

An Investigation of Machine Learning-Augmented Vision Systems for Human Action Understanding

Chaolong Zhang

Submitted for the Degree of

Doctor of Philosophy

From the University of Huddersfield

University of
HUDDERSFIELD
Inspiring global professionals

School of Computing and Engineering

University of Huddersfield

Queensgate, Huddersfield, HD1 3DH, UK

June 2022

Copyright Statement

- I. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and he has given The University of Huddersfield the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.
- II. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.
- III. The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of copyright works, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Professor Zhijie Xu for his perpetual patience, continued encouragement, and enthusiastic supervision during this PhD programme. I am impressed by his persistence and responsibility in teaching and research. I will take him as a role model in my academic career in the future.

Special thanks go to Professor Yuanping Xu from Chengdu University of Information Technology, who led me to the door of scientific research on computer science and made constructive comments on my research work.

Last but certainly not least, I own my sincere gratitude to my family for their understanding and support, with whom I can share my joy of success and from whom I can gain motivation and encouragement.

List of Publications

List of refereed papers:

- [1] **Zhang, C.**, Xu, Y., Xu., et al. (2022). Concurrent Spatial-temporal Aggregation Model for Human Action Recognition. *Computer Vision and Image Understanding*. (Under Review).
- [2] **Zhang, C.**, Xu, Y., Xu, Z., Huang, J., & Lu, J. (2022). Hybrid handcrafted and learned feature framework for human action recognition. *Applied Intelligence*.
- [3] **Zhang, C.**, Xu, Y., Xu, Z., Gong, M., Guo, B., & Yao, D. (2020). An Augmented Treble Stream Deep Neural Network for Video Analysis. *International Conference Information Visualisation (IV)*, 301-306.
- [4] Xu, Y., **Zhang, C.**, Xu, Z., Zhou, J., Wang, K., & Huang, J. (2019). A generic parallel computational framework of lifting wavelet transform for online engineering surface filtration. *Signal Processing*, 165, 37-56.

Statement for the reproduction of publications:

The main content of Paper 1 to Paper 4 in the list has been reproduced in this thesis:

- The literature review in Paper 1 to 4 was reproduced in Chapter 1 and Chapter 2;
- The main content of Paper 2 was reproduced in Chapter 3;
- The main content of Paper 2 and 3 was reproduced in Chapter 4;
- The main content of Paper 1 was reproduced in Chapter 5.

I am the lead author for the four papers and was the sole PhD student. I carried out all research work and wrote and revised the manuscripts with minor changes from my co-authors.

Abstract

Recognising and understanding the complex visual world is the ultimate goal of intelligent vision systems. Computer vision and artificial intelligence have been a long-lasting research hotspot with increasing major discoveries and breakthroughs. Human action understanding is one of the crucial topics due to its potential value in both academia and industry. Various steep challenges remain due to semantically implicit and ambiguous definitions of video events and their inherent signal complexities from streamed videos ill-affected by target occlusion and variation of illumination conditions.

Classic strategies and techniques for addressing these critical challenges of human action understanding have been investigated in this research. An innovative machine learning-augmented analytical framework for visual behaviour understanding has been proposed. The corresponding operational pipeline first integrates the discrete wavelet transform technique into the dense trajectory model to gain more defining human action features. Then the end-to-end multimodality neural networks are deployed for automatic feature learning and action classification. Performance enhancement has been achieved through the innovation of an efficient two-stream aggregation network by adopting optical flow-guided features and spatial-temporal fusion blocks in a cascaded spatial and temporal space.

This research has also addressed the context-biased problem causing long aggravation to the deep-learning community when dealing with generalisation issues. A long-short-term motion encoding scheme is presented to interpret human actions based on their semantic meanings embedded in pose skeletons, which has greatly alleviated the open-set action recognition problem by introducing the Euclidean and Additive Angular Margin Loss.

To facilitate the real-world implementation of the devised human action understanding models and techniques, the state-of-the-art and future trends of edge computing have also been explored. Corresponding experiments have demonstrated the viability and effectiveness of open format-based transferrable model generation for rapid and mass deployment in live.

List of Symbols & Abbreviations

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
ACL	Arm Compute Library
ArcFace	Additive Angular Margin Loss
AI	Artificial Intelligence
ANN	Artificial Neural Network
BN	Batch Normalization
BOF	Bag of Feature
BoSVW	Bag of Spatio-visual Words
BOVW	Bag of Visual Words
BOW	Bag of Word
C ² LSTM	Correlational Convolutional LSTM
C3D	Convolutional 3D
CBP	Compact Bilinear Pooling
CCTV	Closed-circuit Television
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DAG	Directed Acyclic Graph
DGNN	Directed Graph Neural Network
DNN	Deep Neural Networks
DT	Dense Trajectory
DWT	Discrete Wavelet Transform
EP	Execution Providers
FC	Fully Connected
FCNN	Fully Connected Neural Network
FLOPs	Floating Point Operations per Second

FPS	Frames Per Second
FstCN	Factored spatio-temporal Convolutional Networks
FV	Fisher Vector
GCN	Graph Convolutional Network
GEMM	General Matrix Multiplication
GFLOPs	gigaFLOPS (10^9 FLOPs)
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HCI	Human Computer Interaction
HMM	Hidden Markov Model
HOF	Histogram of Optical Flow
HRI	Human Robot Interaction
HRNet	High-Resolution Net
I3D	Two-stream Inflated 3D CNN
iDT	improved Dense Trajectory
IoT	Internet of Things
IoU	Intersection-over-Union
IP	Intellectual Property
KLT	Kanade-Lucas-Tomasi
KVMF	Key Volume Mining Framework
LCR-Net	Localization-Classification-Regression Network
LDA	Latent Dirichlet Allocation
LRCN	Long-term Recurrent Convolutional Networks
LSF	Long-short-term Spatiotemporal Features
LSME	Long-short-term Semantic Motion Encoding
LSTM	Long Short-term Memory
LTC	Long-term Temporal Convolutions
MA	Multi-assignment
MBH	Motion Boundary Histogram
MLP	Multi-layer Perceptron

MTC3D	Multi-scale Trajectory-pooled 3D Convolutional Descriptor
NN	Neural Network
NLP	Natural Language Processing
NPU	Neural Processing Unit
NVDLA	NVIDIA Deep Learning Accelerator
ONNX	Open Neural Network Exchange
ORT	ONNX Runtime
P3D	Pseudo-3D Residual
PAF	Part Affinity Fields
PCA	Principal Component Analysis
PD	Platform-dependent
PIM	Platform Independent Model
PTQ	Post-Training Quantization
QAT	Quantization-Aware-Training
ReLU	Rectified Linear Unit
RMPE	Regional Multi-person Pose Estimation
RNN	Recurrent Neural Network
SDH	Software Defined Hardware
SFV	Stacked Fisher Vector
SIFT	Scale Invariant Feature Transform
SGD	Stochastic Gradient Descent
SIMD	Single Instruction Multiple Data
SoC	System-on-a-chip
STFB	Spatial-temporal Fusion Block
ST-GCN	Spatial Temporal Graph Convolutional Networks
STIP	Space-Time Interest Point
STV	Spatio-temporal Volume
STT	Spatial-temporal Texture
SURF	Speeded Up Robust Features
SVM	Support Vector Machine

TFLOPS	Trillion Floating-point Operations Per Second
TCN	Temporal Convolutional Network
TDD	Trajectory-pooled Deep-convolutional Descriptor
TIN	Temporal Interlacing Network
TS	Trajectory Shapes
UK	United Kingdom
VIBE	Video Inference for Body Pose and Shape Estimation

List of Figures

Figure 1-1. Five categories of human actions.....	3
Figure 1-2. The procedure of human action recognition systems.....	6
Figure 1-3. Summarisation and thesis structure.....	11
Figure 2-1. Pipeline and components of handcrafted approaches.	12
Figure 2-2. Pipeline and components of learning-based approaches.....	13
Figure 2-3. The 1D multi-level Mallat wavelet decomposition algorithm.	15
Figure 2-4. The 1D multi-level Mallat wavelet reconstruction algorithm.....	15
Figure 2-5. The demonstration of the multi-level 2D wavelet decomposition.....	15
Figure 2-6. Visualization of the two flow algorithm results.....	18
Figure 2-7. Visualization of human optical flow dataset and the results on both synthetic and scenes (Ranjan et al., 2020).....	18
Figure 2-8. Visualization of RNN Unit.....	24
Figure 2-9. Visualization of LSTM Unit.	24
Figure 2-10. The generic architecture of LRCN.....	25
Figure 2-11. A specific instantiation of the LRCN model for human action recognition.	26
Figure 2-12. An overview of Ng’s approach.	26
Figure 2-13. Five types of Feature Pooling Architectures.....	26
Figure 2-14. The five stacked LSTM layers architecture.	27
Figure 2-15. LTC-CNN based network architecture.	28
Figure 2-16 Sample frames from the action recognition datasets.....	37
Figure 3-1. The handcrafted feature processing and representations based pipeline of the human action recognition model.....	41
Figure 3-2. The processing steps of DWT-driven DT-based feature extractor.	42
Figure 3-3. A demonstration of DWT pre-processing for a video frame coming from the UCF 101 action dataset.....	42
Figure 3-4. Feature points extracted from an original spatial scale.....	43
Figure 3-5. Feature points extracted from DWT coefficients.....	43

Figure 3-6. The encapsulated STV block for storing feature trajectories.....	44
Figure 3-7. The computation progress of the HOG3D descriptor.	47
Figure 3-8. Producing $BoTF_{tc}$ instances based on the BoF and the CoTrans templates.	50
Figure 3-9. Visualization of trajectory results.	53
Figure 3-10. The obtained trajectories from a “walking” action video.	53
Figure 3-11. Visualization of optical flow and the corresponding motion boundaries.	54
Figure 4-1. The CNN-RNN based dual-stream network architecture.	61
Figure 4-2. The architecture of the two-stream concurrent interactive spatial-temporal aggregation model.....	63
Figure 4-3. The structure of an OFF layer.	67
Figure 4-4. STFB in a residual block pair.....	69
Figure 4-5. Accumulating visual and motion features of a video across time.....	72
Figure 4-6. The baseline network design of CNN-based optical flow estimation.....	75
Figure 4-7. The structure of a 3-level pyramid network.....	76
Figure 4-8. Visualization of optical flows estimation methods.....	78
Figure 4-9. Feature maps extracted from the “TaiChi” action video in the UCF dataset.	79
Figure 5-1. Examples of misleading and absent contexts.....	86
Figure 5-2. Examples of the human-masked-out video frames.	87
Figure 5-3. The proportions of the accuracy change per class action.....	88
Figure 5-4. The proposed long-short-term semantic motion encoding architecture for human action understanding.	90
Figure 5-5. Architectural element in a three-layer 2D TCN structure.....	96
Figure 5-6. The architecture of 2D TCN blocks.	96
Figure 5-7. The accuracy/complexity trade-off on NUT-60 action dataset.....	102
Figure 6-1. The architecture of Arm NN.	107
Figure 6-2. The architecture of ONNX Runtime.	108
Figure 6-3. The processing flow of ONNX runtime inference.....	109

Figure 6-4. The overall workflow of the AI on edge scheme.....	110
Figure 6-5. Case study of fully connected layer ONNX graphs.....	113
Figure 6-6. Case study of convolutional layer ONNX graphs.....	113
Figure 6-7. The comparison accuracy (in %) of original and quantized models.....	117

List of Tables

Table 3-1. The recognition accuracy rate (in %) of different features and event representations on the UCF 50 dataset.....	55
Table 3-2. Performance comparison to the state-of-the-art approaches on UCF 50, HMDB51 and JHMDB51 datasets (in %).	56
Table 4-1. Architecture of visual and motion streams.....	65
Table 4-2. The recognition accuracy of different CNN in the dual-stream deep learning architecture on the UCF 50 dataset.....	80
Table 4-3. The comparison results of OFF and baseline two-stream networks.....	80
Table 4-4. The classification results for STFB integration into different network locations.....	80
Table 4-5. The accuracy (in %) of different number of STFBs insertion on UCF-101.	82
Table 4-6. Comparison of various streams in combination with a 3D sub-network (in %).	82
Table 4-7. Performance comparison between the proposed aggregation model with other state-of-the-art methods on UCF101 and HMDB51 datasets.....	83
Table 4-8. Extensibility on UT-Interaction dataset.....	84
Table 5-1. Accuracy (in%) when testing the pre-trained DNN models on the Kinetics-400 dataset by using the original videos and masked videos, respectively.	88
Table 5-2. Classes with the increased accuracy (in %) on the original training set and tested on original and masked Kinetics-400 settings.....	89
Table 5-3. The architectures of the 3D CNN sub-network and spatial fusion network.	94
Table 5-4. The mean accuracy (in%) and computational performance of different backbones.....	100
Table 5-5. The mean accuracy (in%) of different pose methods.	100
Table 5-6. The mean accuracy (in%) of different sequence modelling methods.	102

Table 5-7. Comparison of mean accuracy (in %) between the proposed model with other state-of-the-art methods on the NTU-60 action dataset.	102
Table 5-8. Comparison of accuracy among skeleton-based methods in out-of-context datasets.	103
Table 5-9. The mean accuracy (in%) on unseen actions.	104
Table 6-1. Comparison of model size between fp32 and uint8 types.	116
Table 6-2. The developing and testing platform setups.	117
Table 6-3. The comparison processing time (in milliseconds) of original and quantized models on different computational platforms.	117

Table of Contents

An Investigation of Machine Learning-Augmented Vision Systems for Human Action	
Understanding	I
Copyright Statement.....	II
Acknowledgements	III
List of Publications.....	IV
Abstract	V
List of Symbols & Abbreviations.....	VI
List of Figures	X
List of Tables.....	XIII
Table of Contents	XV
CHAPTER 1 Introduction	1
1.1 Motivation	1
1.2 Background.....	2
1.2.1 Categories of Human Actions.....	2
1.2.2 Applications.....	3
1.2.3 Approaches	4
1.3 Key Challenges for Human Action Recognition	6
1.4 Project Objectives and Thesis Structure	8
CHAPTER 2 Literature Review	12
2.1 Pipeline for Human Action Recognition	12
2.2 DWT for Data Pre-processing	13
2.3 Handcrafted Feature Extraction.....	16
2.3.1 Spatial-temporal Features	16
2.3.2 Flow based Features	16
2.3.3 Trajectory Features	19
2.4 Feature Representation	20
2.4.1 Bag of Features.....	20
2.4.2 Fisher Vector	21
2.5 Action Classification	22
2.5.1 Support Vector Machine.....	22
2.5.2 Artificial Neural Network.....	22
2.6 Deep Learning Approaches	22
2.6.1 Deep Learning Techniques	23
2.6.2 Long-term Recurrent Convolutional Networks	24
2.6.3 Long Time Periods-based Networks	26
2.6.4 Long-term Temporal Convolutions	27
2.6.5 Two-stream Networks	28

2.6.6	3D CNN based Models.....	30
2.6.7	Learning Temporal Features.....	31
2.7	Skeleton based Approaches	32
2.7.1	Pose Estimation	32
2.7.2	Skeleton for Action Recognition	34
2.8	Model Inference on Edge Computing.....	36
2.9	Datasets.....	36
2.9.1	Traditional Datasets.....	36
2.9.2	Modern Datasets	38
2.10	Summary.....	39
CHAPTER 3	Feature Engineering for Video Analysis	40
3.1	Introduction	40
3.2	Overview System Design	41
3.3	DWT-based Decomposition	41
3.4	Motion Feature Extraction.....	43
3.4.1	Dense Trajectory Formation.....	43
3.4.2	Low-level Feature Extraction	44
3.5	Video Event Representation	47
3.5.1	Spatial-temporal Bag of Features	47
3.5.2	Soft Assignment	48
3.5.3	BoTF Formulation.....	49
3.6	Action Classification	50
3.6.1	Feature Fusion and Dimensionality Reduction.....	50
3.6.2	SVM based Classifier	51
3.7	Experimental Results.....	52
3.7.1	Visualisation of Trajectories.....	52
3.7.2	Camera Motion Removal Effect.....	54
3.7.3	Feature Descriptor Efficiency.....	54
3.7.4	Event Representation Validation.....	56
3.7.5	Comparison With the Other Approaches.....	56
3.8	Summary.....	56
CHAPTER 4	Multimodality Neural Networks.....	58
4.1	Introduction	58
4.2	Learning Video Features by DNN.....	59
4.2.1	Pre-trained Feature Adaptation.....	59
4.2.2	Dual-stream CNN-RNN Network	60
4.2.3	Training	62
4.2.4	Transfer Learning	62
4.3	Concurrent Spatial-temporal Network.....	63

4.3.1	The Overall Network Architecture	63
4.3.2	Baseline Two-stream Network	64
4.3.3	OFF Fundamentals	65
4.3.4	OFF Layers.....	66
4.3.5	OFF based Motion Stream.....	68
4.4	Spatial-temporal Aggregation.....	68
4.4.1	STFB.....	69
4.4.2	Stream Fusion	70
4.4.3	3D CNN Representation.....	71
4.4.4	Network Implementation and Training Strategy	73
4.5	Learning Optical Flow.....	74
4.5.1	CNN for Optical Flow Estimation.....	74
4.5.2	Spatial Pyramid Networks.....	75
4.6	Experimental Results.....	77
4.6.1	Visualisation of Feature Maps	77
4.6.2	Comparison of Pre-trained DNNs	79
4.6.3	OFF Efficiency	80
4.6.4	STFB Location	81
4.6.5	Numbers of STFB.....	81
4.6.6	Evaluation of 3D Sub-network.....	82
4.6.7	Comparison With the State-of-the-art Results.....	82
4.6.8	Applicability and Extensibility.....	83
4.7	Summary.....	84
CHAPTER 5	Towards Understanding Human Actions.....	85
5.1	Introduction	85
5.2	Understanding the Biases for Action Recognition	87
5.2.1	Human Masked Data Processing.....	87
5.2.2	Biased Models in Action Recognition.....	87
5.2.3	Analysis and Discussion.....	89
5.3	Encoding Semantic Human Actions.....	90
5.3.1	Human Pose Sequence Extraction.....	90
5.3.2	3D Pose Heatmap	91
5.3.3	Long-short-term Learning Strategy	92
5.3.4	Short-term Semantic Motion Encoder.....	93
5.3.5	Long-term Semantic Action Encoder	95
5.4	Action Recognition.....	96
5.4.1	Softmax-based Classification	96
5.4.2	Recognition for Unseen Actions.....	97
5.4.3	Spatial Fusion	99

5.5	Experimental Results.....	99
5.5.1	Evaluation of Backbones	99
5.5.2	Evaluation of Pose Methods	101
5.5.3	Evaluation of Sequence Modelling.....	101
5.5.4	Comparison with State-of-the-art Methods	102
5.5.5	Comparison of Out-of-Context Dataset.....	103
5.5.6	Evaluation of Unseen Actions	103
5.6	Summary.....	104
CHAPTER 6	Model Inference on Edge Computing.....	105
6.1	Introduction	105
6.2	Computational Platforms.....	106
6.2.1	GPU	106
6.2.2	Arm NN	106
6.2.3	NPU	107
6.3	Platform Independent Model Design.....	107
6.3.1	ONNX.....	107
6.3.2	ONNX Runtime.....	108
6.4	Workflow of AI on Edge.....	109
6.5	Model Quantization	110
6.5.1	Concept of Quantization.....	110
6.5.2	Case: Fully Connected Layer	112
6.5.3	Case: Convolutional Layer	114
6.6	Model Partitioning.....	114
6.7	Experimental Results and Validation	115
6.7.1	Evaluation of Quantization Methods.....	115
6.7.2	Evaluation on Accelerators.....	116
6.8	Summary.....	117
CHAPTER 7	Conclusion and Future Work.....	119
7.1	Contributions to Knowledge.....	119
7.2	Future Work.....	120
References	123

CHAPTER 1 Introduction

1.1 Motivation

Recognising and understanding the complex visual world is a relatively easy task for the human visual system, but it is complicated for computer systems (Li et al., 2009). Computer vision has been a long-lasting research hotspot for about half-a-century with prominent discoveries and breakthroughs in every decade, namely a few, pictorial and geometrical representation in the 70s, quantitative image and scene analysis in the 80s, recognition in the 90s, feature engineering at the turn of the millennium, and deep learning in the 2010s. Research on computer vision systems has drawn wide attention from academia and industry, primarily because of the incrementally growing number of closed-circuit surveillance television (CCTV) cameras that produce an enormous amount of video data every second. Discovering semantic information from these video data has potential value in daily life, public safety, and industrial areas, while manual data processing is critical, painstaking, and not scalable. Consequently, increasing achievements have been gained in computer vision, such as image classification, object tracking, and facial recognition have achieved great successes (Felzenszwalb et al., 2010; Krizhevsky et al., 2012; Li et al., 2020). However, these algorithms and approaches are still struggling to cope with the demands of the applications of understanding various complex scenes and activities. There is no single, universal, intelligent, flexible, and robust approach to recognise complicated human activities. Human behaviour analysis or human action recognition is one of the most intriguing research areas in computer vision due to its wide range of applications in abnormal behaviour detection, novel human-computer interaction (HCI) design, intelligent video surveillance, healthcare system, and even game and entertainment. However, human action recognition remains a challenging task due to the semantic implicit and ambiguous definitions of video events, e.g., the classification and categorisation of individual and crowd motions (Sigurdsson et al., 2017), never mention the inherent signal complexities from recorded or streamed videos' ill-affected by target occlusion

and variation of illumination conditions (Sargano et al., 2017; Sigurdsson et al., 2017). This research aims to extract, model, and recognise motion patterns to build a general-purpose, high-performance, and flexible machine vision system for human action recognition and interpreting human actions based on their semantic definitions. It is anticipated that the contributions made in this research will be valuable for real-world applications and problem-solving such as autonomous vehicles, public security, game and metaverse. These methodologies will push a new paradigm for edge intelligence where the Internet of Things (IoT) is evolving to the Internet of Intelligent Things, and to the Intelligent Internet of Intelligent Things.

1.2 Background

1.2.1 Categories of Human Actions

Intuitively, an action is considered a human agent performing a sequence of basic or atomic movements, so recognising actions from still images is very difficult and has low accuracy. In contrast, a video contains sequences of frames representing one or more movements; thus, researchers mainly focus on recognising actions from videos.

According to human behaviour complexity and semantic definition, human actions can be classified into five categories (Sargano et al., 2017), i.e., gesture, individual action, human-human interaction, human-object interaction, and group activities, as shown in Figure 1-1. A gesture is a basic movement of human body parts that presents some meanings, e.g., “head shaking”, “hand waving”, and the OK gesture. Individual action is performed by a single person, “walking”, “running”, “jumping”, and “Tai Chi” are cases of it. Interactions are performed by at least two actors that can be divided into human-human and human-object interactions, e.g., “handshaking”, “ice dancing”, and “wrestling between two persons” are the former interactions, whilst “playing the guitar”, “golf driving” and “a person uses a phone” are the latter case. Group activity, also called crowd behaviour, is performed by a group of people, containing typically gestures, individual actions, and interactions, e.g., “cheerleading”, “marathons”, and “a crowd of people dispersing” are cases of it.



Figure 1-1. Five categories of human actions.

1.2.2 Applications

Human action recognition is one of the important research areas in computer vision because of its wide range of potential areas (Guo & Lai, 2014; Sargano et al., 2017; Sigurdsson et al., 2017), including intelligent video surveillance, HCI, and autonomous vehicles, etc.

a) Intelligent Video Surveillance

During the last decade, there is an increasing number of cameras have been set up. For example, up to 2018, it was estimated that approximately 5.9 million CCTV video cameras had been deployed in city centres, bus and railway stations, airports, supermarkets, and even private areas in the UK, averagely of one camera for every 11 people in Britain. These cameras produce a great deal of video data every second. However, these video data are limited values with conventional surveillance systems and manual video analysis platforms since they require laborious human monitoring and process an insufficient quantity of “shallow” information, e.g., the systems can only detect the change of motion and background. In contrast, intelligent video surveillance and analysis systems, driven by modern computer vision and artificial intelligence (AI) techniques, aim to analyse the semantic representations automatically and recognise events intelligently from streaming videos. With these advantages, the video analysis and monitoring workload will be significantly reduced, while more high-level and semantic information and activity patterns are discovered.

b) Human-Computer Interaction

Human gestures and actions provide natural ways to interact with robots and computers without using a keyboard and mouse. Vision-based HCI is becoming very popular in home and industry because users do not be required to remember any

instructions and operation steps of mouse clicks. They just perform natural actions with their body to express purposes and instructions (Kong & Fu, 2016). With this property, HCI applications require real-time data processing and response, i.e., when a person acts, the computer should recognise it and give feedback immediately. Furthermore, since robots are becoming a part of our lives, robots must be capable of understanding human actions and behaviours and even interacting and cooperating with humans. In this case, human-robot interaction (HRI) typically uses the camera installed in the robot for real-time video capturing, and the on-chip algorithms are performed to recognise human actions. Unlike the offline video analysis, this application requires a real-time perception of human activities and identifying the action before its completion, which is a complex challenge.

c) Autonomous Vehicles

Human action recognition techniques are also applied to assist drivers. It is a reasonable solution to avoid accidents by recognising and alerting the violation behaviours of drivers, e.g., smoking, eating, and answering the phone while driving the vehicle. More significantly, the self-driving car requires recognising the actions of pedestrians to determine the following operations.

1.2.3 Approaches

Historically, there are two main research strategies for human action recognition, i.e., 1) using “handcrafted” features for representing and identifying action types, and 2) using “learned” features in an end-to-end manner for classifying behaviours (Herath et al., 2017). The prior follows a bottom-up strategy, which consists of three phases: foreground detection, feature extraction and representation, and action classification. For instance, the Gaussian Mixture Model (GMM) method is usually applied for background and foreground detection (Chauhan & Krishan, 2013); the Scale Invariant Feature Transform (Ju et al., 2009), Harris detector (Laptev, 2005) and dense sampling (Wang et al., 2013) functions are then performed for visual feature extraction and representation; and finally, the machine learning based classifiers are used to predict the action types. Among these works, the motion trajectory-based approaches have

shown significant breakthroughs compared to frame-to-frame processing methods coming from the traditional image processing era because the motion-trajectory methods process spatial and temporal features simultaneously, such as optical flow, trajectory shapes, and time series. Moreover, spatial-temporal features can be obtained from spatio-temporal volume (STV) data structure in a 3D coordinates system denoted by x -, y - and t - (time-dimension) axes, which is followed by spatial-temporal texture (STT) (Hao et al., 2017) and sequence algorithms for feature representation and classification, e.g., Hidden Markov Model (HMM) (Bahl et al., 1986). Among these algorithms, the dense trajectory (DT) and its enhanced models (e.g., improved dense trajectories (iDT) (Wang & Schmid, 2013) and stacked fisher vector (SFV) (Peng et al., 2014) offer improved accuracy and recall rate on human action types that are defined not just by their rigid postures over video frames, but corresponding information on motion pattern and even camera pose. DT-based models were mainstream research strategies in the pre-deep learning era. Nevertheless, the handcrafted approaches heavily depend on sophisticated feature engineering design and domain-dependent representations. As a result, the handcrafted feature models are weak in generalisation and robustness. For instance, the SFV model shows high accuracy on the YouTube action dataset (93.38%), but it gains relatively poor performance on the HMDB51 action dataset (66.79%) (Peng et al., 2014).

Machine learning, especially the recent deep learning wave, supports direct feature abstraction and pattern recognition that has become a mainstream pipeline due to its brute force approach and robustness for certain application tasks such as image classification (Chang et al., 2017) and object detection (Liu et al., 2016; Redmon et al., 2016). The ground-breaking Convolutional Neural Network (CNN), the foundation of deep learning, avoids the laborious feature crafting steps, hence initiating a paradigm shift from an “engineering” one to an “architectural” one. Recently, deep learning based human action recognition has seen significant breakthroughs, including the two-stream (spatial and temporal) CNN and 3D CNN models (Ji et al., 2013; Simonyan & Zisserman, 2014; Tran et al., 2015). However, these designs only track a short period for the temporal features in video clips (normally processing 16 frames), leading to

difficulty when handling “longer” video events. Another interesting work focused on handling sequential temporal information in videos by integrating Recurrent Neural Networks (RNN) and long-term temporal convolution techniques (Li et al., 2017; Varol et al., 2018). To date, many deep learning methods have since been piloted, producing the varied level of “performance gain” in different signal spectrums, from spatial, frequency and temporal.

1.3 Key Challenges for Human Action Recognition

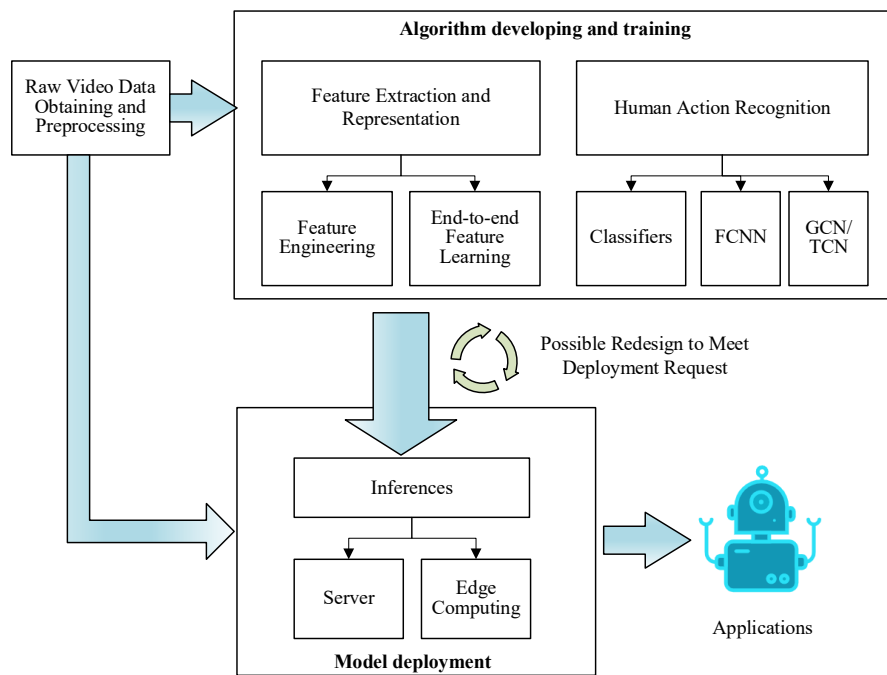


Figure 1-2. The procedure of human action recognition systems.

As illustrated in Figure 1-2, the four main phases involved data capturing, algorithm developing and training, and model deployment must be implemented to obtain the results for human action recognition systems. Raw video data received from either surveillance cameras or movies is pre-processed by digital image processing methods such as denoising and background subtraction. For the second phase, visual and motion features are extracted by either handcrafted or end-to-end learning algorithms and encoded as high-level semantic representations. Features are fused as holistic descriptors for action classification by using machine learning-based classifiers, fully connected neural network (FCNN), graph convolutional network (GCN) (Yan et al., 2018) and temporal convolutional network (TCN) (Bai et al., 2018), etc. Once the

model is obtained, the next phase is deploying the algorithm into target platforms such as graphics processing unit (GPU) servers and edge computing devices for real-world applications. Redesigning and adopting the algorithm may be involved to meet the specific deployment requests. This research tackled three essential issues in the algorithm development and deployment phases. The main challenges of human action recognition systems are listed as follows.

- In the research of human action recognition and video analysis, a wide range of features and descriptors have been explored. One question remains on identifying the approximate features and descriptors for better performance on human action recognition, which includes both feature engineering and designing sophisticated neural networks for feature extraction and high-level semantic abstraction. Therefore, the critical challenge is investigating both handcrafted and learned features to achieve successful motion detection and classification. The most significant challenge is extracting and representing temporal information in videos which plays a fundamental factor in event representation. Another open-up problem is how to bridge the semantic gaps between handcrafted features often carrying distinctive “meanings” and the automated latent ones “hidden” in the ever-sprawling webs and deeper layers.
- As the research continues, the explainable of models becomes a key challenge, i.e., exploring the evidence or parameters of the perception results, and investigating the differences between computer vision and human vision systems. Recent models have shown reasonable performance on video action recognition according to the benchmarks (Feichtenhofer, 2020; Ji et al., 2013; Jiang et al., 2021; Mao et al., 2021; Tran et al., 2018; Xu et al., 2019a). However, these approaches tend to model static contexts such as objects and scenes instead of interpreting human actions based on their semantic definitions (Weinzaepfel & Rogez, 2021). For instance, the model tends to predict the “shooting goal” result on football field background videos. In contrast, humans have different strategies for understanding the actions, e.g., it is straightforward to distinguish the actions of “yoga” and “shooting”,

regardless of whether the actions are played indoors or on a football field. An interesting example is a mime performed by body language given by mime artists without using any props. Human vision can still understand the typical actions despite the absence of contexts, but computer vision shows weak performance in that case.

- After the action recognition algorithms are obtained, another major challenge is encountered, i.e., how to deploy models into edge computing systems such as mobile phones and autopilots, which are powered by embedded GPUs and neural processing units (NPU) (Lee, 2021; Shi et al., 2016), known as AI on edge, which requires real-time data processing on resource-constrained and heterogeneous edge devices while maintaining high performance obtained in the developing environment, hence to support the real-world problem-solving.

1.4 Project Objectives and Thesis Structure

To tackle the three challenges encountered above, this research aims to investigate some possible innovative computer vision and AI techniques for the human action recognition task by hybrid handcrafted and learned features to improve the performance of real-time vision systems. The main objectives of this research are listed as follows.

- Exploring the state-of-the-art handcrafted features and descriptors which will bring effective man-made “meaningful” contexts. The handcrafted model processes the raw video data by a sequence of computer vision and machine learning algorithms, including pre-processing by discrete wavelet transform (DWT) technique, dense sampling for feature point extraction and tracking, STV data construction, high-level video feature describing and representation.
- Devising novel multimodality neural network architectures with advanced techniques for video analysis to support automatic feature extraction and action classification and enhance its accuracy, computational performance, and generalisation. The method learns motion information from videos with an end-to-end scheme by the multiple network streams constructed by CNN

and long short-term memory (LSTM) with advanced loss functions and learning strategies.

- Evaluating the influence of biases of the learned models contributing to the final recognition results, so as to design the semantic action encoding method for better understanding the human action definitions. This object aims to understand generically semantic representation and interpret human actions from pose skeletons.
- Deploying the algorithms into edge computing systems for real-time model inference. The open neural network exchange (ONNX, 2021), which is an open format built to represent machine learning models, is applied for platform-independent model (PIM) design. The quantization strategy will be investigated to map a large machine learning model to a lightweight one suitable for real-time data processing on resource-constrained platforms such as NPU-based edge computing.

The contributions made in this thesis are summarised below:

- 1) In the feature extraction and representation phase, a DWT-driven DT model is devised to dissect videos in the form of multi-resolution representations and extract textural features representing motion characteristics for harnessing their distinctive characteristics over the spatial and temporal spaces. Then a Fisher Vector and bag-of-temporal-features-based model are proposed to encode holistic event representation. This contribution attempts to handle various orientations and separable frequencies in multiple scales of video actions and enables video-based event representation.
- 2) In the learned-based feature learning phase, a concurrent spatial-temporal aggregation model is introduced to improve feature extraction effectiveness and efficiency and learn the hybrid spatial and temporal pattern in a video. The appearance features rich in the modality of RGB video frames while the motion pattern is formulated in optical flows. Therefore, the devised multimodality neural network is capable of the coarse-to-fine scene and

motion interactions from the joint spatial-temporal exploitation, which is critical in the action recognition process.

- 3) To explain what knowledge and biases are truly learned from a model, this research evaluated the influence of biases of the learned model, which shows that most models tend to model contextual features instead of interpreting inherent human actions. Therefore, a human pose skeleton-based model is developed for encoding long-short-term action representation to understand the semantic definition of video actions. This contribution attempts to solve the open-set action recognition challenge in modern AI-powered applications, where the large-scale training dataset is unavailable.
- 4) In the model inference phase, the methodologies of ONNX-based PIM design, model quantization, graph partitioning and edge computing are introduced. This contribution attempts to tackle the challenges of low-precision arithmetic, computational graph optimization, parallel execution, and hardware acceleration in resource-constrained and heterogeneous systems, suggesting a novel solution for the research on edge intelligence.

The rest of this thesis is organised as follows, with all chapters structured as shown in Figure 1-3.

- Chapter 2 offers a comprehensive literature review of the preliminaries and related works in the research area.
- Chapter 3 explores the work on handcrafted feature extraction, description, and event representation.
- Chapter 4 describes the methodology and implementation of the multimodality deep neural network design. Specifically, this chapter presents an end-to-end model for both visual and motion feature extraction, high-level semantic information abstraction, and feature fusion for action classification.
- Chapter 5 investigates the impact of biases contributing to the recognition results. Based on the observation, this chapter designs the long-short-term semantic motion encoding method to interpret the human action definitions from skeleton data and human pose sequences.

- Chapter 6 introduces the open format machine learning representation with a platform-independent model. Then this chapter moves to quantize a heavy algorithm running on large GPUs to a lightweight model for enabling efficient, high-performance computation on small mobile devices and partition a computational graph into sub-graphs for optimisation in heterogeneous edge computing environments.
- Chapter 7 summarises the research and discusses the ongoing future works.

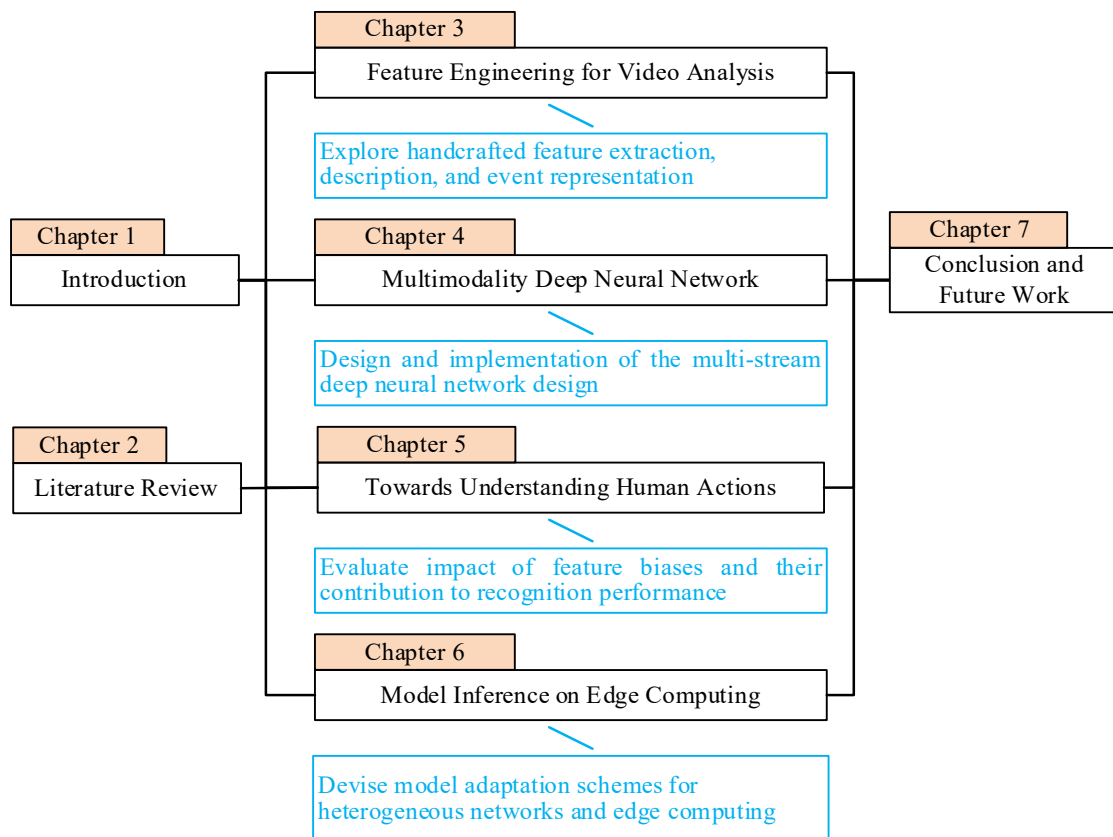


Figure 1-3. Summarisation and thesis structure.

CHAPTER 2 Literature Review

2.1 Pipeline for Human Action Recognition

As stated before, the traditional approach of the entire pipeline for human action recognition includes video capturing, video data pre-processing, feature extraction, feature representation, and classification, as shown in Figure 2-1. In addition, model deployment may also be involved when tackling real applications. Video data is normally recorded from either surveillance cameras or movies, and the video streams and frames are pre-processed by signal processing and digital image processing methods such as background subtraction and filtrations. Feature extractors and descriptors are manually designed for video feature detection and assembly. Finally, a trainable classifier is integrated for event classification that outputs action labels.

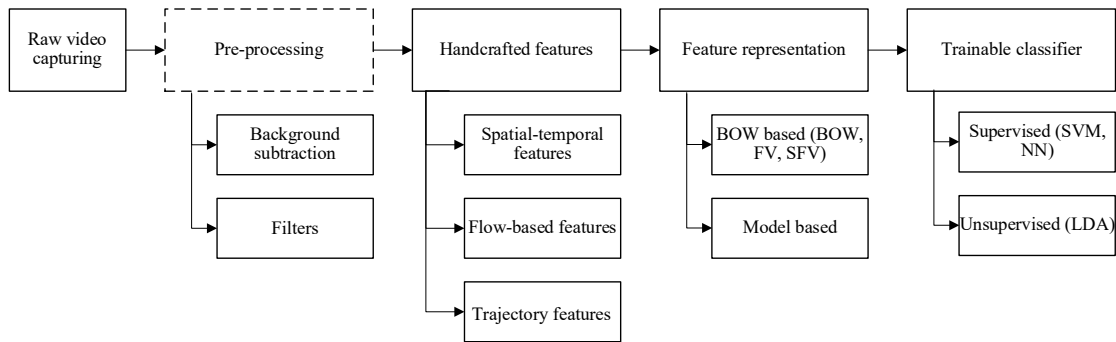


Figure 2-1. Pipeline and components of handcrafted approaches.

In the video data pre-processing phase, GMM-based methods are applied for background subtraction, and signal filters such as Gaussian filter and DWT are also integrated for image denoising and frequency analysis to generate high-quality frames or domain-specific images. Traditional feature methods can be classified into spatial-temporal features, flow-based features, and trajectory features, as illustrated in Figure 2-1. The features obtained from videos are further assembled by descriptors for high-level event representation. These approaches include Bag-of-Word (BOW) (Bolovinou et al., 2013) and Fisher Vector (FV) (Peng et al., 2014), etc. In terms of action classification, the supervised Support Vector Machine (SVM) is a dominant model that has shown superior performance over others on most classification tasks, such as image classification and object detection (Chandra & Bedi, 2018). In recent years, neural

network (NN) and Latent Dirichlet Allocation (LDA) models have emerged as effective methods for classification applications (Liu et al., 2011; Vishwakarma & Kapoor, 2015).

In contrast, the learning-based approaches, especially the recently emerged deep learning-based methods (Herath et al., 2017; Szegedy et al., 2015a), eliminate the handcrafted feature detectors and descriptors by using a trainable feature detector before a learnable action classifier is integrated to introduce a so-called end-to-end manner of feature extraction and action classification, as shown in Figure 2-2.

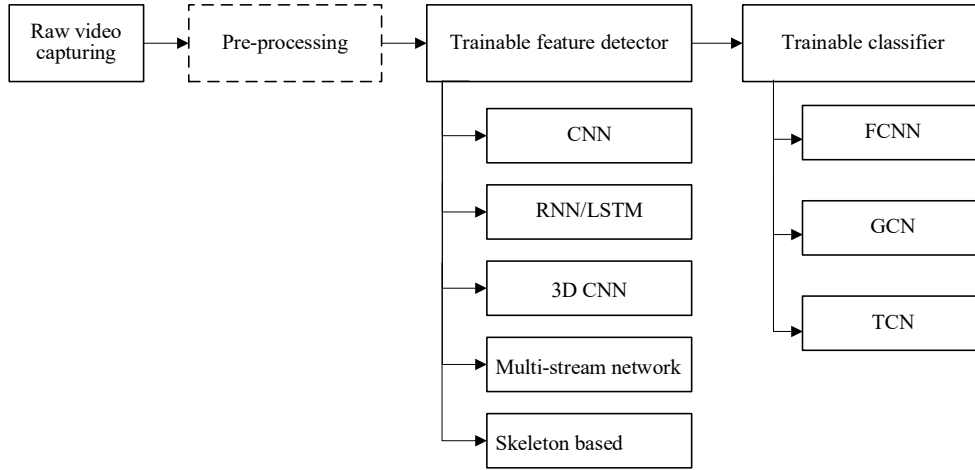


Figure 2-2. Pipeline and components of learning-based approaches.

2.2 DWT for Data Pre-processing

DWT has been widely applied in research areas such as signal processing and computer vision. Its multi-scale analytical ability is unparalleled when abstracting region-of-interest and features from real-world problems (Xu et al., 2019b).

The fundamental thought behind wavelet analysis is converting a complex frequency analysis into a simple scalar analysis. In 1D continuous signal processing, mother wavelets $\Psi(x)$ can be constructed by the scalar factor and shift parameter, as shown in the following:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right), \quad 2-1$$

where a is the scalar that indicates dynamic transmission bandwidths or covers different frequency ranges, and b is the shift parameter defining the time location centre of a wavelet. In digital signal processing applications, signals are discrete data sets, so the

scalar factor and shift parameter also take discrete values. Discrete wavelet is formulated as the following:

$$\psi_{j,k}(x) = a^{-j/2} \psi(a^j x - k), \quad 2-2$$

where j is the scalar factor that defines the corresponding bandwidth or the range of frequency, k is the shift parameter, and a indicates the scalar that is the same as Equation 2-1. A fast wavelet transform method is applied to the first-generation wavelet for decomposition (forward DWT) and reconstruction (inverse DWT) called the Mallat algorithm, which is also known as the two-channel sub-band filter or convolution scheme. For 1D discrete signal data, the Mallat decomposition algorithm can be defined as follows:

$$\begin{cases} A[n] = \sum_m X[m]H[2n-m] \\ D[n] = \sum_m X[m]G[2n-m] \end{cases}, \quad 2-3$$

where X denotes the raw signal, the combination of H and G is called the filter-bank in wavelet decomposition, H is the low-pass coefficient while G is the high-pass filter coefficient, A and D represent approximation coefficients (low-frequency) and detail coefficients (high-frequency), respectively. The bandwidths of wavelet coefficients A and D in the filter outputs are half of the bandwidth of the input data, which allows down-sampling of the outputs A and D without losing any information. It implements the multi-level decomposition by using A as input data for performing the wavelet decomposition in the next level. The basic concept of the 1D multi-level Mallat wavelet decomposition algorithm is illustrated in Figure 2-3. Inverse DWT can be used to reconstruct the signal from wavelet approximation coefficients and the corresponding detail coefficients, as shown in the following:

$$X[i] = \sum_m A[m]h[2m-n] + \sum_m D[m]g[2m-n], \quad 2-4$$

where the combination of h and g is filter-bank in wavelet reconstruction, and h is low-pass filter coefficients while g is high-pass filter coefficients. Definitions of A , D and X are the same as the Equation 2-3. There are two steps involved in the wavelet reconstruction, i.e., up-sampling and filtering. The up-sampling stage extends the

length of A and D by adding zeros in the alternate data values of them, respectively, and filtering performs convolution on the up-sampling outputs with filters h and g , respectively. Then the raw signal can be obtained by summing these convolution results. The basic idea of the 1D multi-level Mallat wavelet reconstruction algorithm is illustrated in Figure 2-4.

In terms of image processing, 2D DWT can be realised through the two-stage 1D wavelet transform along its x- and y-axes separately and concurrently. With these properties, 2D DWT decomposes 2D data into approximation coefficients (A) and detailed coefficients along horizontal (H), vertical (V) and diagonal (D) directions, respectively. Multi-level DWT is implemented by applying A as the input data and continuously performing 2D DWT on the next level. As a demonstration, Figure 2-5 shows that A_1 obtained from the raw data (A_0) is applied as the input data for performing 2D wavelet decomposition on the next level; hence the multi-level 2D wavelet decomposition can be implemented. 2D wavelet reconstruction also performs vertical 1D wavelet reconstruction for each column and horizontal 1D reconstruction for each corresponding row of a 2D input signal in sequence separately and concurrently.

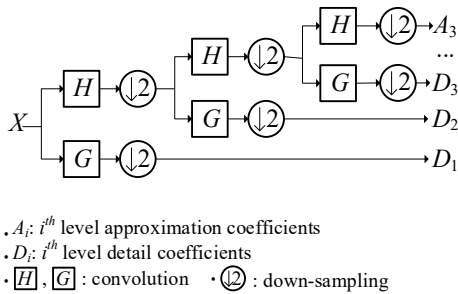


Figure 2-3. The 1D multi-level Mallat wavelet decomposition algorithm.

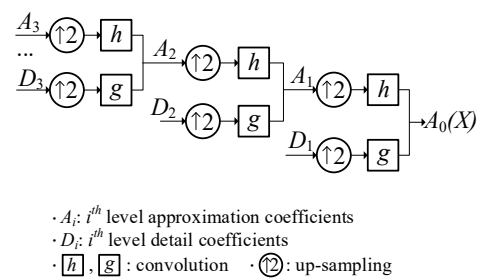


Figure 2-4. The 1D multi-level Mallat wavelet reconstruction algorithm.

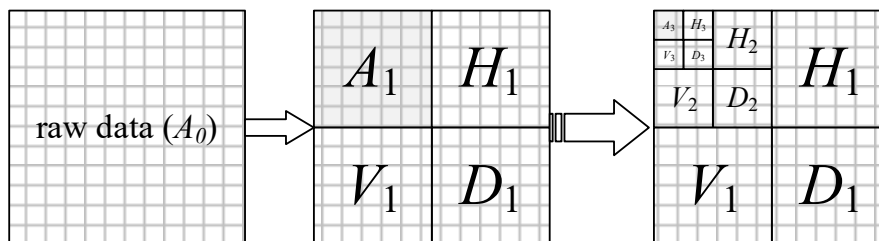


Figure 2-5. The demonstration of the multi-level 2D wavelet decomposition.

2.3 Handcrafted Feature Extraction

2.3.1 Spatial-temporal Features

When the human visual system recognises an image, different pixel areas play variance roles in understanding the whole context. For instance, the white background is less valuable, while the human and object areas are worth to be attention. In addition to that, the edges of humans play critical roles in recognising actions. Based on this consideration, researchers carried out a lot of efforts to achieve similar performance on computer vision systems. One of the fundamental approaches is the space-time interest point (STIP) detector which detects interest pixels in images and assigns them different weights and meanings. Laptev (2005) introduced a STIP model by extending the Harris detector with a significantly improved detection rate. Sipiran and Bustos (2011) improved the Harris operator to the Harris 3D model that can extract interest points from 3D data volumes effectively. Harris and Harris 3D models are sparse detectors which mainly extract local features. To handle global features, Wu et al. (2010) proposed a Scale Invariant Feature Transform (SIFT) based model to extract feature points. SIFT can robustly extract features from images because of its invariance to uniform scaling, orientation, and illumination changes. However, SIFT fails to handle 3D data volumes (e.g., videos). Liu et al. (2011) extended SIFT to 3D space that can extract interest points from 3D space-time video volumes efficiently. Generally, the space-time feature point approaches have shown sound effectiveness. It is suitable for recognising simple movements such as “hand waving” and “walking” actions in the KTH action dataset (Laptev & Lindeberg, 2003).

2.3.2 Flow based Features

The mainly used flow-based method is the optical flow which is a significant feature for video processing and motion analysis. Lots of research and literature suggest that optical flow plays an essential role in encoding movements and recognising human actions (Peng et al., 2014; Sun et al., 2010; Wang & Schmid, 2013).

a) Traditional optical flows

Optical flow was first proposed by Gibson (1950) to describe the visual stimulus provided to animal movements. Lucas and Kanade (1981) proposed a local optical flow algorithm called the Lucas-Kanade method, which is a popular optical flow algorithm due to the less sensitivity to image noise, and it resolves the inherent ambiguity of the optical flow equation. However, this method computes optical flow for a sparse feature set, so it cannot provide uniform region flow information of an image. To tackle this disadvantage, dense optical flow algorithms were proposed. Farneback (2003) presented a dense optical flow calculated in two continuous frames. This method first approximates each neighbourhood of two successive video frames by quadratic polynomials and then estimates displacement fields from the polynomial expansion coefficients. The Farneback optical flow method is embedded in the OpenCV library (Bradski, 2000), and it has been used for object tracking, segmentation and human action recognition, etc. (Anthwal & Ganotra, 2019; Chauhan & Krishan, 2013; Sevilla-Lara et al., 2018; Shantaiya et al., 2015). Recently, most deep learning models for video analysis use optical flows as one modality of input data (Simonyan & Zisserman, 2014; Xu et al., 2019a). However, traditional optical flow methods match pixels from one frame to the next one based on colour, which not only leads to erroneous results but also is time-consuming. These optical algorithms are relatively complicated and time-consuming, which is unsuitable for real-time applications. Tao et al. (2012) presented a so-called SimpleFlow optical flow algorithm with high computational performance. SimpleFlow only computes a sparse set of samples in regions with a uniform motion, and pixels are processed independently and only once. This property guarantees the effectiveness of the result and low computational complexity. Moreover, SimpleFlow can be easily implemented on parallel architectures such as multi-CPU (Central Processing Units) and GPU (Graphics Processing Units) to accelerate computational procedures. Figure 2-6 illustrates the two categories of optical flows by performing Farneback and SimpleFlow algorithms to process two frames, respectively. It can be seen qualitatively that the result of the SimpleFlow algorithm contains more approximation information that indicates more detailed motions, and the histograms of the two types of optical flows also suggest that the SimpleFlow method is more robust than Farneback.

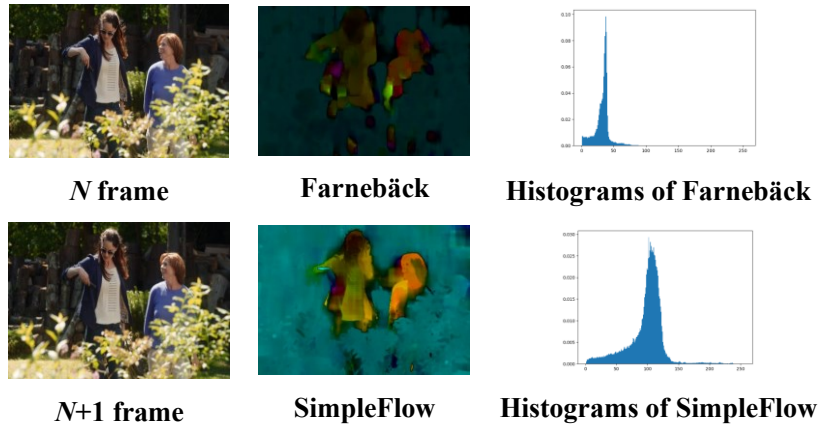


Figure 2-6. Visualization of the two flow algorithm results.

b) Deep learning-based optical flows

With the development of deep learning, these techniques are also migrated to support automatic optical flow generation. FlowNet, proposed by Dosovitskiy et al. (2015), is the first deep learning-based optical flow estimation that is constructed by CNN. However, the performance lags behind the traditional optical flow methods. Ilg et al. (2017) improved the flow accuracy by stacking several FlowNet modules and introducing a warping operation between intermediate optical flow and the second image, namely a few, FlowNet2. However, a large model costs a lot of computational resources and memory. To tackle this issue, Sun et al. (2018a) presented an effective CNN based optical flow estimator called PWC-Net by using cost volume, pyramidal processing and warping, which increases the accuracy but reduces the model size. Ranjan et al. (2018) presented a CNN based human optical flow model that extracts human motion directly from original video frames. They also introduced a dataset to train this deep learning model. Human optical flow is superior to generic flow methods. Figure 2-7 demonstrates the human optical flow dataset and the algorithm results on synthetic and real-world scenes (Ranjan et al., 2020).

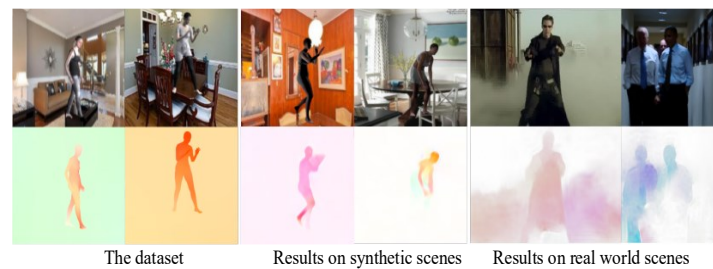


Figure 2-7. Visualization of human optical flow dataset and the results on both synthetic and scenes (Ranjan et al., 2020).

2.3.3 Trajectory Features

Trajectories based methods show reasonable results on several datasets. Messing et al. (2009) developed a Harris 3D and Kanade-Lucas-Tomasi (KLT) based model to track feature points and obtain trajectories from videos. Ju et al. (2009) developed a SIFT based tracker to obtain trajectories. Later, Sun et al. (2010) combined these two trackers to increase the density of trajectories. However, both KLT and SIFT trackers are insufficient to handle the frame boundaries and describe complex motion patterns.

To tackle this shortage, Wang et al. (2013) presented a dense trajectories (DT) model that densely samples feature points on each spatial scale and then tracks the points in the following frames with a preset length l . The trajectories (P_1, P_2, \dots, P_l) are obtained when the number of tracked frames is completed, where P_i indicates a feature point in the i -th frame. Aligned with the trajectories, four features are extracted, including trajectory shapes (TS), histogram of oriented gradients (HOG), histogram of optical flows (HOF), and motion boundary histogram (MBH). After that, the Bag-of-features (BOF) concept is applied for feature assembly. The DT model is more robust in handling complex motion patterns when compared with KLT and SIFT. Since its appearance, the DT model has been gaining popularity and being tested on various action datasets with significant improvements over the state-of-the-art. It has drawn wide attention and optimism (Jiang et al., 2017; Peng et al., 2014; Wang & Schmid, 2013). Wang and Schmid (2013) further improved their works (named iDT) by investigating Speeded Up Robust Features descriptor (SURF) and FV. Peng et al. (2014) proposed Stacked Fisher Vectors (SFV) with multi-layer nested FV encoding for human action recognition. Jiang et al. (2017) developed an action prediction method based on dense trajectories and dynamic image models, which is capable of predicting evolutionary trends of actions in videos.

Since its birth in 2012, iDT has become the baseline for performance evaluation in video event analysis. It remains a widely adopted benchmark even in the deep learning era. However, DT models lack the mechanism to distinguish dominant motions from secondary ones for differentiating human actions over separable frequency bands

and directions. This research explores the integration of wavelet techniques into the dense trajectory domain to gain the descriptive action patterns and better harness the advantages of the semantically more representative handcrafted video features.

2.4 Feature Representation

2.4.1 Bag of Features

Bag of Feature (BOF) was inspired by the Bag of Words (BOW), and it is often referred to as bag-of-visual-words (BOVW) in computer vision studies (Bolovinou et al., 2013). In this case, a feature of an image or a video frame is considered a “visual word”. The first stage of BOF implementation is to train a codebook. All low-level features extracted from training videos are clustered into N categories by the K-means clustering scheme. Such that each centre of a quantised area of a category becomes a visual word, and all visual words (cluster centres) construct the corresponding codebook. Thus, the length of a codebook is equal to the number of visual words in this codebook.

In the calculating histogram stage, the low-level features extracted from a video are represented as the histograms of visual words, denoted as:

$$C = (c_1, c_2, \dots, c_n), \quad 2-5$$

where c_i indicates the value of i -th visual word in the codebook, the value of c_i is normalised by the maximum-minimum functions:

$$n_i = \frac{c_i - \min(C)}{\max(C) - \min(C)}. \quad 2-6$$

Hence, a video event can be represented as the following histogram of visual words:

$$V = (n_1, n_2, \dots, n_N). \quad 2-7$$

BOF directly assigns a feature to one of the nearest visual words. This “hard” assignment is rigid and inaccurate. It is more flexible in assigning a feature to different visual word bins when the distances between the feature and these visual words can be “weighted”. Moreover, a feature may be assigned to other visual words when the scale of codebooks can be varied. This research further investigates this issue and develops a soft-assignment method to encode the visual words.

2.4.2 Fisher Vector

Let $X = \{x_i, i \in [1, T]\}$ be the series of low-level features extracted and formulated from videos. Fisher vector assumes the generation process of X can be modelled by a probability density function $p(u; \theta)$ with parameters θ , the X is described by the gradient vector:

$$G_\theta^x = \frac{1}{T} \nabla_\theta \log p(X; \theta) \quad 2-8$$

The length of the gradient vector is fixed, which only depends on the number of parameters (i.e., the dimensionality of θ), but not the actual number of features. The probability density function is widely used by models such as GMM: $p(u; \theta) = \sum w_i u_i(x)$, and $\theta = \{\pi_i, \mu_i, \sigma_i, i \in [1, K]\}$, where π_i , μ_i , and σ_i are the mixture weight, mean vector and diagonal GMM, respectively; K denotes the mixture number of GMM. Then the fisher vector is formulated as follows:

$$\begin{aligned} g_{u,k}^x &= \frac{1}{T \sqrt{\pi_k}} \sum_{t=1}^T \gamma_t(k) \left(\frac{X_t - \mu_k}{\sigma_k} \right) \\ g_{\sigma,k}^x &= \frac{1}{T \sqrt{2\pi_k}} \sum_{t=1}^T \gamma_t(k) \left[\frac{(X_t - \mu_k)^2}{\sigma_k^2} - 1 \right], \end{aligned} \quad 2-9$$

where $\gamma_t(k)$ indicates the weight of low-level feature x_i for the j -th Gaussian function, as shown in the following:

$$\gamma_i(k) = \frac{\pi_k \mu_k(x_i)}{\sum_{j=1}^K \pi_j \mu_j(x_i)}. \quad 2-10$$

where $\mu_k(x_i)$ is D -dimensional Gaussian distribution, then the fisher vector of the set of features is given by the concatenation of $g_{u,k}^x$ and $g_{\sigma,k}^x$, as shown in the following:

$$f_{Fisher} = [(g_{u,k}^x)^T, (g_{\sigma,k}^x)^T]^T, k \in [1, K]. \quad 2-11$$

Fisher vector encodes the average first and second-order differences between the features and the centres of a GMM, which can be considered a soft visual vocabulary demonstrating better performance than the bag of feature method for classification. To optimise the runtime performance of the design, the Principal Component Analysis (PCA) technique was first applied to reduce the low-level feature dimensionality. The number of Gaussians was set at $K=512$ to train and estimate the GMM. Therefore, a

single video event can be represented by a $2DK$ dimensional FV (see Equation 2-11) before a L2-normalization.

2.5 Action Classification

2.5.1 Support Vector Machine

The powerful mathematical foundation of Support Vector Machine (SVM) enables efficient methods for classification and regression (Chandra & Bedi, 2018). Mathematically, SVM constructs a hyper-plane in high dimensional space. A suitable separation can be obtained by searching the so-called support vector that defines the decision boundary and gives the largest distance to the points belonging to different classes (Smola & Schölkopf, 2004). SVM has drawn wide attention and applications in the classification task, such as most traditional human action recognition methods apply SVM as the classifier (Peng et al., 2014; Wang et al., 2013; Wang & Schmid, 2013). Even in the early stage of the deep learning era, SVM is still a good choice for classification when the feature points are extracted from the last convolutional layers of CNN models (Simonyan & Zisserman, 2014).

2.5.2 Artificial Neural Network

An artificial neural network (ANN) or multi-layer perceptron (MLP) is an important supervised learning algorithm which gains insight from biological neurons. According to Haykin (2009), ANN can perform a similar function in the human brain and produces a specific task through the multi-layer artificial neurons and activation functions. Trained by the backpropagation algorithm, ANN performs good accuracy on classification tasks (Abiodun et al., 2018). Recently, ANN has evolved into deep learning with complex multilayers and connections (Albawi et al., 2017).

2.6 Deep Learning Approaches

Unlike handcrafted features, deep learning models extract features automatically from the input data (e.g., images and videos). Of this “unsupervised” style, it has gained tremendous popularity in many application domains. For example, image classification

tasks have experienced almost a complete overhaul through varied forms of CNN implementations (He et al., 2016; Huang et al., 2017). Object detection and facial recognition have also achieved encouraging results (Jin et al., 2017; Wu et al., 2017). Recently, deep learning based human action recognition has seen major breakthroughs.

2.6.1 Deep Learning Techniques

a) CNN

CNN is a simple neural network module that is constructed by convolution operations to calculate feature maps. CNN has played a significant role in the history of deep learning (Goodfellow et al., 2016), and it is the first neural network to solve critical commercial applications. Lecun et al. (1998) proposed a CNN model named LeNet for document recognition. Inspired by this research, Krizhevsky et al. (2012) developed the AlexNet, which won the ImageNet image classification challenge. Later, the GoogleNet (Szegedy et al., 2015b), VGG (Simonyan & Zisserman, 2015), and ResNet (He et al., 2016) were proposed for better performance on image classification; the R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2017) and SSD (single shot multibox detector) (Liu et al., 2016) were proposed for object detection. All these network architectures are based on CNN and have gained increasing achievements in vision application.

b) RNN/LSTM

RNN is firstly proposed by Rumelhart et al. (1986) for processing sequential data by preserving a memory of its hidden states over time and maintaining a feedback loop among them, i.e., the current hidden state of RNN units will affect the subsequent states, such that it supports the sequential learning, i.e., learning connections between inputs and the corresponding previous states continuously. The traditional RNN unit is shown in Figure 2-8. It maps the sequences of input, hidden states, and outputs, as shown in the following:

$$\begin{aligned} h_t &= g(w_{xh}x_t + w_{hh}h_{t-1} + b_h) \\ y_t &= g(w_{hz}h_t + b_z) \end{aligned} \quad , \quad 2-12$$

where g is an activation function, such as the Hyperbolic Tangent (Tanh) function or rectified linear unit (ReLU), x_t is the input, h_t is the hidden state, y_t is the output at time t , and w indicates the weights. For a length L input sequence $[x_1, x_2, \dots, x_L]$ and setting $h_0=0$, the outputs are computed sequentially as $[(h_1, y_1), (h_2, y_2), \dots, (h_L, y_L)]$.

LSTM is the most important RNN implementation. As shown in Figure 2-9, LSTM updates for timestep t given inputs x_t , h_{t-1} , and c_{t-1} are shown in Equation 2-13 (Donahue et al., 2017). LSTM has performed good results on sequence-based tasks such as Natural Language Processing (NLP) applications (Guadarrama et al., 2013). It is also applied for video-based applications such as video description and human action recognition (Donahue et al., 2017; Varol et al., 2018).

$$\begin{aligned}
 i_t &= \sigma(w_{xi}x_t + w_{hi}h_{t-1} + b_i) \\
 f_t &= \sigma(w_{xf}x_t + w_{hf}h_{t-1} + b_f) \\
 o_t &= \sigma(w_{xo}x_t + w_{ho}h_{t-1} + b_o) \\
 g_t &= \phi(w_{xg}x_t + w_{hg}h_{t-1} + b_c) \\
 c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
 h_t &= o_t \circ \phi(c_t)
 \end{aligned}
 \tag{2-13}$$

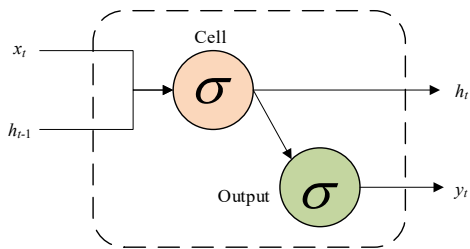


Figure 2-8. Visualization of RNN Unit.

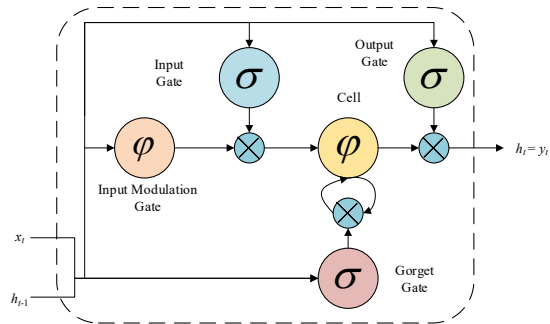


Figure 2-9. Visualization of LSTM Unit.

2.6.2 Long-term Recurrent Convolutional Networks

Donahue et al. (2017) applied RNN modules for temporal learning from a sequence of CNN features and developed a so-called long-term recurrent convolutional network (LRCN) which is a generic CNN-RNN framework for large-scale visual learning applications such as image description and video analysis. The architecture of LRCN is illustrated in Figure 2-10. It contains multiple stream deep networks, and each one has four components: visual input, visual feature extraction, sequence learning and predictions/classifications. The visual input component takes data into a visual feature

extraction component. This data can be the original video frames, optical flows, or both. The visual feature extraction is implemented by CNN, which has good capability to learn spatial features from still images and video frames. It can also learn temporal features from optical flows. The outputs of CNN are treated as the inputs for the subsequent RNN. The RNN can automatically discover appropriate sequential information, so it is the best choice for dealing with actions that have both the temporal model (atomic movements linked by time) and the sequential model (order information). However, RNN is not suitable for directly learning sequential features from high-dimensional data, such as original frames. Therefore, using the outputs of CNN as the inputs of RNN is a good choice for sequence modelling. In practice, the LSTM unit is used to implement the recurrent module due to LSTM enables them to remember their states over a long period by introducing forget gate units (Goodfellow et al., 2016). The final part is applied for classification or prediction depending on the applications.

In terms of human action recognition, one of the implementations based on LRCN is constructed as in Figure 2-11. It has two-stream CNNs that are fed by two continuous video frames. The outputs of CNNs are inputted into the LSTM model for sequence learning. Finally, the outputs of LSTM are treated as the inputs of a classifier that outputs an action label. LRCN only uses RGB video frames for spatial and temporal feature learning to speed up the computational progress. Nevertheless, the accuracy on UCF 101 is approximate 65.6%, which has yet to be improved. The primary reason is that the fine motion information is lost in the multiple layers of CNN feature maps; thus, the model fails to handle local human and object motions.

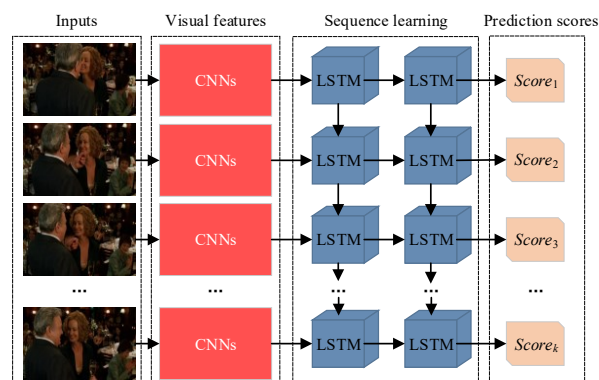


Figure 2-10. The generic architecture of LRCN.

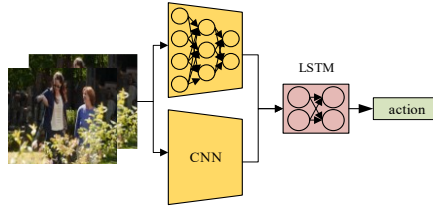


Figure 2-11. A specific instantiation of the LRCN model for human action recognition.

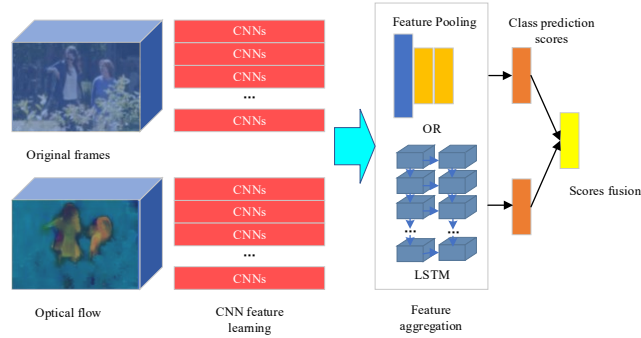


Figure 2-12. An overview of Ng's approach.

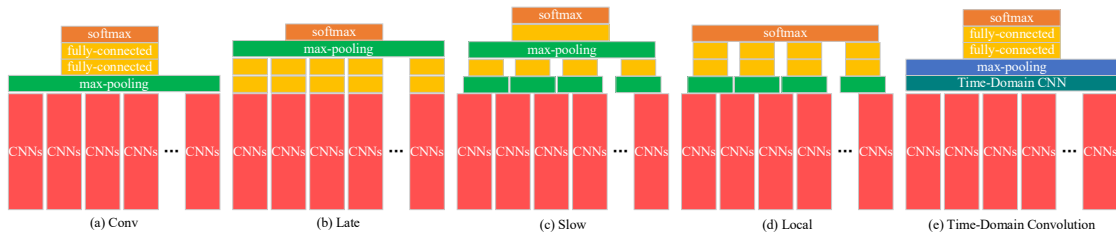


Figure 2-13. Five types of Feature Pooling Architectures: “CNNs” presents stacked CNNs. (Ng et al., 2015).

2.6.3 Long Time Periods-based Networks

To handle full-length temporal information, Ng et al. (2015) presented a long time period network architecture which contains two stages: firstly, the CNN modules are applied to learn spatial features from frames and optical flows; and then they proposed two feature aggregation approaches to model variable length videos with fixed size video-level feature vectors, namely, the feature-pooling and RNN. The overview of this model is shown in Figure 2-12. This approach firstly uses a typical CNN backbone such as VGG or GoogleNet to process each video frame, producing a feature vector from the fully connected layers and developing two temporal feature descriptors for the event representation and classification.

Feature Pooling. Five types of temporal feature pooling strategies have been developed by Ng et al. (2015), including Conv Pooling, Late Pooling, Slow Pooling,

Local Pooling and Time-Domain Convolution, as shown in Figure 2-13. The Conv Pooling applies the max-pooling operation over the whole outputs from the final CNN layers of the full-length videos. The Late Pooling adds two fully connected layers for each final CNN layer and then applies the max-pooling layer for entire frames. Slow Pooling is designed as a hierarchical framework using a two-stage pooling strategy, which is similar to Local Pooling employing a single pooling layer followed by two fully connected layers before a softmax layer. The Time-Domain Convolution model employs a CNN layer before the pooling layer, fed by all feature maps extracted by the final CNN layers from full-length frames.

LSTM based sequences learning. Ng et al. (2015) also tested different LSTM settings and found that the five stacked LSTM layers model shows the best performance. The LSTM architecture is illustrated in Figure 2-14, in which each LSTM layer contains 512 LSTM units. A softmax function is applied in the final LSTM layer to predict actions for each video frame.

To accelerate computational performance, this work only processes a single frame per second (FPS) (Ng et al., 2015), resulting in the loss of implicit motion information. Even though the optical flow is employed to compensate motion information explicitly, it still loses a great deal of valuable temporal information.

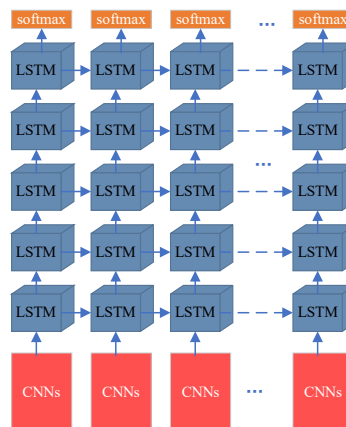


Figure 2-14. The five stacked LSTM layers architecture.

2.6.4 Long-term Temporal Convolutions

To model full temporal extent, Varol et al. (2018) proposed a so-called long-term temporal convolutions (LTC) model, as shown in Figure 2-15. It contains five 3D CNN

layers with 64, 128, 256, 256 and 256 filters, and $3 \times 3 \times 3$ filters implement each layer. Finally, three fully connected layers of sizes 2048, 2048 and the number of action categories are followed by the last CNN layer. 3D CNN has a high capability to handle long temporal features from video frames (Ji et al., 2013). With these advantages, LTC has archived good results on UCF 101 and HMDB51 action datasets with accuracies of 92.7% and 67.2%, respectively.

The implementation of the LTC model is straightforward but highly efficient. Nevertheless, a video is divided into t -frame clips to learn spatial-temporal features. As a result, the outputs of the LTC model contain the “inner” spatial-temporal information of the individual clips only, whereas the “outward” temporal information between different video clips is lost.

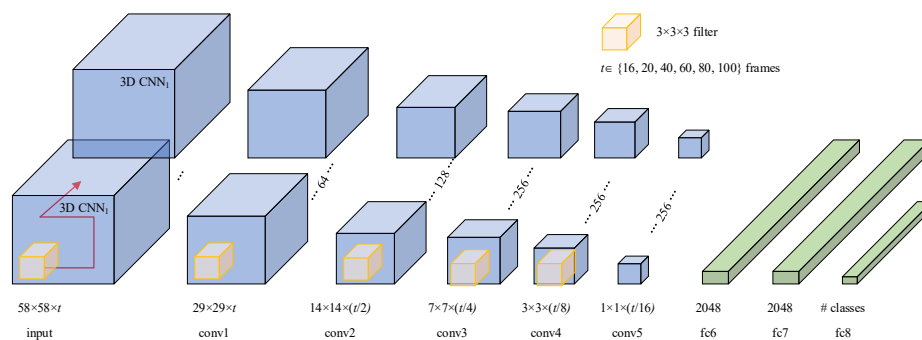


Figure 2-15. LTC-CNN based network architecture.

2.6.5 Two-stream Networks

Simonyan and Zisserman (2014) presented a so-called two-stream network to learn spatial and temporal features simultaneously, in which a spatial stream is applied to extract appearance information from video frames, while the movement information is learned from the stacked optical flows by the temporal stream, and the last result is obtained by averaging the two features at the last convolutional layer. The two-stream model has become a classical model for extracting video features, and various improved solutions have been proposed for performance improvement (Gammulle et al., 2017; Zhao & Snoek, 2019; Zhu et al., 2019). Ye et al. (2015) tested the important factors relating to two-stream CNN performance, including network architecture, learning

parameters, model fusion and final prediction methods. Recently, ResNet has been integrated to implement the two-stream network due to its outstanding ability in image feature extraction (He et al., 2016). Feichtenhofer et al. (2016) built a two-stream CNN and ResNet combination model by introducing residual connections. Carreira and Zisserman (2017) presented a two-stream inflated 3D CNN (I3D) model that is pre-trained on a large-scale video dataset (Kinetics). To tackle the disadvantage of lacking time-scale diversity in the temporal domain, Wan et al. (2020) developed a dual-stream convolutional network with the long-short-term spatiotemporal features (LSF CNN) which indicates a promising direction for consistently handling motion features in both spatial and temporal domains. These models have a high ability to encode both spatial and temporal information and achieved promising results on the human action recognition tasks. However, these models extract information separately from the spatial and temporal domains, and the joint key “nodes” of the two are ignored.

Currently, the mainstream two-stream networks require the optical flow as one data modality for temporal pattern modelling, whereas computing optical flow is time-consuming and inefficient (Ilg et al., 2017). The stage of computing optical flow occupies approximately 90% of the running time at both training and testing stages, hence it limits its application and real-time processing. Zhang et al. (2016) proposed a motion vector that is an optical-like feature computed directly from compressed videos. Later, in order to obtain the motion modality efficiently, Shou et al. (2019) proposed a lightweight Generator network to capture fine motion details and reduce noises in the motion vector; thus, it achieved a better discriminative motion representation than the classic optical flow methods. These optical flow generators have reduced the optical flow computational cost with a small amount of noise-induced and some extra operations. It is anticipated that the more effective mechanism is to directly extract optical flow-liked features with minor information loss in an online mode.

In general, two-stream networks show reasonable performance on many video-based applications. However, it still faces two main drawbacks: 1) the spatial and temporal streams are trained separately, and the final predictions are combined by simply averaging the corresponding outputs of two streams into the classification scores.

These two-stream CNN models have inherent weaknesses in learning spatial-temporal relationships; 2) the contemporary model has limited capacity in the temporal domain due to the spatial stream being fixed to operate on one video frame per cycle, while the temporal stream can process up to 10 continuous frames with optical flows. These two shortcomings have led to the failure of recognition of long-time sequence behaviours due to information loss. Thus, Spatiotemporal Residual Networks (ST-ResNets) (Feichtenhofer et al., 2016) are designed by building residual connections between the two streams to fuse spatial and temporal information, so that better learning results can be achieved by giving iterative interactions between the spatial and temporal streams. However, the interactions only operate on the pixels that are of low-level feature categories. It is widely accepted that features in different network layers are of different semantic levels, e.g., the bottom layer generates low-level visual features such as contours and edges. In contrast, features in the top layer carry semantic significance. In this research, different fusion approaches have been investigated to identify suitable information extraction techniques for the so-called concurrent two-stream CNNs. As a result, an innovative fusion approach has been devised to integrate information at all network layers between motion and visual streams through building a feature fusion block – the spatial-temporal fusion blocks (STFB). Hence, it enables the extraction of multiple level features through interactions covering the entire low, middle, and high feature spectrum in the spatial and temporal signal domain.

2.6.6 3D CNN based Models

3D CNN directly processes 3D convolution operations on the original 3D video volumes to learn spatial information in the RGB frames and the pixel movements along the time axis, hence it naturally supports spatial-temporal feature learning from videos. Ji et al. (2013) applied 3D convolution kernels to extract spatial-temporal features from videos for human action recognition. Tran et al. (2015) presented a C3D (Convolutional 3D) model for obtaining generic spatial-temporal features by applying $3 \times 3 \times 3$ convolutional kernels in all layers. Sun et al. (2015) proposed a method to decompose the 3D convolution into a 2D space convolution followed by a 1D temporal convolution

for learning spatial-temporal relationships. At the same time, Qiu et al. (2017) proposed a so-called Pseudo-3D Residual ResNets (P3D) for training a very deep model with a relatively cheap computational cost and memory demand. Later, Tran et al. (2018) explored the effects of 2D and 3D CNN modules on action recognition. The result suggested the accuracy advantages of 3D CNN over 2D CNN within the residual learning. Then, they further factorise the 3D CNN into separate spatial and temporal components, named the “R(2+1)D” block, which achieves superior performance on action datasets. Recent work of X3D proposed by Feichtenhofer (2020) expands a tiny 2D image classification architecture to 3D video recognition accosting space, time, width and depth dimensions. X3D achieved competitive performance on action datasets while keeping low computational cost, which is suitable for the “mobile-regime” action recognition. 3D CNN shows reasonable performance on video analysis. However, it still has a notable drawback that harms performance improvement, e.g., they tend to leverage contextual information such as objects and scenes. At the same time, human movement is weakly abstracted (Weinzaepfel & Rogez, 2021), hampering accurate motion information extractions. Furthermore, 3D CNN requires a lot of computational resources, limiting the usage of real-time applications and embedded systems.

2.6.7 Learning Temporal Features

Most CNN and 3D CNN based models only track a short period for the temporal features in video clips, e.g., 16 frames, which leads to difficulty when dealing with “longer” event sequences. Another significant drawback of the current CNN implementation is its limitation in dealing with sequential information such as plots in movies. Li et al. (2017) introduced a LSTM-based model for handling spatial-temporal features. Shortly after, Majd and Safabakhsh (2020) presented a correlational convolutional LSTM (C^2 LSTM) to handle both the spatial and motion structure of surveillance video data. Wang et al. (2015) presented a so-call trajectory-pooled deep-convolutional descriptor (TDD) that embeds the features from both handcrafted and deep-learning models. Motivated by TDD, Lu et al. (2017) developed a multi-scale trajectory-pooled 3D convolutional descriptor (MTC3D) by combining dense

trajectories and 3D CNN. TDD and MTC3D are capable of automated learning of temporal features from motion trajectories. However, these models have been trained in a clip-level or single-frame-level loss, which has failed to capture long-term temporal information. To alleviate this major problem, this research proposes a long-short-term learning strategy training a deep neural networks (DNN) model on an entire video and updating the model weights in the video-level gradients. The details are discussed in Chapter 5.3.3.

2.7 Skeleton based Approaches

2.7.1 Pose Estimation

A human pose defines the body joint positions in an image in the form of 2D or 3D coordinates, which can be easily captured by the RGB-D depth sensors. For instance, Shahroudy et al. (2016) captured 56,880 RGB-D video clips by using the Microsoft Kinect v2 depth sensor and then proposed the NTU RGB+D dataset that has 60 classes of actions (NTU-60), including daily, medical, and mutual actions coming from 40 different human subjects. Later, this dataset was extended by adding other 60 classes and additional 57,600 videos, hence introducing NTU RGB+D 120 (NTU-120) dataset (Liu et al., 2020a). The large-scale datasets enable the training of sophisticated DNN models for human action recognition and activity understanding. However, to the best of my knowledge, the hardware-based pose estimation has never been deployed to real-world human action recognition systems in public areas since it requires special sensors along with surveillance cameras. The more natural way is extracting human poses from RGB frames directly by pose estimation methods (Cao et al., 2021; Kocabas et al., 2020; Rogez et al., 2020; Sun et al., 2019; Wang et al., 2021a).

The pose estimation methods require not only identifying the human joints but must building the connection between joints for each person in multi-person settings. Pishchulin et al. (2016) proposed the DeepCut model which firstly detects all body parts and builds pairwise connections between the detections, then the Integer Linear Program is applied for body part clustering, and each clustered body parts generate a pose belonging to one person. To solve the time-consuming process of the Integer

Linear Program, an advanced pose estimation named OpenPose has been developed (Cao et al., 2021; Cao et al., 2017). OpenPose simultaneously learns the heatmap for body part localisation and the Part Affinity Fields (PAF) vector for associating body parts with distinct persons to achieve better accuracy while reducing the computational cost (Cao et al., 2021). However, the bottom-up approaches, which predict all body parts and then group the parts to each person, are still complex in computation, while the accuracy has yet to be improved when facing the multi-person setting.

The top-down approaches, on the other hand, detect the human boxes firstly and then estimate joints for each person separately, and the advantages of object detection techniques can be applied in the first stage to generate accurate human bounding boxes (Fang et al., 2017; He et al., 2017). More significant, the two stages can be combined in an end-to-end manner to optimise the two stages simultaneously. He et al. (2017) proposed Mask R-CNN which is a general-purpose framework extended from Faster R-CNN (Girshick, 2015) for multiple vision tasks. By adding an extra mask branch which predicts the heatmaps for human joints localisation, it also serves as a strong baseline for human pose estimation (He et al., 2017). To improve the robustness of handling inaccurate and redundant human bounding boxes, Fang et al. (2017) proposed a so-called regional multi-person pose estimation (RMPE) model, which achieved good performance on multi-person benchmarks. Nevertheless, most deep learning methods follow a high-to-low feature representation and recover the high-resolution heatmaps from low-resolution representations, which may reduce the preciseness of spatial information due to the high-resolution representations are important for pose estimation. To tackle this issue, Sun et al. (2019) proposed High-Resolution Network (HRNet) to maintain the high-resolution representations during the entire process. As a result, the predicted pose heatmaps are spatially more precise and more accurate. This work is then extended to position-sensitive vision tasks and achieved good performances on pose estimation, object detection, and semantic segmentation (Wang et al., 2021a).

Besides the 2D human pose estimation, 3D approaches can also predict 3D poses; e.g., Rogez et al. (2020) proposed a so-called Localization Classification Regression Network (LCR-Net) pose estimation which can predict 2D and 3D poses of multiple

persons concurrently. Kocabas et al. (2020) presented the Video Inference for Body Pose and Shape Estimation (VIBE) to exploit temporal information for estimating 3D pose motions of the body from videos by developing improved temporal pose and shape regression networks.

2.7.2 Skeleton for Action Recognition

Most human action recognition methods employ appearance and optical flow modalities (Jiang et al., 2021; Simonyan & Zisserman, 2014; Tran et al., 2015), while the modelling of body skeletons has received less attention. Human body skeletons are natural body language representations which have a strong capacity against context change and scene variation. Early approaches for skeleton-based action recognition rely on handcrafted formulations. These could be relative position joints, rotations, and translations between body parts. For instance, a “fall” action can be defined from the condition that the angle between the head and hip is small. Although it is very easy, these handcrafted methods are less robust when facing complex actions. The recent success of deep learning has led to automatic learning event representation from human poses. In general, human poses in a video can be presented by a sequence of coordinates and then learning the action pattern by recurrent structures (Shahroudy et al., 2016). RNN can learn temporal information from the sequences, but the local spatial information of joint locations is ignored. Liu et al. (2017) presented an image-based model for spatial-temporal skeleton representation. In this method, the sequence joints are transformed into pseudo-image series, and then a CNN-based model is applied to extract features from the pseudo-images along with RGB frames. However, the context-biased problem is still existed due to the image-based approaches replay on appearance features.

To understand actions from skeletons, the spatial information which represents joint locations and the temporal describing the movements of joints are both important, and the relationships between near joints are also more significant than the distant joints. Moreover, an action normally performed by several body part movements, e.g., the “walk” action contains “hand swing”, “foot lifting”, and “foot setting down”. Therefore,

a better action recognition model should learn local (body parts) spatial-temporal features from human pose sequences, which is similar to the small (3×3) convolutional kernels used in the image classification tasks (Simonyan & Zisserman, 2015). Based on this consideration, Yan et al. (2018) constructed the skeletons as the form of spatial-temporal graphs, where the nodes correspond to the human joints, and the spatial edges conform to the natural connectivity of joints, while the temporal edges connect the same joints across continues frames. Then a so-called Spatial Temporal Graph Convolutional Networks (ST-GCN) model is designed to handle the graph-based skeletons. Graph Convolutional Networks (GCN)-based methods have drawn wide attention since it was first proposed because of their notable performance on human action recognition. To incorporate human joint and bone information, Shi et al. (2019) constructed the skeleton data as a directed acyclic graph (DAG), and the joint, bone and their relationships are extracted by the specific-designed directed graph neural network (DGNN), hence improving the performance of action recognition. Liu et al. (2020b) presented a MS-G3D model that can remove redundant dependencies between node features by applying the multi-scale aggregation scheme, and it can directly learn cross-spacetime joint dependencies by using the unified spatial-temporal graph convolution (G3D) operators. Although the good accuracy, the computational complexity of GCN-based approaches is extremely heavy, e.g., the ST-GCN model costs 16.2 GFLOPs for recognising a video clip, while the complexity of the DAG-based GCN model is approximate 100 GFLOPs (Cheng et al., 2020). The inflexibility of both spatial and temporal graphs is another critical issue. Cheng et al. (2020) proposed the lightweight shift graph convolutional network (Shift-GCN) to cope with these drawbacks, which achieves remarkable improvements in both accuracy and computational cost. However, the GCN-based methods are still limited in the aspects of robustness, interoperability, and scalability. This research directly learns the action embedding from 3D heatmap volumes that implicitly contain human pose representations instead of using explicit joint coordinates by the advanced 3D CNN model, which is simpler, effective, efficient, and robust.

2.8 Model Inference on Edge Computing

Along with the rapid developments in deep learning and edge computing technologies, deploying models on mobile devices is a trend for modern applications. However, the resource-constrained and heterogeneous edge devices fail to cope with complicated model inference. Edge computing (Shi et al., 2016) and edge intelligence (Deng et al., 2020), although in the early stage, are pushing a brand-new computation and AI paradigm, which has the potential to tackle the cues of hardware and bandwidth cost-saving, real-time response, and data privacy and security. There exist several challenges and opportunities for further investigation (Shi et al., 2016), and carrying out AI to edge computing (namely a few, AI on edge) has various brand-new challenges due to the energy and cost of an edge device are always limited for processing such large volumes of data by a complicated model, such as the platform-independent model (PIM) design, quantized computation (Krishnamoorthi, 2018; Nagel et al., 2021), computation graph optimization, AI hardware design, and software-defined hardware (SDH), which have drawn wide attention of research. Edge intelligence is considered to be one of the key absent components in 5G networks, and it will be an essential factor for future 6G networks (Gupta et al., 2021b; Peltonen et al., 2020). It is envisioned that there is a transition from IoT to the Internet of Intelligent Things, and to the Intelligent Internet of Intelligent Things for the future of 6G Intelligent edges.

2.9 Datasets

2.9.1 Traditional Datasets

Lots of action datasets were introduced for training and evaluating human action recognition algorithms. The most popular ones are the KTH, Weizmann, Hollywood2, UCF and HMDB51, UT-Interaction datasets, etc. Sample videos of these datasets are shown in Figure 2-16.

- The **KTH** (Schuldt et al., 2004) dataset has six classes of actions: running, walking, boxing, jogging, waving, and clapping. Each action occurred in four scenarios: indoors and outdoors with various clothes and scales.

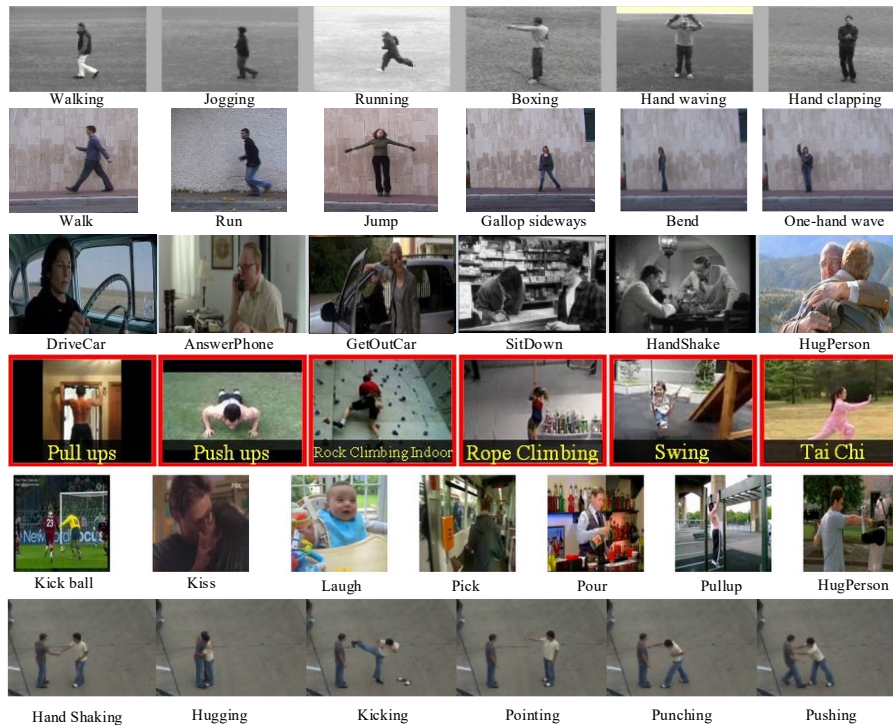


Figure 2-16 Sample frames from the action recognition datasets. From top to bottom: KTH, Weizmann, Hollywood2, UCF 101, HMDB51 and UT-Interaction.

- The **Weizmann** (Gorelick et al., 2007) dataset has ten actions with static background: walk, jump, bend, run, two-hands wave, gallop sideways, jump in place, one-hand wave, skip, and jumping jack.
- The **Hollywood2** (Marszalek et al., 2009) dataset was collected from 69 different movies, and it contains 12 action types: eating, running, sitting down, standing up, kissing, hugging, handshaking, fighting, driving a car, getting out of a car, and answering the phone. These videos have severe camera motions of the special (and old movie) effects of the scenes.
- The **UCF 11** dataset is an annotated version of YouTube clip collections (Liu et al., 2009). It includes 11 individual actions, namely, basketball shooting, cycling, diving, golf swinging, horse riding, football juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking, and walking with a dog. **UCF 50** (Reddy & Shah, 2013) is an extension of UCF 11 that contains 50 action categories collected from YouTube; **UCF 101** (Soomro et al., 2012) is an extension of UCF 11, having 13,320 videos from 101 action categories.

- The **HMDB51** (Kuehne et al., 2011) dataset has been collected from YouTube videos and movies, and there are 51 action classes.
- The **UT-Interaction** (Ryoo & Aggarwal, 2010) contains six types of human-human interactions: shake hands, point, hug, push, kick and punch. The videos are divided into two sets of different environment settings.

The KTH and Weizmann action datasets are relatively simple since the background is static and homogeneous, whereas the Hollywood2, UCF and HMDB51 datasets are complex in action types and background, and these datasets can be considered real-life videos.

2.9.2 Modern Datasets

Deep learning requires a huge amount of training video data, while the above datasets are not so large as the ImageNet dataset that is used on image classification. Therefore, large-scale datasets have been introduced in recent years, e.g., **Kinetics** is an action dataset of up to 306,245 videos with 400 action categories (Kinetics-400) (Kay et al., 2017), which is extended to larger datasets covering 400/600/700 human action classes (Carreira et al., 2018; Smaira et al., 2020). All videos come from YouTube, including individual actions, human-object interactions, and human-human interactions with various backgrounds.

Mimetics (Weinzaepfel & Rogez, 2021) is a specially collected dataset for out-of-context human action recognition because backgrounds and objects are absent in most videos. The Mimetics dataset contains 50 action categories and 713 video clips of mimed human actions. This dataset is only used for testing purposes due to its small scale. **Skeleton-Mimetics**, proposed by Gupta et al. (2021a), is another dataset for evaluating out-of-context action recognition.

NTU RGB+D (Shahroudy et al., 2016) action recognition dataset has 60 action categories (NTU-60) with 56,880 samples. Each dataset is captured by three Microsoft Kinect V2 cameras to obtain the RGB videos, depth maps, IR videos and skeletal data concurrently. Later, this dataset was extended by adding additional 57,600 video

samples of other 60 action categories, hence introducing the **NTU RGB+D 120** (NTU-120) dataset (Liu et al., 2020a).

2.10 Summary

In this chapter, a comprehensive literature review for a full pipeline of human action understanding is given. The traditional techniques are firstly introduced, including digital image processing, feature engineering and machine learning-based classification. The mainstream techniques of deep learning are reviewed, and the skeleton-based approaches are surveyed. Then the trend of edge intelligence is introduced. Finally, as one of the key elements of research, a taxonomy of action datasets is summarised.

CHAPTER 3 Feature Engineering for Video Analysis

3.1 Introduction

Among handcrafted features based human action recognition approaches, trajectories-based methods show better performance than others due to it guarantees not only coverage of dense interest points but temporal tracks as well. The dense trajectories method and its improved model (iDT) offer accurate recording of motions over time that is rich in dynamic information (Wang et al., 2013; Wang & Schmid, 2013). Since its appearance, the DT model has been gaining popularity and being tested on various action datasets with significant improvements over the state-of-the-art. It has drawn wide attention and optimism, and it has become the mainstream of handcrafted methods and is still important even in the deep learning era. However, DT models lack the mechanism to distinguish dominant motions from secondary ones over separable frequency bands and directions. To take advantage of semantical meaningful and “handcrafted” video features through feature engineering, this research integrates the DWT technique into the DT model for gaining more descriptive human action features. Another drawback is that the BoF method in DT encodes the low-level features as an unordered set, causing a large loss of spatial and temporal information. To tackle this problem, Bolvinou et al. (2013) presented the Bag of Spatio-Visual Words (BoSVW) to encode ordered spatial information for scene classification. Later, Zhao et al. (2014b) further improved this model by combining multiscale features, and it gained better performance on scene classification. BoSVW significantly improved the BoF encoder by integrating spatial context. Inspired by these achievements, this research explores a so-called bag-of-temporal-features (BoTF) technique to encode temporal information, i.e., it can encapsulate the ordered motion information of action in a video clip. This chapter introduces the methodology and theoretical model involved in the handcrafted feature extraction, event representation, and action classification. Then, the prototype modules and experiments are explained, as well as the discussion of the results.

3.2 Overview System Design

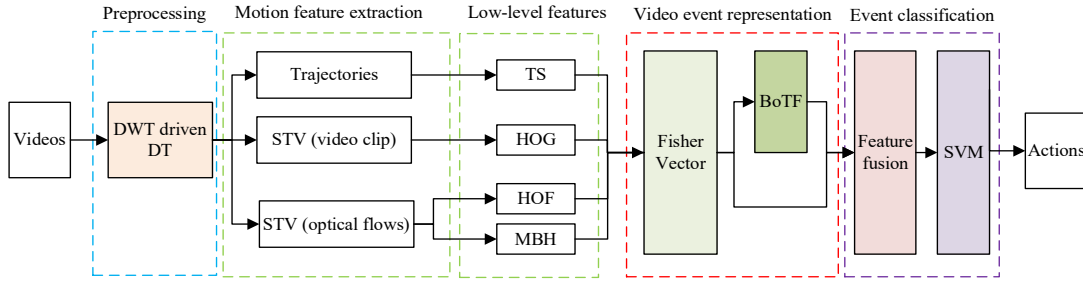


Figure 3-1. The handcrafted feature processing and representations based pipeline of the human action recognition model. It contains four stages. The raw pixel data are pre-processed by DWT and DT. After getting the low-level features by the motion feature extractions from training videos, Fisher Vector and BoTF schemes are applied to generate the codebook. At the end of the pipeline, SVM is applied for action recognition.

The processing pipeline of the handcrafted human action recognition model is shown in Figure 3-1, in which an input video is pre-processed for feature point extraction and tracked by the DWT-enabled DT model. The outputs are a series of low-level handcrafted features describing the trajectory patterns inherited from the STV data. Then, the handcrafted features are encoded into Fisher Vector and annotated by the proposed BoTF representation scheme. Finally, all video features are fused into a holistic video event representation scheme. It will then be classified by a SVM classifier for action recognition. The sections below explain the relevant techniques in detail.

3.3 DWT-based Decomposition

Traditional DT-based approaches extract feature points and then track them in video frames, which lacks detail and interpretable information on the separable frequency and movement orientation. Wavelet transform has the ability to record the coarse-to-fine presentation of spatial features. It has been demonstrated that DWT models can not only dissect an image in the form of multi-resolution representations but also extract textural features representing motion characteristics, hence contributing to semantic feature representation such as the BoW models (Zhao et al., 2014a). Inspired by the pilot work, the proposed technique decomposes video frames into different frequencies and orientations of multiple scales by applying the DWT filter. In practice, the lifting scheme is applied, and the Daubechies 4 wavelet is chosen to

compute wavelet coefficients. Note that other types of mother wavelets can also be used in this stage. As shown in Figure 3-2, the single level 2D DWT algorithm is applied to decompose a video frame into A , H , V and D components, where A is the approximation coefficients and H , V , and D donate detailed coefficients along horizontal, vertical and diagonal orientations, respectively. Figure 3-3 demonstrates a sample of a video frame coming from the UCF dataset and the corresponding DWT transform result. Compared with the original video frame, these four components are smaller in total size, and A contains information on the overall context. In contrast, H , V and D possess dominant movement information along varied orientations. Hence this approach enables a more effective feature extraction and tracking model.

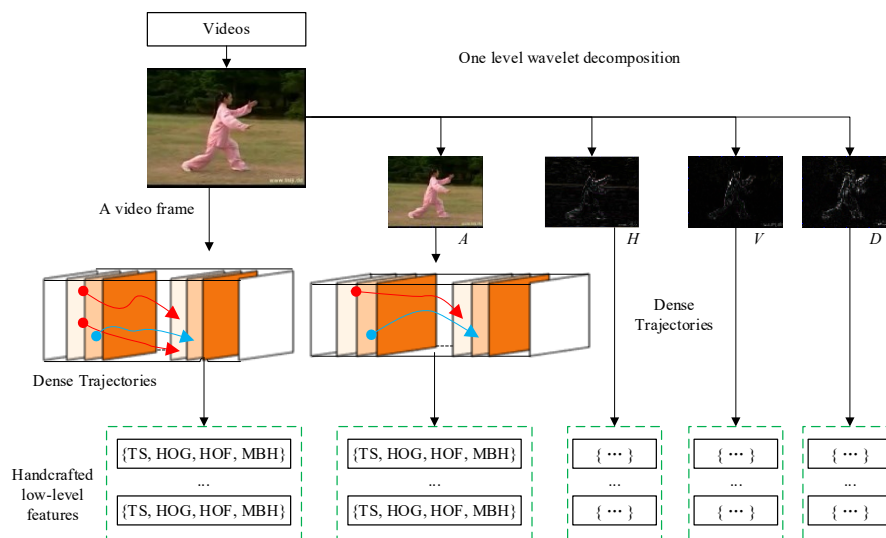


Figure 3-2. The processing steps of DWT-driven DT-based feature extractor. This model decomposes the original video frames into four coefficients. Along with the original frame, the DT method is applied to generate the trajectories (red and blue curves) and the low-level features.

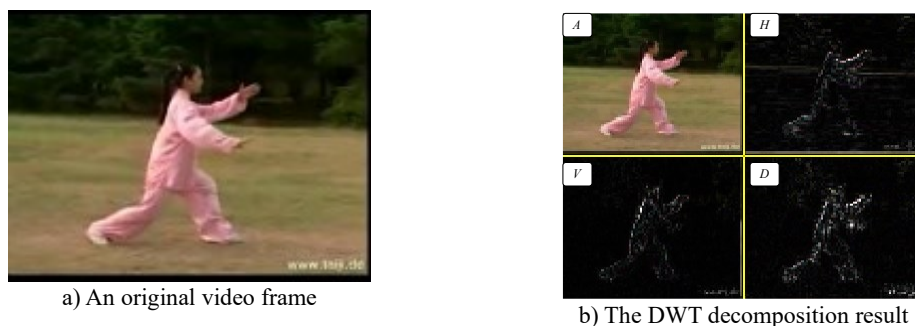


Figure 3-3. A demonstration of DWT pre-processing for a video frame coming from the UCF 101 action dataset. (a) illustrates a video frame from a “TaiChi” action video clip, while (b) shows the corresponding DWT coefficients.

3.4 Motion Feature Extraction

3.4.1 Dense Trajectory Formation

This research samples feature points densely on a grid of 5×5 for the input frames. In this process, the first spatial scale data is the input frame itself, and its spatial scale increases by a factor of $1/\sqrt{2}$. To reduce the amount of trivial and redundant feature points in homogeneous areas, a threshold T is deployed on the eigenvalues for each scale as shown in the following equation:

$$T = k \times \max_{i \in I} \min(\lambda_i^1, \lambda_i^2), \quad 3-1$$

where $(\lambda_i^1, \lambda_i^2)$ are the eigenvalues of i -th point in the spatial scale data I and its corresponding DWT coefficients. The value of k is taken as 0.001 for A , H and V of the original spatial scale data, while k is set as 0.01 for D . Dense sampling across all spatial scales ensures the comprehensiveness of feature points extracted and their motion potentials. For example, Figure 3-4 demonstrates the feature points extracted from the original (first) spatial scale, while Figure 3-5 illustrates the feature points extracted from the corresponding downward scales.



Figure 3-4. Feature points extracted from an original spatial scale.

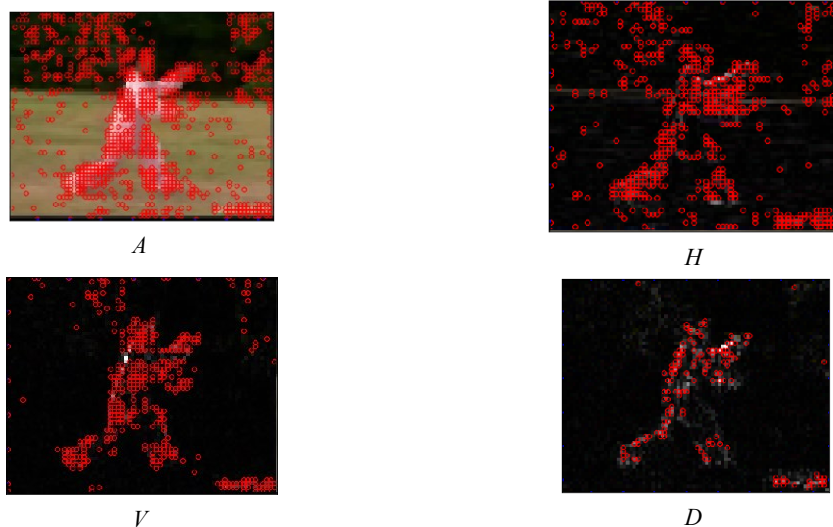


Figure 3-5. Feature points extracted from DWT coefficients.

Feature points from continuous input frames are then batch processed and tracked on each spatial scale respectively, before median filtering is performed on the dense optical flow fields m_t . The feature point tracking strategy is shown as the following:

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * m_t), \quad 3-2$$

where P_{t+1} is a tracked point in the consecutive frames, M is a median filtering kernel with the size of 3×3 , and (x_t, y_t) indicates a feature point in the t -th frame, and m_t is the dense optical flow.

The length of a typical action tracked is set at 15 frames (roughly two-thirds of a second) based on human behavioural studies (Wang et al., 2013). Once a tracked action is completed, a trajectory will be obtained in the form of $(P_t, P_{t+1}, P_{t+2}, \dots, P_{t+14})$. For storing feature trajectories, this research has devised a STV structure for encapsulating motions denoted by tracked features from all 15 video frames, as shown in Figure 3-6. Furthermore, this research also encapsulates the corresponding optical flows for later feature descriptions. The design ensures a compact and comprehensive representation of motion and context information inherited from a video event (human action) under study.

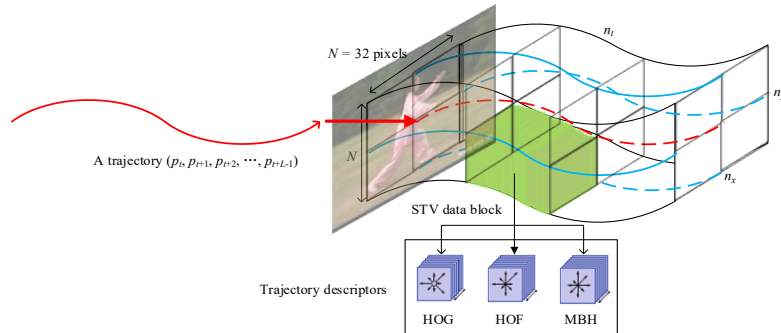


Figure 3-6. The encapsulated STV block for storing feature trajectories. The left red curve is a trajectory that is constructed by 15 tracked points.

3.4.2 Low-level Feature Extraction

Once the STV formatted video clip and the corresponding optical flows are obtained, four handcrafted motion and contextual feature descriptors are formulated, i.e., Trajectory Shapes, three categories of 2D histogram descriptors and their 3D counterparts.

a) Trajectory Shapes (TS) Descriptor

TS is denoted by a vector $(\Delta P_t, \Delta P_{t+1}, \Delta P_{t+2}, \dots, \Delta P_{t+14})$ based on a trajectory $(P_t, P_{t+1}, P_{t+2}, \dots, P_{t+14})$, in which $\Delta P_i = (\Delta x_i, \Delta y_i) = P_{i+1} - P_i$. TS records the normalised derivative of the trajectory tendency that can be calculated as the following:

$$TS = \frac{(\Delta P_t, \Delta P_{t+1}, \Delta P_{t+2}, \dots, \Delta P_{t+14})}{\sum_{i=t}^{t+14} \|\Delta P_i\|}, \quad 3-3$$

where $\|x\|$ is the L2-norm method. TS calculation is rooted in the tracked point coordinates, reflecting the shape information of a trajectory representing movements at each spatial scale and orientation. As the trajectory length is fixed at 15 frames and each point contains 2-dimensional coordinates, a single TS descriptor is a 30-component vector.

b) 2D Appearance and Motion Descriptors

To handle dynamic structures in the video clip, this research computes the histogram descriptors to encode appearance and motion information. As shown in Figure 3-6, given a STV cuboid with the size of $N \times N$ pixels and L frames, it is divided into a set of spatial-temporal cells with $n_x \times n_y \times n_t$ in size, where $n_x = n_y = 2$, and $n_t = 3$, and the green cell is one of the subdivided cells. The research computes histogram descriptors in each cell and then merges all descriptors as the final descriptors.

Histogram of Oriented Gradients (HOG) encodes static appearance information from video frames, and it especially focuses on the structure and shape information of objects. Followed the HOG computation method developed by Laptev et al. (2008), his research computes the HOG of each cuboid and sets eight quantization bins for gradient weighting. Then, these histograms in the grid are normalised by the L2 norm and concatenated into the final HOG descriptor vectors. The HOG descriptor outputs a 96-component ($2 \times 2 \times 3 \times 8$) vector.

Histogram of Optical Flow (HOF) formulates local motions from optical flows; its computation method is the same as HOG, except that the input data is replaced by the extracted dense optical flow. Optical flow is a significant feature of video processing. It tracks the motion information between two sequential frames, such that the HOF can encode the movements efficiently. The dense optical flows have already

been computed in the tracking stage, so the feature descriptor stage can reuse the optical flows and compute HOF directly. This research computes the HOF of each cuboid, and the number of quantization bins has been increased to nine to accommodate the zero bin. The HOF descriptor outputs a 108-component ($2 \times 2 \times 3 \times 9$) vector.

Motion Boundary Histogram (MBH) is proposed to correct the camera motion that often occurs in realistic videos. Optical flow estimates the global motion between two frames, including foreground and background motions. The foreground motion is normally captured from human and object movements, which is significant for recognising human actions. On the other hand, the background motion is caused by the camera motion, such as zooming, tilting, and rotation. In addition to that, the tracking shot is typically used in films. It will cause side effects if the camera motion is encoded in the foreground motion. Noted that, in many cases, the movements caused by camera motion are varied smoothly and regularly. Based on this oversedation, Dalal et al. (2006) presented the motion boundary coding to resist dynamic backgrounds, in which local constant camera motions are removed while preserving human and object motions through computing derivatives of optical flows, as shown in follows:

$$\begin{cases} w'_x = dw_t | (x_t, y_t) dx = \frac{\partial w_t | (x_t, y_t)}{\partial x} \\ w'_y = dw_t | (x_t, y_t) dy = \frac{\partial w_t | (x_t, y_t)}{\partial y} \end{cases} \quad 3-4$$

where w'_x and w'_y are the horizontal and vertical motion boundaries, respectively.

Aligning a trajectory, this research stacks the motion boundaries along the x- and y-axis for all continuous optical flows and then compute the histograms for each stacked motion boundary. The computation process is the same as HOG, and the number of quantization bins is set to eight, hence generating two histograms along X (MBH_x) and Y (MBH_y) directions with the size of ($2 \times 2 \times 3 \times 8$), and the final MBH descriptor (192-component ($2 \times 2 \times 3 \times 8 \times 2$) vector) is a concatenation of these two histograms.

c) Histogram of 3D Gradient Orientations

The 2D histogram descriptors come from the concepts of visual recognition in static images and are extended to video sequences by integrating normalisation across

the video frames. However, many 2D descriptors have derived their 3D counterparts, e.g., Klaeser et al. (2008) proposed HOG3D, which generalizes the HOG concepts to 3D, whose overview is illustrated in Figure 3-7 (Klaeser et al., 2008). Based on this idea, the HOG, HOF, and MBH can be extended followed the same operations. For a given STV cuboid with the size of $N \times N$ pixels and L frames, it is divided into a set of spatial-temporal cells with $n_x \times n_y \times n_t$ in size, which is the same as the 2D programme. The 3D gradient in each cell is then computed through a fast computation before it is quantized by using regular polyhedrons. Afterwards, the final 3D descriptor is a concatenation of all cell histograms. This research does not furth divide the cell into blocks because the frame is only 32×32 in size, which is very small; hence, the cell histogram is directly obtained without summing up all blocks. This research applies a so-called Fast HOG3D algorithm proposed by Li et al. (2014) to compute 3D histograms due to it is more compact and computational effect than the classical HOG3D algorithm. This research computes the 3D histograms by using the same parameters, and the bin number is set to 26, hence the HOG3D and HOF3D descriptors are 312-component ($2 \times 2 \times 3 \times 26$) vectors and the final MBH3D descriptor is a 624-component ($2 \times 2 \times 3 \times 26 \times 2$) vector.

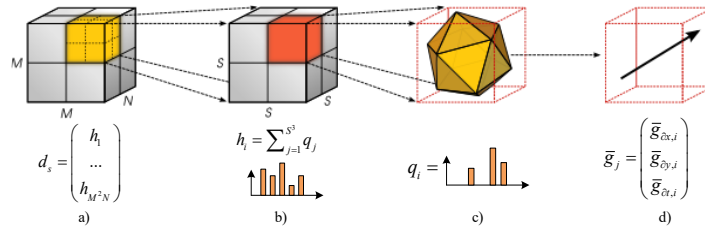


Figure 3-7. The computation progress of the HOG3D descriptor. a) a STV cuboid is subdivided into a grid of cells, and each one can be furth subdivided into blocks; b) the 3D gradient in each block is computed and summed up to the cell histogram; c) a regular polyhedrons-based quantization method is performed on each gradient orientation; d) the gradient is obtained from the whole videos.

3.5 Video Event Representation

3.5.1 Spatial-temporal Bag of Features

The extracted spatial-temporal features are used as input features for high-level video event representation. This research applies the BOF approach to encode the

trajectory shape, HOG, HOF, MBHx, MBHy, and their 3D counterparts separately. The first step is visual vocabulary generation and codebook creation. In this method, each trajectory generates a feature x with the length of L . Supposing each video clip contains M trajectories, then an action can be represented as a matrix $A \in R^{M \times L}$, and each row is a single feature generated from the corresponding trajectory. Therefore, the training set is a concatenation of all video actions, i.e., $TA = A_1 \cap A_2 \cap \dots \cap A_N$, where N is the number of training videos, and the total number of rows, which equals the total trajectory number in the training dataset, is calculated by $TM = \sum_{i=1}^N M_i$.

Given a training set TA , a codebook is obtained using the K-means clustering algorithm. Then, each centre of a quantized area of a cluster is defined as a visual word, and the cluster number is set to $K=1024$, which shows great performance on action datasets. A subset of 100,000 features is randomly selected from the whole training set to reduce the computational cost. Once the codebook is constructed, the BOF representation assigns each feature to the nearest visual word and accumulates the account of visual words. Then the histogram is normalised to characterise video event representation; thus, an action matrix A is represented as a K -component feature vector.

3.5.2 Soft Assignment

This research has applied a “soft-assignment” approach to rectify the aforementioned disadvantages based on the multi-assignment (MA) technique that can “split” a feature into multiple visual words (Bolvinou et al., 2013). In this case, a top- N nearest visual words method is devised for computing the weights for each visual word, and then the weights for a complete video sequence can be calculated as:

$$u_k = \sum_{i=1}^N \sum_{j=1}^{M_i} \frac{1}{2^{i-1}} \text{sim}(j, k), k \in [1, K]. \quad 3-5$$

where u_k indicates the weight of k -th visual word, M_i describes the number of features whose i -th nearest neighbour is the visual word k , and function $\text{sim}(j, k)$ calculates the similarity between the feature j and visual word k . Generally speaking, $N = 4$ achieves notable improvements compared with the previous work (Bolvinou et al., 2013). Finally, a video event can be represented by the vector $TV=[u_1, u_2, \dots, u_K]$.

3.5.3 BoTF Formulation

As stated earlier, a BOF encodes a video event as a set of unordered local features. As a result, it struggles to deal with the temporal sequences of features, which could lead to problems in distinguishing “longer” or various actions that constitute similar atomic components but in different orders, such as the motions of standing up and sitting down. To address this issue, this research devises a new feature representation method: Bag-of-Temporal-Features (BoTF) that embeds temporal information into BOF representation by employing the visual word correlograms and a co-occurrence transaction (CoTrans) scheme (Kieu et al., 2017). A correlogram not only contains the global spatial feature distribution of a video frame but also has the corresponding spatial and temporal information encapsulated together (Bolovinou et al., 2013). Moreover, the CoTrans template has been applied to form feature patterns and calculate the BoTF instances.

As a live implementation strategy, DT produces a set of low-level feature vectors $V=\{v_i\}$, where v_i represents a low-level feature of a video event. To explore the temporal information, this research introduced the time information into v_i , so the feature is extended as $[t, v]$, where t indicates the time coordinate. In particular, t is the time centre belonging to its trajectory. All features of a video event are ordered by temporal sequences (frame indexes), see Figure 3-8. Under the proposed system, the sequence for an event in a given time range l is denoted as: $PT = [t_c, l, ori, \mathbf{v}]$, where t_c is the time centre, l denotes the number of frames on the time-axis for the corresponding patch, $ori=\pm 1$ represents the orientation of polar axis, so a patch is defined as the following:

$$PT(t_c, l, ori, \mathbf{v}) = \{[t_c, l, ori, v_i]\}, v_i \in V . \quad 3-6$$

And then, the CoTrans template is applied to calculate the BoTF instances based on all defined feature patches. The histogram $h(t_c, l, ori)$ encodes features in a feature patch PT by calculating the number of every visual word in PT . It is defined as the following:

$$h(t_c, l, ori) = (c_1, c_2, \dots, c_k) . \quad 3-7$$

where k is the length of the codebook and c_i is the number of features in patch PT belonging to the i -th visual word.

In this step, similar to BOF, all low-level features extracted from a training dataset are clustered by using K-means for generating the codebook (a visual word set) of BoTF, and the vector length of $h(t_c, l)$ is equal to the length of the codebook generated by BOF. Moreover, the radial axis (R) is divided into $N^r = 4$ bins (N^r is equal to the number of feature patches on a quadrant of the radial axis), the length of R is 60 frames, and the polar axis ($\pm th$) is divided into $N^{\pm th} = 2$ bins, which is equal to the number of orientations of the polar axis, see Figure 3-8. Finally, the BoTF descriptor can be formulated as the following:

$$BoTF_{t_c} = [h(t_c, l, 1)_1, h(t_c, l, -1)_1, \dots, h(t_c, l, 1)_4, h(t_c, l, -1)_4]. \quad 3-8$$

The set of CoTrans reference time centres is denoted as $C = \{t_1, t_2, \dots, t_n\}$ that are sampled from the time-axis by the successive 30 frames. With the BoTF descriptor, input video streams can be represented as a set of $BoTF_{t_c}$ descriptor instances. In conclusion, a video event is first described as a histogram of BoTF based visual words, and then the Equation 3-5 will be applied to assign a $BoTF_{t_c}$ into multiple visual words.

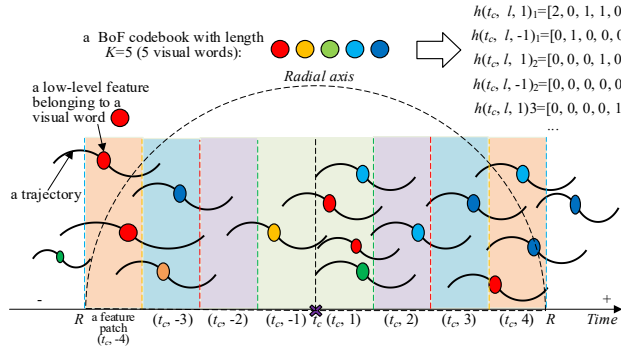


Figure 3-8. Producing $BoTF_{t_c}$ instances based on the BoF and the CoTrans templates.

3.6 Action Classification

3.6.1 Feature Fusion and Dimensionality Reduction

The feature fusion strategy developed in this work enabled robust human action classification through a SVM based classifier. The three event representations (FV, BOF and BoTF) derived from the aforementioned models are fused into a final holistic video representation:

$$fv = [\lambda_1 FV, \lambda_2 BOF, \lambda_3 BoTF], \quad 3-9$$

where λ_i ($i = 1, 2, 3$) indicates the weight of each feature vector, this research considers all feature representations are equally weighting with a normalised $\lambda_i = 1$, so a video event can be represented as the holistic feature vector in real-time: $[FV, BoF, BoTF]$. FV, BOF and BoTF event representation have 1024-competent feature vectors for each category of feature descriptors; hence the length of the combination of FV, BOF and BoTF is $1024 \times 3 = 3072$. For classifying human actions, one or more types of feature descriptors can be used, i.e., either a single descriptor or the random combination of these descriptors can be used for event representation. Consequently, the holistic video representation is very high dimensionally. For instance, when fusing the TS, HOG, HOF and MBH, the length of the final feature vector reaches $1024 \times 5 = 5120$, the curse of dimensionality is a critical problem in this method due to directly using the finite high-dimension feature set to train a SVM classifier will cause low convergence rate (Spruyt, 2014). Therefore, the dimensionality reduction method is indispensable in this method. This research adopts the unsupervised Principal Component Analysis (PCA) technique to reduce the holistic video event representation dimensionality. The PCA method projects each data point in the original space onto the first few principal components to map low-dimensional data while still retaining the maximal data variance. In practice, the fused holistic vector is projected into a lower dimension of a 1024-component vector. Noted that if only a single descriptor (e.g., TS) and one representation (e.g., BOF) are used, the PCA method is not required.

3.6.2 SVM based Classifier

SVM is the optimal choice for dealing with relatively small sizes of handcrafted features. Thus, to test and evaluate the validity and efficiency of the devised framework, this research investigated a SVM based classifier by comparing its performance when handling different handcrafted features and representations. To classify multiple categories of actions, multi-SVM units have been generated, and each performs the

“one-versus-the-rest” multi-class evaluation. In this research, a dataset splits into three parts, i.e., the training subset (70%), the validation subset (10%) and the test subset (20%). A cross-validation strategy (Wong, 2015) has been applied to train the SVM-based classifier to ensure accuracy and repeatability.

3.7 Experimental Results

The experiments are carried out on UCF 11, UCF 50, HMDB 51 and JHMDB 51 datasets. Details of the datasets can be found in Section 2.9. In this experiment, single-level DWT is performed to decompose the original video frames into low scales, and the following settings are defined: in the trajectory phase, the window size of tracked frames is 32×32 pixels, and the trajectory length $L=15$; in the event representation phase, the cluster number $K=1024$ is used for the K-means clustering algorithm, which constructs a 1024 length codebook, and a feature is assigned into its top-4 nearest visual words by their Euclidean distances; the fused event representation is reduced to a 1024-component vector by PCA dimensionality reduction before it is fed into the Gaussian kernel SVM classifier.

3.7.1 Visualisation of Trajectories

For qualitative analysis of the DWT-based dense sampling and tracking effect, this experiment chooses three action videos from the UCF 50 dataset for visualisation, i.e., “basketball shooting”, “football juggling”, and “walking” actions. DWT algorithm is applied to decompose the original frame and then extract and track the spatial-temporal interest points by dense sampling and median filter methods on each scale. The results are shown in Figure 3-9, where the red points indicate the interest points extracted by dense sampling, and the green lines are trajectories. It can be seen that the obtained trajectories are mainly located in the human and movement areas. Taking the “football juggling” action as an example, almost all trajectories are generated by the player, while the background is eliminated. Based on this observation, the DWT driven DT method can character movements gracefully, and the trajectories have the capacity to embed human motions for later processing. Nevertheless, the sample of “walking” action is

not good enough; the trajectories not only occur in the motion areas but in the background as well. More significantly, as shown in Figure 3-10, many trajectories are generated in the background area in the latter part of the video clip since the viewport is smoothly moving. This camera motion causes the side effect on motion description. The following section will discuss the camera motion removal experiment.




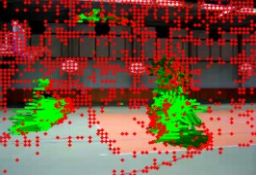
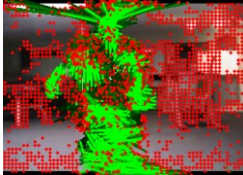
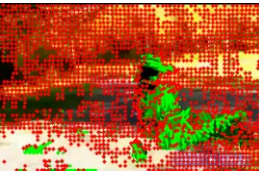
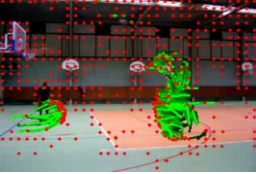
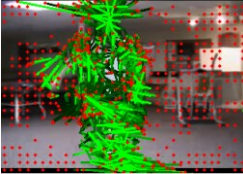
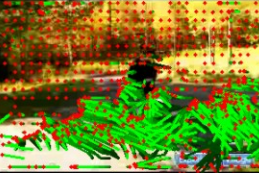
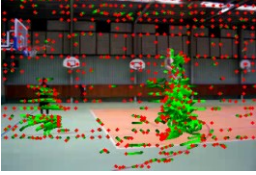
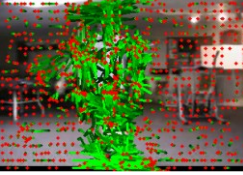
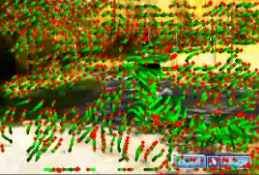
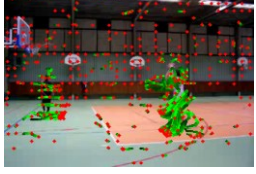
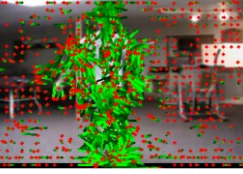

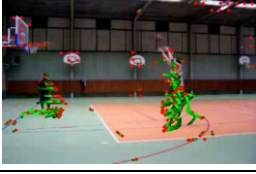
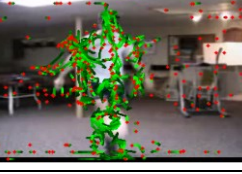

Action	basketball shooting	football juggling	walking	
Frame				
Feature points and trajectories	Raw			
	A			
	H			
	V			
	D			

Figure 3-9. Visualization of trajectory results.



Figure 3-10. The obtained trajectories from a “walking” action video.

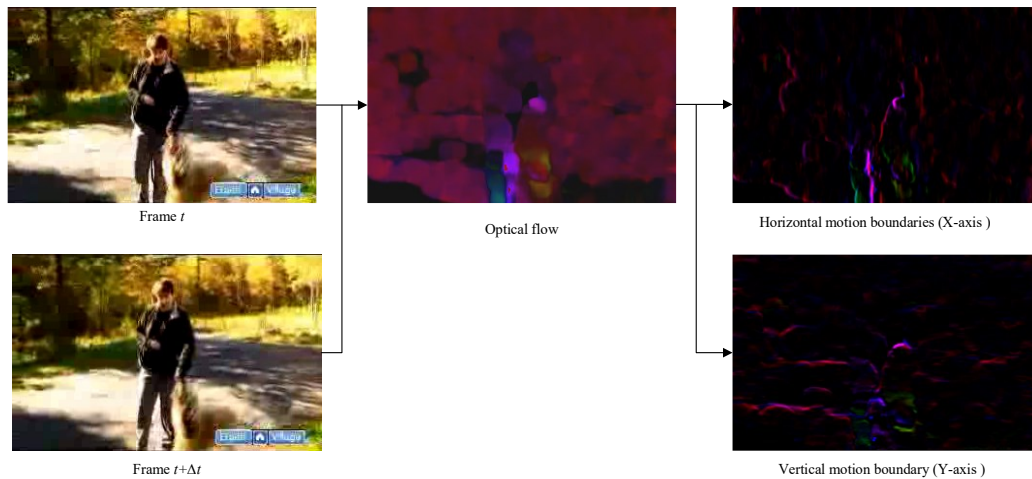


Figure 3-11. Visualization of optical flow and the corresponding motion boundaries.

3.7.2 Camera Motion Removal Effect

For qualitative analysis of the motion boundary for camera motion removal, this experiment computes and visualises the original frames and their corresponding optical flow and the motion boundaries along with X and Y directions, respectively. Two consecutive frames of a “walking” action video coming from the UCF dataset are extracted. These video frames contain both human movement and camera motion; then, the dense optical flow is computed by performing the Farnebäck method (Farnebäck, 2003), and the result is shown in the middle of Figure 3-11, in which the human shapes can be seen clearly, and there is also plenty of noises in the background area. Then, the method from Equation 3-4 is further performed, resulting in the horizontal and vertical motion boundaries, as shown in the right of Figure 3-11, in which the human shapes are kept while the inherent noises came from the camera motions are removed. This result qualitatively proved the performance gain of MBH and MBH3D descriptors.

3.7.3 Feature Descriptor Efficiency

To evaluate the effectiveness of different feature strategies, this experiment compares various combinatory feature descriptors involving TS, HOG, HOF, MBH and their 3D counterparts. The combined video representation based on the BOF, FV and BoTF methods was applied to encode a video event as a holistic feature vector that reduced dimensionality before going through a SVM classifier for action classification. The UCF 50 dataset was used, and the results are shown in Table 3-1. It can be seen

that a single TS feature generally offers the weakest performance, while the MBH(3D) achieves the best accuracy rate among those individual features due to the MBH descriptor focuses on tracking human foreground motions, whilst the camera motions and background change are removed. Unsurprisingly, the combined descriptor demonstrates the best performance by harnessing the advantages of underlying feature types, with the recognition rate reaching 93.8%. One possible reason is that certain actions in UCF videos have more salient motion information (e.g., “TaiChi” and “High Jump” actions) while other actions possess less distinctive motions within different scenes, e.g., “biking” and “horse-riding” actions. An individual feature descriptor (e.g., MBH) can extract motion information on the former type of action, and it often falls short in handling actions of the latter type, resulting in the loss of overall accuracy. In contrast, when applying the combination features, both motion and scene information can be extracted more thoroughly, and the temporal can also be encoded by the BoTF event representation to improve recognition rates for a wider spectrum of action types.

As shown in Table 3-1, the performance of the combined handcrafted features (i.e., the combinations of TS, HOG, HOF and MBH) is better than any individual one, which indicates the relevance of all aspects of handcrafted features towards the final prediction results. Based on this observation, this research integrated the combined handcrafted features for the rest of the work. Moreover, this research also tested the performance of 2D and 3D descriptors separately to gain insight into their respective impacts on the outcome. The 3D appearance and motion descriptors have further demonstrated their superiority over the 2D-based feature descriptors on all tested benchmarks drawn from ablation studies.

Table 3-1. The recognition accuracy rate (in %) of different features and event representations on the UCF 50 dataset.

Features	Representations		
	BOF	FV	FV+BoTF
TS	67.2	75.2	76.5
HOG	68.0	82.6	83.6
HOF	68.2	85.1	87.5
MBH	82.2	88.9	90.3
TS+HOG+HOF+MBH	84.5	91.2	92.5
HOG3D	NA	NA	85.8
HOF3D	NA	NA	89.3
MBH3D	NA	NA	92.6
TS+HOG3D+HOF3D+MBH3D	NA	NA	93.8

3.7.4 Event Representation Validation

The performance of BOF, FV and BoTF have been applied to the UCF50 dataset for evaluation. Table 3-1 illustrates the recognition accuracy variation of these three event representation approaches. It is shown that the pure FV implementation is better than the BOF method on TS, HOG, HOF and MBH descriptors. Unsurprisingly, when combined with FV and BoTF, it achieves the best performance on tested UCF50 instances. One key reason is that the BoTF representation encodes the features in different time patches, and it can describe the temporal sequences of features to handle the “longer” and various actions. The superiority is rooted in the presence of both local and global features over the spatial and temporal domains.

3.7.5 Comparison With the Other Approaches

This experiment has compared the proposed handcrafted model with other trajectories-based methods on UCF 50, HMDB51, and JHMDB51 datasets, including the classical DT model (Wang et al., 2013), iDT (Wang & Schmid, 2013)), SFV (Peng et al., 2014). HMDB51 and JHMDB51 datasets are more complex than UCF datasets in terms of action types, video quality, and background. Experiments show a superior output from the devised model in this research, as highlighted in Table 3-2. The performance of the devised model consistently levels up or surpasses the current benchmark approaches. The superior performance stems from the trajectory-based features among multiple scales, separable frequency bands and directions, and the spatial-temporal video event representations.

Table 3-2. Performance comparison to the state-of-the-art approaches on UCF 50, HMDB51 and JHMDB51 datasets (in %).

Method	UCF50	HMDB51	JHMDB51
DT	84.5	46.6	NA
iDT	91.2	57.2	62.8
SFV	NA	66.79	69.03
TS+HOG+HOF+MBH	91.3	68.3	70.4
TS+HOG3D+HOF3D+MBH3D	92.5	70.2	71.8

3.8 Summary

In this chapter, to tackle the shortfalls of lacking orientations and separable frequencies in multiple scales of traditional DT-based action classification models, this

research has developed an innovative DT model by integrating the DWT technique. The 2D DWT method is employed to decompose the video frames into separable frequency and orientation components for abstracting motion information. The dense trajectories method is applied to extract feature points for tracking through consecutive frames. FV and a novel handcrafted event representation - BoTF, have been developed to encode the “longer” temporal information on video clips. The holistic representation of video-based events over time, specifically human actions in this research, enables efficient and accurate analysis through SVM-based classification. The preliminary experiments carried out on the UCF and HMDB datasets show that the proposed handcrafted model has a superior recall, robustness, and extensibility performance over benchmarked systems and approaches. However, the handcrafted approaches have critical drawbacks when facing real-world applications: 1) it primarily depends on expert-designed and dataset-specific feature extractions and representations, which are less robust; 2) the real-time processing is always unavailable since the methods are complex and, it requires a lot of computational resources. In addition, the volume of storage and memory are also indispensable for storing the middle features; 3) only the classifier is learnable, while the other stages are manual work which cannot be evolved automatically from observational data. A better strategy is building end-to-end mechanisms and automatically searching for the best formulas for feature design and action prediction. The following chapters concentrate on the deep learning-based methodology to explore these mechanisms.

CHAPTER 4 Multimodality Neural Networks

4.1 Introduction

The deep learning technique offers end-to-end feature extraction and classification by contracting sophisticated network architectures based on neural models such as CNN, RNN, LSTM, and GCN (Goodfellow et al., 2016; Spinelli et al., 2021). Since its birth almost ten years ago, deep learning has gained tremendous interest and outstanding results in tasks such as image classification (He et al., 2016), object detection (Ren et al., 2017; Wang et al., 2021b), natural language processing (NLP), and many other industrial applications. Complex video analytical tasks such as single human and crowd behaviour understanding are still ongoing challenges due to many ill-posed real-world application problems. The most remarkable approach is the two-stream CNN model (Simonyan & Zisserman, 2014), in which a spatial stream is designed for “appearance” feature extraction, and an additional temporal stream is integrated for motion feature learning. The multi-stream based method has drawn wide attention, and many improved models have been developed. The details have discussed in Chapter 2.6.5.

The multimodality deep neural networks have shown great potential in handling complex spatial-temporal features that are essential for video event analysis. However, the two-stream network only provides a coarse integration of both appearance and motion features, which omits critical information such as spatial and temporal interactions. Furthermore, since the temporal stream only receives ten consecutive optical flow images in most two-stream models (Simonyan & Zisserman, 2014; Xu et al., 2019a; Ye et al., 2015), it may be confused if two actions are similar in such a short snippet, even they are different in the longer timeframe. Moreover, extracting optical flows from video frames is a time-consuming operation; as a result, the two-stream network method can only be trained and tested in offline mode.

To tackle these shortcomings, this research first constructs a novel two-stream network model based on residual networks, which boosts the learning capability of each stream. Then, this research significantly reduces the computational cost by integrating

the devised Optical Flow-guided Feature (OFF) layers in the motion stream. In this design, the time-consuming optical flow extraction stage is no longer required, hence supporting online training and testing. Furthermore, the proposed concurrent two-stream aggregation network can learn the coarse-to-fine scene and motion interactions from the joint spatial-temporal exploitation by building residual connections between the motion and the visual streams, named spatial-temporal fusion blocks (STFB). In the constructed network, each stream is trained separately, and the outputs (feature maps) from each stream are accumulated in time sequences, which is followed by a 3D CNN sub-network for learning long-term semantic information in both spatial and temporal domains. Finally, a softmax layer is adopted for action classification. The concurrent interactive spatial-temporal aggregation model achieved promising improvements.

4.2 Learning Video Features by DNN

4.2.1 Pre-trained Feature Adaptation

Training DNN models is very time-consuming and requires a huge amount of labelled data. However, many popular action datasets are not adequate for the particular task, which limits the applications of DNN models. On the other hand, a deep learning model is constructed as a hierarchical structure, whose bottom level primarily focuses on general image features such as STIP and edges, while the abstract object and motion features are described in a multilayer nonlinear structure. The top layers, which are consistently implemented by FCNN and softmax modules, are task-specific functions for particular classification or recognition decisions (Samek et al., 2017). Based on this concept, a pre-trained DNN model can be considered a general feature extractor. Here, a DNN architecture can be depicted as follows:

$$\begin{cases} F=f(wI,b) \\ G=g(w'F,b') \end{cases} \quad 4-1$$

where w and w' are learnable weights belonging to CNN kernels and the fully-connected layers, respectively, b and b' are learnable biases, I indicates an image or a video frame, $f(\cdot)$ is a learnable function which presents the CNN layers of deep learning architecture, while $g(\cdot)$ indicates the fully-connected layers. According to Zeiler and

Fergus (2014), $f(\cdot)$ is more generic, and it extracts image features such as interest points, lines and edges in different CNN layers, so it is reasonable to apply the pre-trained CNN models to extract features. In practice, this research extracts the vector values from the fully connected layer of the networks and removes the rest layers since only the specified features from CNN models are of interest. In this method, only the first (fc_1) and second (fc_2) fully-connected layers of G in Equation 4-1 are kept. It is worth noting that various pre-trained networks and fully-connected layers can be integrated, e.g., AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2015), and ResNet (He et al., 2016). Both of them were pre-trained on the ImageNet ILSVRC-2014 classification dataset.

From a given video clip, it can be formalized as $V = \{I_i | i \in [1, L]\}$. This research firstly extracts the learned image feature h_i from each frame I_i by the pre-trained CNN model. A series of image feature vectors $H = \{h_i | i \in [1, L]\}$ of the clip can be obtained, and then the feature fusion method is applied by averaging the series of feature vectors, which outputs a video feature p , where

$$p = \frac{1}{N} \sum_{i=1}^N h_i, N = L. \quad 4-2$$

4.2.2 Dual-stream CNN-RNN Network

This research explores a dual-stream CNN-RNN architecture due to its remarkable capacity to encode visual (RGB) and motion (optical flow) features simultaneously in the spatial and temporal domains. The dual-stream model has achieved reasonable results on human action recognition at the accuracy of 88.0% and 59.4% on UCF 101 and HMDB51, respectively (Simonyan & Zisserman, 2014), and the performance achieves the benchmark level of the improved DT method in the pre-DL era. However, the dual-stream CNN model neglects the intrinsic differences between temporal and spatial domains. To alleviate this shortcoming, the devised framework incorporates the strengths of both the 3D CNN in the spatial domain and the RNN for handling the temporal features. The whole network design is shown in Figure 4-1.

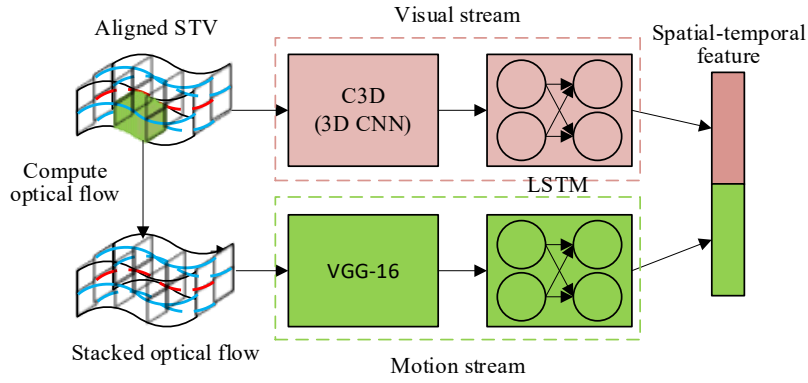


Figure 4-1. The CNN-RNN based dual-stream network architecture. The visual (RGB) stream (on top) is designed by combining 3D CNN and LSTM to extract “appearance” context and sequence information from raw pixels of STV data, while the motion (optical flow) stream (on bottom) learns motion information from optical flows by the CNN-LSTM structure.

The visual (RGB) stream is comprised of two components: the 3D CNN-based “appearance” feature extractor and the RNN-based sequential descriptor. In practice, this research applies the C3D model as the CNN component. C3D uses 3D convolution and 3D pooling operations on each layer. This research uses a $3 \times 3 \times 3$ convolution kernel for convolutional layers, and all pooling layers are max pooling with kernel size $2 \times 2 \times 2$. With this configuration, C3D is trained on 15 consecutive frames (STV) with the input size of 3 (channel) \times 15 (frames) \times 112 (pixel) \times 112 (pixel) and outputs 2049 units in the last fully connected layer, which is followed by a RNN structure for sequential modelling. The RNN units can automatically discover appropriate sequential information (Zhang et al., 2017), i.e., learning connections between inputs and the corresponding previous states continuously, which is ideal for extracting temporal information in videos. However, RNN is not suitable for directly learning sequential features from high-dimensional data. Therefore, in this design, the features generated by CNN become the input of the subsequent RNN for optimisation. A LSTM model has been applied in this design instead of the traditional RNN module for its unique ability to remember “states” over a long period of time by using the “forget” mechanism. The devised RNN structure has two LSTM layers, and each of them has 1024 hidden states, so the RNN component outputs a 1024-component feature vector.

The motion (optical flow) stream is also constructed by the CNN and RNN components. Different from the visual stream, the motion stream mainly extracts

temporal action features from the successive flow fields. This research adopts the VGG-16 network as the CNN component. With this configuration, VGG-16 is trained on a stacked optical flow computed from the STV block, so the input size is 2 (channel) \times 15 (frames) \times 112 (pixel) \times 112 (pixel), and the output is a 2049-component vector in the last fully connected layer that is followed by a RNN for sequential modelling.

4.2.3 Training

For training the 3D CNN model, this research applied the same parameter settings in accordance with Tran et al. (2015), and the C3D network was trained directly by using UCF 101 video clips. The developed CNN models (i.e., VGG and C3D) are used as general feature extractors, whilst temporal features are identified through training LSTM-based sequence models. In practice, the features from the last fully connected layer are fed into LSTM units with M inputs $\langle x_1, x_2, \dots, x_M \rangle$ and M outputs $\langle y_1, y_2, \dots, y_M \rangle$, where x_i presents a feature vector and y_i is the corresponding action label. The learnable weights (WR) of the LSTM-based sequence components can be optimised by maximising the likelihood of the ground truth outputs y_t calculated on the input data and the action labels. For a given training sequence $(x_m, y_m)_{m=1}^M$, this study minimises the negative log-likelihood $L(WR) = -\sum_{m=1}^M \log P_{WR}(y_m | x_{1:m}, y_{1:m-1})$ using stochastic gradient descent (SGD) (Lecun et al., 1998) with a backpropagation algorithm to compute the gradient of the objective L with respect to the weights (WR).

4.2.4 Transfer Learning

It is a challenge when handling real applications where datasets are often referring to noisy and untrimmed videos. As a result, many deep learning methods only achieved a low performance that is even worse than the shallow handcrafted representations. Transfer learning suggests a significant advancement to utilise and be benefitted from small datasets (Yosinski et al., 2014), i.e., through training an initial network from scratch on a very large dataset (e.g., an ImageNet-like dataset) and then fine-tuning the model on a task-specific dataset. However, the datasets used in this research are different from ImageNet. Directly applying transfer learning will cause the underfitting

problem. Motivated by this analysis, this research developed a multi-stage training strategy based on transfer learning. A public CNN model (e.g., VGG-16 or ResNet) pre-trained on the ImageNet ILSVRC-2014 dataset was adopted as the initial network. These pre-trained models can be derived from online model repositories such as PyTorch Hub (Paszke et al., 2019). Then the model is fine-tuned on a small action dataset (e.g., UCF action dataset) to ensure the robustness of the trained model. The small action dataset supplies sufficient videos to fine-tune the entire network from image classification to motion analysis.

4.3 Concurrent Spatial-temporal Network

4.3.1 The Overall Network Architecture

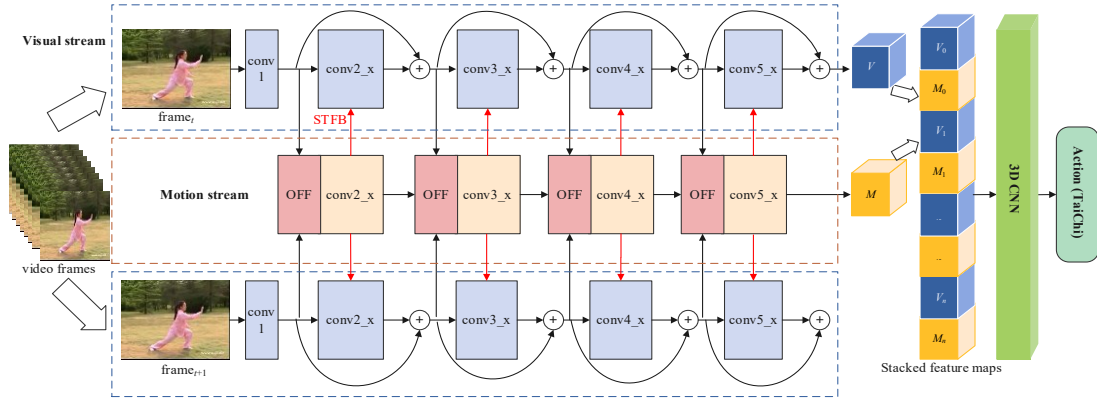


Figure 4-2. The architecture of the two-stream concurrent interactive spatial-temporal aggregation model. It contains two network streams: the visual stream that learns spatial features from a video frame, and the motion stream that derives temporal features from visual feature maps by OFF and residual blocks.

The proposed concurrent interactive spatial-temporal aggregation model is shown in Figure 4-2, which is constructed by the classic two-stream network model with several improvements: firstly, instead of using the original convolutional filters of the two-stream network, this research applies ResNet and removes the last fully-connected layer for both visual and motion streams, which is considered the baseline network of this research. Then, this design removes the first convolutional layer of ResNet in the motion stream and breaks the rest layers into four convolutional modules, whose module contains several residual blocks, and each module integrates an OFF layer in the front part to derive motion information. This research also injects the constructed

fusion blocks (STFB) between motion and visual stream to generate more comprehensive spatial-temporal interactions at different residual blocks, hence introducing the joint coarse-to-fine motion-visual interactions. In this devised spatial-temporal aggregation network, only the successive RGB video frames are required for the inputs, and the spatial and temporal feature maps can be generated concurrently and stacked in time order. Finally, a 3D CNN sub-network is integrated for abstracting the long-term semantic and event representation in spatial and temporal domains.

4.3.2 Baseline Two-stream Network

The devised concurrent two-stream aggregation model stems from the classic two-stream idealism: the visual stream extracts appearance information from RGB frames, and the motion stream is used to extract movement information by using the optical flow that is a robust motion descriptor. A residual network has been applied to implement both the visual stream and motion stream because ResNet shows an acceleration of the training speed and the outstanding capacity for feature extraction (Zhu et al., 2016). Different configurations of ResNets (e.g., ResNet-18/34/50/101/152) (He et al., 2016) can be applied in this model, and this research applies ResNet-50 for both visual and motion streams for balancing the performance and computational cost trade-offs. The ResNet-50 contains three major components: a first convolutional layer with a kernel size of 7×7 , a max-pooling layer with a kernel size of 3×3 , and four convolutional modules, i.e., conv2_x, conv3_x, conv4_x, conv5_x, respectively, and each have several residual blocks followed by a pooling layer. Finally, a fully connected layer is followed for classification. The main advantage of ResNet is that it has a bypass function directly connecting the convolutional layer to a latter layer. This operation is defined as a residual block, as shown in the following:

$$y = F(x, W_i) + x, \quad 4-3$$

where x and y represent input and output tensors of the target layer, respectively, the function $F(x, W_i)$ defines the residual mapping to be learnt. The bypass function allows the latter layer to learn residual representations to maintain gradients for improving the learning accuracy and reducing the learning complexity. The detailed network setups

of ResNet-50 for the two streams are presented in Table 4-1. In this design, the last fully connected layer of both visual and motion streams is removed due to this design only preserves the feature maps from the final convolutional layer before stacking them to learn long-term semantics and relationships in both spatial and temporal domains. Further, the first convolutional layer (conv1) in the motion stream is also removed because this model does not process optical flows.

Table 4-1. Architecture of visual and motion streams. This research applies ResNet-50 to construct both streams. Other settings of ResNets (ResNet-18/34/101/152) can also be adopted.

Layers/modules	Residual Blocks	
	Vision Stream	Motion Stream
Conv1		
Pool1		
Conv2_x	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	STFB $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv3_x	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	STFB $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
Conv4_x	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	STFB $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
Conv5_x	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	STFB $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
Pool	7×7 avg, stride 1	

4.3.3 OFF Fundamentals

Instead of taking optical flows directly as inputs for temporal information learning, this model generates motion features from the feature maps of the visual stream based on the constructed OFF layers in the motion stream. The OFF was firstly presented by Sun et al. (2018b), and it was inspired by the traditional optical flow to capture the motion information of a video. In principle, the traditional optical flow assumes the brightness of any pixel at time t to $t+\Delta t$ remains constant, i.e., supposed that a pixel of a video frame I at the position (x, y) and time t is donated by $I(x, y, t)$, then the corresponding pixel at time $t+\Delta t$ is $I(x+\Delta x, y+\Delta y, t+\Delta t)$, where Δx and Δy are the spatial displacements along the x - and y -axis, respectively, and:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \quad 4-4$$

At the feature level, Equation 4-4 is rewritten as:

$$\phi(I; \theta)(x, y, t) = \phi(I; \theta)(x + \Delta x, y + \Delta y, t + \Delta t) , \quad 4-5$$

where $\phi(I; \theta)$ is a convolutional network based feature extraction and θ is the trainable parameters. By calculating derivatives of Equation 4-5 and assuming $q = (x, y, t)$, then:

$$\frac{\partial \phi(I; \theta)(q)}{\partial x} \Delta x + \frac{\partial \phi(I; \theta)(q)}{\partial y} \Delta y + \frac{\partial \phi(I; \theta)(q)}{\partial t} \Delta t = 0 , \quad 4-6$$

by dividing Δt of Equation 4-6, the following equation can be calculated:

$$\frac{\partial \phi(I; \theta)(q)}{\partial x} v_x + \frac{\partial \phi(I; \theta)(q)}{\partial y} v_y + \frac{\partial \phi(I; \theta)(q)}{\partial t} = 0 , \quad 4-7$$

where (v_x, v_y) is the speed of the feature point q along with two directions. The first two fractions are the spatial gradients along the x - and the y -axis, respectively, and the third fraction is the time gradient along the t -axis. If the $\phi(I; \theta)$ function meets the constant brightness constraint, i.e., $\phi(I; \theta) = I(q)$, then according to the optical flow definition, (v_x, v_y) is the optical flow which can be computed by solving the constraint optimization problem of each pixel point q by using Equation 4-6, as shown in the following:

$$\vec{Q}(I; w)(q) = \left[\frac{\partial \phi(I; \theta)(q)}{\partial x}, \frac{\partial \phi(I; \theta)(q)}{\partial y}, \frac{\partial \phi(I; \theta)(q)}{\partial t} \right], \quad 4-8$$

where $\partial \phi(I; \theta)(q) / \partial t$ denotes the differentiation of two continuous video frames. Based on this definition, this research optimises the optical flow representation from $I(q)$ into the feature $\phi(I; \theta)(q)$, such that (v_x, v_y) becomes the feature flow (Sun et al., 2018b); $Q(I; w)(q)$ is complementary to the feature flows and is also orthogonal to the optical flow; thus, it contains the optical-like spatial-temporal information and is guided by the feature; hence an Optical Flow guided Feature (OFF). It can replace the time-consuming optical flow computation for keeping entire motion information and significantly reduce computational cost.

4.3.4 OFF Layers

According to the OFF principle, this research designs a so-called OFF layer for formulating feature flows. Figure 4-3 illustrates the OFF-layer structure that contains a

1×1 convolutional layer to reduce the channel number of feature maps outputted from the visual stream. The element-wise subtraction obtains the temporal gradient, and Sobel obtains the spatial gradient. After the feature flows have been produced, a final concatenation operation is integrated to fuse the temporal and spatial gradients along with the last low-level feature flows, and then the combined features will be outputted to the residual modules for obtaining the fine spatial-temporal features. According to Equation 4-7, OFFs have both spatial and temporal gradients. The devised network applies the Sobel operator to get the spatial gradient, as shown in the follows:

$$G_x = \left\{ \begin{bmatrix} -1, 0, 1 \\ -1, 0, 1 \\ -1, 0, 1 \end{bmatrix} \otimes \phi(I, n) \mid n = 0 \dots N_c - 1 \right\}, \quad 4-9$$

$$G_y = \left\{ \begin{bmatrix} 1, 1, 1 \\ 0, 0, 0 \\ -1, -1, -1 \end{bmatrix} \otimes \phi(I, n) \mid n = 0 \dots N_c - 1 \right\}. \quad 4-10$$

where G_x and G_y express the spatial gradients of OFFs along with x and y directions, respectively; \otimes is a convolution operation; $\phi(I; n)$ is n -th channel of feature map $\phi(I)$ that has N_c channels. The element-wise subtraction is integrated to compute the temporal gradient G_t , formulated in the following:

$$G_t = \{ \phi_t(I, n) - \phi_{t-\Delta t}(I, n) \mid n = 0 \dots, N_c - 1 \}. \quad 4-11$$

Along with the corresponding feature flows, the OFF layer fuses the three gradients G_x , G_y and G_t to form the OFF outputs for learning the fine spatial-temporal features.

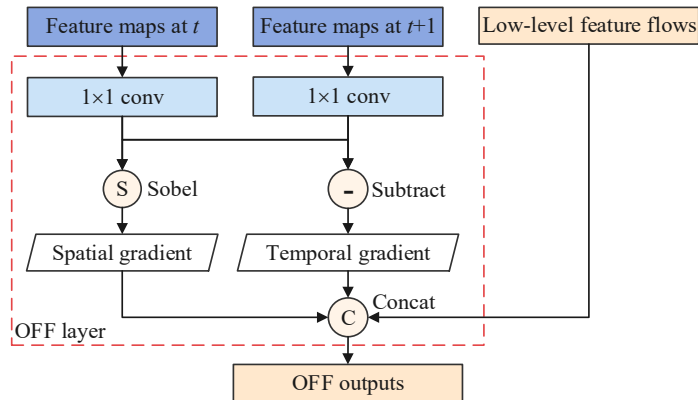


Figure 4-3. The structure of an OFF layer. It contains two 1×1 CNNs followed by a Sobel and a subtracter generating spatial and temporal gradients concurrently. A final concatenation operation fuses the gradients and the last low-level feature flows.

4.3.5 OFF based Motion Stream

In the devised motion stream, the ResNet-50 is broken into four residual modules, and an OFF layer has been constructed at the front of each module, hence the OFF-based motion stream consists of four OFF layers and four residual modules having different resolutions on different feature levels. As shown in Figure 4-2, two successive video frames are served as two inputs for the two visual streams, respectively, to obtain basic features (feature maps). Only the same network layers will have the same resolution feature maps that can be concatenated by the corresponding OFF layers, e.g., the feature maps outputted from conv2_x in the visual stream are served as the input of the first OFF layer in the motion stream. Based on the improved network design, the motion stream does not require any optical flow computation but directly extracts spatial-temporal features and their joint information at multiple convolutional levels. The visual and temporal streams are also processed concurrently on processors, hence greatly reducing the computational cost in addition to the enhanced human action representation abilities for the network.

4.4 Spatial-temporal Aggregation

The OFF based network predicts action labels from visual and motion streams separately, and a single score-based classifier is used to fuse the two contributions. This coarse fusion method ignored the finer grain interactions between motion and visual streams at variant levels. It is unreliable to represent the object (or human) in the visual stream and abstract movements in the motion stream concurrently. To better fuse the predictions of the two streams, this research enables a fine interaction and fusion from bottom-to-top levels through three innovative steps: 1) injecting STFB to achieve low-level motion and spatial interactions; 2) performing the summing operation at the last convolutional layer to fuse the middle-level visual and motion features; 3) applying 3D CNN sub-network for high-level semantic and event representation.

4.4.1 STFB

It is widely accepted that different convolutional layers in a convolutional network extract the different levels of features, i.e., from bottom to top layers, they refer to STIP, edges, lines, objects, semantic abstraction, and event representation. It is anticipated that comprehensive connections between visual and motion streams can gain fine interactions at different feature levels. The ST-ResNets model integrates spatial-temporal information by injecting temporal information into the spatial stream (Feichtenhofer et al., 2016). Accordingly, this research explores the STFB module to achieve coarse-to-fine interactions between motion and visual streams.

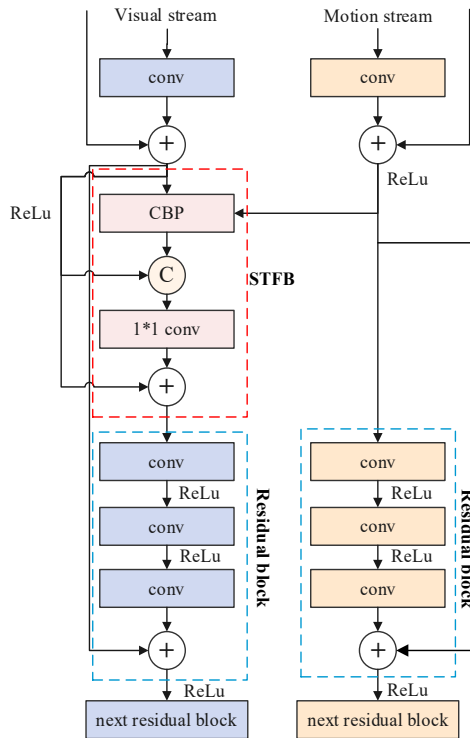


Figure 4-4. STFB in a residual block pair. This research inserts STFBs between motion and visual streams with the corresponding residual block pair.

As shown in Figure 4-4, this research devised a residual connection mechanism for implementing the two-stream fusion called STFB. The core component in the devised STFB is the integrated compact bilinear pooling (CBP) structure for combining spatial-temporal features in the compact representation, which is of vital importance to high-quality information learning for human action recognition. A good fusion strategy should maximize the interaction of features while best preserving spatial-temporal information; so, the bilinear fusion is applied through the cross-product calculation on

the two feature maps, which allows all spatial-temporal features of different dimensions to interact with each other and integrates all channels of the feature maps from the two streams. The fusion function can be calculated as the following:

$$y^{bil} = \sum_{i=0}^H \sum_{j=0}^W x_{i,j}^s T x_{i,j}^t, \quad 4-12$$

where $x^s \in \mathcal{R}^{C \times H \times W}$ and $x^t \in \mathcal{R}^{C \times H \times W}$. However, the high dimensionality of fused features severely limits its application in real-world problems. To overcome this drawback, CBP is explored to retain the effect of the bilinear fusion while reducing the size of fusion features substantially. This research applies the Tensor Sketch projection proposed by Gao et al. (2016) to realise CBP. After CBP fusion operation, a CNN layer with the 1×1 convolution kernel is performed, followed by a Batch Normalization (BN) and ReLU activation to resize the channel number of feature maps. Finally, this research adds the resized feature maps into the visual stream, see Figure 4-4. Assuming interactions are established between the visual stream (x^s_l) and the motion stream (x^t_l), then the visual stream inputs can be formulated as:

$$y_l^s = x_l^s + F([x_l^s, x_l^t], W_l), \quad 4-13$$

where y_l^s is the outputs of l -th layer, and x_l^f represents the fusion feature of x_l^s and x_l^t ; $F(\cdot)$ is the 1×1 convolutional layer.

In the two-stream concurrent CNN model, these two streams have the same network architecture and the same input size, and each pair of layers between the two streams have the same feature map size, such that the two-stream structure can interact with spatial-temporal information at any layer in a concurrent manner. In the preliminary test, this research injects STFBS from the motion stream into the corresponding visual stream for a hybrid spatial-temporal action representation.

4.4.2 Stream Fusion

The visual and motion streams represent different features respectively, and both of them can provide various contextual information alone for intelligent video analysis applications. This research aims to fuse the feature maps of both streams on the last convolutional layer of ResNet-50. It is based on the assumption that different channels

(spatial regions) in the current visual stream are responsible for different visual regions (head, hand, etc.), and different channels in the motion network are responsible for different sizes of motion periods. Thus, the devised fusion model first defines a sum function $f: X^a, X^b \rightarrow y_t$ to aggregate two feature maps $X^a_t \in R^{C \times H \times W}$ and $X^b_t \in R^{C \times H \times W}$ at time t , which then generates a feature map $y_t \in R^{C \times H \times W}$, where W and H denote the width and height of the feature map, respectively; C is the number of channels. This study defined a fusion function $f^{sum}(\ast)$ to sum the feature maps of the last layer of the visual stream and the motion stream, as shown in the following:

$$y^{sum} = f^{sum}(X^a, X^b), \quad 4-14$$

two feature maps are summed across the feature channel d in the same spatial position (i, j) , as shown in the following:

$$y_{i,j,d}^{sum} = x_{i,j,d}^a + x_{i,j,d}^b, \quad 4-15$$

where $X^a, X^b, y \in R^{C \times H \times W}$ and $1 \leq i \leq H, 1 \leq j \leq W, 1 \leq d \leq C$. Because channel numbers are arbitrary, the sum fusion also defines an arbitrary number of connectors, such that subsequent learning can maximize this flexible design and optimizes the filters on all streams. This interwoven structure aggregates the corresponding feature map pairs to combine hierarchical contextual information between appearance information from the visual stream and movement information from the motion stream.

4.4.3 3D CNN Representation

A human action video clip typically contains hundreds if not thousands of frames. Classic two-stream models, including the baseline network model adopted in this research, only take a single RGB frame from the visual stream and ten successive optical flows from the motion stream concurrently. Hence, only short-term spatial-temporal patterns can be encapsulated. These short-term spatial-temporal features can be applied to classify instantaneous actions such as “golf swinging” and “diving” actions in the UCF101 dataset. However, it falls short of handling long duration and complicated actions such as “triple jump” and “TaiChi” movements also coming from the UCF101 dataset.

To enable the learning of long-term semantic event representations from a video clip, this research accumulates visual and motion features of a video across the timeline, as shown in Figure 4-5. The successive video frames are continuously processed by the visual stream, so the feature maps belonging to different frames are obtained. Then, each consecutive feature map is further processed on the corresponding motion stream using the devised OFF layers and residual blocks. The devised network model accumulates the visual (S_i) and motion (M_i) feature maps in time order to obtain the stacked spatial-temporal feature maps. It is then followed by a 3D CNN sub-network for learning long-term semantic event representation and action predictions.

Considering the computational complexity of the whole model, this research only applies one 3D CNN layer and a 3D pooling on the stacked spatial-temporal feature maps. The 3D CNN sub-network can learn the inherent correlations between the highly abstract information of visual and motion streams, hence capturing long-term information on a refined time scale.

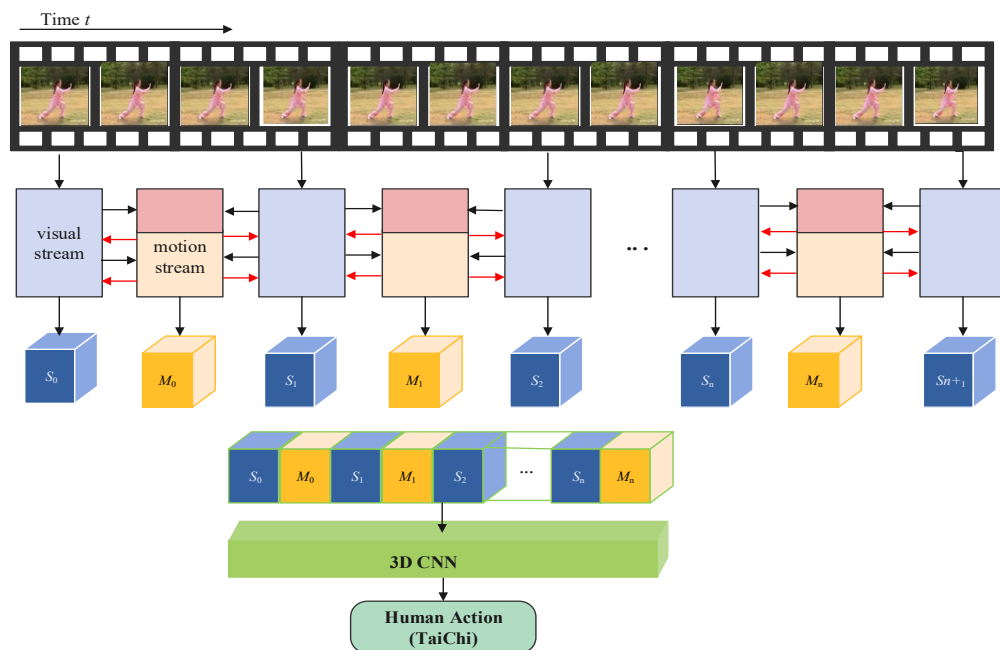


Figure 4-5. Accumulating visual and motion features of a video across time. The successive video frames are processed by the visual stream across the time sequence, and the motion stream processes every two continuous feature maps outputted by the visual stream. The successive visual (S_i) and motion features (M_i) are then accumulated into the stacked spatial-temporal feature maps.

According to Tran et al. (2015), when the domain depth is fixed at 3, the best time domain depth information can be captured; thus, the $3 \times 3 \times 3$ convolution kernel is used

to implement the convolutional fusion for convolving the inputs $X \in R^{H \times W \times T \times D}$ with a sequence of D filters ϕ and the biases $b \in R^n$, as shown in the following:

$$o = \phi * X_i + b, \quad 4-16$$

where ϕ indicates a 3D convolution kernel in size of $3 \times 3 \times 3 \times 2n$, and n is the channel number. After the 3D convolution operation, the 3D max-pooling with the size of $W \times H \times T$ is applied to the stacked data, such that it extends the 2D pooling directly into the time domain.

4.4.4 Network Implementation and Training Strategy

The overall concurrent spatial-temporal network is implemented using PyTorch (Paszke et al., 2019). This research begins with the implementation and training of the visual stream with individual video frames based on ResNet-50, in which all activation functions are ReLU, and the pooling method is Max Pooling. This research initializes the visual stream with the ResNet-50 model pre-trained on the ImageNet to extract visual features. This research removes the fully connected layer and only preserves the convolutional layer since only the outputted feature maps are of interest. For training the visual stream, the batch size is set to 64, the dropout rate is 0.8, and the number of training epochs is 200. It starts with an initial learning rate of 0.001, which decreases to 1/10 at the 20-th epoch and the 40-th epoch.

The parameters of the motion stream are learned by the mini-batch stochastic gradient algorithm through an initial learning rate of 0.02, which is gradually decreased with a factor of 1/10 at the epoch of 9000, 13000 and 16000; the total epoch is 20000. In this stage, the trained visual stream with all the weights frozen is applied to produce visual feature maps, and only the weights of the motion stream should be learned, i.e., the OFF layers calculate spatial and temporal gradients of the feature maps from the visual stream, and the channel number of OFF layers is decreased to 128. Then, several residual blocks are connected in the rear part of the OFF layers at different levels.

This research applies the trained visual and motion models to train the interactive two-stream network based on the devised STFBs. The network is trained with 100

epochs by an initial learning rate of 0.001, which is reduced by a factor of 1/10 at the epoch of 30-th and 50-th. Once the training stage of the STFB-based interactive two-stream network is completed, the feature maps outputted by both visual and motion streams on the last convolutional layer are summed and accumulated. Finally, the 3D sub-network based action representation is trained by using the stacked feature maps, it is followed by the softmax layer to gain the action classification results.

4.5 Learning Optical Flow

Most optical flow estimation methods are derived from a “traditional formulation”, i.e., various assumptions about video frames have been made for solving the optical flow estimation problem, including brightness constancy to spatial smoothness assumptions. As a result, these assumptions cause low accuracy and are less robust. In recent years, researchers proposed a new route that abandons the traditional formulations and assumptions while using CNN architectures to learn flow generation. These works suggest a new direction for developing “learnable” methods for optical flow estimation. This research started with learning optical flow from image pairs by a simple CNN architecture and then improving the model by integrating the advantages from both traditional formulation and recent neural networks.

4.5.1 CNN for Optical Flow Estimation

According to the image classification and semantic segmentation applications, a CNN model should provide an end-to-end manner that directly extracts image (video) features and outputs the results. Computing optical flow has to solve two problems: 1) to estimate the long-range correlation and 2) to compute precise motion boundaries and detailed sub-pixel optical flow (Ranjan & Black, 2017). Ideally, a deep neural network would learn to solve both problems end-to-end. Based on this consideration, this research developed a baseline CNN architecture to learn optical flow, as shown in Figure 4-6, in which all CNN layers use 7×7 convolutional kernels, and it is followed by a ReLU active function. The numbers of feature maps of each layer are: 32, 64, 32, 16 and 2. The output is a 2-channel optical flow. To keep the same size of each feature

map, this study sets stride = 1 and padding = 3 for each convolutional kernel. Two successive video frames (RGB image with 3 channels) are stacked as an image pair (6 channels) before being inputted into a network that outputs a 2-channel optical flow image.

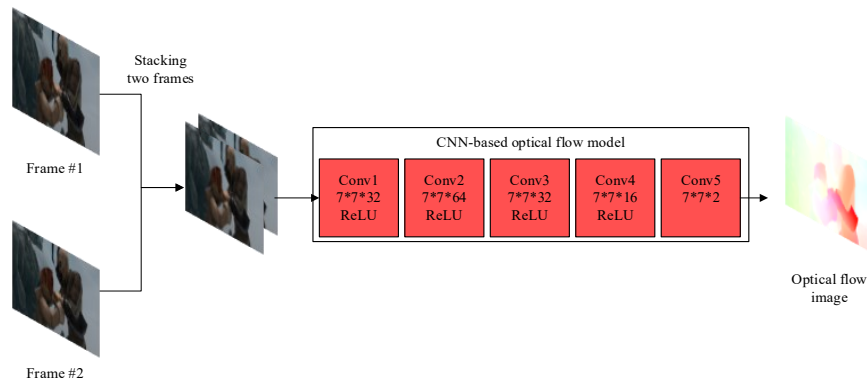


Figure 4-6. The baseline network design of CNN-based optical flow estimation.

This research trained the network on the Flying Chairs dataset that is a simulated video dataset with the ground-truth optical flow (Dosovitskiy et al., 2015). The Adam learning algorithm and mean squared error loss function is applied to train the network. The learning rate is initialized to 0.004, and the total epoch is formed to 100. After the training stage is finished, the CNN model can generate flow fields from the input frame pairs, as shown in Figure 4-8. It is clear that CNNs have the capacity to learn optical flow from image pairs and the ground-truth data by a supervised learning scheme. However, it is impossible to solve the challenge with a simple convolutional network.

4.5.2 Spatial Pyramid Networks

The original CNN model fails to provide an accurate optical flow solution, and one of the main reasons is that the network performs a weak ability to solve the short- and long-range correlations (one of the optical flow estimation problems) since the network only extracts image features in a single spatial scale. To tackle this problem, Dosovitskiy et al. (2015) presented a so-called FlowNet model that learns spatial-temporal filters for optical flow estimation by using CNNs. The motion information is first spatially compressed in a contractive part of the network, and the refinement part is applied to refine the coarse feature maps (low-resolution flows) to the high-resolution optical flow prediction. FlowNet and its improved model - FlowNet 2.0 (Ilg et al., 2017),

show promising results. However, the computational performance is relatively low since the complex network design and a lot of parameters, and it fails to support real-time applications, especially for developing embedded and mobile applications. To tackle this shortage, Ranjan and Black (2017) presented the so-called SpyNet model by combining a traditional spatial pyramid, image warping and tidy convolutional neural networks. Inspired by this original work, this research developed a pyramid and deep learning based model for coarse-to-fine optical flow estimation. A 3-level structure of this approach is shown in Figure 4-7, in which the two input frames (I^1 and I^2) are down-sampled into three pyramid levels. Each level trains a CNN model (G) from low-level to high using the images at corresponding pyramid levels and the up-sampled flow from its preview level.

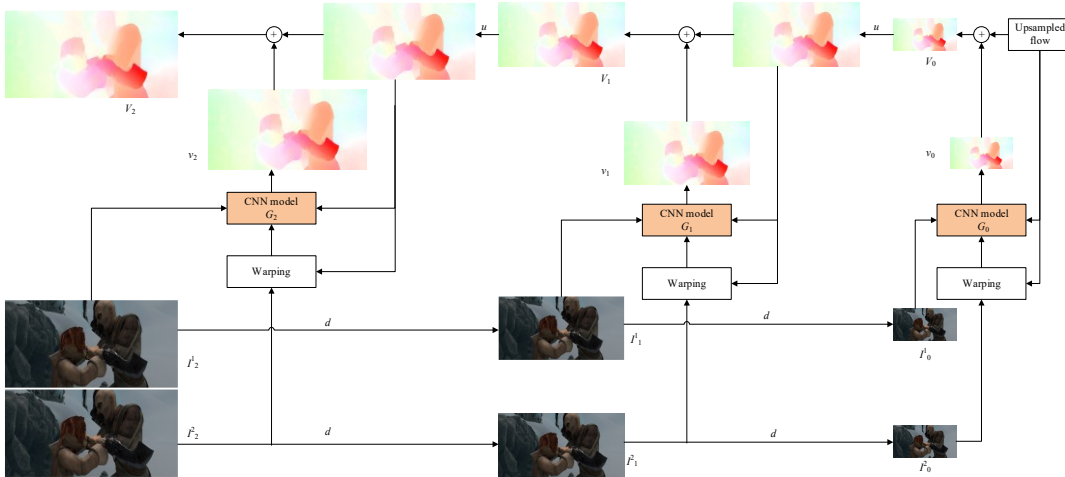


Figure 4-7. The structure of a 3-level pyramid network.

a) Spatial Pyramid Sampling

Let I be a video frame with $m \times n$ pixels in size that are powers of 2. Let $d(x)$ be a down-sampling function with the factor of 2, i.e., the output $d(I)$ is a low-resolution image with $(m/2) \times (n/2)$ pixels in size. Let $u(x)$ be an up-sampling function with the factor of 2, i.e., the output of $u(I)$ is a high-resolution image with $(m \times 2) \times (n \times 2)$ pixels in size. In Figure 4-7, I_k^t indicates the t -th video frame at k -th spatial pyramid level; v_k is the residual flow at k -th spatial pyramid level while V_i is the corresponding optical flow. Accordingly, suppose the number of pyramid levels is K , then I_K^1 and I_K^2 are the raw video frames, while V_k is the full-resolution optical flow that is the target of the pyramid network.

b) Pyramid Networks

Let $\{G_0, \dots, G_K\}$ are K trained CNN optical flow estimation models (e.g., the baseline model, see Figure 4-6). Each CNN model learns the residual flow at the corresponding pyramid level, as shown in the following:

$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1})), \quad 4-17$$

where v_k is the residual flow at k -th spatial pyramid level. The CNN model G_k is used to learn v_k from video frames I_k^1 and I_k^2 , and the up-sampled optical flow $u(V_{k-1})$ from the previous pyramid level. The $w(I_k^2, u(V_{k-1}))$ is a standard warping function that warps the second video frame and the up-sampled optical flow, and the warping operator comes from the traditional approach for optical flow estimation (Brox et al., 2004), which outputs a 3-channel (RGB) image. After v_k is obtained, the optical flow at k -th level V_k is then computed by the following:

$$V_k = u(V_{k-1}) + v_k. \quad 4-18$$

This method repeats the operations (warping, CNN model and up-sampling) from the low-pyramid level to the high, as shown in Figure 4-7. It is worth noting that the zero-level optical flow is initialized to zero to compute the optical flow $V_0 = 0 + v_0$.

In practice, this research implements a 5-level pyramid ($K = 4$) and trains each CNN model $\{G_0, \dots, G_K\}$ independently at the corresponding spatial pyramid level. The previous optical flow output from the low level is applied to train the next-level CNN model. Therefore, this structure has the capacity of a coarse-to-fine scheme for estimating optical flow. Figure 4-8 demonstrates the flow images computed by the pyramid networks. Compared to the simple CNN model, the pyramid network-based model performs with better accuracy and robustness.

4.6 Experimental Results

4.6.1 Visualisation of Feature Maps

For qualitative analysis of the DNN models, this experiment extracts and visualises some feature maps from different CNN layers in the vision and motion streams. Two consecutive frames are extracted from a video that performs the “TaiChi”

action coming from the UCF dataset. The frame is then inputted into the visual stream, while the corresponding optical flow image is fed into the motion stream. The original images and feature maps of Conv1, Conv2 and Conv5 of the two-stream network are visualised in Figure 4-9, where the bottom-up “appearance” features are extracted automatically. For example, the human body can be easily seen from the feature maps of Conv1, whereas Conv5 describes more abstract information. Meanwhile, the temporal stream also encodes the optical flow into “high-level” motion information.

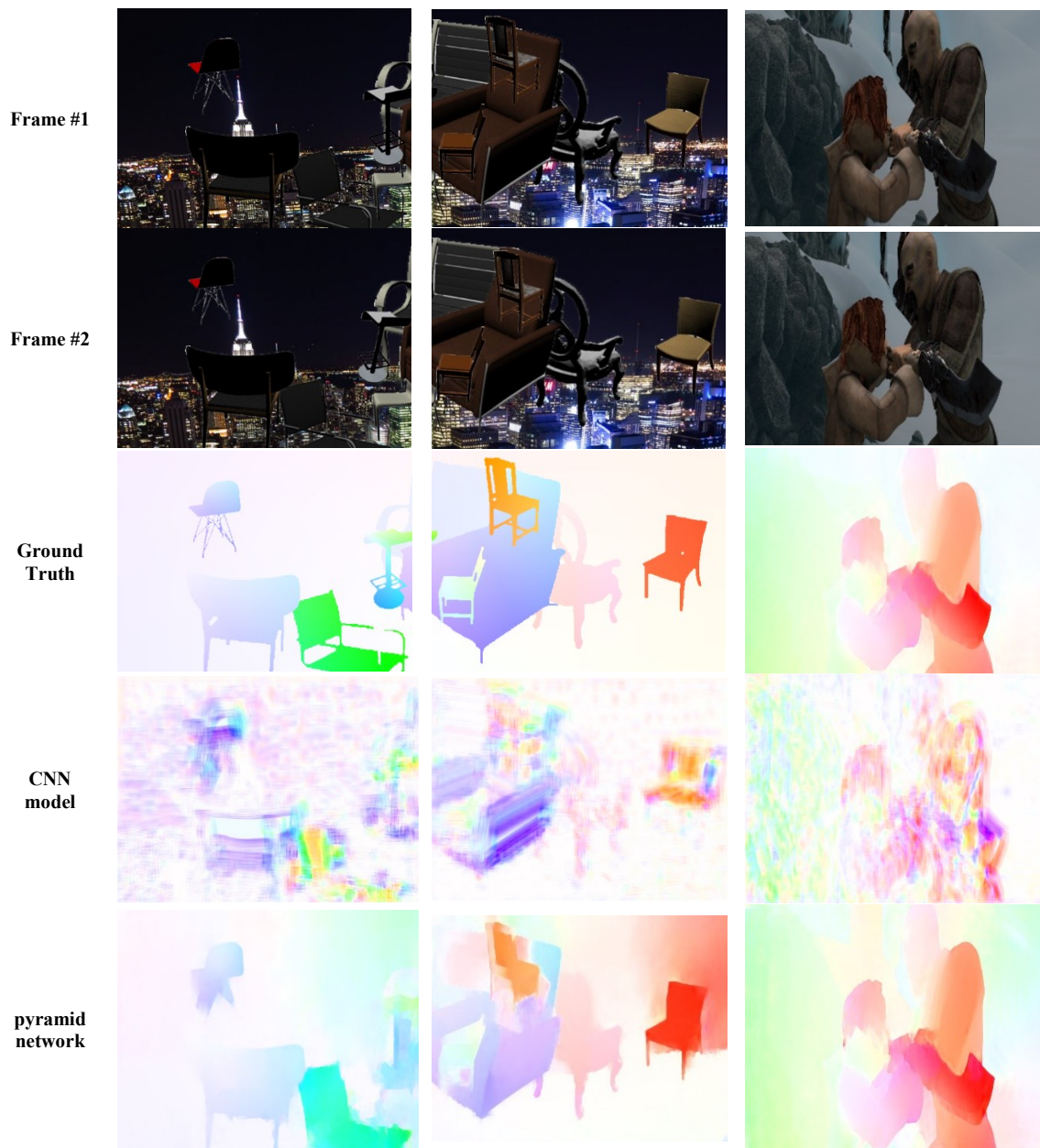


Figure 4-8. Visualization of optical flows estimation methods

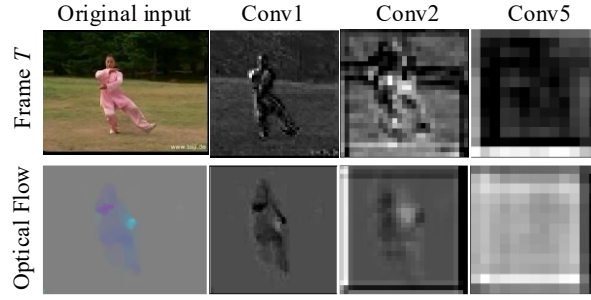


Figure 4-9. Feature maps extracted from the “TaiChi” action video in the UCF dataset.

4.6.2 Comparison of Pre-trained DNNs

This experiment examines different CNN architectures in the dual-stream deep learning models to identify a suitable one for the devised framework. Four popular CNN models for image classification were implemented to extract video features, namely, AlexNet (Krizhevsky et al., 2012), VGG-16, VGG-19 (Simonyan & Zisserman, 2015), and C3D network (Tran et al., 2015). The former three CNN models are pre-trained by the ImageNet image classification dataset (Krizhevsky et al., 2017), and the C3D was trained by the UCF 101 dataset. Then, the FC and softmax layers are applied for action classification. The UCF 50 dataset was used to test these implementations. It is clearly shown in Table 4-2 that the accuracies of AlexNet for both streams are lower than the VGG models, while the performance of VGG-16 is identical to VGG-19. However, VGG-19 requires more computational resources than VGG-16 due to its extra network depth. Hence, in this research, parameters from the pre-trained VGG-16 are inherited as the generic learned feature extractor to achieve the best accuracy-cost trade-offs. It is worth noting that the performance improvement is significant when adapting the C3D network in the visual stream, and the main reason is that the 3D CNN used in C3D is more effective when extracting spatial-temporal features from STV data. However, the accuracy does not improve when adopting the C3D network in the motion stream; one of the main factors is that C3D mainly focuses on capturing high-level abstract and semantic information from RGB video clips, while optical flow only abstracts motion information. According to the performance comparison and processing time, this research has adopted the C3D and VGG-16 networks to implement the transferred feature extractors.

This experiment compared the dual-stream model with the individual stream settings (i.e., using either visual or motion stream), and the result is shown in Table 4-2. Unsurprisingly, the dual-stream model performed consistently better than the single-stream settings. According to the experimental result, this research adopted the C3D and VGG-16 configurations in the dual-stream network for further experiments.

Table 4-2. The recognition accuracy of different CNN in the dual-stream deep learning architecture on the UCF 50 dataset.

Deep feature model		Accuracy (%)
Visual stream	Motion stream	
AlexNet	AlexNet	76.4
VGG-16	VGG-16	85.6
VGG-19	VGG-19	85.8
C3D	VGG-16	89.6
C3D	VGG-19	89.8
C3D	C3D	86.4
C3D	-	85.2
-	VGG-16	79.5

Table 4-3. The comparison results of OFF and baseline two-stream networks.

Method	Speed (fps)	Accuracy (%)
Baseline visual stream (RGB)	268	82.3
Baseline motion stream (Optical Flow)	30	79.1
Baseline two-stream (RGB + Optical Flow)	14	87.6
OFF-based two-stream (RGB)	203	90.6

Table 4-4. The classification results for STFB integration into different network locations.

Network setting	STFB location		
	Visual (%)	Motion (%)	Two-stream (%)
Visual stream	91.9	87.1	90.8
Motion stream	91.2	83.4	89.7
Two streams	93.5	89.8	91.7

4.6.3 OFF Efficiency

As an ablation exploration, the performance of OFF under the proposed network architecture is evaluated using the UCF101 dataset. For a fair comparison, the OFF-based models are trained and tested using the same ResNet-50 configurations described in Section 4.3.2. This experiment evaluates the processing speeds and accuracies between the baseline and the OFF-based two-stream implementations. The results are listed in Table 4-3. When applying OFF in motion stream, 90.6% of the competitive accuracy can be obtained on UCF101 by using RGB frames as inputs. This result is comparable with most two-stream based approaches (Simonyan & Zisserman, 2014). More significantly, the OFF-derived two-stream network is more effective as it can run

on over 200 FPS which is almost the same as the visual stream in the baseline network. In comparison, the baseline two-stream network model only achieves 14 FPS due to the computation for optical flows, which occupies over 90% of processing time, and the motion stream in the baseline network only archives 30 FPS. These experimental results indicate that the OFF application in the motion stream can effectively improve the human action recognition rate by reducing computational workload.

4.6.4 STFB Location

This experiment evaluates the STFB effectiveness by inserting STFB constructs at different locations of the two-stream network. As can be seen in Table 4-4, when inserting STFBs into the visual stream, the performance has a significant improvement, and this indicates that the STFB can augment significant movement information in the motion stream to the corresponding object features in the visual stream, such that fine spatial-temporal interaction representation can be extracted. By contrast, when inserting STFBs into the motion stream or both streams, the performance appears inefficient, and the accuracies are even inferior to the pure visual stream integration. One of the main contributing factors is that the pure motion stream STFB integration makes it dominant in the whole architecture, which helps to eliminate the side effects of the visual stream. Therefore, this experiment concluded that the additive spatial and temporal interactions in the visual stream could improve the performance of human action recognition. The effectiveness of pure visual stream STFB integration is superior to the pure motion stream one or two-stream integration.

4.6.5 Numbers of STFB

This experiment further explores the effect of varied numbers of STFB by inserting multiple STFBs into different layers of the residual models in the visual stream. Compared with inserting only one STFB in the first layer, the accuracy can be improved by 1.4% when inserting STFBs in the first three layers. However, the accuracy dropped by 1.1% when inserting STFB into all layers, see the results in Table 4-5. The main

reason is that high-level features extracted by Conv5_x are sparse and lack correlation. Thus, this research inserts STFBs into the first three layers for the rest of the work.

4.6.6 Evaluation of 3D Sub-network

Another benchmarking study compares the influence of the 3D CNN sub-network for action representation. This experiment tests individual streams, i.e., the visual stream and the motion stream, and the combined two-stream network with a 3D sub-network. It is clearly illustrated in Table 4-6 that simply appending a 3D sub-network in the visual stream has limited performance gain. The main reason is that although 3D convolution can capture the object appearance information, it has a certain extent of overlap with the corresponding visual network. By contrast, adding the 3D sub-network in the motion network can greatly improve the performance due to its capacity to learn the motion interactions across time from the continuous streamed frames. Therefore, the 3D sub-network generates richer dynamic information that has a significant effect on performance improvement. Moreover, the combination of the 3D sub-network and the two-stream network boosted outstanding performance due to the concurrent interactive feature extraction ability from the two-stream structure and the high-level semantic representation advantage from the 3D CNN model.

Table 4-5. The accuracy (in %) of different number of STFBs insertion on UCF-101.

Insert position	Accuracy
Conv2_x	92.1
Conv2_x, Conv3_x	92.7
Conv2_x, Conv3_x, Conv4_x	93.5
Conv2_x, Conv3_x, Conv4_x, Conv5_x	92.4

Table 4-6. Comparison of various streams in combination with a 3D sub-network (in %).

Model	UCF101		HMDB51	
	Baseline network	With 3D sub-network	Baseline network	With 3D sub-network
Visual stream	75.5	78.4	48.4	50.2
Motion stream	86.8	89.4	58.5	59.6
Two-stream	92.9	93.7	65.4	66.9

4.6.7 Comparison With the State-of-the-art Results

This experiment has compared the proposed two-stream aggregation model with the state-of-the-art approaches on UCF101 and HMDB51 datasets, including iDT

(Wang & Schmid, 2013), C3D (Tran et al., 2015), MTC3D (Lu et al., 2017), Factozed spatio-temporal convolutional networks (FstCN) (Sun et al., 2015), trajectory-pooled deep-convolutional descriptors (TDD) (Wang et al., 2015), ST-ResNets (Feichtenhofer et al., 2016), key volume mining framework (KVMF) (Zhu et al., 2016), the two-stream model (Simonyan & Zisserman, 2014) and its improved methods, namely a few, two-stream with LSTM (Gammulle et al., 2017), hidden two-stream (Zhu et al., 2019), two-in-one stream (Zhao & Snoek, 2019) and C²LSTM (Majd & Safabakhsh, 2020). The experimental results shown in Table 4-7 show that the recognition rate of the proposed aggregation model on UCF101 and HMDB51 are 94.6% and 67.5%, respectively, which is better than the current handcrafted and deep learning methods. The superior performance stems from the effective OFF-based two-stream network and the coarse-to-fine joint between spatial and temporal dimensions. It is also contributed by the long-term semantic action representation ability of the integrated 3D CNN sub-network.

Table 4-7. Performance comparison between the proposed aggregation model with other state-of-the-art methods on UCF101 and HMDB51 datasets.

Method	UCF101 (%)	HMDB51 (%)
IDT (2013)	86.4	61.7
C3D (2015)	85.2	NA
FstCN (2015)	88.1	59.1
TDD (2015)	90.3	63.2
Two-stream (2014)	88.0	59.4
Two-stream + LSTM (2017)	88.6	NA
Hidden Two-stream (2017)	90.3	60.5
two-in-one stream (2019)	92.8	NA
C ² LSTM (2020)	92.8	61.3
MTC3D (2019)	90.1	64.5
KVMF (2016)	93.3	63.3
ST-ResNets (2016)	93.4	66.4
The proposed aggregation model	94.6	67.5

4.6.8 Applicability and Extensibility

To investigate and evaluate the generalisation of the proposed hybrid model, this research also tested extended human action categories such as those depicted in UT-Interaction dataset that mainly focuses on human-human interactions (Ryoo & Aggarwal, 2010). The same configuration settings described in Section 5.3 have been adopted for the test. Since videos in this series (set1 and set2) contain combinatory actions, segmented datasets were deployed in this experiment. Compared to the BoF

(Ryoo, 2011) with the deep representation proposed by Lee and Lee (2019), the devised framework demonstrates the robustness and greatly extended applicability to complex human interactions evidenced by the state-of-the-art performances shown in Table 4-8. In conclusion, with the holistic features and coarse-to-fine interactions, the proposed models have gained significant performance advancements in both human actions and interactions with convincing promise on crowd action understanding.

Table 4-8. Extensibility on UT-Interaction dataset.

Method	Set #1 (%)	Set #2 (%)
BoF	81.67	80.00
deep representation	90.22	89.40
Dual-stream model	91.35	91.50
Aggregation model	93.26	93.45

4.7 Summary

In this chapter, the DNN models for the end-to-end video feature extraction and event prediction have been introduced. Then, this research devised a two-stream concurrent interactive network model by exploring innovative techniques, including OFF layers, STFB blocks, and 3D CNN for action representation. The breakthroughs of this innovation include: 1) the use of OFFs in the motion stream to replace the time-consuming optical flow computation with proven promising results; 2) the innovation of STFB constructs to build compact fusion representations for spatial and temporal feature interactions; 3) and the long-term semantic event representation is enabled by the 3D CNN sub-network. Another important finding is that different fusion locations have significant and varied contributions to the final action classification outcome. It has been evaluated to identify optimal settings by inserting different amounts of STFBs on different feature levels. The devised concurrent spatial-temporal aggregation model shows better performance than the state-of-the-art action recognition methods.

As a portion of the research, the CNN and spatial pyramid based optical flow approach with a supervised learned style has been investigated, which shows good robust and effective, suggesting a new direction of optical flow estimation by learning algorithms and combing the engineered architectures.

CHAPTER 5 Towards Understanding Human Actions

5.1 Introduction

Human action recognition has achieved competitive performance on various benchmarks because of the advancement of DNN and the large-scale training datasets (Chen et al., 2019; Jiang et al., 2021; Simonyan & Zisserman, 2014; Tran et al., 2015; Xu et al., 2019a; Zhao & Snoek, 2019). However, these approaches tend to model static contexts such as objects and scenes instead of interpreting human actions based on their semantic definitions. Taking the two-stream model as an example, its spatial stream achieves 73% accuracy on the UCF 101 dataset, while the accuracy only increases by 13.9% when fusing the temporal stream (Simonyan & Zisserman, 2014). Considering the spatial stream extracts appearance information from frames only, the performance improvement is not significant from the temporal stream that encodes motion information. Moreover, the two-stream model only gains 58.0% accuracy on the HMDB 51 dataset, which is far less than UCF 101. One of the main reasons is that the backgrounds and scenes in HMDB 51 videos are very complex and diverse, while the spatial stream focuses on learning static scenarios and it is less robust in handling various contexts, resulting in a context-biased model that fails to generalise (Bahng et al., 2020). For example, the model tends to predict the “shooting goal” result on football field background videos. However, actions may be occurred in a misleading context or even missing content. As shown in Figure 5-1, in the first video, the football players are mimicking a “bowling” action on a football field, and the last video is a mime performance where a mime artist is mimicking the “drinking” action with a black background and an “imaginary” bottle.

To evaluate the impact of biases on action recognition, this research measures the recent DNN models on the revised datasets by masking all humans in the videos. According to the evaluation, the DNN models have still gained a very high performance when considering they have never detected any human in the testing videos. The results show that the predictions mainly rely on the objects and scenes instead of the semantic

definitions of human actions, and the human movements are not correctly interpreted by the models. In contrast, humans have a more robust and intelligent vision system to recognise actions. For instance, a person who has never seen mime performances can still understand the mime actions from the body language given by actors while the objects and scenes are absent. Therefore, although the scene and object information are important, a robust intelligent vision system should also be capable of extracting the fundamental meanings of various actions, even in the absence of contextual information.



Figure 5-1. Examples of misleading and absent contexts. First row: the players are mimicking a “bowling” action on a football field; the second is a mime performance mimicking a “drinking” action without using any objects.

Human pose skeletons leverage a high-level body language which is not affected by circumstances and backgrounds, and it explicitly exploits the spatial shapes and relationships of the human joints (Sun et al., 2019). Consequently, encoding the semantic representations and understanding the true human actions based on pose skeletons have become the upcoming frontiers and received increasing attention recently (Gupta et al., 2021a; Shi et al., 2019; Yan et al., 2018). However, the performance has yet to be improved when considering the limited performance gain on the dataset-specific action models, and it is less robust on generic videos. It is still an open challenge to understand the semantic action representations, which is especially true when facing the unseen actions in videos. To tackle this problem, this research presents a long-short-term semantic motion encoding (LSME) method to abstract the high-level action representation from pose skeleton sequences. Furthermore, a novel method is proposed to recognise unseen actions for real-world applications, where a large-scale training dataset is unavailable. Experiments show that the proposed model achieved better performance on human action recognition. Further, this method can predict new actions which are never occurred in the training set.

5.2 Understanding the Biases for Action Recognition

5.2.1 Human Masked Data Processing

To understand which biased weights are leveraged by the spatial-temporal CNN based action recognition models, this research firstly revises the action datasets by masking out humans in videos. To do that, humans are detected in each video frame by using the Faster R-CNN algorithm which is a general object detection model (Ren et al., 2017). Only the “human” label is extracted in the detect results, and the bounding boxes are tracked over continuous frames by matching the highest Intersection-over-Union (IoU) score between the current and next frames and performing linear interpolation in the missing frames. The tracking is stopped if there is no match during 10 continuous frames and starting a new trajectory for the next bounding box. Finally, all human tubes are masked by colouring them grey. These videos are called masked action datasets. Examples coming from Kinetics-400 are shown in Figure 5-2. These videos are considered a no-action dataset because there is no human in the videos.



Figure 5-2. Examples of the human-masked-out video frames. The top shows the original frames, while the bottom shows the corresponding masked frames.

5.2.2 Biased Models in Action Recognition

This research measures the problem of biased weights on four DNN models, i.e., R(2+1)D (Tran et al., 2018), X3D (Feichtenhofer, 2020), TIN (Shao et al., 2020), and TimeSformer (Bertasius et al., 2021) models. These models are tested by only using RGB frames and without any other modality fusion and evaluated the top-1 and top-5 accuracies, respectively, on the Kinetics-400 dataset. To understand which bias is learned by the models, the pre-trained models are applied to the standard training set

and tested on the masked test set. From Table 5-1, the R(2+1)D model yields 57.47% top-1 accuracy and 78.02% top-5 accuracy, respectively, on the masked test videos. Its top-1 accuracy only drops by 9.54% compared to the test result on the original test set, while the top-5 accuracy only decreases by 8.82%. The results of other models show a similar situation. Overall, the top-1 accuracies of all models on masked data are over 50%, and the top-5 accuracies are extremely high when considering the models have never detected humans in test videos. This observation proves that these models classify human actions primarily by modelling static contextual information instead of interpreting human actions based on their semantic definitions.

To better understand the context-bias problem, the accuracy change of the pre-trained TimeSformer model (Bertasius et al., 2021) is evaluated by calculating the accuracy of each action in original and masked settings, respectively. As shown in Figure 5-3, 27 categories of actions increase their accuracy when masking out the humans at the testing stage, which counts for 6.75% of total actions, and about 4.75% of action classes keep the same accuracies as the original setting. Moreover, 176 categories of actions decrease their accuracy by less than 15%, which counts for 44% of total actions.

Table 5-1. Accuracy (in%) when testing the pre-trained DNN models on the Kinetics-400 dataset by using the original videos and masked videos, respectively.

Model	Top-1			Top-5		
	Original data	Masked data	Diff.	Original data	Masked data	Diff.
R(2+1)D	67.01	57.47	-9.54	86.84	78.02	-8.82
X3D-S	70.68	52.66	-18.02	89.45	73.4	-16.05
TIN	69.55	51.23	-18.32	88.92	71.38	-17.54
TimeSformer	74.25	59.17	-15.08	91.75	77.86	-13.89

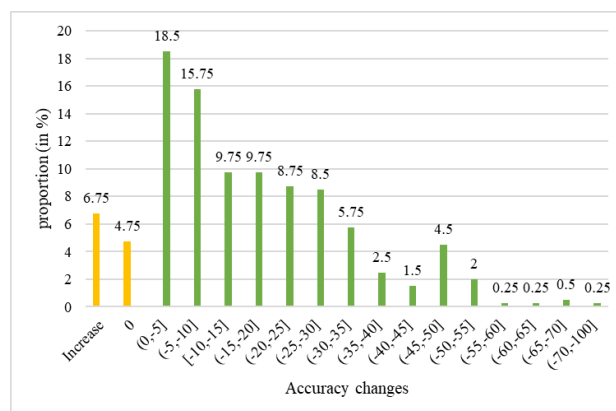


Figure 5-3. The proportions of the accuracy change per class action.

Table 5-2. Classes with the increased accuracy (in %) on the original training set and tested on original and masked Kinetics-400 settings.

Class (27)	Original data	Masked data	Diff.
shooting basketball	28.87	40.21	11.34
cleaning floor	54.64	62.89	8.25
triple jump	50.52	58.76	8.25
recording music	49.48	55.67	6.19
cleaning gutters	81.82	87.88	6.06
driving car	76.40	79.78	3.37
trimming trees	69.47	72.63	3.16
cooking on campfire	70.53	73.68	3.16
playing cards	71.88	75.00	3.13
cleaning windows	68.37	71.43	3.06
bobsledding	64.77	67.05	2.27
faceplanting	18.95	21.05	2.11
dunking basketball	67.01	69.07	2.06
garbage collecting	78.35	80.41	2.06
springboard diving	85.57	87.63	2.06
grooming horse	85.71	87.76	2.04
kicking field goal	85.71	87.76	2.04
training dog	68.37	70.41	2.04
sailing	84.69	86.73	2.04
diving cliff	94.95	96.97	2.02
using remote controller (not gaming)	77.08	78.13	1.04
cooking sausages	61.86	62.89	1.03
surfing crowd	90.72	91.75	1.03
swimming butterfly stroke	71.43	72.45	1.02
changing oil	96.94	97.96	1.02
building cabinet	86.87	87.88	1.01
strumming guitar	46.46	47.47	1.01

This research then shows the classes with increased accuracy on the masked testing set. From Table 5-2, it can be seen that several actions are very dependent on props and venues, such as “cleaning floor”, “cleaning gutters”, “bobsledding”, and “cooking sausages”; thus, they achieved higher accuracy than the original data which seems like the “human noise” is removed when masking out the human areas. Moreover, other actions, which mainly depend on body movements, also increase the accuracies in the masked set. For instance, the “shooting basketball” action dramatically increases its accuracy on masked data by 11.34%.

5.2.3 Analysis and Discussion

Most DNN models tend to learn static contextual information such as objects and scenes, and the context-biased weights benefit from the masked videos. Although the recognition accuracy is quite high, these models do not differentiate humans from other contextual settings in the videos. Therefore, a robust intelligent action recognition

model should be capable of understanding the fundamental meanings of various actions, even in the absence of context information. To tackle this challenge, this research presents a semantic encoding algorithm based on pose skeletons to encode semantic action definition and understand human actions and then introduces a novel method for unseen action recognition in the test data.

5.3 Encoding Semantic Human Actions

This research presents LSME based on pose skeletons. The framework of the proposed method is shown in Figure 5-4, where humans are detected by the Faster R-CNN object detector before a HRNet-based pose estimation is performed to extract human pose from each video frame, followed by an IoU-based tracker to obtain human pose sequences across over an entire video. Then a 3D CNN sub-network is designed to encode short-term spatial-temporal features from the stacked 3D pose heatmap volume before a 2D temporal convolution network (TCN) is developed for modelling long-term semantic action representation, hence generating an encoded action representation for human action understanding in the wild.

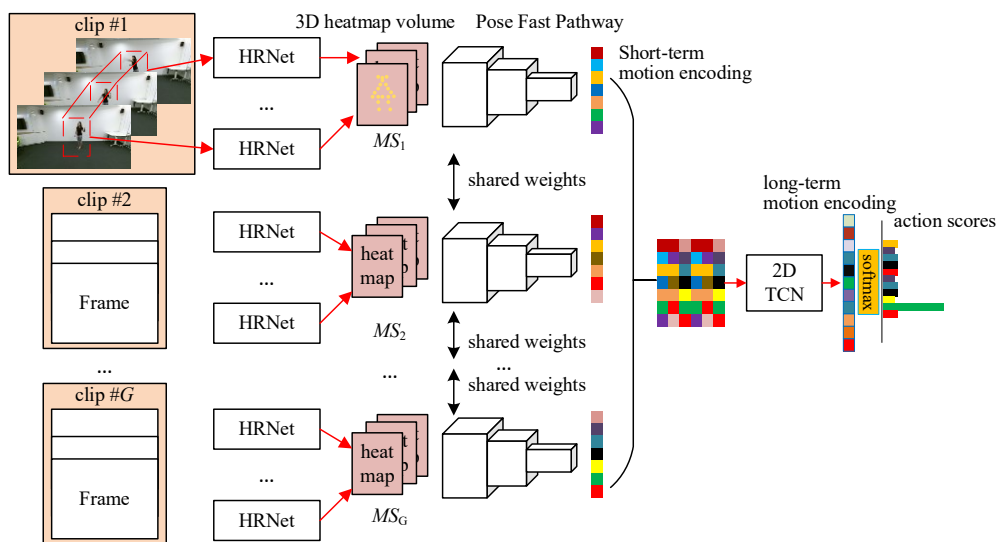


Figure 5-4. The proposed long-short-term semantic motion encoding architecture for human action understanding.

5.3.1 Human Pose Sequence Extraction

Human pose skeletons fundamentally determine the action representation and recognition performance. Human poses can be captured correctly by specific sensors.

However, this solution is neither affordable nor deployable in real applications. Pose estimation is an alternative method for extracting human poses from RGB frames (Cao et al., 2021; Kocabas et al., 2020; Sun et al., 2019). This research uses 2D poses instead of 3D poses because of the unstableness and computational complexity of existing 3D pose estimators. In contrast, the top-down 2D pose estimation approach shows outstanding performance and effectiveness, and these advantages are still preserved when applied for action recognition. To encode poses over time in a video, it needs to detect and track each individual and then obtain human pose sequences. To do that, the Faster R-CNN algorithm (Ren et al., 2017) is firstly performed and filtering only the “human” label to obtain the bounding boxes in every frame. Then the human boxes are tracked in the following frames based on the IoU threshold, i.e., matching the detections from the highest score of IoU which is over 0.3. The linear interpolation is also applied in the missing frames. It stops the tracker if there is no match during 10 continuous frames, hence obtaining a human sequence which performs an individual action.

Then HRNet (Sun et al., 2019) trained on the COCO-keypoints dataset is performed to detect poses for all frames in a human tube, hence obtaining a human pose sequence labelled by the corresponding action class. Noted that various pose estimation approaches can be used in this process, and the pose quality will sensitively influence the final recognition accuracy. Based on this aspect, HRNet is a good solution for pose estimation because of its superior results over benchmarks. Moreover, it maintains high-resolution representations during the entire process for more accurate and spatially precise heatmaps (Sun et al., 2019). This advantage is preserved for spatial-temporal feature extraction when facing human action understanding tasks.

5.3.2 3D Pose Heatmap

Once human poses are extracted, the skeletons can be represented by sequence vectors, pseudo-images, and spatial-temporal graphs, for learning action patterns through recurrent networks (Shahroudy et al., 2016), convolutional networks (Liu et al., 2017), and GCNs (Shi et al., 2019; Yan et al., 2018), respectively. These approaches depend on handcrafted formulations or specific-designed convolutional kernels, which

are complex and inefficacy. On the other hand, the heatmaps from the pose estimation models at the last convolutional layer should implicitly contain pose representation. These heatmaps could carry information about both 2D and 3D poses, according to the corresponding pose approaches. Inspired by this prospect, this research transfers the heatmaps as mid-level features for further event encoding instead of using explicit joint coordinates. Here, for a given heatmap, it is defined by $M \in \mathbb{R}^{K \times H \times W}$, while K is the number of joints, H and W indicate the height and width of the frame, respectively. This research accumulates a sequence of heatmaps over T continues frames as a 3D heatmap volume, defined as $MS \in \mathbb{R}^{K \times T \times H \times W}$, and a whole video can be divided several groups of 3D heatmap volumes, i.e., $MS \in MV \in \mathbb{R}^{G \times K \times T \times H \times W}$, where G is the number of groups. This research directly uses the heatmap M assigned to the corresponding bounding box location, and it is zero-padded to fit the original size of the frame. It is worth noting that this process assumes a single person setting, but the multi-person case can be extended by repeating the heatmap extraction for each human tube and matching the locations according to the corresponding human bounding boxes of all detections.

In the cases of only joint coordinates of skeletons are given, or for storage saving, since storing such large heatmaps requires a great deal of storage, the heatmap M can be reconstructed by performing a K Gaussian blob for every joint (Cao et al., 2021), as shown in the following:

$$M_k(x, y) = \exp\left(-\frac{(x-x_k)^2 + (y-y_k)^2}{2 * \sigma^2}\right) * c_k, \quad 5-1$$

where x_k and y_k are the coordinates of k -th joint, c_k is the confidence score of the corresponding joint, and σ is set to 0.5 which controls the variance of gaussian maps.

5.3.3 Long-short-term Learning Strategy

As shown in Figure 5-4, all heatmaps from an entire video are grouped into G heatmap volumes $MV \in \mathbb{R}^{G \times K \times T \times H \times W}$, and then each one is fed into the 3D CNN sub-network sequentially. The learnable parameters of 3D CNN are shared on all 3D heatmap volumes, and the long-term semantic sequence encoder fuses the feature maps at the last convolutional layer to yield a video-level prediction. Unlike the previous works, which normally randomly select a fixed length of frames for training and

updating weights in a clip-level or single-frame-level gradients, this research trains the 3D CNN on an entire video and updates weights in the video-level gradients.

In this design, the short-term spatial-temporal parts are extracted by a 3D CNN sub-network, while the long-term semantic information is characterized by a semantic sequencing encoding method to represent the overall pose skeletons in a video. A whole video heatmap volumes MV is then divided into G heatmap volumes $MV = \{MS_1, MS_2, MS_3, \dots, MS_G\}$ ordered by time. Then, G feature maps are obtained by performing the 3D CNN sub-network on each 3D heatmap volume, followed by an aggregating model to encode video-level features. The whole process is formulated as the following:

$$Y_v = Q(F(MS_1; W); F(MS_2; W); \dots; F(MS_G; W)), \quad 5-2$$

where $F(MS_i; W)$ represents the 3D CNN sub-network with the shared weights W and the heatmap volume MS_i , and $Q(X)$ is the aggregating method which generates the final class score Y_v .

The differentiability of the temporally aggregating method allows for updating the 3D CNN parameters by using backpropagation by extending the standard cross-entropy loss function on whole parts, formulated in the following:

$$L(y, Y) = \sum_{i=1}^N y_i (Y_i - \log \sum_{j=1}^G \exp Y_j), \quad 5-3$$

where N indicates the number of classes and y_i is the ground-truth label of i -th class. Based on the loss function, the gradients of the weights on the 3D CNN sub-network can be calculated by the following:

$$\frac{\partial L(y, Y)}{\partial W} = \frac{\partial L}{\partial W} \sum_{g=1}^G \frac{\partial Y}{\partial F(MS_g)} \frac{\partial F(MS_g)}{\partial W}. \quad 5-4$$

In this optimization, the parameters are updated through the global differentiability derived from all 3D heatmap volumes.

5.3.4 Short-term Semantic Motion Encoder

Considering an action may be performed very fast, e.g., the “shooting goal” action may occur in less than one second, and the body movements of the “dancing” action are changed very quickly. However, previous models that sample frames by random steps will lose the fine temporal information. Such a network should maintain a high

temporal rate to effectively model the inherently fast-changing movement. Therefore, the architecture of the devised 3D CNN sub-network is inspired by the Fast pathway of SlowFast (Feichtenhofer et al., 2019) which achieved competitive results in RGB-based action recognition. The Fast pathway has a fine representation along with the temporal dimension by its high frame rate ratio and high temporal resolution features. The architecture of the 3D CNN sub-network is shown in Table 5-3. Compared to the original Fast pathway (Feichtenhofer et al., 2019), this research introduces three improvements for better adopting pose skeleton data: 1) the smallest temporal stride $\tau = 1$ is used, i.e., all heatmaps are used without sampling, which keeps the finest motion information for fast actions; 2) the Pose Fast pathway has no down-sampling layers to maintain a high resolution of feature maps; 3) the res2 layer is removed due to the 3D heatmap volumes are already considered mid-level features for event representation. The Pose Fast pathway is very lightweight because it is designed to have fewer feature channels (the **green** colour numbers in Table 5-3) because this network primarily focuses on learning temporal information concerning body joint movements.

Table 5-3. The architectures of the 3D CNN sub-network and spatial fusion network. The 3D CNN model is implemented by the Fast pathway, while the spatial network comes from the Slow pathway. The ResNet50 is used as the backbone.

Stage	Pose Fast pathway	RGB Slow pathway	Output sizes $T \times S^2$
data modality	3D heatmap volume	raw RGB frames	
data layer	stride 1, 1	stride 16, 1	Pose: 3×56^2 RGB: 5×224^2
conv1	$5 \times 7^2, 8$	$1 \times 7^2, 64$	Pose: 3×56^2 RGB: 5×224^2
pool1	N.A.	$1 \times 3^2, \text{max}$	Pose: 3×56^2 RGB: 5×224^2
res2	N.A.	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	Pose: 3×56^2 RGB: 4×56^2
res3	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	Pose: 3×28^2 RGB: 5×28^2
res4	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 3^2, 128 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 3^2, 1024 \end{bmatrix} \times 6$	Pose: 3×14^2 RGB: 5×14^2
res5	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 3^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 3^2, 2048 \end{bmatrix} \times 3$	Pose: 3×7^2 RGB: 5×7^2
	Global average pool		
	TCN	N.A.	
	Later fusion, FC		# Classes

5.3.5 Long-term Semantic Action Encoder

As mentioned above, the feature maps over the entire video are aggregated for the video-level parameter optimisation. The classical aggregation functions such as average pooling, maximum pooling, weighted pooling and attention pooling will cause the missing of temporal information, while other sequence methods derived from the CNN-RNN concept (Donahue et al., 2017) are difficult to train. In contrast, recent research shows that convolutional architectures are better than recurrent networks in sequence modelling (Bai et al., 2018; Dauphin et al., 2017; Zecha et al., 2018). This research presents a 2D temporal convolutional network (2D TCN) based on the work of Bai et al. (2018) by extending 1D sequence modelling to a 2D task. In practice, the feature map output from the Pose Fast pathway is a feature vector after the global average pooling process, defined as $x \in \mathbb{R}^{G \times L}$, where L is the length of the feature vector, and G is the number of feature vectors concerning the number of 3D heatmap volumes, then a whole video generates a sequence of feature vectors: $\{x_1, x_2, x_3, \dots, x_G\}$. Formally, the sequence function is defined as: $F: x \in \mathbb{R}^L \rightarrow y \in \mathbb{R}^L$, i.e., the length of the output is the same as an input sequence, just like RNNs. Based on the TCN idealism, sequence modelling can be achieved by performing causal convolutions and dilated convolutions (Bai et al., 2018). In the 2D sequence situation, the causal and dilated convolution is defined as the following:

$$F(i, j) = (X * df)(i, j) = \sum_{u=1}^k \sum_{v=1}^N x(i-d \times u, j-v) \cdot K(u, v), \quad 5-5$$

where d presents the dilation factor, K is the convolutional kernel with the size of $L \times k$, hence $i-d \times u$ indicates the skipping units in the direction of the past. Noted that a dilated convolution transfers to a standard convolution if $d=1$, and the larger dilation factor enables characterising a wide range of temporal (past) features, hence encoding long-term information. Figure 5-5 demonstrates an architectural element in a three-layer 2D TCN structure, the dilation factors $d=1, 2, 4$ are set for the dilated causal convolutions, and the last output feature vector can cover all history values from the input sequence.

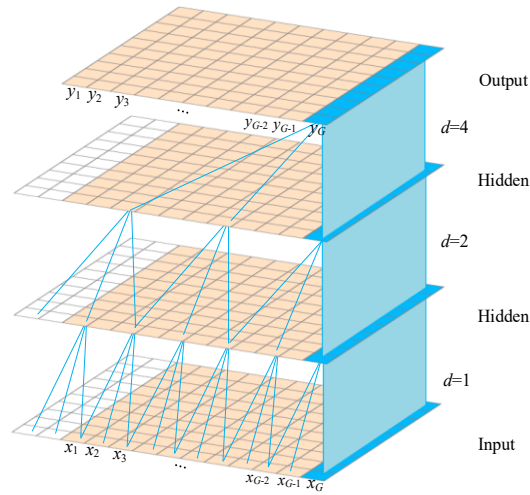


Figure 5-5. Architectural element in a three-layer 2D TCN structure.

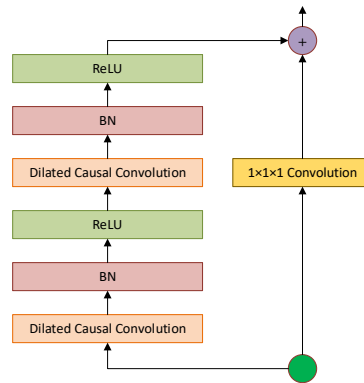


Figure 5-6. The architecture of 2D TCN blocks.

Therefore, this research stacks the dilated causal convolution blocks into a deep network for long-term semantic sequence encoding. To preserve longer information while keeping the network lightweight and efficient, the dilation factor is set to $d=2^q$, where q is the number of dilated causal convolution blocks. A typical 2D TCN block is shown in Figure 5-6, where the batch normalization (BN) and ReLU activation are integrated. In addition, the residual connections (He et al., 2016) are added between the convolutional layers to prevent the gradient vanishing problem. Such that the 2D TCN can encode long-term semantic information concerning pose skeleton-based action representation.

5.4 Action Recognition

5.4.1 Softmax-based Classification

Following the 2D TCN, a linear classifier that uses the softmax activation function is applied for action classification. For an action dataset with N types of action labels,

the classifier outputs a vector with N length: $P=\{p_1, p_2, p_3, \dots, p_N\}$, where element p_i refers to the probability belonging to i -th action label. The softmax activation function ensures that the sum of whole probabilities is 1. Noted that this classifier is a dataset-specific model, so one should train different models for various action datasets.

5.4.2 Recognition for Unseen Actions

When LSME is trained, one of the biggest challenges is the generalisation in the wild action videos, such as recognising the unseen action classes. Most human action recognition methods focus on the closed-set classification task and achieve good performances; however, these methods are not critical for the open-set action recognition challenge, i.e., a model trained on the NTU-60 action dataset can only test on the same dataset but cannot classify the videos from the Kinetics datasets, and the model even cannot recognise the “cross arms” action existing in the NTU-120 dataset but not in the NTU-60 dataset, such actions are considered “unseen” actions due to the model has never seen these classes of actions in the training stage, and it will never output these classes in the test stage. However, there are hundreds of thousands of actions in real-world scenarios. It is impossible to train such large identities with limited computational capacity and training videos. Therefore, a generic recognition model should be capable of identifying unseen actions with negligible data and computational cost adjunction, i.e., a model should directly learn an embedding instead of a multi-class classifier. A similar task is face recognition which can recognise an unknown face by comparing it with known faces in a specific database (Kortli et al., 2020).

By observing that the feature vectors of the 2D TCN-based long-term semantic action representation are conceptual similarities centres of each action class, unseen actions can be identified and verified by measuring their similarity and distances when the features have small intra-class and large inter-class distances (Deng et al., 2019). Formally, given a set of features $F \in \mathbb{R}^{L \times C}$ encoded from C classes of actions and L is the length of each feature vector, and an unknown video action with the feature $f \in \mathbb{R}^L$, the identification function is defined as: $D: (f, F) \rightarrow S \in \mathbb{R}^L$, and the highest score of $s_i \in S$ indicates its probability of belonging to i -th action. This research designs

two measurement methods, i.e., the Euclidean distance and the ArcFace-based (Deng et al., 2019) learnable method.

Euclidean distance computes the direct distance between two vectors. Suppose there are two feature vectors P and Q , the Euclidean distance is defined as the following:

$$d_E(P, Q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}. \quad 5-6$$

Then, this research matches the unknown video with known actions stored in a specific database by starting from the minimum scored distance and setting the threshold to 0.5.

Deng et al. (2019) presented a so-call Additive Angular Margin Loss (ArcFace) function for the maximum capacity of the discriminative power for large-scale face recognition. This research extends this idealism to unseen action recognition by learning the centres of each action class in a specific action database. The ArcFace is derived from the softmax loss function which is widely used for classification tasks (He et al., 2016; Simonyan & Zisserman, 2015), as shown in the following:

$$L_1 = -\frac{1}{B} \sum_{i=1}^B \log \frac{e^{y_i^T x_i + b_{y_i}}}{\sum_{j=1}^N e^{W_j^T x_i + b_j}}, \quad 5-7$$

where N indicates the number of classes, B is the batch size; $x_i \in \mathbb{R}^d$ is an embedding feature vector of i -th action video belonging to y_i -th class, d is the length of the feature, $W_j \in \mathbb{R}^d$ is the j -th columns of weight $W \in \mathbb{R}^{d \times n}$, and $b_i \in \mathbb{R}^d$ is the bias term; By transforming $W_j^T x_i = \|W_j\| \|x_i\| \cos \theta_j$, where θ is the angle between the feature x_i and the weights W_j , and the weights are also normalised by the L2 norm: $\|W\| = 1$, while the embedding feature $\|x_i\|$ is fixed by L2 norm and rescaled to s . Then, Equation 5-7 is rewritten as the following:

$$L_2 = -\frac{1}{B} \sum_{i=1}^B \log \frac{e^{s \times \cos \theta_j} + b_{y_i}}{e^{s \times \cos \theta_{y_i}} + \sum_{j=1, j \neq y_i}^N e^{s \times \cos \theta_j + b_j}}. \quad 5-8$$

The embedding features are distributed around each feature centre. Therefore, an additive angular margin penalty m between x_i and W_j^y is introduced to enhance the similarity for intra-class and diversity for inter-class. Then, ArcFace is defined as:

$$L_3 = -\frac{1}{B} \sum_{i=1}^B \log \frac{e^{s(\cos(\theta_{y_i} + m))} + b_{y_i}}{e^{s(\cos(\theta_{y_i} + m))} + \sum_{j=1, j \neq y_i}^N e^{s \times \cos \theta_j + b_j}}. \quad 5-9$$

To recognise unseen actions, this research freezes the parameters of the model trained on a large-scale dataset (e.g., NTU-60 action dataset) and replaces the softmax layer after the last convolutional layer by the ArcFace module. Then, the fully connected-based classifier is trained with litter video samples (10 to 20 videos per class) in a specific action database.

5.4.3 Spatial Fusion

One main drawback of the LSME method is the absence of contextual information. However, this minor weakness can be adapted by fusing an additional contextual model for integrating object and scene information. In a typical action video, the context is changed slowly, i.e., the progress of the “shooting goal” action does not change the identification of the player, football, and football field backgrounds. Therefore, this research applies the Slow pathway of the SlowFast model (Feichtenhofer et al., 2019) to provide spatial information from RGB video frames. The detailed architecture of the RGB Slow pathway is illustrated in Table 5-3, and a large temporal stride $\tau = T/4$ is used, i.e., only four RGB frames are randomly sampled for training the Slow pathway. Finally, both features are fused by a weighted pooling function which produces a set of linear weights to perform element-wise weighted linear fusion between the two feature vectors, defined as: $\sum_{c=1}^C w_c \cdot F_c$, where $C=2$, and F_c indicates the category of feature, w_c is fusion weight. Noted that both network parameters and fusion weights can be optimised simultaneously by end-to-end. In practice, this weight pooling function can be implemented by a convolutional layer with the kernel size of $(L_1+L_2) \times 1$, where L_1 and L_2 indicate the length of two feature vectors, respectively.

5.5 Experimental Results

5.5.1 Evaluation of Backbones

This research developed the short-term semantic encoding method based on the Fast pathway of the SlowFast model. However, the model is designed by using 3D heatmap volumes as input data, and it supports various 3D CNN backbones for the 3D sub-network implementation. This experiment evaluates different backbone settings by

measuring the top-1 and top-5 accuracy for performance comparison and static analysis of model efficiency. The number of parameters is used to define the model complexity, i.e., the large number of parameters corresponds to a heavy model which costs more computational resources and memory. Floating-point operations per second (FLOPs) is another sensitive measure of computational performance since it reflects a hardware-agnostic measure of model complexity.

This experiment compared the performance and computational cost of the I3D, R(2+1)D and Fast pathway networks. In practice, two types of Fast pathway network settings are explored, i.e., Fast pathway-50 and Fast pathway-101 based on ResNet-50 and ResNet-101, respectively. The NTU-60 skeletal data is used, and the experimental result is provided in Table 5-4. The result shows that although the lowest computational cost of the I3D model, the accuracy is very low. In contrast, the R(2+1)D backbone achieves a better performance than I3D because it has the largest trainable parameters and costs a lot of computational resources. Unsurprisingly, the Fast pathway backbones achieve the best performance, and its top-1 accuracy is just over 91%, while the top-5 accuracy is almost 100%. Furthermore, both Fast pathway-50 and Fast pathway-101 have significantly lower parameters than I3D and R(2+1) models, and the GFLOPs (10^9 FLOPs) of Fast pathway-50 backbone is also lower than R(2+1)D, which is capable of supporting real-time video analysis. Based on this observation, this research applies the Fast pathway-50 backbone to implement the 3D CNN-based sub-network for better performance and computation trade-off.

Table 5-4. The mean accuracy (in%) and computational performance of different backbones.

Backbone	Top-1	Top-5	Parameters (million)	GFLOPs
I3D	74.58	88.328	28.26	5.03
R(2+1)D	87.46	93.596	63.89	14.27
Fast pathway (50)	91.26	99.995	2.03	11.48
Fast pathway (101)	92.31	99.998	3.78	22.52

Table 5-5. The mean accuracy (in%) of different pose methods.

Pose method	Top-1	Top-5
Joints	91.26	99.995
HRNet (Heatmaps)	91.27	99.998
HRNet (Joints)	91.25	99.983
LCR-Net++ (Heatmaps)	90.54	99.960
LCR-Net++ (Joints)	91.04	99.948

5.5.2 Evaluation of Pose Methods

The proposed LSME model allows the use of any off-the-shelf pose methods or even directly reconstructing heatmaps from joint coordinates by performing a Gaussian blob. Table 5-5 provides the top-1 and top-5 accuracy on the NTU-60 dataset from three pose methods. For HRNet and LCR-Net++ pose estimation methods, each method is tested by two settings. i.e., the models were tested directly using heatmaps and reconstructing heatmaps from joints. The result shows that different pose methods have less influence on the final action recognition. Based on this observation and for storage-saving, HRNet is performed to extract human poses and store joint coordinates for each frame in the pose-processing stage. Then, the joint coordinates are transferred to heatmaps for model training in the training stage. In contrast, the heatmaps outputted from HRNet have directly applied for action recognition in the test process.

5.5.3 Evaluation of Sequence Modelling

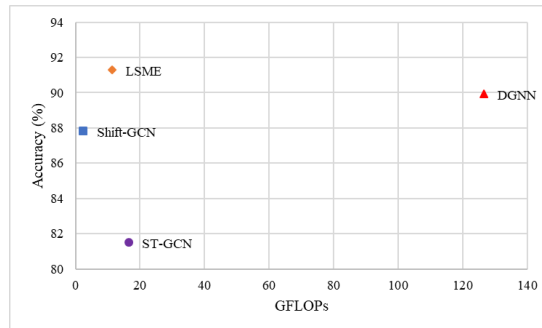
As an ablation exploration, the performance of sequence modelling under the proposed LSME model has been evaluated by using the NTU-60 skeleton dataset. The skeletons are transferred into 3D heatmap volumes for training a 3D sub-network. Then different sequence models are integrated for sequence modelling, including LSTM, GRU and the proposed 2D TCN modules. For a fair comparison, the 3D sub-network and linear classifier are implemented by the same structures and hyperparameters and trained on the same dataset (i.e., NTU-60). This experiment evaluates the accuracy of the three sequence models, as shown in Table 5-6. It can be seen that all of the sequence modules achieved good performance, which is over 90% for top-1 accuracy and greater than 99% in terms of top-5 accuracy. The possible reason is that the sequence of input feature vectors has a good short-term motion representation, while the order of feature vectors has remarkable sequence definitions; thus, the simplest recurrent modules can handle the temporal information gracefully. Nevertheless, 2D TCN is applied for long-term sequence modelling due to the TCN module shows not only the better performance among these sequence modules but also simpler and clear architecture, and it supports parallelisation in both training and test stages.

Table 5-6. The mean accuracy (in%) of different sequence modelling methods.

Sequence module	Top-1	Top-5
LSTM	90.24	99.987
GRU	90.23	99.960
2D TCN	91.26	99.995

Table 5-7. Comparison of mean accuracy (in %) between the proposed model with other state-of-the-art methods on the NTU-60 action dataset.

Method	Accuracy	Parameters (million)	GFLOPs
ST-GCN	81.5	2.8	16.7
DGNN	89.9	NA	126.8
1s-Shift-GCN	87.8	NA	2.5
LSME with softmax	91.26	2.03	11.48
+ Spatial Fusion	93.82	+31.63	+36.1
LSME with ArchFace	90.35	2.04	11.49
+ Spatial Fusion	92.59	+31.63	+36.1

**Figure 5-7. The accuracy/complexity trade-off on NUT-60 action dataset.**

5.5.4 Comparison with State-of-the-art Methods

This research then compared the proposed model with the state-of-the-art GCN-based methods, including ST-GCN (Yan et al., 2018), DGNN (Shi et al., 2019) and Shift-GCN (Cheng et al., 2020). In practice, the 1-stream setting of Shift-GCN (1s-Shift-GCN), which only uses the joint coordinates, was tested. The experimental result is shown in Table 5-7, where the accuracy of the Shift-GCN model is stability higher than ST-GCN and DGNN models because it is composed of spatial and temporal shift graph convolutions for adjusting the receptive field adaptively. The LSME method exceeds all skeleton-based methods. The superior performance of the proposed approach stems from the implicit pose representation of 3D heatmap volume and the fine motion encoding of the Pose Fast pathway. The performance gain is also contributed by the long-term semantic action representation ability of the 2D TCN module. Unsurprisingly, performance improvement can be further achieved when fusing the spatial Slow pathway model, as shown in Table 5-7.

The number of parameters and computational cost are also compared. As shown in Table 5-7, the GCN-based methods have a lot of parameters. In contrast, the Pose Fast pathway-based LSME model has the smallest number of parameters; thus, it is more lightweight than GCN-based implementations. When considering the processing speed, the ST-GCN costs 16.7 GFLOPs for a video clip, while DGNN reaches 126.8 GFLOPs because of the complex DGN blocks and multi-stream fusion mechanism. By contrast, the Pose Fast pathway costs low computational resources because it preserves fewer feature channels in the network, and the input size is 3×56^2 which is smaller than other models. The trade-off between accuracy and complexity is illustrated in Figure 5-7, showing that the LSME method has the best accuracy and computation trade-off.

5.5.5 Comparison of Out-of-Context Dataset

Skeleton-Mimetics is a very complicated action dataset because of the absent or misleading backgrounds, objects, and scenarios in most video samples which are out of context. From Table 5-8, all methods obtain relatively poor performance in this dataset, i.e., MS-G3D and 4s-Shift-GCN achieve approximately 50% accuracy in the Skeleton-Mimetics dataset, while the accuracies in the NTU-60 dataset are 91.5 % and 85.9%, respectively. In conclusion, although LSME achieves better accuracy, which is still poor in the Skeleton-Mimetics dataset compared to the results in the NTU-60 dataset.

Table 5-8. Comparison of accuracy among skeleton-based methods in out-of-context datasets.

Method	Skeleton- Mimetics (in %)	
	Top-1	Top-5
MS-G3D	49.22	NA
4s-Shift-GCN	51.10	NA
LSME with softmax	52.83	78.59
LSME with ArchFace	51.04	75.32

5.5.6 Evaluation of Unseen Actions

This research evaluated the performance of the proposed LSME model in the unseen action scenario. 10 action categories are selected as unseen action videos from the NTU-60 action dataset. Then, the model is retrained by using the other 50 action categories of NTU-60. Finally, this experiment separately tested the performance in the unseen action videos by using Euclidean distance and ArcFace methods, respectively.

The experimental result shown in Table 5-9 obviously indicates that the performance in the seen actions is similar to the model trained on the original datasets, with a minor improvement since there are fewer action categories. The performance on unseen actions is 55.6% and 60.34% for Euclidean and ArcFace methods, respectively. Considering the model has never seen these actions in the training stage, this performance is encouraging.

Table 5-9. The mean accuracy (in%) on unseen actions.

Method	NTU-60	
	Seen (50)	Unseen (10)
Euclidean distance	93.53	55.6
ArcFace ($NS=10$)		60.34

5.6 Summary

Although the encouraging performance of recent DNN methods on human action datasets, the influence of biases of the learned models is less explored. According to the evaluation described in this chapter, most DNN models tend to model contextual features instead of interpreting inherent action definitions and semantic representations, which fails to cope with the cases of video actions in the absence and misleading context. This research has presented a human pose skeleton-based LSME model for encoding long-short-term action representation to understand the semantic definition of actions. By introducing Euclidean and ArcFace methods, this research aims to solve the open-set action recognition challenges. Experiments carried out on the NTU and Skeleton-Mimetics datasets show better performance than previous works and a good trade-off between accuracy and computational cost, suggesting a vital direction of research on understanding human actions and solving action recognition problems in real-world applications, where the large-scale dataset is unavailable.

CHAPTER 6 Model Inference on Edge Computing

6.1 Introduction

When a DNN algorithm is trained, an ongoing stage is to deploy the model into modern applications, known as model inference. A typical solution is deploying an AI model into a cloud-based service which consumes original data (e.g., videos) coming from distributed edge devices (e.g., cameras, robots, and kiosks) and gives responses from server to client for decision-making, namely a few, AI on Cloud. This implementation requires expensive AI infrastructures such as energy-consuming GPUs, large amounts of memory, and extensive communication bandwidth; thus, it is far from practical when facing hundreds of thousands of Internet of Things (IoT) devices (Shi et al., 2016). For instance, gigabytes (GB) of videos will be captured by an autonomous vehicle every second, and the real-time response of data processing is required to determine the next operations. The unreliable network connection fails to cope with the correct decision-making if all original videos send to the cloud service for processing. Data privacy is another sensitive problem under the General Data Protection Regulation (GDPR) (Voigt & Bussche, 2017) since transferring such a large quality of videos crossing the Internet will easily cause personal information leakage. In contrast, edge computing technology has the potential to tackle the cues of energy/cost saving, real-time response, and data privacy and security (Deng et al., 2020; Shi et al., 2016). In that case, the models are deployed at the mobile devices where data are produced for storing, processing, and analysing, while only the valuable results are posted into cloud services, hence supporting more efficient data processing, shorter response time, and reliable decision making, namely a few, AI on Edge, or edge intelligence.

Carrying out AI to edge computing has various brand-new challenges due to the energy and cost of an edge device are always limited for processing such large volumes of data by a complicated model. To tackle these issues, this research firstly explores the characteristics of different edge accelerators, followed by the Open Neural Network

Exchange (ONNX)-based PIM model representation. Then, this research investigates the core concepts and technologies of model quantization to transfer a heavy model to a lightweight one, followed by a partitioning mechanism to partition a computational graph into sub-graphs for parallel execution in accelerators in a heterogeneous system.

6.2 Computational Platforms

The classic practice of algorithm development is training and testing DNN models on GPU-enabled servers or workstations, which is neither affordable nor energy-saving for massive applications. Since the birth of edge computing, there have been alternative computing platforms for model inference, i.e., models are trained on GPUs and deployed into edge platforms and hardware, e.g., Arm NN and NPU.

6.2.1 GPU

Enabled by the Compute Unified Device Architecture (CUDA), modern GPUs are not only powerful graphics engines but also parallel arithmetic and programmable processors. All deep learning frameworks support CUDA acceleration since training a model requires a great deal of computational cost and a large amount of memory, while GPU naturally support parallel arithmetic, such as convolution and GEMM (general matrix multiplication) (Kurzak et al., 2012; Qin et al., 2020) which are the basic modules of DNN architectures. However, GPU is not suitable for mobile and embedded systems due to the restricted hardware budget, limited energy supply and small space for hardware integration. Therefore, the more applicable edge chips are required for lightweight model inference, such as system-on-a-chip (SoC).

6.2.2 Arm NN

The Arm NN is a machine learning platform optimized for the Arm NEON SIMD (Single instruction multiple data) architecture, which uses the Arm Compute Library (ACL) as a backend to map target programmable cores. Arm NN supports machine learning programs on the edge and mobile devices through a set of software and tools, hence providing a bridge between general deep learning frameworks and the power-

efficient embedded CPUs, Ethos NPUs, and Mali GPUs for model inference. The software supports models created from other deep learning frameworks and transfers the models into internal Arm NN format particularly designed for the target hardware. The architecture of Arm NN is shown in Figure 6-1.

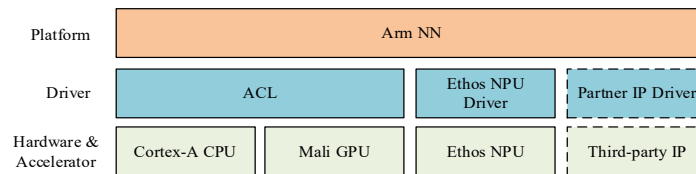


Figure 6-1. The architecture of Arm NN.

6.2.3 NPU

NPU is an AI accelerator of a specialised microprocessor that implements all the necessary controls and arithmetic logics to execute deep learning models. It is specific-designed for executing CNN and RNN modules, typically in the low-precision (e.g., 8/4-bit (unsigned) integer) arithmetic for high-performance acceleration. Noted that NPU cannot be used for general-purpose computing due to it may be a part of a large SoC. Some of the current NPU engines are AWS Inferentia, NVIDIA Deep Learning Accelerator (NVDLA), Neural Engine by Apple, Rochchip NPU, Samsung NPU, etc. However, different accelerators require hardware-specific SDKs. It is inflexible and less robust to transfer an AI application from one platform to another.

6.3 Platform Independent Model Design

6.3.1 ONNX

Different deep learning frameworks have various internal formats for model storing. However, most inference platforms do not directly support these formats; thus, it is necessary to transfer a framework-specific format to a hardware-specific format, which is still inefficient when facing various platforms. It is especially true when the quantization process is involved in numerical optimization. To tackle this problem, the Open Neural Network Exchange (ONNX) was developed by Facebook and Microsoft (ONNX, 2021). ONNX is an open format built to represent models, including both deep learning and machine learning algorithms. It provides a definition of an extensible

computation graph model and definitions of build-in operations and standard data types. As shown in Figure 6-5 and Figure 6-6, the fully connected and CNN models are represented as Directed Acyclic Graph (DAG) formats, in which a node is an operation, and the arrow indicates the direction of dataflow. Current deep learning frameworks support ONNX export, such as *torch.onnx* and *Tensorflow-ONNX* tools for PyTorch and TensorFlow model converting, respectively.

6.3.2 ONNX Runtime

Executing models on accelerators is very dependent on hardware-specific programming libraries and does not compatible with other platforms. ONNX Runtime (ORT), developed by Microsoft, is a model inference framework supporting multiple software platforms and hardware accelerations (Microsoft, 2021). ORT provides performance improvements compared to the original frameworks benefiting from its built-in optimizations. Furthermore, an extensible Execution Providers (EP) framework is also designed to optimally execute an ONNX model on the target hardware platforms (Microsoft, 2022), hence supporting various acceleration libraries and hardware, as shown in Figure 6-2 (Microsoft, 2021). Based on this concept, ORT partitions a model represented by a graph into sub-graphs based on available hardware-specific accelerators, and then it assigns the sub-graphs into different EP libraries in supported hardware for execution. The ORT-based model inference process is shown in Figure 6-3 (Microsoft, 2022). This research builds a Linux version of ORT using the open-source code within the Arm NN and extended EP libraries to support model execution on NPUs.

Platform	Linux	Windows	Android/iOS	Web Browser
API	C/C++	Python	Java	JS
Architecture	X86/64	Arm32/64	IBM Power	Apple M1
Hardware Acceleration	CPU	CUDA	OpenVINO	CoreML
	TensorRT	Arm NN	NPU	MiGraphX

Figure 6-2. The architecture of ONNX Runtime.

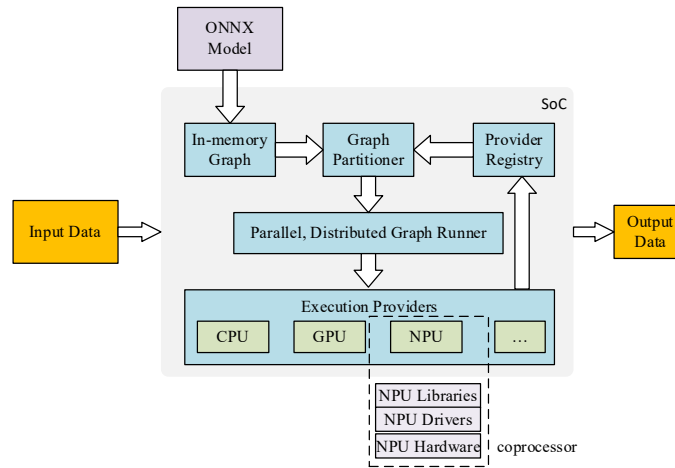


Figure 6-3. The processing flow of ONNX runtime inference.

6.4 Workflow of AI on Edge

In contrast with the model training process on cloud-based systems, the inference stage on edges is considerably less expensive due to the limited compute and storage capacity. How to carry out the model inference on resource-constrained edge devices is a serious issue. Approaches primarily improve the existing frameworks and libraries to make them more suitable for edge computing by forming model adaptation to hardware acceleration (Deng et al., 2020). Inspired by this direction, this research mainly explores model partitioning and quantization for hardware acceleration. Figure 6-4 shows the overview of the AI on edge scheme. Starting from a trained model exported by a specific deep learning library, e.g., PyTorch and TensorFlow. A model is represented as a DAG in standardised ONNX format. Then, the quantization method converts the full-precision model into low-precision. It is followed by a partitioning approach to partition the model into sub-graphs according to the supported operations enabled by target accelerators, i.e., the CPU supports all operations, but the computational performance is relatively lower. In contrast, the accelerators only support a portion set of operations (e.g., the 3D CNN, Einsum, and Gemm operations cannot execute on the NPU) with high-speed execution, so the DAG is spitted into sub-graphs, and the sub-graphs contain unsupported operations execute on CPU. In contrast, others sub-graphs support running on NPU or Arm Mali GPU accelerators. Finally, the sub-graphs are compiled to generate a platform-dependent (PD) model for inference on

disparate edge devices. Note that the compilation stage is not required when applying ONNX Runtime for model inference.

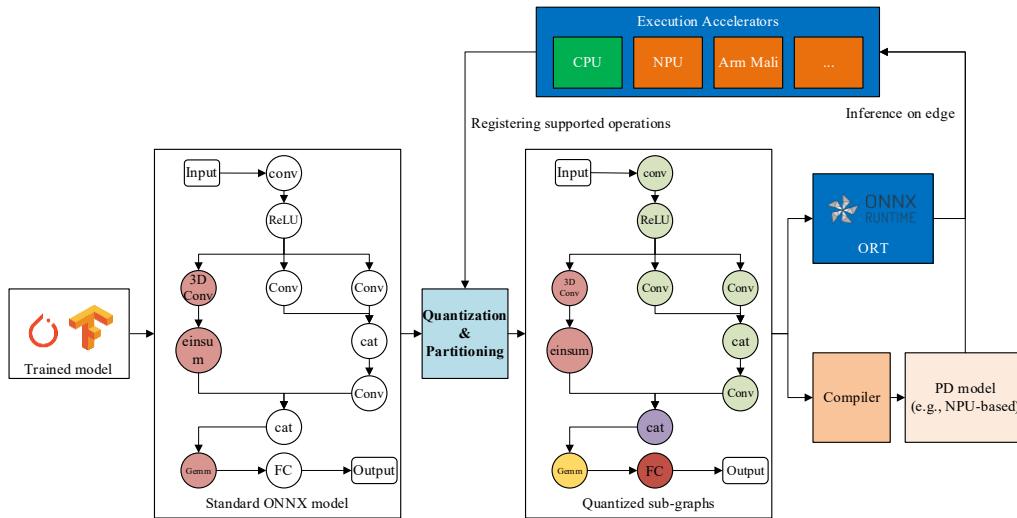


Figure 6-4. The overall workflow of the AI on edge scheme.

6.5 Model Quantization

6.5.1 Concept of Quantization

Reducing the computational cost, communication latency, and power consumption is a key challenge of model inference on edge devices. One mechanism is to develop efficient network architectures such as MobileNets (Howard et al., 2019; Sandler et al., 2018). Although the processing speed is improved, the performance is dropped because of the fewer learnable parameters and simplified model structures. Furthermore, this approach requires re-designing and training the specific models instead of applying the state-of-the-art models in applications. Therefore, the other strategy is to optimize existing models for reduced model size and improved efficiency by performing neural network quantization (Nagel et al., 2021), and then accelerating the processing speed through SIMD processors.

The fundamental of quantization is representing full-precision (i.e., 32-bit floating-point – FP32) arithmetic to low-precision, e.g., 8-bit integer number (int8 or uint8), which can naturally reduce a mode size by a factor of 4, and the computational cost for convolution operation reduces by a factor of 16. Further, moving 8-bit integer numbers is more efficient than the 32-bit floating-point data, and the AI accelerators efficiently support 8-bit integer arithmetical computation. Supposed a convolutional

layer of a deep learning model is defined as $Y=WX+B$, where X is the input tensor, W and B are weight and bias, respectively, and Y indicates the output tensor. Two quantizer parameters are defined: the scale factor s , and the zero-point z , to map a FP32 value to an 8-bit integer number. Then, the quantized convolutional layer is defined as:

$$Y_Q = X_Q * W_Q + B_Q, \quad 6-1$$

where X_Q , W_Q and B_Q are quantized input data, weight, and bias, respectively, defined as follows:

$$X_Q = \text{clamp}\left(\left\lfloor \frac{X}{S_X} \right\rfloor + z_X; 0; N_{level} - 1\right), \quad 6-2$$

$$W_Q = \text{clamp}\left(\left\lfloor \frac{W}{S_W} \right\rfloor + z_W; 0; N_{level} - 1\right), \quad 6-3$$

$$B_Q = \text{clamp}\left(\left\lfloor \frac{B}{S_B} \right\rfloor + z_B; 0; N_{level} - 1\right), \quad 6-4$$

where S_X , S_W and S_B are scale factors for X , W and B , respectively, while Z_X , Z_W and Z_B are the corresponding zero-points; $N_{level}=2^8=256$ for 8-bit integer; $\lfloor \cdot \rfloor$ is the round-to-nearest operation to convert a floating-point number to an integer, and the clamping function is defined as the following:

$$\text{clamp}(x; a, b) = \begin{cases} a, & x < a \\ x, & a \leq x \leq b \\ b, & x > b \end{cases} \quad 6-5$$

Once the quantized output Y_Q is obtained, the de-quantization operation is performed to obtain the full-precision output, as shown in the following:

$$Y = \frac{Y_Q - Z_Y}{S_Y}, \quad 6-6$$

where S_Y and Z_Y are scale factor and zero-point, respectively, for the final output data Y in full precision. The above definition is called asymmetric quantization. By restricting the zero-point to 0, a simplified symmetric quantization version can be defined. In that case, the calculation of X_Q in Equation 6-2 can be rewritten as follows:

$$X_Q = \text{clamp}\left(\left\lfloor \frac{X}{S_X} \right\rfloor; 0; N_{level} - 1\right), \text{ for unsigned integers (uint8)}, \quad 6-7$$

$$X_Q = \text{clamp}\left(\left\lfloor \frac{X}{S_X} \right\rfloor; -N_{\text{level}-1}; N_{\text{level}-1} - 1\right), \text{ for signed integers (int8)}. \quad 6-8$$

The calculation of W_Q and B_Q have the same definition, and $Y=Y_Q/S_Y$. Symmetric quantization has higher efficiency than asymmetric quantization due to it does not deal with the zero-point offset operations.

Based on the quantization definition, the key is to determine the quantizer parameters, which can be implemented by two main classes of algorithms, i.e., Post-Training Quantization (PTQ) and Quantization-Aware-Training (QAT). PTQ directly converts a pre-trained FP32 model into a low-precision model without re-training the model, which allows for quantization with data-free or small calibration data (Nagel et al., 2021). According to the experiments carried out by Krishnamoorthi (2018), the per-channel asymmetric quantization performs close accuracies to the floating-point of various neural network models. However, the accuracy will decrease when facing extremely low-bit quantization, e.g., 4-bit precision. In contrast, QAT quantizes models during the training stage, allowing the network weights bias quantized models to provide higher accuracy than PTQ. This method starts fine-tuning a floating-point pre-trained model by using a similar training dataset and then updating the weights in float points with the gradients before quantization operations are applied to quantize weights. The SGD based weight updating is given as follows:

$$w_{float} = w_{float} - \eta \frac{\partial L}{\partial w_Q} \cdot I_{w_Q}, \quad 6-9$$

$$w_Q = s_w \text{clamp}\left(\left\lfloor \frac{w_{float}}{s_w} \right\rfloor; -z_w; 0; N_{\text{level}} - 1\right), \quad 6-10$$

where $\partial L/\partial w_Q$ is the backpropagation error of the loss function. QAT can mitigate the quantization noise during the finetuning stage. Therefore, it improves the performance of quantized models and enables lower bit inference with negligible accuracy decrease.

6.5.2 Case: Fully Connected Layer

As a basic case, Figure 6-5 shows two fully connected (FC) layers with a ReLU activation, which can be executed within the ORT. The original (left) and quantized

(right) ONNX models are visualised as DAG by using the Netron tool (Roeder, 2017). A single linear layer is converted into quantization, int8 matrix-matrix-multiply, and rescaling operations, and the *DynamicQuantizeLinear* operation quantize the input, model weights, and biases of layers from FP32 into int8 types. Then, the *MatMulInteger* operation performs 1D convolution within the int8 type, which outputs an int32 result. Finally, the quantized result is rescaled into full precision by casting, multiplication with scale parameter, and adding zero-point parameters.

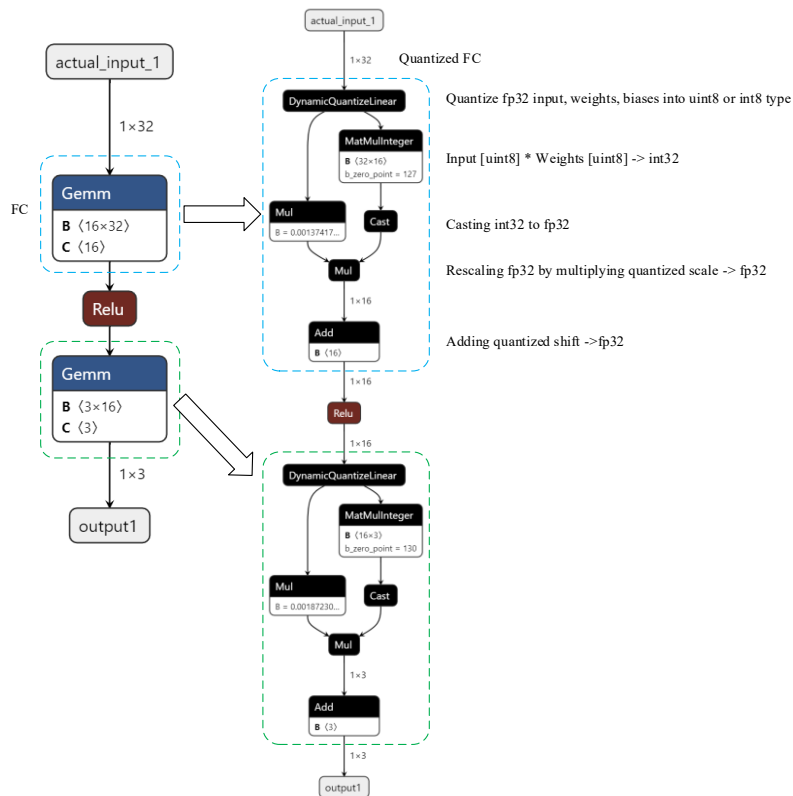


Figure 6-5. Case study of fully connected layer ONNX graphs.

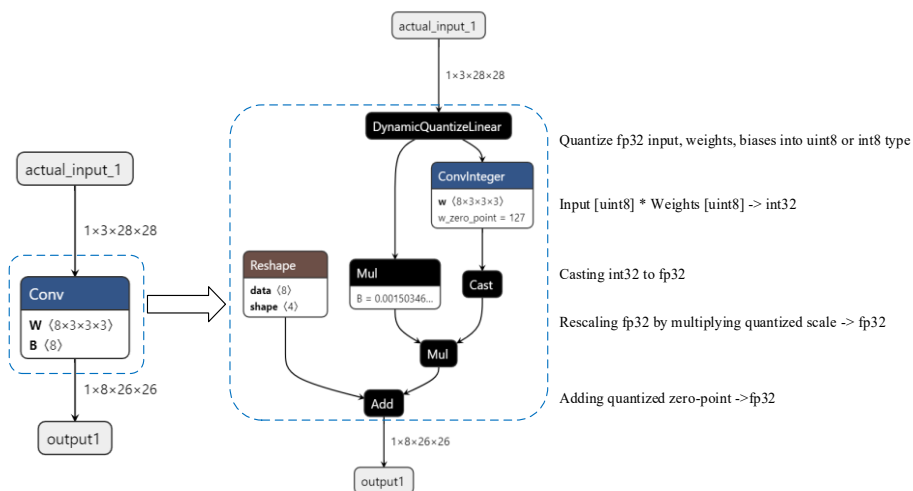


Figure 6-6. Case study of convolutional layer ONNX graphs.

6.5.3 Case: Convolutional Layer

The same as a fully connected layer graph, the quantized 2D CNN layer based ONNX graph has similar processing operations. As shown in Figure 6-6, a single convolution layer is converted into quantization, 2D integer Convolution, and rescaling operations. All the operations process 2D data with the same programs of the fully connected graph. Followed by the two cases, other complicated models have the same graph formulation, and both original and quantized ONNX graphs can be executed using ORT. It is worth noting that for a specific accelerator, the ONNX graphs can compile into a target platform model for faster inference, i.e., the scale and zero-point parameters can be calculated in advance to reduce the computational cost.

6.6 Model Partitioning

Due to the unbalanced development between software and hardware technologies, the advanced operations may not be supported by accelerators. For instance, most NPUs cannot execute 3D CNN and Einstein notation (Laue et al., 2020) operations; as a result, the whole network cannot be loaded on the accelerators. Zecha et al. (2018) partition a neural network model into two parts and offload the computationally intensive one to the cloud, while the lightweight part performs at the local edge to reduce the latency and speed up the applications on mobile devices. This concept can migrate into edge intelligence, i.e., edge devices should be able to execute a model in a heterogeneous platform with disparate processors and accelerators. To do that, a DAG model is defined as $G = (V \cup \{e, c\}, \mathcal{L})$, where the set of vertices $V = \{v_1, v_2, v_3, \dots, v_n\}$ indicate the operations of DNN models, which can be convolutional layers or activations, etc. e and c are the input and output nodes. $(v_i, v_j) \in \mathcal{L}$ denotes a link representing the output of v_i feeds to v_j . Let $P = \{p_i\}$ represents all operations of DNN models, and P^{cpu} is the set of supported operations by CPU, and $P^{cpu} = P$ since CPU supports all operations. In contrast, accelerators only support a portion of operations, denoted by $P^{npu} \subseteq P$ for NPU accelerator. Due to the processing speed on accelerators is faster than CPU, the operations should be maxillary assigned to accelerators, and only the unsupported operations should be performed on the CPU. Mathematically, the set of vertices $V_s \subseteq$

V represent the unsupported operations, where $v_i \in V_S$ if $v_i \notin P^{npu}$, then $V_S \cap P^{npu} = \phi$. Removing V_S causes the rest of G (denoted by $V_E = V - V_S$) becomes several disconnected components, and each one is a sub-graph. Finally, two groups of sub-graphs are obtained, i.e., $\cup G_i^{cpu} = V_S$ that performs on CPU, and $\cup G_i^{npu} = V_E$ that executes on NPU. Noted that although the above partitioning process assumes a single accelerator in a heterogeneous system, it can be easily extended to the cases containing disparate accelerators.

Based on the partitioning definition, this research defines t_i^{cpu} and t_i^{npu} as the processing time of operation v_i on CPU and NPU, respectively. Then, the processing time on CPU is

$$T^{cpu} = \sum_{v_i \in V_S} t_i^{cpu} . \quad 6-11$$

And the processing time on NPU is

$$T^{npu} = \sum_{v_i \in V_E} t_i^{npu} . \quad 6-12$$

The data communication time can be ignored because CPU and NPU share the same memory on edge devices; then, the total executing time is $T = T^{cpu} + T^{npu}$. In practice, CPU and NPU can execute parallelly, so the real executing time $\hat{T} \leq T$.

6.7 Experimental Results and Validation

6.7.1 Evaluation of Quantization Methods

Transferring models from 32-bit to 8-bit integer mode allows for a smaller model size. This experiment compared the model sizes between full precision and low precision by using PQT and QAT quantization methods, respectively. The models are represented in ONNX format, except the compiled ones for NPU hardware acceleration. As shown in Table 6-1, this experiment evaluated ResNet (He et al., 2016), DenseNet (Huang et al., 2017), MobileNet v2 (Sandler et al., 2018), and HRNet (Wang et al., 2021a). The size of the PQT-based model is approximate quartern of the original model due to the weights being stored in 8-bit integer type, which is reduced by a factor of 4 compared to 32-bit floating-point numbers. In contrast, the QAT-based model is slightly larger than PQT. The main reason is that the QAT-based ONNX graph

introduced extra calculations for quantization parameter estimation. Nevertheless, when compiling the ONNX model into an NPU-based platform-dependent model, the model size is reduced to the level of the PQT method.

Table 6-1. Comparison of model size between fp32 and uint8 types.

Model	Model size (KB)			
	Original	PQT	QAT	QAT (Complied)
ResNet-34	85.121	21.342	22.842	21.332
ResNet-50	99.739	25.065	31.064	25.047
ResNet-101	173.836	43.715	49.715	43.681
DenseNet-169	55.862	8.442	15.012	19.886
MobileNet v2	13.638	3.511	7.260	3.499
HRNet	302.482	76.088	82.088	76.006

6.7.2 Evaluation on Accelerators

This experiment evaluated the efficiency and effectiveness of model quantization by comparing the processing time between original models and quantized ones. The experimental platforms are shown in Table 6-2. The workstation has a powerful CPU and GPU. In contrast, the edge device has limited computational resources, i.e., the computational performance of NPU is only 2.0 TFLOPS (trillion floating-point operations per second), and only the integer computation is supported. The image classification models are carried out in this experiment by using the ILSVRC2012 test set, including, ResNet-50, MobileNet v2, DenseNet-169, and HRNet. The accuracy between the original and quantized models was compared, and the result is shown in Figure 6-7. The accuracies of quantized models are almost equal to the original models, with only a slight decrease, which proves that the quantized models can still preserve high effectiveness.

This research then tested the processing time of model inference on different computational platforms, and the experiment result is shown in Table 6-3. Both CPU and GPU of the workstation show high computational performance because of the powerful processors. However, the processing time rapidly increases when executing the original models on the Arm CPU because of the resource-constrained platform. Nevertheless, running a quantized model on the same platform requires less processing time which is reduced by a factor of two. The main reason is that accessing 8-bit numbers is faster than accessing 32-bit values. Furthermore, when compiling the

ONNX model into a NPU platform-dependent model, which will partition the ONNX graph into sub-graphs to enable the NPU acceleration, the processing time is significantly dropped, reaching the level of CPU of the workstation. Based on this observation, the quantized models perform faster inference while preserving high performance, suggesting a novel solution for the research on edge intelligence.

Table 6-2. The developing and testing platform setups.

	Workstation	Edge device
CPU	AMD Ryzen 5 3600 6-Core Processor	Quad core ARM Cortex-A7
Memory	16 GB	1 GB
Accelerator	NVIDIA RTX 2080 Ti, 13.45 TFLOPS (fp32)	NPU, 2.0 TFLOPS (int8/16)
System	Ubuntu 20.04	Linux
Library	ONNX Runtime	ONNX Runtime, NPU Driver

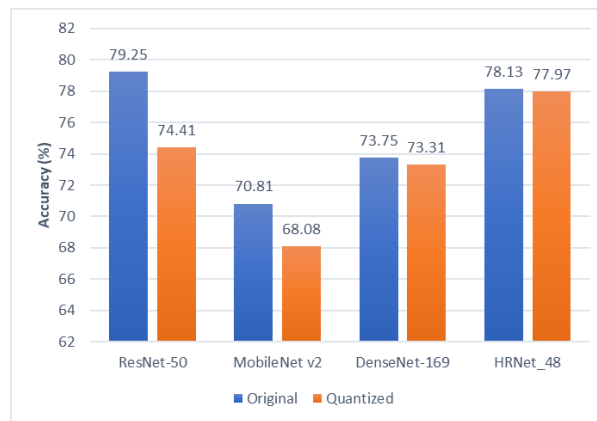


Figure 6-7. The comparison accuracy (in %) of original and quantized models.

Table 6-3. The comparison processing time (in milliseconds) of original and quantized models on different computational platforms.

Model	Workstation		Edge device		
	Original (CPU)	Original (GPU)	Original (CPU)	Quantized (CPU)	Quantized (NPU)
ResNet-50	15	5	3919	1787	28
DenseNet-169	24	11	3413	2234	66
MobileNet v2	2	7	531	619	6
HRNet	71	18	16524	7759	122

6.8 Summary

Although it is in the early stage, edge intelligence has become the trend of the next computing paradigm, and model inference on resource-constrained edge devices is a huge challenge. To reduce the computational cost and speed up faster execution. This chapter represents a deep learning model as a DAG by using ONNX for platform-independent design and optimisation. The computational cost and memory demand are

significantly reduced through model quantization, and the model partitioning scheme is explored for hardware acceleration on heterogeneous systems. The experiments show that the quantization and partitioning methods have significantly reduced the processing time while maintaining similar performance to the original models. This work suggests a vital direction of research on edge intelligence.

CHAPTER 7 Conclusion and Future Work

7.1 Contributions to Knowledge

This research aims to address key challenges of human action understanding. A collection of methods regarding image and video processing, feature extraction and learning, semantic discovery, and action classification/prediction have been presented in this thesis. In addition, the methodology of model inference on edge computing has been investigated, and the trend of edge intelligence has also been explored. These contributions have delivered the objectives set at the start of the research. The main contributions to the domain knowledge are summarised in the following aspects.

- 1) In Chapter 3, the advantages and drawbacks of traditional feature methods have been explored, leading to the innovation of DWT-driven DT feature extractor and the FV, BOF, and BoTF-based event representations to identify the approximate features and descriptors for better performance on video analysis. This enables the attainment of more accurate and multi-resolution features within spatial space to encode the “longer” temporal information in videos.
- 2) In Chapter 4, the end-to-end multimodality neural networks have been devised for automatic feature extraction and action classification. To improve the effectiveness and efficiency of spatial-temporal feature extraction and to learn the hybrid spatial and temporal pattern in a video, an interactive two-stream aggregation network based on OFF is proposed to replace the time-consuming optical flow computation. It further enables the coarse-to-fine scene and motion interactions by innovating STFB constructs between vision and motion cues. Then, the 3D CNN-based aggregation model is capable of representing long-term semantic movements, which contributes to better performance on action classification.
- 3) In Chapter 5, the context-biased problem of DNN models has been evaluated, which shows that the DNN models tend to leverage contextual information for vision classification instead of interpreting inherent human actions based on

their semantic definitions. To learn generically semantic representations and to understand human action definitions, the LSME method is proposed based on human pose skeletons and 3D convolutional networks. A long-short-term learning strategy is presented for training DNN models and updating learnable parameters in the video-level gradients instead of the clip-level. By introducing Euclidean and ArcFace methods, LSME is capable of encoding signatures of unseen actions for solving the open-set action recognition challenge in test videos from real-world applications.

- 4) In Chapter 6, the opportunities and challenges of edge computing and AI on edge have been investigated, highlighting a trend of edge intelligence for the transition from IoT to Intelligent Internet of Intelligent Things based on the future of the 6G Intelligent Edge paradigm. Specifically, the ONNX-based PIM design is introduced to represent a model in the form of DAG that enables further graph partitioning, mathematical quantization, parallel execution, and hardware acceleration. The related methodologies have been explored in this research which suggests a novel direction of edge intelligence.

It is demonstrated that these proposed approaches are valuable for real-world applications and problem-solving such as surveillance video analysis, autopilot, and healthcare systems. Furthermore, these methodologies and technologies have potential and benefits for handling other tasks in artificial intelligence, computer vision, computational optimization, and edge intelligence.

7.2 Future Work

In addition to the encouraging performance of the newly developed algorithms and techniques, other opportunities and challenges have risen for future exploration:

- 1) Although the DNN-based methods show better performances on various computer vision tasks, there are still demands and opportunities to further research, develop, and modularise feature engineering algorithms for embedded and node-level usage. It is demonstrated that the image processing algorithms such as element-wise subtraction and Sobel operations can generate spatial and

temporal gradients from CNN feature maps, hence the motivation for developing the OFF layer in this research.

- 2) One of the limitations of this research is that the devised human action recognition models can only process trimmed videos with clear action boundaries, such as the video clips from the UCF, HMDB and NTU datasets. A number of aspects have been explored as a preparation for the following ups, including tailored deep learning networks and adaptive feature weighting to better handle varying lengths of ambiguous crowd behavioural events.
- 3) The DNN model should be further examined for what knowledge and biases are learned during the training process. Model complexity is another sensitive cue for modern DNN-based applications, which is especially true when facing resource-constrained mobile devices that require lightweight model size, high performance, and lifelong data processing.
- 4) Edge intelligence, which is in its early stage, has many challenges and opportunities worth researching, including graph optimization, mathematical quantization, AI hardware design, software (AI) defined hardware, and data security and privacy. For example, although the hardware architectures and programming platforms keep on improving at a rapid rate, several advanced operations still cannot be supported by AI accelerators, resulting in failure to execute the computation-intensive DNN models on mobile devices. Inspired by software-defined networking that allows the control of communication networks flexibly by programmable interfaces. AI technologies can provide optimal solutions to key problems in computing infrastructure design and Intellectual Property (IP) development (Deng et al., 2020).
- 5) In the edge computing era, data are produced and processed by widespread and geographically distributed IoT and mobile devices. However, high performance and energy-saving are incompatible simultaneously. A potential solution is the co-inference with device-edge synergy (Li et al., 2018), where a large amount of original data are pre-processed on edge devices, whilst the small middle features transfer to the cloud for global processing. This scheme also

encapsulates better data security and privacy than offloading original video data directly to cloud services.

It is anticipated that these challenges and their solutions will push the advancement of computer vision and edge intelligence for future more ubiquitous and pervasive applications. I hope that this thesis can inspire professional and fruitful discussions to accelerate this trend.

References

- Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), e00938. <https://doi.org/10.1016/j.heliyon.2018.e00938>
- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. International Conference on Engineering and Technology (ICET), 1-6. <https://doi.org/10.1109/ICEngTechnol.2017.8308186>
- Anthwal, S., & Ganotra, D. (2019). An overview of optical flow-based approaches for motion segmentation. *The Imaging Science Journal*, 67(5), 284-294. <https://doi.org/10.1080/13682199.2019.1641316>
- Bahl, L., Brown, P., Souza, P. d., & Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. IEEE International Conference on Acoustics, Speech, and Signal Processing, 49-52. <https://doi.org/10.1109/ICASSP.1986.1169179>
- Bahng, H., Chun, S., Yun, S., Choo, J., & Oh, S. J. (2020). *Learning De-biased Representations with Biased Representations* International Conference on Machine Learning, Proceedings of Machine Learning Research.
- Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*.
- Bertasius, G., Wang, H., & Torresani, L. (2021). Is Space-Time Attention All You Need for Video Understanding? Proceedings of the International Conference on Machine Learning (ICML),
- Bolvinou, A., Pratikakis, I., & Perantonis, S. (2013). Bag of spatio-visual words for context inference in scene classification. *Pattern Recognition*, 46(3), 1039-1053. <https://doi.org/10.1016/j.patcog.2012.07.024>
- Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11), 120-123.
- Brox, T., Bruhn, A., Papenbergh, N., & Weickert, J. (2004). High Accuracy Optical Flow Estimation Based on a Theory for Warping. In T. Pajdla & J. Matas, *Computer Vision - ECCV 2004* European Conference on Computer Vision, Berlin, Heidelberg, 25-36.
- Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2021). OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1), 172-186. <https://doi.org/10.1109/TPAMI.2019.2929257>
- Cao, Z., Simon, T., Wei, S., & Sheikh, Y. (2017). Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1302-1310. <https://doi.org/10.1109/CVPR.2017.143>
- Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., & Zisserman, A. (2018). A short note about kinetics-600. *arXiv:1808.01340*.
- Carreira, J., & Zisserman, A. (2017). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4724-4733. <https://doi.org/10.1109/CVPR.2017.502>

- Chandra, M. A., & Bedi, S. S. (2018). Survey on SVM and their application in image classification. *International Journal of Information Technology*, 13(5), 1-11. <https://doi.org/10.1007/s41870-017-0080-1>
- Chang, J., Wang, L., Meng, G., Xiang, S., & Pan, C. (2017). Deep Adaptive Image Clustering. IEEE International Conference on Computer Vision (ICCV), 5880-5888. <https://doi.org/10.1109/ICCV.2017.626>
- Chauhan, A. K., & Krishan, P. (2013). Moving object tracking using gaussian mixture model and optical flow. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 243-246.
- Chen, J., Xu, Y., Zhang, C., Xu, Z., Meng, X., & Wang, J. (2019). An Improved Two-stream 3D Convolutional Neural Network for Human Action Recognition. International Conference on Automation and Computing (ICAC), 1-6. <https://doi.org/10.23919/ICoAC.2019.8894962>
- Cheng, K., Zhang, Y., He, X., Chen, W., Cheng, J., & Lu, H. (2020). Skeleton-Based Action Recognition With Shift Graph Convolutional Network. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 180-189. <https://doi.org/10.1109/CVPR42600.2020.00026>
- Dalal, N., Triggs, B., & Schmid, C. (2006). Human Detection Using Oriented Histograms of Flow and Appearance. In A. Leonardis, H. Bischof, & A. Pinz, *Computer Vision – ECCV 2006 European Conference on Computer Vision*, Berlin, Heidelberg, 428-441.
- Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). *Language modeling with gated convolutional networks* Proceedings of the 34th International Conference on Machine Learning - Volume 70, Sydney, NSW, Australia.
- Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 4685-4694. <https://doi.org/10.1109/CVPR.2019.00482>
- Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020). Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet of Things Journal*, 7(8), 7457-7469. <https://doi.org/10.1109/JIOT.2020.2984887>
- Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & Darrell, T. (2017). Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 677-691. <https://doi.org/10.1109/TPAMI.2016.2599174>
- Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., . . . Brox, T. (2015). FlowNet: Learning Optical Flow with Convolutional Networks. IEEE International Conference on Computer Vision (ICCV), 2758-2766. <https://doi.org/10.1109/ICCV.2015.316>
- Fang, H. S., Xie, S., Tai, Y. W., & Lu, C. (2017). RMPE: Regional Multi-person Pose Estimation. IEEE International Conference on Computer Vision (ICCV), 2353-2362. <https://doi.org/10.1109/ICCV.2017.256>
- Farneback, G. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. In J. Bigun & T. Gustavsson, *Image Analysis Scandinavian Conference on Image Analysis*, Berlin, Heidelberg, 363-370. https://doi.org/10.1007/3-540-45103-X_50

- Feichtenhofer, C. (2020). X3D: Expanding Architectures for Efficient Video Recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 200-210. <https://doi.org/10.1109/CVPR42600.2020.00028>
- Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). SlowFast Networks for Video Recognition. IEEE/CVF International Conference on Computer Vision (ICCV), 6201-6210. <https://doi.org/10.1109/ICCV.2019.00630>
- Feichtenhofer, C., Pinz, A., & Wildes, R. P. (2016). *Spatiotemporal residual networks for video action recognition* International Conference on Neural Information Processing Systems, Barcelona, Spain.
- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Trans Pattern Anal Mach Intell*, 32(9), 1627-1645. <https://doi.org/10.1109/TPAMI.2009.167>
- Gammulle, H., Denman, S., Sridharan, S., & Fookes, C. (2017). Two Stream LSTM: A Deep Fusion Framework for Human Action Recognition. IEEE Winter Conference on Applications of Computer Vision (WACV), 177-186. <https://doi.org/10.1109/WACV.2017.27>
- Gao, Y., Beijbom, O., Zhang, N., & Darrell, T. (2016). Compact Bilinear Pooling. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 317-326. <https://doi.org/10.1109/CVPR.2016.41>
- Gibson, J. J. (1950). *The perception of the visual world*. Houghton Mifflin.
- Girshick, R. (2015). Fast R-CNN. International Conference on Computer Vision (ICCV), 1440-1448. <https://doi.org/10.1109/ICCV.2015.169>
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Computer Vision and Pattern Recognition, 580-587. <https://doi.org/10.1109/CVPR.2014.81>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Gorelick, L., Blank, M., Shechtman, E., Irani, M., & Basri, R. (2007). Actions as Space-Time Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12), 2247-2253. <https://doi.org/10.1109/TPAMI.2007.70711>
- Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., & Saenko, K. (2013). YouTube2Text: Recognizing and Describing Arbitrary Activities Using Semantic Hierarchies and Zero-Shot Recognition. IEEE International Conference on Computer Vision, 2712-2719. <https://doi.org/10.1109/ICCV.2013.337>
- Guo, G., & Lai, A. (2014). A survey on still image based human action recognition. *Pattern Recognition*, 47(10), 3343-3361. <https://doi.org/10.1016/j.patcog.2014.04.018>
- Gupta, P., Thatipelli, A., Aggarwal, A., Maheshwari, S., Trivedi, N., Das, S., & Sarvadevabhatla, R. K. (2021a). Quo Vadis, Skeleton Action Recognition? *International Journal of Computer Vision*, 129(7), 2097-2112. <https://doi.org/10.1007/s11263-021-01470-y>
- Gupta, R., Reebadiya, D., & Tanwar, S. (2021b). 6G-enabled Edge Intelligence for Ultra - Reliable Low Latency Applications: Vision and Mission. *Computer Standards & Interfaces*, 77, 103521. <https://doi.org/10.1016/j.csi.2021.103521>

- Hao, Y., Wang, J., Liu, Y., Xu, Z., & Fan, J. (2017). Extracting Spatio-Temporal Texture signatures for crowd abnormality detection. *International Conference on Automation and Computing (ICAC)*, 1-5. <https://doi.org/10.23919/ICoAC.2017.8082051>
- Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *IEEE International Conference on Computer Vision (ICCV)*, 2980-2988. <https://doi.org/10.1109/ICCV.2017.322>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. <https://doi.org/10.1109/CVPR.2016.90>
- Herath, S., Harandi, M., & Porikli, F. (2017). Going deeper into action recognition: A survey. *Image and Vision Computing*, 60, 4-21. <https://doi.org/10.1016/j.imavis.2017.01.010>
- Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L. C., Tan, M., . . . Le, Q. (2019). Searching for MobileNetV3. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 1314-1324. <https://doi.org/10.1109/ICCV.2019.00140>
- Huang, G., Liu, Z., Maaten, L. V. D., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261-2269. <https://doi.org/10.1109/CVPR.2017.243>
- Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1647-1655. <https://doi.org/10.1109/CVPR.2017.179>
- Ji, S., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. *IEEE Trans Pattern Anal Mach Intell*, 35(1), 221-231. <https://doi.org/10.1109/TPAMI.2012.59>
- Jiang, G., Jiang, X., Fang, Z., & Chen, S. (2021). An efficient attention module for 3d convolutional neural networks in action recognition. *Applied Intelligence*, 51(10), 7043-7057. <https://doi.org/10.1007/s10489-021-02195-8>
- Jiang, J., Deng, C., & Cheng, X. (2017). Action prediction based on dense trajectory and dynamic image. *Chinese Automation Congress (CAC)*, 1175-1180. <https://doi.org/10.1109/CAC.2017.8242944>
- Jin, S., Su, H., Stauffer, C., & Learned-Miller, E. (2017). End-to-End Face Detection and Cast Grouping in Movies Using Erdős-Rényi Clustering. *International Conference on Computer Vision (ICCV)*, 5286-5295. <https://doi.org/10.1109/ICCV.2017.564>
- Ju, S., Xiao, W., Shuicheng, Y., Cheong, L. F., Chua, T. S., & Jintao, L. (2009). Hierarchical spatio-temporal context modeling for action recognition. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2004-2011. <https://doi.org/10.1109/CVPR.2009.5206721>
- Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., . . . Natsev, P. (2017). The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

- Kieu, T., Vo, B., Le, T., Deng, Z.-H., & Le, B. (2017). Mining top-k co-occurrence items with sequential pattern. *Expert Systems with Applications*, 85, 123-133. <https://doi.org/10.1016/j.eswa.2017.05.021>
- Klaeser, A., Marszalek, M., & Schmid, C. (2008). A Spatio-Temporal Descriptor Based on 3D-Gradients. British Machine Vision Association, 99.91-99.10. <https://doi.org/10.5244/C.22.99>
- Kocabas, M., Athanasiou, N., & Black, M. J. (2020). VIBE: Video Inference for Human Body Pose and Shape Estimation. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 5252-5262. <https://doi.org/10.1109/CVPR42600.2020.00530>
- Kong, Y., & Fu, Y. (2016). Action Recognition and Human Interaction. In Y. Fu (Ed.), *Human Activity Recognition and Prediction* (pp. 23-48). Springer International Publishing.
- Kortli, Y., Jridi, M., Al Falou, A., & Atri, M. (2020). Face Recognition Systems: A Survey. *Sensors*, 20(2). <https://doi.org/10.3390/s20020342>
- Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv:1806.08342*.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*,
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90. <https://doi.org/10.1145/3065386>
- Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: A large video database for human motion recognition. 2011 International Conference on Computer Vision, 2556-2563. <https://doi.org/10.1109/ICCV.2011.6126543>
- Kurzak, J., Tomov, S., & Dongarra, J. (2012). Autotuning GEMM Kernels for the Fermi GPU. *IEEE Transactions on Parallel and Distributed Systems*, 23(11), 2045-2057. <https://doi.org/10.1109/TPDS.2011.311>
- Laptev, & Lindeberg. (2003). Space-time interest points. IEEE International Conference on Computer Vision, 432-439 vol.431. <https://doi.org/10.1109/ICCV.2003.1238378>
- Laptev, I. (2005). On Space-Time Interest Points. *International Journal of Computer Vision*, 64(2-3), 107-123. <https://doi.org/10.1007/s11263-005-1838-7>
- Laptev, I., Marszalek, M., Schmid, C., & Rozenfeld, B. (2008). Learning realistic human actions from movies. 2008 IEEE Conference on Computer Vision and Pattern Recognition, 1-8. <https://doi.org/10.1109/CVPR.2008.4587756>
- Laue, S., Mitterreiter, M., & Giesen, J. (2020). A Simple and Efficient Tensor Calculus. *AAAI*, 4527-4534. <https://doi.org/10.1609/aaai.v34i04.5881>
- Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324. <https://doi.org/10.1109/5.726791>
- Lee, D.-G., & Lee, S.-W. (2019). Prediction of partially observed human activity based on pre-trained deep representation. *Pattern Recognition*, 85, 198-206. <https://doi.org/10.1016/j.patcog.2018.08.006>

- Lee, K. J. (2021). Chapter Seven - Architecture of neural processing unit for deep neural networks. In S. Kim & G. C. Deka (Eds.), *Advances in Computers* (Vol. 122, pp. 217-245). Elsevier.
- Li, E., Zhou, Z., & Chen, X. (2018). *Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy* Proceedings of the 2018 Workshop on Mobile Edge Communications, Budapest, Hungary. <https://doi.org/10.1145/3229556.3229562>
- Li, L., Mu, X., Li, S., & Peng, H. (2020). A Review of Face Recognition Technology. *IEEE Access*, 8, 139110-139120. <https://doi.org/10.1109/access.2020.3011028>
- Li, L., Socher, R., & Li, F.-F. (2009). Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2036-2043. <https://doi.org/10.1109/CVPR.2009.5206718>
- Li, N., Cheng, X., Zhang, S., & Wu, Z. (2014). Realistic human action recognition by Fast HOG3D and self-organization feature map. *Machine Vision and Applications*, 25(7), 1793-1812. <https://doi.org/10.1007/s00138-014-0639-9>
- Li, W., Wen, L., Chang, M.-C., Lim, S. N., & Lyu, S. (2017). Adaptive RNN Tree for Large-Scale Human Action Recognition. IEEE International Conference on Computer Vision (ICCV), 1453-1461. <https://doi.org/10.1109/ICCV.2017.161>
- Liu, J., Jiebo, L., & Shah, M. (2009). Recognizing realistic actions from videos “in the wild”. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 1996-2003. <https://doi.org/10.1109/CVPR.2009.5206744>
- Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L. Y., & Kot, A. C. (2020a). NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10), 2684-2701. <https://doi.org/10.1109/TPAMI.2019.2916873>
- Liu, M., Liu, H., & Chen, C. (2017). Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68, 346-362. <https://doi.org/10.1016/j.patcog.2017.02.030>
- Liu, P., Wang, J., She, M., & Liu, H. (2011). Human action recognition based on 3D SIFT and LDA model. 2011 IEEE Workshop on Robotic Intelligence In Informationally Structured Space, 12-17. <https://doi.org/10.1109/RIISS.2011.5945790>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science* European Conference on Computer Vision, 21-37. https://doi.org/10.1007/978-3-319-46448-0_2
- Liu, Z., Zhang, H., Chen, Z., Wang, Z., & Ouyang, W. (2020b). Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 140-149. <https://doi.org/10.1109/CVPR42600.2020.00022>
- Lu, X., Yao, H., Zhao, S., Sun, X., & Zhang, S. (2017). Action recognition with multi-scale trajectory-pooled 3D convolutional descriptors. *Multimedia Tools and Applications*, 78(1), 507-523. <https://doi.org/10.1007/s11042-017-5251-3>

Lucas, B. D., & Kanade, T. (1981). *An iterative image registration technique with an application to stereo vision* International joint conference on Artificial intelligence Vancouver, BC, Canada.

Majd, M., & Safabakhsh, R. (2020). Correlational Convolutional LSTM for human action recognition. *Neurocomputing*, 396, 224-229. <https://doi.org/10.1016/j.neucom.2018.10.095>

Mao, W., Liu, M., Salzmann, M., & Li, H. (2021). Multi-level Motion Attention for Human Motion Prediction. *International Journal of Computer Vision*, 129(9), 2513-2535. <https://doi.org/10.1007/s11263-021-01483-7>

Marszalek, M., Laptev, I., & Schmid, C. (2009). Actions in context. IEEE Conference on Computer Vision and Pattern Recognition, 2929-2936. <https://doi.org/10.1109/CVPR.2009.5206557>

Messing, R., Pal, C., & Kautz, H. (2009). Activity recognition using the velocity histories of tracked keypoints. IEEE International Conference on Computer Vision, 104-111. <https://doi.org/10.1109/ICCV.2009.5459154>

Microsoft. (2021). *ONNX Runtime*. <https://onnxruntime.ai/>

Microsoft. (2022). *ONNX Runtime Execution Providers*. <https://onnxruntime.ai/docs/execution-providers/>

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., & Blankevoort, T. (2021). A White Paper on Neural Network Quantization. *arXiv:2106.08295*.

Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4694-4702. <https://doi.org/10.1109/CVPR.2015.7299101>

ONNX. (2021). *Open Neural Network Exchange*. <https://onnx.ai/>

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Antiga, L. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Peltonen, E., Bennis, M., Capobianco, M., Debbah, M., Ding, A., Gil-Castiñeira, F., . . . Kliks, A. (2020). 6G white paper on edge intelligence. *arXiv:2004.14850*.

Peng, X., Zou, C., Qiao, Y., & Peng, Q. (2014). Action Recognition with Stacked Fisher Vectors. *Lecture Notes in Computer Science* European Conference on Computer Vision, 581-595. https://doi.org/10.1007/978-3-319-10602-1_38

Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P., & Schiele, B. (2016). DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4929-4937. <https://doi.org/10.1109/CVPR.2016.533>

Qin, E., Samajdar, A., Kwon, H., Nadella, V., Srinivasan, S., Das, D., . . . Krishna, T. (2020). SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. IEEE International Symposium on High Performance Computer Architecture (HPCA), 58-70. <https://doi.org/10.1109/HPCA47549.2020.00015>

- Qiu, Z., Yao, T., & Mei, T. (2017). Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. *IEEE International Conference on Computer Vision (ICCV)*, 5534-5542. <https://doi.org/10.1109/ICCV.2017.590>
- Ranjan, A., & Black, M. J. (2017). Optical Flow Estimation Using a Spatial Pyramid Network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2720-2729. <https://doi.org/10.1109/CVPR.2017.291>
- Ranjan, A., Hoffmann, D. T., Tzionas, D., Tang, S., Romero, J., & Black, M. J. (2020). Learning Multi-human Optical Flow. *International Journal of Computer Vision*, 128(4), 873-890. <https://doi.org/10.1007/s11263-019-01279-w>
- Ranjan, A., Romero, J., & Black, M. J. (2018). Learning Human Optical Flow. *British Machine Vision Conference*, <https://github.com/anuragranj/humanflow>
- Reddy, K. K., & Shah, M. (2013). Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5), 971-981. <https://doi.org/10.1007/s00138-012-0450-4>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779-788.
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans Pattern Anal Mach Intell*, 39(6), 1137-1149. <https://doi.org/10.1109/TPAMI.2016.2577031>
- Roeder, L. (2017). *Netron, Visualizer for neural network, deep learning, and machine learning models*. <https://github.com/lutzroeder/netron>
- Rogez, G., Weinzaepfel, P., & Schmid, C. (2020). LCR-Net++: Multi-Person 2D and 3D Pose Detection in Natural Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(5), 1146-1161. <https://doi.org/10.1109/TPAMI.2019.2892985>
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536. <https://doi.org/10.1038/323533a0>
- Ryoo, M. S. (2011). Human activity prediction: Early recognition of ongoing activities from streaming videos. *2011 International Conference on Computer Vision*, 1036-1043. <https://doi.org/10.1109/ICCV.2011.6126349>
- Ryoo, M. S., & Aggarwal, J. K. (2010). *UT-Interaction Dataset*. ICPR contest on Semantic Description of Human Activities (SDHA). http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html
- Samek, W., Binder, A., Montavon, G., Lapuschkin, S., & Muller, K. R. (2017). Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Trans Neural Netw Learn Syst*, 28(11), 2660-2673. <https://doi.org/10.1109/TNNLS.2016.2599820>
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4510-4520. <https://doi.org/10.1109/CVPR.2018.00474>
- Sargano, A., Angelov, P., & Habib, Z. (2017). A Comprehensive Review on Handcrafted and Learning-Based Action Representation Approaches for Human Activity Recognition. *Applied Sciences*, 7(1), 110. <https://doi.org/10.3390/app7010110>

- Schuldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: a local SVM approach. *International Conference on Pattern Recognition*, 32-36 Vol.33. <https://doi.org/10.1109/ICPR.2004.1334462>
- Sevilla-Lara, L., Liao, Y., Güney, F., Jampani, V., Geiger, A., & Black, M. J. (2018). On the integration of optical flow and action recognition. *German Conference on Pattern Recognition*, 281-297. https://doi.org/10.1007/978-3-030-12939-2_20
- Shahroudy, A., Liu, J., Ng, T. T., & Wang, G. (2016). NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1010-1019. <https://doi.org/10.1109/CVPR.2016.115>
- Shantaiya, S., Verma, K., & Mehta, K. (2015). Multiple object tracking using Kalman filter and optical flow. *European Journal of Advances in Engineering and Technology*, 2(2), 34-39.
- Shao, H., Qian, S., & Liu, Y. (2020). Temporal Interlacing Network. *AAAI*, <https://doi.org/10.1609/aaai.v34i07.6872>
- Shi, L., Zhang, Y., Cheng, J., & Lu, H. (2019). Skeleton-Based Action Recognition With Directed Graph Neural Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7904-7913. <https://doi.org/10.1109/CVPR.2019.00810>
- Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5), 637-646. <https://doi.org/10.1109/jiot.2016.2579198>
- Shou, Z., Lin, X., Kalantidis, Y., Sevilla-Lara, L., Rohrbach, M., Chang, S. F., & Yan, Z. (2019). DMC-Net: Generating Discriminative Motion Cues for Fast Compressed Video Action Recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1268-1277. <https://doi.org/10.1109/CVPR.2019.00136>
- Sigurdsson, G. A., Russakovsky, O., & Gupta, A. (2017). What Actions are Needed for Understanding Human Actions in Videos? *IEEE International Conference on Computer Vision (ICCV)*, 2156-2165. <https://doi.org/10.1109/ICCV.2017.235>
- Simonyan, K., & Zisserman, A. (2014). Two-Stream Convolutional Networks for Action Recognition in Videos. *NIPS'14 International Conference on Neural Information Processing Systems*, 568-576.
- Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* *International Conference on Learning Representations*, <http://arxiv.org/abs/1409.1556>
- Sipiran, I., & Bustos, B. (2011). Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 27(11), 963-976. <https://doi.org/10.1007/s00371-011-0610-y>
- Smaira, L., Carreira, J., Noland, E., Clancy, E., Wu, A., & Zisserman, A. (2020). A short note on the kinetics-700-2020 human action dataset. *arXiv:2010.10864*.
- Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3), 199-222. <https://doi.org/10.1023/B:STCO.0000035301.49549.88>
- Soomro, K., Zamir, A. R., & Shah, M. (2012). UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *CRCV-TR-12-01*.

- Spinelli, I., Scardapane, S., & Uncini, A. (2021). Adaptive Propagation Graph Convolutional Network. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10), 4755-4760. <https://doi.org/10.1109/TNNLS.2020.3025110>
- Spruyt, V. (2014). The Curse of Dimensionality in classification. *Computer vision for dummies*, 21(3), 35-40. <https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/>
- Sun, D., Yang, X., Liu, M.-Y., & Kautz, J. (2018a). PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8934-8943. <https://doi.org/10.1109/CVPR.2018.00931>
- Sun, J., Mu, Y., Yan, S., & Cheong, L. F. (2010). Activity recognition using dense long-duration trajectories. *IEEE International Conference on Multimedia and Expo*, 322-327. <https://doi.org/10.1109/ICME.2010.5583046>
- Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep High-Resolution Representation Learning for Human Pose Estimation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5686-5696. <https://doi.org/10.1109/CVPR.2019.00584>
- Sun, L., Jia, K., Yeung, D., & Shi, B. E. (2015). Human Action Recognition Using Factorized Spatio-Temporal Convolutional Networks. *IEEE International Conference on Computer Vision (ICCV)*, 4597-4605. <https://doi.org/10.1109/ICCV.2015.522>
- Sun, S., Kuang, Z., Sheng, L., Ouyang, W., & Zhang, W. (2018b). Optical Flow Guided Feature: A Fast and Robust Motion Representation for Video Action Recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1390-1399. <https://doi.org/10.1109/CVPR.2018.00151>
- Szegedy, C., Wei, L., Yangqing, J., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015a). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9. <https://doi.org/10.1109/CVPR.2015.7298594>
- Szegedy, C., Wei, L., Yangqing, J., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015b). Going deeper with convolutions. *Computer Vision and Pattern Recognition (CVPR)*, 1-9. <https://doi.org/10.1109/CVPR.2015.7298594>
- Tao, M., Bai, J., Kohli, P., & Paris, S. (2012). SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm. *Computer Graphics Forum*, 31(2pt1), 345-353. <https://doi.org/10.1111/j.1467-8659.2012.03013.x>
- Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. *IEEE International Conference on Computer Vision (ICCV)*, 4489-4497. <https://doi.org/10.1109/ICCV.2015.510>
- Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A Closer Look at Spatiotemporal Convolutions for Action Recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6450-6459. <https://doi.org/10.1109/CVPR.2018.00675>
- Varol, G., Laptev, I., & Schmid, C. (2018). Long-Term Temporal Convolutions for Action Recognition. *IEEE Trans Pattern Anal Mach Intell*, 40(6), 1510-1517. <https://doi.org/10.1109/TPAMI.2017.2712608>

- Vishwakarma, D. K., & Kapoor, R. (2015). Hybrid classifier based human activity recognition using the silhouette and cells. *Expert Systems with Applications*, 42(20), 6957-6965. <https://doi.org/10.1016/j.eswa.2015.04.039>
- Voigt, P., & Bussche, A. v. d. (2017). *The EU General Data Protection Regulation (GDPR)*. Springer. <https://doi.org/10.1007/978-3-319-57959-7>
- Wan, Y., Yu, Z., Wang, Y., & Li, X. (2020). Action Recognition Based on Two-Stream Convolutional Networks With Long-Short-Term Spatiotemporal Features. *IEEE Access*, 8, 85284-85293. <https://doi.org/10.1109/access.2020.2993227>
- Wang, H., Kläser, A., Schmid, C., & Liu, C.-L. (2013). Dense Trajectories and Motion Boundary Descriptors for Action Recognition. *International Journal of Computer Vision*, 103(1), 60-79. <https://doi.org/10.1007/s11263-012-0594-8>
- Wang, H., & Schmid, C. (2013). Action Recognition with Improved Trajectories. *IEEE International Conference on Computer Vision*, 3551-3558. <https://doi.org/10.1109/ICCV.2013.441>
- Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., . . . Xiao, B. (2021a). Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10), 3349-3364. <https://doi.org/10.1109/TPAMI.2020.2983686>
- Wang, L., Qiao, Y., & Tang, X. (2015). Action recognition with trajectory-pooled deep-convolutional descriptors. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4305-4314. <https://doi.org/10.1109/CVPR.2015.7299059>
- Wang, W., Xu, Y., Xu, Z., Zhang, C., Li, T., Wang, J., & Jiang, H. (2021b). A Detection Method of Electro-bicycle in Elevators Based on Improved YOLO v4. *International Conference on Automation and Computing (ICAC)*, 1-6. <https://doi.org/10.23919/ICAC50006.2021.9594217>
- Weinzaepfel, P., & Rogez, G. (2021). Mimetics: Towards Understanding Human Actions Out of Context. *International Journal of Computer Vision*, 129(5), 1675-1690. <https://doi.org/10.1007/s11263-021-01446-y>
- Wong, T.-T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48(9), 2839-2846. <https://doi.org/10.1016/j.patcog.2015.03.009>
- Wu, G., Mahoor, M. H., Althloothi, S., & Voyles, R. M. (2010). *SIFT-Motion Estimation (SIFT-ME): A New Feature for Human Activity Recognition* Proceedings of the 2010 International Conference on Image Processing, Computer Vision, & Pattern Recognition,
- Wu, W., Kan, M., Liu, X., Yang, Y., Shan, S., & Chen, X. (2017). Recursive Spatial Transformer (ReST) for Alignment-Free Face Recognition. *International Conference on Computer Vision (ICCV)*, 3792-3800. <https://doi.org/10.1109/ICCV.2017.407>
- Xu, H., Das, A., & Saenko, K. (2019a). Two-Stream Region Convolutional 3D Network for Temporal Activity Detection. *IEEE Trans Pattern Anal Mach Intell*, 41(10), 2319-2332. <https://doi.org/10.1109/TPAMI.2019.2921539>

- Xu, Y., Zhang, C., Xu, Z., Zhou, J., Wang, K., & Huang, J. (2019b). A generic parallel computational framework of lifting wavelet transform for online engineering surface filtration. *Signal Processing*, 165, 37-56. <https://doi.org/10.1016/j.sigpro.2019.06.019>
- Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. AAAI,
- Ye, H., Wu, Z., Zhao, R.-W., Wang, X., Jiang, Y.-G., & Xue, X. (2015). *Evaluating Two-Stream CNN for Video Classification* Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, Shanghai, China. <https://doi.org/10.1145/2671188.2749406>
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*,
- Zecha, D., Eggert, C., Einfalt, M., Brehm, S., & Lienhart, R. (2018). *A Convolutional Sequence to Sequence Model for Multimodal Dynamics Prediction in Ski Jumps* International Workshop on Multimedia Content Analysis in Sports, Seoul, Republic of Korea. <https://doi.org/10.1145/3265845.3265855>
- Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars, *Computer Vision – ECCV 2014* European Conference on Computer Vision, Cham, 818-833.
- Zhang, B., Wang, L., Wang, Z., Qiao, Y., & Wang, H. (2016). Real-Time Action Recognition with Enhanced Motion Vector CNNs. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2718-2726. <https://doi.org/10.1109/CVPR.2016.297>
- Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., & Zheng, N. (2017). View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition from Skeleton Data. *IEEE International Conference on Computer Vision (ICCV)*, 2136-2145. <https://doi.org/10.1109/ICCV.2017.233>
- Zhao, J., & Snoek, C. G. M. (2019). Dance With Flow: Two-In-One Stream Action Detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 9927-9936. <https://doi.org/10.1109/CVPR.2019.01017>
- Zhao, L., Tang, P., & Huo, L. (2014a). A 2-D wavelet decomposition-based bag-of-visual-words model for land-use scene classification. *International Journal of Remote Sensing*, 35(6), 2296-2310. <https://doi.org/10.1080/01431161.2014.890762>
- Zhao, L. J., Tang, P., & Huo, L. Z. (2014b). Land-Use Scene Classification Using a Concentric Circle-Structured Multiscale Bag-of-Visual-Words Model. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(12), 4620-4631. <https://doi.org/10.1109/Jstars.2014.2339842>
- Zhu, W., Hu, J., Sun, G., Cao, X., & Qiao, Y. (2016). A Key Volume Mining Deep Framework for Action Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991-1999. <https://doi.org/10.1109/CVPR.2016.219>
- Zhu, Y., Lan, Z., Newsam, S., & Hauptmann, A. (2019). Hidden Two-Stream Convolutional Networks for Action Recognition. *Computer Vision – ACCV 2018*, Cham, 363-378.