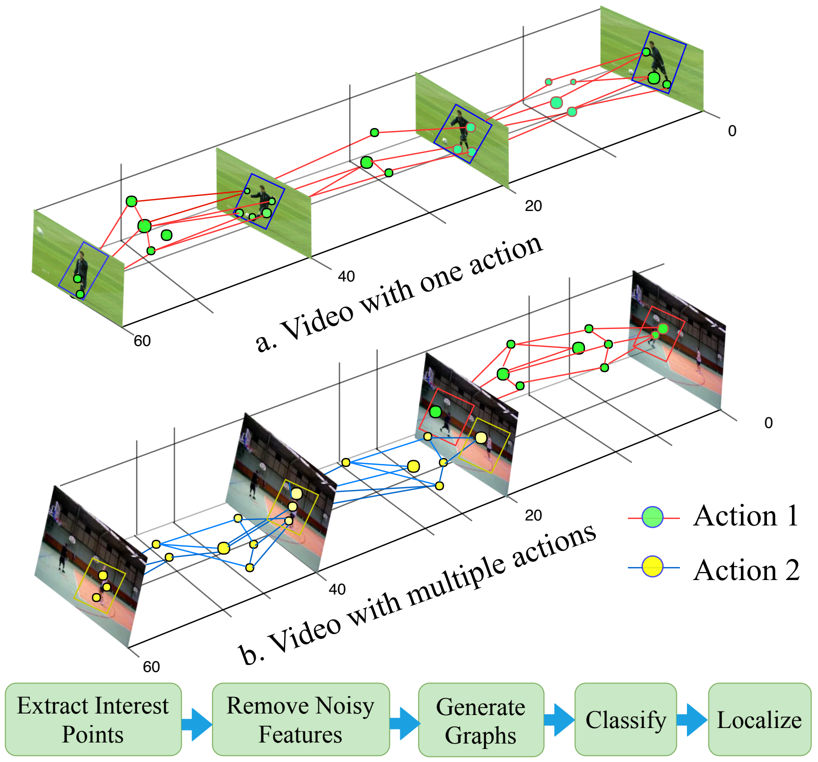


Figure 3.1: The proposed approach constructs a spatio-temporal graph of local actions and generates non-cubical shapes for efficient localization of the action.

availability of accurate spatiotemporal annotations, they cannot leverage the benefit of rapid growth in unstructured video data having only clip level annotations. Also, most of the existing deep learning methods [4, 5, 3] treat each frame independently thus ignore temporal continuity (i.e., motion information) in a video clip which is crucial for the action recognition and localization.

To address the above issues, we propose a novel unified approach for action recognition and localization using only class label information as shown in Fig. 3.1. For each video, we construct a 3D graph in which nodes describe the interest points in space and time, and edges between nodes represent the correlation between two local descriptors for a particular action. Since our graph representation of the action videos contains the structural relationship among the regions of interest, it helps in better recognition of the action in the video clips. Each node of the graph is associated with a score indicating the degree to which its appearance and motion support the action class of interest. We discard all those regions of interest which mislead or do not contribute to the class of interest. This also reduces the number of nodes in the graph which ultimately results in a fast method as the time taken during graph generation and kernel computation also decrease. The score and the descriptor of a node is determined using a weakly supervised deep regression network. The network is trained from the appearance and motion information around the nodes using deep multiple instance learning framework to curb the inter-action similarity and intra-action variability. The contributions of the proposed approach are:

- Use of weakly supervised deep Multiple Instance Learning (MIL) ranking model for joint representation learning and scoring of the salient spatio-temporal regions using clip level annotations only.

- The use of graph based action recognition method using a max-margin graph classifier makes our approach robust to slight variations in the videos which may occur due to viewpoint variations, illumination change, a difference in the video quality, etc.

- Localization of the recognized action as non-cubical or arbitrary shaped-portion of the video on the basis of nodes (i.e. local interest points) in the graph.

3.2 Proposed methodology

We approach the problem of action recognition and localization in four steps, as shown in Fig. 3.2. Firstly (§ 3.2.1), we extract local descriptors from each video clip at key interest points using Harris-3D corner detector, and by extracting appearance and motion information around the interest points, we obtain a joint representation of the action clip. Secondly (§ 3.2.2), we train a deep multi-instance learning framework separately for each action class to curb the intra-class variability and inter-class similarity which arise due to the weak annotation of the interest points. In the third step (§ 3.2.3), we generate a set of undirected labeled graphs of local actions using the local action descriptors and their scores that are generated in the previous step. The nodes in a graph are the interest points in space and time with action scores. An edge represents a possible correlation between the two local actions. Finally (§ 3.2.4), using a max-margin binary classifier we decide whether the graph belongs to the positive action class or some other class. In this case, we use a support vector machine trained using a graph kernel as our classifier. The positively classified graph can be viewed as a 3D volume, which represents the action in the video, and bounding boxes are generated based on the nodes in the graph(§ 3.2.4).

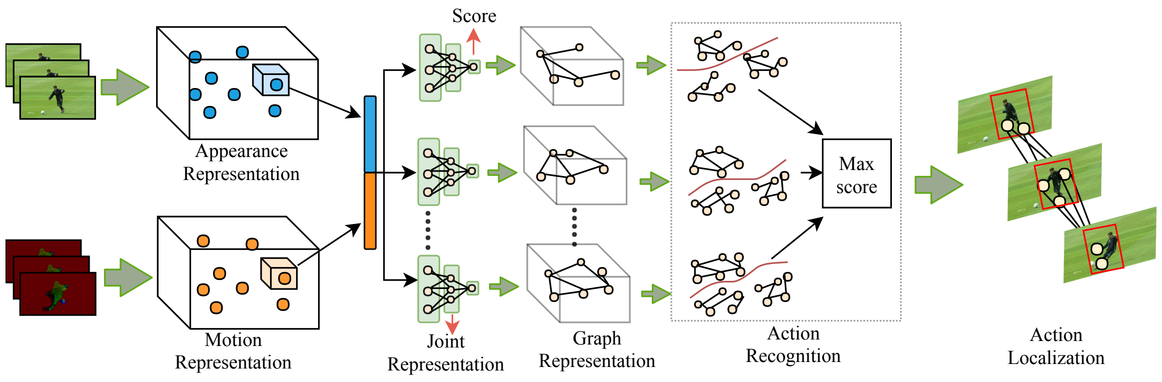


Figure 3.2: Proposed approach for weakly supervised spatio-temporal action recognition and localization in untrimmed video clips using graph of local actions. The local actions at 3D salient points along with their feature vector represented using appearance and motion information are used to construct a graph of interaction's between them.

3.2.1 Extracting Interest Points

The main objective of extracting interest points in a video is to locate key demeanour in a video, which denote a significant change in spatial, as well as temporal dimension. Similar to space-time interest points [36], the spatio-temporal interest points are regions in $f : R^2 \times R \rightarrow R$ space with eigenvalues λ_1, λ_2, and λ_3 of a second-moment matrix $\boldsymbol{\mu}$ and Harris-3D corner function (H) can be used to detect these points in spatial as well as temporal domain by combining both the determinant and the trace of μ and can be written as:

$$H = det(\boldsymbol{\mu}) - k\ trace^3(\boldsymbol{\mu}) \tag{3.1}$$

$$H = \lambda_1\lambda_2\lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3, \tag{3.2}$$

Here, k is a constant and $\boldsymbol{\mu}$ is a matrix of 3×3 dimension consisting of first order spatial and temporal derivatives averaged using a gaussian weighting function $g(.; \sigma_i^2, \tau_i^2)$ and scaled with spatial variance(σ_i^2) and temporal variance(τ_i^2). $\boldsymbol{\mu}$ can be computed as

$$\boldsymbol{\mu} = g(.; \sigma_i^2, \tau_i^2) * \begin{pmatrix} L_x^2 & L_xL_y & L_xL_z \\ L_xL_y & L_y^2 & L_yL_z \\ L_xL_z & L_yL_z & L_z^2 \end{pmatrix}, \tag{3.3}$$

Here, L_x, L_y, and L_z are the partial derivatives with respect to x, y, and z of the linear scale-space representation $L : R^2 \times R \times R_+^2 \rightarrow R$ of f constructed by convolution of f with a Gaussian kernel $g(.; \sigma_l^2, \tau_l^2)$ with local scales σ_l^2 (spatial variance) and τ_l^2 (temporal variance). L can be computed as:

$$L(.; \sigma_l^2, \tau_l^2) = g(.; \sigma_l^2, \tau_l^2) * f(.). \tag{3.4}$$

Solving equation (3.2) results into a set of m 3D interest point $P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m\}$. To represent the local action at each 3D interest point $\mathbf{p}(x, y, z)$, a feature vector is computed by considering a smaller size volume around it called spatio temporal video volume (STVV) similar to [37], with different scales in both space and time jointly from both appearance and motion (optical flow) modalities. STVV at a spatio-tamporal point can be seen in Fig. 3.3

3.2.2 Weakly Supervised Feature Extraction at Interest Points

Since a local feature vector $\mathbf{f}_i \in \mathbf{F}$ contains appearance and motion information from a limited spatio-temporal region from a full video clip, different action may contain significantly similar local feature vectors. However, obtaining precise annotations for each local feature vector is laborious and time-consuming. So, we assigned the label of the video clip to all its local action vectors. For each class, a transformed representation can be learned to curb the inter-class similarity and intra-class variability using a deep multiple instance learning based ranking model similar to [38] as shown in Fig. 3.4. Instead of applying ranking on each local feature, we apply ranking only on two instances, one from a clip of action of interest $\mathcal{F}^+$ and other from a clip of any other action $\mathcal{F}^*$ having the highest score for the action of interest. The objective is to encourage the high scores for the features

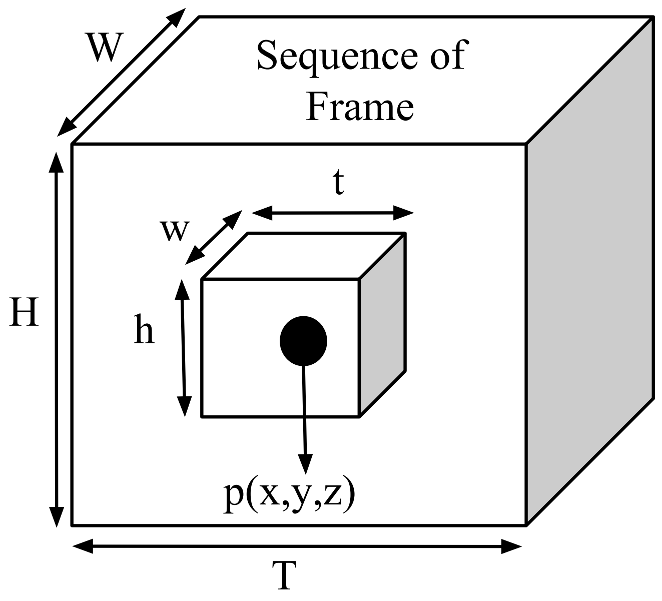


Figure 3.3: STVVs from a video can be extracted by gathering the pixels around a point $\mathbf{p}(x, y, z)$ using a 3-D sliding volume of size (w, h, t).

from $\mathcal{F}^+$ as compared to features from $\mathcal{F}^*$ as

$$\max_{i \in \mathcal{F}^+} f(\mathbf{f}_i^+) > \max_{i \in \mathcal{F}^*} f(\mathbf{f}_i^*). \tag{3.5}$$

To keep the score of the features in $\mathcal{F}^*$ away from the score of the features in $\mathcal{F}^+$, we use the hing-loss function

$$l(\mathcal{F}^+, \mathcal{F}^*) = \max(0, 1 - \max_{i \in \mathcal{F}^+} f(\mathbf{f}_i^+) + \max_{i \in \mathcal{F}^*} f(\mathbf{f}_i^*)). \tag{3.6}$$

Finally, the model weights $\mathcal{W}$ are computed by minimizing the objective function

$$\mathcal{J}(\mathcal{W}) = l(\mathcal{F}^+, \mathcal{F}^*) + \lambda_3 \|\mathcal{W}\|_{\mathcal{F}}, \tag{3.7}$$

3.2.3 Graph Representation of an Action in a Video

Once we obtain a set of interest points $\mathcal{P}$ in space and time and their respective feature vectors $\mathcal{F}$ for each video clip, the video can be represented as a graph $G(\mathcal{P}, \mathcal{E})$, where nodes $\mathcal{P}$ represent the set of spatio-temporal points and $\mathcal{E}$ denotes the set of edges which represents the correlation between the points for a particular action class. An edge A_{ij} in the graph G is calculated based on the edge weight e_{ij} computed using

$$e_{ij} = \frac{\kappa(\mathbf{f}_i, \mathbf{f}_j)}{||\mathbf{p}_i - \mathbf{p}_j||_2}, \tag{3.8}$$

where, $\kappa(\mathbf{f}_i, \mathbf{f}_j)$ is a kernel function used to compute the correlation between the local descriptors $\mathbf{f}_i$ and $\mathbf{f}_j$ extracted at points $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively. The value of e is high if $\mathbf{f}_i$ and $\mathbf{f}_j$ has high correlation which shows that these points belong to the same action class or the object at point $\mathbf{p}_i$

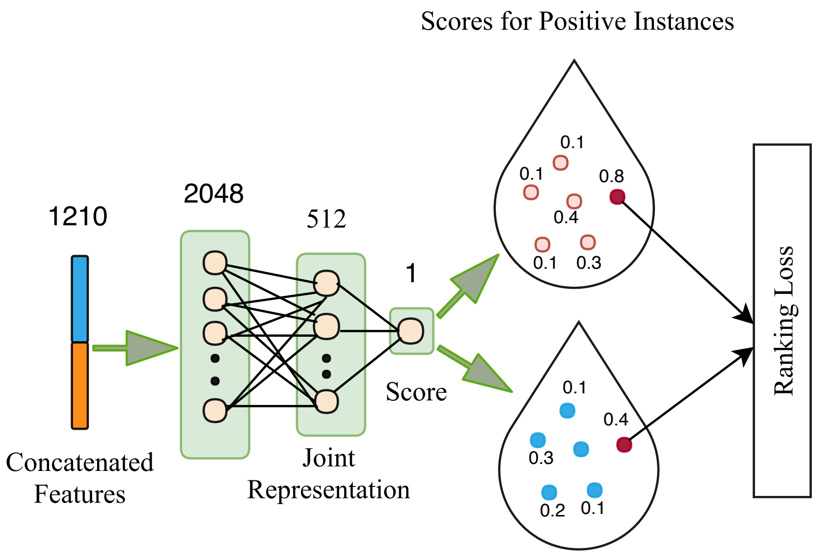


Figure 3.4: Deep multiple instance learning framework for Weakly supervised transformation of the local action features with video-clip level annotations.

moved to point $\mathbf{p}_j$ over the time. Both the cases are vital for action recognition and localization. The distance $||\mathbf{p}_i - \mathbf{p}_j||_2$ between points $\mathbf{p}_i$ and $\mathbf{p}_j$ is inversely proportional to the edge score i.e the distance between two points impacts the correlation between them. The adjacency matrix of the graph G is computed as

$$A_{ij} = \begin{cases} 0 & \text{, if } e_{ij} < e_T \\ 1 & \text{, otherwise} \end{cases} \tag{3.9}$$

Here, e_T is some threshold. Fig. 3.5 shows an example graph generated for a sample video from UCF101-24 dataset for basketball action.

3.2.4 Action Recognition and Localization

After representing an action in a video as a graph, we classify the graphs into one of the action classes. The graph obtained can be viewed as a 3D volume in space and time, and the action is localized based on the nodes in the graph.

Action Recognition

We train a max-margin binary classifier using the graphs for a particular action category by labeling them as the positive class and the rest of the graphs are labelled as negative class. Let $\{\mathcal{G}_i, y_i\}_{i=1}^{n}$ denote the corresponding labeled graphs for n training action clips, where the label y_i is $+1$ for graphs of action of interest and -1 for graphs of other actions. The SVM [39] can be trained using the data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in R^d$ and $y_i \in \{-1, +1\}$ as:

$$\min J = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j y_i y_j K(\mathbf{x}_i^T, \mathbf{x}) - \sum_{i=1}^{n} \beta_i, \tag{3.10}$$

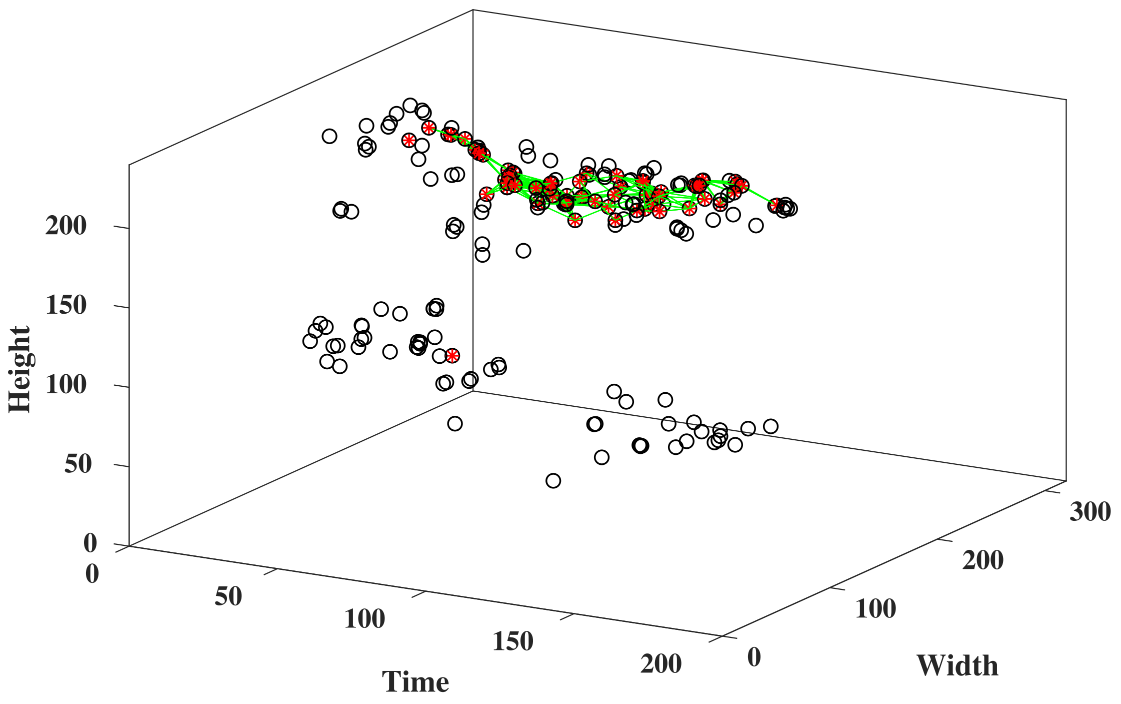


Figure 3.5: Graph generated from a sample video from UCF101-24 dataset for basketball action. The points with red asterisk [*] are those points having high score for the basketball action and the points with black circle [o] are the eliminated points having low score for the basketball action. The tube like shaped graph connecting the red asterisks with a green line [—] is the generated graph used for the recognition and localization of the action in this video. [Best viewed in colors]

$$\text{subject to } \sum_{i=1}^{n} \beta_i y_i = 0 \text{ and } 0 \leq \beta_i \leq C,$$

Here, C is a parameter. After we find the solution by solving the above optimization problem, we get m support vectors (SV), with respective β_is, and the bias b. Now, the decision function can be written as:

$$f(x) = sign\left(\sum_{i=1}^{m} \beta_i y_i K(\mathbf{x}_i^T, \mathbf{x}) + b \right), \tag{3.11}$$

Here, β_is are lagrange multipliers. $K(\mathbf{x}_i^T, \mathbf{x})$ is the kernel function similar to [40, 41, 42, 43] which is used to compute the similarity between two points. This approach can also be used for graph classification using a suitable kernel. In this work, we have used random walk graph kernel to measure the similarity between two graphs as it is computationally efficient and is suitable for such tasks.

The random walk graph kernel [44] counts the number of common random walks which is used to compare two graphs. The number of walks of length l between the two graphs can be calculated directly using the product graph [44]. The l^{th} power of the adjacency matrix of the resultant graph obtained after direct product gives us the number of common walks.

The standard SVM classifier in Equation (3.10) with a graph kernel can now be rewritten as:

$$\min\ J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j y_i y_j K(G_i, G_j) - \sum_{i=1}^{n}\beta_i, \tag{3.12}$$

$$\text{subject to } \sum_{i=1}^{n}\beta_i y_i = 0 \text{ and } 0 \leq \beta_i \leq C.$$

And the decision function in Equation 3.11 for a test graph G will be

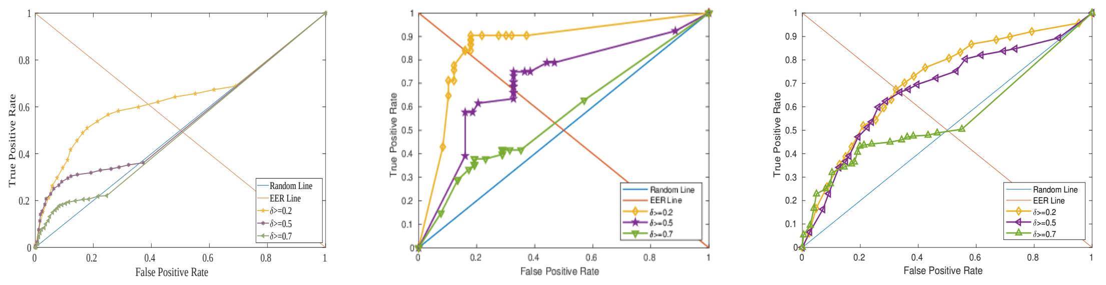
$$f(x) = sign\left(\sum_{i=1}^{m}\beta_i y_i K(G_i, G) + b\right). \tag{3.13}$$

Equation (3.12) can thus be solved for the graphs built from the training videos for each action and using this and solving 3.13 gives the decision for a test video.

Spatio-temporal Action Localization

After successful recognition of the action class, the next objective is to localize the recognized action in space and time. For that, we take a union of the spatio-temporal regions in the proximity of each node in the graph representing the action in a video clip. As the ground truths provided for the benchmark datasets are of actors instead of the action, we use the SSD [9], a CNN based object detector for person detection in those regions of each frame where there is a node in the graph in order to predict a tube of bounding boxes localizing the actors playing action across the video frames.

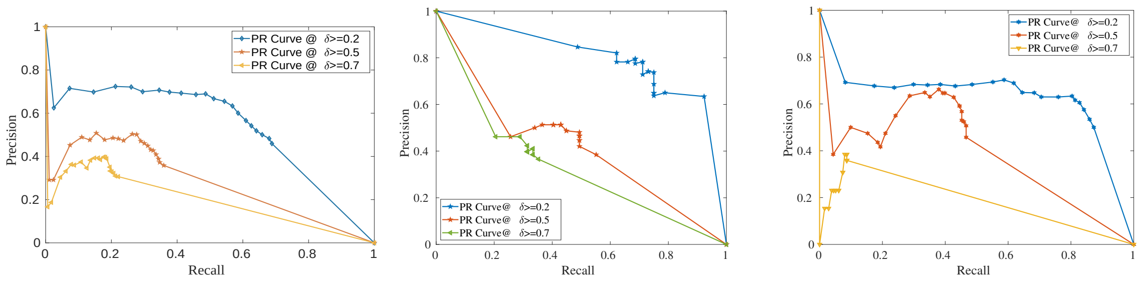
3.3 Experiments



(A) UCF101-24 dataset (B) UCF-Sports dataset dataset (C) JHMDB

Figure 3.6: The ROC curve for the action localization using the proposed approach for various IOU threshold.

We have performed our experiments on the three challenging benchmarks datasets, namely, UCF-101-24 [45], UCF-Sport [46] and J-HMDB-21 [47]. UCF101-24 is a subset of UCF101 action recognition dataset which is large and diversified and a challenging action dataset. In the THUMOS-2013 challenge, the spatiotemporal bounding box annotations are released for 24 classes out of 101.



(A) UCF101-24 dataset (B) UCF-Sports dataset (C) JHMDB dataset

Figure 3.7: The precision-recall (PR) curve for the action localization using the proposed approach for various IOU threshold.

Similar to previous spatiotemporal action localization approaches [4, 5, 48], we test our method on split 1.

To measure the recognition performance, we use the overall accuracy and to measure the localization performance, we use the widely accepted localization metrics: AUC (area under the curve) and mAP (mean average precision).

Table 3.1 shows the comparison of the action recognition in each of the class while training an SVM classifier with graph kernel between the graphs of the action of interest as positive class and graphs of other action categories as the negative class. To avoid the overfitting due to imbalance class classification during training, we randomly select the equal number of graphs from all other actions for the negative class. The results in the table below show that the our proposed method outperforms the existing approaches such as $T - Sliding$, $ST - Cube - subvol$, $T - Subgraph$, and $T - Jumb - Subgraph$ in term of correctly recognizing each individual action. This improvement in the capability of recognizing action is due to the ability of labeled graph generated from the joint local feature representation which is free from the curse of the intra-class variability and the inter-class similarity.

Table 3.1: Performance (%) of the classification of each action in UCF Sport dataset

Verbs	**T-Sliding**	**ST-Cube-Subvol**	**T-Jump-Subgraph**	**Our**
Diving	0.8106	0.7561	0.9091	0.7186
Lifting	0.7899	0.8058	0.8096	**0.8430**
Riding	0.5349	0.5075	0.3888	0.3207
Running	0.4602	0.3269	0.4705	**0.5344**
Skate	0.1407	0.1057	0.1803	**0.6301**
Swing-B	0.5520	0.6259	0.4582	**0.7170**
Swing-S	0.6728	0.3478	0.7212	0.4601
Walking	0.4085	0.3462	0.4657	**0.8150**

The ROC curve at various IoU threshold δ are presented in Fig. 3.6 for three datasets. The high area under the curve shows the better recognition as well as efficient localization of the action in space and time. The performance of the localization increases while reducing the threshold on IoU. The consistent performance on all three datasets shows the generalization capability of the proposed approach. Also, the precision-recall (PR) curve δ presented in Fig. 3.6 for three datasets are also showing high mAP performace at various IoU threshold.

Table 3.2, 3.3 & 3.4 present the mean average precision (mAP) results for the action localization in space and time with a comparison to several existing approaches for fully-supervised and weakly-supervised approaches. The mAP for the proposed approach performs better than all the existing weakly-supervised approaches and achieves a comparable performance with the supervised approaches.

Table 3.2: Spatio-temporal action localization results (mAP) on untrimmed videos of UCF101-24 dataset

IoU threshold δ	**0.2**	**0.5**	**0.75**
Weinzaepfel et al.†[5]	46.8	-	-
Yu et al.†[17]	26.5		
Peng and Schmid†[35]	73.5	32.1	02.7
Saha et al.† [8]	66.6	36.4	07.9
Hou et al.† [34]	47.1	-	-
Jain et al.† [3]	48.1	-	-
Kalogeiton et al.† [2]	77.2	-	-
Singh et al.† [3]	73.5	46.3	15.0
Li et al.[49]	37.7	-	-
Mettes et al.* [50]	37.4	-	06.2
Our model	58.4	31.9	20.1

For all the datasets, we compute the mAP of our approach using three thresholds on IoU i.e. $\delta = 0.2, 0.5, 0.75$ similar to [3]. The reason behind the success of the proposed approach in spatio-temporal localization of the actions over the existing approaches is because of the incorporation of the MIL ranking framework for computing the action score and a joint representation of appearance and information at the carefully selected salient points. In particular, this framework assign a high score to those local action features which are unique to action of interest and also learn a joint representation for them with improved discrimination ability by keeping them far-apart from the local features of other actions.

Table 3.3: Spatio-temporal action localisation results (mAP) on untrimmed videos of UCF Sports dataset

IoU threshold δ	**0.2**	**0.5**	**0.75**
Weinzaepfel et al.†[5]	-	90.5	-
Kalogeiton et al.† [2]	-	91.7	-
Hou et al.† [34]		86.7	-
Mettes et al.* [50]	81.7	37.8	-
Our model	79.6	56.0	29.8

Table 3.4: Spatio-temporal action localisation results (mAP) on untrimmed videos of JHMDB21 dataset

IoU threshold δ	**0.2**	**0.5**	**0.75**
Gkioxari and Malik[4]	-	53.3	-
Wang et al.[48]	-	56.4	-
Weinzaepfel et al.†[5]	63.1	60.7	-
Saha et al.][8]	72.6	71.5	43.3
Peng and Schmid†[35]	74.1	73.1	02.7
Our model	68.8	63.4	

The comparison of the experimental results of proposed approach with the state-of-the-art approaches on three challenging benchmark datasets for action localization confirms the superiority of the proposed approach for both action recognition as well as action localization with class label annotations only. Fig. 3.8 & 3.9 provide a detailed comparison of the proposed approach with the some of the existing fully supervised approaches showing a trade-off between recall and IoU at several threshold for UCF101-24 and UCF-Sports datasets, respectively. Although our approach is weakly supervised, our proposed method gives a comparable performance to fully supervised approaches.

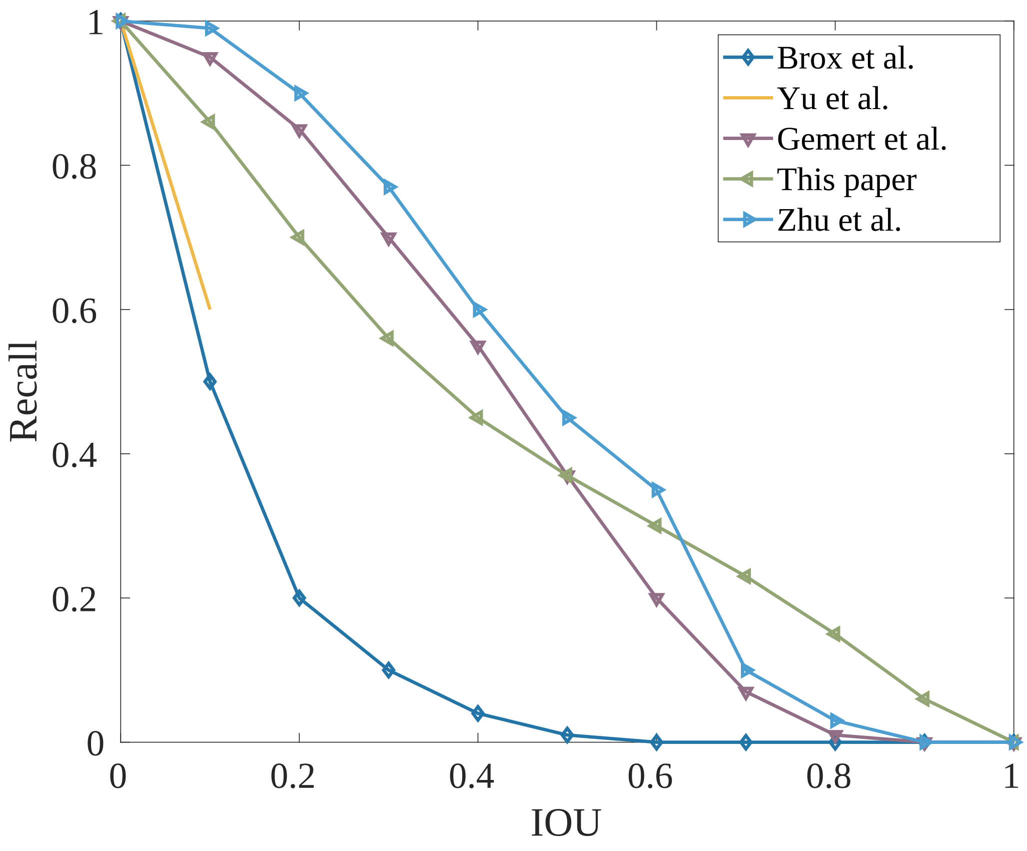


Figure 3.8: Recall and IoU trade-off curve for the UCF101-24 dataset

3.4 Conclusion

The proposed approach for the action recognition and localization performs better than existing approaches because of the efficient graph representation using the relevant local actions where the MIL ranking model learns a joint representation to rid of the curse of inter-class similarity and intra-class variations. Also, the proposed approach is fast and able to search candidates in a large space-time region in short time in comparison to existing approaches because it does not compute a bag-of-words histogram on a large vocabulary as used in [1].

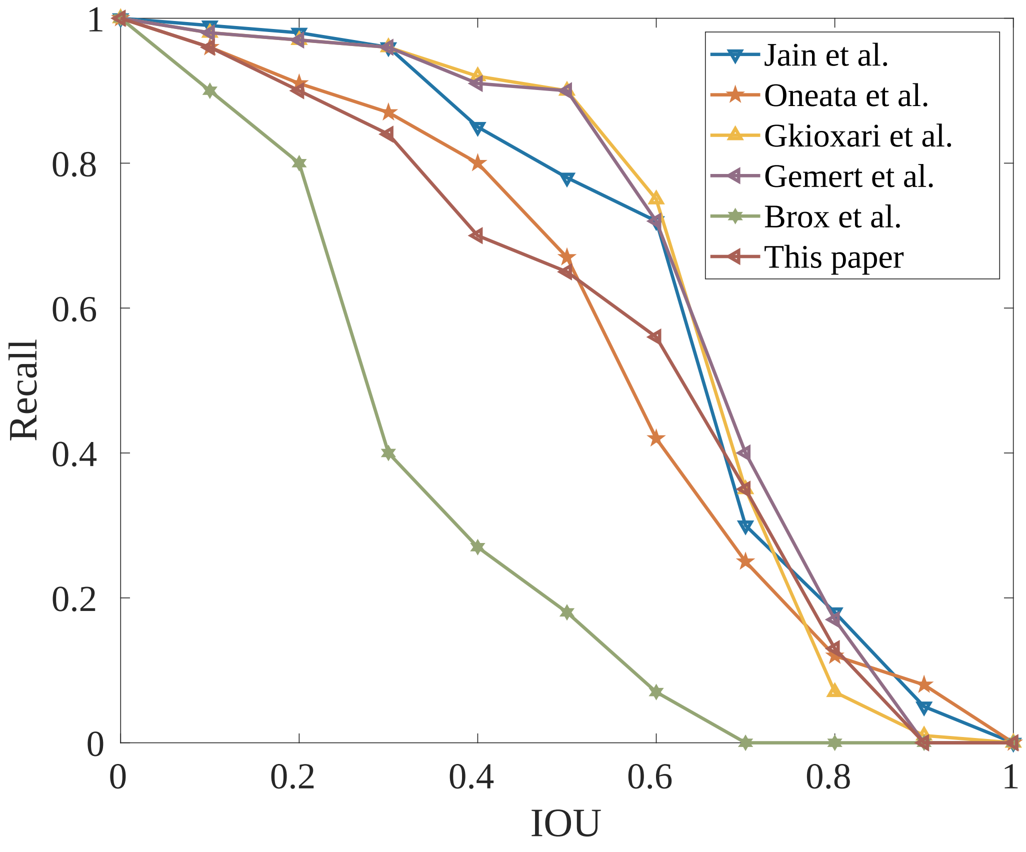


Figure 3.9: Recall and IoU trade-off curve for the UCF-Sports dataset

Chapter 4

Deep Representation Learning using Subspace Attention

Convolution neural networks [51, 52, 53] or CNNs have achieved exceptional performance in various cognitive tasks. The unprecedented performance of CNNs stems from the rich representational power of CNNs, which in-turn stems from the deeper and wider layers in networks. Deeper and wider layers boost the expressiveness and discrimination ability of the network by circumventing the limitations of convolution operators, viz., locality [21] and linearity [54]. In CNNs, convolution operators capture the local (e.g., 3×3) feature correlations and enable weight sharing to reduce the number of learn-able parameters [55]. Multiple convolution operators are stacked in CNNs to enlarge the receptive field and capture the long-range dependencies [21], which makes the CNNs deeper. Further, since the linearity of convolution operation leads to inefficient capturing of the non-linear abstraction of input data [54], CNNs employ higher number of filters per layer which are learnt to capture all the possible variations of the same latent concept [54]. However, this make the CNNs wider. Altogether, deeper and wider layers in CNNs leads to high computational cost (measured in the number of floating point operations or FLOPs) and the number of parameters increase which makes deployment of CNNs on resource-constrained platforms quite challenging.

4.1 Subspace Attention

Learning to capture long-range inter-dependencies in visual data is of primary interest for deep convolution neural networks. However, convolution operations in vanilla CNNs are responsible to capturing local relations, hence, are inefficient to captue long range dependencies.

In this work, we try to overcome the issues associated with CNNs by capturing long-range dependencies in visual data. We aim to achieve the same while keeping in mind the computation and parameter overhead. We have observed that by dividing the feature maps into multiple subspaces and learning to capture spatial as well as cross channel interactions among the feature maps provides us with compute efficient way to capture long-range dependencies in the data.

We split the feature maps from an intermediate convolution layer and learn spatial attention maps individually for each group. Also, we learn to capture spatial interaction among the feature maps using Global Attention Pooling operation.

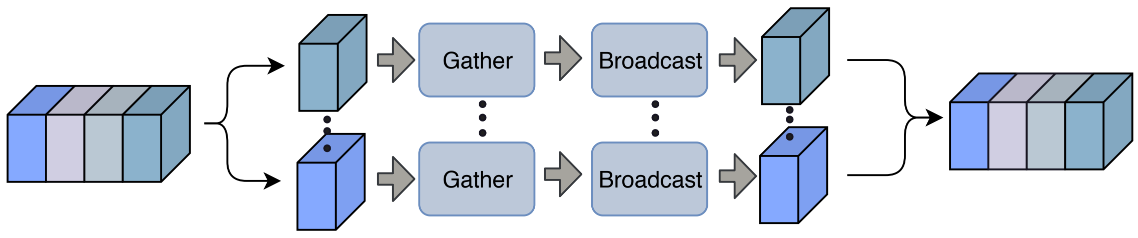


Figure 4.1: Gather spatial as well as cross channel interactions from the feature maps from multiple sub-spaces

The subspace attention network is related to a number of recent works. Capturing cross channel interactions among the feature maps is inspired from Convolution Block Attention Module(CBAM) [30] and Global Attention mechanism is inspired from Squeeze-and-excitation networks [31]. However, compared to the above methods, subspace attention networks benefits from several unique advantages : (a) computation efficient way of generating attention maps, (b) parameter overhead is very less (c) capture spatial and cross-channel information among the feature maps.

4.1.1 GrAB : Grouped Attention Block for compact CNNs

Let $F \in R^{c \times h \times w}$ be an input tensor, i.e. feature maps from an intermediate convolution layer, where c is the number of channels, h and w are the spatial dimensions of the feature maps. Our objective is to learn to capture the cross-channel inter-dependencies in the feature maps without incurring significant parameter and computation overhead. As shown in Figure 4.2, GrAB divides the input feature maps (F) into g mutually exclusive groups $[F_1, F_2, ...F_{\tilde{n}}, ...F_g]$ where each group have G feature maps. We define $F_{\tilde{n}}$ as a group of intermediate feature maps and proceed as follows.

$$A_{\tilde{n}} = softmax(PW^1(maxpool^{3\times3,1}(DW^{1\times1}(F_{\tilde{n}})))) \tag{4.1}$$

$$\hat{F}_{\tilde{n}} = (A_{\tilde{n}} \otimes F_{\tilde{n}}) \oplus F_{\tilde{g}} \tag{4.2}$$

$$\hat{F} = concat([\hat{F}_1, \hat{F}_2, ...\hat{F}_{\tilde{n}}, ...\hat{F}_g]) \tag{4.3}$$

In Eq. 4.1, $maxpool^{3\times3,1}$ is maxpool with kernel size $= 3 \times 3$ and padding $= 1$, $DW^{1\times1}$ is depthwise convolution with kernel size $= 1 \times 1$, PW^1 is pointwise convolution with only one filter, and $A_{\tilde{n}}$ is an attention map inferred from the a group of intermediate feature maps ($F_{\tilde{n}}$). Attention map ($A_{\tilde{n}}$) in each group (subspace) capture the non-linear dependencies among the feature maps by learning to gather cross channel information. To ensure $\sum_{i,j} A_{\tilde{n}}(i,j) = 1$ i.e. a valid attention weighting tensor, we employ a gating mechanism with a softmax activation in Eq. 4.1. After the feature-redistribution (in Eq. 4.2), each group of feature maps gets the refined set of feature maps ($\hat{F}_{\tilde{n}}$). In Eq. 4.2, $\otimes$ denotes element-wise multiplication and $\oplus$ denotes element-wise addition. The final output of GrAB ($\hat{F}$) is obtained by concatenating the feature maps from each group (Eq. 4.3). Note that, unlike [21] and [28] we use only one filter in pointwise convolution which makes our block compute-efficient which is desirable for integration in compact CNNs.

The idea of generating attention maps to gather information across the channels is inspired from

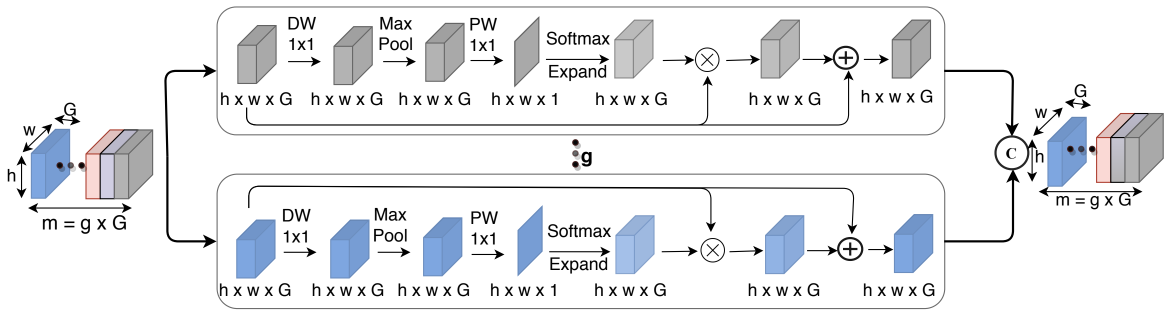


Figure 4.2: GrAB divides the input feature maps into g mutually exclusive groups where each group contains G feature maps.

the SE-Net [31]. SE-Net first captures the spatial information using squeeze operation in spatial dimension and further, captures the channel-wise inter-dependencies using parameter-heavy MLP in excitation operation. Unlike SE-Net, first we perform depthwise convolution followed by maxpool operation to highlight local informative regions [56] (Eq. 4.1). Further, we perform squeeze operation along the channel dimension using pointwise convolution with only one filter and we eschew using parameter-heavy MLP which makes our block ultra-lightweight and become amenable for compact CNNs.

In effect, GrAB learns to capture the complex interaction of cross channel information with very few parameters and computations unlike the stacking of layers with same configuration in MobileNet-V1 and MobileNet-V2 which is compute and parameter heavy. We analyze GrAB by considering three prominent cases:

Case 1 : $g = 1$: In this case, there is only one group which implies that the cross channel information for the whole feature volume will be captured by a single attention map. Intuitively, a single attention map is not sufficient to capture the complex relations in the entire feature space.

Case 2 : $1 < g < m$: Dividing the feature maps into g groups implies that g attention maps will be generated. Each attention map is capable of capturing the cross channel information from the feature maps in their respective groups. We have performed our experiments (Section 4.2) with $g = 2, 4, 8$ and have obtained performance improvement for $g = 4$ on ImageNet and $g = 8$ on fine-grained datasets. *We notice that diving the feature maps into groups does not incur additional parameter and computation overhead while learning better cross-channel information.*
Case 3 : $g = m$: When number of groups equals to the number of channels, attention map is generated from a single feature map in each group. Therefore Eq. 4.1 can be re-written as:

$$A_{\tilde{n}} = softmax(\alpha_2 \otimes (maxpool^{3\times3,1}(\alpha_1 \otimes F_{\tilde{n}}))) \tag{4.4}$$

Here α_1 and α_2 are the parameters for depth-wise and point-wise convolution operation. We can see that in each group there is only one feature map i.e $G = 1$ and nothing is to be learned along the channel dimension. Hence, the attention map will not be able to capture the cross channel information and the process of generating attention maps reduces to a non-linear transformation of the feature map itself.

From the above discussion, it is clear that better interaction of cross-channel information can be obtained when $1 < g < m$ and this intuition is confirmed by the experimental results shown in Table 4.2 and Table 4.8. To keep the model simple and desirable for compact CNNs, we avoid intelligent division of feature maps into groups. Also, it is worth noticing that diving the feature maps into groups does not incur additional parameter and computation overhead while learning better cross-channel information. In other words, amount of computation and number of parameters are constant for a given number of channels (m) irrespective of the number of groups(g) formed.

4.1.2 Subspace Attention Pooling

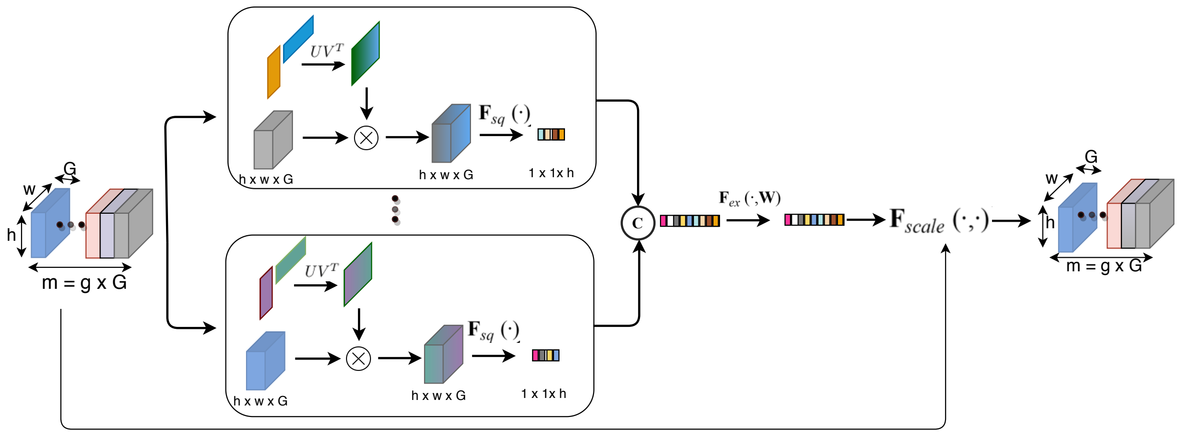


Figure 4.3: Subspace attention pooling captures spatial relations among the feature maps

For a set of intermediate feature maps $F \in R^{c\times h\times w}$, where c is the number of channels, h and w are the spatial dimensions of the feature maps. Our objective is to learn to capture spatial inter-dependencies in the feature maps without incurring significant parameter and computation overhead. As shown in Figure 4.3, Subspace Attention Pooling divides the input feature maps (F) into g mutually exclusive groups $[F_1, F_2, ...F_{\tilde{n}}, ...F_n]$ where each group have G feature maps. For a set of attention maps , We define $F_{\tilde{n}}$ as a group of intermediate feature maps and proceed as follows.

$$A_{\tilde{n}} = softmax(UV^T) \tag{4.5}$$

$$\hat{S}_{\tilde{n}} = G_{sq}(A_{\tilde{n}} \otimes F_{\tilde{n}}) \tag{4.6}$$

$$\hat{S} = concat([\hat{S}_1, \hat{S}_2, ...\hat{S}_{\tilde{n}}, ...\hat{S}_n]) \tag{4.7}$$

$$\hat{S} = mlp(G_{ex}(\hat{S})) \tag{4.8}$$

$$\hat{F} = G_{scale}(\hat{S}, F)) \tag{4.9}$$

Here, U and V are attention parameters. We approximate the attention map A_{map} as a low-rank approximation : $A_{map} = UV^T$ where U, V $\in R^{f \times 1}$. $G_{sq}(.)$ denotes squeeze operation i.e weighted sum of individual feature maps. This way of pooling enables the network to capture spatial relations in the feature maps. The pooled features are then passed through a multi-layer perceptron which broadcasts the spatial information from each subspace. The final set of feature maps are obtained by $G_{scale}(.)$ operation which distributes the spatial information among the feature maps.

4.2 Experimental Evaluation

Experimental setup and dataset : We perform our experiments with MobileNet-V1 and MobileNet-V2 as baseline architectures and to enable fair comparison and show the effectiveness of GrAB, we reproduce the results of the baseline networks (MobileNet-V1 and MobileNet-V2). We perform all our experiments using PyTorch [57] framework. Our experiments are performed on ImageNet-1K [58] and fine-grained datsets, Caltech-UCSD Birds 200 [59] and Stanford Dogs [60], for image classification. We have reported validation accuracy for single crop with the input image 224 × 224. We use 4 Nvidia-P100 GPU for ImageNet-1K experiments and 1 Nvidia-P100 GPU for fine-grain classification experiments.

Layer no.	**Layer config. (in, out, stride)**
1 (Std.)	(3, 32, 2)
2,3 (DWS)	(32, 64, 1), (64, 128, 2)
4,5 (DWS)	(128, 128, 1), (128, 256, 2)
6,7 (DWS)	(256, 256, 1), (256, 512, 2)
8 - 12 (DWS)	$5 \times (512, 512, 1)$
13, 14 (DWS)	(512, 1024, 2), (1024, 1024, 1)
AvgPool, FC, softmax	

Table 4.1: MobileNet-V1 architecture

Layer no.	**Layer type**
8	conv DWS (512, 512, 1)
8:1	GrAB
9	conv DWS(512, 512, 1)
9:1	GrAB
10	conv DWS(512, 512, 1)
11	GrAB
12	conv DWS(512, 512, 1)

Table 4.2: MobileNet-V1 layers 8-12 with GrAB

Layer no./type	**(in, out)**
1 / conv2d	(3, 32)
2 / residual block	(32, 16)
3-4 / residual block	$2 \times (16, 24)$
5 / residual block	(24, 32)
6-7 / residual block	$2 \times (32, 32)$
8 / residual block	(32, 64)
9-11 / residual block	$3 \times (64, 64)$
12 / residual block	(64, 96)
13-14 / residual block	$2 \times (96, 96)$
15 / residual block	(96, 160)
16-17 / residual block	$2 \times (160, 160)$
18 / residual block	(160, 320)
19 / conv2d	(320, 1280)
AvgPool	
20 / conv2d	(1280, num classes)

Table 4.3: MobileNet-V2 architecture

Where to insert GrAB in CNNs ? The GrAB module can be inserted between the layers and/or it can substitute the layers from the stack of repeated convolution layers with same configuration. For example, MobileNet-V1 has a stack of 5 layers with 512 input and output feature maps (layers 8-12 in (Table 4.1)). Similarly, MobileNet-V2 has stacks of repeated residual blocks with same configuration e.g., layer 9-11 and layer 13-14 (Table 4.3)

4.2.1 MobileNet-V1 + GrAB

As shown in Table 4.1, MobileNet-V1 performs DWS convolution in all the layers except the first convolution layer. Layers 8 to 12 have the same configuration and they account for 46% of total computation. We use GrAB to optimize this part of the network by inserting GrAB between the layers as well as by substituting layers with GrAB. First, we insert GrAB after 8^{th} and 9^{th} layers and substitute 11^{th} layer with GrAB as shown in Table 4.2.

Experiments on ImageNet-1k: We perform experiments on ImageNet-1K dataset with a group size of 1,4,8,16 and find that our model achieves 70.43% accuracy with the group size of 4. We train the MobileNet-V1 for 90 epochs with batch size of 128 and use SGD optimizer with 0.9 momentum. The initial learning rate is 0.1 and it reduces to $1/10^{th}$ every 30 epochs. Experimental results on validation set with different group sizes are shown in the Table 4.4.

Models	**Positions**	**#Params**	**#FLOPs**	**Top-1 (%)**	**Top-5 (%)**
1.0 MNet-V1 (vanilla)	–	4.2M	569M	70.65	89.76
1.0 MNet-V1 + GrAB ($g = 1$)	(8:1, 9:1, 11)	3.9M	517M	69.92	89.25
1.0 MNet-V1 + GrAB ($g = 4$)	(8:1, 9:1, 11)	3.9M	517M	**70.43**	89.92
1.0 MNet-V1 + GrAB ($g = 8$)	(8:1, 9:1, 11)	3.9M	517M	70.29	89.96
1.0 MNet-V1 + GrAB ($g = 16$)	(8:1, 9:1, 11)	3.9M	517M	70.04	**89.98**

Table 4.4: Image classification accuracy (single-crop) of MobileNet-V1 with input size 224 on ImageNet-1k validation set. Using three GrAB blocks, we achieve 9.2% and 6.4% reduction in FLOPs and parmeters (respectively) with only 0.22% drop in the top-1 accuracy.

We achieve 70.43% accuracy when number of groups are 4 with 9% less FLOPs than original. As we increase the number of groups there is negligible decrease in accuracy. The reason is that the number of channels in each group reduces as we increase the number of groups which inhibits the cross-channel information exchange.

We also perform experiments on scaled version of MobileNet-V1 architecture where number of filters in each layer are scaled down by a factor (width multiplier) α, where $\alpha \in \{0.5, 0.75\}$. Table 4.5 shows the results.

Models	**Positions**	**#Params**	**#FLOPs**	**Top-1 (%)**	**Top-5 (%)**
0.75 MNet-V1 (vanilla)	–	2.6M	325M	67.81	88.00
0.75 MNet-V1 + GrAB ($g = 1$)	(8:1, 9:1, 11)	2.4M	296M	**67.98**	88.06
0.75 MNet-V1 + GrAB ($g = 4$)	(8:1, 9:1, 11)	2.4M	296M	67.48	**88.43**
0.50 MNet-V1 (vanilla)	–	1.3M	149M	63.22	84.93
0.50 MNet-V1 + GrAB ($g = 1$)	(8:1, 9:1, 11)	1.2M	136M	**63.30**	84.70
0.50 MNet-V1 + GrAB ($g = 4$)	(8:1, 9:1, 11)	1.2M	136M	63.25	**84.81**

Table 4.5: Results on Scaled MobileNet-V1 with GrAB

We achieve the highest accuracy with group size 1 instead of the group size 4. This happens due to the fact that the scaled version of MobileNet-V1 already have few feature maps which gets further divided into groups and leads to less number of feature maps per group as happened in case of $g = 4$ thus very less information comes into the group.

In Table 4.4 we notice that the top-1 accuracy of MobileNet-V1 integrated with GrAB is lower than the baseline network whereas top-5 accuracy is higher than the baseline (except $g = 1$). This implies that mis-classification which lead to lower top-1 accuracy is correctly predicted in top-5 predictions by the GrAB integrated network.

Fine-Grain classification: Fine-grain classification is very challenging due to high inter-class similarity and substantial intra-class variation. We now show the effectiveness of GrAB on fine-grained classification task. Table 4.6 and Table 4.7 show the experiments on UCSD-Birds 200 and Stanford Dogs dataset, respectively.

The highest Top-1 accuracy reported is 64.44% on UCSD-Birds 200 and 63.30% on Stanford Dogs dataset with number of groups $g = 8$.

Models	**Positions**	**#Params**	**#FLOPs**	**Top-1 (%)**	**Top-5 (%)**
1.0 MNet-V1 (vanilla)	–	4.2M	569M	62.88	86.05
1.0 MNet-V1 + GrAB ($g = 1$)	(8:1, 9:1, 11)	3.9M	517M	62.46	86.01
1.0 MNet-V1 + GrAB ($g = 4$)	(8:1, 9:1, 11)	3.9M	517M	63.52	85.80
1.0 MNet-V1 + GrAB ($g = 8$)	(8:1, 9:1, 11)	3.9M	517M	**64.44**	**86.60**
1.0 MNet-V1 + GrAB ($g = 16$)	(8:1, 9:1, 11)	3.9M	517M	63.47	84.96

Table 4.6: Results on MobileNet-V1 with GrAB on UCSD-Birds 200 dataset

Models	**Positions**	**#Params**	**#FLOPs**	**Top-1 (%)**	**Top-5 (%)**
1.0 MNet-V1 (vanilla)	–	4.2M	569M	62.20	89.66
1.0 MNet-V1 + GrAB ($g = 1$)	(8:1, 9:1, 11)	3.9M	517M	62.73	88.8
1.0 MNet-V1 + GrAB ($g = 4$)	(8:1, 9:1, 11)	3.9M	517M	63.06	89.58
1.0 MNet-V1 + GrAB ($g = 8$)	(8:1, 9:1, 11)	3.9M	517M	**63.30**	**89.68**
1.0 MNet-V1 + GrAB ($g = 16$)	(8:1, 9:1, 11)	3.9M	517M	62.75	89.35

Table 4.7: Results on MobileNet-V1 with GrAB on Stanford Dogs dataset

4.2.2 MobileNet-V2 + GrAB

MobileNet-V2 has residual bottleneck blocks from layer 2 to 18 as mentioned in Table 4.3. Each residual bottleneck block consists of 3 convolution layers. The first convolution layer use pointwise convolution to expand the number of feature maps by a factor of $k = 6$; the second layer is the depthwise convolution with filter size 3×3 and the third layer is standard convolution layer with filter size 1×1. To train the network, we use SGD optimizer with 0.9 momentum and 0.00004 weight decay, initial learning rate is set to 0.045 and decays to $0.98\times$ after every epoch. We train the network for 400 epochs for ImageNet-1K classification.

Experiments on Imagenet-1k: Following the observations from MobileNet-V1 results on ImageNet-1K dataset, we perform all the experiments of MobileNet-V2 on ImageNet-1K with $g = 4$. Table 4.8 shows the results.

Models	**Positions**	**#Params**	**#FLOPs**	**Top-1 (%)**	**Top-5 (%)**
MNet-V2 (Vanilla)	–	3.4M	300M	71.60	90.19
MNet-V2 + GrAB ($g = 4$)	(14, 17)	3.17M	255.01M	71.52	90.25
MNet-V2 + GrAB($g = 4$)	(16, 17)	2.77M	269.15M	70.74	89.15
MNet-V2 + GrAB ($g = 4$)	(13, 14, 16, 17)	2.54M	224.16M	69.72	87.79

Table 4.8: Experiments on Imagenet using MobileNet-V2

Fine-Grain classification:

Since fine-grained datasets have much higher intra-class variation, we perform our experiments with g=1, 4, 8, and 16. On Birds dataset, MNet-V2+GrAB performs better than vanilla MNet-V2 (Table 4.10). For instance, with g = 8 and using GrAB at positions (14, 17), the top-1 accuracy is increased from 62.94% to 65.41% and top-5 accuracy is increased from 84.92% to 86.01%. Also, the FLOPs and parameter count are reduced by 12.59% and 12.63%, respectively. Similarly, with g = 4 and using GrAB at positions (13,14,16,17), there is 25.28% and 25.27% reduction in FLOPs and parameter count, respectively and the top-1 accuracy is 64.15%.

The results with Dogs dataset are shown in Table4.10. Here also, our technique reduces the computational and storage costs significantly for all configurations, while improving the accuracy for most of the configurations. Clearly, GrAB is highly effective for fine-grained image classification.

2*Model	**2*Positions**	**2*#Params**	**2*#FLOPs**	g=1		g=4		g=8		g=16	
				top-1	**top-5**	**top-1**	**top-5**	**top-1**	**top-5**	**top-1**	**top-5**
MNet-V2 (vanilla)	–	3.4M	300M	**top-1** = 62.94, **top-5** = 84.92							
MNet-V2 + GrAB	13	3.28	277.51	63.01	85.17	63.05	83.48	63.05	84.79	64.32	84.62
MNet-V2 + GrAB	16	3.08	284.57	63.98	86.22	64.44	84.87	65.03	85.63	63.47	84.45
MNet-V2 + GrAB	(9,13)	3.23	267.40	63.43	85.55	63.47	84.41	63.10	84.96	62.21	84.62
MNet-V2 + GrAB	(16,17)	2.77	269.15	64.19	85.46	64.57	84.92	64.61	85.25	65.03	85.64
MNet-V2 + GrAB	(14,17)	3.17	255.01	63.35	85.29	64.70	86.98	65.41	86.01	63.31	84.92
MNet-V2 + GrAB	(13,14,16,17)	2.54	224.16	64.11	86.77	64.15	84.92	63.22	85.21	63.98	85.12

Table 4.9: Results on MobileNet-V2 with GrAB on UCSD-Birds 200 dataset

2*Model	**2*Positions**	**2*#Params**	**2*#FLOPs**	g=1		g=4		g=8		g=16	
				top-1	**top-5**	**top-1**	**top-5**	**top-1**	**top-5**	**top-1**	**top-5**
MNet-V2 (vanilla)	–	3.4M	300M	**top-1** = 61.81, **top-5** = 86.88							
MNet-V2 + GrAB	13	3.28	277.51	61.63	86.58	61.8	86.68	62.05	87.3	60.31	86.63
MNet-V2 + GrAB	16	3.08	284.57	62.60	88.20	62.90	87.30	63.01	88.00	61.70	87.40
MNet-V2 + GrAB	(9, 13)	3.23	267.40	61.73	87.08	61.73	86.86	62.66	87.86	62.08	88.11
MNet-V2 + GrAB	(16, 17)	2.77	269.15	63.28	88.86	64.30	89.58	64.10	88.58	62.48	88.41
MNet-V2 + GrAB	(14, 17)	3.17	255.01	62.86	87.88	62.50	88.03	64.33	89.31	61.67	87.18
MNet-V2 + GrAB	(13, 14, 16, 17)	2.54	224.16	63.20	88.86	63.53	88.63	62.75	88.18	62.50	88.56

Table 4.10: Results on MobileNet-V2 with GrAB on Stanford Dogs dataset

4.3 Attention visualization

In this section, we show the effectiveness of our approach of compute-efficient feature re-distribution, through the human-interpretable visual explanation. For this purpose, we use the Grad-Cam++ [61] which is a generalized formulation of Grad-CAM [62] and provides heatmaps with better object completeness. We apply the Grad-Cam++ visualization on MobileNet-V1 and MobileNet-V2 using images from Caltech-UCSD Birds 200 test set. Figure 4.4 shows the visualization results where the masked regions in the images are important (considered by the networks) for predicting the class.

In Figure 4.4, we compare the visualization results MobileNet-V1 and MobileNet-V2 with the GrAB intergated versions of respective networks. From the visual observations of masks generated by Grad-CAM++, we see that GrAB intergated versions of MobileNets covers the target images better than the baseline networks. This implies that, the integration of GrAB in baseline networks helps in better use of features and increases the representational power of networks. In other words, attention maps in GrAB helps in learning better use of features.

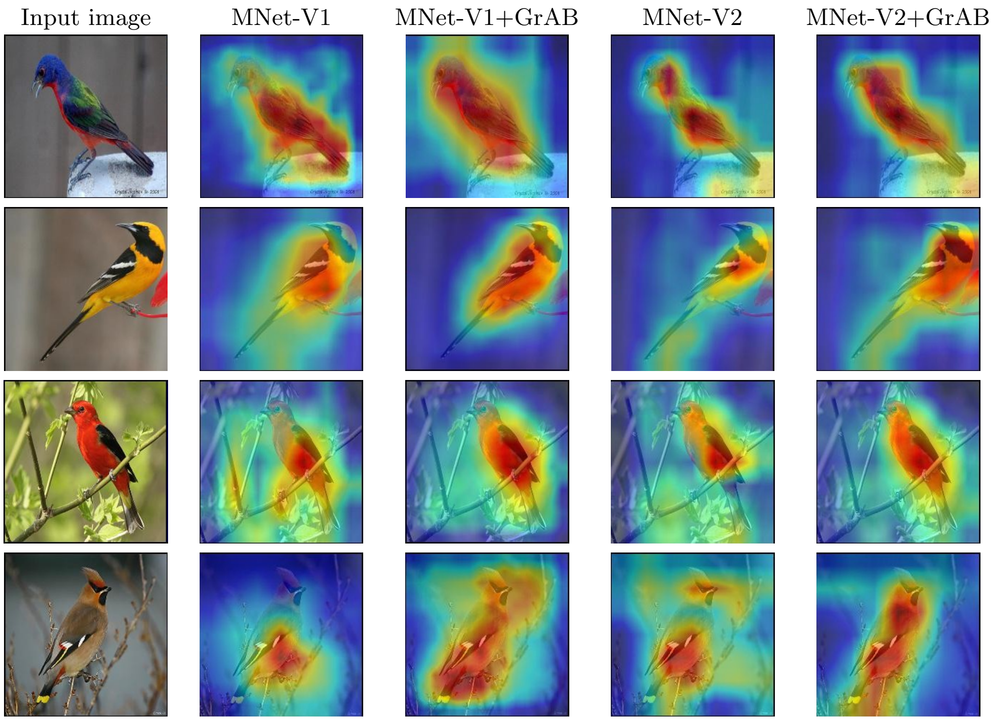


Figure 4.4: **Grad-CAM++ visualization results.** Here the input images are Painted bunting, Hooded Oriole, Scarlet Tanager, and Bohemian Waxwing (in order). MNet-V1 and MNet-V2 are MobileNet-V1 and MobileNet-V2 respectively.

References

[1] C. Y. Chen and K. Grauman. Efficient Activity Detection in Untrimmed Video with Max-Subgraph Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, (2017) 908–921.

[2] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action Tubelet Detector for Spatio-Temporal Action Localization. In IEEE International Conference on Computer Vision (ICCV). Venice, Italy, 2017 3155–3163.

[3] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin. Online Real-Time Multiple Spatiotemporal Action Localisation and Prediction. In International Conference on Computer Vision. Venice, Italy, 2017 3657–3666.

[4] G. Gkioxari and J. Malik. Finding Action Tubes. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, Massachusetts.USA, 2015 759–768.

[5] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In IEEE International Conference on Computer Vision (ICCV). Santiago, Chile, 2015 3164–3172.

[6] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision* 104, (2013) 154–171.

[7] S. Saha, G. Singh, and F. Cuzzolin. Tube Convolutional Neural Network (T-CNN) for Action Detection in Videos. In IEEE International Conference on Computer Vision (ICCV). Venice, Italy, 2017 4424–4433.

[8] S. Saha, G. Singh, M. Sapienza, P. Torr, and F. Cuzzolin. Deep Learning for Detecting Multiple Space-Time Action Tubes in Videos. In Proceedings of the British Machine Vision Conference (BMVC). York, UK, 2016 58.1–58.13.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In European Conference Computer Vision (ECCV). Amsterdam, The Netherlands, 2016 21–37.

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems 25. Montreal, Canada, 2015 91–99.

[11] H. Zhu, R. Vial, and S. Lu. TORNADO: A Spatio-Temporal Convolutional Regression Network for Video Action Proposal. In IEEE International Conference on Computer Vision (ICCV). Venice, Italy, 2017 5813–5821.

[12] E. G. J. van Gemert, M. Jain and C. Snoek. APT: Action localization proposals from dense trajectories. In Proceedings of the British Machine Vision Conference (BMVC). Swansea, UK, 2015 .

[13] W. Chen and J. J. Corso. Action Detection by Implicit Intentional Motion Clustering. In IEEE International Conference on Computer Vision (ICCV). Santiago, Chile, 2015 3298–3306.

[14] M. Jain, J. van Gemert, H. Jegou, P. Bouthemy, and C. G. Snoek. Action Localization with Tubelets from Motion. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbus, Ohio, USA, 2014 740–747.

[15] M. Jain, J. C. van Gemert, H. Jégou, P. Bouthemy, and C. G. M. Snoek. Tubelets: Unsupervised Action Proposals from Spatiotemporal Super-Voxels. *International Journal of Computer Vision* 124, (2017) 287–311.

[16] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-Temporal Object Detection Proposals. In Europen Conference on Computer Vision (ECCV). Zurich,Switzerland, 2014 .

[17] G. Yu and J. Yuan. Fast Action Proposals for Human Action Detection and Search. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, Massachusetts.USA, 2015 1302–1311.

[18] S. A. Klaser, M. MarszaekC. and A. Zisserman. Human focused action localization in video. In Trends and Topics in Computer Vision,(TTCV). Santiago, Chile, 2012 219233.

[19] D. Tran and J. Yuan. Max-Margin Structured Output Regression for Spatio-Temporal Action Localization. In Advances in Neural Information Processing Systems 25. Harrahs and Harveys, Lake Tahoe,USA, 2012 350–358.

[20] T. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN Models for Fine-Grained Visual Recognition. In 2015 IEEE International Conference on Computer Vision (ICCV). 2015 1449–1457.

[21] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. Aˆ2-Nets: Double Attention Networks. In Advances in Neural Information Processing Systems 31, 352–361. 2018.

[22] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The Building Blocks of Interpretability. *Distill* Https://distill.pub/2018/building-blocks.

[23] C. Olah, A. Mordvintsev, and L. Schubert. Feature Visualization. *Distill* Https://distill.pub/2017/feature-visualization.

[24] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds., Computer Vision – ECCV 2014. Springer International Publishing, Cham, 2014 818–833.

[25] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of *Proceedings of Machine Learning Research*. PMLR, Lille, France, 2015 2048–2057.

[26] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua. SCA-CNN: Spatial and Channel-Wise Attention in Convolutional Networks for Image Captioning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017 .

[27] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual Attention Network for Image Classification. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017 .

[28] X. Wang, R. Girshick, A. Gupta, and K. He. Non-Local Neural Networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018 .

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is All you Need. In Advances in Neural Information Processing Systems 30, 5998–6008. 2017.

[30] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In The European Conference on Computer Vision (ECCV). 2018 .

[31] J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018 .

[32] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews* 34, (2004) 334–352.

[33] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong. End-to-End Dense Video Captioning with Masked Transformer. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, Utah, USA, 2018 8739–8748.

[34] R. Hou, C. Chen, and M. Shah. Tube Convolutional Neural Network (T-CNN) for Action Detection in Videos. In IEEE International Conference on Computer Vision (ICCV). Venice, Italy, 2017 5822–5831.

[35] X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In European Conference on Computer Vision (ECCV). Amsterdam, Netherlands, 2016 .

[36] I. Laptev, I. Inria, C. Beaulieu, and R. Cedex. On Space-Time Interest Points. *Int. Journal of Computer Vision (IJCV)* 64, (2005) 107–123.

[37] N. Li, H. Guo, D. Xu, and X. Wu. Multi-scale Analisis of Contextual Information within Spatio-temporal Video Volumes for Anomaly Detection. In Proc. IEEE Int. Conf. Image Processing (ICIP). Paris, France, 2014 2363–2367.

[38] W. Sultani, C. Chen, and M. Shah. Real-world Anomaly Detection in Surveillance Videos. *CVPR* .

[39] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning* 20, (1995) 273–297.

[40] M. Gönen and E. Alpaydn. Multiple kernel learning algorithms. *Journal of Machine Learning Research* 12, (2011) 2211–2268.

[41] C. Xu, D. Tao, and C. Xu. A Survey on Multi-view Learning. *CoRR* abs/1304.5634.

[42] C. Xu, D. Tao, and C. Xu. Multi-View Learning with Incomplete Views. *IEEE Transactions on Image Processing* 24, (2015) 5812–5825.

[43] C. Cortes, M. Mohri, and A. Rostamizadeh. Multi-class classification with maximum margin multiple kernel. *Proceedings of the International Conference on Machine Learning (ICML)* 28, (2013) 46–54.

[44] T. Gärtner, P. A. Flach, and S. Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In Proc. of the Computational Learning Theory and Kernel Machines. Washington, DC, USA, 2003 129–143.

[45] V. Escorcia, C. D. Dao, M. Jain, B. Ghanem, and C. Snoek. Guess Where? Actor-Supervision for Spatiotemporal Action Localization. *CoRR* abs/1804.01824.

[46] .

[47] S. Z. C. S. H. Jhuang, J. Gall and M. Black. Towards understanding action recognition. Santiago, Chile, 2013 .

[48] L. Wang, Y. Qiao, X. Tang, and L. V. Gool. Actionness Estimation Using Hybrid Fully Convolutional Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA, 2016 2708–2717.

[49] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. VideoLSTM convolves, attends and flows for action recognition. *Computer Vision and Image Understanding* 166, (2017) 41–50.

[50] P. Mettes, C. G. Snoek, , and S.-F. Chang. Localizing actions from video labels and pseudo-annotations. In Proceedings of the British Machine Vision Conference (BMVC). London, UK, 2017 .

[51] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016 .

[52] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25, 1097–1105. 2012.

[54] M. Lin, Q. Chen, and S. Yan. Network In Network. *CoRR* .

[55] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, (1998) 2278–2324.

[56] S. Zagoruyko and N. Komodakis. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. 2017 .

[57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch .

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, (2015) 211–252.

[59] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology 2010.

[60] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel Dataset for Fine-Grained Image Categorization. In First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition. Colorado Springs, CO, 2011 .

[61] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). 2018 839–847.

[62] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In The IEEE International Conference on Computer Vision (ICCV). 2017 .