Deep Learning Final Project
Swimming Pose Estimation with Cartoonization

Maya Breslauer, May Hakim

July 2023

Contents

Abstract

The analysis of athletes' performance, particularly swimmers, has traditionally relied on manual observation, which is time-consuming and subject to human error. In recent times, the integration of computer vision applications in sports has gained traction, leveraging the advancements in deep learning and Human Pose Estimation (HPE) techniques. HPE enables the identification of human joints in images and videos, representing their postures as a "skeleton" in space. However, most existing HPE tools have been trained on data from non-aquatic environments, which poses challenges in accurately identifying swimmers' joints in an underwater setting.
This project is based on a paper that proposes a novel approach called "Cartoonization" for object detection, which manipulates images to resemble cartoons using edge masking, greyscale, and color palette reduction. The conspicuousness of objects resulting from cartoonization led us to hypothesize that it could address some of the challenges encountered in underwater HPE.
The main objective of this project is to investigate whether the use of cartoonization can improve HPE of swimmers. This analysis is a crucial part of broader research aimed at utilizing HPE to comprehensively assess swimmers' techniques and provide personalized correction feedback.
To conduct the study, various versions of cartoonization were tested on Yolov8's pretrained model what was further trained on swimming-specific data. The most accurately detected cartoon was used for additional training to a new Yolov8 pretrained model. Eventually, the cartoon-trained model has shown better prediction results for both cartoonized and uncartoonized images [https://github.com/MayHakim/DLswimmingProject.git].

1 Introduction

Coaching athletes and analyzing their performance is an essential aspect of sports training, but it can be time-consuming and require the expertise of a professional coach. Traditionally, coaches either observe athletes in real-time, providing immediate feedback, or film their performances for later analysis. However, these methods are limited to human vision and necessitate the hiring of skilled individuals.
In recent times, the integration of computer vision applications in sports has gained momentum, revolutionizing the way athletes' performances are evaluated and enhanced. Advanced deep learning techniques, particularly Human Pose Estimation (HPE), have proven to be reliable and efficient in identifying human joints in images and videos, enabling motion analysis and performance estimation. HPE algorithms detect people in an image as objects, compound each one in a rectangle called a bounding box and locate their joints as keypoints.
However, when it comes to swimming, applying HPE poses unique challenges. Most existing HPE tools have been primarily trained on data captured in non-aquatic environments, lacking corresponding swimming-specific body positions and filming angles. Moreover, the aquatic environment introduces additional complexities, such as occlusions caused by bubbles, blur, and other water-related factors.
To address these challenges and enhance underwater HPE, we have turned to the innovative technique of "cartoonization". Cartoonization is a process of image manipulation that transforms a regular photograph into a cartoon-like representation using edge masking, grayscale, and color palette reduction. The original article that presented this method tested it for the task of object detection. We would like to examine whether it can be used for HPE as well. Cartoonizing an image makes the object conspicuous. We believe this will not only help detecting the swimmer's figure in the challenging aquatic environment, but also make locating keypoints easier due to the marked body edges.
This project aims to investigate the impact of cartoonization on joint detection for freestyle swimmers using side-view videos filmed underwater.
By employing the novel approach of cartoonization in underwater HPE, this project contributes to a deeper understanding of how computer vision can be utilized to analyze swimmers' techniques and provide valuable correction feedback.

2 Details of the Proposal

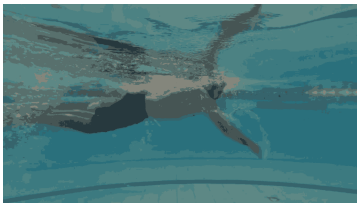
To investigate whether the use of cartoonization can improve swimmers' HPE, we propose this method. First, we explored various combinations of cartoonization techniques, applied each combination on our test set and compared the prediction accuracy of the different combinations. The model we used for this part is Yolov8n pretrained on Coco dataset that underwent additional training with our original swimming dataset. The best-detected cartoon was used to train another Yolov8n pretrained model.

Then, we've compared the performance of both models on the original data and the cartoonized data to assess the impact of cartoonization on the model's ability to accurately predict human poses.
The dataset used for our experiments consists of underwater freestyle swimming videos featuring seven different male swimmers, all filmed from a side angle. The training set comprises four videos with a total of 776 frames, the validation set includes two videos with 515 frames, and the test set consists of one video with 107 frames. All frames were annotated using Coco Annotator.
The cartoonization techniques used for this project are:

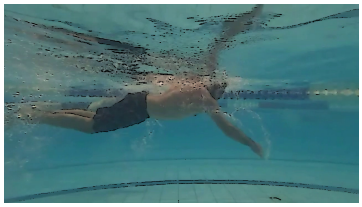
- Blurring - applying bilateral filtering to the input image. Using a non-linear edge-preserving smoothing technique it reduces noise while preserving important edges in the image. It replaces each pixel with a weighted average of the pixels around it with different pixel intensities to preserve edges (figure 1a).
- Color quantization - Color quantization is a technique used to reduce the number of colors in an image while still preserving its overall visual appearance. Color quantization is accomplished using the K-means clustering method. The number of colors in the output picture can be determined using different values of K. In this technique, the colors in the image are grouped into clusters, and each pixel is then assigned to the nearest cluster (figure 1b).
- Edge masking- Edge masking involves creating an edge mask function to identify the edges of an image's objects. It finds the edges in the image using an adaptive threshold and highlights them with a black line. The edges combined with the color-processed image create a cartoon-like effect. In our manipulations, we changed the value of the blur value parameter- edging sensitivity, and the line size (figure 1c).

The original cartoon from the paper was with blur value = 7, total color = 9 and created a cartoon as seen in figure 1d.

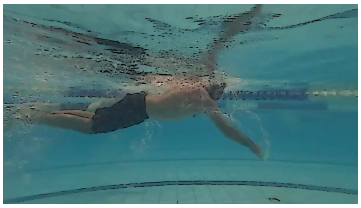
(a) blurring the image



(b) color quantization



(c) edge masking



(d) combination of blurring, color quantization and edge masking

(e) Gaussian blur

(f) sharpen

(g) mean blur

Figure 1: cartoonizations and filters

The paper's code has also presented other manipulations, that are not a part of the cartoonization.

- Gaussian blur - Weighted image averaging using a Gaussian filter based on pixel distances from the kernel center, minimizing the impact of distant pixels (figure 1e).

- Sharpen - Utilizing a 2D-convolution kernel to amplify edges and details. Creating a custom 2D kernel precedes applying convolution via the filter 2D method, making them more pronounced and improving overall image clarity (figure 1f).
- Mean Blur - technique that replace pixels with median of kernel region values in source image. This technique is useful for reducing noise in an image while preserving its edges (figure 1g).

The model we used is Yolov8n- Yolov8 pose nano. Yolov8 pose is Ultralytics' latest version of Yolo algorithm-based pose estimation model (Yolo- You Only Look Once). Nano is the smallest model Yolov8 offers, with only 3.3M parameters.
Yolov8 utilizes three main components for its architecture. First, it adopts the CSP-darknet53 as its backbone, which is a deep neural network architecture known for its efficiency and strong feature extraction capabilities. The CSP-darknet53 backbone helps extract meaningful features from the input data, enabling better understanding of the pose-related information.
To handle features from different levels of abstraction effectively, Yolov8 incorporates PANet (a part of the neck component), which is a feature fusion technique. PANet stands for "Path Aggregation Network" and is designed to combine features from various levels produced by the backbone network. This fusion process enhances the model's ability to capture both fine-grained and coarse-grained information about the pose. It also uses SPPF (Spatial Pyramid Pooling Fusion) which is a module responsible for capturing context information at multiple scales.
Following the feature fusion with PANet, Yolov8 Pose includes the head part, which is the part that responsible for generating the final output. For this purpose, Yolov8 uses the Yolov3 Head. The head divides the input image into a grid of cells. Within each cell, the model's intricate architecture harnesses anchor boxes and regression techniques to predict bounding boxes, accurately outlining the spatial extent of potential objects or pose keypoints, and concurrently estimates associated class probabilities. This grid-based strategy allows the algorithm to systematically explore the image, making predictions in a structured manner, and effectively deciphering complex scenes to identify and locate objects or poses within them.

Our evaluation metrics are:

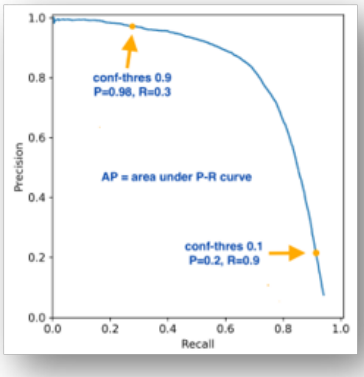
- mean Average Precision (mAP@[.5:.95])- a mean of average precisions for IoU values from 0.5 to 0.95 in steps of 0.05 (IoU- a threshold determining whether a detection is correct). We used the results of mAP(P)- an evaluation of keypoint detection (P=pose).

$$\text{IoU (Intersection over Union)} = \frac{\text{Area of overlap}}{\text{Area of union}}$$

TP = True positive FP = False positive
TN = True negative FN = False negative

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

AP = The area under the Precision - Recall curve
$mAP@[.5:.95]$ = the average AP for IoU from 0.5 to 0.95 with a step size of 0.95



- Percentage of Detected Joints (PDJ)- the percentage of visible keypoints that were detected within a radius of 0.05*bounding box diagonal from the ground truth.

$$\text{PDJ} = \frac{\sum_{i=1}^{n} bool(d_i < 0.05 * diagonal)}{n}$$

d_i−The euclidian distance between ground truth keypoint and predicted keypoint.
bool(condition) - a function that returns 1 if the condition is true, 0 if it is false.
n−the number of visible keypoints on the image

- Object Keypoint Similarity (OKS)- an exponent of the average of distance between predicted keypoints and ground truth keypoints normalized by the scale of the person and the keypoint constant. Keypoint constant should equalize the importance of each keypoint. For instance, neck location needs to be more precise than hip location.

$$\text{OKS} = \frac{\sum_i e^{-\frac{d_i^2}{2s^2k_i^2}} \delta(V_i > 0)}{\sum_i \delta(V_i > 0)}$$

d_i-The euclidean distance between ground truth keypoint and predicted keypoint.
$s-$Object scale.
$k-$Per keypoint constant that controls fall off.
v_i- visibility flags of the ground truth (0-not labeled, 1-not visible, 2-visible and labeled)

3 Results

After comparing several manipulations with the first model, we have concluded that the most accurately detected cartoonization was manipulation 30- adding adaptive edges to the image with line size of 5 (table 1). This manipulation got almost the same results compared with the original data. However, after training a new model on data manipulated with manipulation 30, we have received worse results (table 6). This will be further explained in the conclusions section. We chose the second best manipulation which is manipulation 6- adding adaptive edges to the image with line size of 7. This manipulation performed almost the same as the original data, even slightly worse. To further explore the potential benefit of this manipulation, we applied it on our swimming dataset and used it to train a new model based on the pretrained model of Yolov8n.
Manipulation 6 and 18 include adaptive edges with the same line size added to the original image, 18 has a lower blur threshold which means it is more sensitive to changes and finds more edges. This sensitivity has led to slightly worse results. Manipulation 19 included blur value of 5 (masking threshold) and image blurring, which has led to almost similar results as only using edges with blur value 5 (manipulation 18) so the blur had no additional benefit.

Table 1: Original Model Training - Top 5 By mAP

	mAP@[.5:.95]	PDJ	OKS
manipulation 30: blur value=7, line size=5	0.5761	0.7702	0.6507
original data	0.5730	0.7802	0.6526
manipulation 6: blur value=7	0.5695	0.7517	0.6459
manipulation 19: blur value=5, image blurring	0.5357	0.7504	0.6392
manipulation 18: blur value=5	0.5343	0.7339	0.6267

We also tested the non-cartoon manipulations. Manipulations 24 to 29, 31-32 (table 2) are the non-cartoon filters with and without adaptive edges. We added versions with edges since it was the best cartoon manipulation on the original data. The results we received were almost similar to the original data. Since we wanted to test the effect of the cartoonization itself and the filters didn't improve the results drastically, we've decided to train the new network with manipulation 6 only.

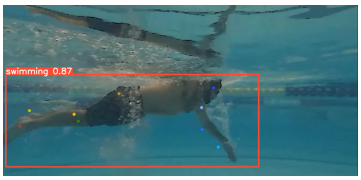
Table 2: Original Model Training - non-cartoon manipulations

	mAP@[.5:.95]	PDJ	OKS
manipulation 25: Gaussian blur, blur value=7	0.5546	0.7455	0.6337
manipulation 24: Gaussian blur	0.5547	0.7438	0.6371
manipulation 26: sharpen	0.5751	0.7679	0.6443
manipulation 31: blur value=7, line size=5, sharpen	0.5767	0.7676	0.6435
manipulation 28: meanBlur, blur value=7	0.5771	0.7848	0.6613
manipulation 27: meanBlur	0.5779	0.7831	0.6543
manipulation 29: sharpen, blur value=7	0.5794	0.7564	0.6394
manipulation 32: blur value=7, line size=5, meanBlur	0.5800	0.7736	0.6549

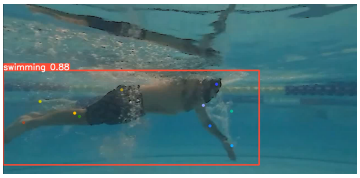
The accuracy of the new cartoon-trained model was tested on all the manipulations. In average, the new model shows better results in all the metrics for all the manipulations (table 5). Looking only at the best mAP results, we can see that the same manipulations made the list with a different order. The best manipulation is now manipulation 6, the manipulation that the model was trained with. We can also see an overall improvement in mAP, a decrease in PDJ and almost no change in the OKS values compared with the first model (table 3). Moreover, from looking at the predicted images by both models, we could see an improvement detecting joints that were closer to the swimmer's body edges, especially in the upper body (like elbows or wrists). This happened for both the manipulated and the original images (figure 2).

Table 3: Cartoonized Model Training (on Manipulation 6) - Top 5 By mAP

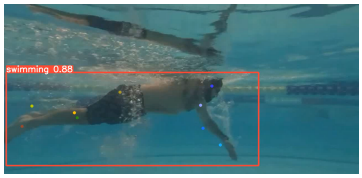
	mAP@[.5:.95]	PDJ	OKS
manipulation 6: blur value=7	0.6082	0.7366	0.6480
manipulation 30: blur value=7, line size=5	0.6065	0.7341	0.6456
original data	0.6049	0.7362	0.6433
manipulation 18: blur value=5	0.5891	0.7185	0.6329
manipulation 19: blur value=5, image blurring	0.5848	0.7051	0.6197



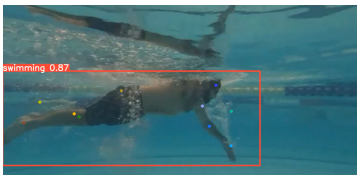
(a) original model-manipulation 6



(b) cartoonized model-manipulation 6



(c) original model-regular image



(d) cartoonized model-regular image

Figure 2: Pose estimation results

4 Conclusions

From the results of the first model we can conclude that the manipulations didn't improve the model's performance. Intuitively, this makes sense since the model was only trained with "regular" images. One of the main challenges in swimming HPE as already mentioned is blurry images and occlusions. Cartoonization doesn't solve this challenge and actually makes the image more blurry and adds edges that occlude more parts of the image. On the other hand, Yolov8's training includes several augmentation. Augmentations are supposed to generalize the model to diverse input by adding copies of the training data with different manipulations. To further explain the fact that the cartoonization didn't improve the results, we've searched for Yolov8's augmentations.
Most augmentations didn't change the actual image, like image rotation, scaling, flipping, etc. However, there are three augmentations that change the image itself: hsv-h, hsv-s, hsv-v. HSV is a color space with three components- Hue, Saturation, and Value. Hue represents the color of the image, so hsv-h augmentation will slightly shift the colors. Saturation determines the intensity of colors, so hsv-s augmentation will increase or decrease color vibrancy. Value affects the brightness of the image, so

hsv-v augmentation will make the image brighter or darker.
Let's consider the manipulations in cartoonization:

- Color Quantization: This technique reduces the number of distinct colors in an image. When applying color quantization to the input images, it introduces a new type of variation that the model has not seen during training. If the model has only seen regular images with higher color fidelity, the altered images with color quantization might be quite different from what the model learned during training. In this case, the augmentations designed for hue, saturation, and value adjustments (HSV augmentations) might not be as effective because they were not trained on color-quantized images.

- Blur: Adding blur to images smooths out details and might affect edges and structures that the model relies on for accurate predictions. The augmentations can still be beneficial to improve the model's ability to handle different poses and viewpoints. However, they may not fully compensate for the blurring effect, as blurriness was already a challenge using the original data.

- Edge masking: The model wasn't trained on anything like this manipulation. However, we were hoping it could help the model since some of the low-level features are related to edge detection.

While some augmentations like rotation, scaling, and flipping can still be useful in enhancing the model's robustness to pose and viewpoint changes, other augmentations like HSV augmentations might not be as effective when the input images undergo significant color quantization, blur, or other forms of preprocessing that create new types of variations.
We believe there could be an explanation for the results of manipulation 6 and 30. As mentioned, the two manipulations add adaptive edges with the same blur threshold and different line sizes. In CNNs, some of the low-level features extracted from images are edges. Perhaps the edges in manipulation 30 were thick enough to help the model with detection yet not too thick to keep the information in the image, they fit the image more naturally. In manipulation 6, we think the lines were too thick and therefore didn't improve the results for the original model, maybe it made the image less informative by blocking too many pixels. This could also explain why the lower edging threshold didn't improve the results- it added more edges to the image and blocked more parts in it.
Despite the good results manipulation 30 got for the original model, the model that was trained with manipulation 30 didn't show good results (table 6). We think that since the edges are so thin, they didn't add much information to the model during training and there was no improvement. Moreover, manipulating the image causes a small reduction of resolution, so it can actually make the model worse.

The second model was trained with images that underwent manipulation 6- edge masking. As mentioned, the average scores were better for the second model. It's important to notice that most of the manipulations included adaptive edges so the improvement is reasonable.
As for the highest mAP results- we can see that the same manipulations made it to the top 5 table. Manipulation 6 had the best results, which makes sense since the model was trained with this manipulation. The results show an improvement in mAP, decrease in PDJ and no change in OKS. mAP averages precision for IoU thresholds from 0.5 to 0.95 and therefore considered rigorous. OKS and PDJ are both metrics that rely on the euclidean distance between the predicted joints and the ground truth. However, OKS takes in consideration the importance of each keypoint in the calculation, which is important for our task of movement analysis. Considering the improvement in mAP and the fact that there was no change in OKS, we can say that the new model performed better for our task.
Moreover, looking at some prediction visualizations we could see that in some cases the second model was more accurate detecting keypoints near the body edges. Especially for the upper body joints- shoulders, elbows and wrists. This happened for both the manipulated and the original data. Perhaps training the model with emphasized edges made it rely more on the edges for keypoint detection and therefore more accurate around edgy areas, like the arm. We recommend exploring this phenomenon in future research.
Another important note is that the model wasn't trained with much data. For example, Coco dataset contains 250,000 images while our training data contains only 776 images. The models could be overfitted to the data and therefore we also recommend testing this method with bigger datasets.

In summary, although there was no benefit in using cartoonized data for HPE models trained with regular images, training models with cartoonized images could improve swimming HPE.

5 Bibliography

References

[1] Abhishek, A. V. S., Kotni, S. (2021). Detectron2 object detection manipulating images using cartoonization. Int. J. Eng. Res. Technol.(IJERT), 10.

[2] Chen, Y., Tian, Y., He, M. (2020). Monocular human pose estimation: A survey of deep learning-based methods. Computer Vision and Image Understanding, 192, 102897.

[3] Dang, Q., Yin, J., Wang, B., Zheng, W. (2019). Deep learning based 2d human pose estimation: A survey. Tsinghua Science and Technology, 24(6), 663-676.

[4] Einfalt, M., Zecha, D., Lienhart, R. (2018, March). Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming. In 2018 IEEE winter conference on applications of computer vision (WACV) (pp. 446-455). IEEE.

[5] Ford, B., Effortless Swimming.

[6] Gong, W., Zhang, X., Gonzàlez, J., Sobral, A., Bouwmans, T., Tu, C., Zahzah, E. H. (2016). Human pose estimation from monocular images: A comprehensive survey. Sensors, 16(12), 1966.

[7] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (pp. 740-755). Springer International Publishing.

[8] Wood, L., Chollet, F. (2022). Efficient Graph-Friendly COCO Metric Computation for Train-Time Model Evaluation. arXiv preprint arXiv:2207.12120.

6 Appendix

Table 4: Original Model's Full Results

	mAP@[.5:.95]	PDJ	OKS
original data	0.5730	0.7802	0.6526
manipulation 1: blur value=5, total color=9, image blurring	0.1694	0.4288	0.4216
manipulation 2: blur value=7, total color=8, image blurring	0.1857	0.3788	0.3771
manipulation 3: blur value=5, total color=8, image blurring	0.1577	0.4044	0.3939
manipulation 4: blur value=7, total color=10, image blurring	0.2394	0.4840	0.4698
manipulation 5: blur value=7, total color=15, image blurring	0.3606	0.5972	0.5291
manipulation 6: blur value=7	0.5695	0.7517	0.6459
manipulation 7: blur value=7, total color=9	0.1603	0.3738	0.3739
manipulation 8: blur value=3, total color=9, image blurring	0.0453	0.2725	0.2797
manipulation 9: total color=9, image blurring	0.2442	0.4506	0.4432
manipulation 10: blur value=7, total color=15,10, image blurring	0.2684	0.4711	0.4470
manipulation 11: blur value=1	0	0	0
manipulation 12: blur value=3	0.2099	0.4377	0.4264
manipulation 13: total color=9	0.1478	0.3725	0.3722
manipulation 14: total color=7	0.1151	0.3009	0.2893
manipulation 17: total color=15, image blurring	0.3469	0.5604	0.5223
manipulation 18: blur value=5	0.5343	0.7339	0.6267
manipulation 19: blur value=5, image blurring	0.5357	0.7504	0.6392
manipulation 20: total color=20	0.4258	0.6397	0.5726
manipulation 21: total color=17	0.3823	0.6070	0.5508
manipulation 22: total color=25, image blurring	0.5001	0.6985	0.6161
manipulation 23: total color=25	0.4713	0.6943	0.6145
manipulation 24: Gaussian blur	0.5547	0.7438	0.6371
manipulation 25: Gaussian blur, blur value=7	0.5546	0.7455	0.6337
manipulation 26: sharpen	0.5751	0.7679	0.6443
manipulation 27: meanBlur	0.5779	0.7831	0.6543
manipulation 28: meanBlur, blur value=7	0.5770	0.7848	0.6613
manipulation 29: sharpen, blur value=7	0.5794	0.7564	0.6394
manipulation 30: blur value=7, line size=5	0.5761	0.7702	0.6507
manipulation 31: blur value=7, line size=5, sharpen	0.5767	0.7676	0.6435
manipulation 32: blur value=7, line size=5, meanBlur	0.5800	0.7736	0.6549

Table 5: Cartoonized Model's Full Results (Trained with Manipulation 6)

	mAP@[.5:.95]	PDJ	OKS
original data	0.60493	0.7362	0.6433
manipulation 1: blur value=5, total color=9, image blurring	0.2866	0.5115	0.4737
manipulation 2: blur value=7, total color=8, image blurring	0.3145	0.4817	0.4411
manipulation 3: blur value=5, total color=8, image blurring	0.2857	0.4809	0.4453
manipulation 4: blur value=7, total color=10, image blurring	0.3729	0.5744	0.5143
manipulation 5: blur value=7, total color=15, image blurring	0.4303	0.6330	0.5638
manipulation 6: blur value=7	0.6082	0.7366	0.6480
manipulation 7: blur value=7, total color=9	0.3082	0.4656	0.4265
manipulation 8: blur value=3, total color=9, image blurring	0.0982	0.3135	0.3124
manipulation 9: total color=9, image blurring	0.3322	0.5230	0.4823
manipulation 10: blur value=7, total color=15,10, image blurring	0.3318	0.5013	0.4557
manipulation 11: blur value=1	0	0	1.02E-50
manipulation 12: blur value=3	0.3648	0.5218	0.4886
manipulation 13: total color=9	0.3228	0.4758	0.4389
manipulation 14: total color=7	0.2440	0.3034	0.2911
manipulation 17: total color=15, image blurring	0.4246	0.6096	0.5495
manipulation 18: blur value=5	0.5891	0.7185	0.6329
manipulation 19: blur value=5, image blurring	0.5848	0.7051	0.6197
manipulation 20: total color=20	0.4697	0.6236	0.5720
manipulation 21: total color=17	0.44973	0.6380	0.5734
manipulation 22: total color=25, image blurring	0.5000	0.6674	0.5991
manipulation 23: total color=25	0.5040	0.6619	0.5987
manipulation 24: Gaussian blur	0.5810	0.7235	0.6314
manipulation 25: Gaussian blur, blur value=7	0.5811	0.7296	0.6356
manipulation 26: sharpen	0.6030	0.7330	0.6361
manipulation 27: meanBlur	0.5948	0.7248	0.6368
manipulation 28: meanBlur, blur value=7	0.6024	0.7326	0.6465
manipulation 29: sharpen, blur value=7	0.6028	0.7339	0.6397
manipulation 30: blur value=7, line size=5	0.6065	0.7341	0.0.6456
manipulation 31: blur value=7, line size=5, sharpen	0.5992	0.7375	0.6408
manipulation 32: blur value=7, line size=5, meanBlur	0.5963	0.7318	0.6406

Table 6: Cartoonized Model's Results (Trained with Manipulation 30) - Top 5 By mAP

	mAP@[.5:.95]	PDJ	OKS
original data	0.5191	0.5023	0.5142
manipulation 30: blur value=7, line size=5	0.5142	0.6266	0.5741
manipulation 6: blur value=7	0.5023	0.6029	0.5718