Utilizing Advancements in Transfer Learning and Video Analysis to Improve Golf Swing Action Recognition

Zachary Dawson, Andre Green, Benjamin Kinard
Georgia Institute of Technology
{zdawson7, agreen67, bkinard7}@gatech.edu

Abstract

This paper explores alternative architectures to SwingNet [4], a lightweight deep neural network for golf swing sequencing. Using the GolfDB dataset, provided by the authors of SwingNet, we trained and evaluated 3 model architectures. The first model explored the use of alternative loss functions in the training of the original architecture, while the other 2 models experimented with the use of 3D CNNs. Results showed that the R3D model architecture was capable of achieving superior performance than the baseline model.

1. Introduction

Golf swing sequencing is the process of identifying key events in a video of someone swinging a golf club. SwingNet [4] is a deep neural network designed to do just that. Introduced in 2019, the model is a hybrid convolutional and recurrent network that extracts features from video frames using MobileNetV2 [1] and then passes those features to a bidirectional LSTM network. The output of the LSTM layer is finally passed through a fully-connected layer with softmax to predict the probability a frame contains one of 8 key golf swing events.

SwingNet was able to correctly identify key golf swing events at a rate of 76.1% at the time of publication in 2019. As recent as 2023, there have been improvements to 83.4% [7]. The improvements came from new model architectures using a Multi-Scale Temporal MLPFormer and Gaussian-based classification. This suggests that using some different temporal feature extractor and a loss function that allows for noise can lead to great improvements over the baseline model.

We attempted to replicate and improve upon the original SwingNet architecture using different loss functions. SwingNet uses a 2D CNN followed by an LSTM. We utilized class-weighted cross-entropy (CE), KL Divergence, and class balanced focal loss. We also explored two model architectures that leveraged 3D CNNs pre-trained on the Kinetics400 dataset [6]. One was followed by a 1D CNN and the other by a Fully-Connected Layer.

We trained and evaluated our models on the GolfDB dataset, the same dataset SwingNet was trained and evaluated on. The GolfDB dataset includes 1,400 videos of golf swings that are both down the line and face on. The 1,400 samples were clipped from publicly available YouTube videos and include both male and female professional golfers. All videos have been trimmed and cropped to include only the golf swing with minimal background and to not include significant amounts of time before and after the golf swing. The videos vary between normal speed and slow motion. The total runtime of all the videos is 219 hours and contains 390k total frames. The dataset is interesting from a Deep Learning perspective because it is highly imbalanced, with no-event frames outnumbering event frames at approximately 35:1. The golf swing is also very fast (¡ 2 seconds), and different instants in the swing look similar but are differentiated only by temporal sequencing (e.g., halfway through the back-swing vs. halfway through the down-swing).

Improving a golf swing is a difficult and meticulous process. Reviewing video footage of previous attempts is a common strategy used by golfers to make improvements to their swing. Golf swing sequencing has the potential to enhance this analysis and we hope that improving upon the originally proposed model will make it even more useful. Any promising results could be utilized in other sports as well, since many sports contain actions that are fast and irregular.

2. Approach

Our experiments ranged from improving existing model architectures to building new model architectures. We trained and tuned our models in an attempt to match or improve upon the baseline set by SwingNet in 2019. Our training utilized many of the evaluation and training loops from the original paper [4]. All of their code was available on

GitHub and made replication easier. We implemented all of the models ourselves from scratch using PyTorch [5]. We will now explain each model and any model specific experimentation details below.

For training, each model used sequence lengths of 64. This means, each video was cut down to a random segment of 64 frames. This is a necessary step to enable loading multiple videos into the GPU memory at once. Since you have to run predictions over the entire sequence to calculate our main metric, PCE, we could not calculate per epoch PCE on the training set. Validation was always done one video at a time to allow for PCE calculations and to manage GPU memory.

We used a train/val/test split of 0.5625/0.1875/0.25. The original paper tuned their model on the validation set and then used the validation set as the test set. This calls into question the validity of their results, as tuning a model to the test set leads to data leakage and may overestimate the performance of the original SwingNet architecture. However, we will compare our results to the original paper as if this mistake had not occurred.

2.1. Model 1

Model 1 is intended to replicate the results from the original SwingNet architecture and experiment with different loss functions. The architecture includes a 2D CNN (MobileNetV2, pre-trained on ImageNet), for feature extraction. This sequence of feature representations is then used in a bidirectional LSTM to predict the class of each frame. The original paper used weighted cross-entropy loss for the classification task of each frame. We believed that this was an inefficient way of determining the loss for this problem. We believed this because CE does not effectively measure how close in time the predicted frame is to the true frame. In order to address this issue, we implemented a custome loss function (soft class-based KL-divergence) which will be explained in detail later. We believed this would help learning because the loss function does not punish predictions that are 1 frame off as severely as those that are 5 frames off. We also used CB Focal loss as a comparison loss function. We believed this function may benefit the classification of the harder classes (namely address and finish). From now on, we will be referring to the three versions of Model 1 as Model 1 CE, Model 1 KL, or Model 1 CB referring to weighted cross-entropy loss, KL Divergence loss, or CB Focal loss respectively.

2.2. Model 2

Model 2 used the X3D medium architecture (with pretrained weights from the Kinetics400 dataset) for feature extraction and a classification head consisting of 2 1D CNN layers with a ReLU activation in between. The X3D architecture consists of a "stem" (3D convolution, batch-normalization, and Swish activation), four ResNet-inspired stages or "Bottleneck Blocks" (each with 3D convolutions, Swish activations, batch normalization, and optional Squeeze-and-Excitation modules), followed by global average pooling and a 400-node fully-connected layer. We replaced this fully-connected head with our proposed multilayer 1D CNN head described above. The stem and four "Bottleneck Blocks" were frozen and only the weights of the final 1D CNN block were trained.

A weighted Cross-Entropy loss, similar to the one used to train the original SwingNet model, was used to train this model. "Percentage of Correct Events" (PCE) was used as the primary evaluation metric when training the model and tuning hyperparameters. A sample-based tolerance, as described in the original paper, was used for calculating PCE (see Section 3 for more details).

2.3. Model 3

Model 3 used the ResNet R3D 18 architecture (with pretrained weights from the Kinetics400 dataset) for feature extraction and a small 256-node fully-connected layer as the classification head. The ResNet R3D 18 architecture consists of a 'stem' (a 3D convolution, batch-normalization, and ReLU), four 'basic block' layers (each with two convolutions, batch normalization, ReLU, and down-sampling) followed by global average pooling and a 512-node fully-connected layer. The fully-connected layer is followed by our 256-node classification head, which outputs a logit matrix. In testing, softmax is applied to these logits to obtain class probability distributions over time; in training, the log softmax is used for calculating the KL divergence with respect to a 'softened' target distribution.

3. Experiments and Results

Our experimentation was aimed at improving the performance on the GolfDB dataset compared to the SwingNet architecture. Performance is measured using a metric called "Percentage of Correct Events" (PCE). PCE determines the percentage of correctly labeled events within each full length video. An event is labeled correctly if the predicted label falls within a tolerance of the true label. The tolerance δ is defined as:

$$\delta = \max(\left\lfloor \frac{n}{f} \right\rfloor, 1)$$

where n is the number of frames from address to impact, and f is the sampling rate (eg. 30 frames per second). This allows for dynamic amounts of tolerance for each video based on frame count and swing speed.

3.1. Model 1

For Model 1, our experimentation was aimed at determining whether modifying loss functions could improve the

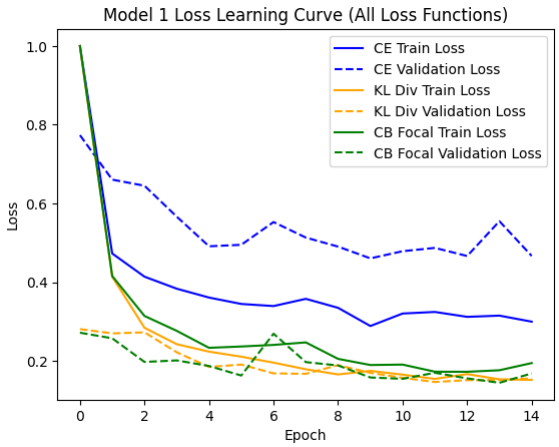

Figure 1. Model 1 Loss Learning Curve by loss function.

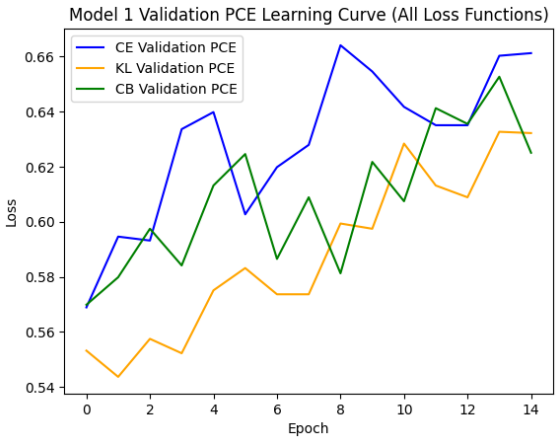

Figure 2. Model 1 PCE learning curve (Validation set only).

performance on the GolfDB dataset using the SwingNet architecture. We trained and tuned each model using a similar methodology to the original paper. To do this we experimented with different backbones (MobileNetV3, MobileNetV2), learning rates (1e-4, 1e-3, 1e-2), and layers of backbone that are left trainable (0, 5, 10). We found that the optimal values within this grid of parameters were the same as the original paper: utilizing MobileNetV2, 1e-3 learning rate, and 10 trainable layers in the backbone. We did not modify the parameters of the LSTM including hidden dimension, number of layers, or dropout to allow for a fair comparison with the original paper's results.

Figure 1 shows the training and validation losses for each different model and loss function combination. The values needed to be standardized, as they were on different scales. Their relative scales show that the specialized loss functions (KL and CB) quickly reduce the loss. CE on the other hand has a much steadier decrease in loss. We can see in Figure 2 that this steadier decrease leads to better validation PCE across the entire training cycle. This is likely because KL and CB Focal can decrease their loss faster by better modeling the underlying data structure and problem definition.

Table 1 shows the relative performance of each loss function on the validation and test dataset compared to the original SwingNet architecture. The results show that Model 1 CB achieved the best test PCE relative to our new models. No model was able to achieve the PCE of the SwingNet architecture. We see that the generalization ability of the two tailored loss functions was superior to that of CE. This highlights that there may be some benefit in more complex loss functions in limiting overfitting.

Table 1 shows the per-class PCE for the model and loss function combinations. Again, the SwingNet architecture does better in all parts of the swing sequence, especially the address and finish. Model 1 CB performs slightly better in address and finish when compared to the other loss functions we tested. It also performs comparably in all other swing sequence classes. This again highlights that targeting hard to classify tasks can lead to increased performance. Surprisingly, Model 1 KL does not perform as well in most of the golf sequence classes when compared to the other loss functions. This may indicate that the probability distribution of classes after the 1D convolution does not provide a useful gradient for identifying the correct frame. More testing and trials would need to be done to identify why this loss function has underperformed, as it seemed the most promising theoretically.

Figure 3 shows an example prediction for each model on the same video. Each event has its probability distribution across all of the frames. We predict each class as the frame with the highest probability for that action. The figure shows that the distributions for each model look similar. The model is more confident in its predictions of the events around impact and less confident for those around address and finish. This aligns with our PCE calculations and our understanding of the data. A golfer spends a long time in poses similar to address and finish but relatively less time in all the other positions. In practice, getting general address and finish positions are enough, as the key positions of the swing are in the middle. Surprisingly, there is a peak for each model predicting mid-backswing after mid-follow through has occurred. This is a strange phenomenon that we suspect is related to the similarity of the mid-backswing and follow-through when you allow for horizontal flips. Horizontal flips are common in golf swing videos due to the presence of both right- and left-handed golfers. We included horizontal flips in our data augmentation (because there are far more right-handed swings than left-handed), but there is still an error in the models ability to tell the two apart.

Model	A	TU	MB	T	MD	I	MFT	F	Train PCE	Val PCE	Test PCE
SwingNet	31.7%	84.2%	88.7%	83.9%	98.1%	98.4%	97.6%	26.5%	N/A	N/A	76.1%
Model 1 CE	18.3%	57.1%	70.0%	68.0%	95.4%	97.1%	94.0%	14.3%	66.6%	66.4%	64.3%
Model 1 KL	17.7%	58.0%	69.1%	58.9%	94.6%	96.3%	95.1%	12.9%	63.7%	63.2%	62.8%
Model 1 CB	19.1%	71.1%	67.1%	62.9%	97.1%	96.0%	95.1%	15.1%	66.6%	65.3%	65.5%

Table 1. Model 1 Per-Class PCE on Test Dataset by Loss Function

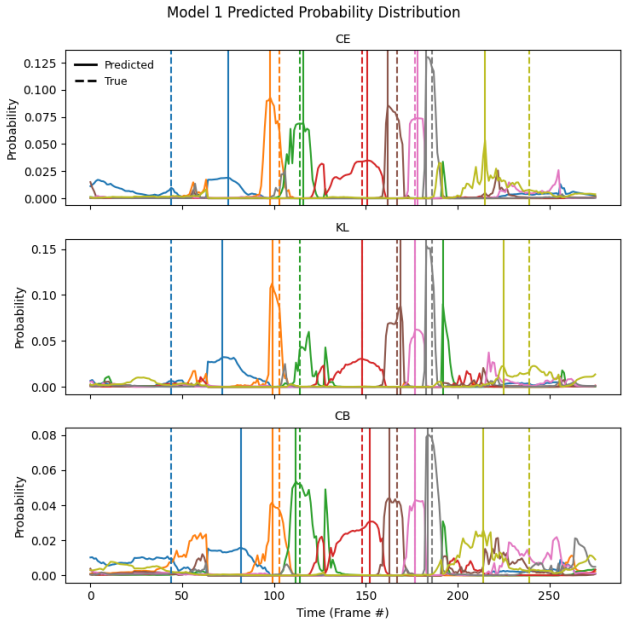

Figure 3. Model 1 example prediction distribution.

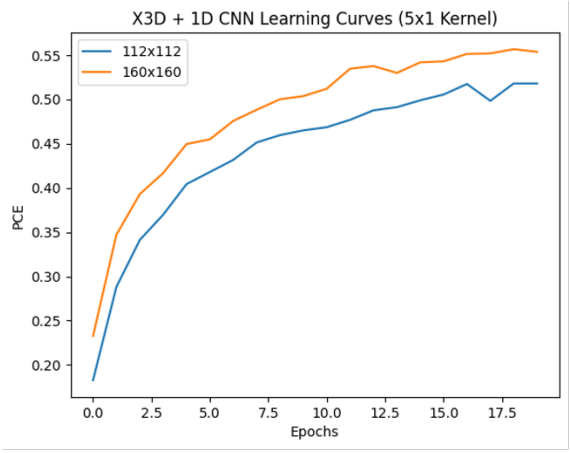

Figure 4. Model 2 PCE Learning Curves by Image Size.

3.2. Model 2

The experimentation of Model 2 involved tuning hyperparameters such as image size, kernel size for the 1D CNN layers, dropout percentage, and learning rates. The Adam optimizer was used for consistency with the other models. While some of the other models were restricted to an image size of 112x112, Model 2 did not suffer from that constraint. Figure 4 shows that, intuitively, leveraging larger dimensions resulted in better model performance. However, unsurprisingly the larger the image size the longer the training process took, with 160x160 images taking about twice as long to train as 112x112 images.

Various kernel sizes for the 1D CNN layers were tested to explore the importance of leveraging temporal context in model performance. While the X3D backbone of Model 2 is capable of extracting spatiotemporal information from the raw input to create downstream features, the kernel size of the 1D CNN head determines how much context is leveraged from those learned features. Nodes created from a kernel of size 1, for example, will only incorporate 1 input feature, whereas kernels of size 3 or 5 will incorporate 3 or 5 input features, respectively. Figure 5 shows that using a kernel of size 3 results in better performance than using a kernel of size 1, and that a kernel of size 5 will further outperform the kernel of size 3. This provides strong evidence for the importance of leveraging temporal context in a golf swing sequencing model.

The importance of temporal context is a key point in explaining the performance of Model 2 compared to the original SwingNet model and the other 2 models we explored. Although utilizing a 5x1 kernel increases the temporal context utilized by the 1D CNN when compared to kernels of smaller size, this still pales in comparison to the abilities of a bidirectional LSTM, as implemented in the baseline model. This largely explains why the results shown in Table 2, are inferior to the results produced by the baseline model and Model 3. Notice also Figure 6, where some incorrect predictions are seemingly way off and do not make sense from a chronological perspective. This provides a clear demonstration of the limitations of the architecture of Model 2.

3.3. Model 3

For Model 3, our experimentation was aimed at fine-tuning the R3D 18 ResNet architecture to obtain a higher PCE than SwingNet on the GolfDB dataset. The soft class-based KL-divergence loss function described previ-

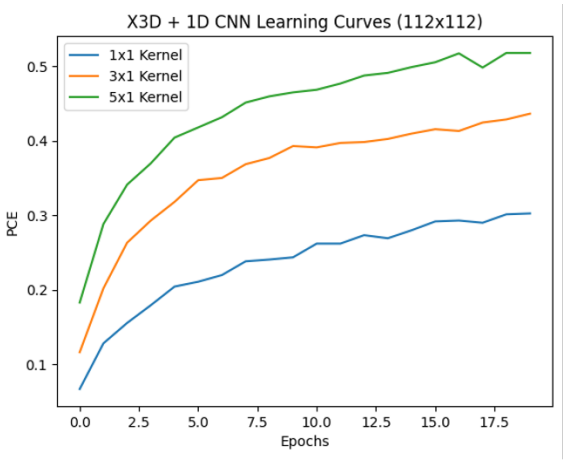

Figure 5. Model 2 PCE Learning Curves by Kernel Size.

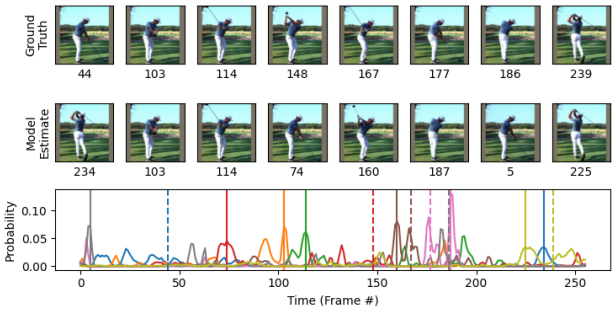

Figure 6. Model 2 example prediction distribution.

Model	Train PCE	Val. PCE	Test PCE
SwingNet	N/A	N/A	76.1%
Model 2	69.4%	68.3%	66.8%
– Address	31.0%	30.0%	24.3%
– Toe-up	78.3%	76.7%	77.4%
– Mid-backswing	69.5%	66.7%	64.3%
– Top	71.3%	70.5%	71.1%
– Mid-downswing	93.4%	91.4%	91.1%
– Impact	95.6%	94.8%	95.7%
– Mid-follow-through	93.0%	93.3%	91.4%
– Finish	22.0%	22.9%	18.9%

Table 2. (Model 2) PCE Performance

ously was used both for training and to determine the epoch at which to stop training (Fig. 8). We tried training with/without pre-trained weights, modifying the number of layers frozen, and several modifications on the soft KL-divergence loss (e.g. various kernels, time-based KL divergence, and various class-weights). For consistency with the other models, we used the same learning rate (1E-3) and optimizer (Adam).

We obtained the best results by freezing the stem and first

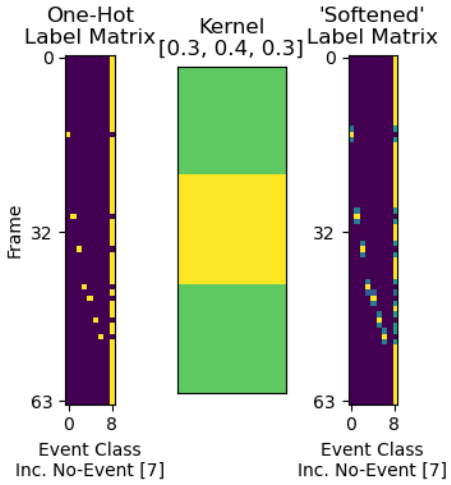

Figure 7. (Soft KL-Divergence) The one-hot label matrix for a single 64-frame sequence (left), the convolution kernel used to relax the exact timing constraint (center), and the softened label matrix used for calculating loss (right).

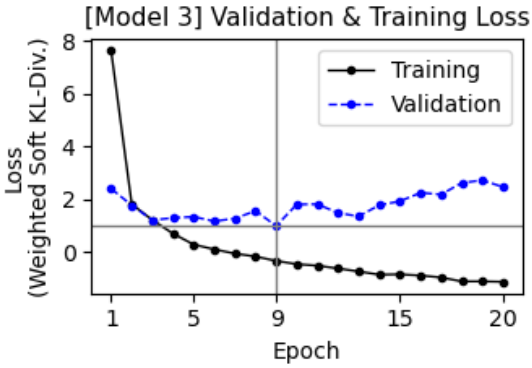

Figure 8. (Model 3) Training & validation loss (weighted soft KL divergence) over 20 epochs. The training loss continues to drop even after epoch 10 even though validation only rises after this epoch. The loss can be negative even though it's generated using KL divergence because of the weighting applied before reduction.

two 'layers' (basic block assemblies), using KL-divergence class weights set inversely proportional to the frequency at which the events appeared after sequence generation, and using a convolution kernel of size (3x1) as shown in Fig. 7 to soften the loss function.

Freezing layers beyond the the stem and first two basic blocks severely reduced performance; we suspect this is because the network has lost some flexibility and is underfitting. Conversely, unfreezing and training earlier layers would require substantially more computing resources than we had available and did not appear to be required insofar as the model was already able to over-fit the data when trained in excess.

With uniform class weighting, the network focuses on the most frequent class (no-event) and tends to predict that

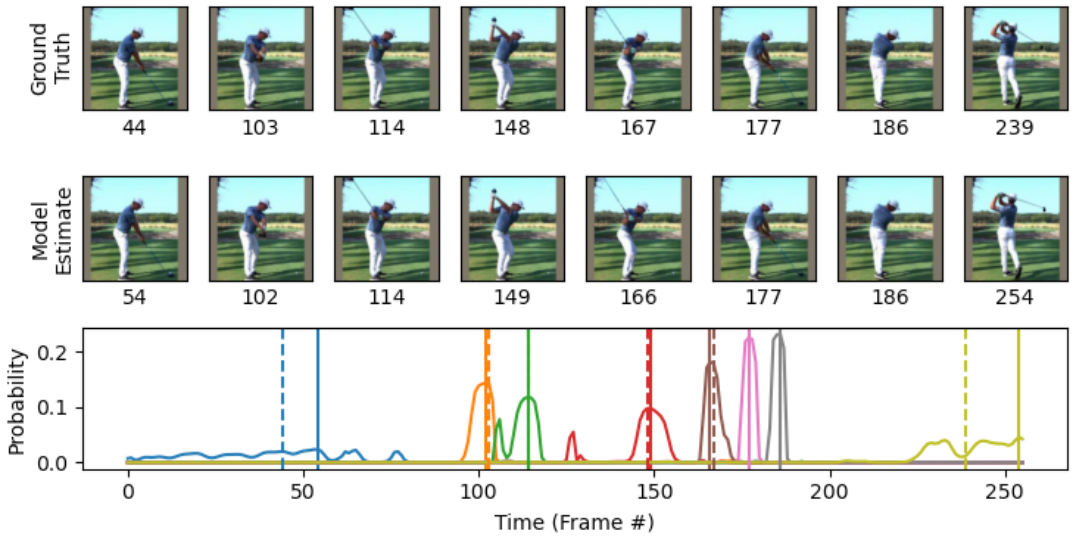

Figure 9. (Model 3) Comparison of ground-truth event frames against the frames at the event-class probability maxima output by Model 3. Although the first (Address) and last (Finish) events are not localized precisely in time, the position of the golfer in the estimated frame is semantically very similar to the ground-truth.

class everywhere. When the weight for no-event is set too low however, the probability distributions (over time) generated at test time do not have to have clear maxima (because there was no penalty for predicting an event so long as it did not conflict with another); consequently finding the peak on distributions with large plateaus is unstable and did not fall within tolerance. Setting the weights inversely proportional to the frequency appears to provide a good balance, generating sharply peaked distributions for most event classes with the peaks at the correct locations.

We explored several kernel sizes, but expanding the kernel size to tolerate error beyond +/-1 frames appeared to generate worse PCE scores in validation, perhaps because (for the real-time videos) the tolerance was only +/-1 frame. We considered 2D kernels (e.g. to encourage event-ordering), but rejected this idea both because the time between events could vary significantly for each golfer and because there was no constraint on the frame at which the event in the sequence occurred, meaning that the following and preceding event may not be in the sequence visible to the network at the time of convolution.

We tried including time-based KL-divergence, wherein the columns of the one-hot label matrix (Fig. 7) were transformed into probability distributions rather than the rows because we believed this would encourage the model to predict class events only at the correct times and so cut down on multi-model event prediction distributions. However, in limited testing the addition of this loss metric did not positively affect the PCE of the model during or after training. We believe this may be because this temporal localization gradient information is already implicit in the use of the no-event class for the class-based soft KL-divergence.

Table 3 shows the total and per-class PCE for Model 3 on the training, validation, and testing sets.

Model	Train PCE	Val. PCE	Test PCE
SwingNet	N/A	N/A	76.1%
Model 3 (KL)	91.0%	80.8%	81.4%
– Address	68.06%	41.27%	43.62%
– Toe-up	98.96%	96.03%	93.42%
– Mid-backswing	96.88%	90.48%	89.71%
– Top	95.83%	80.95%	90.53%
– Mid-downswing	99.65%	100.00%	98.77%
– Impact	100.00%	99.21%	98.77%
– Mid-follow-through	100.00%	99.21%	97.53%
– Finish	68.57%	38.89%	39.09%

Table 3. (Model 3) PCE Performance

4. Conclusions

The main discrepancy in model performance came from 'Address' and 'Finish'; the SwingNet paper acknowledges the "...compounding factors of subjective labeling and inherent difficulty...precisely localizing these events temporally" and "...frequent clubhead waggling" make determining the exact time at which the Address and Finish events occur challenging. Moreover, the mis-estimated and ground-truth frames for these classes look similar, begging the question of whether the accuracy gain the R3D 18 based network offers is worth the cost of nearly 10x ([1], [3], [2]) the number of parameters. Although the R3D-18 based model is the most accurate, the practical ramifications (e.g. memory constraints, run-time, power consumption) may make the other models more appealing.

Student Name	Contributed Aspects	Details
Zach Dawson	Implementation and Analysis	Trained, ran experiments, and analyzed Model 1 (CNN and LSTM). Built MVP data extraction, training, and evaluation loop that was iterated on by team.
Ben Kinard	Implementation and Analysis	Trained and analyzed the X3D model. Identified data leakage and miscalculation of weighted CE loss by authors of GolfDB paper and shared corrections with team.
Andre Green	Implementation and Analysis	Fine-tuned and analyzed the R3D 18 (ResNet) model (Model 3) and generated supporting code for team (e.g. faster data-loaders and code for generating figures).

Table 4. Contributions of team members.

5. Work Division

Please add a section on the delegation of work among team members at the end of the report, in the form of a table and paragraph description. This and references do **NOT** count towards your page limit. An example has been provided in Table 4.

References

[1] B. Chen D. Kalenichenko W. Wang T. Weyand M. Andreetto A. G. Howard, M. Zhu and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. 1, 6

[2] Lorenzo Torresani Jamie Ray Yann LeCun Manohar Paluri Du Tran, Heng Wang. A closer look at spatiotemporal convolutions for action recognition, 2018. 6

[3] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition, 2020. 6

[4] William McNally, Kanav Vats, Tyler Pinto, Chris Dulhanty, John McPhee, and Alexander Wong. Golfdb: A video database for golf swing sequencing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 1

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 2

[6] Karen Simonyan Brian Zhang Chloe Hillier Sudheendra Vijayanarasimhan Fabio Viola Tim Green Trevor Back Paul Natsev Mustafa Suleyman Andrew Zisserman Will Kay, Joao Carreira. The kinetics human action video dataset, 2017. 1

[7] Zijian Wang Wenjing Guo Dandan Zhu Yanting Zhang, Fuyu Tu. Learning golf swing key events from gaussian soft labels using multi-scale temporal mlpformer, 2023. 1