Hanoi University of Science and Technology

Master Thesis

Human Action Recognition using Deep Learning and Multiview Discriminant Analysis

Submitted by: Tran Hoang Nhat

Date & place of birth: December 28, 1996, Hanoi

Submission date: October 20, 2020

Advisor: Assoc. Prof. Dr. Tran Thi Thanh Hai

Faculty: School of Electronics and Telecommunications

Abstract

Human action recognition (HAR) under different viewpoints is one of the most critical requirements for practical deployment.

Acknowledgements

This thesis would not have been possible without the help of many people. First of all, I would like to express my gratitude to my primary advisor, Prof. Tran Thi Thanh Hai, who guided me throughout this project. I would like to thank Prof. Le Thi Lan and Prof. Vu Hai for giving me deep insight, valuable recommendations and brilliant idea.

I am grateful for my time spent at MICA International Research Institute, where I learnt a lot about research and enjoyed a very warm and friendly working atmosphere. In particular, I wish to extend my special thanks to PhD candidate. Nguyen Hong Quan and Dr. Doan Huong Giang who directly supported me.

Finally, I wish to show my appreciation to all my friends and family members who helped me finalizing the project.

Kì này mà không ra được trường là tại Samsung hết!
(The author, "Sogno di Liberté")

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
CCA	Canonical Correlation Analysis
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GRU	Gated Recurrent Unit
HAR	Human Action Recognition
HoG	Histogram of oriented Gradient
iDT	improved Dense Trajectories
KCCA	Kernel Canonical Correlation Analysis
kNN	k-Neareast Neighbor
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
MICA	Multimedia, Information, Communication & Applications International Research Institute
MLP	Multilayer Perceptron
MST-AOG	Multiview Spatio-Temporal AND-OR Graph
MvCCA	Multiview Canonical Correlation Analysis
MvCCDA	Multiview Common Component Discriminant Analysis
MvDA	Multiview Discriminant Analysis
MvDA-vc	Multiview Discriminant Analysis with View-Consistency
MvFDA	Multiview Fisher Discriminant Analysis
MvL	Multiview Learning
MvMDA	Multiview Modular Discriminant Analysis
MvML-LA	Multiview Manifold Learning with Locality Alignment
MvPLS	Multiview Partial Least Square
pc-LDA	Pairwise-Covariance Linear Discriminant Analysis
pc-MvDA	Pairwise-Covariance Multiview Discriminant Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network

SOTA	State Of The Art
SSM	Self Similarity Matrix
SVM	Support Vector Machine

1 Introduction

1.1 Motivation

Human action and gesture recognition aims at recognizing an action from a given video clip. This is an attractive research topic, which has been extensively studied over the years due to its broad range of applications from video surveillance to human machine interaction [3, 4]. Within this scope, a very important demand is independence to viewpoint. However, different viewpoints result in various human pose, background, camera motions, lighting conditions and occlusions. Consequently, recognition performance could be dramatically degraded under viewpoint changes.

To overcome this problem, a number of methods have been proposed. View independence recognition such as [5, 6, 7] [8] generally require a careful multi-view camera setting for robust joint estimation. View invariance approach [9, 10] is usually limited by inherent structure of view-invariant features. Recently, knowledge transfer technique is widely deployed for cross-view action recognition, for instance bipartite graph that bridge the semantic gap across view dependent vocabularies [11], AND-OR graph (MST-AOG) for cross-view action recognition [12]. To increase discriminant and informative features, view private features and shared features are both incorporated in such frameworks to learn the common latent space [13, 14]. While existing works for human action and gesture recognition from common viewpoints explored different deep learning techniques and achieved impressive accuracy. In most of aforementioned multi-view action recognition techniques, the features extracted from each view are usually hand-crafted features (i.e improved dense trajectories) [15, 14, 13]. Deep learning techniques, if used, handle knowledge transfer among viewpoints. Deployment of deep features in such frameworks for cross-view scenario is under active investigation.

In parallel with knowledge transfer techniques, building a common space from different views has been addressed in many other works using multi-view discriminant analysis techniques. The first work of this approach was initiated by Canonical Component Analysis (CCA) that tried to find two linear transformations for each view [16].

Various improvements of CCA have been made to take non-linear transformation into account (kernel CCA) [17]. Henceforth, different improvements have been introduced such as MULDA [18], MvDA [19], MvCCA, MvPLS and MvMDA [20], MvCCDA [21]. All of these techniques try to build a common space from different views by maximizing the cross-covariance between views. However, most of these works are still experimented with static images, none of them have been explored with videos. Particularly, their investigation for the case of human action recognition is still actively under taken.

1.2 Objective

Motivated by the two aforementioned under investigation problems, in this thesis, an unified framework is proposed for cross-view action recognition which consists of two main components: individual view feature extraction and latent common space construction. For feature extraction from individual view, I investigate a range of deep neural networks, from 2D CNNs with different pooling strategies (average pooling, temporal attention pooling or using LSTM) to 3D CNNs with two most recent variations (C3D [22] and ResNet-50 3D [23]). These networks have been successfully deployed for human action and gesture recognition in general, but not investigated yet for cross-view recognition scenarios.

For building a latent common space, I am inspired by idea of multi-view discriminant analysis (MvDA). On the one side, this technique has been shown to be efficient for images based tasks, but not deployed for video based tasks and mostly with deep features extracted from videos as input. In addition, the MvDA's objective has no explicit constraint to push class centers away from each other. In this step, I extend the original MvDA by introducing the pairwise-covariance constraint while modifying the optimization model. This helps to make between-classes to be more separated in the common feature space than using the baseline MvDA.

The main contributions of this thesis are summarized as follows:

- Firstly, I improve the MvDA technique by incorporating pairwise-covariance of between-classes that helps to improve the performance. The proposed method is called pc-MvDA. When the number of views is large, the optimization problem is hard to be solved, I then propose a feasible technique to solve optimization problem of pc-MvDA.

- Secondly, various recent neuronal networks for feature extraction are investigated

and incorporated in an unified framework with multiview analysis algorithms for robust cross-view action and gesture recognition.

- Finally, I evaluate the proposed framework on three datasets (IXMAS, MuHAVi and MICAGes). The experimental results show improvement of the proposed method compared to state of the art techniques.

1.3 Thesis Outline

The thesis is structured into 5 chapters:

1 Introduction This chapter. Motivates the work and describes the research goals.

2 Background and Related Works Describes the deep learning based approach for feature extraction, dimensionality reduction and multiview learning algorithms. Also briefly reviews the existing approaches on human action recognition in single and cross-view scenarios and multi-view analysis techniques.

3 Proposed Method Proposes the general architecture and the technical contribution for solving the mentioned research objective.

4 Experiments Reports information regarding experiments: datasets, evaluation protocol, technical setup, results and discussions.

5 Conclusion Summarizes the work, points out the contributions, drawbacks and suggests future research directions.

2 Technical Background and Related Works

2.1 Introduction

2.2 Technical Background

2.2.1 Deep Neural Networks

2.2.1.1 Artificial Neural Networks

Artificial neural networks (ANNs), sometimes a.k.a. multi-layer perceptron (MLP) or feed-forward neural network, are inspired by "real" neural networks - i.e. the human brain and nervous system. They consist of neurons grouped in multiple connected layers, each of which subsequently transformed by an activation function.

Linear Layer The *linear layer*, or generally known as *fully-connected layer*, basically composes of several perceptron units. Mathematically, it is simply a linear function of input features with 2 learnable parameters weight $W = \{\omega_1, \omega_2, ..., \omega_d\}$ as coefficient of multiplication and bias b as additional term, simulating the biological group of d perceptrons:

$$y = W \cdot x + b \tag{2.1}$$

Activation Layer The non-linear *activation layers* are responsible to create complex smooth mappings between the input and the output. They are element-wise operators responsible to squash the value of each element within the boundaries of specified function. Some common activation functions are:

- Sigmoid: $sigmoid\,(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic tangent: $tanh\,(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

- Rectifier: $relu\,(x) = \max(0, x)$
- Swish: $swish\,(x) = x \cdot sigmoid\,(x)$

For classification layer (the last linear layer), the number of neurons is equal to the number of classes to be recognized and a *softmax* operator (Equation (2.2)) is usually applied as activation function to get the probabilities of each classes.

$$\sigma_i = \frac{e^{y_i}}{\sum_{n=1}^{N} e^{y_n}} \tag{2.2}$$

Training Neural network training is usually performed via backpropagation algorithm. This algorithm is based on the calculation of a loss function L which represents the difference between the network output and the expected output. Partial derivatives of the cost $\frac{\partial L}{\partial p_i}$ are calculated with regards to each trainable parameters p_i using the chain rule. Then, each parameter is adjusted accordingly:

$$\Delta p_i = -\eta \frac{\partial L}{\partial p_i} \tag{2.3}$$

where η is called learning rate, which must be choosen carefully to ensure convergence.

Loss functions are usually applied on the last layer. The most common criterion for classification tasks is *Cross Entropy Loss*:

$$L_{CE} = -\frac{1}{N} \sum_{k=1}^{N} \log \frac{e^{W_{class(x_k)} x_k + b_{class(x_k)}}}{\sum_{i=1}^{c} e^{W_i x_k + b_i}} \tag{2.4}$$

where N is number of samples, c is number of classes; W and b are parameters of classification layer, W_i and b_i denote the i^{th} column of the weight W and bias b respectively; x_k denotes the deep feature of k^{th} sample belonging to the $class(x_k)$ class.

2.2.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs), inspired from the biological process in the visual cortex of animals, have emerged as the most efficient approach for image recognition and classification tasks. They are able to extract and aggregate highly abstract information from images and videos. As a result of huge research and engineering efforts, the effectiveness and performance of such algorithms have considerably improved,

outperforming handcrafted methods for visual information embedding and becoming the state-of-the-art in image and video recognition.

There are 5 main building blocks in architecture of a modern CNN:

Convolution Layer The *convolutional layer* implements sliding kernel on the input tensor and for every position perform the summation of element-wise multiplication between sliced input and learnable weight matrices to compute the output. It can have multiple numbers of kernels such that more features from the input tensor can be extracted.

The mathematical operation performed by each 2D convolutional kernel is:

$$o_{i,j}^{l} = \sum_{c=0}^{Ch} \sum_{h=0}^{K_H} \sum_{w=0}^{K_W} \left(\omega_{c,h,w}^{l} \cdot x_{c,i+h,j+w} \right) + b^{l} \tag{2.5}$$

2D convolution is limited to spatial data and requires extra steps to manipulate temporarily continuous sequence of images. On the other hand, 3D convolution could intrinsically comprehend and establish abstract spatio-temporal relationship in 3D input tensor. The mathematical operation performed by each 3D convolutional kernel is:

$$o_{i,j,k}^{l} = \sum_{c=0}^{Ch} \sum_{d=0}^{K_D} \sum_{h=0}^{K_H} \sum_{w=0}^{K_W} \left(\omega_{c,d,h,w}^{l} \cdot x_{c,i+d,j+h,k+w} \right) + b^{l} \tag{2.6}$$

Batch Normalization Layer The *batch normalization layer* introduced in [24] is a pervasive component in modern CNN architectures. It generally escorts after every convolution layer and before an activation layer, responsible for bringing all the pre-activated features to the same scale. The mathematical equation is as follow:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} \cdot \gamma + \beta \tag{2.7}$$

where $\mathrm{E}[x]$ and $\mathrm{Var}[x]$ stands for mean and standard deviation calculated per-dimension over the input mini-batches x; γ and β are learnable parameters and ϵ is a small number added to the denominator to ensure numerical stability.

Activation Layer Similar to activiation in ANNs, *activation layer* in CNNs are element-wise operators that apply for each pixel of input tensor.

Pooling Layer The *pooling layer* is usually inserted after one or a group of convolutional layers. The purpose of pooling is to progressively decrease the size of the elaborated data and make sure that only the most relevant features will be forwarded to the next layers. It follows the sliding kernel principle of convolution, but uses a much simpler operator without learnable parameter, such as:

- Max Pooling: Select the pixel with maximum value.
- Min Pooling: Select the pixel with minimum value.
- Average Pooling: Compute the mean of the sliced input pixels.

A special class of pooling layer is called *global pooling*, which has flexible filter sizes and shifts exact to the shape of input tensor, squeezing each channel to a single scalar value. This type of pooling is generally used in the very end of a large-scale CNN, transforming high-level features of possibly unascertained shapes to a single vector of fixed length. After that, the output feature vector can be forwarded to further linear layers without being flattened or perform classification directly.

Linear Layer Similarly, *linear layer* is the essential building block of classical ANNs, but might be optional in CNNs in case of fully convolutional neural networks.

2.2.1.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a feed-forward neural network that takes previous time steps into account. The input of RNNs is a sequentially ordered collection of samples. Therefore, they excel in tasks in which order is important, e.g. time series forecasting, natural language processing. Relating to the research topic of this thesis, they can be used to handle chronical relationship of high-level representation of frames extracted from videos.

In practice, either Long-Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) are used instead. The main difference is that information that is deemed important is allowed to pass on to later time-steps without too much interference from hidden dot products and activation functions.

Long Short-Term Memory The LSTM architecture, contrary to regular RNNs, has an additional hidden state that is never directly outputted (see Figure 2.1). This additional hidden state can then be used by the network solely for remembering previous relevant information. Instead of having to share its "memory" with its output, these

values are now separate. During the training process, an LSTM learns what should be remembered for the future and what should be forgotten, which is achieved by using its internal weights.

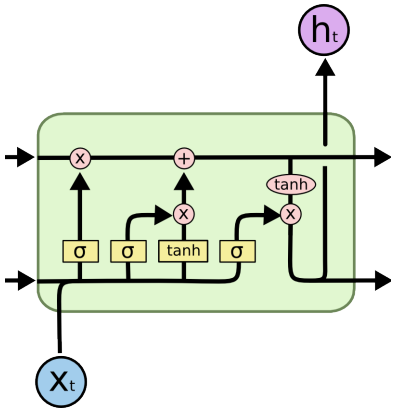


Figure 2.1: A single LSTM cell. From [1]

As can be seen in the Figure 2.1, there are quite a few more parameters in this cell than in a normal RNN cell. The calculation of the output vector and the hidden vector involves several operations. First of all the network determines how much of the hidden state to forget, also called the forget gate. This is done by pushing both the previous iteration's output (c_{t-1}) and the forget gate vector (f_t) through a matrix multiplication, allowing the network to forget values at specific indices in the previous iteration's output vector. f_t can be obtained by using formula in Equation (2.8), where W contains the weights for the input and U contains the weights for the previous iteration's output vector, x_t refers to the input, h_{t-1} to the previous iteration's output vector and b is bias:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.8}$$

The network then determines what to remember from the input vector. This, commonly referred to as the input gate, is done by pushing the previous forget gate's output as well as the input gate through a matrix addition. The output of the input gate (i_t) can be found by using the following formula:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.9}$$

The final hidden state vector (c_t) can then be found by using the previous two results as follows:

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma(W_c x_t + U_c h_{t-1} + b_c) \tag{2.10}$$

where $\circ$ denotes the Hadamard product (where each value at index ij is the product of the values at the indices ij in the two input matrices). This vector is then passed on to the next iteration. Now the output gate vector o_t and the output state h_t can be optained:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.11}$$

$$h_t = o_t \circ \sigma(c_t) \tag{2.12}$$

This results in a version of an RNN that is able to remember more and is more liberal in choosing what information it wants to keep in the hidden state and what it wants to discard. This makes LSTM networks better suited for tasks involving series of data and become the predominant RNN architecture.

Gated Recurrent Units Another RNN architecture is the GRU, introduced in [**?**]. This architecture combines the input and forget gates into a single so-called "update gate" and also merges the cell state and hidden state (see Figure 2.2). The calculation of the merged output vector once again consists of several operations. The network first computes the "reset gate" r_t using the following function, where W_r are the weights for the reset gate and $[h_{t-1}, x_t]$ signifies the concatenation of h_{t-1} and x_t:

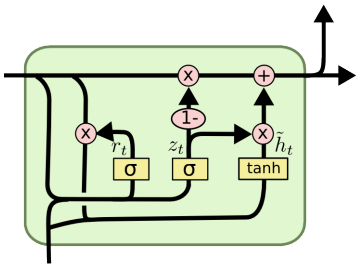


Figure 2.2: A single GRU variation cell. From [1]

$$r_t = \sigma\left(W_r[h_{t-1}, x_t]\right) \tag{2.13}$$

After this, the "update gate" z_t is computed as follows, where W_z holds the weights of the update gate:

$$z_t = \sigma\left(W_z[h_{t-1}, x_t]\right) \tag{2.14}$$

The output vector h_t (representing both the cell's output and its state) can then be computed by the following formula:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h_t} \tag{2.15}$$

where $\tilde{h_t} = \tanh(W * [r_t * h_{t-1}, x_t])$.

Bidirectional RNNs It should be mentioned that when RNNs are used, they are often wrapped with a bidirectional layer. This simply reverts the input sequence and enters the sequence in both the original and the reverse direction to two separate RNNs, usually LSTM or GRU. When processing the entry $x^{(j)}$, only the entries $t < j$ are known. However, tokens later in the sequence might have an impact on the previous outputs of the model. Bidirectional RNNs are able to capture patterns that are overlooked by regular RNNs.

2.2.2 Dimensionality Reduction Algorithms

Dimensionality reduction techniques are important in many applications related to machine learning. They aim to find low-dimensional embedding that should preserve sufficient information from the original dimension. Let's define $X = \{\boldsymbol{x}_{ik} | i = (1, .., c); k = (1, .., n_i)\}$ as training samples where $\boldsymbol{x}_{ik} \in R^d$ is the k^{th} sample of the i^{th} class, d is the original dimension of data, m is the desired dimension of data after transformation such that $m < d$.

2.2.2.1 Linear discriminant analysis

The linear discriminant analysis (LDA) technique is developed to linearly transform the features into a lower dimensional space where the ratio of the between-class variance to the within-class variance is maximized, thereby guaranteering the optimal class separability. The projection results of X on the lower dimensional space is denoted by $Y = \{\boldsymbol{y}_{ik} = w^T \boldsymbol{x}_{ik} | i = (1, .., c); k = (1, ..., n_i)\}$. $\boldsymbol{S}_B^y$ and $\boldsymbol{S}_W^y$ are computed as follows:

$$\boldsymbol{S}_W^y = \sum_{i=1}^{c}\sum_{k=1}^{n_i}(y_{ik} - \boldsymbol{\mu}_i)(y_{ik} - \boldsymbol{\mu}_i)^T \tag{2.16}$$

$$\boldsymbol{S}_B^y = \sum_{i=1}^{c} n_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{2.17}$$

where $\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{k=1}^{n_i}\boldsymbol{y}_{ik}$ is the mean of all samples of the i^{th} class in the lower dimensional space; $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n_i}\boldsymbol{y}_{ik}$ is the mean of all samples of all classes; $n = \sum_{i=1}^{c} n_i$ is the total number of data samples. The between-class $\boldsymbol{S}_B^y$ and within-class $\boldsymbol{S}_W^y$ covariance matrices in new dimension are not known yet but can be formulated as the linearly transformed versions of their counterparts $\boldsymbol{S}_B^x$ and $\boldsymbol{S}_W^x$ in original dimension.

$$\boldsymbol{S}_W^y = \omega^T \boldsymbol{S}_W^x \omega \tag{2.18}$$

$$\boldsymbol{S}_B^y = \omega^T \boldsymbol{S}_B^x \omega \tag{2.19}$$

$\boldsymbol{S}_B^x$ and $\boldsymbol{S}_W^x$ are easily calculable as follow:

$$\boldsymbol{S}_W^x = \sum_{i=1}^{c}\sum_{k=1}^{n_i}(x_{ik} - \boldsymbol{\mu}_i^{(x)})(x_{ik} - \boldsymbol{\mu}_i^{(x)})^T \tag{2.20}$$

$$\boldsymbol{S}_B^x = \sum_{i=1}^{c} n_i(\boldsymbol{\mu}_i^{(x)} - \boldsymbol{\mu}^{(x)})(\boldsymbol{\mu}_i^{(x)} - \boldsymbol{\mu}^{(x)})^T \tag{2.21}$$

Then the objective function is formulated by a Rayleigh quotient:

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \frac{trace(\boldsymbol{S}_B^y)}{trace(\boldsymbol{S}_W^y)} = \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \frac{trace(\omega^T \boldsymbol{S}_B^x \omega)}{trace(\omega^T \boldsymbol{S}_W^x \omega)} \tag{2.22}$$

The Fisher's criterion in Equation (2.22) can be reformulated as:

$$\boldsymbol{S}_W^x \omega = \lambda \boldsymbol{S}_B^x \omega \tag{2.23}$$

where λ represents the eigenvalues of the transformation matrix ω. The analytical solution of $\boldsymbol{\omega}^*$ is optained by calculating the eigenvectors $V = \{v_1, v_2, ..., v_m\}$ sorted by the scalar values of corresponding eigenvalues $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$ of the matrix $\boldsymbol{S} = {\boldsymbol{S}_W^x}^{-1}\boldsymbol{S}_B^x$.

2.2.2.2 Pairwise-covariance linear discriminant analysis

Pairwise-covariance linear discriminant analysis (pc-LDA) is an extension of LDA introduced in [25] that overcomes it's drawbacks by formulating pairwise distances between pairs of classes. The pairs of a and b classes are regarded as two Gaussian distributions $\mathcal{N}_a(\mu_a, \boldsymbol{S}^y_{Wa}), \mathcal{N}_b(\mu_b, \boldsymbol{S}^y_{Wb})$ and the objective distance between two classes is defined as their Kullback-Leibler divergence [26]:

$$D_{KL}\left(\mathcal{N}_a \parallel \mathcal{N}_b\right) = \frac{1}{2}\left(\mu_a - \mu_b\right)^T \left(\boldsymbol{S}^y_{Wab}\right)^{-1}\left(\mu_a - \mu_b\right), \tag{2.24}$$

where $\boldsymbol{S}^y_{Wab}$ is pairwise covariance matrix (Equation (2.26)), calculated as the β parameterized convex sum of global within-class scatter matrix $\boldsymbol{S}^y_W$ used in LDA with the within-class scatter matrix of each class $\boldsymbol{S}^y_{Wi}$ (Equation (2.25)). The author theorizes it would better represent the data distribution within two classes.

$$\boldsymbol{S}^y_{Wi} = \sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}\left(y_{ik} - \mu_i\right)\left(y_{ik} - \mu_i\right)^T \tag{2.25}$$

$$\boldsymbol{S}^y_{Wab} = \beta\frac{n_a\boldsymbol{S}^y_{Wa} + n_b\boldsymbol{S}^y_{Wb}}{n_a + n_b} + \left(1 - \beta\right)\boldsymbol{S}^y_W \tag{2.26}$$

where n_a and n_b are number of samples belonging to class a and b. The final objective is properly weighted to focus on classes with more samples:

$$\min_{\boldsymbol{\omega}} J = \sum_{a=1}^{c}\sum_{b=a+1}^{c}\frac{n_a n_b}{\left[2D_{KL}\left(\mathcal{N}_a \parallel \mathcal{N}_b\right)\right]^q}, \quad s.t.\ \omega^T\omega = \boldsymbol{I} \tag{2.27}$$

here $q \geq 1$ is a hyper-parameter that controls how much the pairs of classes with smaller objective distances are biased over the others.

The new model of pc-LDA is solved with a variant of gradient descent described in [25], where $\nabla J\left(\omega\right)$ is computed and ω is updated as Equation (2.28) in order to enforce ω on the Stiefel manifold. Every several iterations, due to numerical error, the learnt transformation is unitarized $\omega \leftarrow \omega\left(\omega^T\omega\right)^{-\frac{1}{2}}$ to ensure that the constraint $\omega^T\omega = \boldsymbol{I}$ is satisfied.

$$\omega \leftarrow \omega - \eta\left(\nabla J - \omega[\nabla J]^T\omega\right) \tag{2.28}$$

2.2.3 Multiview Learning Algorithms

The motivation of multiview learning (MvL) algorithms is to construct a common low-dimensional embedding that should preserve sufficient information or even be more informative than each individual view.

Let's define $X = \{\boldsymbol{x}_{ijk} | i = (1, .., c); j = (1, .., v); k = (1, .., n_{ij})\}$ as samples from v views where $\boldsymbol{x}_{ijk} \in R^{d_j}$ is the k^{th} sample from the j^{th} view of the i^{th} class, d_j is the dimensions of data at the j^{th} view. Here $\boldsymbol{x}_{ijk}$ is a feature vector extracted from the k^{th} sample from the j^{th} view of the i^{th} class. Different methods for features extraction from single view have been presented in previous sections.

2.2.3.1 Multiview discriminant analysis

Multiview discriminant analysis (MvDA) is an extension of LDA for multi-view scenario [19]. It tries to determine a set of v linear transformations to project all action samples from each view $j = (1, .., v)$ to a common space. The projection results of X on the common space is denoted by $Y = \{\boldsymbol{y}_{ijk} = w_j^T \boldsymbol{x}_{ijk} | i = (1, .., c); j = (1, .., v); k = (1, ..., n_{ij})\}$. The common space is built by maximizing the between-class variation $\boldsymbol{S}_B^y$ while minimizing the within-class variation $\boldsymbol{S}_W^y$ from all views. $\boldsymbol{S}_B^y$ and $\boldsymbol{S}_W^y$ are computed as follows:

$$\boldsymbol{S}_W^y = \sum_{i=1}^{c} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} (y_{ijk} - \boldsymbol{\mu}_i)(y_{ijk} - \boldsymbol{\mu}_i)^T \tag{2.29}$$

$$\boldsymbol{S}_B^y = \sum_{i=1}^{c} n_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{2.30}$$

where $\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \boldsymbol{y}_{ijk}$ is the mean of all samples of the i^{th} class from all views in the common space; $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} \boldsymbol{y}_{ijk}$ is the mean of all samples of all classes from all views in the common space; $n = \sum_{i=1}^{c} n_i$ is the total data samples from all views.

In order to separate the unknown transformation vectors, the between-class and within-class scatter matrices are reformulated as:

$$\boldsymbol{S}_W^y = W^T X \left(\boldsymbol{I} - \boldsymbol{E}\right) X^T W \tag{2.31}$$

$$\boldsymbol{S}_B^y = W^T X \left(\boldsymbol{E} - \frac{1}{n} \mathbb{1}\right) X^T W \tag{2.32}$$

where $W = \{\omega_1, \omega_2, ..., \omega_v\}$ is concatenation of transformation vectors of all views; $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is identity matrix; $\mathbb{1} \in \mathbb{R}^{n \times n}$ is matrix of ones; $\boldsymbol{E} \in \mathbb{R}^{n \times n}$ is a square matrix whose elements satisfy:

$$\boldsymbol{E}_{kl} = \left\{ \begin{array}{ll} \frac{1}{n_i}, & \text{if } class(x_k) = class(x_l) = i \\ 0, & \text{otherwise} \end{array} \right\} \tag{2.33}$$

Then the objective function is formulated by a Rayleigh quotient:

$$(\boldsymbol{\omega}_1^*, \boldsymbol{\omega}_2^*, ..., \boldsymbol{\omega}_v^*) = \underset{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, ..., \boldsymbol{\omega}_v}{\operatorname{argmax}} \frac{trace(S_B^y)}{trace(S_W^y)} \tag{2.34}$$

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmax}} \frac{trace(W^T[X\left(\boldsymbol{E} - \frac{1}{n}\mathbb{1}\right)X^T]W)}{trace(W^T[X\left(\boldsymbol{I} - \boldsymbol{E}\right)X^T]W)} \tag{2.35}$$

According to [27], the problem satisfies the optimization form of generalized eigenvalue problem and could be analytically solved through eigenvalue decomposition. The concatenated $\boldsymbol{W}^* = \{\boldsymbol{\omega}_1^*, \boldsymbol{\omega}_2^*, ..., \boldsymbol{\omega}_v^*\}$ could be computed using the same procedure as $\boldsymbol{\omega}^*$ in LDA. Similarly, MvDA can be used as a dimensionality reduction algorithm by choosing m eigenvectors corresponding to m leading eigenvalues.

2.2.3.2 Multiview discriminant analysis with view-consistency

In [27], the authors observed that as multiple views correspond to the same objects, there should be some correspondence between multiple views. They then introduce a view consistency constraint into the objective function, that means if X_j, X_r are observed at j^{th} and r^{th} views, there exists a certain transformation $\boldsymbol{R}$ such that $X_j = \boldsymbol{R}X_r$. As a result, the transformations obtained from two views (i.e. the projection of features extracted from singe view to common view) should have similar relationship: $\omega_j = \boldsymbol{R}\omega_r$. Let's define β_i that captures the structure of the transformation ω_i.

$$\omega_i = X_i\boldsymbol{\beta}_i \tag{2.36}$$

Then the β_j and β_r capturing the structures of two transformations of two views j and r should be identical $\beta_j = \beta_r$.

Generalizing to v views, suppose that $\boldsymbol{\beta}_j, j = (1, .., v)$ captures the structures of v transformations w_j. Following the above observation, the $\boldsymbol{\beta}_r, r = (1, .., v)$ should

resemble mutually. That means the similarity between the pair of $\boldsymbol{\beta}_j$ and $\boldsymbol{\beta}_r$ should be minimized.

$$\sum_{j,r=1}^{v} ||\boldsymbol{\beta_j} - \boldsymbol{\beta_r}||_2^2 \tag{2.37}$$

From Equation (2.36), we have:

$$\boldsymbol{\beta}_i = \left(X_i^T X_i\right)^{-1} X_i^T \omega_i \triangleq \boldsymbol{P}_i w_i \tag{2.38}$$

Replacing in Equation (2.37) we can reformulate it as:

$$\sum_{j,r=1}^{v} ||\boldsymbol{\beta_j} - \boldsymbol{\beta_r}||_2^2 = trace\left(W^T \boldsymbol{P}^T \left(2((v-1)\boldsymbol{I} - \mathbb{1})\right) \boldsymbol{P} W\right) \tag{2.39}$$

where $\boldsymbol{P} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_v\}$; $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is identity matrix; $\mathbb{1} \in \mathbb{R}^{n \times n}$ is matrix of ones. This term is called in [27] *view consistency* and will be added to the denominator of Equation (3.7)

$$(\boldsymbol{\omega}_1^*, \boldsymbol{\omega}_2^*, ..., \boldsymbol{\omega}_v^*) = \underset{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, ..., \boldsymbol{\omega}_v}{\text{argmax}} \frac{trace(S_B^y)}{trace(S_W^y) + \alpha \sum_{j,r=1}^{v} ||\boldsymbol{\beta_j} - \boldsymbol{\beta_r}||_2^2} \tag{2.40}$$

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\text{argmax}} \frac{trace(W^T [X \left(\boldsymbol{E} - \frac{1}{n}\mathbb{1}\right) X^T] W)}{trace(W^T [X \left(\boldsymbol{I} - \boldsymbol{E}\right) X^T + 2\alpha \boldsymbol{P}^T \left((v-1)\boldsymbol{I} - \mathbb{1}\right) \boldsymbol{P}] W)} \tag{2.41}$$

This optimization problem could also be analytically solved by relaxing to the trace ratio optimization problem as Equation (3.7). In the Equation (2.41), α is an empirically chosen parameter that puts a weight on the view-consistency assumption. When $\alpha = 0$, the MvDA-vc becomes the original MvDA.

2.3 Related Works

2.3.1 Human action and gesture recognition

Action recognition has been an attractive research topic since the last decade [4]. Early methods represented human actions by extracting 2D/3D key-points such as Harris-3D, SIFT-3D, HOG-3DHOF [28], ESURF [29] then computed a descriptor from the detected key-points. Action representation by a set of key-points could loose the temporal information. Therefore, Wang and Schmid in [30] proposed a feature named

improved dense trajectories (iDT) that densely sample and track optical flow points along trajectories. iDT has become state-of-the-art hand-crafted features and widely used for many video-based tasks. However, when working with large-scale datasets, iDT becomes intractable on due to its expensive computational cost and poor performance.

To work with more challenging datasets, effective action recognition approaches rely on powerful learning methods, particularly the deep learning techniques. Early works applied 2D CNN on frames of video sequence and then aggregated the information using pooling techniques [31]. To exploit the temporal information, different architectures such as LSTM with the internal mechanisms called gates that can deal with short-term memory are proposed [32]. Recently, instead of using 2D convolutional operators, different 3D CNN have been proposed [33, 22], [34]. Besides, to boost the recognition performance, different approaches tried to combine multiple streams [35, 36], [37] or to combine both multiple features [38],[39].

These aforementioned approaches focus on single view action recognition, cross-view action recognition is more challenging and requires additional techniques to be taken into account. Junejo et al. in [9] proposed a descriptor, namely self-similarity matrix (SSM), which is an exhaustive table of distances between image features taken by pair from the image sequences. Liu et al. [11] employed cuboids extracted from each video and BoW model to build video descriptor for each single view. Then, a bipartite graph is built to model two view-dependent vocabularies. Li et al. [40] described each video by concatenating spatio-temporal interest-point-based descriptor with shape flow descriptor. Then, to deal with cross-view, they construct 'virtual views', each is a linear transformation between action descriptors from one viewpoint and those from another. The method in [41, 42] employed the same video representation manner as in [40]. However, a transferable dictionary between source and target view has been learnt to force features of the same action extracted from two views having the same sparse representation.

Previous cross-view action recognition techniques usually connect source and target views with a set of linear transformations, that are unable to capture the non-linear manifolds on which real actions lie. In [15], the authors find a shared high-level non-linear virtual path that connects multiple source and target views to the same canonical view. This virtual path is learnt by a deep neural network. In [13], a deep learning technique that stacks multiple layers of feature learners is designed to incorporate both private and shared view features. In [14], the authors concatenated both private and shared view features and learnt transferable dictionary pair from a pair of views. In

[43], the authors proposed a framework to jointly learn a a view-invariant transfer dictionary and a view-invariant classifier using synthetic data during the pre-training phase to extract view-invariance between 3D and 2D videos.

2.3.2 Multiview analysis and learning techniques

As many objects in the real-world can be observed from different viewpoints, to exploit the consensual and complementary information between different views, Multiview learning (MvL) techniques are employed. MvL is a strategy for fusing data from different sources or subsets.

Canonical Correlation Analysis (CCA) [44] can be considered as the first approach of MvL with the aim to find pairs of projections for two views so that the correlations between these views are maximized. As CCA can only handle the linear correlation, Kernel CCA (KCCA) was proposed to take non-linear correlation relationship of data into account [45]. However, both CCA and KCCA are unsupervised methods and can not leverage the label information. In [46], a supervised approach named Multiview Fisher discriminant analysis (MvFDA) was proposed for binary classification problem. All of aforementioned methods are only applicable for two views problem.

To extend to multiple view cases, a natural extension is to maximize the sum of the pairwise correlations. In a general case, it would be better to build a common shared feature space that captures latent information of the object from all observed views. For this propose, Multiview CCA (MvCCA) is proposed in 2010 to build a common feature space of all views [47]. However, MvCCA did not consider the discrepancy information but only maximizing the correlation between every two views, so that it may be ineffective for classification across views.

In [19], Multi-view discriminant analysis (MvDA), an extension of linear discriminant analysis (LDA) for multi-view problem was proposed. MvDA tries to optimize jointly view correlation, intra-view and inter-view discriminability. An extension of MvDA which considers view-consistency was also introduced and achieved significant performance improvement.

In [48], the authors proposed multiview manifold learning with locality alignment (MvML-LA) framework to realize manifold learning under multi-view scenario.

Most recently, [21] proposed Multiview Common Component Discriminant Analysis (MvCCDA) technique that both integrates supervised information and local geometric information into the common component extraction process. This helps to effectively handle view discrepancy, discriminability and non-linearity in a joint manner.

2.4 Summary

3 Proposed method

3.1 Introduction

3.2 General framework

We propose a framework for cross-view action recognition as illustrated in Figure 3.1. It composes of two phases: training phase and recognition phase.

- **Training phase**: Suppose we have v views. The training phase consists of three main steps:
  1. *Feature extraction at separated view*: All training samples from each separated view will be passed though a feature extraction step. Different deep techniques will be investigated, including 2D CNNs based feature extraction combined with aggregation techniques and 3D CNNs based feature extraction, to output the final descriptor for each video sample. Let's call the features extracted at this step as features at single views or separated view features.
  2. *Common space construction*: This step builds a common feature space so that samples belonging to the same class will be close to each other even they are captured from different viewpoints. It takes all separated view features extracted from training set from the previous step and find a set of v linear transformations $(\omega_1, \omega_2, ..., \omega_v)$ that minimize within-class variation while maximizing between-class variation of features in the projected space (common space).
  3. *Training classifier*: Once v transformations have been computed, the projected features in the common space of each view will be utilized to train a simple predictive model F (i.e. kNN).
- **Recognition phase**. Multi-view recognition consists of two following steps:

1. *Feature extraction* Features of the testing sample x_j from the view V_j are extracted. This feature will be projected in the pre-built common space by the corresponding transformation ω_j. The projected feature is denoted as $y_j = \omega_j^T * x_j$.
2. *Class prediction*: The projected feature y_j will be passed into the classifier $F(y_j)$ that outputs the label of action.

3.3 Feature extraction at individual view using deep learning techniques

3.3.1 2D CNN based clip-level feature extraction

First, the ResNet-50 CNN [49] is used as a 2D CNN network to extract spatial features of each frame in video. Figure 3.2 illustrates the architecture of ResNet-50 which composes of five convolutional blocks stacked on top of each other. The network is pre-trained on ImageNet then fine-tuned using training sets described in Section **??**. Deep residual features are extracted from the output of the last convolutional block of the network which is a 2048-D feature vector. After that, frame-level features are aggregated to create video-level features. In this work, three temporal modeling techniques are implemented: 1) average pooling (AP); 2) recurrent neural network (RNN) and 3) temporal attention (TA). Figure 3.3 illustrates three techniques.

Average Pooling - AP : Let f be the video-level feature, f_t be the frame-level feature at time t, T be the number of frames in video. Average pooling technique simply averages all frame-level features uniformly to create the video-level feature:

$$f = \frac{1}{T} \sum_{t=1}^{T} f_t \tag{3.1}$$

As a frame-level feature is a 2048-D vector, the video-level feature is of same dimension.

Recurrent Neural Network - RNN : An RNN cell encodes a t^{th} frame-level feature at time t of the sequence and passes the hidden state h_t into the next time step. The RNN is a single-layer with T cells. In this work, the cell is LSTM (Long Short Term Memory). Each cell outputs a 512-D feature vector that contains information

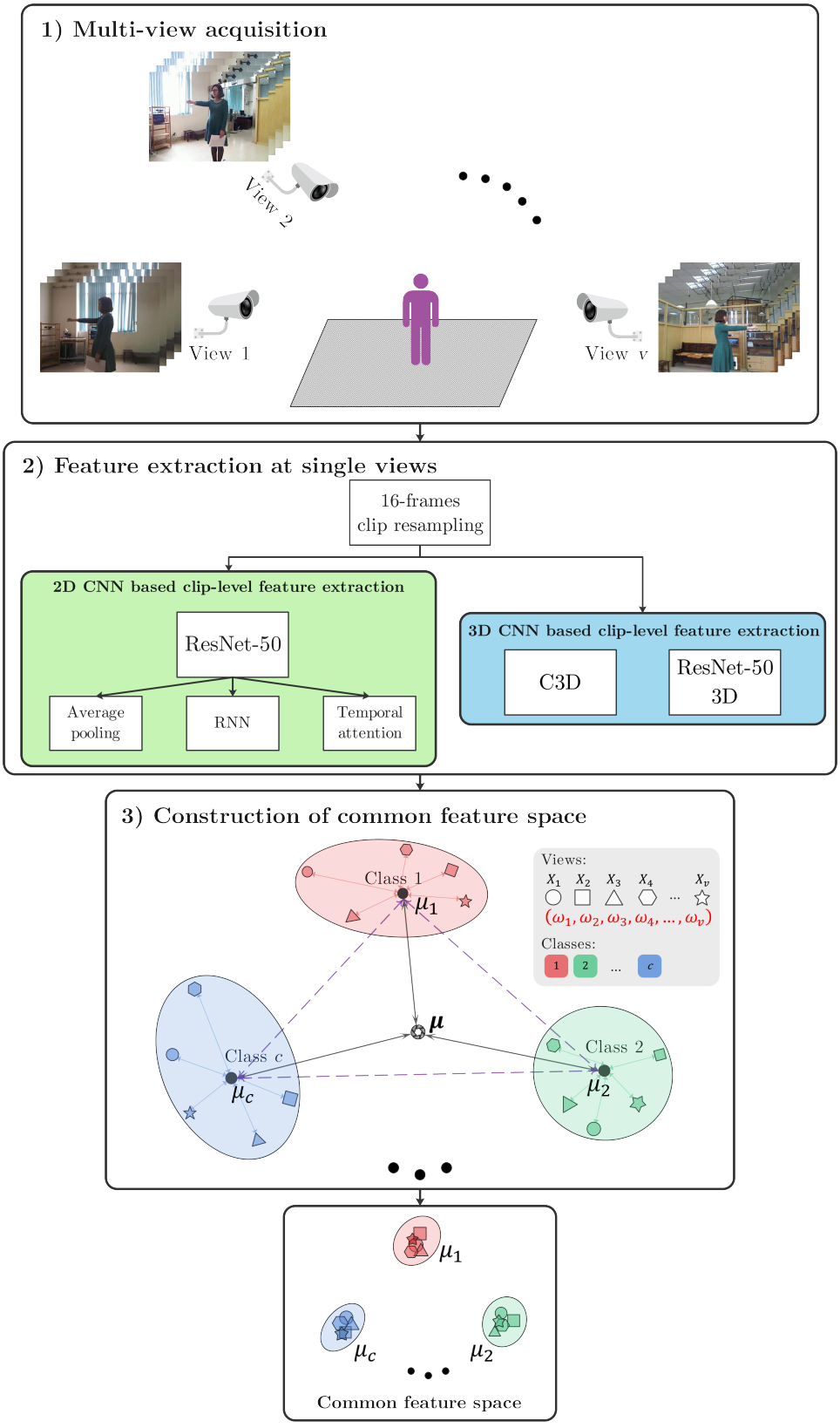


Figure 3.1: Proposed framework for building common feature space with pairwise-covariance multi-view discriminant analysis (pc-MvDA)

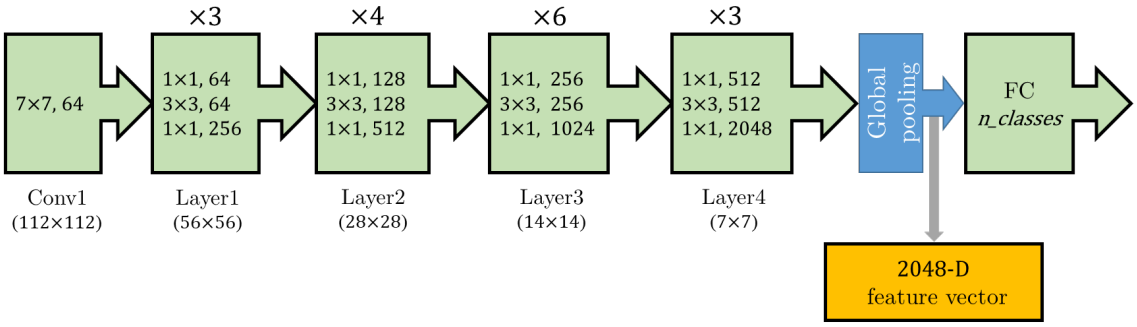


Figure 3.2: Architecture of ResNet-50 utilized in this work for feature extraction at each separated view

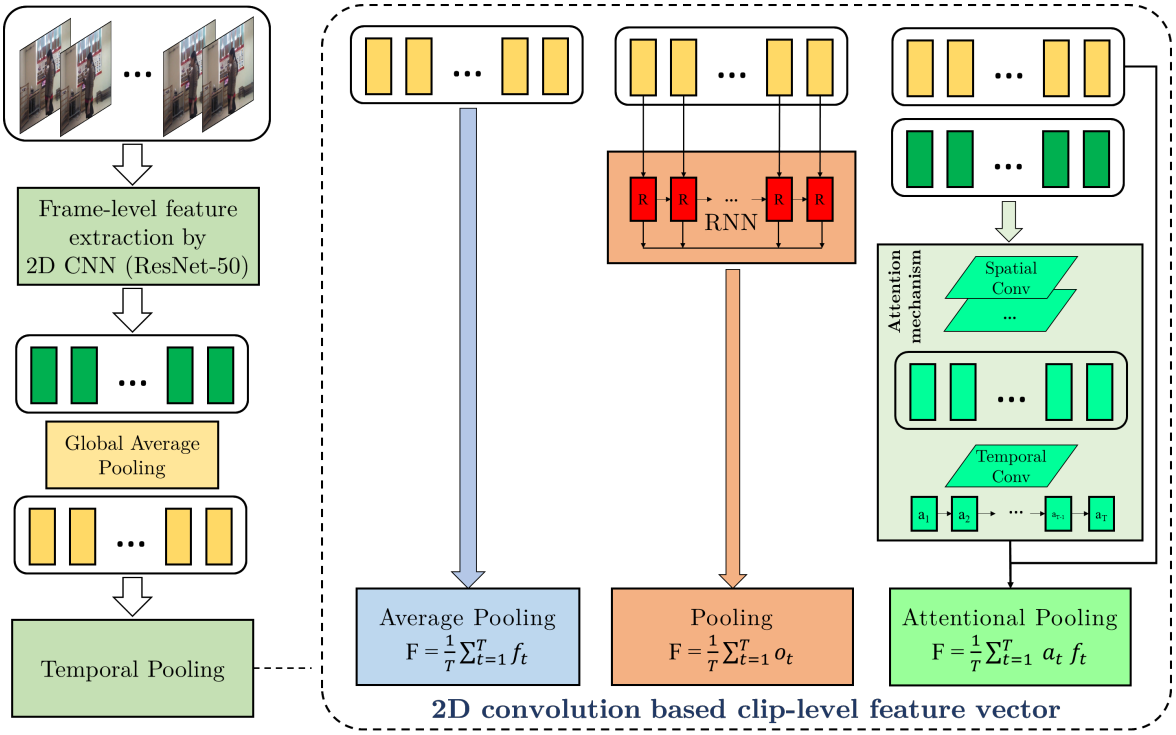


Figure 3.3: Three pooling techniques: Average Pooling (AP), Recurrent Neural Network (RNN) and Temporal Attention Pooling (TA)

of the current frame and the previous ones. A sequence of frame-level features is aggregated into a video-level feature f by calculating the average of the RNN outputs $o_t = R(f_t, o_{t-1}); t \in [1; T]$.

$$f = \frac{1}{T}\sum_{t=1}^{T} o_t \tag{3.2}$$

Temporal Attention - TA : In the above pooling techniques, frame-level features are equally aggregated. In reality, some frames might have more important roles than remaining ones for recognizing an action. In temporal attention model, a weight a_t is learnt for each frame f_t and an attention weighted average is applied on the sequence of frame-level features as follows:

$$f = \frac{1}{T}\sum_{t=1}^{T} a_t f_t \tag{3.3}$$

To learn the weights $a_t, t \in [1, T]$, the attention generation network proposed by Jiyang Gao et al. [50] is adopted. The network takes a sequence of frame-level features $[T, w, h, 2048]$, each is tensor extracted from the last convolution layer of ResNet-50. The network architecture consists of two main components: Spatial Convolution and Temporal Convolution. Firstly, a conv layer with shape $\{w, h, 2048, d_t\}$ is applied, then we get a d_t-dimensional feature for each frame of the clip (d_t = input channel). A temporal conv layer $\{3, d_t, 1\}$ is then applied on these frame-level features to generate temporal attentions s_c^t. Once s_c^t is obtained, the final attention score a_t is computed by Softmax function:

$$a_t = \frac{e^{s_c^t}}{\sum_{k=1}^{T} e^{s_c^k}} \tag{3.4}$$

3.3.2 3D CNN based clip-level feature extraction

3D convolution network architectures is increasingly concerned with video based problems. In this thesis, two 3D CNN architectures are deployed: C3D [51] and ResNet-50 3D [23].

ResNet 3D: ResNet-50 3D adopts 3D convolution kernels with ResNet-50 architecture. The architecture of ResNet-50 3D network is described in the Figure 3.4. It has similar architecture as the ResNet-50 but the convolution layers uses 3D operation.

C3D: 3D deep convolution neural network, which was introduced in [22], has shown to be very efficient for action recognition tasks. C3D takes input as an image sequence

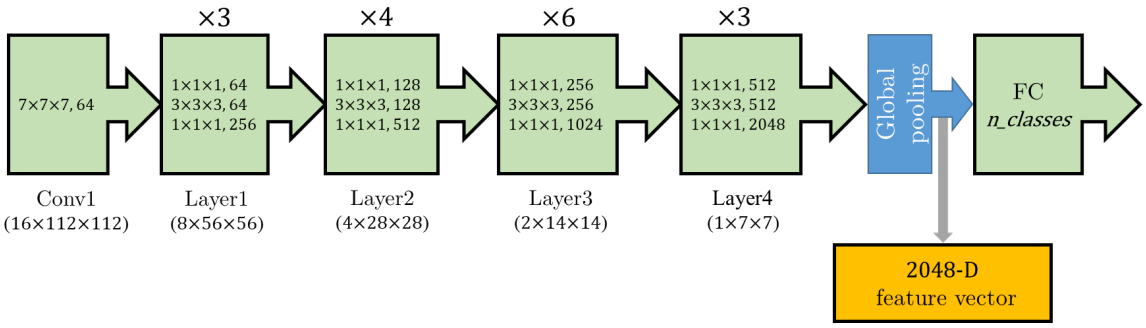


Figure 3.4: Architecture of ResNet-50 3D utilized in this work for feature extraction

instead of a static image, computes the 3D convolution on each 3D cubes from video clip. By doing so, C3D captures both spatial and temporal characteristics of action at the same time. A C3D network contains 8 convolution, 5 max-pooling and 2 fully

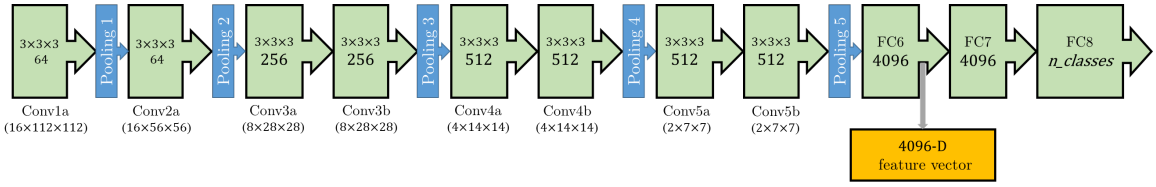


Figure 3.5: Architecture of C3D utilized in this work for feature extraction

connected layers as illustrated in Figure 3.5. The feature vector of 4096 dimensions extracted from FC6 layer will be served for training and testing classifiers in further steps.

To apply in the proposed framework of action and gestures recognition, transfer learning technique is applied for pretrained models on each data stream corresponding to each individual view. Details of pre-trained weights and training process of the networks will be presented in Section 4.4.

3.4 Construction of common feature space

3.4.1 Brief summary of Multi-view Discriminant Analysis

In this thesis, I proposed an improvement to MvDA. The experimental results are also compared to the performance of MvDA and its first variant namely MvDA-vc, both proposed by the same authors in [19] and [27]. Let us revisit the brief summary of the baseline MvDA before introducing the formulation of the proposed algorithm. The between-class scatter matrix $\boldsymbol{S}_B^y$ and minimize the within-class scatter matrix $\boldsymbol{S}_W^y$ are defined as:

$$\boldsymbol{S}_W^y = \sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}(y_{ijk} - \boldsymbol{\mu}_i)(y_{ijk} - \boldsymbol{\mu}_i)^T \tag{3.5}$$

$$\boldsymbol{S}_B^y = \sum_{i=1}^{c} n_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{3.6}$$

Then the objective function is formulated by a Rayleigh quotient:

$$(\boldsymbol{\omega}_1^*, \boldsymbol{\omega}_2^*, ..., \boldsymbol{\omega}_v^*) = \underset{\boldsymbol{\omega}_1,\boldsymbol{\omega}_2,...,\boldsymbol{\omega}_v}{\operatorname{argmax}} \frac{trace(S_B^y)}{trace(S_W^y)} \tag{3.7}$$

3.4.2 Pairwise-covariance Multi-view Discriminant Analysis

MvDA emphasizes on finding a common space with minimal within-class variation while the distances between class means and global mean are jointly maximized. However, the distances between some pairs of classes can be disregarded. In this work, to obtain this property, I modified MvDA with reformulated between and with-in class scatter matrices terms in a pairwise manner. First, let's define a new inter-class scatter matrix formula that takes paired distance into account:

$$\boldsymbol{S}_B^y = \sum_{a=1}^{c}\sum_{b=a+1}^{c}(\mu_a - \mu_b)(\mu_a - \mu_b)^T \tag{3.8}$$

where a, b are two distinct classes. For each pairs of class a and class b, the between covariance is calculated as:

$$\boldsymbol{S}_{Bab}^y = (\mu_a - \mu_b)(\mu_a - \mu_b)^T \tag{3.9}$$

The intra-class scatter matrix $\boldsymbol{S}_W^y$ of MvDA is calculated as simple summation of all covariance matrices of each i^{th} class, $i = 1, ..., c$:

$$\boldsymbol{S}_{Wi}^y = \sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}(y_{ijk} - \mu_i)(y_{ijk} - \mu_i)^T \tag{3.10}$$

Mathematically, it assumes data samples from each class among all views are identically Gaussian distributed, and contribute evenly to the minimization of intra-class scatter. In reality, it is hardly the case as data variations of samples from different

classes and different view points are usually drastically diverse and may also have different dimensions.

To better represent the distribution of data, pc-MvDA uses a paired intra-scatter matrix which is denoted as:

$$\boldsymbol{S}_{W\,ab}^{y} = \beta \frac{n_a \boldsymbol{S}_{W\,a}^{y} + n_b \boldsymbol{S}_{W\,b}^{y}}{n_a + n_b} + (1-\beta)\,\boldsymbol{S}_{W}^{y} \tag{3.11}$$

where $0 \leq \beta \leq 1$ is a hyper-parameter for regularization between the original global intra-covariance $\boldsymbol{S}_{W}^{y}$ and the novel two local class covariances $\boldsymbol{S}_{W\,a}^{y}$ and $\boldsymbol{S}_{W\,b}^{y}$. This formulation is closer to the value of covariances of both classes than the standard intra-covariance.

Instead of solving the vanilla generalized eigen value problem for the whole dataset, it is now splitted into sub-problems for each pairs of class a and b, in each of which I define the objective distance to be minimized between two corresponding classes.

Difference of the proposed algorithm compared to pairwise-covariance LDA (pc-LDA) In pc-LDA [25], the pairs of a and b classes are regarded as two Gaussian distributions $\mathcal{N}_a(\mu_a, \boldsymbol{S}_{W\,a}^{y}), \mathcal{N}_b(\mu_b, \boldsymbol{S}_{W\,b}^{y})$ and the objective distance between two classes is defined as their Kullback-Leibler divergence [26]:

$$D_{KL}\left(\mathcal{N}_a \parallel \mathcal{N}_b\right) = \frac{1}{2}\left(\mu_a - \mu_b\right)^T \left(\boldsymbol{S}_{W\,ab}^{y}\right)^{-1} \left(\mu_a - \mu_b\right), \tag{3.12}$$

Then the final objective is properly weighted to focus on classes with more samples:

$$\min_{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_v} J = \sum_{a=1}^{c} \sum_{b=a+1}^{c} \frac{n_a n_b}{\left[2 D_{KL}\left(\mathcal{N}_a \parallel \mathcal{N}_b\right)\right]^q} \tag{3.13}$$

where $q \geq 1$ is a hyper-parameter that controls how much the pairs of classes with smaller objective distances are biased over the others.

In my observation, the minimization of KL divergence for each pairs of classes a and b can be substituted by a generalized eigenvalue problem with the pairwise $\boldsymbol{S}_{W\,ab}^{y}$ and the $\boldsymbol{S}_{B\,ab}^{y}$ defined above. Although these sub-problems do not have an unified analytical solution to be solved concurrently, the criteria can be formulated in various differentiable ways like in [52] so I can use gradient descent algorithm:

$$J_1 = \frac{tr(S_B)}{tr(S_W)}; \quad J_2 = tr\left(\frac{S_B}{S_W}\right); \quad J_3 = tr\left|\left|\frac{S_B}{S_W}\right|\right|; \quad J_4 = \frac{det(S_B)}{det(S_W)} \tag{3.14}$$

In this thesis I choosed the former Fisher loss as the ratio of scalar values is computationally cheaper. Comparing to the objective of pc-LDA in Equation (3.13), my proposed model is obviously more efficient since it contains only simple operators and the need to inverse a singular-prone matrix is negated. Therefore, it is better fit in the scenario where we want to train the model multiple iterations over multi-view high dimensional output.

The final objective function of pc-MvDA is sum of all pairwise Fisher criteria with normalized weights:

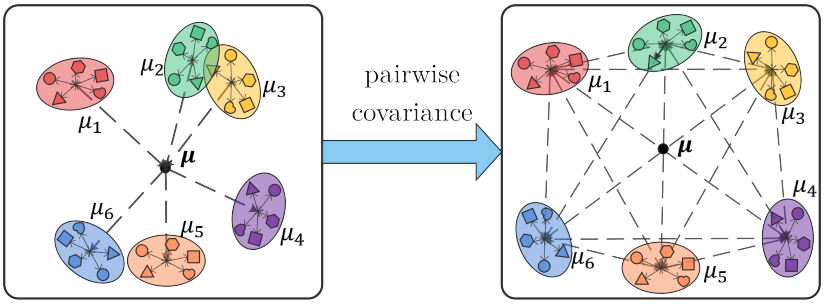
$$\min_{\boldsymbol{\omega}_1,\boldsymbol{\omega}_2,...,\boldsymbol{\omega}_v} J = \sum_{a=1}^{c}\sum_{b=a+1}^{c} \frac{n_a n_b}{n_{cc}^2}\left[\frac{trace\left(\boldsymbol{S}_{W\,ab}^{y}\right)}{trace\left(\boldsymbol{S}_{B\,ab}^{y}\right)}\right]^q \tag{3.15}$$

where $n_{cc} = \frac{n}{v}$ is the number of common components as coined in [21] or simply number of samples in each view. This added normalization term assures that the objective does not depend on size of dataset.

Let's look at the Figure 3.1, the nominator of MvDA(-vc) $\boldsymbol{S}_B^y$ is the sum of all distances from mean of every class μ_i, $i = 1, ..., c$ to the mean of all classes μ. This helps to push classes far away from the global mean (the dotted green lines in Figure 3.1). However, it might not ensure that the classes would be separated from each other (Figure 3.6a). In this work, pairwise distance is taken into account and integrated in an multi-view framework (the dotted violet lines in Figure 3.1). Thank to this, different classes will be more discriminated (Figure 3.6b).

Illustrative examples I illustrate the advantage of pc-MvDA on artificial datasets. Firstly, using function provided by *scikit-learn* library, we can generate several isotropic Gaussian blobs equal to desired number of classes as data samples from one single view. To generate other views, one can simply clone and apply translation, point-wise noise and rotation with random orthogonal group multiplication to the samples of first view.

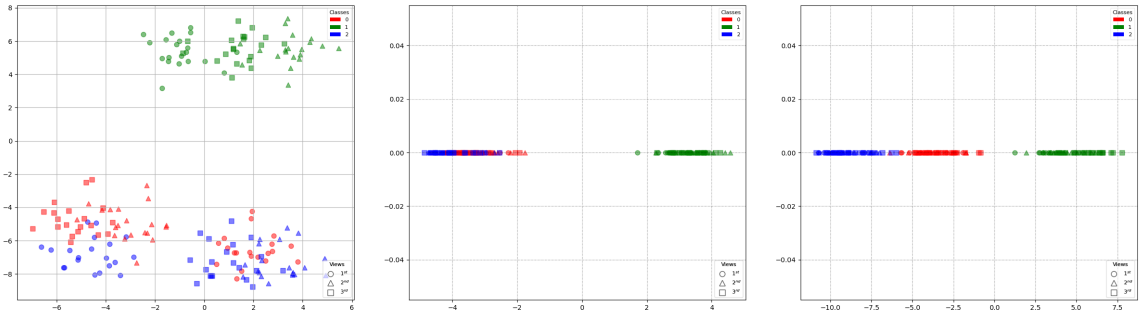
Figure 3.7 shows MvDA and pc-MvDA results on 2D synthetic dataset of 180 data points in 2-D space. There are three classes (denoted by colors), observed at three viewpoints (denoted by shapes). This Figure displays original data distribution and 1-D projection results using MvDA and pc-MvDA respectively. Look closely, we can notice that data points of two classes (red and blue) are well discriminated in each individual view but seems to have interchanged position in original space. In essence, 2^{nd} ($\triangle$) and 3^{rd} ($\square$) views share a similar layout while the 1^{st} ($\bigcirc$) view is flipped around the y (vertical) axis. After projected in common space by MvDA, these classes are still close together as the common space constructed satisfies the criteria that distances



a) MvDA b) pc-MvDA

Figure 3.6: a) MvDA does not optimize the distance between paired classes in common space. b) pc-MvDA takes pairwise distances into account then distinguish better the classes.

between each class mean and global mean are maximized. In contrast, these classes are much more distinguishable when projected by pc-MvDA as it can find projections that undo the "flipping" in 1^{st} (◯) view to maximize pairwise distances.

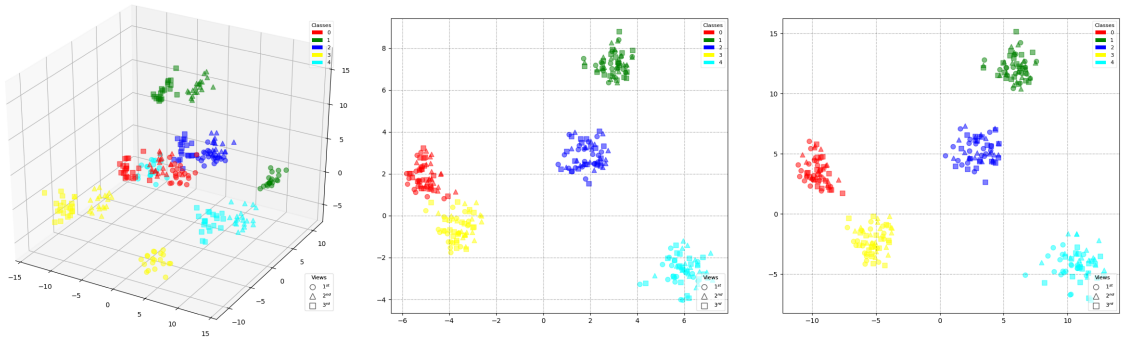


a) Original distribution b) MvDA c) pc-MvDA

Figure 3.7: A synthetic dataset of 180 data points, evenly distributed to 3 classes among 3 different views; a) 2-D original distribution; b) 1-D projection of MvDA; c) 1-D projection of pc-MvDA

Similar to the first example, the second example consists of 300 data points of 5 classes observed from 3 different views (Figure 3.8). We can observe that the five clusters of pc-MvDa common space contract strongly and become more separated as compared to the MvDA results (especially the red and yellow classes). The ablation study on synthetic datasets shows superiority of the proposed method.

Algorithm to solve pc-MvDA As the proposed objective does not match any optimization form of generalized eigenvalue problems, gradient descent algorithm is used to solve pc-MvDA. This approach also allow the algorithm to update incrementally even



a) Original distribution b) MvDA c) pc-MvDA

Figure 3.8: A synthetic dataset of 300 data points, evenly distributed to 5 classes among 3 different views; a) 3-D original distribution; b) 2-D projection of MvDA; c) 2-D projection of pc-MvDA

in higher dimensional data where the computations of matrix inversion and eigenvectors are not conducive. The derivative of Equation (3.15) is:

$$\frac{\partial J\left(W\right)}{\partial W} = -\sum_{a<b}^{c} \frac{qn_a n_b}{{n_{cc}}^2 J_{ab}\left(W\right)^{q+1}} \frac{\partial J_{ab}\left(W\right)}{\partial W} \tag{3.16}$$

where $J_{ab}\left(W\right) = tr\left(\boldsymbol{S}^y_{Bab}\right)/tr\left(\boldsymbol{S}^y_{Wab}\right)$ is the Fisher loss of class pair a and b. Its gradient is computed using the funky trace derivative and quotient rule:

$$\frac{\partial J_{ab}\left(W\right)}{\partial W} = \frac{tr\left(\boldsymbol{S}^y_{Wab}\right) W^T \left(\boldsymbol{S}^x_{Bab} + {\boldsymbol{S}^x_{Bab}}^T\right) - tr\left(\boldsymbol{S}^y_{Bab}\right) W^T \left(\boldsymbol{S}^x_{Wab} + {\boldsymbol{S}^x_{Wab}}^T\right)}{tr\left(\boldsymbol{S}^y_{Wab}\right)^2} \tag{3.17}$$

The superscript x replacing y in scatter matrices notations means that they denote the pre-transformed version. With simple algebra, we can rewrite the formulas in matrix multiplication form, separating transformation vectors ω_j in a concatenated matrix W and a multi-view covariance matrix constructed from $v \times v$ cells, each cell S_{jk} stands for the covariance between view j and view k.

$$\boldsymbol{S}^y = W^T \boldsymbol{S}^x W = \begin{bmatrix} \omega_1^T & \omega_2^T & \cdots & \omega_v^T \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1v} \\ S_{21} & S_{22} & \cdots & S_{2v} \\ \vdots & \vdots & \ddots & \vdots \\ S_{v1} & S_{v2} & \cdots & S_{vv} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_v \end{bmatrix} \tag{3.18}$$

The complete algorithm to compute ∇J is given in Algorithm 1.

No further constraint is imposed to the proposed model. Although most modern

Algorithm 1: Computation of $\nabla J(W)$ (i.e. gradient of Equation (3.15))

Input : W, $\{X, \mu\}$, q
Output: $\nabla J(W)$

$\boldsymbol{F} = 0$
Compute $\boldsymbol{S}_W^x$ according to Equation (3.5)
for $a = 1$ **to** c **do**
  for $b = a + 1$ **to** c **do**
    Compute $\boldsymbol{S}_{Bab}^x$ according to Equation (3.9)
    Compute $\boldsymbol{S}_{Wab}^x$ according to Equation (3.11)
    Compute $\boldsymbol{S}_{Bab}^y = W^T \boldsymbol{S}_{Bab}^x W$
    Compute $\boldsymbol{S}_{Wab}^y = W^T \boldsymbol{S}_{Wab}^x W$
    Compute $J_{ab} = tr\left(\boldsymbol{S}_{Bab}^y\right) / tr\left(\boldsymbol{S}_{Wab}^y\right)$
    Compute ∇J_{ab} according to Equation (3.17)
    $\boldsymbol{F} = \boldsymbol{F} + n_a n_b \nabla J_{ab} / J_{ab}^{q+1}$
  end
end
$\nabla J = -q\boldsymbol{F} / n_{cc}^2$
Output: ∇J

programming frameworks do support the automatic deduction of $\nabla J(W)$ as backward process of computing the objective J, the above algorithm is derived for reimplementation in those that do not (i.e. Matlab). In fact, in my implementation of pc-MvDA in Pytorch, the computation of gradient is handled by automatic differentiation. To have better starting point and mitigate variances, pc-MvDA is initialized with transformation learnt by MvDA.

3.5 Summary

4 Experiments

4.1 Introduction

4.2 Datasets

In this thesis, three datasets are used in evaluation of the proposed algorithm. Two of which are benchmark datasets IXMAS [53], MuHAVi [2] and the other (namely MICAGes) is recorded at computer vision department of MICA Institute.

4.2.1 IXMAS dataset

The IXMAS dataset is a multi-view dataset built by Perception project [53]. It contains 12 action classes (check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave, punch, kick, point and pick up) recorded simultaneously by 5 cameras (4 side view cameras and 1 top view camera). Each action is performed 3 times by 10 actors (5 males / 5 females). Some representative frames of action *check watch* are shown in Figure 4.1.

Figure 4.1: Illustration of frames extracted from action *check watch* observed from five camera viewpoints.

During the time of writing this thesis, the original version of the dataset is no longer publicly available due to the privacy issues. Only a subset of 1692 samples containing samples from four side camera views (excluding the top down view) that have been downloaded previously was utilized with the agreement of the data-set's authors. The comparison with SOTA methods on this dataset will only be taken from the four side view cameras.

4.2.2 MuHAVi dataset

The MuHAVi dataset is constructed and introduced by [2]. It is usually referred as MuHAVi-uncut because it contains full, raw videos of 17 actions, asynchronously captured by eight cameras that provide completely overlapping coverage of a rectangular action zone from different viewing directions as in Figure 4.2.

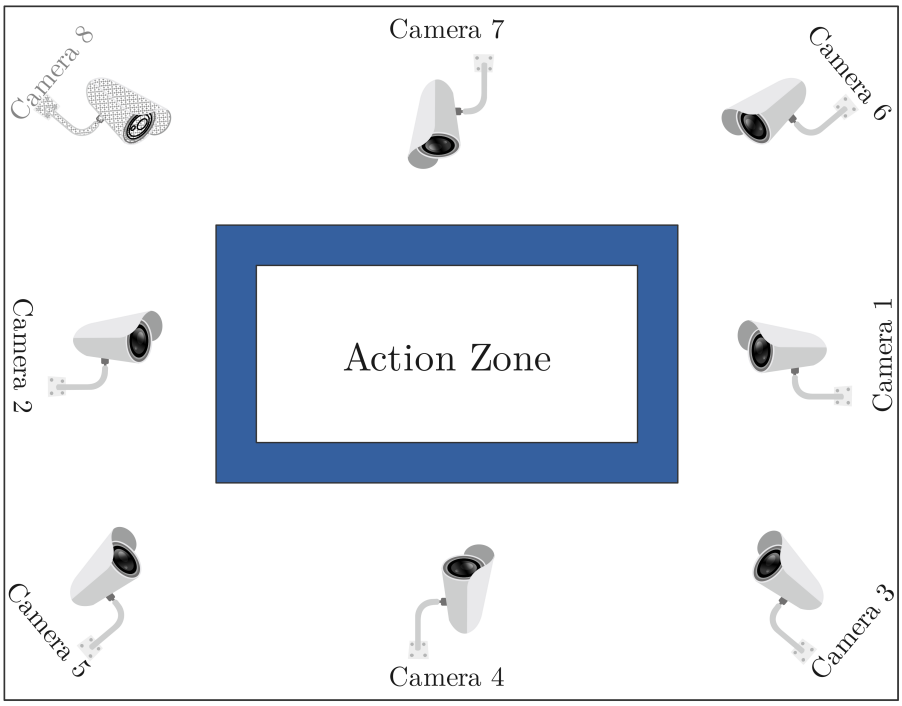


Figure 4.2: Environment setup to collect action sequences from 8 views [2].

The actions are are walk turn back, run stop, punch, kick, shotgun collapse, pull heavy object, pickup throw object, walk fall, look in car, crawl on knees, wave arms, draw graffiti, jump over fence, drunk walk, climb ladder, smash object, and jump over gap. Each was performed several times by 7 actors (5 males / 2 females). The videos were collected at rate of 25 fps with resolution of 720 $\times$ 576, except for the 8^{th} camera whose data is not included in experiments of this thesis due to absence of annotation. Some representative frames of action *punch* are shown in Figure 4.3.

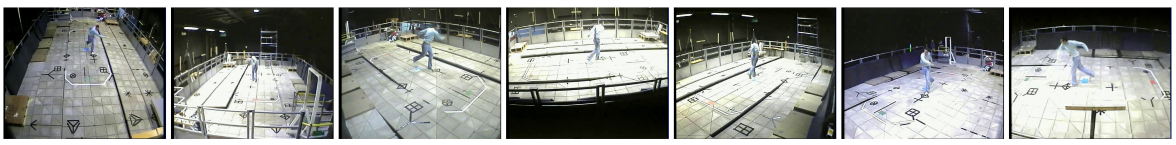


Figure 4.3: Illustration of frames extracted from an action *punch* observed from Camera 1 to Camera 7.

4.2.3 MICAGes dataset

There does not exist a dataset publicly available in the literature which is dedicated to the evaluation of robustness of hand gesture recognition w.r.t. viewpoint changes. Therefore, thanks to the work of computer vision department at MICA Institute, a dataset was carefully designed and collected from multiple viewpoints in indoor environment with complex background. It consists of 9 dynamic hand gestures which correspond to control commands of electronic home appliances. Each gesture is a combination of hand movement following a pre-defined direction and changing of hand shape in a natural manner.

Five Kinect sensors K1, K2, K3, K4, K5 are setup at five positions in a square simulation room of $16m^2$ (Figure 4.4). This work aims to capture hand gestures under multiple viewpoints synchronously. The subjects are invited to stand at a fixed position approximately 2 meters in front of the center view. The Kinect sensors provide both RGB and Depth data recorded at frame rate of 20 fps and resolution of 640 × 480, which legitimates the capture a multi-view and multi-modal dataset.

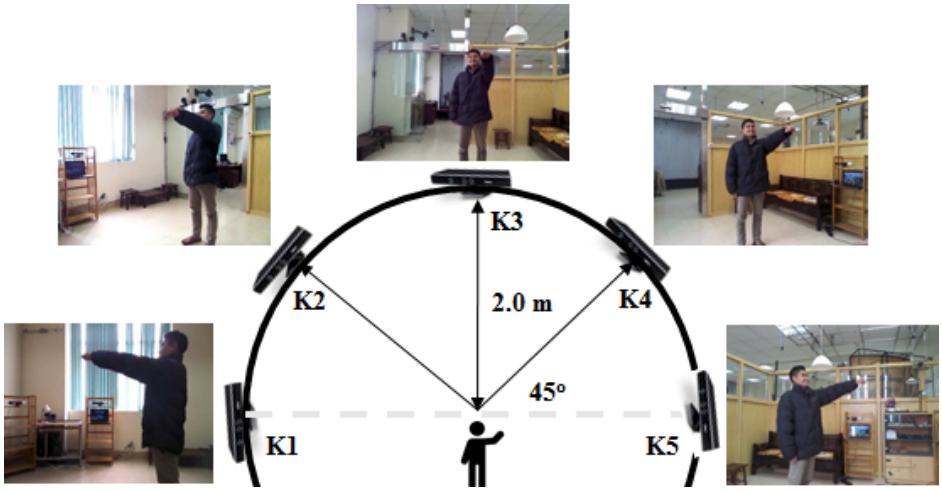


Figure 4.4: Environment setup to capture MICAGes dataset.

Twelve participants (08 males / 04 females) are voluntary to perform gestures one after another, each gesture three times. In the experiments of this thesis, only RGB information is concerned. Totally, the dataset contains 1620 (5 views × 9 gestures × 12 subjects × 3 times) dynamic hand gestures. Illustrations of segmented hand poses of a gesture are shown in Figure 4.5.

Figure 4.5: Illustration of a gesture belonging to the 6^{th} class observed from 5 different views.

4.3 Evaluation protocol

The leave-one-actor out cross validation strategy represented in [54] is employed and average accuracy (%) are computed as means of comparison. In order to evaluate the proposed framework in terms of cross-view performance, it is further combined with cross-view validation. That means at a moment only two views are picked to evaluate, each consists of a train and a test part defined by the leave-one-actor out strategy. Apart from the proposed evaluation protocol, from now named "*protocol 1*", another "*protocol 2*" is assessed, which is regularly used in the literature. The main difference is in second protocol, the notation of train or test is omitted after feature extraction stage, every features samples from single view are included to train the multi-view metrics, then the classifier is fit on common features of one view and tested on common features of another view. It is noticed that the first evaluation protocol is more challenging because the in the second protocol, the constructed common space is already fit on the whole multi-view dataset. As a result, the later predictive models are trained and evaluated on features generated by a severely over-fitted projector. The difference between two protocols are illustrated in Figure 4.6.

The overall cross-view evaluation score is computed as average of all evaluation scores of each split, whose method of computation is described specifically in Figure 4.6:

$$acc_{ij} = \frac{\sum_{k=1}^{M} acc_{ij}^{(k)}}{M} \tag{4.1}$$

where M is the number of actors. Both protocols finally generate a cross-view accuracy

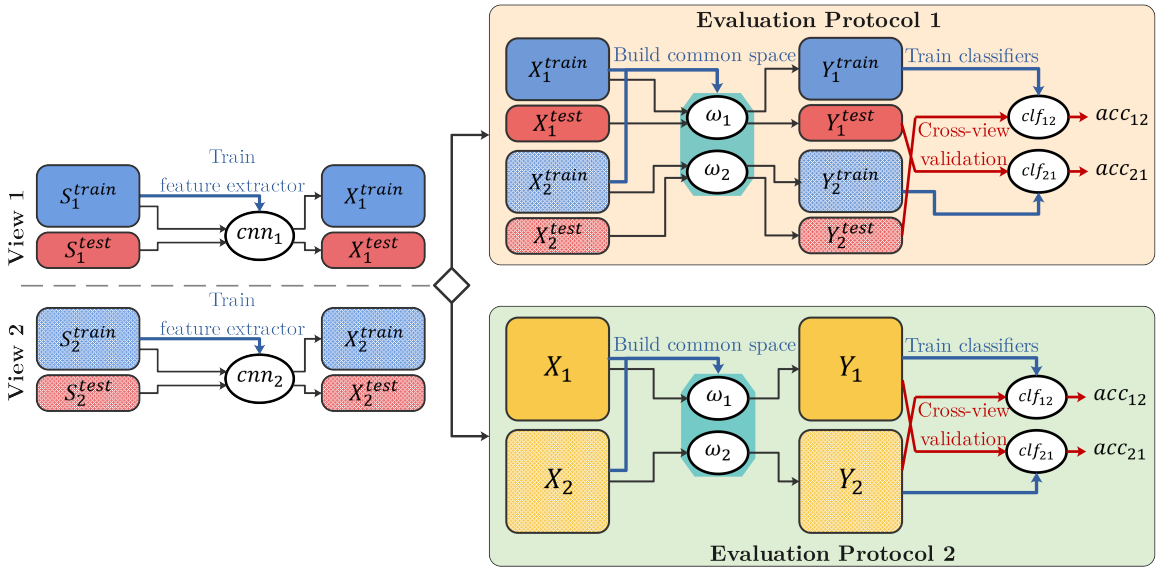


Figure 4.6: Two evaluation protocols used in experiments.

matrix as follows:

$$\begin{bmatrix} acc_{11} & acc_{12} & \cdots & acc_{1v} \\ acc_{21} & acc_{22} & \cdots & acc_{2v} \\ \vdots & \vdots & \ddots & \vdots \\ acc_{v1} & acc_{v2} & \cdots & acc_{vv} \end{bmatrix} \tag{4.2}$$

For eager expression, let's define function accuracy $\left(Y, \tilde{Y}\right)$ as the accuracy score when fitting a classifier on Y features and evaluating its prediction on $\tilde{Y}$ features. Each cell can be expressed accordingly:

$$score_{jr}^{\textbf{protocol1}} = \left\langle \left\{ \text{accuracy}\left(Y_j^{train}, Y_r^{test}\right) \middle|\ j, r = (1, .., v) \right\} \right\rangle \tag{4.3}$$

$$score_{jr}^{\textbf{protocol2}} = \langle \{ \text{accuracy}\left(Y_j, Y_r\right) |\ j, r = (1, .., v) \} \rangle \tag{4.4}$$

Multi-view strategy: As MvDA and its variants can theoretically scale for an arbitrary number of views, in order to fairly compare the proposed algorithm with others that only apply for 2 views, I also experiment 2 different multi-view strategies. The key idea is that each strategy restricts the number of views participated in the training process of MvL algorithms.

In "*multi-view*" strategy, only one set of $W^* = \{\omega_1, \omega_2, ..., \omega_v\}$ is learnt and all separated view features are transformed $Y_j = \omega_j^T X_j,\ j = (1, ..., v)$ before computation of accuracy.

In "*cross-view*" strategy, each cell $score_{jr}$ of (4.2) must be calculated exclusively

with the inclusion of only features of those views j and r (wihtout information from other views). Therefore, for each distinct combination jk, $W^*_{jr} = \{\omega_j, \omega_r\}$ is learnt to transform X_j and X_r. The diagonal of (4.2) is also ignored in this strategy.

Since there are many scores generated by such extra complicated cross validation process when cycling through every actors, the final reported is average of all. Further, it is noticed that in existing works, the evaluation have been done mostly similar to the second evaluation protocol. Hence, in comparison with state of the art techniques, only the scores resulted from the second evaluation protocol are taken.

4.4 Experimental Setup

4.4.1 Programming Environment and Libraries

The majority of my code implementation is written in Python, fueled by PyTorch [55] deep learning framework. The framework offers powerful supports for matrix manipulation, automatic diffirentiation, accelerated training of deep neural networks on parallel computing platforms and flexibility to efficiently research new algorithmic approaches. The MvDA and MvDA-vc Matlab implementations are inherited from the repository published by [19] with slight modifications. The implementation of proposed pc-MvDA is published at *https://github.com/inspiros/pcmvda*.

4.4.2 Configurations

Feature extractors All models are trained for 60 epoches with SGD optimizer, initial learning rate is 0.0003 and momentum is 0.9. The choosen pre-trained models are:

- ResNet-50 model was pre-trained on ImageNet dataset and is available in sub-package *torchvision.models* of PyTorch.
- C3D model was pre-trained on Sports-1M [31] then fine-tuned on Kinetics dataset [56]. The model is implemented by [57] in Caffe2 deep learning framework, which was improved from original model presented in [22] by inserting Batch Normalization layers after each Convolution layer.
- ResNet-50 3D was pre-trained on Kinetics dataset and weights are given by [23].

Multiview analysis algorithms Output dimensions of both algorithms are 200. MvDA has no alterable parameter and λ of view-consistency term in MvDA-vc is set to 0.03.

For pc-MvDA, the hyperparameters are $\beta = 1$ and $q = 1$ and the optimizer utilized is Adam with initial learning rate set to 0.01, on each experiment trained 300 epochs.

Classifier The final classifier used is kNN with $k = 1$.

4.5 Experimental Results and Discussions

4.5.1 Experimental results on IXMAS dataset

1) Cross-view validation: In this experiment, I train on data from one view (training view) and testing on data from another view (testing view) then compute the accuracies for two evaluation protocols. Table 4.1 shows comparative results of using different deep features (C3D, ResNet-50 3D, ResNet-50 RNN, ResNet-50 TA, ResNet-50 AP) and multi-view discriminant analysis techniques (MvDA, MvDA-vc and my proposed pc-MvDA).

With the first evaluation protocol, the proposed pc-MvDA, when combined with deep features, gives mostly best accuracy for both evaluation protocols. With C3D features, accuracy by pc-MvDA (88.43%) is 6.6% higher than MvDA (81.84%). pc-MvDA is even better than MvDA-vc (2.87%). pc-MvDA achieved higher accuracy (about 6% higher) than MvDA and MvDA-vc with ResNet-50 TA and ResNet-50 AP features. In average, pc-MvDA is 3.09% higher than MvDA and 1.4% higher than MvDA-vc.

Table 4.1: Cross-view recognition comparison on IXMAS dataset

Deep features	Protocol 1			Protocol 2		
	MvDA	MvDA-vc	pc-MvDA	MvDA	MvDA-vc	pc-MvDA
C3D	81.84	85.56	**88.43**	96.54	99.29	**99.98**
ResNet-50 3D	91.25	92.19	**92.89**	97.30	**99.73**	99.65
ResNet-50 RNN	75.51	76.12	**76.54**	99.99	99.71	**100**
ResNet-50 TA	79.44	80.58	**82.05**	99.33	99.34	**99.84**
ResNet-50 AP	77.00	79.04	**80.58**	99.68	99.81	**99.92**

For the second evaluation protocol, pc-MvDA almost always keeps better performance compared to MvDA and MvDA-vc. The recognition accuracy scores obtained by both pc-MvDA and MvDA-vc are nearly 100% for every kind of deep features. With C3D features and ResNet-50 3D features, pc-MvDA is 99.98% and 99.65%, which is 3.44% and 2.35% higher than the orginal MvDA (96.54% and 97.3%) respectively. In average, pc-MvDA is 1.31% better than MvDA and 0.3% better than MvDA-vc.

In terms of deep features, with the first evaluation protocol, ResNet-50 3D gives the best accuracy (92.89%) following by C3D (88.43%), ResNet-50 TA (82.05%), ResNet-50 AP (80.58%). The lowest accuracy obtained by ResNet-50 RNN is only 76.54%. It shows that ResNet-50 3D produces the most discriminative feature space.

Table 4.2: Cross-view recognition results of different features on IXMAS dataset with pc-MvDA method. The result in the bracket are accuracies of using features C3D, ResNet-50 3D, ResNet-50 RNN, ResNet-50 TA, Restnet-50 AP respectively. Each row corresponds to training view (from view C0 to view C3). Each column corresponds to testing view (from view C0 to view C3)

Training \ Testing	C0	C1	C2	C3
C0	N/A	(90.7, **91.9**, 81.6, 84.6, 83.6)	(86.9, **93.9**, 78.0, 79.6, 78.1)	(89.9, **93.4**, 76.3, 82.8, 82.1)
C1	(86.6, **91.9**, 71.5, 81.1, 77.5)	N/A	(85.9, **94.2**, 78.6, 79.3, 80.3)	(88.9, **93.7**, 74.8, 82.3, 81.3)
C2	(87.6, **92.4**, 72.8, 81.8, 78.5)	(90.7, **91.7**, 80.6, 83.6, 82.6)	N/A	(89.4, **93.7**, 75.5, 82.3, 80.6)
C3	(87.6, **92.9**, 70.2, 82.1, 78.8)	(90.7, **91.2**, 81.6, 84.1, 82.8)	(86.4, **93.7**, 77.3, 80.6, 80.8)	N/A

2) Multi-view validation Table 4.3 shows multi-view recognition results. The conclusion is consistent with the case of cross-view evaluation: ResNet-50 3D is the best feature extractor. When it is combined with pc-MvDA, the framework gives the highest accuracy for the first protocol (92.82%), following by C3D (88.19%), ResNet-50 TA (81.49%), ResNet-50 AP(80.43%), and ResNet-50 RNN (76.47%). Both MvL algorithms give similar accuracies for the second evaluation protocols (nearly 100%). Again, pc-MvDA enhances the performance of MvDA by 2.43% for the first protocol and 0.83% for the second protocol. It only gives slightly better result than MvDA-vc (0.83% for the first protocol and 0.74% for the second protocol).

Table 4.3: Multi-view recognition comparison on IXMAS dataset

Deep features	Protocol 1			Protocol2		
	MvDA	MvDA-vc	pc-MvDA	MvDA	MvDA-vc	pc-MvDA
C3D	86.93	87.04	**88.19**	**99.99**	99.44	**99.98**
ResNet-50 3D	91.84	92.33	**92.82**	**100**	99.80	99,67
ResNet-50 RNN	72.44	75.95	**76.47**	99.34	**99.97**	**99.96**
ResNet-50 TA	76.74	81.01	**81.49**	95.80	98.25	**99.79**
ResNet-50 AP	79.28	78.93	**80.43**	**100**	98.11	99.85

Figure 4.7 compares the performance of feature extractors regarding each action class in multi-view evaluation scheme combined with the first evaluation protocol. There are clear margins between the performance of ResNet-50 3D and followed by C3D with other types of deep features, especially in harder actions (the first class to the third class and the eighth class to the eleventh class). These actions (check watch, cross arm, scratch head, wave, punch, kick, point) involve the most part static pose of the

body and only movement of limbs, whereas other actions 4 - sit down, 5 - get up, 6 - turn around, 7 - walk and 12 - pickup include movement of the whole body and are easier to be recognized. This suggests that 3D convolution can better deal with small difference of movement in action images, which in turn generates a much more classification-ready feature space for latter stage of recognition.

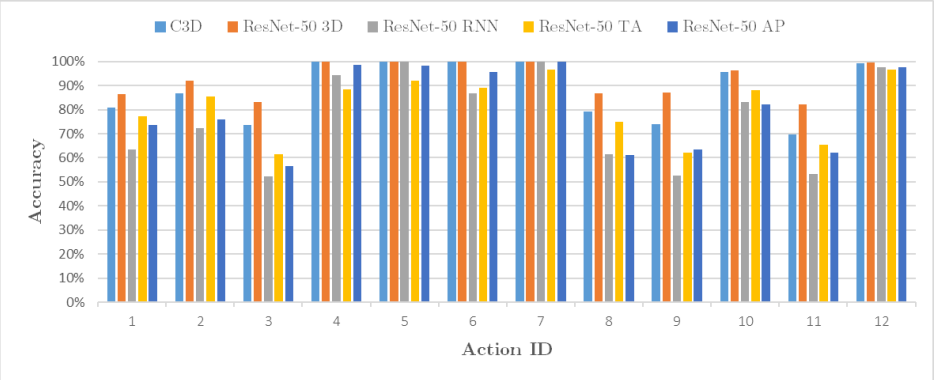


Figure 4.7: Comparison of accuracy on each action class using different deep features combined with pc-MvDA on IXMAS dataset

Table 4.4 compares the best combination of ResNet-50 3D and pc-MvDA with state-of-the-art frameworks. The comparison is for reference only because in other works, low-level hand-crafted video representation is used as private features and train-test strategy is leave-one-class out, which is not applicable to supervised deep feature extractors in this work. However, the results are still commensurable.

Table 4.4: Comparison of proposed methods with SOTA methods on IXMAS dataset according to the second evaluation protocol

Methods	Cross-view	Multi-view
Liu et al. [11]	76.32	N/A
Zheng et al. [41]	95.1	99.32
Zheng et al. [58]	97.8	99.4
Kong et al. [13]	99.92	100
Ulhaq et al. [59]	66.82	92.47
Zhang et al. [60]	84.1	N/A
Liu et al. [61]	N/A	90.3
Liu et al. [14]	99.95	99.67
Proposed method (ResNet-50 3D + pc-MvDA)	99.67	99.65

4.5.2 Experimental results on MuHAVi dataset

1) Cross-view validation: Table.4.5 illustrates cross-view recognition results. The proposed pc-MvDA consistently produces a better average accuracy of around 96.19%

for protocol 1, approximately 4.55% and 1.51% higher than that of MvDA (91.64%) and MvDA-vc (94.67%) respectively.

Table 4.5: Cross-view recognition comparison on MuHAVi dataset

Deep features	Protocol 1			Protocol 2		
	MvDA	MvDA-vc	pc-MvDA	MvDA	MvDA-vc	pc-MvDA
C3D	92.85	95.15	**97.65**	98.95	99.76	**100**
ResNet-50 3D	97.94	99.15	**99.18**	96.77	**99.94**	**99.94**
ResNet-50 RNN	95.54	95.14	**96.18**	98.70	99.05	**99.99**
ResNet-50 TA	85.80	88.95	**91.12**	96.57	97.03	**99.88**
ResNet-50 AP	86.09	94.98	**96.82**	89.04	98.62	**99.93**

Table 4.6: Cross-view recognition results of different features on MuHAVi dataset with pc-MvDA method. The result in the bracket are accuracies of using features C3D, ResNet-50 3D, ResNet-50 RNN, ResNet-50 TA, ResNet-50 AP respectively. Each row corresponds to training view (from view C1 to view C7). Each column corresponds to testing view (from view C1 to view C7)

Training \ Testing	C1	C2	C3	C4	C5	C6	C7
C1	N/A	(96.1, **100**, 96.8, 88.5, 98.5)	(98.6, **99.6**, 96.6, 88.9, 96.3)	(98.6, **99.6**, 98.0, 93.1, 98.2)	(97.7, **99.6**, 95.4, 91.3, 96.1)	(97.7, **98.4**, 92.3, 93.1, 96.3)	(98.2, **98.5**, 97.2, 89.8, 97.1)
C2	(95.6, **99.0**, 95.2, 91.3, 95.5)	N/A	(98.9, **99.6**, 96.9, 90.3, 96.1)	(98.4, **99.6**, 97.6, 92.6, 98.2)	(97.7, **99.6**, 95.4, 90.9, 96.1)	(97.8, **98.4**, 92.7, 93.6, 96.5)	(98.6, **99.0**, 97.4, 90.6, 96.9)
C3	(96.1, **99.0**, 95.6, 90.3, 95.1)	(97.0, **100**, 96.7, 88.4, 97.8)	N/A	(98.1, **99.2**, 98.2, 92.8, 98.2)	(97.9, **99.6**, 95.4, 90.2, 95.6)	(97.9, **98.2**, 93.0, 94.2, 96.5)	(97.8, **98.7**, 97.4, 89.4, 97.6)
C4	(96.4, **98.8**, 95.4, 89.6, 95.3)	(96.3, **100**, 96.6, 91.0, 98.5)	(98.6, **99.6**, 97.3, 89.5, 96.1)	N/A	(97.7, **99.4**, 95.6, 91.3, 96.1)	(**98.4**, 98.0, 93.2, 93.8, 95.6)	(97.7, **98.7**, 97.4, 91.3, 96.7)
C5	(95.9, **99.0**, 95.6, 91.3, 95.5)	(96.6, **100**, 97.1, 89.9, 98.3)	(98.8, **99.4**, 96.8, 90.7, 96.1)	(98.4, **99.6**, 98.0, 92.1, 97.6)	N/A	(97.9, **98.2**, 93.4, 94.4, 96.8)	(98.0, **98.7**, 96.9, 89.0, 97.8)
C6	(95.8, **99.0**, 95.8, 90.3, 95.0)	(97.0, **100**, 97.6, 89.7, 98.5)	(98.8, **99.6**, 96.9, 88.7, 96.1)	(98.4, **99.4**, 98.0, 92.8, 98.2)	(97.5, **99.6**, 95.9, 91.2, 95.5)	N/A	(98.2, **99.0**, 97.2, 90.7, 97.4)
C7	(96.2, **98.8**, 95.7, 91.5, 96.2)	(96.4, **100**, 97.5, 90.3, 98.5)	(98.4, **99.0**, 97.3, 90.0, 96.7)	(**98.8**, 98.8, 98.2, 92.8, 98.0)	(97.5, **99.8**, 95.9, 91.5, 95.9)	(**98.9**, 98.2, 92.8, 93.9, 97.1)	N/A

2) Multi-view validation : Table.4.7 shows multi-view recognition results. The first protocol shows very close capability of MvDA-vc and pc-MvDA at 95.54% and 95.83% whereas MvDA is about 3% behind at 92.7%. For the second protocol, my variant exceeds MvDA by 2.21% and MvDA-vc by 1.49%. The near perfect results of the second protocol give us the same indication that it is not an inequitable method of comparison of MvL algorithms.

Figure.4.8 compares the performance of feature extractors regarding each action class in multi-view evaluation scheme combined with protocol 1. For this dataset, ResNet-50 3D persistently yields out-standing performance while ResNet-50 TA has the worst recognition rates in all action classes.

Table.4.8 compares the best investigated combination of ResNet-50 3D and pc-MvDA with state-of-the-art frameworks. The comparison is for reference only because of aforementioned difference in setup of number of views and cross validation scheme.

Table 4.7: Multi-view recognition comparison on MuHAVi dataset

Deep features	Protocol 1			Protocol2		
	MvDa	MvDA-vc	pc-MvDa	MvDa	MvDA-vc	pc-MvDa
C3D	81.80	96.05	**97.37**	88.39	99.61	**100**
ResNet-50 3D	99.07	99.12	**99.15**	**99.98**	99.96	99.89
ResNet-50 RNN	**96.18**	95.55	95.97	**100**	99.00	99.98
ResNet-50 TA	**90.45**	89.73	90.22	**99.92**	98.17	99.72
ResNet-50 AP	96.01	**96.75**	96.42	**100**	95.13	99.68

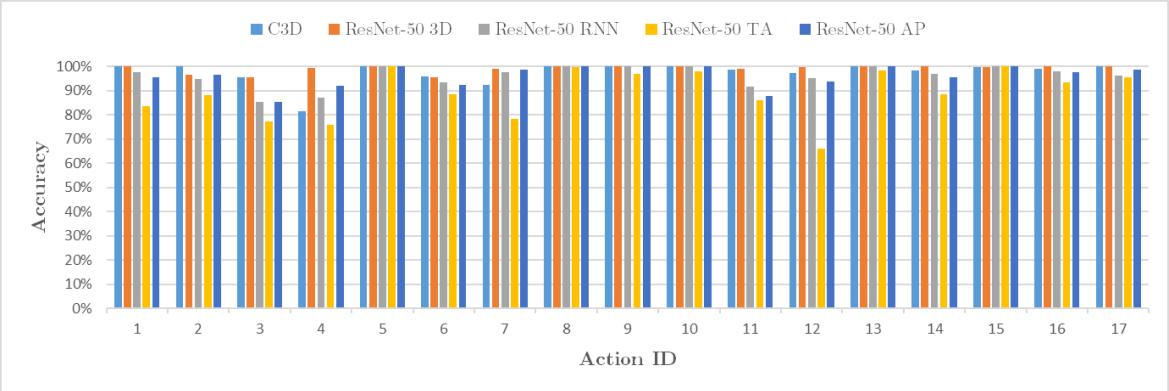


Figure 4.8: Comparison of accuracy on each action class using different deep features combined with pc-MvDA.

Table 4.8: Comparison of the proposed methods with SOTA methods on MuHAVi dataset according to the second evaluation protocol.

Methods	Cross-view	Multi-view
Wu et al. [62]	N/A	94.5
Zheng et al. [58]	94.88	99.8
Liu et al. [61]	N/A	91.2
Liu et al. [14]	99.91	99.8
Proposed method (ResNet-50 3D + pc-MvDA) [1]	99.90	99.88

4.5.3 Experimental results on MICAGes dataset

1) Cross-view validation: It can be seen in Table.4.9 that pc-MvDA outperforms in almost every cases. With the first protocol, the norm accuracy of pc-MvDA of 74.52% surpasses that of MvDA-vc (72.12%) by 2.4% and that of MvDA (64.94%) by a large margin of 9.58%. Especially in case of C3D features, the proposed algorithm achieved 90.2% recognition rate whereas MvDA returns only 67.13%.

Table 4.9: Cross-view recognition comparison on MICAGes dataset

Deep features	Protocol 1			Protocol 2		
	MvDA	MvDA-vc	pc-MvDA	MvDA	MvDA-vc	pc-MvDA
C3D	67.13	85.98	**90.20**	93.74	99.80	**100**
ResNet-50 3D	94.91	95.25	**95.62**	97.43	**99.97**	99.79
ResNet-50 RNN	58.01	61.64	**64.13**	100	100	100
ResNet-50 TA	53.58	**59.55**	58.62	96.70	99.48	**99.83**
ResNet-50 AP	51.07	58.19	**64.04**	99.73	**99.99**	99.89

The top ranking of features applies identically and radically proves the superiority of 3D convolution based clip-level feature extraction in video recognition: ResNet-50 3D at 95.62% and C3D at 90.2%. The rest are ResNet-50 RNN (64.13%), ResNet-50 AP (64.04%) and ResNet-50 TA (58.62%). For the second protocol, MvDA-vc (99.85%) and pc-MvDA (99.9%) are nearly even in performance while MvDA achieved 97.52%.

Table 4.10: Cross-view recognition results of different features on MICAGes dataset with pc-MvDA method. The result in the bracket are accuracies of using features C3D, ResNet-50 3D, ResNet-50 RNN, ResNet-50 TA, RestNet-50 AP respectively. Each row corresponds to training view (from view K1 to view K5). Each column corresponds to testing view (from view K1 to view K5)

Training \ Testing	K1	K2	K3	K4	K5
K1	N/A	(92.9, **95.5**, 69.4, 63.9, 65.1)	(93.7, **98.0**, 79.1, 75.8, 74.7)	(90.5, **97.6**, 64.5, 68.8, 70.8)	(89.2, **92.6**, 56.0, 43.7, 58.2)
K2	(84.7, **94.6**, 51.3, 41.4, 53.7)	N/A	(94.6, **97.6**, 78.9, 75.4, 77.5)	(91.3, **97.6**, 64.4, 68.5, 68.0)	(87.4, **92.7**, 56.1, 42.0, 55.6)
K3	(87.5, **94.9**, 51.9, 43.6, 52.7)	(93.0, **95.5**, 70.1, 64.7, 63.4)	N/A	(90.0, **97.6**, 63.2, 62.6, 67.5)	(88.9, **92.2**, 57.2, 41.8, 56.6)
K4	(86.7, **94.7**, 50.8, 48.9, 54.9)	(90.2, **95.5**, 71.5, 68.2, 61.8)	(92.5, **98.0**, 81.0, 76.0, 74.4)	N/A	(89.7, **92.3**, 54.3, 44.9, 56.6)
K5	(84.9, **94.9**, 48.4, 41.5, 57.9)	(90.4, **95.5**, 72.6, 64.1, 65.7)	(94.0, **97.8**, 79.1, 72.5, 76.3)	(91.9, **97.3**, 62.7, 64.1, 69.0)	N/A

[1]Results when choosing only 4 views Camera 1, Camera 3, Camera 4 and Camera 6 following the other works.

2) Multi-view validation : The multi-view recognition results in Table.4.11 shows an almost alike trend for both protocols. In general, pc-MvDA is 8.95% and 0.72% higher in accuracy for protocol 1, and 2.75% and 0.04% for protocol 2, in comparison with MvDA and MvDA-pc respectively.

In virtually every experiments of MICAGes and two earlier benchmark datasets, the proposed extension is superior. The average accuracies of pc-MvDA is, in comparison on protocol 1 (and protocol 2 resp.), 5.29% (1.21% resp.) higher than MvDA, 1.21% (0.62% resp.) better than MvDA-vc. Despite not being as intuitive and straightly intelligible as pairwise-covariance, the multi-view resemblance added in MvDA-vc achieved nearly the performance of pc-MvDA. However, this view-consistency can be easily splitted into pairwise terms and intergrated into the objective of pc-MvDA in future works for further analysis.

Table 4.11: Multi-view recognition comparison on MICAGes dataset

Deep features	Protocol 1			Protocol2		
	MvDA	MvDA-vc	pc-MvDA	MvDA	MvDA-vc	pc-MvDA
C3D	68.31	87.70	**89.99**	88.79	99.54	**100**
ResNet-50 3D	95.32	95.26	**95.54**	**100**	99.96	99.80
ResNet-50 RNN	57.50	63.05	**63.83**	99.90	**100**	99.97
ResNet-50 TA	54.43	**61.46**	58.25	96.72	99.49	**99.69**
ResNet-50 AP	50.93	60.20	**63.66**	**99.99**	99.93	99.67

To better study the behavior of investigated multiview analysis algorithms, t-SNE embedding of original private spaces, MvDA and pc-MvDA common space generated by protocol 1 are plotted in Figure 4.9. In these scatter plots, colors denote action classes, shapes as different views and train/test data are distinguished by border type. They explicitly denotes the surpassing capability of the proposed algorithm in finding a common space with prominent extra-class discrepancy. The data samples involved in training process are generally clustered in compact and separated blobs while testing data samples dissolve nearby. The better convergence of pc-MvDA compared to the baseline is usually only visible for harder features, whereas with the inputs of ResNet-50 3D features, which are already highly discriminated in private spaces, the improvement of pc-MvDA is negligible. Note that these illustrations of t-SNE embedding do not strictly depict an identical distribution of data.

MICAGes also reveals the robustness of 3D convolution to generate fine clustered private feature space because this dataset contains exclusively hand gestures, which take part in a relatively small region of the captured videos. ResNet-50 3D and C3D predominantly distances other features (Figure 4.10).

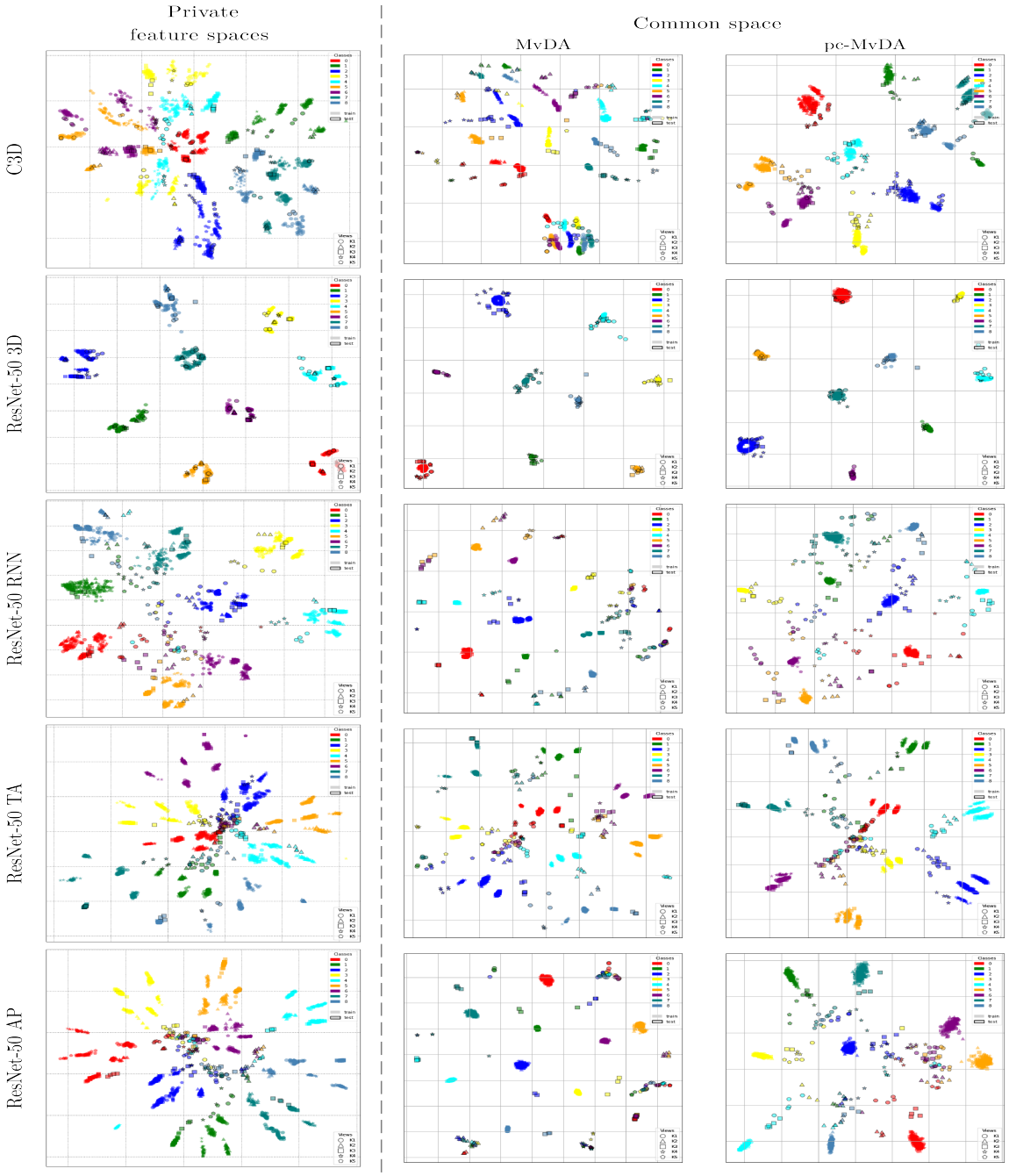


Figure 4.9: First column: private feature spaces stacked and embedded together in a same coordinate system; Second column: MvDA common space; Third column: pc-MvDA common space

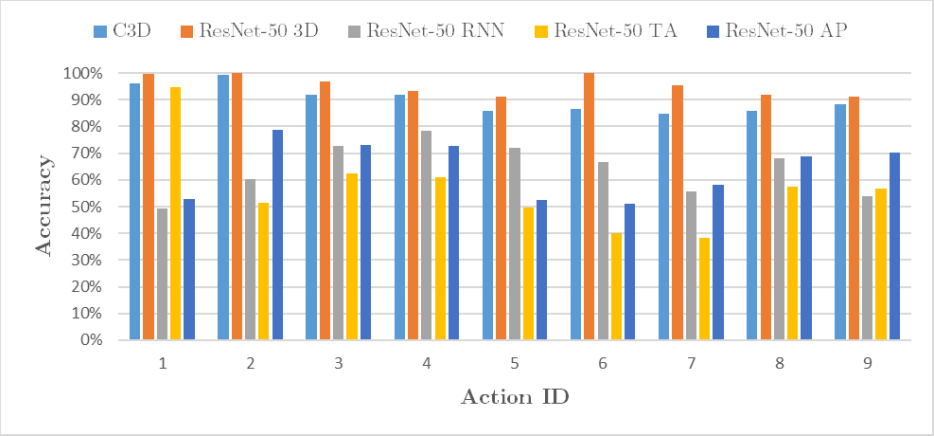


Figure 4.10: Comparison of accuracy on each action class using different deep features combined with pc-MvDA on MICAGes dataset

4.6 Summary

5 Conclusion

5.1 Accomplishments

This thesis has presented a novel framework for human action recognition. The framework integrates various successful deep features with multi-view discriminant analysis to deal with cross-view human action recognition. In particular, five deep models have been utilized: ResNet with three pooling techniques; ResNet-3D and C3D. These deep features have been universally used for action recognition from a common view but rarely utilized for evaluating cross-view HAR. Besides, three variations of multiview discriminant analysis: the original MvDA, MvDA with view consistency (MvDA-vc) and my proposed pairwise-covariance MvDA (pc-MvDA) have been investigated. Multiview analysis algorithms have been successfully deployed for cross-view recognition of static images but never applied for spatio-temporal features. The experiments show that the proposed algorithm achieved highest average accuracy (5.29% higher than MvDA and 1.21% higher than MvDA-vc).

5.2 Drawbacks

On the other hand, there remains some noticible limitations of the work done:

- The proposed framework is too cumbersome and could not be end-to-end trainable. Both MvDA, MvDA-vc and my contribution pc-MvDA are batch algorithms, in effect, it is essential to process whole training dataset at once to compute class means at each optimization step. This makes multiview analysis algorithms unsuitable to be a loss function that trains multiple large-scale neural networks concurrently, especially when input data is 3-dimensional spatio-temporal tensors. Existing works [27] and [20] integrated Fisher Loss to train relatively small-scale neural networks and only able to handle very small dataset of low-dimensional features.
- The experiments are conducted on relatively small and simple action datasets

given the potency of deep CNNs. As a results, recognition accuracies sometimes approach 99% $\rightarrow$ 100%, which make the task not challenging enough.

5.3 Future Works

To sum up, it is feasible to further explore different directions and improvements for the proposed work in this thesis to be more practically deployable:

- The framework is built and tested on fine-cut video datasets with the assumption of activation of a gesture detection module beforehand. To deal with real life continous video streams, a fast lightweight gesture detector must be applied and only activate the HAR once a gesture candidate is present. Another practical approach commonly used in the literature is to use continual multi-clip sampling instead of gesture detection combined with 16-frame sampling.
- As multiview analysis algorithms inherently allows each view to have different dimensions, other modalities such as optical flow or depth can be included and interpreted as new views so as to increase the robustness of overall framework.
- Test the framework with newer backbone CNN architectures, for example I3D [63], P3D [64], ResNeXt 3D [23], R(2+1)D [65], CSN [66].
- Evaluate on a larger and more challenging benchmark dataset, such as the multi-modal NTU dataset [67] which contains 120 action classes and more than 114,480 video samples in total.
- More research can be done to make the framework end-to-end trainable and eliminate the need of a final classifier. One potential approach is to make means of classes learnable, and optimize them simultaneously with the backbone networks as in [68].
- Since all the current multiview analysis algorithms are limited to samples from learnt views and viewpoint information must be a known priori, further research is needed to make the framework capable of recognizing a novel viewpoint.

Bibliography

[1] C. Olah, "Understanding lstm networks," 2015.

[2] F. Murtaza, M. H. Yousaf, and S. A. Velastin, "Multi-view human action recognition using 2d motion templates based on mhis and their hog description," *IET Computer Vision*, vol. 10, no. 7, pp. 758–767, 2016.

[3] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and vision computing*, vol. 60, pp. 4–21, 2017.

[4] H.-B. Zhang, Y.-X. Zhang, B. Zhong, Q. Lei, L. Yang, J.-X. Du, and D.-S. Chen, "A comprehensive survey of vision-based human action recognition methods," *Sensors*, vol. 19, no. 5, pp. 1005, 2019.

[5] D. M. Gavrila and L. S. Davis, "3-d model-based tracking of humans in action: a multi-view approach," in *Proceedings cvpr ieee computer society conference on computer vision and pattern recognition*. IEEE, 1996.

[6] F. Lv and R. Nevatia, "Single view human action recognition using key pose matching and viterbi path searching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007.

[7] D. Weinland, E. Boyer, and R. Ronfard, "Action recognition from arbitrary views using 3d exemplars," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007.

[8] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer vision and image understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[9] I. N. Junejo, E. Dexter, I. Laptev, and P. PÚrez, "Cross-view action recognition from temporal self-similarities," in *European Conference on Computer Vision*. Springer, 2008.

[10] B. Li, O. I. Camps, and M. Sznaier, "Cross-view activity recognition using hankelets," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012.

[11] J. Liu, M. Shah, B. Kuipers, and S. Savarese, "Cross-view action recognition via view knowledge transfer," in *CVPR 2011*. IEEE, 2011.

[12] J. Wang, X. Nie, Y. Xia, Y. Wu, and S.-C. Zhu, "Cross-view action modeling, learning and recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[13] Y. Kong, Z. Ding, J. Li, and Y. Fu, "Deeply learned view-invariant features for cross-view action recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 3028–3037, 2017.

[14] Y. Liu, Z. Lu, J. Li, and T. Yang, "Hierarchically learned view-invariant representations for cross-view action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 8, pp. 2416–2430, 2018.

[15] H. Rahmani, A. Mian, and M. Shah, "Learning a deep model for human action recognition from novel viewpoints," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 667–681, 2017.

[16] B. Thompson, *Canonical correlation analysis: Uses and interpretation*, Number 47. Sage, 1984.

[17] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[18] M. Yang and S. Sun, "Multi-view uncorrelated linear discriminant analysis with applications to handwritten digit recognition," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014.

[19] M. Kan, S. Shan, H. Zhang, S. Lao, and X. Chen, "Multi-view discriminant analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 188–194, 2015.

[20] G. Cao, A. Iosifidis, K. Chen, and M. Gabbouj, "Generalized multi-view embedding for visual recognition and cross-modal retrieval," *IEEE transactions on cybernetics*, vol. 48, no. 9, pp. 2542–2555, 2017.

[21] X. You, J. Xu, W. Yuan, X.-Y. Jing, D. Tao, and T. Zhang, "Multi-view common component discriminant analysis for cross-view classification," *Pattern Recognition*, vol. 92, pp. 37–51, 2019.

[22] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[23] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[25] D. Kong and C. Ding, "Pairwise-covariance linear discriminant analysis," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[26] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 03 1951.

[27] M. Kan, S. Shan, and X. Chen, "Multi-view deep network for cross-view classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[28] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008.

[29] G. Willems, T. Tuytelaars, and L. Van Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *European conference on computer vision*. Springer, 2008.

[30] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE international conference on computer vision*, 2013.

[31] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014.

[32] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese, "Lattice long short-term memory for human action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[33] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[34] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1510–1517, 2017.

[35] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards good practices for very deep two-stream convnets," *arXiv preprint arXiv:1507.02159*, 2015.

[36] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[37] V.-M. Khong and T.-H. Tran, "Improving human action recognition with two-stream 3d convolutional neural network," in *2018 1st International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*. IEEE, 2018.

[38] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[39] R. Christoph and F. A. Pinz, "Spatiotemporal residual networks for video action recognition," *Advances in Neural Information Processing Systems*, pp. 3468–3476, 2016.

[40] R. Li and T. Zickler, "Discriminative virtual views for cross-view action recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012.

[41] J. Zheng, Z. Jiang, P. J. Phillips, and R. Chellappa, "Cross-view action recognition via a transferable dictionary pair," in *bmvc*, 2012, vol. 1.

[42] J. Zheng and Z. Jiang, "Learning view-invariant sparse representations for cross-view action recognition," in *Proceedings of the IEEE international conference on computer vision*, 2013.

[43] J. Zhang, H. P. Shum, J. Han, and L. Shao, "Action recognition from arbitrary views using transferable dictionary learning," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 4709–4723, 2018.

[44] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, 1936.

[45] S. Akaho, "A kernel method for canonical correlation analysis," *CoRR*, vol. abs/cs/0609071, 2006.

[46] T. Diethe, D. R. Hardoon, and J. Shawe-Taylor, "Multiview fisher discriminant analysis," in *NIPS workshop on learning from multiple sources*, 2008.

[47] J. Rupnik and J. Shawe-Taylor, "Multi-view canonical correlation analysis," in *Conference on Data Mining and Data Warehouses (SiKDD 2010)*, 2010.

[48] Y. Zhao, X. You, S. Yu, C. Xu, W. Yuan, X.-Y. Jing, T. Zhang, and D. Tao, "Multi-view manifold learning with locality alignment," *Pattern Recognition*, vol. 78, pp. 154–166, 2018.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[50] J. Gao and R. Nevatia, "Revisiting temporal modeling for video-based person reid," *arXiv preprint arXiv:1805.02104*, 2018.

[51] I. C. Duta, B. Ionescu, K. Aizawa, and N. Sebe, "Spatio-temporal vector of locally max pooled features for action recognition in videos," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[52] K. Fukunaga, "Chapter 10 - feature extraction and linear mapping for classification," in *Introduction to Statistical Pattern Recognition (Second Edition)*, K. Fukunaga, Ed., pp. 441 – 507. Academic Press, second edition edition, 1990.

[53] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," *Computer vision and image understanding*, vol. 104, no. 2-3, pp. 249–257, 2006.

[54] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, no. 2, pp. 111–147, 1974.

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 8024–8035. Curran Associates, Inc., 2019.

[56] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al., "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[57] T. Du, "Vmz: Model zoo for video modeling," `https://github.com/facebookresearch/VMZ`.

[58] J. Zheng, Z. Jiang, and R. Chellappa, "Cross-view action recognition via transferable dictionary learning," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2542–2556, 2016.

[59] A. Ulhaq, X. Yin, J. He, and Y. Zhang, "On space-time filtering framework for matching human actions across different viewpoints," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1230–1242, 2017.

[60] C. Zhang, H. Zheng, and J. Lai, "Cross-view action recognition based on hierarchical view-shared dictionary learning," *IEEE Access*, vol. 6, pp. 16855–16868, 2018.

[61] C. Liu, Z. Li, X. Shi, and C. Du, "Learning a mid-level representation for multiview action recognition," *Advances in Multimedia*, vol. 2018, pp. 1–10, 2018.

[62] X. Wu and Y. Jia, "View-invariant action recognition using latent kernelized structural svm," 10 2012.

[63] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *CoRR*, vol. abs/1705.07750, 2017.

[64] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *proceedings of the IEEE International Conference on Computer Vision*, 2017.

[65] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[66] T. Du, H. Wang, M. Feiszli, and L. Torresani, "Video classification with channel-separated convolutional networks," 10 2019.

[67] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[68] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., 2016.