Video Understanding by A Sequence Model: A Case Study on Golf Swing Sequencing

Yanming Zhu
Stanford University
yanmingz@stanford.edu

Abstract

We investigated the effectiveness of video understanding through a case study of golf swing event sequencing on the GolfDB dataset. The performance of sequencing on spatial dimension only via static image classification is compared with the performance of sequencing with both spacial and temporal information. The static image classification model is implemented by fine-tuning a pretrained network. The sequence model extends the fine-tuned model with a customized transformer encoder to capture the temporal context. The result shows static image classification yields strong baseline results at 71.5% PCE. Incorporating temporal information further improves performance significantly, increasing PCE to 78.1%. We conclude the temporal component is critical in achieving better results in video understanding for this task. Our source code is publicly available at https://github.com/yanmingzhu/cs231n-golf.git.

1. Introduction

Video understanding is a challenging problem in computer vision. While static image classification has made strides in the years past, video understanding hasn't progressed as fast. As experienced in Karpathy [3], static image classification used in video applications usually yields strong baseline results, but performance gains from incorporating additional temporal component may be minimal.

We want to study this problem in a specific domain: golf event sequencing. A correct swing form is critical to the performance of a golf player. Our task is to identify eight events that comprise the swing sequence using computer vision. One of the previous works in this field is GolfDB [1], which provides a dataset consisting of 1,400 short video clips of golf swings with event labels. This problem is particularly interesting to us because it poses a significant challenge for the image classification model, which may result in a gap in performance that a static image model may not be able to overcome. We aim to determine whether a sequence model with the help of a temporal component may be able to fill this gap.

2. Problem Statement

A swing sequence consists of the following consecutive 8 events, as defined in McNally [7].

- Address (A). The moment just before backswing starts.
- Toe-up (TU). Club parallel with ground.
- Mid-backswing (MB). Arm parallel with ground.
- Top (T). The moment when backswing changes to downswing.
- Mid-downswing (MD). Arm parallel with ground.
- Impact (I). Club hitting the golf ball.
- Mid-follow-through (MFT). Club parallel with the ground during the follow-through.
- Finish (F). The moment before the player relaxes.

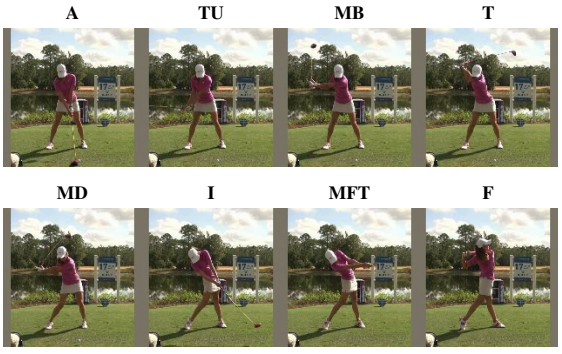

Figure 1. Visualization of the Eight Events

The goal of the model is to identify the frames associated with each event, where an event is associated with one

frame only. For one video clip, we expect one event for each event type and the rest are considered non-event frames.

To identify the frame of an event, we may use image classification based on the pixel value of each frame only. This is a fine approach and may produce strong baseline results as seen in later sections. However, there is also an inherent limitation as image classification looks into one frame only without taking into consideration its relationship with other frames.

Figure 2 shows the event of a single full swing together with their neighboring two frames before and after. For event "I", it can be seen that there is considerable differences between the event frame and its neighbors. Intuitively these differences may provide good heuristics for image classification to perform well.

On the other hand, for type "A" and "F", and to a less degree type "T", their event frames are virtually indistinguishable from their neighbors. The challenge coming from these event types is the result of the player staying still, which produce multiple frames of highly similar images. From image classification point of view, these frames don't provide enough separation among them for the model to make correct decisions. It's highly possible image classification models may struggle with these event types.

3. Related Work

McNally [7] introduced the GolfDB dataset and a reference implementation SwingNet. The GolfDB dataset consists of 1400 short video clips of golf swings of professional golf players. The SwingNet combines a CNN with a bi-directional LSTM to achieve a 76% accuracy detecting the swing events. This paper also proposed PCE as a metric to evaluate the performance for this task.

Hajian [2] trained a SwingNet with rotation augmentation and used it together with another model to compare the swing of two players. It achieved a slightly higher PCE score of 72.5% compared to the pretrained SwingNet model of 71.5%.

Zhang [11] is a more recent work on this problem and appeared to have achieved stronger results. The key ideas behind its approach include an attention mechanism to fuse the temporal information. Additionally it introduced a Gaussian kernel for soft label generation to help better differentiate frames that are visually similar. It achieved an accuracy of 83.4% which is much higher than the previous works.

Liu [5] attempted to evaluate the quality of a golf swing by analyzing human body key points. Though not directly on GolfDB, this work is interesting to us since it suggests a correlation between golf swing and human body key points. As a result of this work, we hypothesize that incorporating human pose features may help improve the accuracy of prediction.

Similarly Ko [4] is a CNN and bi-LSTM golf swing analysis work that focuses on the correct form of a golf swing. The main approach is analyzing the angles and twist of the upper body, head, shoulder and pelvis.

4. Dataset

We use the GolfDB dataset introduced in McNally [7]. A total of 1400 short video clips are collected from 580 YouTube videos. They contain a mixture of videos with regular and slow-motion, male and female players, different face angles, and various club types. Each video clip is also annotated by human annotators and reviewed by experienced golf players. Each of the eight events is annotated with its exact frame within the clip. There are also additional annotations such as bounding boxes which we won't be using in our study.

Among the 1400 clips, 1050 are used for training and the remaining 350 are used for final evaluation. Due to the scarcity of the data, the validation dataset is generated by augmentation.

4.1. Preprocessing

The video clips from GolfDB are cropped from the original video. Each frame of the video has the size of 160 in height and 160 in width.

Since we experimented a lot of image classification models that are pretrained on the ImageNet dataset, we preprocess each video frame by resizing it to 224 in height and width, which is the standard size ImageNet. We further normalize each image by the mean and standard deviation of the ImageNet dataset.

4.2. Class Weighting

Our cross entropy loss function expects 9 classes which includes eight classes corresponding each of the 8 event types, and one additional class non-event class designating a frame absent of any event.

Each video clip contains about 281 frames, where eight of them are assigned to the eight event classes, and the rest are assigned to the non-event classes. Unfortunately this means our class model is highly unbalanced, with the non-event class far outweighs any of the other events. A regular cross entropy loss function likely will yield sub-optimal results.

4.3. Data Sampling

Each video clip has 283 frames on average in our dataset. Clearly it requires multiple sequences to finish each video clip. It also means we cannot make any assumption about the starting frame of the sequence during final evaluation as it may start from anywhere depending on how the frames are chunked.

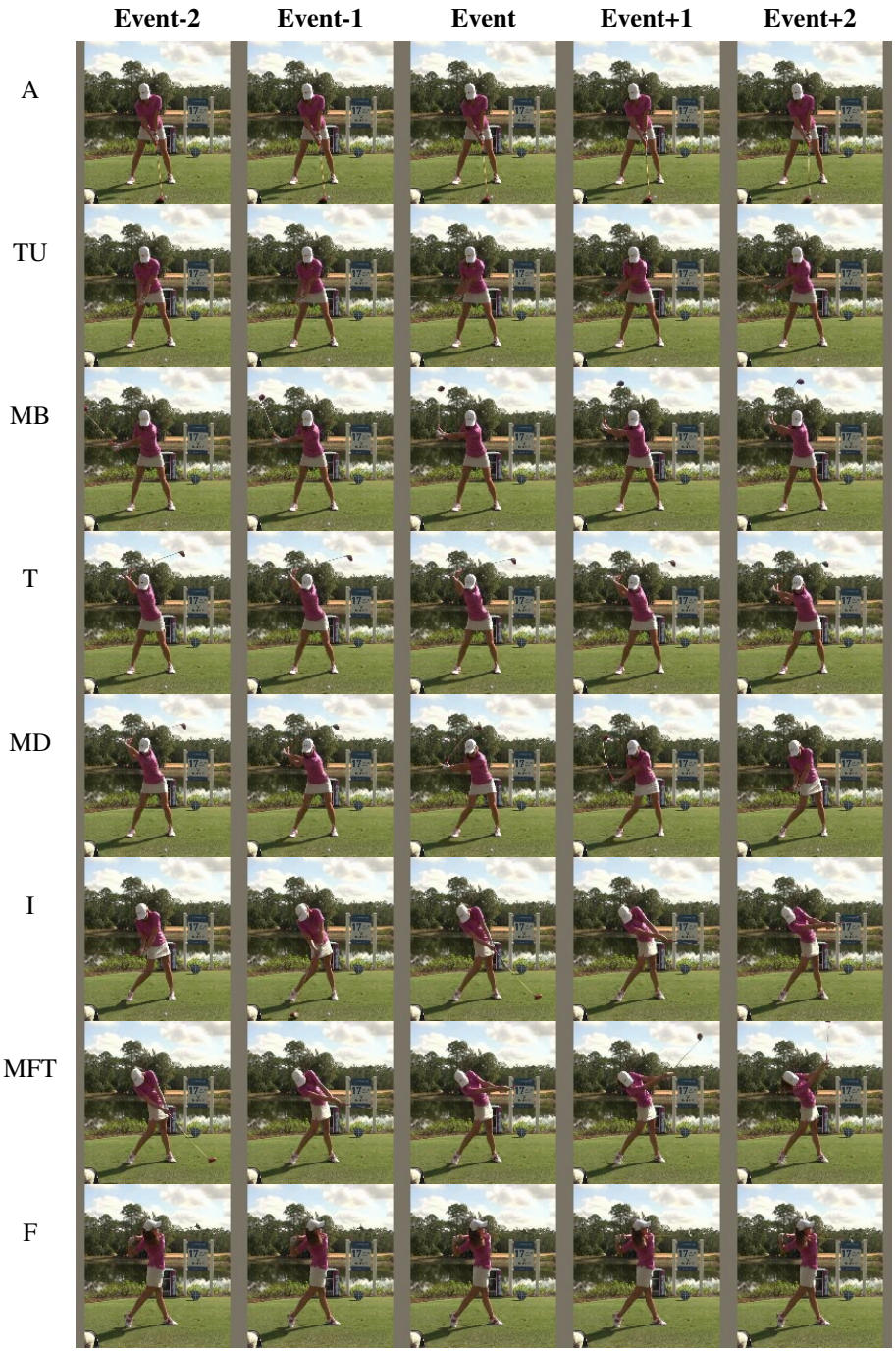

Figure 2. The difficulty of static classification. Each row contains the frame of an event and the two frames before and after. Row "A" and "F" show striking resemblance between all frames.

GolfDB comes with a customized dataloader that samples the starting position of a sequence, which we consider a good choice as it's commonplace in action recognition [1]. However the dataloader also doesn't take into account the end of the video clip. If it encounters the end of the video when constructing a sequence, it simply wraps around to the start of the video.

This loop around approach isn't a problem for image classification models but it has large implications on our sequence models. In real word or evaluations however, the frames are not loop over. The bias introduced by the loop simply doesn't generalize. We consider it a better choice

to stop at the end of the video clip, and leverage padding masks to fill out the rest of the sequence.

We started out with regular class weighting based on the formula

$$w_c = \frac{N}{k \cdot n_c}$$

, where N is the total number of labels, k is the number of classes, and n_c is the number of samples for class c. However, the resulting weight as shown in Table 1 is too extreme where an event class weight that is 320 times higher than the non-event class. To smooth out the class weight, we use a smoothed version of logged class weight below, where μ is a constant factor set to 0.3.

$$log\_w_c = \frac{\log\left(\mu \cdot \frac{\max(n)}{n_c} + 1\right)}{\max_c \log\left(\mu \cdot \frac{\max(n)}{n_c} + 1\right)}$$

5. Methods

We begin with an image classification model that entirely based on its prediction on the pixel values of a single frame image. We want to use this model as a baseline for us to compare test results against, as well as a building block for the sequence model we'll build later.

Since this task is about human sports, we hypothesize there is correlation between human pose and the event type. As a second attempt, we want to try if an image classification based on human pose would yield better results than from pixel value alone.

Finally, we build a sequence model to capture the relationship among the frames. This approach is previously taken by McNally [7] with an bidirectional LSTM model. Since LSTM performance degrades as sequence length grows, we want to use attention based transformer model that may scales better with the sequence length.

5.1. Image Classification with ConvNeXt

This approach seeks to fine-tune a pretrained image classification model with the GolfDB dataset. Our choice of the model weights equally on performance and efficiency. Performance wise we want the model to give us a strong baseline to start with without being the bottleneck when incorporated to the sequence model. Since we expect the sequence model to be considerably more difficult to train in terms of its computing resources required, we also want the model to be lightweight and efficient without taking resources away from the sequence model.

Initially we tested a ViT base 16 model. While it performs decently, it's more difficult to train with the transformer sequence model. In comparison, ResNet family models are much easier to train.

Eventually we found ConvNeXt [5] Tiny is a better choice. ConvNeXt is a modernized ResNet type model that takes inspiration from ViT. It retains the ease of training from ResNet that we found very attractive while providing significant performance improvement over ResNet. For example, ConvNeXT Tiny scores 82.1% top 1 accuracy compared to ResNet 50's 76.1% on ImageNet.

To fine-tune the ConvNeXt model, we replaced its classifier head with a linear layer that projects to 9 dimensions corresponding to the 9 event classes. We also enabled training the rest of the classifier layer, and its "7.2" layer.

5.2. Image Classification with Pose Estimation

Golf swing is a human-centered action. A generic image classification model is generally trained on a diversified dataset of images, with no focus on human action. As an attempt to improve the performance over the simple image classification approach, we want to test if it's helpful to feed the model with additional human pose data. The hypothesis is that the human body position and pose is correlated with the movement of the swing action, and may shed light on the exact phase during which the swing is taking place.

To extract the human pose key points, we considered a few models including ViTPose [10], WHAM [8], and Yolo-pose [6]. Both VitPose and WHAM require considerable effort to integrate into our model while Yolo-pose is relatively simple. Yolo-pose performs both object detection and pose estimation. The model returns both the bounding box of the detected object and the key points associated with the object that's necessary for pose estimation.

To fine-tune this model, we project the output from Yolo-pose into a 780 dimension vector, which is then used as an input to a linear classifier that produces the class logits.

Figure 3 illustrates a sample frame where the bounding box and key points are extracted.

Figure 3. An example of bounding box and key points captured by Yolo-pose

5.3. Sequence Model

As stated previously, image classification may perform well on some event types such as "MD", "I" and "MFT",

Event	A	TU	MB	T	MD	I	MFT	F	Non-event
Count	1400	1400	1400	1400	1400	1400	1400	1400	394177
Class Weighting	32	32	32	32	32	32	32	32	0.1
Log Class Weighting	1	1	1	1	1	1	1	1	0.059

Table 1. A comparison of regular class weighting and log class weighting

where the action is swift and event frame differs visually from its neighboring frames. On the other hand, it may struggle on some other event types such as "A", "T", and "F", where the player stays relatively still, and multiple frames are generated with similar visuals. For these event types, it's very difficult to improve the accuracy by simply improving the image classification models as there isn't enough data for the model to make better predictions. To overcome the limitation that's inherent with prediction based on a single frame, we need to go beyond image classification and capture the relationships among the frames.

For example, while event type "T" is a relatively difficult one for image classification, the event is always sandwiched between event "MB" and "MD". If the sequence model is able to capture this relationship, better prediction may be achieved over image classification models.

LSTM used by McNally [7] is a reasonable choice of sequence model at the time. But we want to see if transformer models would perform better over LSTM. There are two reasons that lead to our preference on transformer. First, LSTM doesn't scale to long sequences. A typical swing sequence on our dataset is around 1 second which takes about 32 frames in a real time video, but it would be much longer in a slow motion video. Secondly, we hypothesize the attention mechanism enables the model to capture more meaningful relationships between swing frames than the "summarization" that an LSTM uses to represent a sequence.

One of the critical considerations of the attention is the positional embedding. In our use case, the absolute position of each frame isn't nearly as important as the relative position between the events. For example, the model shouldn't care if the swing starts at frame 0 or frame 10, but it might find it helpful to know that "T" always follows "MB" in a few frames, and leads "MD" in another number of frames.

Our choice of relative position embedding is RoPE [9]. RoPE is a rotary positional embedding used by a few large LLMs. It's effective and easy to implement. Due to the lack of support from PyTorch on RoPE at the time of this writing, we implemented RoPE with the help of ChatGPT in our code base. The RoPE implementation is encapsulated in a customized transformer encoder layer and integrated into the PyTorch transformer encoder.

We use the fine-tuned ConvNeXt model as a feature extractor for the sequence model. The weights of ConvNeXt is obtained from our training of the image classification where ConvNeXt has already been trained with the GolfDB dataset and achieved good result. We removed the final layer of the classification head and feed the output of the last hidden layer to the sequence model. This representation contains rich embeddings by ConvNeXt that's already effective in this task.

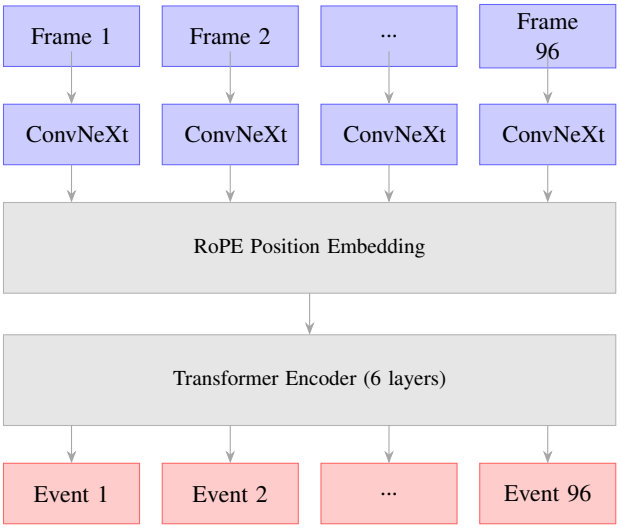

Figure 4. The transformer encoder based sequence model with 96 time steps and RoPE positional embedding

The sequence model is a transformer encoder that contains six encoding layers with RoPE implementation as illustrated in Figure 4. The sequences is set to 96 time steps, where each step takes one frame representation from the ConvNeXt, and outputs a logit of the nine event classes. Internally, the transformer is configured with 8 heads and 256 in hidden dimension. The choices of these parameters are both empirical and a compromise with the computing resource. From our experience six layers scales slightly better than four layers, and 256 hidden dimension works slightly better than 128. The choice of 96 time steps is a compromise between performance and computing limitation. While we want longer sequence to allow the attention see more relevant frames, longer sequence requires more memory and significantly more compute, neither of which we have much to spare. In the end, a length of 96 proved to be effective while being manageable on an A100 GPU.

5.4. Implementation

Our implementation is derived from the SwingNet of McNally[7] at https://github.com/wmcnally/golfdb. The codebase includes the data file of the GolfDB database, the dataloader that reads the datafile and video clips, and implementation of SwingNet. We make heavy use of the dataloader, and some of its training and evaluation utilities.

We also seek help from ChatGPT for the implementation of RoPE since PyTorch doesn't provide direct support.

Our own code base focuses on an implementation of fine-tuned ConvNext model, integration of the human pose estimation from Yolo-pose, as well as the transformer sequence model that's built upon a customized encoder layer with RoPE support. Other area of our implementation include the learning rate schedule, data sampling without looping and its associated masking, and the general training framework that allows checkpointing.

6. Evaluation Metric

We use PCE as the evaluation metric introduced in McNally [7]. PCE stands for "Percentage of Correct Events", measuring the correct prediction of the 8 events listed above.

To compensate for the variability of human annotation, the PCE is calculated with a tolerance that's scaled with frame rate.

$$\delta = max(\lfloor \frac{n}{f} \rceil, 1)$$

where n is the number of frames between Address and Finish, f is the sampling frequency. Since the average time of a swing is about 1 second, for regular frame rate of 30 fps, the δ is 1, meaning a prediction is considered correct if it's within 1 frame of the annotation.

7. Experiments

7.1. Hyper Parameters

We need to makes a few decisions on hyper parameters. Batch size is critical for this task. Our experience shows when the batch size is too small, training doesn't converge due to possible large sample variance. We also don't want to set the batch size too large as it takes more compute as well as exhaust the training set too quickly. The batch size is set to 22 based on our training experience.

AdamW is chosen over Adam as the optimizer due to its better regularization which we consider helpful for training the transformer.

Another critical parameter is learning rate. While ConvNext may be more forgiving on learning rate, the transformer may be more difficult to learn without the proper learning rate setup. To improve the stability of training, we create a learning rate schedule with warm up and cosine annealing, where during the first 50 iterations the learning rate is linearly warmed up, and the rest of the training sees a drop in learning rate according to cosine annealing.

The training is divided into 800 iteration training segments. Each segment may have different base learning rate which is summarized in Table 2

	800	1600	2400	3200	4000	4800
ConvNext	5e-4	5e-4	5e-4	4e-4	-	-
Sequence	2e-4	1e-4	5e-5	4e-5	3e-5	2e-5

Table 2. Base learning rate for each 800 iterations. The real learning rate is subject to warm-up and cosine annealing.

7.2. Regularization

The dataset size of GolfDB may be considered small in today's standards with 1050 training video clips. But since each video clip contains 283 frames on average, the total training frame is about 300,000, which is of decent size on a specific domain. Nonetheless, overfitting is a concern and take a few regularization measures.

- L2 weight decay: This is provided through AdamW optimizer. We found its default rate of 1e-2 works well.
- Dropout: In our custimized RoPE enabled transformer encoder layer, a dropout layer is added just like any typical transformer layer. The dropout rate is set to 0.1
- Random sampling of starting frame. This is already part of the GolfDB dataloader behavior. We consider it provides additional regularization with the noise introduced by randomization.

7.3. Training Classifier with Yolo-pose

While we were able to extract the representation of bounding box and key points from Yolo-pose output, the training wasn't successful. The loss didn't converge despite tuning various hyper parameters.

There are a few possible reasons. The feature representation might be at too high level as they are the final output ready for post processing. A lower level representation may suit better for our purpose.

Additionally, the final representation is only about the information inside the bounding box. Though it captures all human pose key points, the golf club is missing when it lands outside the bounding box.

We consider this approach requires more exploration and research.

7.4. Training ConvNeXt and Transformer

The configuration of the training set up is summarized in Table 3.

	ConvNeXt	**ConvNeXt w/ Transformer**
ConvNeXt Model Spec	ConvNeXt Tiny, ImageNet pre-trained	GolfDB fine-tuned
Sequence Model Spec	–	8 heads, 256 hidden dim
Batch Size	22	22
Sequence Length	32	96
Optimizer	AdamW	AdamW
Weight Decay	1e-2	1e-2
Dropout	None	0.1
Learning Rate	See Table 2	See Table 2

Table 3. Training specifications for ConvNeXt and ConvNeXt with Transformer.

Following the training setup in Table 3, we were able to successfully train both models in a reasonable amount of time.

As illustrated in Figure 5, fine-tuning ConvNeXt took about 2400 iterations before the accuracy tops out at 71.5%. With an additional 800 iterations of training, the accuracy didn't improve and stayed almost the same at 71.4%. We considered the training converged at this point for fine-tuning the pretrained ConvNeXt model.

As expected the transformer based sequence model was more difficult to train. It took 3200 iterations to top out at 78.0%. We further trained the model for another two 800 iterations, which yielded PCE of 77.9% and 78.1% respectively.

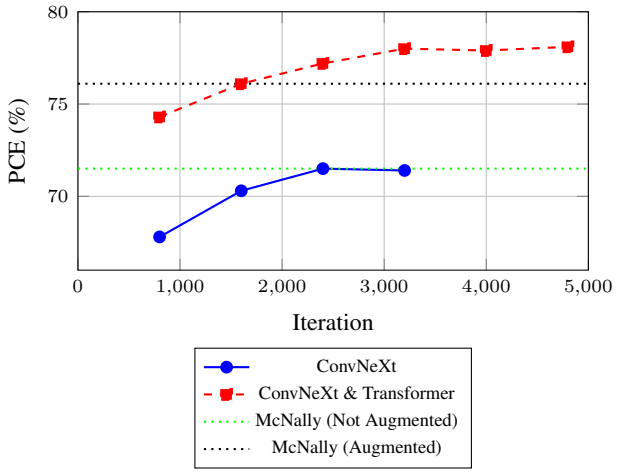

Figure 5. Model accuracy over training iterations with baseline.

7.5. Result Analysis

Table 4 summarizes the test results of our models together with other models we mentioned before. Note that our models were trained with the original video clips without any augmentation due to time constraints. All other models were trained with augmented data, with SwingNet-160 trained on both.

Our ConvNeXt model matches Swing-160 (non-augmented) on overall PCE. Note Swing-160 has both a MobileNet V2 and a bidirectional LSTM. It shows again what Karpathy [3] stated that image classification model can be a very strong baseline for video applications.

The result of our transformer based sequence model showed strong performance improvement over the ConvNeXt model alone, raising PCE from 71.5% to 78.1%. It also outperformed the Swing-160 in both its augmented and non-augmented variations. The class-wise PCE gain is visualized in Figure 6. All event types achieved positive gains.

It's expected that most improvement would come from event "A", "T", and "F" as they have lower PCE in the ConvNeXt result. But it's not without surprise that event type "T" achieves the most improvement, given it is far better than "A" and "F" with ConvNeXt already.

One possible explanation is that attention plays a role in this result. Since event type "T" sits in the middle of the swing sequence, it's able to attend to both left and right side the events. "A" and "F" on the other hand sit at start and end of the sequence, and is only able to attend to one side.

Despite the significant performance improvement over the original SwingNet-160, Zhang [11] still have a 5.3% lead in overall PCE. We will continue investigate whether data augmentation will fill this gap.

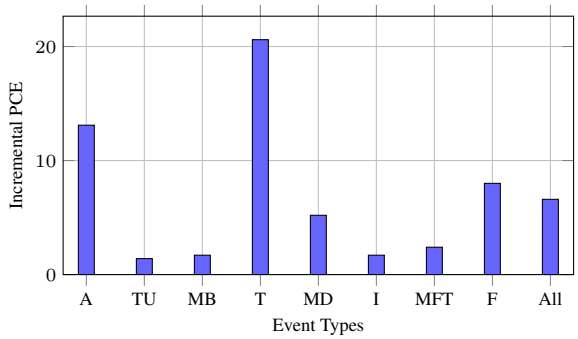

Figure 6. Class-wise Incremental PCE. Event "T" gained the most from the seqence model at over 20%. Event "A" and "F" also had significant gains at 13% and 8% respectively.

8. Conclusion

We considered both the ConvNeXt model and the transformer based sequence model perform well. ConvNeXt as a static image classification model alone matched the performance of the LSTM SwingNet model. The transformer sequence model achieved significantly higher PCE scores, highlighting the power of the attention mechanism in understanding video frame sequences.

Model	Data Augmented	A	TU	MB	T	MD	I	MFT	F	PCE
SwingNet-160	No	-	-	-	-	-	-	-	-	71.5
SwingNet-160	Yes	31.7	84.2	88.7	83.9	98.1	98.4	97.6	26.5	76.1
Hajian	Yes	27.7	80.3	88.0	75.1	97.4	94.6	96.9	20.3	72.5
Zhang	Yes	**50.0**	**94.9**	**93.1**	**91.4**	**98.9**	**99.7**	**99.4**	**39.7**	**83.4**
ConvNeXt (Ours)	No	18.0	86.6	87.7	68.0	93.1	97.7	96.9	24.0	71.5
Sequence (Ours)	No	31.1	88.0	89.4	88.6	98.3	99.4	98.3	32.0	78.1

Table 4. PCE Score Comparison. Our models are trained without data augmentation while most other models are.

However, the performance of the models may be limited by the training data size. As a follow-up, we plan to augment the training data and hope to achieve additional performance gains.

9. Acknowledgements

I want to thank TA Jiaman Li for her suggestions and feedbacks on this project.

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[2] A. Hajian, K. Sookpreedee, K. Phairoh, W. Ruangsang, and S. Aramvith. Deep learning-based golf swing sequence analysis. In *TENCON 2023-2023 IEEE Region 10 Conference (TENCON)*, pages 926–931. IEEE, 2023.

[3] 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 (CVPR)*, pages 1725–1732, 2014.

[4] K.-R. Ko and S. B. Pan. Cnn and bi-lstm based 3d golf swing analysis by frontal swing sequence images. *Multimedia Tools and Applications*, 80(6):8957–8972, 2021.

[5] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s, 2022.

[6] D. Maji, S. Nagori, M. Mathew, and D. Poddar. Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2637–2646, 2022.

[7] W. McNally, K. Vats, T. Pinto, C. Dulhanty, J. McPhee, and A. Wong. Golfdb: A video database for golf swing sequencing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[8] S. Shin, J. Kim, E. Halilaj, and M. J. Black. Wham: Reconstructing world-grounded humans with accurate 3d motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2070–2080, 2024.

[9] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[10] Y. Xu, J. Zhang, Q. Zhang, and D. Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in neural information processing systems*, 35:38571–38584, 2022.

[11] Y. Zhang, F. Tu, Z. Wang, W. Guo, and D. Zhu. Learning golf swing key events from gaussian soft labels using multi-scale temporal mlpformer. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2023.