YOLOv5 Based Visual Localization For Autonomous Vehicles

Wided SOUIDENE MSEDDI[1,2]
[1]*SERCOM Lab, Ecole Polytechnique de Tunisie*
Carthage University, Tunisia
[2]*L2TI, Institut Galilée*
Université Sorbonne Paris Nord
France

Mohamed Ali SEDRINE[1]
[1]*SERCOM Lab*
Ecole Polytechnique de Tunisie
Carthage University
Tunisia

Rabah ATTIA[1]
[1]*SERCOM Lab*
Ecole Polytechnique de Tunisie
Carthage University
Tunisia

***Abstract*—In this paper, we use the advances brought by neural networks for the implementation of a vision based localization framework for autonomous vehicles namely UAVs. We base our work on monocular visual odometry. It is used for incremental localization of autonomous vehicles. This method suffers from drift. Loop closure detection is a way to improve its accuracy. Thus, we introduce a Siamese network able to perform binary classification in order to detect the visited places and the loop closures. This gives us an accurate, light and fast vision based localization framework.**

***Keywords*— Autonomous vehicle, Visual odometry, Loop closure, Neural networks, YOLOv5, Deep Learning**

I. Introduction

The development and implementation of autonomous vehicles are interesting for a large community of researches as well as industrials. They require a multidisciplinary knowledge and competences to ensure the planning, navigation, communication and sensing tasks. The navigation task could be ensured through a precise knowledge of the state of the robot as well as its environment. The state of the robot is given by its orientation, position, speeds (linear and angular) and accelerations. The environment of the Unmanned Autonomous Vehicle (UAV) is explored using the different sensors embedded within the engine. These could be very different from a robot to another. Namely, the UAV could be equipped with camera, GPS sensor, LiDAR, altimeter, accelerometer, gyrometer or Inertial Measurement Unit (IMU). It seems obvious that the price of the UAV is highly correlated to the embedded sensors. The autonomous vehicles on the market are mainly located using GPS, coupled with other sensors to improve the position estimation accuracy. Although GPS-based position estimation techniques have gained in accuracy this decade, their performance is still linked to the price of sensors and to environment conditions. Thus, accuracy decreases in urban canyons for example, and satellite location data is unavailable in closed / indoor environments. In addition, GPS jammers are used, nowadays, to block GPS signals which could be fatal for the UAV. Besides, global positioning systems and satellites are governments and states property which is a strategic matter. To cop with these limitations, we adopt GPS-independent methods and we use visual information for position estimation. Cameras could be either monocular, stereoscopic or omnidirectional. They could also be equipped with depth sensors (RGB-D). On the other hand, in the computer vision community, there are several vision-based localization strategies. These are Map-building systems, Map-based systems and Mapless systems [1]. Map-building systems, like Simultaneous Localization and Mapping (SLAM) [2], aim to build robot surroundings model and localize the UAV locally, relative to the built map. Map-based approaches need a model of the environment such as point cloud or a computer aided design (CAD). The model has to give absolute position. Finally, Mapless systems ensure the localization of the robot without any reference or model of the environment, like the Visual Odometry [3]. The previously mentioned approaches can suffer from drift. In order to overcome this, some works are based on fusion with other sensors such as Inertial Measurement Unit (IMU). Other methods are based on the loop closure detection which allows to estimate the error and to reduce it. Loop closure detection could be performed using place recognition algorithms. But, luminosity, viewpoint and/or noise are environmental parameters that influence the performance of the visual place recognition procedure. Thus, an efficient solution must be robust to these parameter variations. In order to solve this problem, researchers looked for the appropriate representation of a place. Many state-of-the-art methods use the Bag-of-Words (BoW) [4] paradigm. More recent ones exploit the advances brought by neural networks [5], [6].
In this article, we, first, propose an original localization approach based exclusively on visual information and implemented through a deep learning architecture. Then, we update our existing Vision Based Localization Framework with the new proposed architecture. Thus, in section II we draw up the state of the art. Then, in section III we propose a neural network to detect loop closure. Finally, in section IV we update our whole framework to accurately locate the UAV and we present the performances of our proposed system.

II. Related Works

Visual Odometry is a method that estimates the vehicle pose using video stream input. It measures image features transformation between successive frames. Such technique is

cost-efficient and versatile. It suffers from incremental error due to imaging conditions such as light, blur, textures, etc., then it could mislead the position estimation. Furthermore, camera calibration errors are accumulated at each iteration and the scale could not be estimated, particularly for monocular sensor [7]. All these drawbacks and limitations motivated the research community to design innovative solutions in order to reduce drift and errors. Authors in [8] explain drift propagation and propose some ways to reduce the errors. One way is to use the pose optimization. It consists in using known successive transformations as constraints. Loop closure is a constraint that makes it possible to estimate the drift when a place is revisited. Papers, like [9], introduce different approaches to detect loop closure, and different ways to estimate and to correct the drift. Other approaches use Kalman Filters in order to merge data provided by different sensors like IMU , compass and visual odometry. This refines visual odometry pose estimations and bounds the drift, like in [10] and [11].

State-of-the art loop closure detection methods are based on two main approaches. Appearance-based approaches, based on building visual dictionaries and Bag-of-Words (BoW), like in [4], [12], [13] and [14]. The second family, which is more recent, is based on Convolution Neural Networks (CNN), like in [5], [2], [6] and [15]. In [5], a deep CNN is used (MobileNetV2) to extract incoming frames features. These features are used to build Hierarchical Navigable Small World (HNSW) graphs. HNSW is a proximity graph. Besides, SURF [16] features are extracted for each frame. SURF features are used to gain invariance. A distance between CNN features is then calculated to select Loop Closure Detection candidates. This selection is oriented using HNSW. Then, using SURF features and hash function, these candidates are filtered to generate final loop closure detection. In a previous work [15], we compared different architectures, namely ResNet [17], AlexNet [18] and GoogLeNet [19] and deployed them in two versions each. A first architecture called Siamese [20] and the second one called 6-channel [21] were implemented, compared and tested while performing the loop-closure detection task. Our final results showed that Siamese ResNet has a good accuracy, up to 94%.

In this novel research work, we use a very recent architecture called YOLO (You Only Look Once) in its 5th version. The YOLO [22] framework is a single stage detection approach where a CNN is applied to the whole image. This makes it faster than other approaches. Besides, YOLO is accurate and able to detect many classes (Coco dataset [23]). The YOLO architecture divides the image into a grid. Then, each grid cell predicts bounding boxes and class probabilities. The bounding boxes are defined with their center coordinates, height, width and confidence. The YOLO framework was released in five versions. Original YOLO CNN is made of 24 convolutional layers followed by 2 fully connected layers. Then, YOLOv2 [24] adopted Darknet-19 [25], which is made of 19 convolutional layers and 5 maxpooling layers. YOLOv3 [26] used Darknet-53 as a backbone. Its novelty lies in the use of residual blocks. A residual block contains at least a connection that bypasses two or three layers of a network. It is recognized by the direct connection between the output of a layer and a deeper layer. This leads to a fully convolutional network made of 106 convolutional layers. YOLOv5 [27] is the ultimate model of YOLO architecture. It shares the same architecture as YOLOv4 [28]. Major improvements of YOLOv5 affect data augmentation. In fact, mosaic data augmentation and auto learning bounding boxes were used.

Through this research work, we introduce a framework implementing monocular feature-based visual odometry, combined with YOLOv5 to achieve loop-closure detection.

III. Proposed Approach

A. YOLOV5 architecture

In this work, we use the YOLOv5 architecture, which was designed for real time applications, to recognize already visited places. First, we undertake modifications on YOLOv5 architecture, then, the modified network goes through a training then a validation step. YOLOv5 implementation covers the following parts:

1- YOLOv5 Model Architecture which has three important parts : (i) Model Backbone, (ii) Model Neck and (iii) Model Head. (i) The Model Backbone aims to extract rich features from the input image. A Cross Stage Partial Network (CSPNet) [29] is used as model backbone. In fact, CSPNets are faster than deeper networks. (ii) Model Neck constructs feature pyramids. This component helps the model to detect and identify same objects at different sizes and scales. This leads to better performance with unlearned data. Different types of feature pyramids are used by many models, like FPN [30], BiFPN [31], PANet [32] which is used in YOLOv5. (iii) Model Head performs the final detection. It applies anchor boxes on generated features and outputs final vectors containing class probabilities, objectness scores, and bounding boxes. YOLOv5 has the same model head as YOLOv3 and YOLOv4.

2- Activation Function Activation function is an important component as it determines whether the neuron should be activated or not. In YOLOv5 Leaky ReLU [33] and Sigmoid [34] activation functions are used, which ensures a good compromise between precision and saturation. In fact, the Leaky ReLU activation function is used in middle/hidden layers and the Sigmoid activation function is used in the final detection layer.

3- Cost Function or Loss Function The loss generated by the network after each layer is a combination of objectness score, class probability score and bounding box regression. YOLOv5 uses Binary Cross-Entropy.

4- Optimization Function YOLOv5 implementation can use either Stochastic Gradient Descent (SGD) [35] or Adam [36] optimizers. However, the default optimization function is SGD.

B. Proposed modifications on YOLOv5

Our goal is to use a Siamese model like we did with ResNet [17], AlexNet [18] and GoogleNet [19] in [6]. The original network head is designed to apply anchor boxes and predict class probabilities, bounding boxes and scores.

Whereas, we aim to recognize a place and achieve a binary classification. Thus, we replace the network head by a 3-layers block whereas we preserve the model backbone which output is of shape [512x7x7]. Layers added are: first, an adaptive average pool layer in order to down sample the feature map from [512x7x7] to [512x1x1]; second, a flattening layer to get a 1-d feature map which undergoes a linear transformation; finally a sigmoid activation function to bound the output. We limited the length of the final feature vector to [1000x1] in order to be coherent with the methods we will compare our performance with (ResNet, AlexNet, GoogleNet).

In summary, we propose to evaluate the performances of a Siamese network. This one is made of two branches of modified YOLOv5 backbones, namely two CSPDarknet53 to which we append customized top layers. The two branches output two [1000x1] vectors. An Euclidean distance between these vectors is computed. We train the Siamese network so that it delivers a distance close to zero if the two input images are representing the same place. During training, we use a custom dataset based on Visual Place Recognition Dataset from Bonn University (Bonn subset) [37] augmented by a database of simulated images that we produced. This dataset contains images acquired in changing light conditions which is a great advantage for us to make our overall architecture robust towards such variations[1]. We generated 9000 positive and negative pairs from these images. Besides, each image undergoes a random cropping and Gaussian blur. This helps to enlarge the range of spatial activation statistics. Furthermore, the Gaussian blur is applied in order to simulate motion blur.

IV. Experiments

In this section we highlight the set up of our customized Siamese YOLOv5 network, namely training and validation. The goal is to compare the performance of the two networks in terms of accuracy, speed, size, and the impact on the localization framework. Since the available YOLOv5 implementation is written using Pytorch, a Python package, we used the same framework in this research. We performed our experiments on Google Collaboratory platform, using Intel Xeon 2.20GHz processor, 12 GB RAM and NVidia Tesla K80 GPU.

Training: The model used is not pre-trained, thus training process takes around 95 epochs. The training is made using 80% of our dataset described previously, which means 7200 pairs. We use the same hyper parameters like in [15]. Our batch size is 11 image pairs. The loss function is Contrastive Loss and the optimizer is Stochastic Gradient Descent in order to maintain the same configuration as in [15]. However, the learning rate is $10^{-}1$, unlike Siamese ResNet, GoogLeNet and Alexnet. This could be explained by a faster convergence of YOLO.

Validation: For the validation we use 1800 pairs which is 20% of the dataset. These pairs weren't used for the training step. After 95 epochs, the customized YOLOv5 reached 93% accuracy. Fig.2 shows the improvement of accuracy after each epoch. Besides, the following confusion matrix (Fig.1) shows that the network accuracy is around 0.93, its recall is 0.9488 and its specificity is 0.9053.

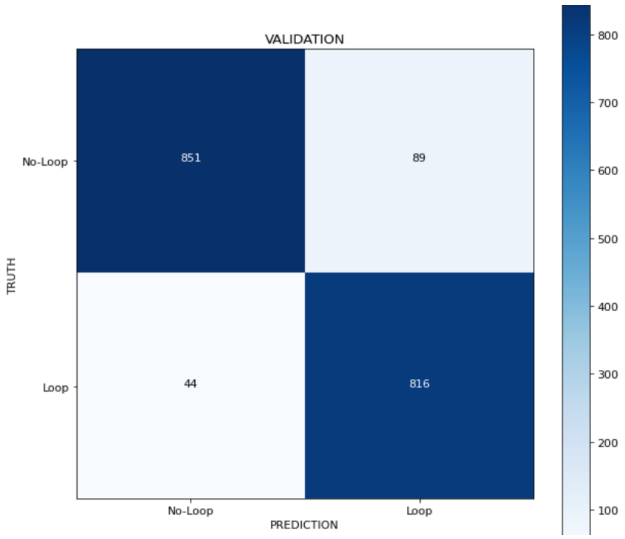

Fig. 1. Trained Siamese YOLOv5 Confusion Matrix

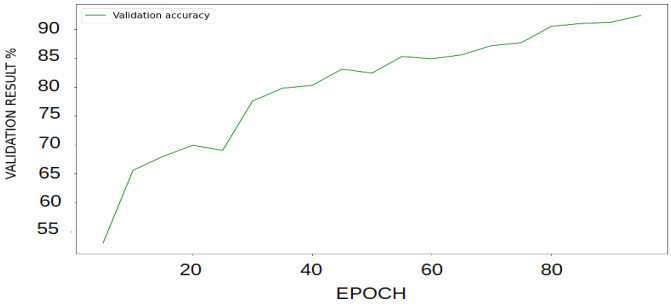

Fig. 2. Network Accuracy Improvement per epoch

In our previous work, we showed that Siamese ResNet outperforms Siamese Alexnet and GoogleNet while recognizing places. Besides, regarding accuracy, Siamese YOLO is the second best network behind Siamese ResNet, using the same dataset: table I. The next subsection will introduce a comparative analysis between these two networks.

TABLE I
Siamese Architectures Accuracy

Network	Accuracy	Recall	Specificity
Siamese Alexnet	93%	91.49%	94.53%
Siamese GoogLeNet	83.61%	88.83%	78.67%
Siamese ResNet	94.83%	96.42%	92.27%
Siamese YOLOv5	93%	94.88%	90.53%

Siamese ResNet against Siamese YOLO: The comparison is based on two main criteria: accuracy (detection performance) and detection rate. Concerning the accuracy, figure

[1]https://www.kaggle.com/mohamedalisedrine/loop-closure-training-dataset

3 shows that the Siamese versions of both ResNet and YOLOv5 have similar performances at distinguishing between places, despite a slightly better precision for Siamese Resnet. In fact, Siamese ResNet's area under curve (auc) is about 0.9859 which is very close to 0.9812, Siamese YOLOv5 auc. Concerning the detection rate, experiments show that

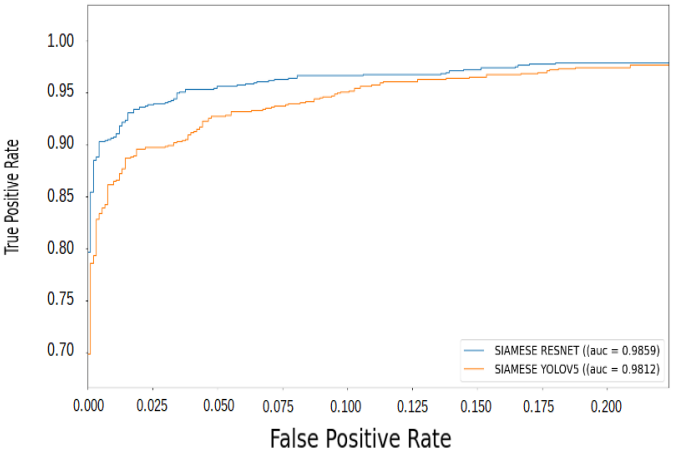

Fig. 3. Siamese ResNet and Siamese YOLOv5 zoomed ROC curves

Siamese YOLO is faster than Siamese ResNet in detecting loop closure when they are executed on CPU or GPU. Figures 4 and 5 prove that the gap between the two networks is more important when using CPU. In fact, Siamese YOLO is 5 times faster than Siamese ResNet on CPU (Intel Xeon 2.20GHz processor), while it is 1.5 times faster on GPU (NVidia Tesla K80). Therefore, Siamese YOLO is more suitable for real time applications, which is our case. Moreover, Siamese YOLO is lighter than Siamese ResNet. Its size is 18MB while Siamese ResNet is 98MB, which makes Siamese YOLOv5 more suitable for embedded applications.

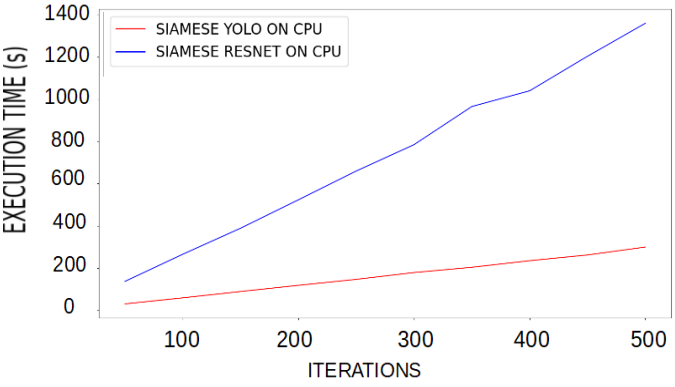

Fig. 4. Siamese ResNet vs Siames YOLOv5 execution time using CPU

As Siamese YOLOv5 performed accurately and quickly in loop closure detection, we decide to replace Siamese ReseNet with Siamese YOLOv5 in our overall vision based localization framework introduced in [6].

Overall Vision Based Localization Framework: We showed above that detection with Siamese YOLOv5 is faster than with

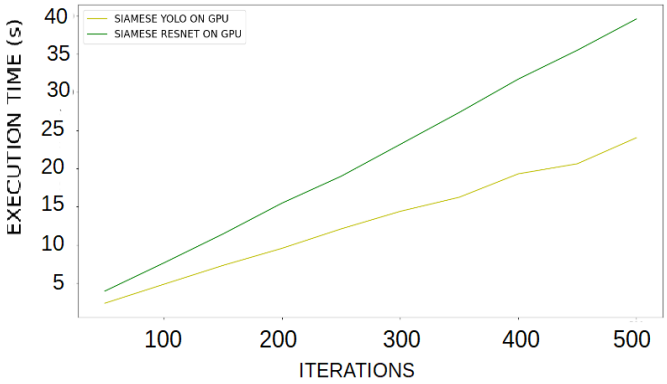

Fig. 5. Siamese ResNet vs Siames YOLOv5 execution time using GPU

Siamese ResNet, thus, we increase the correction rate for loop closure detection. This is to observe the impact of such a modification on the accuracy of the overall system and the ability of Siamese YOLOv5 to be called more frequently. We evaluate the performance of our framework through changing the Correction Spatial Resolution (CSR): the distance travelled by the autonomous vehicle between two consecutive calls of the neural network. Reducing this distance leads to more frequent neural network calls. We also change the Learning Spatial Resolution (LSR): the distance between two stored images during the learning step. Thus, we increase the detection rate four times, i.e we reduce the CSR. Then, we use the framework in three different scenes described in table II. Table III below shows the framework performance based on Siamese ResNet and Siamese YOLOv5. We notice that using YOLOv5 at a higher correction rate leads to better results than using ResNet. Performance increase goes up to 39 centimeters for Scenario 1. This is due to Siamese YOLOv5's ability to quickly classify image pairs.

TABLE II
SCENARIOS SPECIFICATIONS

Scenario	1	2	3
Path shape	Loop	8-like	S-like
Closed Loop	Yes	Yes	No
Length (m)	24	42.5	22
Indoor/Outdoor	Indoor	Indoor	Indoor
UAV speed (m/s)	.5	.5	.5
$\frac{1}{LSR} = a$**(m)**	.125	.125	.125
$\frac{1}{CSR} = b$**(m)**	.5	.5	.5

TABLE III
VISION BASED LOCALIZATION FRAMEWORK PERFORMANCE

Scenario	1	2	3
Siamese ResNet Average Drift (cm)	73.22	27.93	41.98
Siamese YOLOv5 Average (cm)	34.02	22.16	27.71

V. Conclusion

In this research work we develop a stand alone framework for vision based localization of UAVs. We use the very last version of the YOLO network where we modify the head model and we adapt the model neck. We introduce a Siamese network able to perform binary classification in order to detect the visited places and the loop closures for our UAV. The great advantages of our proposed method are, respectively, its high speed compared to state of the art methods as well as its lightness which could encourage an embedded implementation. In a future work, we propose to study trajectory and path planning protocols in order to ensure real time video-surveillance by UAV.

References

[1] Mehmet Serdar Güzel, "Autonomous vehicle navigation using vision and mapless strategies: A survey," *Advances in Mechanical Engineering*, vol. 5, pp. 234747, 2013.

[2] J. Ma, K. Qian, X. Ma, and W. Zhao, "Reliable loop closure detection using 2-channel convolutional neural networks for visual slam," in *37th Chinese Control Conference (CCC)*, 2018, pp. 5347–5352.

[3] Davide Scaramuzza and Friedrich Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Automat. Mag.*, vol. 18, pp. 80–92, 12 2011.

[4] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3921–3926.

[5] S. An, G. Che, F. Zhou, X. Liu, X. Ma, and Y. Chen, "Fast and incremental loop closure detection using proximity graphs," in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 378–385.

[6] Mohamed Ali Sedrine, Wided Mseddi Souidene, and Rabah Attia, "Neural network visual odometry based framework for uav localization in gps denied environment," *COMPUSOFT: An International Journal of Advanced Computer Technology*, vol. 9, no. 8, pp. 3798–3809, 2020.

[7] Dingfu Zhou, Y. Dai, and Hongdong Li, "Reliable scale estimation and correction for monocular visual odometry," in *IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 490–495.

[8] Friedrich Fraundorfer and Davide Scaramuzza, "Visual odometry: Part ii - matching, robustness, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 78–90, 06 2012.

[9] Morteza Daneshmand, Egils Avots, and Gholamreza Anbarjafari, "Proportional error back-propagation (peb): Real-time automatic loop closure correction for maintaining global consistency in 3d reconstruction with minimal computational cost," *Measurement Science Review*, vol. 18, pp. 86–93, 06 2018.

[10] S. Sirtkaya, B. Seymen, and A. A. Alatan, "Loosely coupled kalman filtering for fusion of visual odometry and inertial navigation," in *Proceedings of the 16th International Conference on Information Fusion*, 2013, pp. 219–226.

[11] M. Li and A. I. Mourikis, "Vision-aided inertial navigation for resource-constrained systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1057–1063.

[12] Sameh Megrhi, Marwa JMAL, Wided Mseddi, and Azeddine Beghdadi, "Spatio-temporal action localization and detection for human action recognition in big dataset," *Journal of Visual Communication and Image Representation*, vol. 41, 10 2016.

[13] S. Megrhi, A. Beghdadi, and W. Souidene, "Trajectory feature fusion for human action recognition," in *5th European Workshop on Visual Information Processing (EUVIP)*, 2014, pp. 1–6.

[14] Emilio Garcia-Fidalgo and Alberto Ortiz, "ibow-lcd: An appearance-based loop closure detection approach using incremental bags of binary words," *IEEE Robotics and Automation Letters*, vol. PP, 02 2018.

[15] M. A. Sedrine, W. Souidene Mseddi, T. Abdellatif, and R. Attia, "Loop closure detection for monocular visual odometry: Deep-learning approaches comparison," in *15th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, 2019, pp. 483–490.

[16] H. Bay, A. Ess, T. Tuytelaars, and Luc Van Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, pp. 346?359, 01 2008.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. 2012, vol. 25, pp. 1097–1105, Curran Associates, Inc.

[19] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[20] Jane M. Bromley, Isabelle Guyon, Yann LeCun, Eduard Sackinger, and Roopak Shah, "Signature verification using a siamese time delay neural network," in *7th Annual Neural Information Processing Systems Conference*. 1994, pp. 737–744, Morgan Kaufmann Publishers, Advances in Neural Information Processing Systems 6 Edited by Jack D. Cowan, Gerald Tasauro, Joshua Alspector.

[21] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4353–4361.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision - ECCV*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Eds., Cham, 2014, pp. 740–755, Springer International Publishing.

[24] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[25] Joseph Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[26] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[27] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[28] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode, Christopher-STAN, Liu Changyu, Laughing, Tkianai, YxNONG, Adam Hogan, Lorenzomammana, AlexWang, Ayush Chaurasia, Laurentiu Diaconu, Marc, Wanghaoyang, Doug, Durgesh, Francisco Ingham, Frederik, Guilhen, Adrien Colmagro, Hu Ye, Jacobsolawetz, Jake Poznanski, Jiacong Fang, Junghoon Kim, Khiem Doan, and Lijun Yu, "ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration," Jan. 2021.

[29] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 390–391.

[30] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[31] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10778–10787.

[32] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[33] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li, "Empirical evaluation of rectified activations in convolutional network," 2015.

[34] James Bergstra and Y. Bengio, "Quadratic features and deep architectures for chunking.," 01 2009, pp. 245–248.

[35] Sebastian Ruder, "An overview of gradient descent optimization algorithms.," 2016.

[36] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[37] O. Vysotska and C. Stachniss, "Effective visual place recognition using multi-sequence maps," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1730–1736, 2019.