Optimization of Swim Pose Estimation and Recognition with Data Augmentation

Jonathan Ouyang, Derrick Trinh, Chang Choo
AI/DL FPGA/DSP Systems Laboratory
Department of Electrical Engineering
San Jose State University
San Jose, CA 95158
chang.choo@sjsu.edu

***Abstract*—Swim pose estimation and recognition is a challenging problem in Machine Learning (ML) and Artificial Intelligence (AI) as the body of the swimmer is continuously submerged under the water. The objective of this paper is to enhance existing ML models for estimating swim poses and for recognizing strokes to aid swimmers in pursuing a more perfect technique. We developed a novel methodology augmenting raw video data and adjusting a YOLOv7 base model to enhance swim pose estimation. We found the standard multi-class classification Convolutional Neural Network (CNN) to be insufficient for stroke recognition due to the similarity between strokes, so we designed a hierarchical binary classification tree using multiple ensembles of dense neural networks (DNN), CNN, and residual network (ResNet) models. Through these optimizations, the confidence level of pose estimation has increased by over 30%, and the ensembles of our recognition model have achieved approximately 80% accuracy. Fine-tuning of our recognition models and research combining joint keypoint coordinates with angle measurements as inputs could further increase the accuracy of our models.**

I. Introduction

In swimming, every single motion must be optimized to reduce water resistance and drag. Recent advancements in computer vision and AI open the possibility of automating feedback for technique enhancement through pose estimation. However, pose estimation and recognition for swimming is challenging because of occlusions such as bubbles obscuring parts of the swimmer's body and reflections on the surface of the water [1]. The swimmer's horizontal position also causes models to struggle since they are trained on data of upright humans.

One popular approach has been to create a completely new model tailored specifically for swimming [2]. While this was able to achieve high levels of accuracy and detection in swim pose estimation, training a completely new pose estimation model requires countless hours of dataset preparation and training, as well as laboratory-level computational power. Another approach used special environments such as a completely transparent wall in order to get a full view of both underwater and over-water views of the swimmer [3], which isn't practical within real-life scenarios.

Our research utilizes existing models for a more heuristic approach. We improved the accuracy and confidence of a YOLOv7 Pose Estimation base model [4] through automatic rotations of the frame based on the initial prediction of orientation so that the swimmer is now facing upright. We also introduced a classification tree for stroke recognition, where each tree node contained a voting ensemble of multiple DNN, CNN, and ResNet binary classification models. This new architecture yielded significantly higher accuracies compared to a multi-classification CNN model, and the new dataset with rotation transformations resulted in higher accuracies compared to the previous unaugmented dataset.

II. Methodology

We used the YOLOv7 pose estimation algorithm as a starting point, fine-tuning the confidence and Intersection over Union (IoU) thresholds [5]. To increase dataset robustness, we introduced an automatic transformation of video frames in batches and removed problematic facial keypoints from our dataset. From this, we constructed and trained a neural network tree involving 3 different model types and voting ensembles.

A. Original YOLOv7 Base Model

Three popular pose estimation algorithms were selected to undergo preliminary testing: MoveNet model by TensorFlow [6], OpenPose by Carnegie Mellon University [7], and Pose Estimation by YOLOv7 [4].

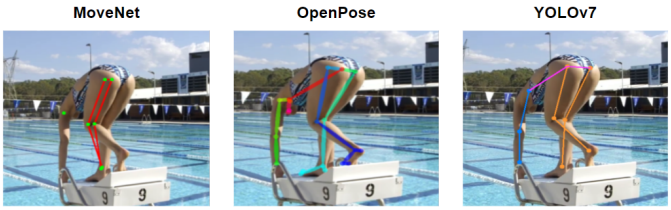

Fig. 1. Pose estimation model performances on dive starts

TensorFlow pose estimation seemed to only work on strictly upright poses outside of the water. For example, when the swimmer is completely bent over in a dive start position (see Fig. 1), TensorFlow only recognized a part of the back thigh and displayed many false positives in random areas in both the pool and the background.

The key difference between OpenPose and YOLOv7 was that YOLOv7 allowed for adjustment of the Intersection over

Union (IoU) and Confidence thresholds [5]. These thresholds essentially control the minimum required confidence for a model to classify the detected object as a positive reading. By lowering both of the thresholds, the model becomes more likely to return a prediction even without a high level of confidence in the joint keypoint, which is needed to create our automatic rotation algorithm.

B. Automatic Transformation Algorithm Introduced

We found that the pose estimation models seemed to be trying to find an upright human to cast a skeleton onto despite the human being clearly sideways (see Fig. 2).

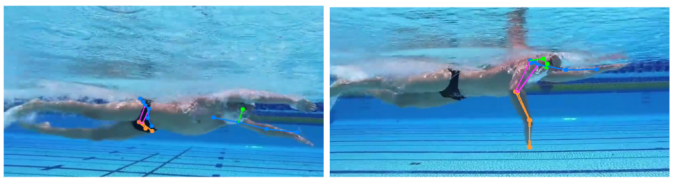

Fig. 2. False Positive pose predictions

We hypothesized that this behavior results from the models being trained primarily on videos of humans standing upright. Despite frequent false positives, we found that the model was capable of guessing the correct joints occasionally. With this in mind, we compared the median x coordinate value of hip and shoulder keypoints over the span of 32 frames to identify which way the swimmer is facing and made necessary rotations to display the swimmer in an upright position.

C. Dataset

1) Structure: From our new rotated video data, An array of 17 joint keypoint coordinates was extracted from every single video frame using YOLOv7 pose estimation (see Fig. 3). The video data was grouped into groups of 32 frames, resulting in a final dataset of (x, 32, 17, 2), where x represents the number of batches of 32 frames (total frames in video divided by 32 & rounded down).

Index	Joint Keypoint		Index
0	Nose	Left Wrist	9
1	Left Eye	Right Wrist	10
2	Right Eye	Left Hip	11
3	Left Ear	Right Hip	12
4	Right Ear	Left Knee	13
5	Left Shoulder	Right Knee	14
6	Right Shoudler	Left Ankle	15
7	Left Elbow	Right Ankle	16
8	Right Elbow		

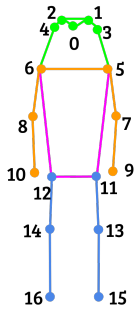

Fig. 3. Joint keypoint list with skeleton visual

Data used in this research are videos featuring underwater side views of a single swimmer, obtained with the consent of the original video's owners. A total of approximately 3400 frames of freestyle, 3500 frames of breaststroke, 4000 frames of butterfly, and 4200 frames of backstroke made up the training dataset and underwent noise injection and data augmentation. This dataset was split during 10-fold cross-validation for model training. A separate dataset of 5000 frames was created for model testing, and excluded from training.

2) Face Keypoints Removed from Data: We found that the pose estimation model was less confident in predicting head keypoints (eyes, nose, ears) than other body keypoints (see Fig. 4). Since the head positioning should not be significant in the context of recognizing swim strokes, head keypoints were removed from our dataset. In the figures below, the y-axis represents the number of coordinate points within each confidence score interval, and the x-axis represents the algorithm's confidence in keypoint.

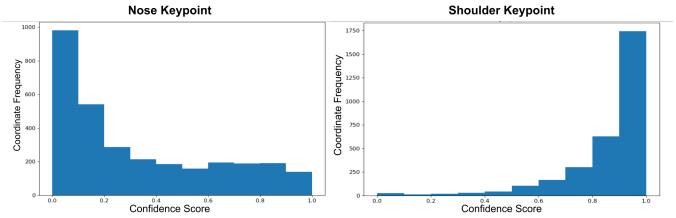

Fig. 4. Graphs of nose keypoint (left), shoulder keypoint (right)

D. Neural Networks for Pose Recognition

1) Initial Architecture: The initial models tested consisted of multi-class classification CNN models. From these base models, we experimented with new DNN, CNN, and ResNet architectures, all using identical 2D dataset input shapes.

2) New Architecture: Due to the overfitting of classes of strokes that were similar in nature (freestyle vs backstroke), we began using a tree of binary classification models (see Fig. 5). The top node consists of a hard voting ensemble of 5 ResNet models which differentiated between the long axis strokes (Free/Back) and short axis strokes (Breaststroke/Fly). The first path consists of a soft voting ensemble of 2 DNNs to differentiate Freestyle vs Backstroke. The second path is a soft voting ensemble of 5 CNNs for Breaststroke vs Butterfly.

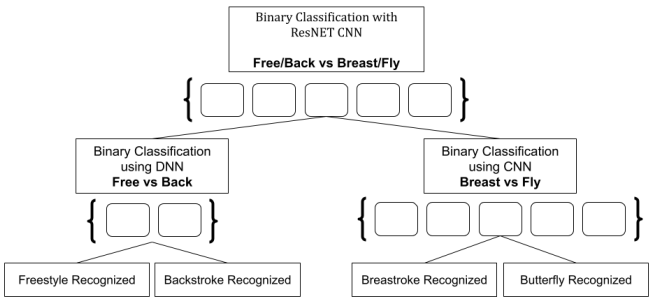

Fig. 5. Binary tree architecture

3) Model Specifics: All models used the same Adam optimizer with a learning rate of 3e-6.

- Simple DNNs contained 2 dense layers of 50 neurons each, both with ReLU activation and no dropout or regularization with categorical cross-entropy loss.
- The CNN layers cycled between Leaky ReLU, batch normalization, and dropout layers. The convolution layers ascended in both the number and size of filters from 60 2x2 filters to 100 5x5 filters.
- The ResNet-101 architecture used 3x3 and 4x4 kernels with added Leaky ReLU, convolution, and batch normalization layers. Dropout layers and L2 regularization were used. Average pooling, flatten, and a final softmax dense layer were included to output predictions.

III. Experimental Results

Our research explores the potential of our methodology to one day match or surpass the performance of the aforementioned swim pose estimation models [2, 3]. Our findings reveal that the automated rotation does positively impact data quality, and our binary models show superior performance compared to the previous 4-category classification CNN.

1) Augmentation Evaluation: During testing, we counted the number of "empty frames" in which the pose estimation algorithm was unable to confidently predict a single joint keypoint. The rotation algorithm showed a decrease in the number of empty frames detected across freestyle, breaststroke, and backstroke (see Table 1). However, there seems to be no improvement in butterfly detection alone.

Datasets	Original % of Empty Frames:	New % of Empty Frames:
Freestyle	7.94%	0.12%
Backstroke	11.98%	2.1%
Butterfly	3.85%	4.50%
Breastroke	7.51%	1.84%
Test Dataset	1.30%	0.35%

Table 1. Original vs New dataset after automatic rotations.

It is possible that the dataset used for butterfly contained video frames that were easier for the model to predict from the very beginning. The butterfly original empty frame percentage is nearly half of the next nearest, being breaststroke (7.51%). Even with the rotation implementation, the percentage of empty frames remained at a reasonably low value, therefore a new dataset was not pursued.

Fig. 6 depicts a histogram of the confidence scores (similar to Fig. 4) before and after the data rotation across all 4 strokes, with the x-axis being the ranges of confidence from 0.0-1.0. The y-axis represents the number of coordinate points within each confidence range, with the maximum represented value being 10,000.

In the original histograms, the number of high confidence keypoints rises near the 0.8-0.9 range but immediately drops in the 0.9-1.0 range. The more negatively skewed the graph is, the more confident YOLO is overall when predicting keypoints. Compared to the older data, the new values contain more keypoints with higher confidence, with an increase of approximately 30%.

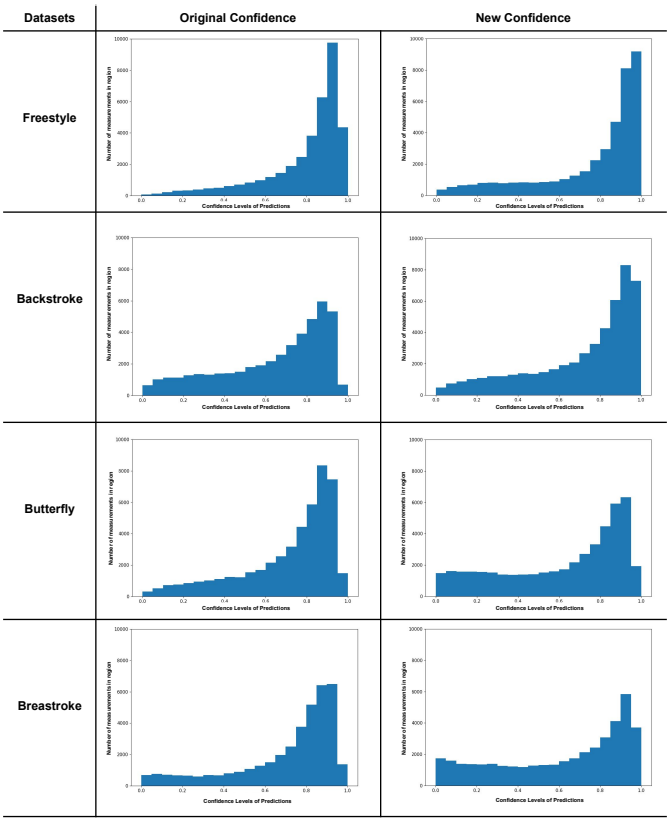

Fig. 6. Original dataset before rotations vs new dataset after automatic rotations

2) Model Improvements: Our original single-network architecture showed problematic unstable training with oscillations in the validation losses and accuracies (see Fig. 7), as well as premature or no convergence.

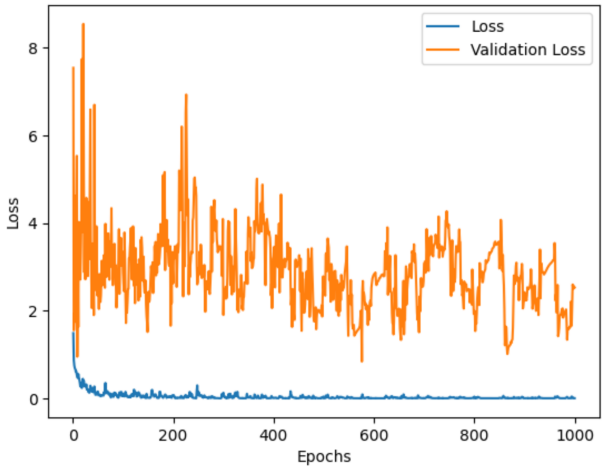

Fig. 7. Model training loss curves

When trained with 5 classes in the dataset (the 4 strokes and dive starts) the model severely overfit to one or two specific classes, to the point where the other remaining classes were never being guessed (see Fig. 8). When we removed the fifth dive class and retrained the model whilst using aggressive regularization and dropout layers, the model was able to somewhat fit the data, returning a much more normal looking confusion matrix. However, some of the classes are still noticeably being overfit (see Fig. 9).

```
[[ 0  0  0 14 37]
 [ 0  0  0 10 36]
 [ 0  0  0 14 39]
 [ 0  0  0 23 22]
 [ 0  0  0 19 10]]
```

Fig. 8. Confusion Matrix 1

```
[[32  1  1 18]
 [ 3 24  7 14]
 [18  6 20  9]
 [ 7 11  3 25]]
```

Fig. 9. Confusion Matrix 2

This over-fitting pattern was common across all CNN, DNN, and ResNet models.

The major issue is that the strokes (mainly freestyle and backstroke) are nearly indifferent when the only input data the model receives is a skeleton-like ”image” from joint keypoint data. We fixed this overfitting issue by switching from multi-class to binary-class classification models to allow our neural networks to focus on the finer details between two similar strokes. We found that DNNs outperformed CNNs for freestyle vs backstroke identification, with over a 30% higher accuracy. On the other hand, the best-performing model detecting breaststroke vs butterfly was a CNN, outperforming the DNN by approximately 25%. Finally, the top performing model for the long axis vs short axis classification model was the ResNet model, outperforming CNNs by 5%. Our decision to switch to a binary tree displayed a significant improvement from a singular model, increasing our accuracies from a peak model performance ranging from around 50-55% accuracy using a single categorical cross-entropy CNN or a single ResNet model with a 40% accuracy. To further increase the accuracies of our models, we created voting ensembles, tested over 100 model combinations with varying biases, and recorded our highest-performing ensembles (see Table 2).

	Solo Model	Hard Voting Ensemble	Soft Voting Ensemble
Long Axis vs Short Axis	76.5%	78.6%	78.2%
Free vs Back Classifier	73%	73%	74%
Fly vs Breast Classifier	72%	70%	81.5%

Table 2. Voting ensemble model comparisons

In theory however, it should have been possible to achieve significantly higher than 78% accuracy on the long axis vs short axis model. Using the angles calculated between the legs of the swimmer, a function was derived to differentiate the two purely based on the angle formed during the kick within the span of 32 frames. With this new function, a new accuracy of 85.9% was achieved.

IV. Conclusion

In this paper, we presented a method of modifying the input video via rotation transformations to increase pose estimation confidence and accuracy, without needing to retrain an entire pose estimation algorithm. By using the YOLOv7 Pose Estimation algorithm as the base model and lowering the Confidence and IoU thresholds, we can create a modified algorithm to first figure the orientation of the displayed swimmer, make a respective rotation, and rerun pose estimation to return a much more accurate skeleton map of joint keypoints. This new data is now suitable for usage in a recognition network built using multiple ensembles of binary classification DNN, CNN, and ResNet networks. Every node has shown significant improvements compared to the original attempted multi-class classification model. With this improved recognition, it is now possible to introduce a feedback algorithm to give advice on technique depending on the recognized stroke.

Future research would benefit significantly from gathering more data than what was used in this project. In addition, training networks using a dataset containing both joint keypoints and angles between joints may yield better results than using either one restrictively.

Overall, this methodology of swim pose recognition shows great promise, and the rotation algorithm even has the potential to be extended to any area in which the input video features humans facing in an “unusual” direction (from upright), without requiring the training and construction of an entirely new pose estimation algorithm.

References

[1] H. Fani, A. Mirlohi, H. Hosseini and R. Herperst, ”Swim Stroke Analytic: Front Crawl Pulling Pose Classification,” in *Proc. 25th IEEE International Conference on Image Processing (ICIP)*, Athens, Greece, 2018, pp. 4068-4072, doi: 10.1109/ICIP.2018.8451756.

[2] N. Giulietti, A. Caputo, P. Chiariotti, P. Castellini, “SwimmerNET: Underwater 2D Swimmer Pose Estimation Exploiting Fully Convolutional Neural Networks,” in *Sensors*. Feb 2023. [Online] https://www.mdpi.com/1424-8220/23/4/2364

[3] M. Einfalt, D. Zecha and R. Lienhart, ”Activity-Conditioned Continuous Human Pose Estimation for Performance Analysis of Athletes Using the Example of Swimming,” in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, NV, USA, 2018, pp. 446-455, doi: 10.1109/WACV.2018.00055.

[4] C. Wang, A. Bochkovskiy, H. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. doi: 10.1109/CVPR52729.2023.00721.

[5] J. Chen, M. Xie, Z. Xing, C. Chen, X. Xu, L. Zhu, G. Li., “Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?” presented at ESEC/FSE, Nov. 12, 2020. doi: 10.1145/3368089.3409691.

[6] Google Research, “MoveNet: Ultra fast and accurate pose detection model. : tensorflow hub,” TensorFlow, https://www.tensorflow.org/hub/tutorials/movenet (accessed Aug. 22, 2023).

[7] Z. Cao, G. Hidalgo, T. Simon, S. -E. Wei and Y. Sheikh, ”OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172-186, 1 Jan. 2021, doi: 10.1109/TPAMI.2019.2929257.