Lightweight Action Quality Assessment for Competitive Diving

Zhaoqi Wang, Iris (Jiayue) Ma, Bowen Zhang
{z833wang, j252ma, b349zhan}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Abstract

With the large amounts of video data generated by major sporting events, and the idea of unbiased AI judging, large three-dimensional neural networks have been developed to achieve excellent accuracy in predicting the scores of divers at Olympic competitions. Our research introduces a lighter weight neural network to quickly complete the same process with similar accuracy using only one 12GB NVIDIA Tesla K80 GPU.

We were able to accomplish this using a lightweight 2D CNN with predictions within about 6 points of the true score that runs fast enough to be used in real-time. Training can also be completed reasonably quickly. With this research, we hope to motivate research into using limited computing resources to complete complex tasks for large amounts of data such as that of video data. This would undoubtedly promote accessibility for more people to complete complex machine learning tasks quickly and with little resource cost.

Introduction

After the first Video Assistant Referee (VAR) appeared to be helpful in the 2018 World Cup, the idea of using video to assist performance assessment in sports has been of interest [2]. Using the vast amounts of high-quality videos produced regularly from competition and training, powerful computer vision machine learning models are becoming increasingly accurate in performing complex regression analysis. However, in addition to video data being naturally intensive to store and process, the demand for increased accuracy is pushing models to become more costly to train and run, as they require more computational power. This paper focuses on crafting a lightweight model to assess a video of a sport judged on action quality to predict the performance score, as an expert human judge would. Ideally, this could create or inspire a future model that is able to perform near real-time analysis of the sport to provide immediate feedback and judging. The goal of this project is to retain the prediction accuracy of heavier models while improving the run-time and requiring few resources to train and run.

By providing action assessment technology rooted in machine learning, we can limit the judging bias and inefficiencies of human judges. Moving this technology into a real-time environment through the use of lightweight models allows it to actually replace the role of a judge. Further, a lightweight model opens accessibility to more users, allowing its usage in training environments. In future more advanced works, quick action assessment may be able to supervise training and competition to predict injury before it occurs and alert athletes. Alternatively, this could also serve as a base to create machine-generated commentary and feedback, for broadcasts and the athletes to review. Depending on the success of this technology, it can be brought outside of athletics, and into high-stakes environments where actions must be precise and focused, such as in surgery or for natural disaster rescues. The future of action assessment is vast and full of potential, compacting this all into a lightweight system would undoubtedly improve its accessibility and usability for more people.

In this paper, we will specifically focus on Olympic diving, though many papers have shown that models that work well for diving can often be generalized and predict figure skating [9] and vault gymnastics [5] as well. The advantage of these videos from major sporting events lies in that they are readily available, commentated on by professionals, and that these actions are performed by athletes that are among the best in their sport in the world.

In Olympic diving, scoring is completed by seven judges who all sit on one side of the diver. They provide a score immediately following the completion of the dive. This score is between 0 (fail) and 10 (near perfect), and is incremented by 0.5 - that is, a judge could judge a dive to be 6.5 or 7.0, but not 6.6 or 6.7. Then, the lowest two and highest two scores are eliminated and the remaining three middle scores are summed together to form the execution score. The execution score is then multiplied by the difficulty score of the dive to produce the final score of the dive. Unlike some other sports, judges are not provided video replays nor slow-mo footage and generate scores based on what is seen once by the naked eye. Even so, judges are in agreement 96% of the time, though non-expert humans poorly predict scores [9].

We are interested in implementing a lightweight transformer network based on a video action classification model [4] to predict performance scores. We want to know if this lightweight model is comparable in accuracy to the recent heavier models while operating at a reasonable training and running time. After initial implementation, we will review

the performance metrics of time and accuracy. Then, if we are unsatisfied with the performance, we will develop improvements of our own invention and take inspiration from existing heavier models.

The data-set we will use for training and testing consists of short video clips of Olympic divers in Olympic competitions in UNLV Olympic Scoring Dataset [8]. This consists of men's 10 metre platform diving at Olympics.

To evaluate the performance of this lightweight model, we introduce a baseline model for comparison. The baseline model is an Action Quality Assessment (AQA) model that uses a Discrete Cosine Transform (DCT) to preprocess data for Support Vector Regression [9]. This model was an early starting point in AQA and is mentioned in almost every subsequent paper as a baseline. While it does not use the heavy 3D convolutional neural networks (eg. C3Ds) of more recent models, its accuracy is much lower.

We will implement this baseline algorithm, train it using the same data-set and compare its execution time and accuracy to our proposed model. The execution time will be measured by the environment we are using and the accuracy will be calculated by the mean squared error.

We will be using the free Google Colab environment for all processes. While building complex models, strong computation power such as cloud computing is necessary for data processing, model training and validating. Since this is available to anybody with internet access, we hope that a successful implementation for us would indicate accessibility to most possible users.

Main Results

Our research suggests that with some minor adjustments and tuning, the lightweight transformer network is successful at achieving similar accuracy to C3Ds, falling within 3% accuracy. This lighter model completes its training at around 12 minutes per epoch and generated scores in seconds on new data using Google Colab's provided GPU, a 12GB NVIDIA Tesla K80.

There were significant challenges addressed in data loading which challenged Colab's RAM and with the accuracy of the model initially. These challenges were overcome through different loading methods and minor adjustments to the model along with tuning.

Contributions

- We implemented a lightweight transformer model on the video data typically evaluated by C3Ds and achieved similar accuracy
- Our implementation trains at a reasonable time and runs very fast on Google Colab's free services
- We made slight but carefully thought-out adjustments to our initial model plan and more than doubled the accuracy
- Our implementation was able to load large amounts of video data on Colab's RAM after calculated changes

Related Work

Pose-based and appearance-based features are the two main architectures proposed for models.

The first major break-through in AQA for diving and figure skating was published in 2014 [9]. This paper separated features into "low-level" and "high-level pose" features that captures edges/velocities and poses, respectively. This team then proposed the use of a Discrete Cosine Transform (DCT) of body pose as features for a support vector regressor (SVR) that estimated a final action quality score. Though the accuracy of this model was lacking when compared to professional judges, it performed better than non-expert humans. This model also provides feedback for athletes and highlights clips from the videos that most impacted the final score.

By 2015, this approach was further improved by replacing the DCT function with concatenated approximate entropy of poses [12]. However, this pose-based method has its own limitations as it ignores information in the background, which may be useful in assessing the quality of the action. Moreover, pose estimation may be challenging during fast movement or convoluted body pose, which is commonly seen in diving. Nevertheless, pose-based features have proved useful when combined with appearance features in some of the more recent models [5].

Due to recent advancement in deep learning and C3Ds, appearance-based methods, which utilize Red Green Blue (RGB) spatio-temporal features, became popular in video action recognition and AQA. Most recent approaches to the AQA problem use either C3D or I3D network. Parmar and Morris proposed using C3D for feature extraction, Long Short-Term Memory (LSTM) for feature aggregation and SVR for regression [8]. This approach achieves state-of-the-art performance on MIT-Dive and UNLV-dive dataset; many of the more recent works are a modification of this method. For example, Xu et al proposed self-attentive LSTM and multi-scale convolutional skip LSTM to learn local and sequential information in the video [13]. Other techniques like transferred learning [6] and modeling the score as an uncertainty-aware score distribution [11] are also used to boost the performance of the AQA frameworks. Parmar and Morris later proposed a multi-task learning approach consisting of action quality scoring, factorized (detailed) action classification and generating verbal commentary [7]; this achieves better performance as compared to methods that only regress a score.

AQA and action recognition in video have some similar components. Both of them require extraction of features in spatial and temporal domains and learning from a sequence of features. These tasks are often considered computationally heavy. A lightweight transformer-based action recognition model was proposed [4], and this model achieved reasonable accuracy and is capable of running in real time at 56 frames per second.

A more recent lighter and accurate model is seen in a 2020 paper about FALCONS [5], a model with a virtual refereeing system that considers both pose-based and appearance-based features. To train pose-based accessors, the team constructed ExPose, a data set of 3000 blurry diving and 1000 blurry gymnastic vault images of extreme body contortions with annotated joints. In the appearance-based execution accessor, the heavyweight C3D models are decomposed into

2D spatial convolutions followed by 1D temporal ones. Finally, a difficulty extractor module estimates the type of dive based on joint positions. This score is multiplied to the virtual referee's execution score for a final score. The proposed network has shown promise in use for unseen gymnastics data, when using the same parameters.

Methodology

This section addresses the general framework of our plan that was created prior to implementation. Naturally, minor changes were made during implementation to improve our model; note that these changes are not mentioned in this section and are instead reserved for the beginning of the following Results section.

Approach

Since efficiency is the main focus of our project, though 3D convolutional neural networks achieved remarkable performance in Olympic sport quality assessment, we adopt a transformer model with 2D convolutional networks in an attempt to achieve comparable results with other AQA models. Figure 2 provides an overview of the architecture. Our lightweight transformer based AQA model is a modification to the earlier proposed action recognition model [4] called Video Transformer Network (VTN). It consists of three components: 2D convolutional layers as the encoder, stacks of multi-head self attention and convolutional feed-forward layers as the decoder, and a fully connected layer as the regressor. We choose this particular model due to its rapid video processing speed, which is about 56 frames per second.

The Convolutional Encoder: The input to the model is a sequence of frames. The encoder is responsible for capturing the spatial features of each frame. It takes in the RGB data of each frame and outputs the feature embedding of it. The convolutional neural network used in the encoder is ResNet-34 and its weights are initialized with pre-trained weights from ImageNet. Re-using the parameters of ResNet-34 takes advantage of the transfer knowledge of feature extraction from image classification tasks. We also notice that ResNet-34 is a heavy convolutional network, we may adjust the model to remove some of the convolutional layers. At the end of ResNet-34, a global average pooling layer is applied to generate the frame embeddings from the extracted features. The final embedding of a frame has the size $d = 512$. The output of the encoder is then fed into the decoder.

The Self-Attention Decoder: The decoder consists of several stacks of multi-head attention and feed-forward layers. The multi-head self attention mechanism in the decoder is to establish a temporal correlation between frames. The tensor $Query(Q)$, $Key(K)$ and $Value(V)$ in the attention mechanism has the same shape as the encoder output $d \times t$ where t is the number of frames in the video. The key space, dimension of Q and K, and value space, dimension of V in computation of attention are d_k and d_v respectively. The proposed values for d_k and d_v are both 64. The weights W_i^Q, W_i^K, W_i^V for Q, K and V transform the inputs to fit the key space and value space. The attention and head output are computed as follows:

$$attention(Q, K, V) = softmax(\frac{Q^T K}{\sqrt{d_k}})V$$

$$head_i = attention(W_i^Q Q, W_i^K K, W_i^V V)$$

Using different weights for each head, we concatenate the h head outputs to compute the multi-head attention:

$$multihead(Q, K, V) = W^o concat(head_1, head_2, ..., head_h)$$

Number of heads to compute is 8 in our case. The multi-head attention is then passed to convolutional feed forward layers, here we use two 1D convolutional layers and residual connections. The first conv1d uses ReLU as the activation function and a length of 2048 convolution window. The second conv1d layer has a linear activation function and a convolution window of 512. The frame representations would undergo the multi-head attention and feed forward refinement processes N times before they are passed into the regressor. Based on previous experiments [4], $N = 4$ is enough to maximize the accuracy for the action classification task; we would also use $N = 4$ in the decoder implementation.

The Regressor: The resultant frame representations together with the difficulty levels of each clip are fed into a fully connected layer to regress a number as the predicted score. The loss function proposed is the mean squared error function. The Adam optimizer would be our choice of optimizer.

Data-set

The diving data-set we plan to use is part of the UNLV Olympic Scoring Data-set which consists of Olympic scoring of diving, gymnastic vault and figure skating. [8]. The diving data consists of 370 videos samples from Olympic diving. Each video is normalized to a length of 151 frames so that it can fit easily into the transformer model. The resolution of each video is 320×240. A low resolution would reduce the amount of information needed to be processed, thereby speeding up the training. The difficulty level of each video clip is also included in the data-set to aid the score prediction. The target label of the data-set is the overall score produced by the Olympic judges. In diving, the overall score is the product of the difficulty score with the execution score. Some of existing models predict only the execution score based solely on the video, however we would use the video together with the difficulty level to predict the overall score. The data-set has already been split into a training set of size 300 and a test set of size 70. We would implement a 10-fold cross validation to train and validate the training set. One limitation to this data-set is its small size of the samples. Even though a larger data-set is preferable for a more accurate result, considering the limited computational power we

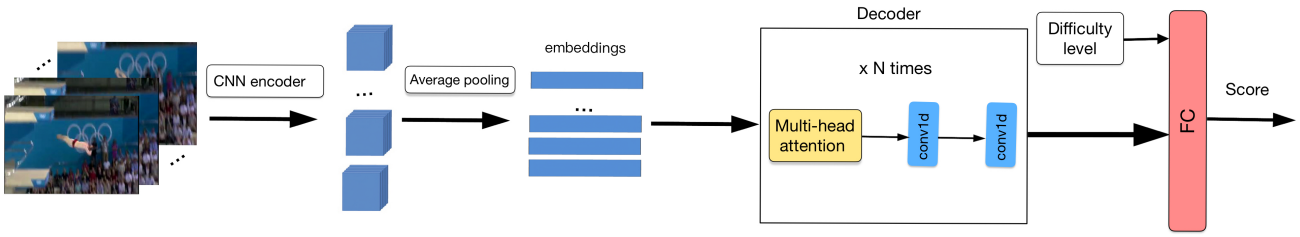

Figure 1: Overview of the VTN architecture for regression. Each frame is fed to a CNN model to produce an embedding. The decoder is repeated N times before the fully connected layer.

have and the computational intensive nature of video processing, we choose this medium size data-set among the publicly available diving data-sets. In addition, we use a batch size of 1 to further reduce time and space required for training.

We have also considered using the MTL-AQA data-set, which is similar to UNLV Olympic Scoring Dataset, but with annotations of diving techniques included. It consists of 1412 diving samples taken from different Olympic games. The videos's length range from 1 hour to 4 hours with HD quality. We opted for the former as it most closely resembled what could be witnessed in real time.

Experiment Planning

In order to quantify the performance of our proposed model, we will compare the score prediction accuracy and execution time against a baseline model: DCT pose-based SVR model [9] trained on the same data set. This baseline model is the first benchmark used for the AQA task. It makes use of mainly pose features to access the quality of the actions. First, the joint positions are extracted from each frame. Each position has x, y coordinates. The joint positions need to be normalized with respect to the head position. If $p^{(j)}(t)$ is the x value of the jth joint at time t, and $p^{(0)}(t)$ is the head, the normalized joint position is defined by:

$$q^{(j)}(t) = p^{(j)}(t) - p^{(0)}(t)$$

Each joint's x or y position in the video can be viewed as a function in time. This can be represented in the frequency domain using the Discrete Cosine Transform (DCT). Let $Q^{(j)} = Aq^{(j)}$ where A is the DCT matrix. Where $Q_k(j)$ selects the first k rows of A, the k lowest frequency components are selected as the feature vector:

$$\phi_j = |Q_k^{(j)}|$$

This is because high frequency components often contain noise due to pose estimate error. The concatenation of the feature vectors for each joint and each x, y coordinates forms the final feature vector. The final feature vector is trained using linear support vector regression (SVR) to predict the final score.

The experimental procedure will occur as follows: first, we will implement the proposed lightweight transformer model and the baseline model to measure the standard performance metrics of time and accuracy in our environment. Following the initial implementation, we will improve the accuracy of the model and its run-time; these methods will include our own inventions and/or be inspired by existing heavier models.

Results

Experiment Implementation

Our proposed methods were built on the Pytorch toolbox. The training was run on Google Colab with GPU acceleration, which is freely available to all users; while there is a time limit for free usage of GPU, note that our implementation will not challenge this restriction and was fully completed using Colab's free service. We implemented a 4-fold cross validation to estimate best training parameters. We opted for another version of the UNLV Olympic Scoring dataset called the AQA-7 dataset - this dataset contains the same data as the proposed UNLV Olympic Scoring dataset, except the videos are normalized to 103 frames instead of 151 frames. Among the 370 diving clips in the dataset, 300 are selected to be the training set and the remaining 70 clips form the test set.

Since our data-set consisted of videos, each data point would contain a large amount of data. This posed a challenge to the RAM required to process and train the data; Google Colab has a limited RAM for free accounts, if we processed and stored all the processed data before training, the program would crash due to a lack of RAM.

Originally, we attempted to split every single video frame-by-frame, then create a corresponding folder for the matching video, and finally store all the frames for that video into the folder. Based on those images, we used various python packages to load them into numpy arrays and perform the training process. However, due to large number of videos (370), and many frames per video (103), we would have 370 $\times$ 103 = 38,110 images, which would be stored in the aforementioned 370 folders.

Due to significant space consumption, we abandoned this approach and switched to saving data directly into numpy

array as files with .npt extensions; we now only store corresponding training and testing data numpy arrays with .npt extensions in our python environment. However, the same challenge remained. Although we converted 370×103 files into just 2 files (training and testing), the sizes of the training data numpy array and the testing numpy array were enormous - for training data, the numpy array is of shape $(300, 103, 224, 224, 3)$. After multiplying them, we would need to save a training file with $4,651,315,200$ entries and a test file with $1,065,306,880$ entries. Though we shrunk the number of files, the file is still far too big to train in our system.

After careful analysis of our code base, we realized: since we load the entirety of the data and whole numpy array into our code base and objects, as long as the size of the numpy array remained large, the data would persist in causing a memory problem. So, we attempted loading 5 videos at a time during the training process. Note that this implementation ensures that during the entirety of data processing, training and testing, there would not exist large numpy array. Not only does this implementation solve the memory issue, this also shed insight on the model's performance (training error, testing error) and training time within each load (we performed training and testing on the 5 videos per load). Hence, this approach was finalized as our final approach for data processing, without outputing any files and without loading any large sized numpy arrays.

The data was trained in batches of size 1. Adam was adopted for network optimization. 103 frames of a video clip were first fed into the convolutional block. The convolutional encoder was built based on the ResNet architecture. Both ResNet18 and ResNet34 were implemented to determine which network would yield the highest accuracy. An average pooling was applied after ResNet to extract features of size 512 for each frame. The features of all the frames then go through 4 stacks of the decoder unit. Each unit consists of a multi-head attention with dropout probability 0.1 and two 1D convolutional layers $Conv1d(512, 2014, 1)$ and $Conv1d(2014, 512, 1)$. We performed temporal average pooling at the feature-level after the decoder. A fully connected layer $nn.Linear(512, 1)$ was applied on the features before the final result was multiplied by the diving difficulty score of the video and returned as the predicted overall score. We trained the whole training set for a maximum of 5 epochs and best test accuracy among the 5 epochs are recorded as the accuracy for the model with a particular set of parameters.

Performance and Improvement

The accuracy of the model was measured on 3 metrics: the mean squared error, the L1 loss and the Spearman's rank correlation coefficient, which were most popular in the papers we assessed. Spearman's correlation is the most important metrics for the action quality assessment task as it assesses monotonic relationships between the actual scores and the our predicted scores. A higher Spearman's correlation coefficients reflects a more accurate prediction.

The model proposed contains a large number of parameters to be optimized and the training parameters also had a significant impact on the accuracy of the results. We conducted some experiments by varying the structure of the model and some training parameters. We will discuss some of these hyper-parameters we tested and adjusted to improve the accuracy of the model.

Both ResNet18 and ResNet34 are considered for the convolutional encoder, ResNet34 has almost double the number of trainable parameters as ResNet18. Keeping all other parameters the same, our experiment result indicates ResNet34 does not learn the features of the images well and resulted in lower test accuracy. Figure 2 illustrates the Spearman's correlation of the two networks as number of epochs trained varies. Model with ResNet34 also took almost double the time to train than a model using ResNet18 as the encoder.

In the model proposed in the proposal, we fed the difficulty score as a input to the final fully connected layer. In the final implementation, inspired by the actual calculation of overall scores in diving competition where the execution score is multiplied by the difficulty score to get the overall score for each dive. We separate the difficulty score from the final layer and instead, multiply the result returned from the last layer by the difficulty score. This means we treat the result returned from the last layer as the execution score and calculate the overall score based on the the difficulty score and the value returned by processing the video. As shown in Figure 3, multiplying the difficulty score separately leads to an improvement of accuracy as compared to our previous approach. Notably, this approach was considered because the original lightweight model which was built for classification. By separating the difficulty score, the model's task becomes solely predicting the sum of the judges' scores, which are each integers between 0 and 10, but limited to increments of 0.5 - this closer resembles a classification problem due to the discrete values.

Learning rate of the optimizer plays an critical role in the test accuracy. As indicated by Figure 4, changing the learning rate from 0.001 to 0.0001 double the Spearman's correlation.

The loss function used to train the model also affects the test accuracy. Mean squared error (MSE) and smooth mean absolute error, also known as smooth L1 loss, are two most commonly used loss functions for regression. Figure 5 suggests that training with mean square error loss function produced a higher accuracy than with the smooth L1 loss function for our action assessment task.

Taking into account the effects of the hyper-parameters, we train a model using ResNet18 as the encoder, with the difficulty score separated from the last feed forward layer and a learning rate of 0.0001 and MSE as the loss function. This model produced the highest Spearman's correlation coefficient and lowest mean squared error and L1 loss.

To judge the performance of of the proposed model, we compare the accuracy of our model to the posed-based DCT model. We modified the MatLab implementation by the author of the baseline model [9] to fit our dataset. The accuracy of trained Pose+DCT model is much lower than the expected result. One possible reason may be the huge difference in image quality and dimension between our dataset and the dataset used for training by the original paper. We

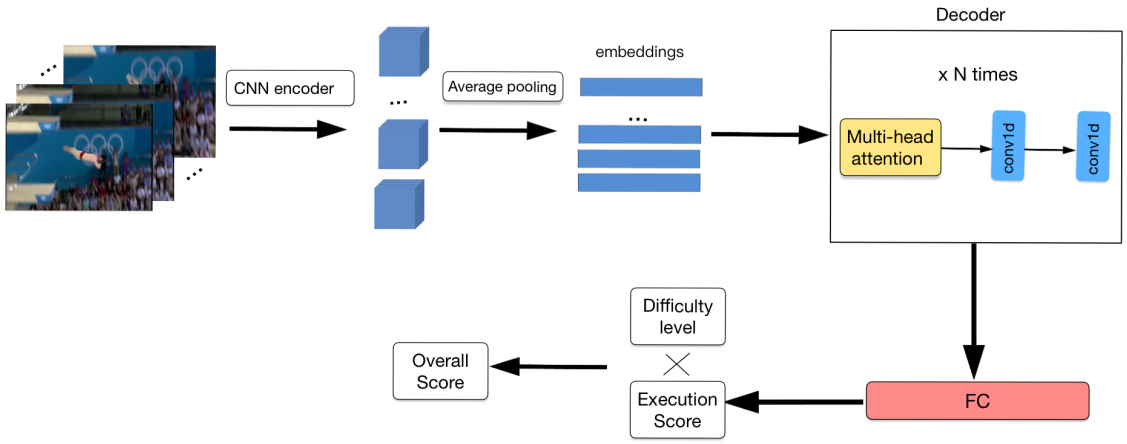

Figure 2: Updated VTN network.

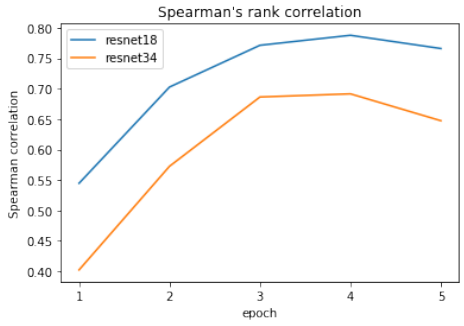

Figure 3: Spearman's correlation of ResNet18 vs ResNet34 encoders

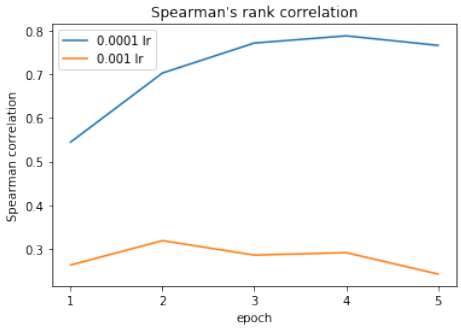

Figure 5: Spearman's correlation of 0.0001 lr vs 0.001 lr

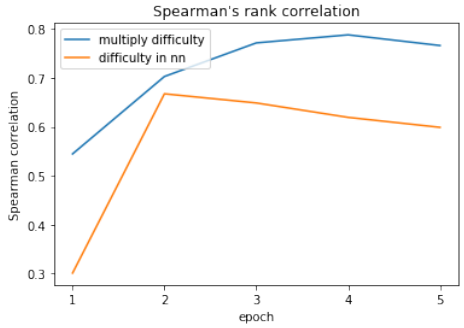

Figure 4: Spearman's correlation, multiplying the difficulty score separately vs feeding difficulty score to the neural network

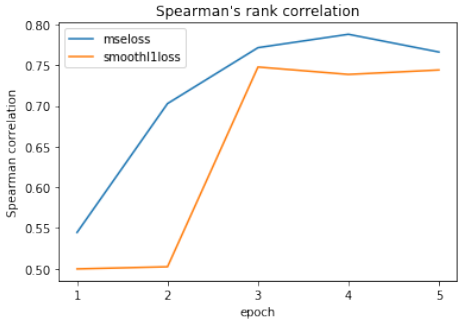

Figure 6: Spearman's correlation of Mean squared error vs smooth L1 loss

re-used the frozen model for pose prediction. The pose predictions by the frozen model are inaccurate and resulted in the low accuracy of Pose+DCT model on our dataset. Due to the inaccuracy of the frozen model, we will also quote the accuracy of the Pose+DCT approach on the AQA-7 dataset by [11].

Models	Sp's corr	MSE	L1 loss
Pose+DCT(trained)	0.1502	300.184	14.172
Pose+DCT(quoted)	0.5300	-	-
Our VTN algorithm	**0.7878**	80.2860	6.2261

Table 1: Comparisons of accuracy on the AQA-7 dataset between proposed VTN model and Pose+DCT model

As shown in Table 1, we achieved a significantly higher accuracy as compared to the both implementations of the Pose+DCT models. The average correlation of our VTN approach gains improvement of 48.7% to our baseline regression model.

Time efficiency is another goal of our proposed model. Training one epoch of 300 data points take about 12 minutes and running the whole test set of 70 predictions takes about 25 seconds on Colab with GPU acceleration (12GB NVIDIA Tesla K80 GPU). This gives an average run time of 0.296 seconds per video of 103 frames. About 95% of training and running time was spent on the convolutional encoder. Though we are unable to obtain run time for 3D-Convolutional models which are usually more accurate because we lack the systems to run them, we can estimate our run time to be shorter than the 3D CNN models as a 3D-Convolutional network usually takes 5-6 times more to train than a similar 2D-convolutional network for the same input data according to an experiment by Will Burton [1].

After finishing the implementation and training the model for the first time, the Spearman's correlation of the predicted score and the actual score was about 0.3. Without making much change to the model, we managed to increase the accuracy to 0.78 by tuning some of the training parameters. One take away from working on improving the model is that besides the structure of the model, there are many other factors such as learning rate and loss function that also have a huge impact on the performance of a neural network model. In the process of implementing the model and doing the training, there were many unforeseen circumstances occurred that led to constant adjusting and improving our method. If we had planned our code more carefully and implement the methods in a way that enables us to tune the parameters later, it could save us more time in parameter tuning so that we do not have to go back to make a big change in the code structure.

Additionally, it was useful to know how the original scores were obtained. This is important for consideration when possibly applying this model to other sports, as the scoring system could differ from diving.

Discussion

Interpretation of results

Revisiting the goal of the research, we aimed to construct a relatively reliable model to predict the the score of diving whiling reducing the run-time so that the model can be run in real time and is accessible to those with limited machine systems. We adapted a relatively lightweight video action classification model to solve an action assessment regression problem. Looking at the experiment result on the accuracy, our model outperforms the baseline model, indicating the approach of our transformer-based model is capable of producing a more reliable prediction as compared to the DCT posed based approach. The ResNet18 in the encoder captures enough details in each frame, together with multi-head self attention blocks in the decoder, our approach is able to extract the spatial-temporal features for the regression problem. The average L1 Loss is about 6.2, this means that the final diving score prediction is off by an average 6.2 points. The score range of the men's 10m platform diving in our dataset set is from 21.60 to 102.60. Considering the subjective elements in judging, a deviation of 6.2 points is relatively small in the range. We also achieved the goal of running the model in real time. The average run time for running a prediction of a 103 frame video takes 0.296 seconds on GPU acceleration (excluding I/O) and about 8 seconds without GPU acceleration. This gives rise to 348 and 13 frames per second respectively. The result suggests that our model is able to produce the a relatively accurate score seconds after the dive finishes.

Discussion of the interpretation

Table 2 shows the experiment results of our methods in comparison with other AQA approaches on the AQA-7 dataset.

Models	Sp's corr
Pose+DCT[9]	0.5300
ST-GCN [10]	0.3286
C3D-LSTM [8]	0.6047
C3D-SVR [8]	0.7902
JRG [3]	0.7630
MUSDL [11]	**0.8099**
Our VTN algorithm	0.7878

Table 2: Comparisons of action quality assessment accuracy on the AQA-7 dataset

Our algorithm outperforms all listed models except for C3D-SVR and MUSDL which use 3D-Convolutional Neural Networks C3D and I3D respectively in architecture. We expect our lightweight model to have lower prediction accuracy than the more complex models with C3D or I3D but the gap between our test accuracy to theirs is only about 3%. This effectively suggests that our approach is efficient and reliable at the same time. Pose+DCT, ST-GCN and JRG are all skeleton pose-based models and JRG which achieves the highest accuracy among the three also has I3D layers in their architecture. Comparing to our simpler appearance-based transformer model, JRG does not show an improvement in accuracy, which implies that pose-based approach may not be suitable for predicting the score of diving where water splash in the background would have a huge effect on the score. Our model also achieves a higher accuracy than the C3D-LSTM approach; this indicates the effectiveness of

attention in capturing the relationship between frames in the time domain.

Limitations

Due to the limited computational resources, we are unable to implement the more complex models involving C3D and I3D for comparison of run-time. Therefore our measurement for time efficiency can not the accurate. We acknowledge that our argument on the run-time comparison with 3-dimensional convolutional networks is only a rough estimate.

The dataset used for training only consists of clips from men's 10 metre platform diving and they all come from the same view. In this case, our model can only predict with high accuracy for 10 metre platform diving, the model may not generalize well for other diving categories and for videos shooting from a different view. There is also certainly the question of if it would be effective for the same event for women or para athletes, who are not represented in the data-set.

There are other convolutional neural networks we can consider for the encoder, we only implementd ResNet18 and ResNet34 due to our limitation in time and computation resources. There may be other networks which can result in a higher accuracy result or shorter run-time. Therefore, we acknowledge that our final setting may not be optimal for the proposed approach.

Conclusion

Our implementation shows that a lighter 2D convolutional network is able to perform at close to the quality of a 3D CNN with a reasonably fast run-time. This is sufficient in achieving our initial goal of building a lightweight neural network that would be more accessible and is likely usable in a real-time live diving competition, while only using what is freely available of Google Colab. Hopefully, this result would motivate further exploration into accomplishing complex machine learning tasks with limited resources, which would improve accessibility and usability to more people and for more purposes.

Through careful tuning, our results show that a ResNet18 encoder with a 0.0001 learning rate and MSE loss function, along with a separated difficulty score produces a predicted score with a Spearman's correlation of 0.7878 to the actual score - this only deviates about 6.2 points from the actual score. In comparison to our initial model, this shows that tuning and slight adjustments can make a major difference in performance. Other careful changes such as the change in the way we loaded videos also have significant impact on the resources needed. Through this research, we have shown that at times, limits in computing resources can be overcome with careful adjustments.

In the future, to improve this model's generalization, it would be worthwhile to include women and para athletes in the data-set for training. Both of these groups are also commonly represented at Olympics and Paralympics, so the data is not difficult to obtain. It would be interesting to see how the model we trained compares, and whether it could be biased for or against these new groups.

Looking further into diving, it would be a fascinating challenge to predict scores for synchronized diving. This becomes more complex with the additional synchronized score and additional diver, but it is not impossible that this model would be applicable in some ways. And of course, it would be worthwhile to attempt this model on for the 3 metre springboard event. The 3 metre event can be more complex with its springboard mechanics prior to the dive. Additionally, there are occasional failed dives (scored 0) in which the diver does not enter the water correctly (i.e. feet first) - this appears more common in 3 metre events where the springboard can be tricky to use and cause slipping or unbalance. Our data-set does not have any such dives, so this is important for consideration. Otherwise, there may be minimal necessary changed made to our tuned model as all other criteria are relatively similar. For an additional diving challenge, the difficulty score could be predicted from the dive rather than given beforehand - this would require an additional classification step.

As papers before ours have shown that their models can be applied to other events such as gymnastics or figure skating with mild success, the same can be asked of our model. Further, we are especially curious for the potential of our model to be applied for training, suggestions, action analysis for real-time injury prevention, and actions in other areas outside of sports.

References

[1] Will Burton. *2D or 3D? A Simple Comparison of Convolutional Neural Networks for Automatic Segmentation of Cardiac Imaging*. Apr. 2019. URL: https://towardsdatascience.com/2d-or-3d-a-simple-comparison-of-convolutional-neural-networks-for-automatic-segmentation-of-625308f52aa7.

[2] FIFA. *2018 FIFA World Cup™ - News - Refereeing and VAR at the 2018 FIFA World Cup: A new era for football*. July 2018. URL: https://www.fifa.com/worldcup/news/refereeing-and-var-at-the-2018-fifa-world-cup-a-new-era-for-football.

[3] Jibin Gao Jia-Hui Pan and Wei-Shi Zheng. "Action assessment by joint relation graphs". In: *The International Conference on Computer Vision (ICCV)*. 2019, pp. 6330–6339.

[4] Alexander Kozlov, Vadim Andronov, and Yana Gritsenko. "Lightweight network architecture for real-time action recognition". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 2020, pp. 2074–2080.

[5] Mahdiar Nekoui, Fidel Omar Tito Cruz, and Li Cheng. "FALCONS: FAst Learner-grader for CONtorted poses in Sports". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 3941–3949. DOI: 10.1109/CVPRW50498.2020.00458.

[6] Paritosh Parmar and Brendan Morris. "Action Quality Assessment Across Multiple Actions". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 1468–1476. DOI: 10.1109/WACV.2019.00161.

[7] Paritosh Parmar and Brendan Tran Morris. "What and How Well You Performed? A Multitask Learning Approach to Action Quality Assessment". In: *CoRR* abs/1904.04346 (2019). arXiv: 1904.04346. URL: http://arxiv.org/abs/1904.04346.

[8] Paritosh Parmar and Brendan Tran Morris. "Learning to Score Olympic Events". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2017, pp. 76–84.

[9] Hamed Pirsiavash, Carl Vondrick, and Antonio Torralba. "Assessing the Quality of Actions". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 556–571. ISBN: 978-3-319-10599-4.

[10] Yuanjun Xiong Sijie Yan and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2018, pp. 7444–7452.

[11] Yansong Tang et al. "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.

[12] Vinay Venkataraman, Ioannis Vlachos, and Pavan K Turaga. "Dynamical Regularity for Action Analysis." In: *BMVC*. 2015, pp. 67–1.

[13] Chengming Xu et al. "Learning to Score Figure Skating Sport Videos". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.12 (2020), pp. 4578–4590. DOI: 10.1109/TCSVT.2019.2927118.