

Assessment of Human Actions from Videos Using Deep Residual Convolutional Neural Networks

Authors

Md Shafkat Rahman Farabi

160041002

S. M. Hasibul Haque Himel

160041018

MD. Fakhruddin Gazzali

160041014

Supervised by

Md. Hasanul Kabir, PhD

Professor

Co-supervised by

Md. Bakhtiar Hasan

Lecturer

**A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of
Bachelor of Science in CSE**



**Department of Computer Science and Engineering
Islamic University of Technology
Organization of the Islamic Cooperation (OIC)**

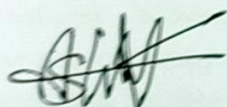
Dhaka, Bangladesh

March, 2021

Declaration of Authorship

This is to certify that the work presented in this thesis, titled, "Assessment of Human Actions from Videos Using Deep Residual Convolutional Neural Networks", is the outcome of the investigation and research carried out by Md Shafkat Rahman Farabi, S. M. Hasibul Haque Himel, MD. Fakhruddin Gazzali, under the supervision of Dr. Md. Hasanul Kabir and Md. Bakhtiar Hasan. It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma, or other qualifications. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:



Md Shafkat Rahman Farabi
Student ID: 160041002

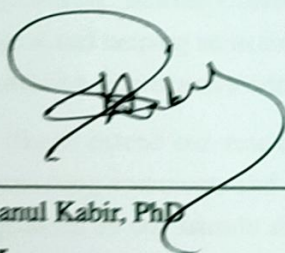


S. M. Hasibul Haque Himel
Student ID: 160041018



MD. Fakhruddin Gazzali
Student ID: 160041014

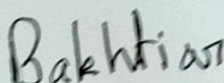
Supervisor:



Md. Hasanul Kabir, PhD
Professor
Department of Computer Science and Engineering
Islamic University of Technology



Co-supervisor:



Md. Bakhtiar Hasan
Lecturer
Department of Computer Science and Engineering
Islamic University of Technology

Acknowledgement

It is an auspicious moment for us to submit our thesis work by which we are eventually going to end our Bachelor of Science study. In the beginning, we want to express our heartfelt gratitude to Almighty Allah for his blessings to bestow upon us which made it possible to complete this thesis research successfully. Without the mercy of Allah, we wouldn't be where we are right now. All thanks and praises be to Allah.

We would like to express our grateful appreciation to Dr. Md. Hasanul Kabir, Professor, Department of Computer Science and Engineering, Islamic University of Technology for being our adviser and mentor. His motivation, suggestions, and insights for this thesis have been invaluable. Without his support and proper guidance, this thesis would not see the path of the proper itinerary of the research world. His valuable opinion, time, and input provided throughout the thesis work, from the first phase of thesis topics introduction, research area selection, the proposition of algorithm, modification, and implementation helped us to do our thesis work properly.

We would also like to thank Md. Bakhtiar Hasan, Lecturer, Department of Computer Science and Engineering, Islamic University of Technology for his enormous help in directing us towards our goal and helping us in time of our need. We are grateful to him for his constant and energetic guidance, constructive criticism, and valuable advice.

We would like to extend our vote of thanks to the jury members of my thesis committee for the many interesting comments and criticism that helped improve this manuscript. Lastly, we are deeply grateful to our friends and family for their unconditional support. This work would have never been completed without the consistent support and encouragement from them throughout the program.

Contents

<i>Declaration of Authorship</i>	i
<i>Acknowledgement</i>	ii
List of Figures	v
List of Tables	vii
<i>Abstract</i>	viii
1 Introduction	1
1.1 Overview of Action Quality Assessment	1
1.2 Significance of AQA	2
1.3 Basic Steps in Performing AQA	2
1.4 Problem Statement	5
1.5 Research Challenges	5
1.6 Research Objectives	7
1.7 Contributions	7
1.8 Organization of the Thesis	8
2 Literature Review	9
2.1 Action Quality Assessment Systems	9
2.1.1 Hand-Crafted Feature-Based AQA Methods	10
2.1.2 Pose and Skeleton Data-Based AQA Methods	11
2.1.3 Deep Feature-Based AQA Methods	13
2.2 Deep CNN Feature Extractors	17
2.2.1 Convolution 3D (C3D) Network	18
2.2.2 Residual Networks (ResNets)	18
2.2.3 Two-Stream ResNets	20
2.2.4 Spatio-Temporal ResNets	22
2.3 Score Predictors	25
2.3.1 Linear Regression	25
2.3.2 Support Vector Regression	26

2.3.3	Long Short-Term Memory	27
3	Proposed Method	30
3.1	General Pipeline Overview	30
3.2	Improving the Feature Extraction	32
3.2.1	34-layer ResNet	36
3.2.2	50-layer ResNet	37
3.2.3	101-layer ResNet	37
3.3	Feature Aggregation with <i>Weight-Decoder</i>	38
4	Result Analysis	45
4.1	Benchmark Dataset: MTL-AQA	45
4.2	Performance Metric	45
4.3	Experimental Setup	46
4.4	Effect of ResNet Depth on the Performance	47
4.5	Effect of Clip Length on Performance	50
4.6	Qualitative Results	51
4.7	Comparison With the State-of-the-Art on the MTL-AQA Dataset	51
5	Conclusion	53
5.1	Summary	53
5.2	Limitations and Future Works	53
	References	55

List of Figures

1.1	Human Action Quality Assessment	2
1.2	Pose information can be used as features for AQA [1]	3
1.3	Extracting Deep Features using CNNs for AQA	4
2.1	Skeleton Data and Pose extracted using motion sensors. [2]	11
2.2	Visualization of pose results on images from FLIC using DeepPose [3]	12
2.3	Architecture of DeepPose [3]	12
2.4	Skeleton model and detection examples extracted using OpenPose [4]	12
2.5	AQA pipeline as proposed by Parmar and Morris [5]	13
2.6	Pseudo-3D (P3D) network based AQA pipeline as proposed by Xiang et al. [6]	14
2.7	AQA approach proposed Zhu et al. [7]	15
2.8	C3D-AVG-MTL architecture as proposed by Parmar and Morris [8]	15
2.9	MSCADC-MTL architecture as proposed by Parmar and Morris [8]	16
2.10	MUSDL architecture for AQA by Tang et al. [9]	17
2.11	Architecture of C3D [10] network	18
2.12	Side by Side comparison of a plain 34-layer CNN (left) and one with residual connections (right). [11]	19
2.13	Two-Stream Approach of Extracting Video Features [12]	21
2.14	Architecture of 3D ResNets [13] network	23
2.15	Performance of ResNets on the Kinetics action recognition dataset. [14]	24
2.16	3D convolution kernel VS (2+1)D convolution kernel [15]	24
2.17	Linear Regression for 2D case	26
2.18	Illustrative example of a Support Vector Regressor fitting a straight line to a set of data points	27
2.19	A Recurrent Neural Network takes input x_t and previous step output h_{t-1} and produce output h_t	27
2.20	x_t can influence h_{t+1} , however x_0 cannot. This is because the contribution of x_0 is suppressed by later time step inputs.	28
2.21	LSTM networks as proposed by Hochreiter and Schmidhuber [16].	28
3.1	Overview of Our Solution Approach	31
3.2	General pipeline of our proposed method for performing AQA	32

3.3	Architecture of C3D [10] network	32
3.4	Poor optimization of C3D-AVG-MTL	34
3.5	Residual Learning Building Block	35
3.6	Spatio-temporally factored convolution kernel	36
3.7	Using ResNets as feature extractor.	37
3.8	The architecture of the WD network,	40
3.9	Proposed Method Architecture with ResNet feature extractor and <i>WD</i> aggregator.	40
3.10	Example: The video being processed by the pipeline. Weights proposed by <i>WD</i>	42
3.11	Example: Weights being normalized.	43
3.12	Example: Final Score Prediction	43
4.1	Train and Test loss curves obtained from training the pipeline using 3D-ResNet 34, 50, and 101 as feature extractors with WeightDecider aggregation.	48
4.2	Train and Test loss curves obtained from training the pipeline using 3D-ResNet 34, 50, and 101 as feature extractors with Average aggregation.	48
4.3	Train and Test Sp correlation curves obtained from training the pipeline using 3D-ResNet 34, 50, and 101 as feature extractors.	49

List of Tables

4.1	Performance comparison of the various types of ResNets as feature extractors in our pipeline	47
4.2	Performance comparison of ResNet(2+1)D-34 with varying input clip size	50
4.3	Qualitative results	51
4.4	Comparison with the State-of-the-Art	52

Abstract

Action quality assessment (AQA) aims at automatically judging human action based on a video of the said action and assigning a performance score to it. Judging the quality of human actions from videos holds huge promise for the future of computer vision. This thesis work focuses on discovering an improved method for action quality assessment. The majority of works in the existing literature on AQA transform RGB videos to higher-level representations using a Convolutional 3D (C3D) network. These higher-level representations are used to perform action quality assessments. Due to the relatively shallow nature of C3D, the quality of extracted features is lower than what could be extracted using a deeper convolutional neural network. Hence, we experiment with deeper convolutional neural networks with residual connections (ResNets) for learning representations for action quality assessment. We assess the effects of the depth and the input clip size of the convolutional neural network on the quality of action score predictions. We also look at the effect of using (2+1)D convolutions instead of 3D convolutions for feature extraction. We think that the current clip-level feature representation aggregation technique of averaging is insufficient to capture the relative importance of features. To overcome this, we propose a learning-based weighted-averaging technique that can perform better. We achieve a new state-of-the-art Spearman's rank correlation of 0.9315 (An improvement of 0.45% over the previous state-of-the-art) on the MTL-AQA dataset using a 34 layer (2+1)D convolutional neural network with the capability of processing 32 frame clips, using our proposed aggregation technique.

Chapter 1

Introduction

In this chapter, we present an overview of our thesis. The overview contains a detailed problem statement and the significance of solving this problem. In the later parts of this chapter, a discussion on the research challenges to be faced is included. Our research motivation, objectives, and our contributions are noted in sections. The end of this chapter contains the description of the organization of the thesis.

1.1 Overview of Action Quality Assessment

Human action or activity refers to human body movement that includes the movement of limbs, joints, or other body parts. Action quality assessment (AQA) addresses the problem of automatically judging the quality of an action performed by a human and assigning a score to it. The target of AQA is to enable computers to quantify how well a human being performed a certain action and provide feedback for improving the body movement of said human so that the performance improves. Usually, the AQA system analyzes an RGB video of the human performing the action and assigns a score. In some cases, additional information in the form of skeleton data and pose is provided through additional sensors.

AQA relies on accurate human motion detection and tracking, segmenting the action from a long video sequence, creating an accurate and dense high-level description of the said action, etc. Currently, human actions are quantified by real-world judges using specific rules. Defining these rules to the computer and analyzing human action using these rules is quite difficult. Different human actions have different human body movement patterns and different rules for assessment. This is why AQA systems are built for specific action classes.

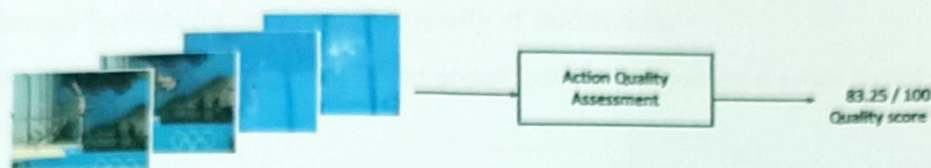


Figure 1.1: Human Action Quality Assessment

1.2 Significance of AQA

In recent years, a lot of works have focused on this problem. Some new datasets have also been introduced. Most of these datasets contain videos of athletes performing some action (diving, snowboarding, etc.) and corresponding scores assigned by human judges. The goal is to design a system that can accurately predict the scores assigned by human judges to the athletes based on the input videos.

An AQA system has potential use in applications such as health care [17], sports video analysis [18], skill discrimination for a specific task [19], assessing the skill of trainees in professions such as surgery [20], driving, swimming, piloting aircraft, etc, video understanding [21]. In healthcare and rehabilitation, physical therapy is a vital recovery step for stroke survivors and sports injury patients. Currently, the patient has to depend on medical professionals and therapy professionals for these treatments. A vision-based rehabilitation system that can evaluate the therapy and provide feedback can offer more economical treatment options. Human action evaluation can also help learners of skills on self-training platforms. For example, simulation-based surgical training platforms have been developed for surgeons, flight simulation exists for pilots, etc. Combining an AQA system with these platforms would enable the trainee to assess their skills as well as receive feedback on how to improve. AQA systems can help automate sports activity scoring. In the case of a lot of sports, replacing a human judge would result in less error. Such a system could also function as a Virtual Assistant Referee (VAR). AQA also has potential usage in video retrieval, as a video retrieval system could rank the videos based on the performed actions and return the best one. These potential use cases were our motivation to engage in research in this field.

1.3 Basic Steps in Performing AQA

Most works implementing AQA do so in 3 stages [1]:

- (i) Detect and segment human action from a long video.
- (ii) Extract features for assessing the quality of said action.
- (iii) Developing a method to assess the action quality and assign a score based on the extracted features.

In this work, we focus more on the later 2 stages as the first one falls under the focus of video action localization.

Based on what type of features are used, AQA systems can be of 3 types:

- **Using Pose Information:** Human action quality is heavily dependant on body movement.

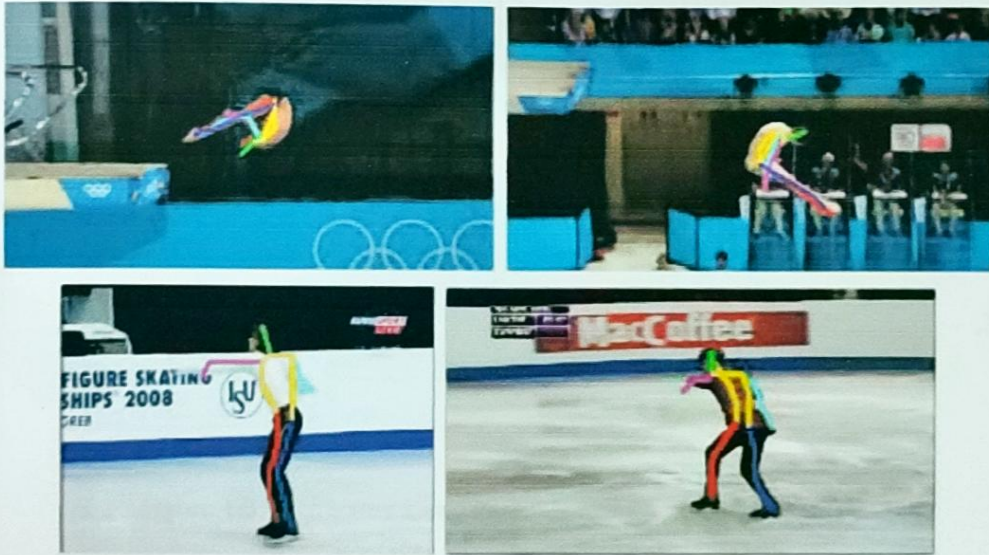


Figure 1.2: Pose information can be used as features for AQA [1]

The movement of limbs and joints contains a lot of information regarding how the action is performed. This information can be achieved using motion capture equipment during the performance of the action. However, in real-world applications, wearing motion capture equipment during action performance is impractical.

Another approach is to extract pose information from the RGB video of said action performance. This is, on the other hand, much more difficult to implement accurately due to the complex body motions the different actions might have. Extracted/ recorded pose information can be transformed using Graph Convolution [22], LSTM [16], DCT [23], etc. into a higher-level representation. Based on this a linear-regressor can predict the quality score.

- Hand Crafted Features:** This approach was popular before the advent of deep learning. Like other disciplines in computer vision, hand-crafted features were extracted from action performance videos. These techniques included but were not limited to Spatial-Temporal Interest Points (STIP) [24], Histogram of Gradients (HOG) [25], Histogram of Optical Flow [26], Scale-Invariant Feature Transform (SIFT) [27], Motion Boundary Histogram (MBH) [28]. These were used to define and extract higher-level features, using which a linear classifier or regressor could predict action quality scores. Optimization techniques such as Bag of Words [29] and Hidden Markov Model [30] were also used to directly formulate the problem of action score prediction as an optimization problem. Using hand-crafted features is computationally inexpensive, but the performance and accuracy of the system are less superior than using skeleton/pose data or directly using RGB video as input to a deep learning framework.
- Deep Feature Method:** Recently, computer vision research has been taking advantage of convolutional neural networks (CNN). CNNs have been successfully used in the domain of image classification [31], object recognition [32], semantic-segmentation [?], etc. In the realm of processing video data, CNNs have also become popular.

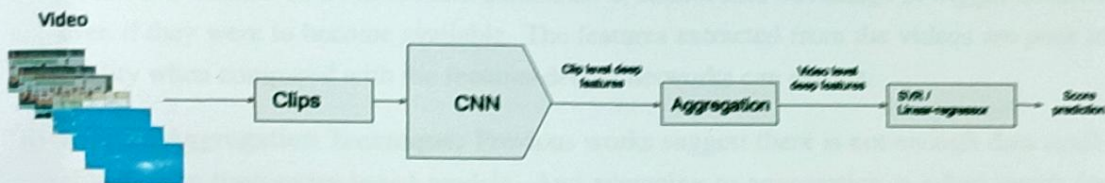


Figure 1.3: Extracting Deep Features using CNNs for AQA

3D convolutional network (C3D) has been successfully used for both action recognition from videos [10] as well as AQA [8]. For AQA, deep convolutional neural networks are used to extract higher-level features from input RGB videos. These higher-level features are used by a Support Vector Regressor (SVR) [33] or by a linear-regressor to make action quality predictions. As shown in figure 1.3, the video is divided into clips. The CNN extracts deep features from individual clips. These clip-level features are then aggregated to form video-level features. These are then used by SVR or linear-regressor for making action score predictions. As this technique has been shown to produce more promising results, we restrict our focus to this approach in this thesis work.

1.4 Problem Statement

The task of developing a good Action Quality Assessment framework has received a lot of attention over the last decade. The more recent approaches have focused more on deep-feature-based approaches with significant improvement in results. The common approach in most works in the literature is to extract features from small clips of the action performance video and later aggregating the features to achieve a global video level description. This is then used as input to a linear regressor or an SVR for predicting action scores. Most works have utilized the C3D network for extracting features. While C3D is a good feature extractor when initialized with pretrained weights from a related big scale dataset, it is relatively shallow when compared to deeper CNNs like ResNets. When it comes to aggregation techniques, the literature has taken one of two paths. One is to average them, another is to use a time series model such as RNN or LSTM. the former is too simple to preserve temporal data significance, the latter requires bigger datasets to train than is currently available. So, the problems we have identified in the approaches taken in the literature are:

- i) **Shallow Feature Extractor:** The C3D [10] used as a feature extractor in the majority of works is a shallow CNN in modern standards. It cannot take advantage of bigger datasets even if they were to become available. The features extracted from the videos are poor in quality when compared with the features deeper networks can extract.
- ii) **Feature Aggregation Technique:** Previous works suggest there is not enough data available to train time-series-based models. And averaging as aggregation is a bad match for video data that has temporal ordering.

These facts present themselves as natural areas of improvement. We aim to solve these problems in this thesis work.

1.5 Research Challenges

Dealing with video data can be challenging in and of itself when compared to image data. To understand why this is the case, let us look at frameworks used to process image data. Usually the images are represented using a 3D tensor of dimensions $C \times H \times W$ (C = Number of channels, H = Height, W = Width). However, most systems that deal with video data also have to look at multiple frames from the input data. Hence the input tensors from video data end up having dimensions $N \times C \times H \times W$ (N = Number of frames). Some earlier works attempted to solve this problem by using the same 2D CNNs to extract frame-level features and then aggregating them through temporal pooling such as averaging or time series mechanisms.

These techniques, while being resource-efficient, were poor at capturing the temporal relation between frames that are close together in time. Some later works attempt at dealing with this problem by dedicating a separate pipeline for performing temporal reasoning from optical flow. Most recent works have however switched to 3D CNNs that take as input small clips made of multiple frames from the input video and perform temporal as well as spatial reasoning. Finally, these clip-level features are pooled temporally to calculate video-level features [7]. Computation resources generally available now permit training 3D CNNs that can operate on small clips containing 8, 16, 32 frames.

To perform Action Quality Assessment accurately, one needs to examine the entire input video. To understand why: imagine an athlete performing a dive. The clip we look at might be the initial phase of the video where the athlete made no mistakes, thus we may conclude the performer should get a good rating. However, the diver might make an error while entering the water. Hence the rating should be much lower than what was initially predicted. What this means is, whatever automated system we design, has to accurately and effectively process all the frames from the entire input video to correctly analyze the athletes' performance. This is especially challenging because analyzing the entire video requires immense memory and processing power. Most works in the literature have opted for dividing the video into non-overlapping clips, processing the clips using 3D CNNs to extract features, and then aggregating features and training a regressor to predict the final score.

Most recent works in computer vision have shifted towards a data-driven approach. Because most recent works depend on deep learning to leverage data available from large-scale datasets. However, building large-scale AQA datasets requires annotating the actions performed. Judging such actions can only be done by domain experts. Human experts of a specific domain have trained for years in that domain to learn complex rules based on which they can assess performance. Thus building large-scale datasets would require annotating every single video by a human apt in his/her domain. As a consequence, even the biggest AQA dataset contains only 1415 samples. Because of the small scale of datasets, designing a deep learning model and training it on the dataset without overfitting is a difficult task.

All the samples in an AQA dataset are very similar (as long as the task being performed is the same). The quality of the action depends on fine-grained details of the action performed. This type of fine-grained action classification is difficult for a deep learning architecture to perform. In other words, any system built for AQA has to deal with poor inter-class discriminant samples and excel at discriminating them.

Although AQA is a new, challenging, and resource-demanding field, the implications of developing a good system that can mimic human judges are massive.

1.6 Research Objectives

- Proposing a novel and improved AQA pipeline that can mimic the judgment of a human judge.
- Studying the effects of deeper convolutional neural network feature extractors on the performance of AQA pipelines in benchmark datasets. Currently, deep CNNs such as ResNets are being used in action recognition tasks. We think they are capable of extracting richer and more complex features from videos due to increased depth. Hence deeper ResNets should improve performance in theory. We aim to confirm this through experimentation.
- Finding out if current big scale datasets hold enough data to train a deep CNN-based AQA pipeline. If so, is there an optimum depth after which going any deeper results in overfitting on the dataset?
- Exploring the effect of the clip-size from which the features are extracted on the accuracy of the final score prediction. Often videos are divided into clips to accommodate them in memory during feature extraction. These features are then aggregated together. We wish to find out if using bigger/smaller clip-size affects the outcome.
- Formulating a better aggregation technique that is also resource-friendly.

1.7 Contributions

- We have demonstrated that 3D and (2+1)D ResNets can be used as good feature extractors for AQA. We have established an optimum depth of the ResNet feature extractor for achieving good results.
- We have shown that number of frames in each clip processed by the feature extractor has a significant impact on the quality of the score predicted by the system. Essentially, a bigger clip-size translates to a significant boost in performance.
- Our proposed aggregation scheme (*WD*) achieves superior performance when compared to averaging as aggregation. Experiments suggest that this performance boost is more significant for smaller clip-sizes and less apparent for bigger clip-sizes.
- Our proposed method outperforms all the previous works on the MTL-AQA dataset.

1.8 Organization of the Thesis

The rest of this thesis is organized as follows:

Chapter 2 provides an overview of different approaches to performing Action Quality Assessment from videos in the published literature. It also outlines various methods of feature extraction from videos that have been used in the existing computer-vision literature.

Chapter 3 includes a detailed explanation of our methods and the design for a better AQA pipeline. It holds a discussion about various feature extractors we have used, how we have incorporated them in our pipeline, our proposed aggregation technique, and how it can improve performance. It contains our proposed framework, methodologies, and other methodologies that we have tested.

Chapter 4 contains an analysis of the results that we have obtained in our experiments, the interpretation of these results, and a comparison with the state-of-the-art.

Chapter 5 presents the conclusion and discusses future works.

Chapter 2

Literature Review

In this chapter, we present a detailed study of AQA systems in the existing literature. We cover different methods used to solve this problem throughout the years. In the next section, we provide a literature review of spatio-temporal feature extractors. Finally, we dedicate a section for various score predictors used in the literature.

2.1 Action Quality Assessment Systems

Human action quality assessment is a rather young field of research in computer vision. This field is somewhat related to the task of action-recognition which has received significantly more focus in the past decades. Computer vision-based systems have been developed for recognizing human actions from videos quite accurately. Recently, the focus has shifted on researching human action quality assessment. This can be attributed to the huge number of applications such a system might have. Gordon and Andrew [34] were the first to propose the problem of automatically assessing human performance from videos. Their work proposed teaching computers to analyze human performance that has been recorded in a video. Ilg et al. [35] attempted to solve a similar problem of estimating the sports skill level of individuals by using 3D trajectory data captured with motion capture gear. Most studies investigating AQA in the early stage [36–41] directly tried to apply and fine-tune state-of-the-art action recognition methodologies to the task of AQA. This was done by either regarding the AQA task as a classification task (good action\bad action) or by changing the classifier with a regressor at the final stage and changing the optimization function. Needless to say, these approaches failed to perform well. The reason behind this is the increased difficulty of performing AQA when compared to action recognition.

From the perspective of what type of features are being used for performing AQA, methodologies can be divided into three categories. These categories are:

- (i) Hand-crafted feature representation methods

- (ii) Pose estimation and skeleton data based methods
- (iii) Deep feature representation methods

2.1.1 Hand-Crafted Feature-Based AQA Methods

Before the era of deep learning, hand-crafted features were used by computer-vision researchers to identify patterns in images and videos. Action quality assessment is no exception to this practice. Some popular handcrafted features are Spatial-Temporal Interest Points (STIP) [24], Histogram of Gradients (HOG) [25], Histogram of Optical Flow [26], Scale-Invariant Feature Transform (SIFT) [27], Motion Boundary Histogram (MBH) [28].

Gordon et al. [34] used motion tracking systems to record body positions. Then the authors used the difference in positions in the spatiotemporal dimensions and used these as features to assess human action quality.

Ilg et al. [35] developed an AQA dataset by capturing human motion using 11 cameras and 41 markers. Their proposed dataset contains 14 videos of action performance. They calculated angular velocity, curvature, and torsion of 3D trajectories as initial features. In the following step, a sliding window was used to correlate features extracted from a test video to the features extracted from training videos to mark the most similar video. Then spatio-temporal morphable models learned to model action sequences based on different styles of movements. Finally, a Radial Basis Function (RBF) was trained with expert ratings to predict the final score.

Wnuk and Soatto [42] proposed an AQA dataset comprising of diving videos from the 13th FINA World Championships. Their proposed AQA method began with subtracting background regions to obtain foreground regions. Then they applied Kalman filtering to keep a track of the center of mass of the diver. Then, various pose descriptors, action modeling methods, and classification approaches were utilized to classify the dive types. Experiments described in this work suggest that using SIFT pose features computed from the result of foreground mask application in a fixed square window, divided in a 4×4 spatial bin configuration and 8 bin orientation gave the best results.

Pirsiavash et al. [23] proposed yet another dataset. This dataset contained 309 diving and figure skating videos from various Olympic events and was named *MIT Olympic Scoring Dataset*. Their work was based on both low-level and high-level features. They used STIP as a low-level feature to assess the quality of human action.

Venkataraman et al. [37] computed multivariate approximate entropy and modeled dynamics in distinct body joints as well as cross approximate entropy to model the body joint interactions. Their approach outperformed that of Pirsiavash et al. [23] on the *MIT Olympic Scoring Dataset*.

2.1.2 Pose and Skeleton Data-Based AQA Methods

Skeleton and pose information have been used as features in AQA pipelines quite frequently. These representations are naturally suitable for AQA. This is because action quality can be analyzed based on how human body parts, limbs, and joints move with respect to each other and the background. The main difficulty of this approach, however, is capturing skeleton data. One approach is to capture this data explicitly using motion capture hardware. For this to work, the action performers have to wear these motion capture systems on various parts of their bodies while performing the actions of interest. This is not practical in real-world applications. Due to the improvement in depth-sensing 3D cameras (Microsoft Kinect, Asus Xtion), it is now possible to capture motion and skeleton data directly without making the action performers wear any tools. This has made capturing skeleton features more and more feasible. As a result of this, skeleton data-based approaches in computer-vision have become popular. Despite this success, there are limitations to such technology. These cameras are heavily susceptible to occlusion and the surrounding environment.

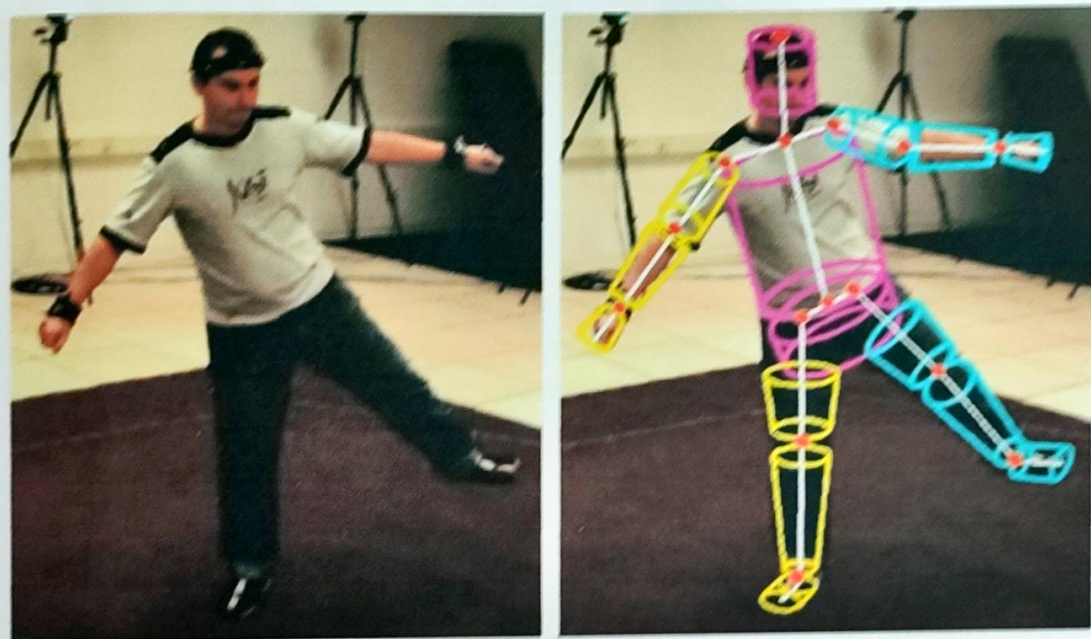


Figure 2.1: Skeleton Data and Pose extracted using motion sensors. [2]

Recently, methods to extract pose information from RGB video have been developed. These are popularly known as pose estimation techniques. Such a technique called DeepPose was proposed by Toshev and Szegedy [3].

This work attempted to estimate the precise position of human body joints to estimate pose. The authors used a two-step process. In the first step, they used a seven-layer convolutional neural network to predict the location of body joints using regression. In the second step, the



Figure 2.2: Visualization of pose results on images from FLIC using DeepPose [3]

regressors try to refine their prediction by using higher-resolution subsets of the original image. DeepPose outperformed the previous hand-crafted pose estimation techniques and remains a baseline for DNN based pose estimation techniques at the time of this writing.

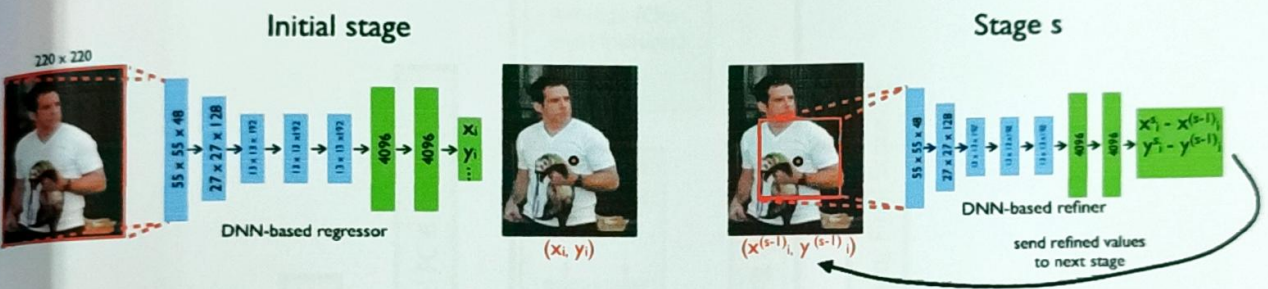


Figure 2.3: Architecture of DeepPose [3]

OpenPose [4] developed in Carnegie Mellon University’s perceptual computing lab, is the first real-time multi-person pose estimation system. This work applies two CNNs as well as a part affinity field to encode the coordinates and orientation of the limbs. This system can handle 2D multiperson and 3D single person pose estimation in real-time.

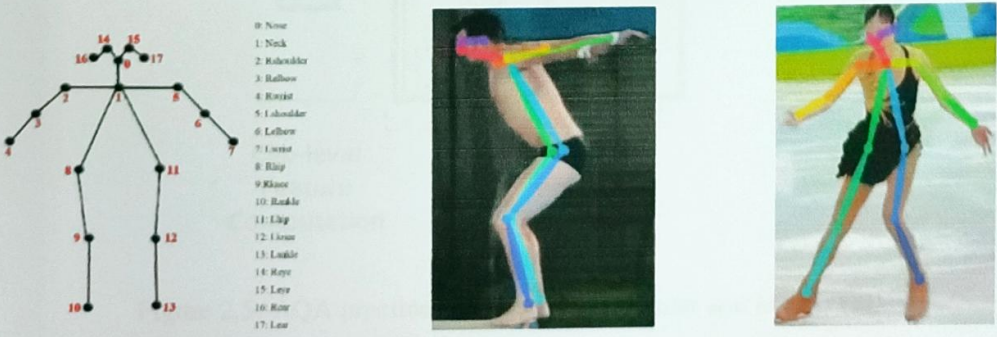


Figure 2.4: Skeleton model and detection examples extracted using OpenPose [4]

Despite these advances, estimated skeleton data can contain noise due to occlusion and background clutter in most real-world use-cases. To obtain good action quality estimation, this noise has to be minimized. This is done through noise filtering.

Paiement et al. [40] extracted skeleton data using a Microsoft Kinect. Then the skeleton data is preprocessed using scaling and spatial alignment. Then diffusion maps are applied for dimensionality reduction of the feature vectors. The method is then tested on gait data of people walking on stairs.

2.1.3 Deep Feature-Based AQA Methods

There has been a trend of using deep features extracted using deep convolutional neural networks. This is not unlike most fields of computer vision.

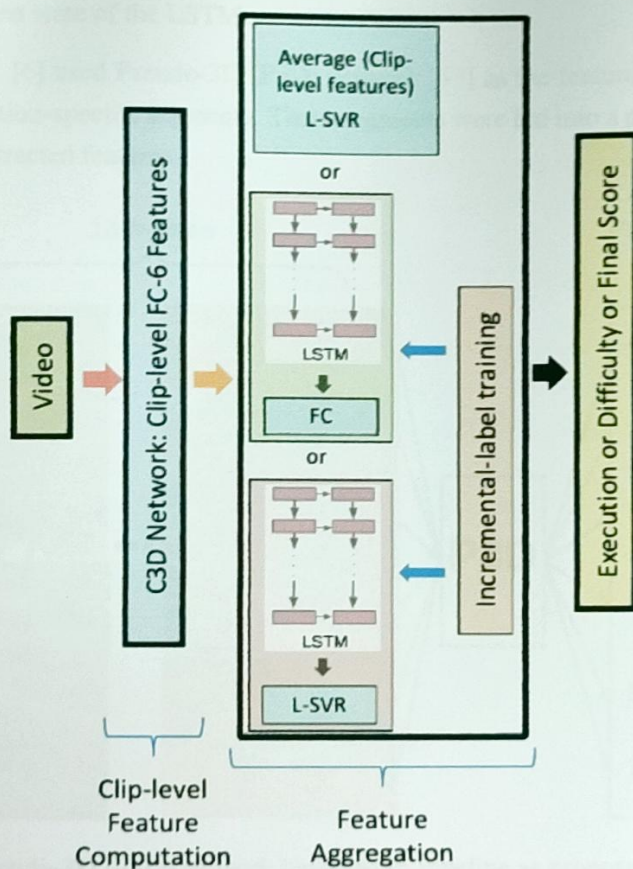


Figure 2.5: AQA pipeline as proposed by Parmar and Morris [5]

Deep neural networks can learn to extract complex patterns and high-level features in an efficient manner. Especially the Convolutional 3D (C3D) network, proposed by Tran et al. [10], has been used as a feature extractor in several recent works. This is not surprising as C3D has been shown to be very successful at capturing salient motion cues and appearance through 3D

spatio-temporal convolutions on the related task of action recognition.

Parmar and Morris [5] proposed three architectures. These are: C3D-SVR, C3D-LSTM, C3D-LSTM-SVR. All of these architectures divided the video into non-overlapping contiguous clips. These clips were then processed by the C3D network and features were extracted. These features were later aggregated for predicting an action score. For C3D-SVR this aggregation was done using averaging. Then a Support Vector Regressor (SVR) predicted action quality scores based on these aggregated features. In C3D-LSTM, the aggregation was done by using an LSTM [16]. The LSTM regarded the features extracted from the clips to be part of a time series. The final hidden state of the LSTM was then used by a fully connected neural network layer to predict the score. In the case of C3D-LSTM-SVR, an SVR predicted the score based on the final hidden state of the LSTM.

Xiang et al. [6] used Pseudo-3D (P3D) network [43] as the feature extractor. They broke the video into action-specific segments. These segments were fed into a pretrained P3D network which output extracted features.

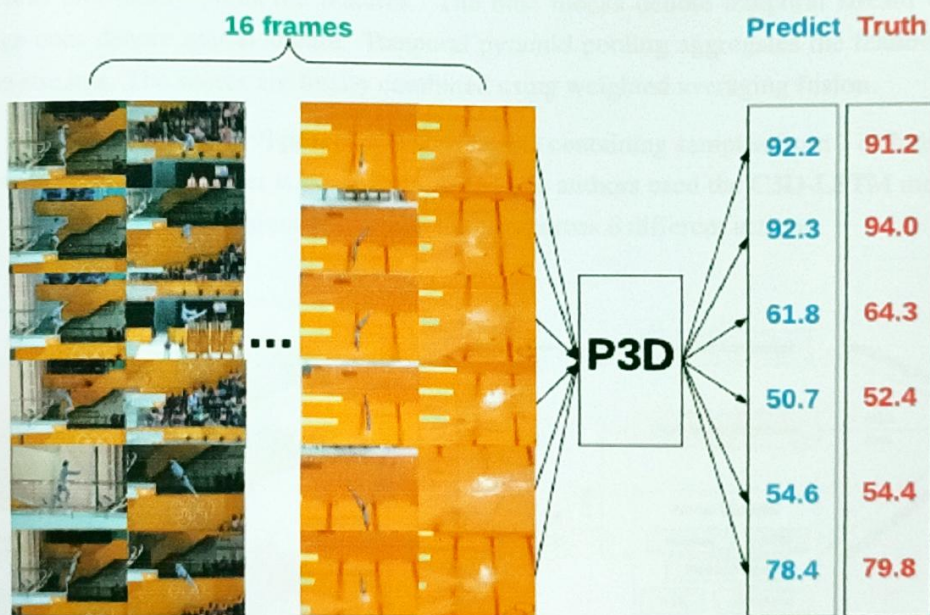


Figure 2.6: Pseudo-3D (P3D) network based AQA pipeline as proposed by Xiang et al. [6]

These features are then processed using a fully connected regression layer, SVR, or a linear-regressor to predict the final action quality score.

Li et al. [7] used 9 different C3D networks to process 9 different clips corresponding to the different stages of diving. These features are aggregated and processed through convolutional and fully-connected layers to produce a final AQA score.

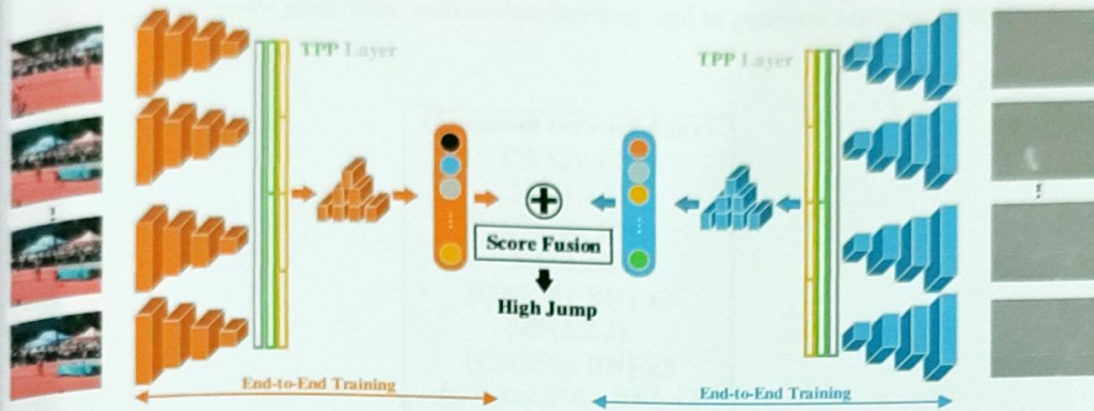


Figure 2.7: AQA approach proposed Zhu et al. [7]

As shown in figure 2.8, the proposed method extracts features from each clip using deep networks and finally pools the features. The blue blocks denote temporal stream while the orange ones denote spatial stream. Temporal pyramid pooling aggregates the features in both of the streams. The scores are finally combined using weighted averaging fusion.

Parmar and Morris [19] proposed a new dataset containing samples from 7 different scores to see if knowledge transfer is possible in AQA. The authors used the C3D-LSTM model from their prior work [5] and trained it to predict scores across 6 different actions.

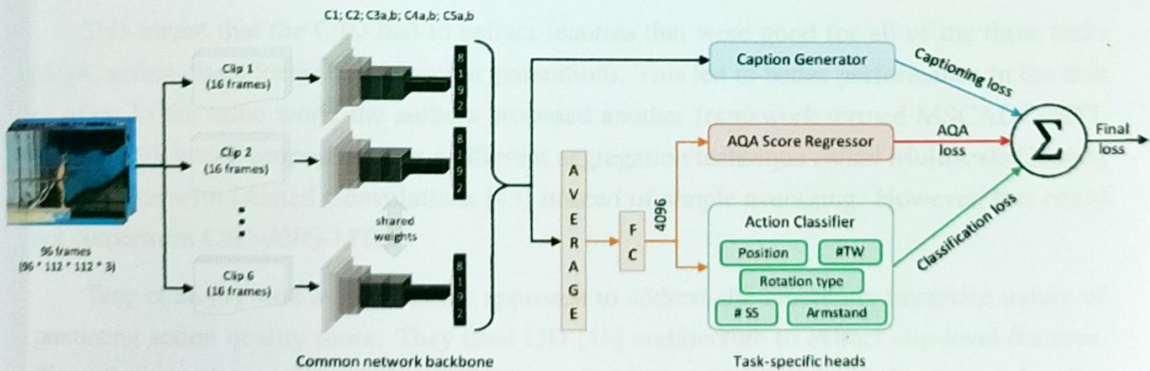


Figure 2.8: C3D-AVG-MTL architecture as proposed by Parmar and Morris [8]

In a later work, Parmar and Morris [8] took a multitask approach towards action quality assessment. Here they released a novel AQA dataset called MTL-AQA. Their proposed multi-task learning-based C3D-AVG-MTL framework (Figure 2.8) outperformed all previous works on the MTL-AQA dataset. Their approach was to extract features using a C3D network and then to aggregate these features through averaging. These averaged video level features were used

by 3 task dependant heads containing linear-regressor, Softmax function, and Gated Recurrence Unit [44] to do score prediction, action classification, and to generate captions in that order.

<i>(Common network body)</i>		
C3(32); BN		
MP(1,2,2)		
C3(64); BN		
MP(2,2,2)		
{C3(128); BN} x2		
MP(2,2,2)		
{C3(256); BN} x2		
{C3(d=2,256); BN} x2		
Dropout(0.5)		
<i>(Task-specific heads)</i>		
<i>(AQA Score Head)</i>	<i>(Action recognition Head)</i>	<i>(Captioning Head)</i>
C1(12)	C1(12)	C1(12)
{Cntxt net}	{Cntxt net}	{Cntxt net}
MP(2,2,2)	MP(2,2,2)	MP(2,2,2)
C3(12); BN	C3(12); BN	C3(12); BN
C3(1)	<i>(Action recognition sub-heads)</i>	Enc. GRU
AP(2,11,11)		Dec. GRU

Figure 2.9: MSCADC-MTL architecture as proposed by Parmar and Morris [8]

This meant that the C3D had to extract features that were good for all of the three tasks (AQA, action classification, and caption generation). This led to better performance in the task of AQA. In the same work, the authors proposed another framework termed MSCADC-MTL (Figure 2.9), which proposed using a different aggregation technique called Multiscale Context Aggregation with Dilated Convolutions [45] instead of simple averaging. However, this could not outperform C3D-AVG-MTL.

Tang et al. [9] took a probabilistic approach to address the inherently uncertain nature of predicting action quality score. They used I3D [46] architecture to extract clip-level features. The authors used a special ordering technique to divide the video into multiple non-overlapping clips. The I3D network then extracted features from these clips. After averaging the clips for aggregation, these video-level features were used to predict parameters of a probabilistic distribution form which the final score prediction was sampled (Figure 2.10). Usually, there are 7 judges scoring Olympic dives. Normally, the middle 5 scores are added together and multiplied by difficulty degree to attain a final score. Tang et al. [9] tried to model 7 different probabilistic models to mimic these 7 judges. They then sample 7 scores and added the middle 5 scores and multiplied with difficulty degree to achieve the final score.

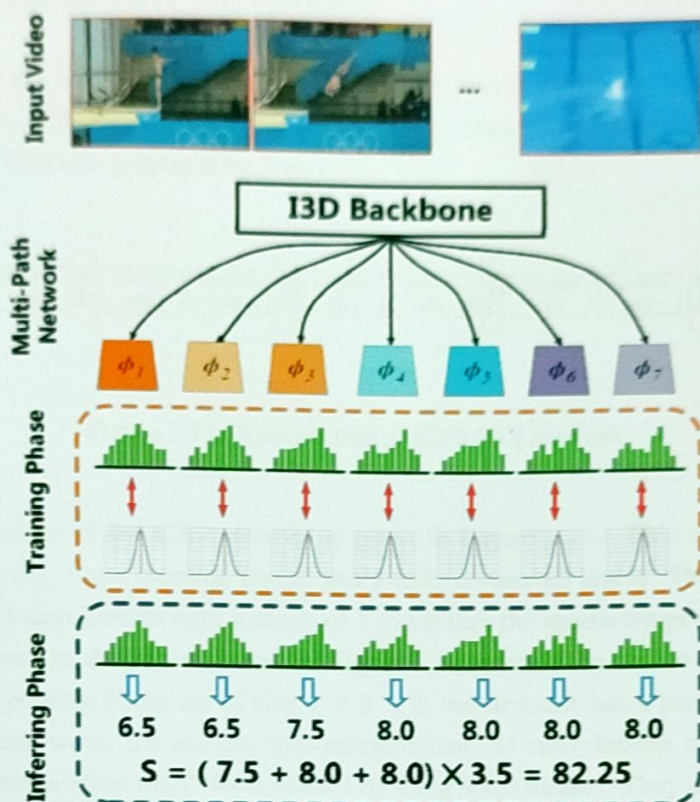


Figure 2.10: MUSDL architecture for AQA by Tang et al. [9]

Our proposed approach differs from these works in that we plan to use 3D and (2+1)D ResNets [13, 15] as feature extractor and we aggregate these features using the *WD* network, which is a light-weight and learning-based feature aggregation scheme.

2.2 Deep CNN Feature Extractors

Processing video data to encode them into higher-level representations is a very important part of computer vision. The specific problem of action-recognition from videos has received significant attention in this case. Initially, this domain was dominated by the tracking of spatiotemporal points [25, 28, 47, 48]. However, with the emergence of deep-learning and the introduction of large-scale datasets, deep convolutional neural networks [49] have become the default choice when it comes to extracting video representations. Here, the most popular architectures are 2D CNNs (2 stream approach using RGB frame and optical flow [50, 51]), Temporal-Segment-Networks [52] and 3D CNNs (C3D [10], I3D [46], P3D [43], (2+1)D CNN [15]).

2.2.1 Convolution 3D (C3D) Network

3D convolutions are becoming the most popular choice with the improvement of processing power and GPUs. C3D [10] network can process video clips of 16 frames to find higher-level representations. This was proposed by Tran et al. [10].

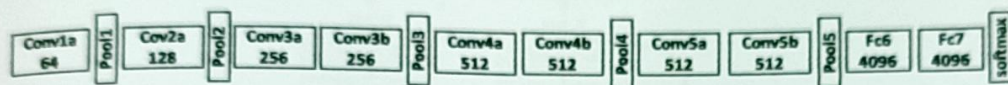


Figure 2.11: Architecture of C3D [10] network

The architecture of the C3D network is given in Figure 2.11. This architecture has 8 convolutional layers, 5 max-pooling layers, and 2 fully connected layers. The authors utilized $3 \times 3 \times 3$ convolution kernels with a stride of 1 to capture the spatio-temporal patterns in the data. The numbers in the “conv” boxes in Figure 2.11 refer to the number of filters in that layer. The max-pooling filters are of size $2 \times 2 \times 2$, the first one has a pooling filter of size $1 \times 2 \times 2$. In this work, the authors demonstrated that 3D convolutions are more effective than 2D convolutions when the CNN is trained on a big scale dataset. They also experimented with various convolution kernel sizes and showed that $3 \times 3 \times 3$ is the optimum configuration. The authors demonstrated that the features learned by C3D are generalized enough for transfer learning to other related tasks.

2.2.2 Residual Networks (ResNets)

Residual Networks (ResNets) were first proposed by He et al. [11]. It was proposed as a means of going deeper with convolutions without degrading the performance of the deep neural networks. Making neural networks deeper generally improved performance up to a point, provided datasets were made bigger proportionally. However, this relation did not hold for very deep networks. Performance usually degraded after a certain point even if the network was trained on a bigger dataset. He et al. [11] investigated this problem and found that the gradients used for weight updates during back-propagation would explode or vanish if a network became too deep. This prevented the earlier layers of the neural network from learning anything useful.

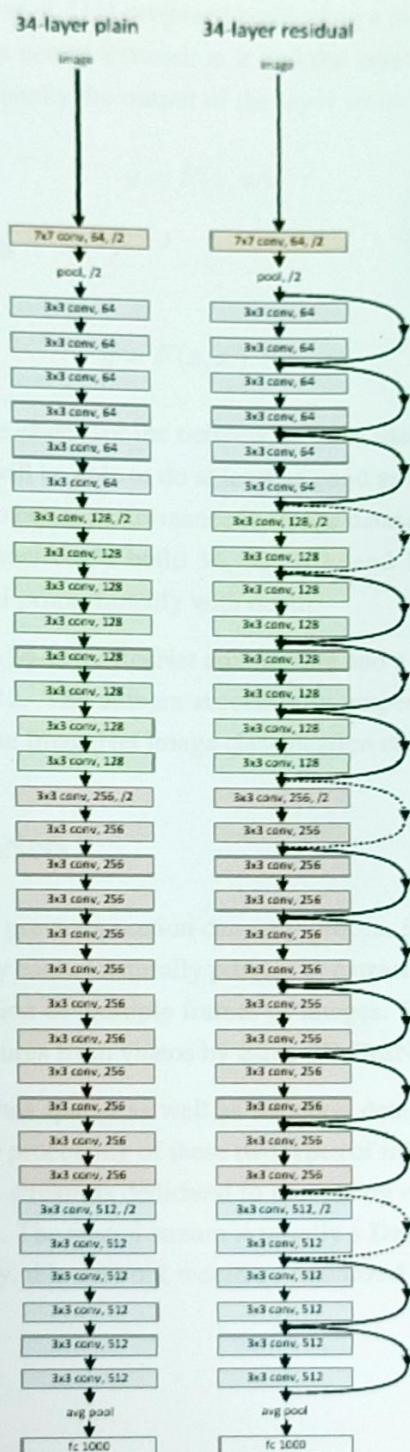


Figure 2.12: Side by Side comparison of a plain 34-layer CNN (left) and one with residual connections (right). [11]

To solve this problem, He et al. [11] proposed employing a technique called residual learning. If the input to a layer in a neural network is x and the layer is represented using F with learnable parameters w , then usually the output of the layer would be:

$$y = F(x, w) \quad (2.1)$$

He et al. [11] replaced this with

$$y = F(x, w) + x \quad (2.2)$$

This meant that it became easier for the network to learn identity mapping. Theoretically, this implied that the network will be able to do at least as good as a shallower one. Experiments showed that this translated into better performance in image datasets like ImageNet [53]. Using residual connections, the authors could build 34, 50, 101, and 152 layer deep networks and showed performance increased proportionally with depth.

The difference between a 34-layer ResNet architecture and a plain 34-layer CNN architecture can be seen in Figure 2.12. The authors successfully outperformed 19 layer deep VGG-19 [54] using ResNet-34 on the ImageNet image classification challenge [53].

2.2.3 Two-Stream ResNets

The ResNets discussed in the previous section can only process images. They can extract features from images. Hence they cannot naturally process or extract features from videos. Videos can be thought of as a collection of multiple frames of images. Early works such as Karpathy et al. [12] tried extracting features from videos by using 2D-ResNets.

A video quite naturally has spatial as well as temporal data encoded in it. A two-stream approach tries to decouple the processing of these two types of information by using two different information streams. One stream is dedicated to processing each frame as an RGB image. This is called a spatial stream. The spatial stream is usually a Deep 2D-ResNet that can extract features from images. Usually, this network would be initialized with pretrained weights from ImageNet [53] dataset.

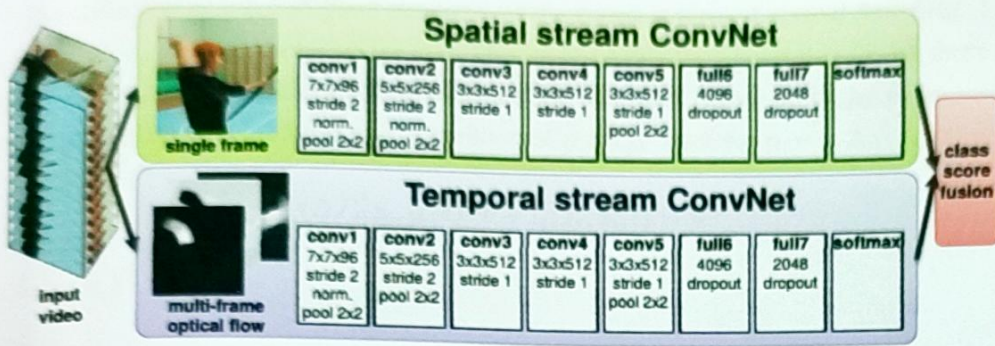


Figure 2.13: Two-Stream Approach of Extracting Video Features [12]

The second stream would be dedicated to processing temporal information present in the data. This is usually done by taking advantage of something called optical-flow [55]. Optical flow assumes that the brightness should be constant from frame to frame if the objects are not moving. Hence, it tries to measure movement using the change in brightness in consecutive frames. The optical-flow calculation assumes that:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \quad (2.3)$$

In other words, it is assumed that for every pixel, the intensity I of a pixel at position (x, y) and at time t should match with the intensity I of some pixel at time $(t + \Delta t)$. *Optical flow vector* is denoted with v and is defined as follows:

$$v = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} dt \quad (2.4)$$

dt can be assumed to be infinitesimally small, which leads to the following approximation in the first order:

$$I(x, y, t) + (\Delta I)(x, y, t)^T v dt + I_t(x, y, t) dt = I(x, y, t) \quad (2.5)$$

This can be simplified to the following expression:

$$(\Delta I)(x, y, t)^T v + I_t(x, y, t) = 0 \quad (2.6)$$

Equation 2.6 can be solved to obtain the optical flow vector v . However, in this case, there are two unknowns (the two components of the vector v) and only one equation. This implies the impossibility of the solution.

The collection of v for all pixel positions of the frame is termed *optical flow field*. Lucas and Kanada [56] propose a solution for this. They assume that for a point $p = (x, y)$, there exist multiple points around p with the same optical flow. Let such points be $p_1, p_2, p_3, p_4, \dots, p_n$. These n points compose the local neighbourhood of point p . Then for $n = 1, 2, \dots, n$:

$$(\Delta I)(x_i, y_i, t)^T v + I_t(x_i, y_i, t) = 0 \quad (2.7)$$

If $v = (v_x, v_y)^T$, we get:

$$I_x(x_i, y_i, t)v_x + I_y(x_i, y_i, t)v_y = -I_t(x_i, y_i, t) \quad (2.8)$$

Stacking all the n equations, we get :

$$\begin{bmatrix} I_x(x_1, y_1, t) & I_y(x_1, y_1, t) \\ I_x(x_2, y_2, t) & I_y(x_2, y_2, t) \\ \vdots & \vdots \\ I_x(x_n, y_n, t) & I_y(x_n, y_n, t) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = - \begin{bmatrix} I_t(x_1, y_1, t) \\ I_t(x_2, y_2, t) \\ \vdots \\ I_t(x_n, y_n, t) \end{bmatrix}$$

$$\implies Av = -b$$

$$\implies v = -(A^T A)^{-1} A^T b$$

Thus, after choosing a neighborhood, the optical-flow field can be calculated. The collection of v for all pixels in the image is known as *optical flow field*. The *optical flow field* calculated from consecutive frames can be thought of as encoding the temporal information in those consecutive frames. Karpathy et al. [12] stacked optical-flow field calculated from L consecutive frames and treated the resulting tensor as a $2L$ channel image (each field vector has a x component and a y component). The resulting $2L$ channel image is input into a 2D CNN that extracts temporal features. Finally, the temporal and spatial stream features are fused together and used for action classification.

2.2.4 Spatio-Temporal ResNets

Spatio-temporal ResNets process the spatio-temporal data using only one stream. This is done by employing spatio-temporal convolutions. That is, the video data is processed by convolving with 3D kernels. This can be thought of as looking for patterns not only in the spatial direction but also in the temporal direction simultaneously.

3D ResNets were popularized by Kensho Hara et al. [13]. The authors replaced 3×3 convolution kernels in the original ResNet in [11] with $3 \times 3 \times 3$ spatio-temporal convolutional kernels. The authors inflated ImageNet [53] weights to 3D and initialised the 3D ResNets

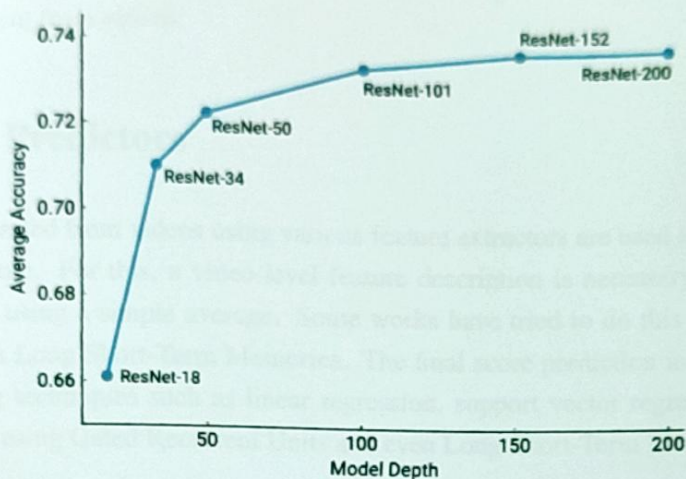


Figure 2.15: Performance of ResNets on the Kinetics action recognition dataset. [14]

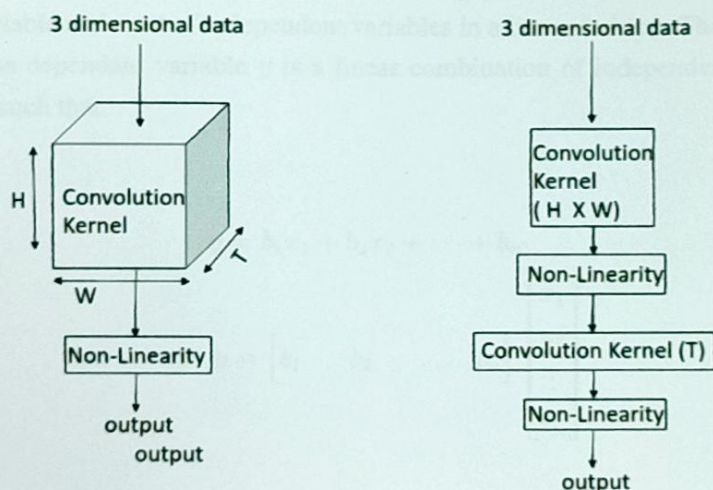


Figure 2.16: 3D convolution kernel VS (2+1)D convolution kernel [15]

These blocks replace the $3 \times 3 \times 3$ spatio-temporal convolutional kernels with the a spatial convolution kernel of size $1 \times 3 \times 3$ followed by a temporal convolution kernel of size $3 \times 1 \times 1$ (Figure 2.16). This provides the ResNet being trained the idea that spatial and temporal dimensions are inherently different. This also increases the number of non-linearities in the neural network. This translates to better performance on the Kinetics action recognition [58] dataset. Open-source implementations and pretrained weights on large-scale action recognition datasets (Kinetics [58], Sports-1M [12], Ig-65M [59]) exist for various depths and input clip

sizes. This makes these networks ideal for transfer learning to a related task such as Action Quality Assessment from videos.

2.3 Score Predictors

The features extracted from videos using various feature extractors are used to predict the final action quality score. For this, a video-level feature description is necessary. Most works in literature do this using a simple average. Some works have tried to do this using time series modeling such as Long Short-Term Memories. The final score prediction is often done using machine learning techniques such as linear regression, support vector regressors, etc. Some works have tried using Gated Recurrent Units and even Long Short-Term Memories.

2.3.1 Linear Regression

Linear Regression is the simplest type of regression analysis. It models the relationship between a dependant variable and a set of independent variables in a linear fashion. That is, this method assumes that the dependant variable y is a linear combination of independent variables (say x_1, x_2, \dots, x_n) such that:

$$y = b_1x_1 + b_2x_2 + \dots + b_n$$

$$\Rightarrow y = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\Rightarrow y = b^T x \quad (2.9)$$

Now, for a given set of datapoints $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$, Linear regression tries to estimate the vector \mathbf{b} such that $y'^{(i)} = b^T x^{(i)}$ is as close to $y^{(i)}$ as possible. More formally, it tries to minimize the mean squared error of the prediction and the real value of y .

$$Error = \frac{1}{m} \sum_{i=1}^m (y'^{(i)} - y^{(i)})^2 \quad (2.10)$$

Linear regression aims at finding the value of \mathbf{b} such that the Error in equation 2.10 is minimized. For two dimensional case, this problem can be thought of as finding the best fitting

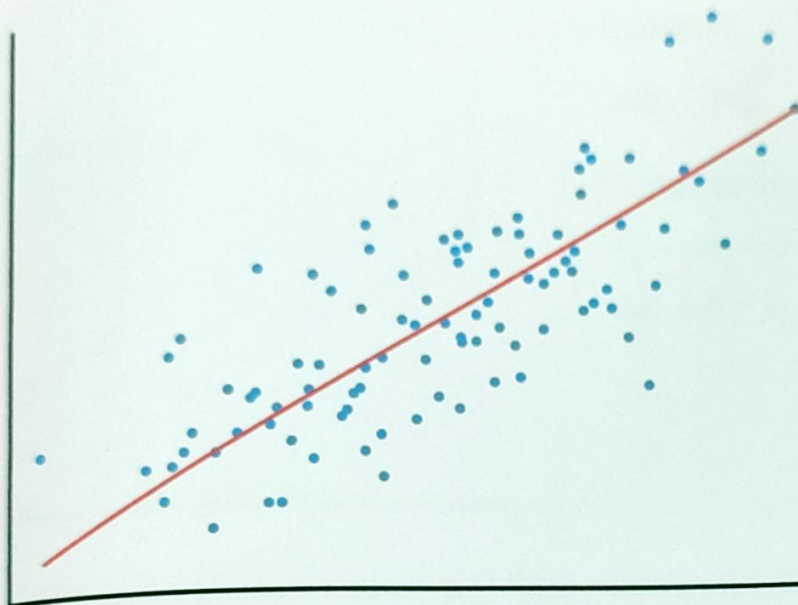


Figure 2.17: Linear Regression for 2D case

straight line that goes through a set of points (Figure 2.17).

2.3.2 Support Vector Regression

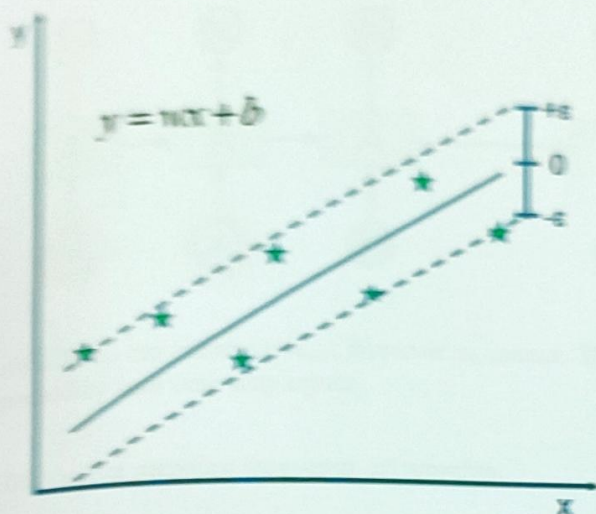
A Support Vector Regressor provides the flexibility to decide how much error is okay to be accepted in the model. This controls the shape of the line/hyperplane that will fit the data. In SVR, the objective function is to minimize the coefficients. In this case, we have to minimize the L2 norm of the coefficient vectors. The absolute error is instructed to be less than a hyper-parameter called ϵ . This can be tuned to gain the desired accuracy of the model. In other words, the objective function that we need to minimize looks like the following:

$$\text{MIN} \frac{1}{2} \|W\|^2 + c \sum_{i=1}^n |\xi_i| \quad (2.11)$$

Given the constraints:

$$|y_i - w_i x_i| \leq \epsilon + |\xi_i| \quad (2.12)$$

Solving these, we get the vector w which defines the hyperplane that fits the data. SVR is a powerful algorithm, as it has the ability to balance the trade-off between generalization and the amount of error on the training data.



• Solution:

$$\min \frac{1}{2} \|w\|^2$$

• Constraints:

$$y_i - wx_i - b \leq \epsilon$$

$$wx_i + b - y_i \leq \epsilon$$

Figure 2.18: Illustrative example of a Support Vector Regressor fitting a straight line to a set of data points

2.3.3 Long Short-Term Memory

A Long Short-Term Memory (LSTM for short) can process sequential data. A traditional neural network produces outputs based on the present input. Previous inputs cannot affect it. Modeling sequential data with a traditional neural network is a challenge for this reason. A recurrent neural network is designed to overcome this challenge by producing the output based on present input and previous output.

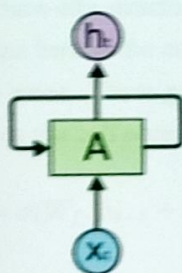


Figure 2.19: A Recurrent Neural Network takes input x_t and previous step output h_{t-1} and produce output h_t .

RNNs can process sequential data. However, the problem of long-term dependency remains. The problem is that inputs from a large number of time steps ago cannot meaningfully influence the output or internal states of much later time steps.

To solve this problem, Hochreiter and Schmidhuber [16] proposed the idea of LSTMs.

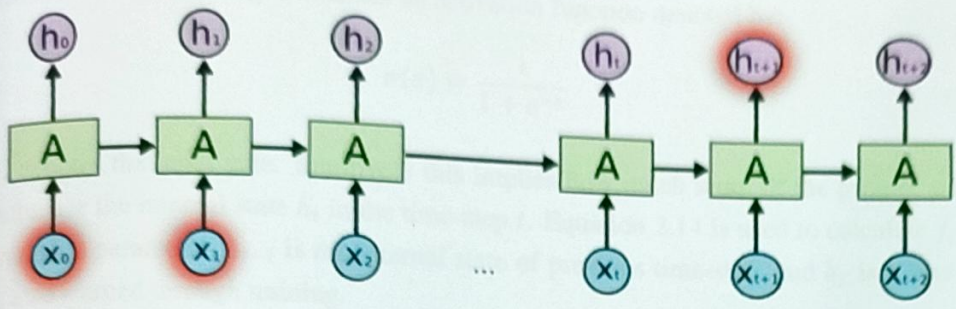


Figure 2.20: x_t can influence h_{t+1} , however x_0 cannot. This is because the contribution of x_0 is suppressed by later time step inputs.

LSTMs excel at retaining long-term information.

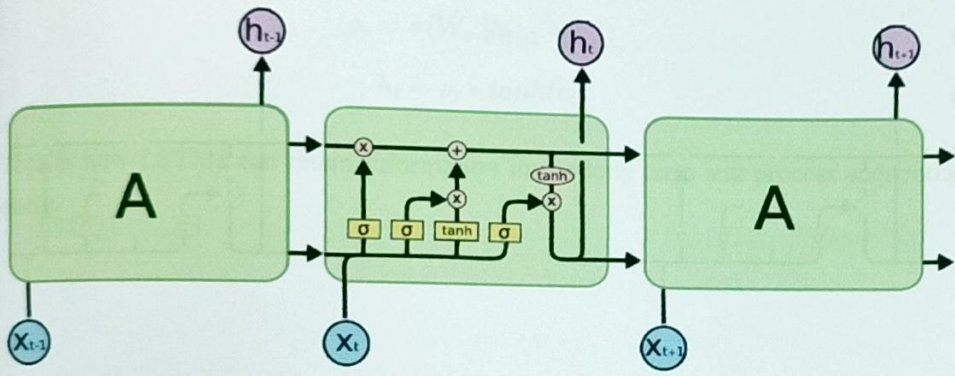


Figure 2.21: LSTM networks as proposed by Hochreiter and Schmidhuber [16].

Like, RNNs, LSTMs also have a chain-like structure. The inner structure of an LSTM cell is presented in Figure 2.21. The key idea here is the cell state C_t , which retains the data from previous states. Another concept is the gates. These gates decide how much new information and how much old information influences the cell state. There are multiple gates.

$$f_t = \sigma(W_f \cdot [h_{t-1} + b_f]) \tag{2.13}$$

$$i_t = \sigma(W_i \cdot [h_{t-1} + b_i]) \tag{2.14}$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1} + b_c]) \tag{2.15}$$

f_t denotes the forget gate. Intuitively this implies how much of the previous state information c_t will the LSTM cell forget on time-step t . Equation 2.13 is used to calculate f_t . W_f is a learnable parameter, h_{t-1} is the internal state of previous time-step, and b_f is a bias term that is

also learned through training. σ denotes an activation function denoted by:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.16)$$

i_t denotes the input gate. Intuitively this implies how much input of the present state x_t will influence the internal state h_t in the time-step t . Equation 2.14 is used to calculate f_t . W_f is a learnable parameter, h_{t-1} is the internal state of previous time-step, and b_f is a bias term that is also learned through training.

\tilde{c}_t is the potential update. W_c, b_c are trainable parameters and biases. The state h_t is calculated using the following set of equations:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (2.17)$$

$$o_t = \sigma(W_o \cdot [h_{t-1} + b_o]) \quad (2.18)$$

$$h_t = o_t * \tanh(c_t) \quad (2.19)$$

This is how LSTMs can retain information in the long term and process sequential data efficiently.

General Pipeline Overview

Designing AQA pipeline in the hierarchical have three parts. A feature extractor, a feature aggregation and a aggregator. We start building an improved AQA pipeline by improving the feature extractor as well as the feature aggregation.

To improve the feature extractor, we experiment with deeper CNNs. Going deeper with convolution usually results in a drop in performance. ResNeXt [11, 13, 15] have been shown to solve this problem by adding residual connections between layers. We, therefore, experiment with using ResNeXt as feature extractor. We experiment with 34, 50, and 101 layer spatial pyramid pooling. We demonstrate with experiments that 34 layer and 50 layer ResNeXt can be good feature extractor for AQA systems when trained on currently available big-scale AQA datasets. We further experiment to see the effect of 3D [13] and (2+1)D [15] convolutions on the performance of the AQA model. We investigate the effect of input clip size as well.

We improve the aggregation technique by introducing a trainable module called Weighted 3D (W3D). The W3D module proposes weights for a weighted average of the clip-level feature vectors. W3D does this based on the importance of the information encoded in the clip-level feature vector. In our experiments, we have found this to be more effective than a simple average of clip-level feature vectors.

Figure 2.1 illustrates our solution approach. Generally, the input video is divided into clips.

Chapter 3

Proposed Method

This chapter describes our proposed methodologies in detail. In the first section, we describe the general approach towards building an AQA pipeline. In the subsequent sections, we explain how we improve upon this system by enhancing the feature extractor and developing an improved aggregation technique.

3.1 Genral Pipeline Overview

Existing AQA pipelines in the literature have three parts. A feature extractor, a feature aggregator, and a regressor. We aim at building an improved AQA pipeline by improving the feature extractor as well as the feature aggregator.

To improve the feature extractor, we experiment with deeper CNNs. Going deeper with convolutions usually results in a drop in performance. ResNets [11, 13, 15] have been shown to solve this problem by adding residual connections between layers. We, therefore, experiment with using ResNets as feature extractors. We experiment with 34, 50, and 101 layers spatio-temporal ResNets. We demonstrate with experiments that 34 layers and 50 layer ResNets can act as good feature extractors for AQA systems when trained on currently available big-scale AQA datasets. We further experiment to see the effect of 3D [13] and (2+1)D [15] convolutions on the performance of the AQA model. We investigate the effect of input clip size as well.

We improve the aggregation technique by introducing a trainable module called *Weight-Decoder (WD)*. The *WD* module proposes weights for a weighted average of the clip-level feature vectors. *WD* does this based on the importance of the information encoded in the clip-level feature vector. In our experiments, we have found this to be more effective than a simple average of the feature vectors.

Figure 3.1 outlines our solution approach. Generally, the input video is divided into clips.

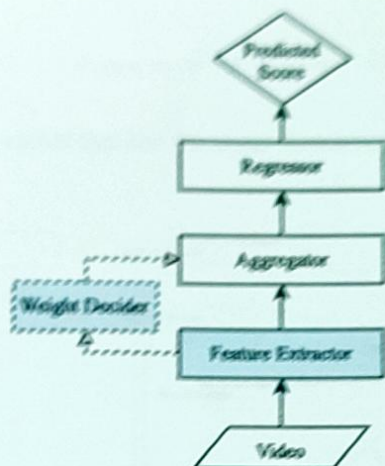


Figure 3.1: Overview of Our Solution Approach

A feature extractor extracts features from these clips. These features are then aggregated into a video-level feature vector. A linear-regressor predicts action quality scores based on this feature vector. We improve the feature extractor by using a ResNet instead of the commonly used C3D. The newly introduced *Weight-Decider* proposes weights based on the clip-level features for better aggregation. The components in light blue color in the figure indicate areas of improvement.

Generally, AQA pipelines deal with short clips of humans performing the action of interest. This is accomplished by dividing the video into non-overlapping clips such that each clip contains N frames. Then each individual clip is input into a feature extractor. This feature extractor extracts higher-level features from the clips. These clip-level features are then temporally pooled to compute video-level features. Finally, the task becomes calculating the final AQA score from these higher-level features.

Mathematically, let's assume the RGB video is divided in N non overlapping clips F_1, F_2, \dots, F_N . The feature extractor g takes in a clip F_i and calculates some features in an m dimensional space denoted by f_i .

$$f_i = g(F_i), \text{ for } i = 1 \dots N \quad (3.1)$$

Using some pooling technique p , video level features f_{video} is calculated

$$f_{video} = p(f_1, f_2, \dots, f_N) \quad (3.2)$$

Following most previous works [5, 8, 19, 23, 43] in this domain, we treat AQA as a regression problem. This makes sense as the action quality score is a real number as opposed to one from a set of discrete values. In other words, the final score prediction is assumed to be a linear combination of the video level features and is modeled as follows:

$$Score = W^T f_{video} + b \tag{3.3}$$

where W is a parameter vector that has the same dimension as f_{video} and b is the bias.

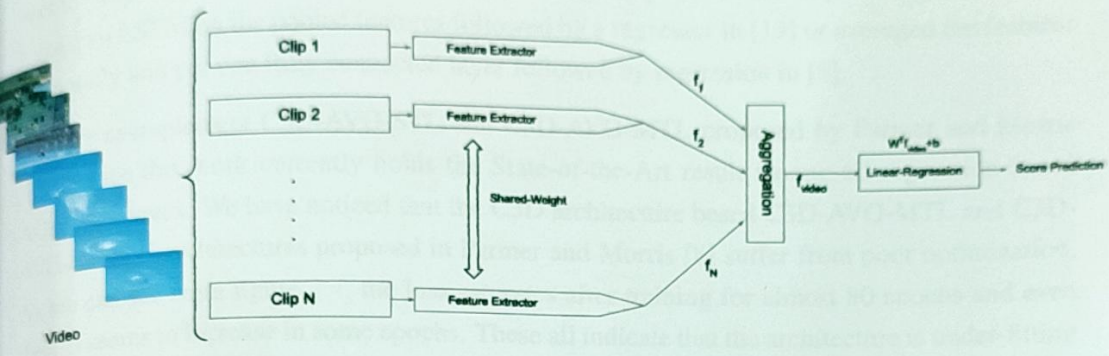


Figure 3.2: General pipeline of our proposed method for performing AQA

3.2 Improving the Feature Extraction

We experiment with various architectures and try to find the best one for this task. Most prior works [5, 8, 19] try to use the C3D [10] network as a feature extractor. Some works experiment with P3D [43]. The C3D network takes in 16 frames of size $3 \times 112 \times 112$, propagates them through 5 convolution layers, each employing 3D convolutional kernels of size $3 \times 3 \times 3$. Finally, it outputs an 8192-dimensional feature vector. Tran et al. [10] showed this architecture is an effective and lightweight feature extractor to use for most video-based works such as *scene and object recognition*, *action similarity assessment*, *action recognition* etc. In the original paper presenting C3D architecture [10], two fully connected layers followed the convolutional layers, and finally, a softmax layer was used for performing classification. Figure 3.3 shows the architecture of C3D as proposed by Tran et al. [10].

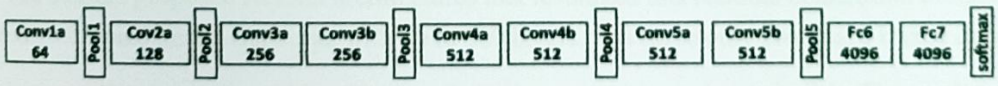


Figure 3.3: Architecture of C3D [10] network

Parmar and Morris used this C3D network (up until the fc6 layer) in a number of recent works in AQA [5, 8, 19]. In all of the aforementioned works, the authors worked on scoring

olympic sport videos. These videos are short and have around 100 frames per sample. The authors divided the videos into 16 frame clips, each having size 112×112 . These RGB clips of size $16 \times 3 \times 112 \times 112$ were then input to the C3D network. The authors collected the output of the final convolutional layer and flattened it to obtain an 8192-dimensional feature vector. The extracted features are aggregated temporally across the clips. Then the authors trained an LSTM on the pooled features followed by a regressor in [19] or averaged the features temporally and put one fully connected layer followed by regression in [8].

We re-implement C3D-AVG-STL and C3D-AVG-MTL proposed by Parmer and Morris [8]. Notice this work currently holds the State-of-the-Art result among all regression based AQA techniques. We have noticed that the C3D architecture based C3D-AVG-MTL and C3D-AVG-STL [8] architectures proposed in Parmer and Morris [8] suffer from poor optimization. As we can see from figure 3.4, the loss saturates after training for almost 80 epochs and even then, it seems to increase in some epochs. These all indicate that the architecture is under-fitting on the training data. Thus we think, to achieve higher performance and better training on the data, the natural approach would be to make the network, especially the convolutional neural network acting as the feature extractor much deeper.

However, making a deep neural network deeper by simply adding more layers usually does not automatically improve performance. As shown by He et al. [?], adding more layers to a deep convolutional neural network can even degrade performance. The authors argue this is due to exploding/vanishing gradient problems. Increasing the number of layers usually means the error gradient has to be backpropagated through all these layers, getting multiplied by a weight matrix at each layer. Soon, these many multiplications can make the gradient either infinitesimally small or exponentially large. Thus preventing the earlier layers from learning anything meaningful. A solution proposed in the same work is residual connections that make learning identity function easy for the deep neural network layers. Thus motivating the network to do at least as well as a shallower counterpart. The authors argued this should give the network an incentive to at least learn the identity mapping. If the network learns to detect some extra patterns, then performance would improve. But residual connections should at least prevent the network from performing worse when it has increased depth.

The authors proposed ResNet architectures that leveraged this residual connection to make deep convolutional neural networks with up to 200 layers. But these only processed image data.

In the video processing realm, Hara et al. [14] proposed something similar. They inflated the 2D ResNets proposed by Tran et al. [11]. To accomplish this, the authors replaced the 3×3 kernels of 2D ResNets with $3 \times 3 \times 3$ spatio-temporal convolution kernels. The authors had access to large scale video datasets such as kinetics-700 [58] which enabled them to train such a deep neural network from scratch without overfitting. The authors demonstrated through experiment that their 34 layers 3d ResNet trained from scratch on kinetics can outper-

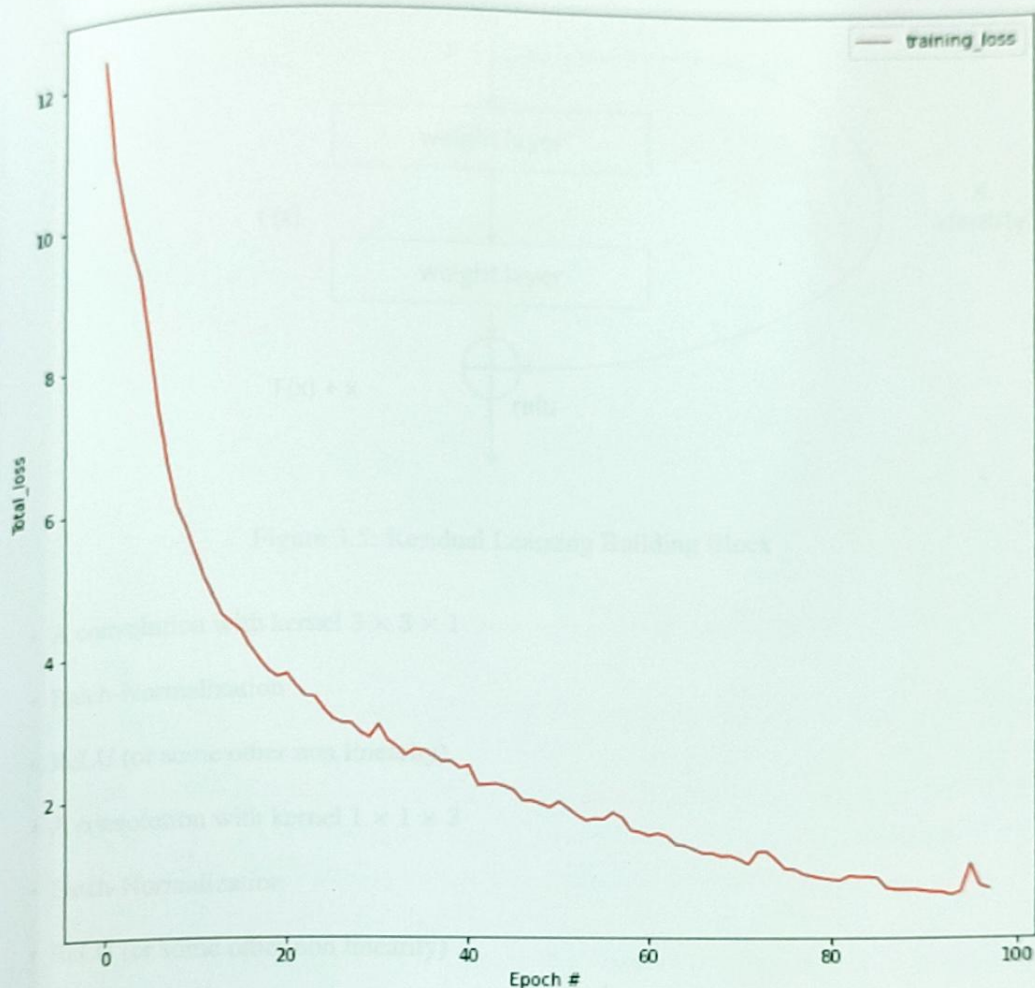


Figure 3.4: Poor optimization of C3D-AVG-MTL

form sports-1m [12] pretrained and fine-tuned C3D [10] architecture quite convincingly on the action recognition task.

Some works in the action-recognition field have demonstrated that factorizing the spatial and temporal components of the 3D convolution kernels used in such ResNets often leads to better results. For example Xie et al. [60] propose something similar for 3d-inception convolution networks [inception] and Tran et al. [15] propose something similar for 3D ResNets. Tran et al. [15] propose to replace each $3 \times 3 \times 3$ convolution block in the ResNet with a 3×3 convolution in the 2 spacial dimensions, followed by a 1D convolution along the temporal dimension using a kernel of size 3. From an implementation point of view, this means replacing a $3 \times 3 \times 3$ convulsion kernel followed by batch-normalization and a ReLU (or some other non-linearity) with a component containing:

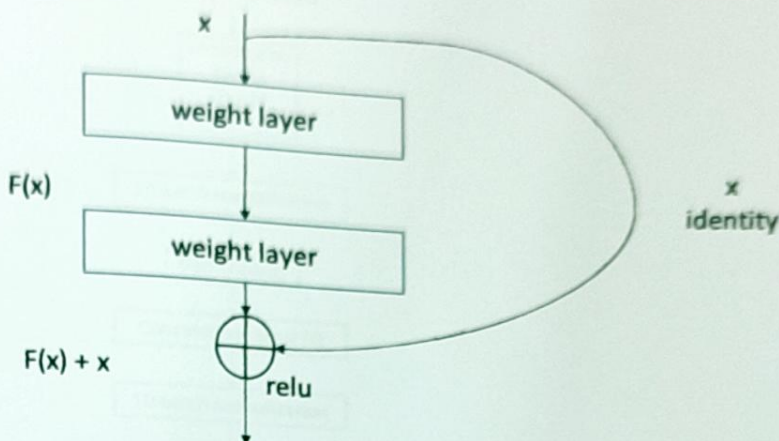


Figure 3.5: Residual Learning Building Block

- A convolution with kernel $3 \times 3 \times 1$
- Batch-Normalization
- ReLU (or some other non linearity)
- A convolution with kernel $1 \times 1 \times 3$
- Batch-Normalization
- ReLU (or some other non linearity)

The authors term this version of the ResNet a (2+1)D Resnet.

A useful aspect of using spatio-temporally factorized kernels is the possibility to introduce more non-linearity. It has been shown that non-linearities help [61] deep neural nets converge faster, and to express and/or approximate more complex patterns. Because we are replacing one 3D kernel with 2 kernels, one with 2 dimensions and another with 1 dimension, it is possible to introduce a non-linearity (for example ReLU) in between the 2 filters. It was showed by Tran et al. [15] this helps (2+1)D ResNets to optimize faster and produce lower error than a 3D ResNets with same depth.

Based on the prior discussions, we believe utilizing a deep ResNet as a feature extractor for action quality assessment is a promising idea. Our initial observation was that the C3D extracted features were less than optimal when it came to using them for AQA. We concluded that we need a deeper CNN that could extract rich features from the video based on which the linear-regressor might predict better scores. We think that a ResNet should be ideal for a feature extractor, as most datasets to be used for supervised training on the task of AQA are quite small

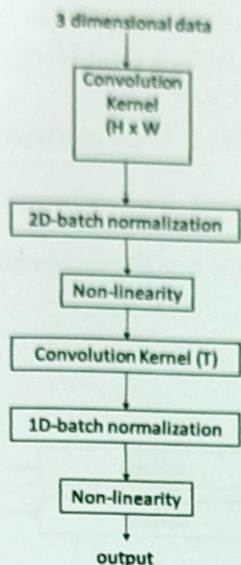


Figure 3.6: Spatio-temporally factored convolution kernel

(contains training samples in the order of a few thousand), thus a network any deeper might end up overfitting. This is supported by the finding discussed by Hara et al. [14], where the authors show through experiments that datasets like sports-1M [12] and kinetics-700 [58] can comfortably train 3D ResNets with layers up to 200 without overfitting. However, small scale on datasets like UCF-101 [57], ResNets having more than 34 layers seemed to overfit. We think a solution to this problem might be pretraining the ResNet on a big scale related dataset. Action-recognition datasets contain video data of humans performing actions. Hence, large-scale action-recognition datasets can be used for pretraining the feature extractor ResNet. Then this feature extractor can be fine-tuned to perform well on AQA tasks. This is made easier as such open-source weights exist on the internet, provided by various authors of related works.

To experiment with the relation of the ResNet feature extractor's depth with the AQA pipeline's ability to learn, we experiment with 3 different depths:

3.2.1 34-layer ResNet

We experiment with both 34 layer 3D ResNets and (2+1)D ResNets. The only difference being 3D ResNet uses $3 \times 3 \times 3$ convolution kernels, on the other hand (2+1)D ResNet uses a $1 \times 3 \times 3$ convolution followed by $3 \times 1 \times 1$ convolution. We take the final average-pool layer outputs, which is a feature vector of size 512, and pass it through 2 back-to-back fully-connected layers having 256 and 128 units. The final 128 dimensional feature vector is defined as the output of the feature extractor. The 3D ResNet takes input 16 frame clips, making the input size

$16 \times 3 \times 112 \times 112$. Due to the availability of pre-trained weights, we additionally experiment with 3 different variations of (2+1)D 34-layer ResNet, each processing different-sized clips.

- 8 frame clips, input clip dimension: $8 \times 3 \times 112 \times 112$.
- 16 frame clips, input clip dimension: $16 \times 3 \times 112 \times 112$.
- 32 frame clips, input clip dimension: $32 \times 3 \times 112 \times 112$.

3.2.2 50-layer ResNet

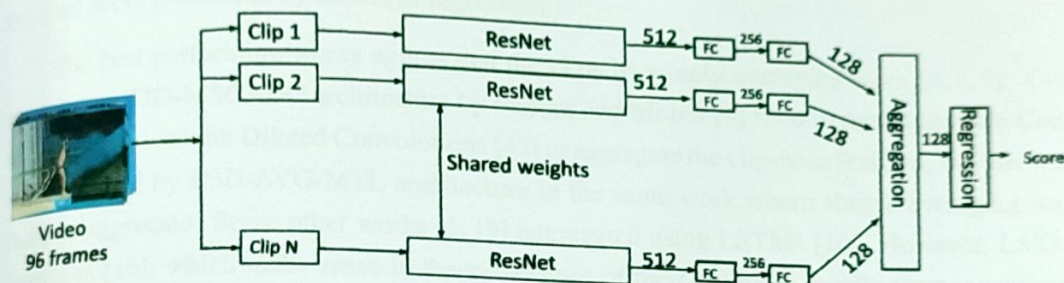


Figure 3.7: Using ResNets as feature extractor.

We experiment with both 50 layer 3D Resnets and (2+1)D ResNets. In this case, the final average-pool layer outputs a feature vector of size 2048. We take this feature vector and input it into 3 back-to-back fully-connected layers having 512, 256, and 128 units. The final 128 dimensional feature vector is defined as the output of the feature extractor. Both the 3D and the (2+1)D ResNets take input 16 frame clips, making the input dimensions $16 \times 3 \times 112 \times 112$.

3.2.3 101-layer ResNet

We experiment with 101 layer 3D ResNet. In this case, the final average-pool layer outputs a feature vector of size 2048. We take this feature vector and input it into 3 back-to-back fully-connected layers having 512, 256, and 128 units. The final 128 dimensional feature vector is defined as the output of the feature extractor. the 3D ResNet take input 16 frame clips, making the input dimensions $16 \times 3 \times 112 \times 112$.

All of our proposed frameworks work with videos of 96 frames. As shown in figure 3.7, it divides the video into non-overlapping and contiguous clips. In the case of 16 frame feature extractors, the original video is divided into 6 clips, for 32 frame feature extractors, the video is divided into 3 clips, and for 8 frame feature extractors, the video is divided into 12 clips. Each clip is processed sequentially using the same feature extractor, later aggregating

the 128-dimensional feature vectors to obtain one video-level feature descriptor to train the linear-regressor on.

3.3 Feature Aggregation with *Weight-Decider*

Most of the previous works dealing with AQA process the entire input video by first dividing it into multiple smaller clips of equal size, due to memory and computational budget. Most CNNs are designed to process 8, 16, or 32 frames at once. Then the features extracted by the CNN are aggregated to form a video-level feature vector. Finally this feature vector is used to calculate the final score prediction by means of regression.

The best performing works aggregated the clips by simply averaging them [5, 8, 9]. One work titled C3D-MSADC architecture by Parmer and Morris [8] tried to use Multiscale Context Aggregation with Dilated Convolutions [45] to aggregate the clip-wise features, but this was outperformed by C3D-AVG-MTL architecture in the same work where simple averaging was done to aggregate. Some other works [5, 19] aggregated using LSTMs [16]. However, LSTM networks [16], which make sense in theory because of their ability to handle time sequences, perform worse due to the lack of big-enough datasets dedicated to AQA.

Previous works like Parmer and Morris [8] argue this is effective. They argue an athlete gaining or losing points throughout action is analogous to an addition operation. Hence, they propose that if good features are learned, linear operations on those features become meaningful. Thus using a linear combination of these features for calculating the final score should make sense. However, the authors do not describe any reasons as to why they prefer averaging over other any other form of linear combination.

We believe simply taking an average to aggregate the feature vectors temporally might be detrimental to the assessment of the quality of actions and a potential area for improvement.

We agree with Parmer and Morris [8] that a linear combination is meaningful when it is modeling a task where a human achieves a score through his/her actions. However, we propose that a more sophisticated linear combination might make more sense than a simple average. We propose that simply averaging the clip-wise features is an ineffective measure. It should not be able to preserve the temporal information available in the data. This follows from the fact we could change the order of the clip level features and we will still get the same average and hence the same score prediction. Furthermore, if we look at real-world judging schemes, we will see those expert judges focus more on mistakes and deviations and these have a bigger impact on the score. Hence we think, a weighted averaging technique might be more suitable, as the linear-regressor in the final layer will be able to base its decision on features more important from each clip.

By taking an weighted average, we believe, the final linear regressor can make its decision in a more targeted way. For instance, when a real world judge scores a participant, he/she does so based on the some key moments in the action performance. The score depends more on the moments in which the performer makes some small mistake or some exceptional movement. The feature vector from individual clips can learn to encode this. However, as soon as the feature vectors are averaged together, this encoding would be smoothed out. Hence, it would be impossible for the linear regressor to make a decision based on only the "important" parts. Instead, its decision would be based on an "overview" of the entire video. On the other hand, our proposed weighted averaging scheme would be able to weight the individual features in the feature vectors so that the final feature vector is representative of the most important elements of the video. This way the feature vector can give more focus on the small errors or the impressive protons that impact the final score the most.

However, proposing to take a weighted average leads to the following question: "how would the weights be determined?". It is quite obvious that the weights are dependant on the video content. Hence there is no way to calculate this weighted average using pre-determined weights. There has to be some way to determine the weights during runtime based on the video. We think, proposing the weights can be done by taking a look at the clip level features assigning an importance to them. Then a normalized version of this feature importance score can be used weights for weighted averaging. we propose to introduce a neural network that can learn to propose importance of each feature. It would take a look at the feature vector proposed by the feature extractor and then it would propose the importance scores. However introducing another trainable neural network in the architecture would undoubtedly introduce more complexity in the system. This would translate to longer training times and richer resources. Thus, this neural network has to be as shallow as possible to introduce as little trainable parameter to the entire system as possible.

We call this shallow neural network *Weight-Decider* or WD for short. The architecture includes 4 fully connected layers containing 64, 32, 64 and 128 neurons. The neurons use ReLU (Rectified Linear Units) for activation. The network can be seen as a very shallow encoder-decoder network. The input and teh output sizes are the same. This is done so that the proposed weight vector can be used to scale the original feature vector. The architecture can be seen in Figure 3.8.

Now, the output of the WD module can be elementwise multiplied with the feature vector. This way the features are scaled before being summed together. The final video level feature vector's each element is a weighted sum of the elements in the same position of the clip level feature vectors. Howeve, we wish to make this weighted sum an weighted average. To do so, we need to somehow normalize the weights in the weight vectors, so that elements in the same indices some upto exactly one. One simple way to do this is to apply the softmax function on the elements. The softmax function is defined using equation 3.4. Using this function on all the



Figure 3.8: The architecture of the WD network,

elements in the same index of the feature vectors, we can normalize them such that they sum upto exactly 1. Then the weighted sum is effectively a weighted average.

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (3.4)$$

Finally, this way, it is possible for the linear regressor to make a more informed and focused decision about the final score prediction.

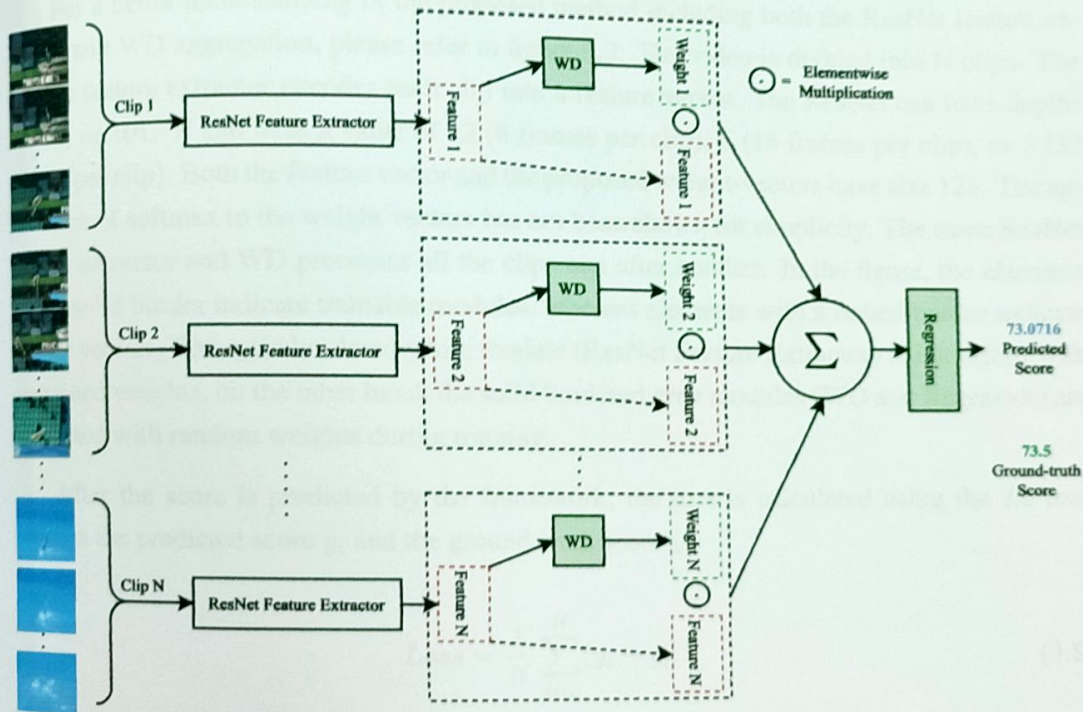


Figure 3.9: Proposed Method Architecture with ResNet feature extractor and WD aggregator.

More concretely, if the feature vector extracted from clip C_i is f_i , we propose the video

level feature vector as

$$f_{video} = \sum_{i=1}^N (f_i \odot w_i) \quad (3.5)$$

where w_i is a weight vector corresponding to the feature vector f_i and \odot represents Hadamard Product or elementwise multiplication. The weights corresponding to w_i are of the same dimensions as f_i and learned using a small neural network of 4 layers (WD). This smaller neural network takes as input 128-dimensional feature vector f_i and runs it through fully connected layers containing 64, 32, 64, and 128 layers. All but the final layers employ a ReLU activation function. The architecture is explained in figure 3.8. Finally, to ensure the weights corresponding to the same element of different weight vectors sum up to one, a softmax is applied along with the corresponding elements of all the weight vectors. We call this shallow neural network Weight-Decider (WD).

$$w'_i = WD(f_i) \quad (3.6)$$

$$[w_0 \ w_1 \ \dots \ w_N] = softmax([w'_0 \ w'_1 \ \dots \ w'_N]) \quad (3.7)$$

Finally, the linear-regressor can predict the score using the feature vector f_{video} as proposed by equation (3.5).

For a better understanding of the proposed method including both the ResNet feature extractor and WD aggregation, please refer to figure 3.9. The video is divided into N clips. The ResNet feature extractor encodes each clip into a feature vector. The ResNet can have depths 34, 50, or 101. N can have a value of 12 (8 frames per clip), 6 (16 frames per clip), or 3 (32 frames per clip). Both the feature vector and the proposed weight-vectors have size 128. The application of softmax to the weight vectors has not been shown for simplicity. The same ResNet feature extractor and WD processes all the clips one after another. In the figure, the elements with a solid border indicate trainable modules, whereas elements with a dotted border indicate feature vectors. The solid bordered white module (ResNet Feature Extractor) is initialized with pretrained weights, on the other hand, the solid bordered grey modules (WD and Regressor) are initialized with random weights during training.

After the score is predicted by the framework, the loss is calculated using the $L2$ loss between the predicted score y_i and the ground-truth score y'_i .

$$Loss = \frac{1}{N} \sum_{i=1}^N (y_i - y'_i)^2 \quad (3.8)$$

Here N is the number of training examples in the minibatch being used to train the model. Using this loss function, supervised learning can be done by optimizing the proposed architecture weight/parameters using an optimization algorithm like Stochastic Gradient Descent or ADAM optimizer [62].

We provide an example of how our entire pipeline works and produces the final prediction score in figures 3.10, 3.11, 3.12 for better understanding the entire process. The video is divided into multiple clips. In this example we assume that the video is divided into 32-frame clips. Therefore there are 3 clips in total. This can be seen in Figure 3.10.

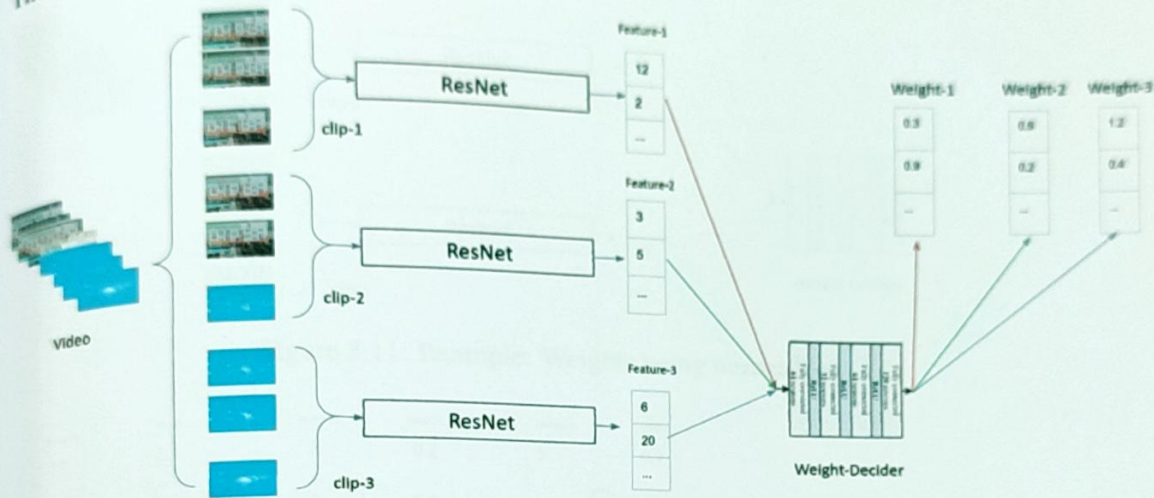


Figure 3.10: Example: The video being processed by the pipeline. Weights proposed by WD

The video is then processed by the ResNet feature extractor. In this example we do not show any specific ResNet. It can be ResNet-34, ResNet-50 or ResNet-101. One can even choose to use a deeper ResNet. The ResNet can use either (2+1)D convolution or 3D Convolution. Finally the ResNet outputs a 128 dimensional feature vector for each of the clips. In Figure 3.10, we these feature vectors are Feature-1, Feature-2 and Feature-3. The *Weight-Decoder* module described in Figure 3.8 takes input each of these feature vectors Feature-1, Feature-2 and Feature-3 and outputs corresponding weight vectors Weight-1, Weight-2 and Weight-3. These weight vectors all have the same dimensions (128) as the feature vectors. Notice that the elements in the same index of the weight vectors do not add up to one. For example the elements at the index 0 of Weight-1, Weight-2 and Weight-3 sum up to $0.3 + 0.6 + 1.2 = 2.1$. Ideally this should be 1. To fix this, these weights are normalized using the softmax function described in 3.4.

The direction of the application of softmax can be seen in 3.11. After the application of softmax, the weights at index 0 of Weight-1, Weight-2 and Weight-3 sum up to $0.1 + 0.2 + 0.7 = 1$. The same can be observed for indices 1, 2, 3, ..., 127. Now these can be used for weighted averaging. Next, weight vectors are elementwise multiplied with the feature vectors. This can be seen in Figure 3.12. After this multiplication, we have 3 weighted-feature vectors. Adding them together gives us a video level feature vector.

Finally, this video level feature vector is used as input to the linear regressor that calculates the final score prediction. The linear regressor does so by multiplying this input feature vector

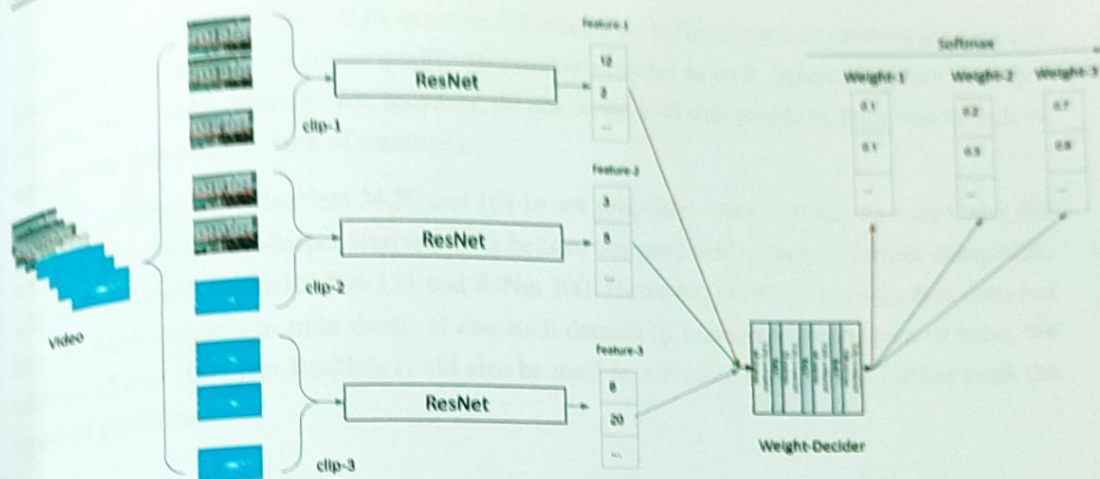


Figure 3.11: Example: Weights being normalized.

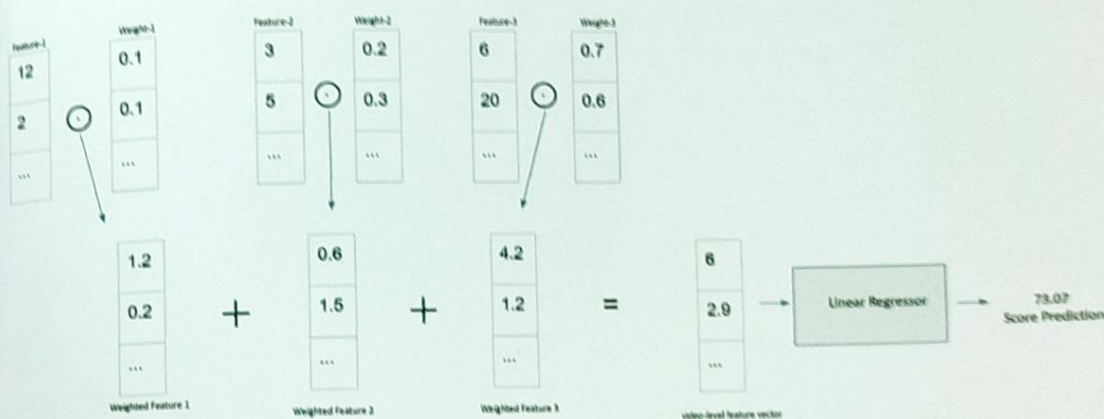


Figure 3.12: Example: Final Score Prediction

with a parameter vector that is learned through training during back-propagation. Finally a bias vector is added to this result to produce the final score. The bias vector is also learned through training during back propagation. Finally, we choose to multiply this score with the corresponding difficulty degree. This is done to decouple the difficulty degree prediction from the score prediction. This increases the accuracy of the system. Notice that the judges in an official event are provided with these score as well. Hence we believe this to be a fair comparison. However, in cases where this difficulty degree was missing from the dataset, we used a value of 1 which is equivalent to making the network predict the difficulty degree as well. However, this puts the system under extra stress.

In this way, using a deeper ResNet and Weigh-Decoder aggregation, better performance can be achieved in Action Quality Assessment. We targeted, trained and tested out system for

sport related short clips based AQA systems. However, we believe that this performance should generalize to other types of action quality assessment systems as well. Translating this to longer snippets prove to be difficult. We, however, do not scope into that problem, both due to lack of appropriate datasets and lack of resources.

We included only ResNets 34,50 and 101 in our proposed system. It can be asked why did we not experiment with deeper ResNets? We believe our proposed system is further compatible with deeper ResNets like ResNet 151 and ResNet 200. However, currently enough data does not exist in AQA datasets to train them. If one such dataset of enough volume were to exist, we believe, 151 or 200 layer ResNets could also be used as a feature extractor to further push the limits of performance.

Benchmark Dataset: MTL-AQA

The MTL-AQA is the largest dataset in the AQA domain published to date. The dataset contains 1000 video samples. All the samples are of Olympic size collected from 1000+ videos. The dataset is made up of 100 classes. They have varying popularity, action types. The dataset contains samples of both male and female athletes, individual and synchronous. It contains samples from both 2D and 3D platform diving. The metadata corresponding to each sample are detailed. It contains the final action quality score from the 7-judge panel, the crowd, and the organized scores (which consist of professional, media, student, and crowd). The crowd scores are used by the corresponding video streams to train the 100% sample training set and 20% sample testing set. The dataset MTL-AQA is available on the website www.mtl-aqa.com. We describe the details of the dataset and its usage in the next section.

3.4. Distance Metric

The distance metric is defined as a measure of the difference between two samples. In this work, we use the L2 distance metric to measure the difference between two samples. The L2 distance is defined as the square root of the sum of the squares of the differences between the corresponding elements of the two samples.

Chapter 4

Result Analysis

In this chapter, we discuss the dataset we have trained and tested our proposed methods on, the evaluation metric that we have used, the experimental setup used, and the result of these experiments. Finally, a comparison with the state-of-the-art is also presented.

4.1 Benchmark Dataset: MTL-AQA

MTL-AQA [8] is the biggest dataset in the AQA domain published to this date. The dataset contains 1412 video samples. All the samples are of Olympic dive collected from 16 events. The videos are made up of 103 frames. They have varying perspectives, camera angles. The dataset contains samples of both male and female athletes, individual and synchronous diving. It contains samples from both 3m and 10m platform diving. The annotations accompanying this dataset are detailed. It contains the final action quality score from the Olympic judges, the scores made by the 7 individual judges, task difficulty level, commentary from the broadcast of the event, and fine-grained action labels (number of somersaults, twists, starting position, rotation type, and armstand). The most popular split used by the contemporary works divides the dataset into a 1059 sample training set and 353 sample testing set. The current SOTA results reported by various works evaluate their work on the MTL-AQA dataset. We, therefore, shift our primary focus on training out methods on this dataset and testing it.

4.2 Performance Metric

Action quality assessment is modeled as a regression problem. In contemporary works in literature, the performance of a system to properly measure the quality of scores is done using Spearman's rank correlation. Spearman's rank correlation can measure the rank correlation be-

tween ordinal, interval, or ratio data. This is a non-parametric method that can measure the similarity between 2 random variables. It does not assume any underlying distribution and only operates under the assumption that the data being presented to it at least has an ordinal relation.

Following equation is used to calculate the Spearman's rank correlation

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (4.1)$$

ρ = Spearman's rank correlation

d_i = the difference ranks of corresponding variables

n = number of observations

Using Spearman's rank correlation to compute the rank correlation between the ground-truth scores and predicted scores by the proposed system tells us how much the rank of the predicted scores matches the rank of the ground truth scores. That is, instead of measuring if the scores match up perfectly, we are trying to conclude if task A has a better ground-truth score than task B, does that same relation hold in the case of the predicted scores. A perfect Spearman's rank correlation of +1 would suggest that the prediction system can mimic the rank correlation present in the ground-truth scores perfectly.

4.3 Experimental Setup

We implemented our proposed methods using PyTorch [63]. We experimented with various types of 3D ResNets [13] and (2+1)D ResNets [15] as feature extractors. All the 3D ResNets and (2+1)D ResNets processing 16 frame clips and were pre-trained on kinetic-700 [58] action recognition dataset¹. The (2+1)D Resnets processing 8 frame clips and 32 frame clips were pre-trained on IG-65M dataset [59] and fine tuned on Kinetic-400 [58] action recognition dataset². For each ResNet feature extractor, we separately experimented using both averaging as feature aggregation and *WD* as feature aggregation.

We did temporal augmentation by randomly picking an ending frame from the last 6 and chose the preceding 96 frames for processing. The video frames are resized to 171×128 and a center crop of 112×112 was taken. We performed spatial augmentation by random horizontal flipping. The 96 frames were divided into 6 clips in case of 16 frame clips, 12 clips in case of 8 frame clips, and 3 clips in case of 32 frame clips. All the clips were non-overlapping and contained frames in the original order present in the video. Batch-normalization [64] was used in the convolutional layers for regularization.

¹Weights available at <https://github.com/kenshohara/3D-ResNets-PyTorch>

²Weights available at <https://github.com/moabitcoin/ig65m-pytorch>

We defined the loss function as L2 loss between the predicted score and ground-truth score and then we optimized this loss. For each experiment, the entire network was trained end-to-end using the ADAM optimizer [62] for 50 epochs. We found through trial and error that a learning rate of 0.0001 for modules with randomly initialized weights and 0.00001 for modules with pretrained weights to produce the best results. In figure 3.9, the modules colored in green are trained using a learning rate of 0.0001 and colorless modules are trained using a learning rate of 0.00001. We used training batches of size 2 and test batches of size 5.

4.4 Effect of ResNet Depth on the Performance

Because the MTL-AQA dataset contains difficulty-degree of the action, and real-world judges multiply their score with difficulty-degree to produce a final score, we choose to multiply the output score of the linear-regressor with difficulty-degree.

In Table 4.1, we present the experiment results of varying the depth of the ResNet feature extractor as well as varying the aggregation scheme. All the ResNets, in this case, are initialized with Kinetics-700 [58] Action Recognition dataset pretrained weights. We can see that 34 layer ResNet with *WD* as aggregation performs the best with a Spearman's correlation of 0.8990. This leads us to conclude that when initialized with pretrained weights on a related task like action recognition, the MTL-AQA dataset has enough data to train at least 34 layer deep ResNets without overfitting. Interestingly, increasing the depth to 50 layers somewhat decreases Spearman's correlation. However, the results are still competitive.

Depth	Convolution Type	Aggregation	
		Average	WD
34	3D	0.8982	0.8951
	(2+1)D	0.8932	0.8990
50	3D	0.8880	0.8935
	(2+1)D	0.8818	0.8814
101	3D	0.6663	0.6033

Table 4.1: Performance comparison of the various types of ResNets as feature extractors in our pipeline

With 101 layer deep ResNets, even when initialized with pretrained weights from Kinetics Action Recognition dataset [58], overfitting occurs fairly quickly. The overfitting is also evident from the train/test curves presented in figures 4.1, 4.2, 4.3. The plot in figure 4.1 is the train and test losses obtained for various feature extractors using *WD* as aggregation.

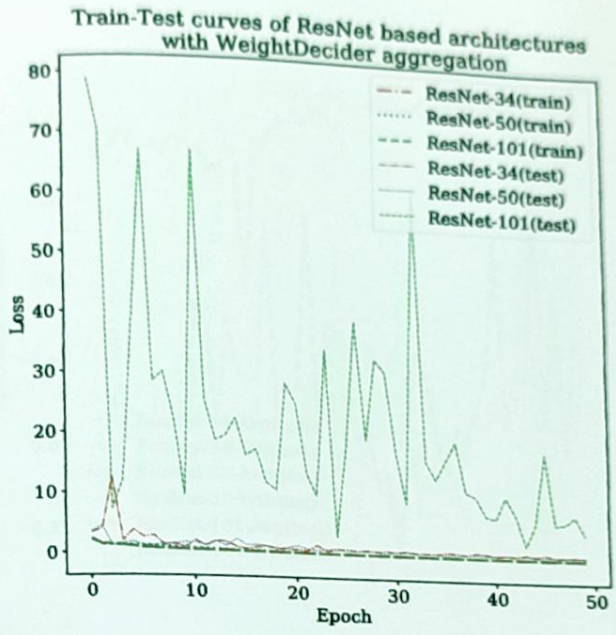


Figure 4.1: Train and Test loss curves obtained from training the pipeline using 3D-ResNet 34, 50, and 101 as feature extractors with WeightDecider aggregation.

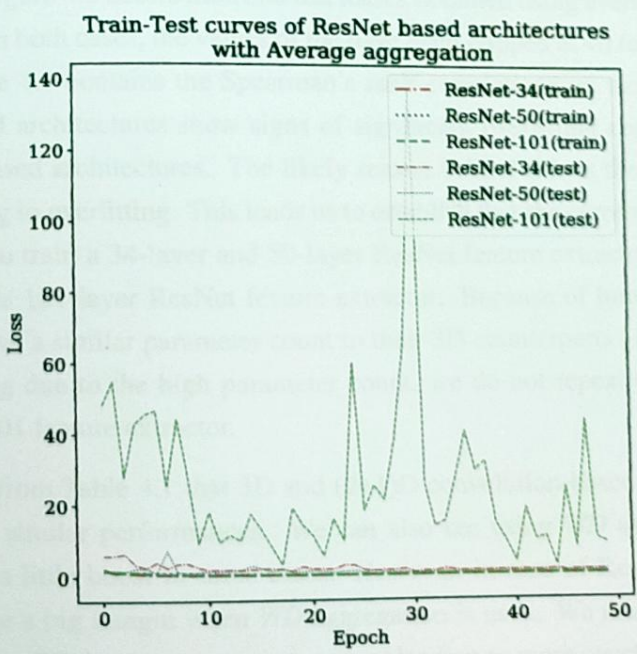


Figure 4.2: Train and Test loss curves obtained from training the pipeline using 3D-ResNet 34, 50, and 101 as feature extractors with Average aggregation.

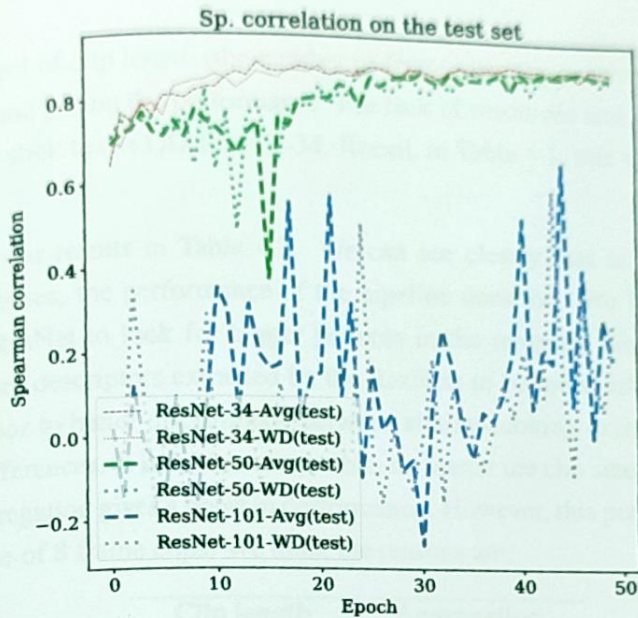


Figure 4.3: Train and Test Sp correlation curves obtained from training the pipeline using 3D-ResNet 34, 50, and 101 as feature extractors.

The plot in Figure 4.2 shows train and test losses obtained using averaging (AVG for short) as aggregations. In both cases, the values of the have been clipped at 40 for better visualization. The plot in Figure 4.3 contains the Spearman's rank correlations on the test set. Notice that ResNet-101 based architectures show signs of significant overfitting compared to ResNet-34 and ResNet-50 based architectures. The likely reason behind this is the increased number of parameters leading to overfitting. This leads us to establish that the current biggest AQA dataset has enough data to train a 34-layer and 50-layer ResNet feature extractor with generalization, however it overfits 101-layer ResNet feature extractor. Because of how (2+1)D ResNets are designed, they have a similar parameter count to their 3D counterparts [15]. Because the overfitting is occurring due to the high parameter count, we do not repeat the experiment with a (2+1)D ResNet-101 feature extractor.

We can see from Table 4.1 that 3D and (2+1)D convolution-based architectures with the same depth have similar performances. We can also see using *WD* as aggregation can give the performance a little boost in some cases. However, in case of ResNet-101, performance decreases by quite a big margin when *WD* aggregation is used. We think the reason might be that introducing the *WD* increases parameter count leading to more overfitting, which translates to worse performance.

4.5 Effect of Clip Length on Performance

We check the effect of clip length (the number of frames per clip processed by the ResNet feature extractor at one go) on the performance. For lack of resources and open-source pretrained weights, we only stick to (2+1)D-ResNet-34. Recall, in Table 4.1, this was the best performing model.

We present our results in Table 4.2. We can see clearly that as the number of frames in each clip increases, the performance of the pipeline does too. We hypothesize that longer clips allow the ResNet to look for bigger patterns in the temporal dimension, which in turn enables the feature descriptors extracted by the ResNets to be more informative. This enables the linear-regressor to better discriminate between similar-looking examples with fine-grained action quality differences. It is further observable, no matter the clip size, using *WD* over simple averaging as aggregation gives a boost in performance. However, this performance boost is quite significant in case of 8 frame clips. We think the reasons are:

Clip length (Input Frames)	Aggregation	
	Average	<i>WD</i>
8	0.8570	0.8853
16	0.8932	0.8990
32	0.9289	0.9315

Table 4.2: Performance comparison of ResNet(2+1)D-34 with varying input clip size

- Using 8 frame clips, the 96 frame video is divided into 12 clips, which means finally 12 clip level feature descriptors need to be aggregated. On the other hand, using 32 frame clips means finally 3 clip-level feature descriptors are being aggregated. Thus, whatever detrimental effect the averaging might have, it will be more prominent when the number of objects being averaged is larger, and less when this number is smaller. Hence using a 32 frame clip, the performance gained by using *WD* aggregation over averaging is only 0.0026 (0.28%), while in case of 8 frame clips, the performance gain is 0.0283 (3.30%).
- CNNs with bigger clips as input can look at more frames, this in effect increases their temporal horizon. It follows that the feature vectors extracted would have a better encoding of action patterns across time, to begin with. Thus they perform well enough even with averaging as aggregation. But using *WD* increases performance nevertheless.

4.6 Qualitative Results

The qualitative results for 4 random samples from the test set are presented in Table 4.3. Due to space constraints, we show every 16th frame processed starting from frame 0. The blue scores correspond to score prediction produced using *WD* as aggregation, while the black scores correspond to score prediction produced using average as aggregation. The 8, 16, and 32 correspond to input clip sizes.




Input Frames	ResNet-34 Prediction				ResNet-50 Prediction		ResNet-101 Prediction	True Score
	(2+1)D			3D	(2+1)D	3D	3D	
	8	16	32					
	54.84 38.76	30.46 18.11	8.39 16.41	7.29 22.93	33.23 38.29	34.10 29.93	45.22 52.21	25.64
	66.94 63.85	59.69 40.88	47.92 53.21	67.92 63.62	43.57 52.80	58.30 52.31	122.20 76.64	
	71.46 69.85	71.34 64.90	69.39 70.40	83.34 81.31	67.38 67.53	80.41 74.25	167.60 132.50	69.59
	67.13 64.54	46.16 32.29	27.73 42.87	34.25 39.62	44.06 49.03	46.61 47.13	54.28 51.51	46.20

Table 4.3: Qualitative results

4.7 Comparison With the State-of-the-Art on the MTL-AQA Dataset

In Table 4.4, we compare our best performing models of each depth with previous state-of-the-art works on the MTL-AQA dataset. We can see that our ResNet34(2+1)D processing 32 frame clips with *WD* as aggregation scheme outperforms all previous works in the literature. This shows the effectiveness of our approach.

If we further look at the results in Table 4.1 and compare with Table 4.4, we see that 34 layers ResNets based pipelines achieve performance comparable to the SOTA when trained on the AQA-MTL dataset. The 50 layer ResNets perform somewhat worse and the 101 layer ResNets overfit and perform poorly.

Again, comparing Table 4.2 with Table 4.4, we can see that 34 layers (2+1)D ResNets processing 8 frame clips perform worse than some C3D based models, even after having the advantage of depth over C3D networks. 16 frame models perform comparable to the best C3D

Method	Sp. Corr.
Pose+DCT [23]	0.2682
C3D-SVR [5]	0.7716
C3D-LSTM [5]	0.8489
MSCADC-STL [8]	0.8472
MSCADC-MTL [8]	0.8612
USDL-Regression [9]	0.8905
C3D-AVG-STL [8]	0.8960
C3D-AVG-MTL [8]	0.9044
MUSDL [9]	0.9273
Proposed ResNet34-(2+1)D-WD (32 frame)	0.9315
Proposed ResNet50-3D-WD (16 frame)	0.8935
Proposed ResNet101-3D-AVG (16 frame)	0.6633

Table 4.4: Comparison with the State-of-the-Art

based approaches, however, are beaten by I3D based approaches. The 32 frame clip processing models clearly outperform all previous works. Hence it can be argued, based on the data already available in MTL-AQA dataset, processing more frames per clip and focusing on improving the aggregation techniques can lead to better performance than going deeper with convolutions.

Chapter 5

Conclusion

5.1 Summary

Action Quality Assessment is a new yet challenging discipline. It has potential application in automatic sports judgment, automated training systems with accurate feedback, video retrieval, etc. However, this is a challenging task due to the huge amount of computation required to accurately measure action quality from video data. The lack of big-scale datasets dedicated to this discipline introduces further complications. In this work, we proposed a ResNet-based regression-oriented pipeline for action quality assessment. We demonstrated experimentally that the MTL-AQA dataset has enough data to train 34 and 50 layer ResNet-based pipelines when initialized with pretrained weights from a related task (like action recognition). Our experiments suggest processing longer clips is more effective than using deeper ResNets. We also propose a sophisticated learning-based aggregation technique called *WD* to replace simple averaging. Experiments show our methods to be more effective than previous works.

5.2 Limitations and Future Works

Self-attention [65] has revolutionized the field of deep learning. NLP has seen success in replacing CNN and LSTMs with transformers that are based on self-attention. Recently the field of computer vision has also taken inspiration. Zhou et al. [66] has shown that the application of Transformers can boost performance significantly on the video to caption generation. We think the application of a similar model utilizing Self-Attention in videos might be able to outperform existing AQA and action recognition pipelines. We also want to evaluate our current architecture on other datasets like MIT-dive [23] and AQA 7 [19] to check whether our proposed method can achieve good results on those datasets as well.

One of the limitations of work is that it only deals with small action sequences. In the real world, action sequences are generally bigger. It might be an interesting avenue to study. Another limitation out work has is its inability to provide feedback and corrections to the action performers. This can be done if skeleton and pose data was also processed side by side. We think this a promising area of study.

References

- [1] Q. Lei, T.-X. Du, H.-B. Zhang, S. Ye, and D.-S. Chen, "A survey of vision-based human action evaluation methods," *Sensors*, vol. 18, p. 4179, 09 2018.
- [2] V. Singh, A. Sabat, and M. J. Black, "HumanFlow: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision*, vol. 87, pp. 4–27, 2010.
- [3] A. Todorov and C. Sngupta, "Detectpose: Human pose estimation via deep action networks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1823–1831, 2018.
- [4] Z. Cao, D. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [5] P. Palmer and B. Tran Morris, "Learning to score olympic events," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: Workshop*, pp. 26–34, 2012.
- [6] X. Zhang, Y. Tian, A. Reiter, G. D. Hager, and T. D. Seng, "K3d: Sketching egocentric gait for action quality assessment," in *2018 IEEE International Conference on Image Processing (ICIP)*, pp. 925–932, 2018.
- [7] Y. Li, X. Chen, and X. Chen, *Real-world learning for action quality assessment*, pp. 121–130, 2012.
- [8] P. Palmer and B. Tran Morris, "What and how well you performed? a simple task learning approach to action quality assessment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 236–243, 2019.
- [9] J. Tang, Z. Wu, Z. Yang, D. Deng, J. Lu, Y. Wu, and L. Zeng, "Unsupervised video motion classification network for action quality assessment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

References

- [1] Q. Lei, J.-X. Du, H.-B. Zhang, S. Ye, and D.-S. Chen, "A survey of vision-based human action evaluation methods," *Sensors*, vol. 19, p. 4129, 09 2019.
- [2] L. Sigal, A. Balan, and M. J. Black, "HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision*, vol. 87, pp. 4–27, Mar. 2010.
- [3] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660, 2014.
- [4] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [5] P. Parmar and B. Tran Morris, "Learning to score olympic events," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–28, 2017.
- [6] X. Xiang, Y. Tian, A. Reiter, G. D. Hager, and T. D. Tran, "S3d: Stacking segmental p3d for action quality assessment," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 928–932, IEEE, 2018.
- [7] Y. Li, X. Chai, and X. Chen, *End-to-end learning for action quality assessment*, pp. 125–134. 2018.
- [8] P. Parmar and B. T. Morris, "What and how well you performed? a multitask learning approach to action quality assessment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 304–313, 2019.
- [9] Y. Tang, Z. Ni, J. Zhou, D. Zhang, J. Lu, Y. Wu, and J. Zhou, "Uncertainty-aware score distribution learning for action quality assessment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

- [10] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, (USA), p. 4489–4497, IEEE Computer Society, 2015.
- [11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.
- [13] K. Hara, H. Kataoka, and Y. Satoh, "Learning spatio-temporal features with 3d residual networks for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 3154–3160, 2017.
- [14] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [15] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, pp. 1735–1780, MIT Press, 1997.
- [17] P. Parmar and B. T. Morris, "Measuring the quality of exercises," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2241–2244, 2016.
- [18] G. Bertasius, H. Soo Park, S. X. Yu, and J. Shi, "Am i a baller? basketball performance assessment from first-person videos," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2177–2185, 2017.
- [19] P. Parmar and B. Morris, "Action quality assessment across multiple actions," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1468–1476, IEEE, 2019.
- [20] I. Funke, S. Mees, J. Weitz, and S. Speidel, "Video-based surgical skill assessment using 3d convolutional neural networks," *International Journal of Computer Assisted Radiology and Surgery*, vol. 14, 05 2019.

- [21] M. Hu, C. Chen, W. Cheng, C. Chang, J. Lai, and J. Wu, "Real-time human movement retrieval and assessment with kinect sensor," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 742–753, 2015.
- [22] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.
- [23] H. Pirsiavash, C. Vondrick, and A. Torralba, "Assessing the quality of actions," in *European Conference on Computer Vision*, pp. 556–571, Springer, 2014.
- [24] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 65–72, 2005.
- [25] Laptev and Lindeberg, "Space-time interest points," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 432–439 vol.1, 2003.
- [26] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.
- [27] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM International Conference on Multimedia, MM '07*, (New York, NY, USA), p. 357–360, Association for Computing Machinery, 2007.
- [28] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *2013 IEEE International Conference on Computer Vision*, pp. 3551–3558, 2013.
- [29] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, pp. 1–2, Prague, 2004.
- [30] V. Kyrki, I. S. Vicente, D. Kragic, and J.-O. Eklundh, "Action recognition and understanding using motor primitives," in *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 1113–1118, IEEE, 2007.
- [31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

- [33] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proceedings of the 9th International Conference on Neural Information Processing Systems, NIPS'96*, (Cambridge, MA, USA), p. 155–161, MIT Press, 1996.
- [34] A. S. Gordon, "Automated video assessment of human performance," pp. 16–19, 1995.
- [35] W. Ilg, J. Mezger, and M. Giese, "Estimation of skill levels in sports based on hierarchical spatio-temporal correspondences," in *Joint Pattern Recognition Symposium*, pp. 523–531, Springer, 2003.
- [36] F. Patrona, A. Chatzitofis, D. Zarpalas, and P. Daras, "Motion analysis: Action detection, recognition and evaluation based on motion capture data," *Pattern Recognition*, vol. 76, 12 2017.
- [37] V. Venkataraman, I. Vlachos, and P. Turaga, "Dynamical regularity for action analysis," pp. 67.1–67.12, 01 2015.
- [38] K. Weeratunga, A. Dharmaratne, and K. Boon How, "Application of computer vision and vector space model for tactical movement classification in badminton," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 76–82, 2017.
- [39] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson, "Automatic evaluation of sports motion: A generic computation of spatial and temporal errors," *Image and Vision Computing*, vol. 64, pp. 67–78, 2017.
- [40] A. Paiement, L. Tao, M. Camplani, S. Hannuna, D. Damen, and M. Mirmehdi, "Online quality assessment of human motion from skeleton data," in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2014.
- [41] M. Antunes, R. Baptista, G. Demisse, D. Aouada, and B. Ottersten, "Visual and human-interpretable feedback for assisting physical activity," in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 115–129, Springer International Publishing, 2016.
- [42] K. Wnuk and S. Soatto, "Analyzing diving: a dataset for judging action quality," in *Asian Conference on Computer Vision*, pp. 266–276, Springer, 2010.
- [43] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," pp. 5533–5541, 2017.
- [44] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 06 2014.

- [57] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012.
- [58] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," vol. abs/1705.06950, 2017.
- [59] D. Ghadiyaram, D. Tran, and D. Mahajan, "Large-scale weakly-supervised pre-training for video action recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12038–12047, 2019.
- [60] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 305–321, 2018.
- [61] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (J. Fürnkranz and T. Joachims, eds.), pp. 807–814, 2010.
- [62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.
- [63] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
- [64] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456, JMLR.org, 2015.
- [65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.
- [66] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 8739–8748, IEEE Computer Society, jun 2018.