[[1]](#footnote-2)Visual Object Tracking with YOLO and DeepSORT

Wenda Jiang   
McMaster UniversityHamilton, Canada  
001329642 / [jiangw25@mcmaster.ca](mailto:jiangw25@mcmaster.ca)

Ziqi Dou  
*McMaster University*  
Hamilton, Canada  
400281554 / [douz6@mcmaster.ca](mailto:douz6@mcmaster.ca)

Jianshen Wang  
*McMaster University*  
Hamilton, Canada  
400421080 / [wangj700@mcmaster.ca](mailto:wangj700@mcmaster.ca)

Sai Kantamuneni  
*McMaster University*

Hamilton, Canada  
400421674/ [kantamus@mcmaster.ca](mailto:kantamus@mcmaster.ca)

*Abstract*—Detecting and tracking objects is a crucial part of modern deep learning applications. To have a general understanding of such topic, this project compared two famous deep learning methods for visual object tracking and reached the initial conclusion of SiamMask has better performance than DeepSORT on a single object tracking dataset.

Keywords—Deep Learning, YOLO, DeepSORT, SiamMask, Visual Object Tracking, Visual Tracking Benchmark

# INTRODUCTION

We have long since entered a phase where autonomous cars have become a regular occurrence on the road and companies like Tesla have mastered the auto-pilot feature in cars. To make sure the vehicle drives perfectly, one of its most important features should be Visual Object Tracking. And with the advancement of deep learning techniques in recent years, they have become the undeniable mainstream for object detection and tracking. There are many different approaches available. This project compares the performance of two of the popular deep learning approaches, DeepSORT and SiamMask, on the topic of visual object tracking specifically for cars.

# RELATED WORK

YOLO was first introduced to the public by Joseph Redmon and his team back in 2016 as an object detector that is both fast and reliable [1]. Its revolutionary design utilizes Convolutional Neural Networks but predicts bounding boxes and confidence level directly from full images in one evaluation. Over the years, it has evolved into many different versions, and has been widely used in multiple research fields because of its speed and great performance. DeepSORT has been used frequently with YOLO. In [2], a DeepSORT model with YOLOv4 detector can accurately detect the pears on a tree and continue to track them. In [3], the technique of using YOLOv4 and DeepSORT is proven to be capable of tracking fast moving objects in sports such as footballs or athletes.

On the other hand, SiamMask was proposed in 2019, which is a framework to perform both visual object tracking and video object segmentation. As stated in [4] and [5], it is also fast and well-performing on a high standard, especially having the potential of handling moving objects that can have irregular bounding boxes. However, the fact that it can only track one object limits its usability.

# METHODOLOGY

The goal of the algorithm is to track the same object on the image input from the given dataset. In our case, we used the BlurCar2 dataset, and the training and validation is done based on that dataset.

The algorithms that we are looking for are mostly for object tracking purposes, although some of the algorithms consist of the functions of object detection. Object tracking algorithm can detect and track the same object on the image input, whereas object detection can only detect and track the object type without identifying the object.

## Selection of datasets

In this experiment, we used a part of the car dataset from the Visual Tracking Benchmark [6]. The VTB datasets consist of around 100 sequences of video images that captured day-to-day life objects, such as cars, people, animals and toys.

The dataset used in this project is BlurCar2. This dataset is used for single target tracking so there is information on only one bounding box for each image. Each image is one frame in a continuous video. In addition, this dataset has the characteristics of containing scale variation, motion blur, and fast motion effects on the target. These effects tell the user that the size of the detection bounding box can change, images can be blurry, and the target does not stay at a relatively constant position. These are challenges that the machine learning algorithm must overcome.

For the annotation of the ground truth bounding boxes, a single text file containing xy coordinates as well as the width and height of the boxes has been provided. However, specific details of the xy coordinates cannot be found on the website.

## Selection of algorithms

### DeepSORT

The first object detection algorithm that we picked is DeepSORT [7]. The DeepSORT algorithms was initially designed for tracking multiple objects, we have implemented it in our dataset to track a single car. The algorithm consists of several major components.

#### Estimating tracks

In this part, the algorithm defined the tracking scenario into several features. The features include the center coordinate of the bounding box, aspect ratio, height of the bounding box, and the respective velocities of bounding boxes in the image space. In kth track, the algorithm will count the number of successful associated frames in the previous frames. The count value will reset to 0 if the algorithm identifies the association.

The algorithm also has an age limit Amax, which is the maximum frames allowed when the object leaves the scene. If the age limit Amax is passed after the object left the scene, after the object returns to the scene again, the algorithms will consider that as a new track.

Lastly, if any tracks have not successfully associated to a track measurement within the first three frames it will be deleted.

#### Assignment Problem

In this part, the algorithm will incorporate the motion and appearance information by two metrics. The first metric will calculate the Mahalanobis distance between the predicted Kalman states from the previous frame and new incoming measurements from the new frame. The Mahalanobis distance is calculated by measuring the Stds between the distance of the detection away from the mean track location. The confidence interval is 95% in this part, so it can exclude any associations with high standard deviations.

Although the Mahalanobis distance can calculate the associations between tracks, it performs poorly when the motion uncertainty is low. For example, unstable camera movement can cause a rapid displacement between frames, therefore cripple the meansurement of Mahalanbis distance to track through camera occlusions. In this situation, another metric is introduced. The second metric calculates the smallest cosine distance between the current tracks and detection, with a gallery of tracks in the last 100 associated appearance descriptors for the current tracks. These tracks also employ a distance threshold to see if the new detection is acceptable.

By combining those two metrics and changing the ratio , the algorithm can work in different images conditions. If we increase the , the algorithm will favor the Mahalanobis distance, therefore performing well under short-term predictions. If we decrease , the algorithm will favor the cosine distance, performing well under constant image occlusions when the camera is shaking.

Finally, the new track is admissible only if it is within the threshold of both Mahalanobis and Cosine distance.

#### Matching cascade

In this part, the algorithm uses a matching cascade that solves a series of subproblems. The subproblems include the Kalman filter prediction will increase the uncertainty of an object when it is under constant occlusions for a long period of time. Also, the Mahalanobis distance favors tracks with higher uncertainties when there are several similar tracks.

#### Deep Apperance Descriptor

The function of the deep appearance descriptor is to provide detections to each frame. The DeepSORT algorithm used a CNN trained with large-scale multiple human tracking datasets [8]. The datasets consist of over 1.1 million images of 1261 pedestrians, which is suitable for validating the multi-tracking functions of the DeepSORT algorithm.

The selection deep appearance descriptor does not limit to the CNN that the authors used in their paper. Any other detection algorithm can replace the current CNN in this case. In our experiments, we used an algorithm that replaced CNN with YOLO detection networks.

The overview of the architecture that includes the YOLO network and DeepSORT algorithms will employ YOLO as the deep appearance descriptor, and the YOLO detection result between the input frames will feed to the Kalman filter. From there, the DeepSORT algorithm will provide the function of object tracking.

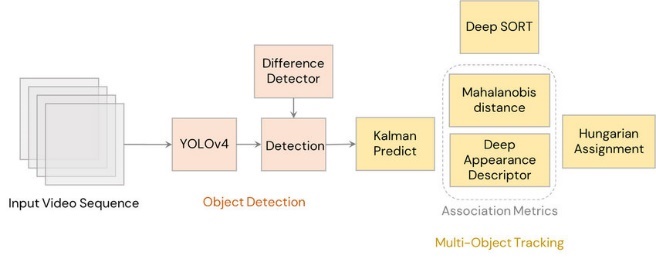


Fig. 1. YOLO + DeepSORT architecture [2]

### YOLO variants

When using the DeepSORT object tracking algorithms, we implemented YOLO networks as its deep appearance descriptor. YOLO networks have lots of variants. From YOLO to YOLOv5, each generation of networks has improved compared to its predecessor. There are changes in networks architecture between generations, but the basic network architectures are like each other. In our experiment, we chose YOLOv5 as the deep appearance descriptor in the DeepSORT algorithm.

The YOLOv5 network architecture is inherited from YOLO and YOLO v4. It consists of three major parts – CSPDarknet for its backbone, PANet for its neck, and Yolo Layer as its head [9].

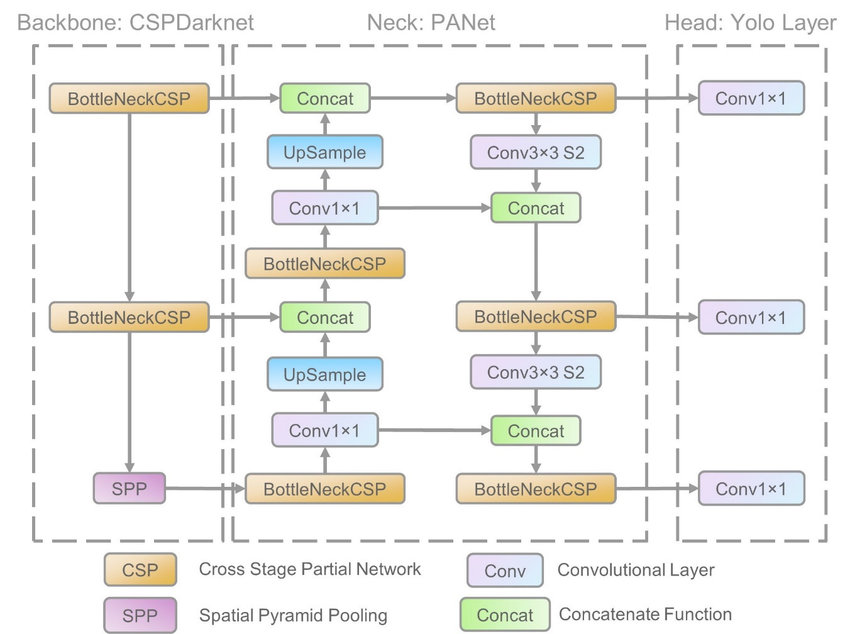


Fig. 2. YOLOv5 network architecture [9]

#### Backbone

The Backbone in YOLOv5 consists of two CSP [10] and a SPP. The CSP is used for resolving the problem of repeated gradient values in the backbone section and adds the gradient changes to the feature map. This approach can reduce the size of the backbone and improve the model speed, by having less parameters. Also, it can improve the model's overall accuracy.

#### Neck

The neck section in YOLOv5 uses the PANet [11]. The PANet uses FPN structure with optimized gradient flow from bottom to top, which will improve the propagation in basic features. The adaptive feature pooling can encourage any useful feature in its feature level to feed to the subnetwork directly. In short, PANet improves the location information flow in low level sub networks, therefore benefits the localization of objects.

#### Head

The head of the YOLO networks has three levels of convolutional network outputs. The levels are 72x72, 36x36, 18x18. The difference in those output sizes makes the YOLOv5 network able to adapt to multi-scale prediction [12], such as small, medium, and large objects. In our experiments, the car dataset has a relatively constant medium object size, so the YOLOv5 can also adapt to the car dataset.

### YOLOv6

YOLOv6 is an object detection framework introduced by Meituan visual intelligence department [13]. Compared to YOLOv5, the YOLOv6 has a significant improvement on FPS vs COCO mAP benchmark.

The YOLOv6 network architecture is similar to YOLOv5. However, one of the major differences is the optimization of the Backbone area in YOLOv6. Unlike YOLOv5 using the CSP block in its PANet structure, YOLOv6 replaced the CSP blocks with RepBlocks. According to the Meituan team, this results in better hardware efficiency and better multi-scale feature fusion.

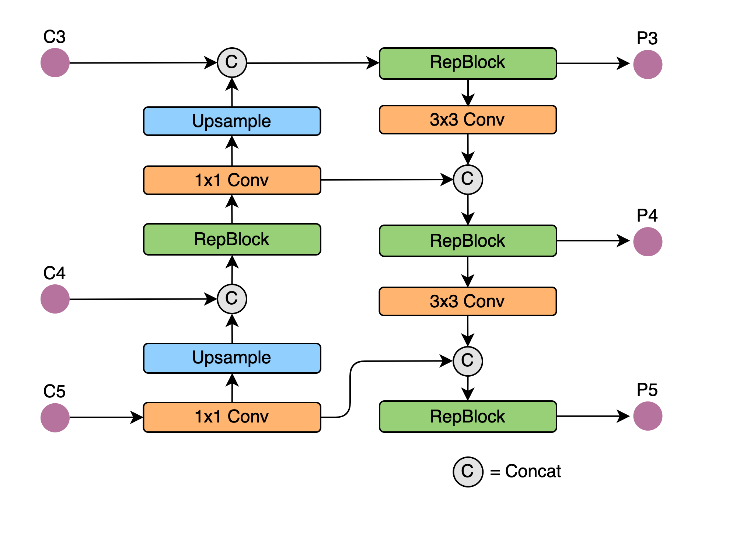


Fig. 3. YOLOv6 Rep-PAN architecture [13]

### SiamMask

SiamMask [4] is a deep learning model architecture which performs both Visual Object Tracking (VOT) and semi-supervised Video Object Segmentation (VOS). Given the location of the object in the first frame of the sequence, the aim of VOT is to estimate an object's position in subsequent frames with the best possible accuracy. Similarly, the main goal of VOS is to output a binary segmentation mask which expresses whether a pixel belongs to the target.

In other words, SiamMask takes as input a single object bounding box for initialization and outputs segmentation mask and object bounding box for each subsequent frame of a video. SiamMask improves over its siamese-network based predecessors by adding a new branch to produce a pixel-wise binary mask. As depicted below, there is a three-branch variant and a two-branch variant.

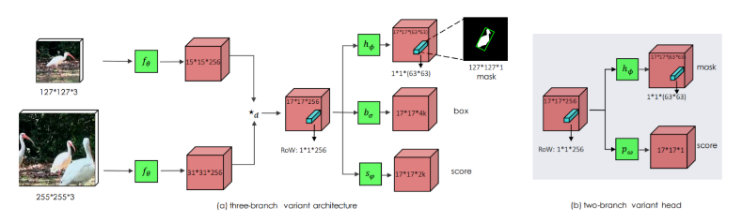


Fig. 4: SiamMask variant architecture [4]

a) Backbone: SiamMask uses ResNet-50 as backbone. The architecture depicted below uses the first 4 stages of ResNet, adjust layer and depth-wise cross-correlation resulting in a feature map of size 17×17.

b) Network heads: The conv5 block in the architecture contains a normalisation layer and ReLU non-linearity activation layer while conv6 only consists of a 1×1 convolutional layer.

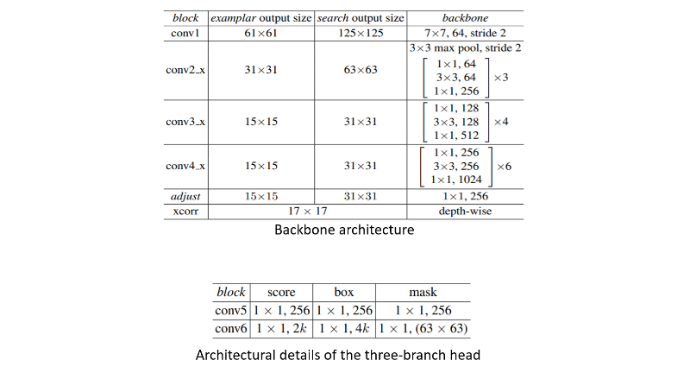


Fig. 5. SiamMask backbone architecture details [4]

c) Refinement: This module merges low- and high-resolution features using multiple refinement steps making use of up sampling layers and skip connections.

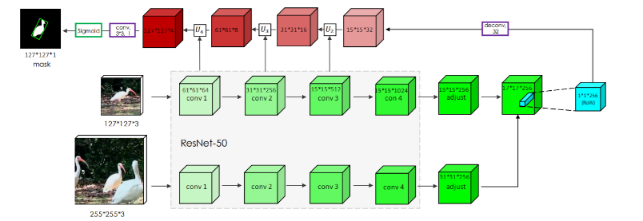


Fig. 6. SiamMask Layers architecture [4]

# EXPERIMENTS

We have experimented with the object tracking algorithm with our dataset. Before we start training the object tracking models, we have set up the required hardware/software environment, as well as preparing augmented training data because it can help reduce the impact of challenging factors such as motion blur and improve model performance as stated in [14] and [15].

## Data augmentation

To achieve high accuracy in the machine learning models, we have performed data augmentation on the original BlurCar2 dataset to create a highly varied dataset of images.

Using the Albumentations library [16] in Python, the following augmentations were done on the dataset:

1. Horizontal Flip (with 40% probability)
2. Vertical Flip (with 40% probability)
3. Rotation (with 40% probability)
4. Brightness Contrast (with 40% probability)

We have implemented the augmentations such that the bounding boxes from the original dataset are retained with the newly generated images.

## Software & Hardware

Most of the models and algorithms that we implemented are running on Python and using the PyTorch machine learning library. We utilized the Cudatoolkit library that comes with PyTorch to accurate our training with GPUs.

For hardware, most of our experiments are done on computers with GTX1080 and RTX2080Ti GPUs.

# RESULTS

In this experiment, we trained the YOLOv5 models first, and used the trained YOLOv5 model in the DeepSORT algorithm as the deep appearance descriptor. After we trained the YOLOv5, we also eperimented with the YOLOv6 algorithm. Although due to the limited time, we have not tested the YOLOv6 model in the DeepSORT algorithm for the deep appearance yet. When we compare the YOLOv6 model with YOLOv5 side by side, we can understand the impact of different models on the deep appearance descriptor.

## YOLOv5

The performance of YOLOv5 model after we trained the model after 80 epochs is listed below. The graph showed the YOLOv5 model achieved 0.99 in precision and 1 in recall. The final mAP0.5:0.95 score is 0.769.

Graphical user interface

Description automatically generated

Fig. 7. Precision of YOLOv5 model trained on the BlurCar2 Dataset

## YOLOv6

We have trained YOLOv6 for 160 epochs. The model reached the mAP0.5:0.95 score at 0.759. During the training of YOLOv6 model, we observed an interesting behavior of the model. At the early stage of the training, the model is not only detecting the labeled car in the center of the screen, but also cars parked on the side of the road. This could be the effect of transfer-learning, as the YOLOv6 model we used in our experiment was pre-trained on the COCO dataset as well as on multi-object detection. Fig. 8 showed a screenshot of validation output of YOLOv6 mode during the early stage of training. Although the model gives the center car a higher confidence level than other detected cars, other cars should not be detected in this case since they are not labelled.



Fig. 8. Early stage of validation output of YOLOv6 model

In the later stage of training, the model can correctly detect the car with true label, without detecting other similar objects.



Fig. 9. Late stage of validation output of YOLOv6 model

## DeepSORT

After we trained the YOLOv5 model, we took the trained weights from YOLOv5 and used it in the DeepSORT algorithm as the deep appearance descriptor. The result of the DeepSORT performance is decent, achieving 0.765 in overall precision and 0.728 in overall recall. Although the author of DeepSORT mentioned about the algorithm combating the rapid object movement by introducing the cosine distance in the calculation, when we validate the performance, the algorithm is still having difficulties in such scenarios. When the camera is shaky and cripples the image quality, the DeepSORT algorithm will output inaccurate object detection.

A picture containing text, sky, road, outdoor

Description automatically generated

Fig. 10. DeepSORT algorithm output

A picture containing text, outdoor, way, road

Description automatically generated

Fig. 11. DeepSORT algorithm output under poor image quality condition

A picture containing text, sky, outdoor, road

Description automatically generated

Fig. 12. Another example of DeepSORT algorithm output under poor image quality condition

## SiamMask

In this algorithm, our experiment environment is Ubuntu Linux system with Python version 3.6. we manually selected the blur white car as our target in the first frame, and the bounding box shows blue color.



Fig. 13. SiamMask algorithm initial frame

In the following frames, the car is tracked by SiamMask and covered with a red mask. In comparison with Fig. 11 and Fig. 12, the SiamnMask gives more clear result on the same frame.

A screenshot of a video game

Description automatically generated with medium confidence

Fig. 14. In comparison with Fig. 11, example frame under poor image quality

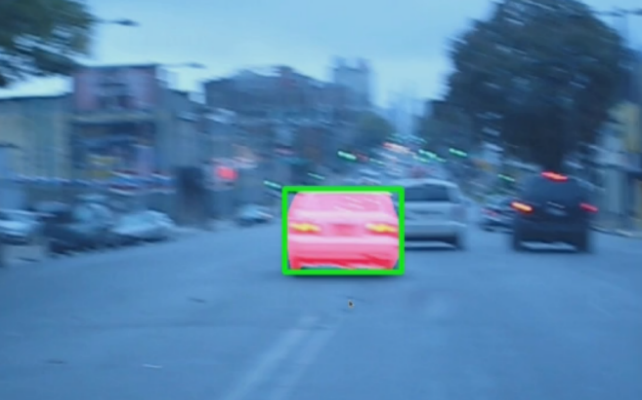


Fig. 15. In comparison with Fig. 12, another example under poor image quality

When comparing Fig. 12 and Fig. 15 with similar frames from the SiamMask output such as Fig. 15 and Fig. 16, there are no similar issues in SiamMask as those in DeepSORT. This might be caused by SiamMask being designed for tracking a single target thus less interferences from other similar objects in the image.

A full tracking video is attached to the file and shown in the presentation.

# CONCLUSIONS

In general, SiamMask achieved better results compared to the YOLO + DeepSORT approach. However, this is only a partial conclusion due to the lack of time and resources. More statistical comparisons need to be conducted in order to fully understand their differences.

For future works, the code structure of the DeepSORT model can be modified to adapt to SOT datasets. In addition, we can add more comparisons for similar deep learning architectures. For example, SiamMask and Siamese. More tracking datasets need be tested as well

##### Acknowledgment

We thank Professor Anwar Mirza for the general help.

##### References

## J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

1. A. I. Parico and T. Ahamed, “Real time pear fruit detection and counting using Yolov4 models and deep sort,” *Sensors*, vol. 21, no. 14, p. 4803, 2021.
2. Y. Zhang, Z. Chen, and B. Wei, “A sport athlete object tracking based on deep sort and Yolo V4 in case of Camera Movement,” *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020.
3. Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, “Fast online object tracking and segmentation: A unifying approach,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
4. B. Chen and J. Tsotsos, “Fast visual object tracking using ellipse fitting for rotated bounding boxes,” *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019.
5. Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
6. N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a Deep Association metric,” *2017 IEEE International Conference on Image Processing (ICIP)*, 2017.
7. L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, “Mars: A video benchmark for large-scale person re-identification,” *Computer Vision – ECCV 2016*, pp. 868–884, 2016.
8. R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, “A forest fire detection system based on Ensemble Learning,” *Forests*, vol. 12, no. 2, p. 217, 2021.
9. C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, “CSPNet: A new backbone that can enhance learning capability of CNN,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.
10. K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, “Panet: Few-shot image semantic segmentation with prototype alignment,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
11. J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv.org*, 08-Apr-2018. [Online]. Available: https://arxiv.org/abs/1804.02767v1. [Accessed: 18-Jul-2022].
12. Y. Chu and H. Kai“Yolov6: The fast and accurate target detection framework is open source,” *美团技术团队*, 23-Jun-2022. [Online]. Available: https://chowdera.com/2022/175/202206241256266607.html. [Accessed: 29-Jul-2022].
13. Z. Cheng, X. Ren, F. Juefei-Xu, W. Xue, Q. Guo, L. Ma, and J. Zhao, “Deepmix: Online auto data augmentation for robust visual object tracking,” *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021.
14. S. Bujimalla, M. Subedar, and O. Tickoo, “Data augmentation to improve robustness of image captioning solutions,” *arXiv.org*, 10-Jun-2021. [Online]. Available: https://arxiv.org/abs/2106.05437. [Accessed: 28-Jul-2022].
15. *Albumentations*. [Online]. Available: https://albumentations.ai/. [Accessed: 16-Jul-2022].

1. [↑](#footnote-ref-2)