A Survey on the State-of-the-Art Instance Segmentation Techniques

Chantelle Saliba
Faculty of Artificial Intelligence
University of Malta
chantelle.saliba.17@um.edu.mt

***Abstract*—In recent years, object detection has been at the centre of various advancements with the introduction of semantic and instance segmentation. In this research study, a survey on the current state-of-the-art instance segmentation architectures is being conducted through an evaluation of the established MsCoco dataset and the Tiny Pascal VOC dataset. Architectures being considered consist of the Mask R-CNN which is to this date referred to by many as the current state-of-the-art, the Yolact and the Yolact++ which are famous for their real-time image inference and the newly implemented TensorMask and CenterMask. Throughout this analysis, it was concluded that the CenterMask achieved the best scores in both considered criteria: AP score and inference time, making it the current state-of-the-art instance segmentation technique.**

***Index Terms*—instance segmentation, object detection, mask r-cnn, yolact, yolact++, tensormask, centermask**

I. Introduction

Artificial Intelligence has seen numerous advancements in recent years in the area of computer vision in which one can analyse the works of famous researchers such as Zisserman, He and LeCun in the fields of image recognition [1], [2], optical character recognition [3] and object detection [4].

Object detection [5], [6] is one such field that has seen rapid progress in the last couple of years, increasing in its popularity amongst researchers. Object detection is the process of locating objects within an image and providing their classifications. Therefore, object detection can be said to consist of two complementary tasks: the first object localization and the second object classification. Object localization constitutes of an algorithm that determines the location of an object within an image generally through the use of bounding boxes or centroids while object classification identifies the class that the object belongs to.

In addition to bounding boxes, object detection nowadays can utilize semantic or instance segmentation [7] to determine the location of an object within an image with utmost precision through the use of masks. Semantic Segmentation [8] can be defined as image classification at a pixel level. This type of segmentation strives to output a class label for every pixel within an image. Differently, instance segmentation [7] is an evolved image segmentation technique that provides a different label for separate instances of objects belonging to the same class as shown in Figure 1 [9]. Object detection is a commonly used computer vision technique and one finds such technique in human behaviour analysis [10], facial recognition [11] and autonomous driving [12].

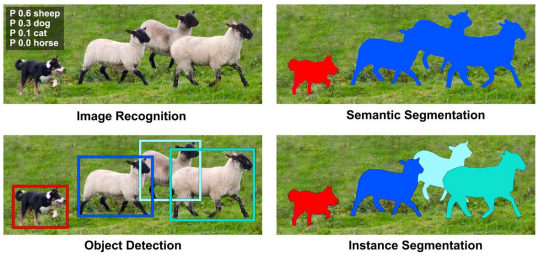


Fig. 1: The difference between Instance Segmentation and Semantic Segmentation [9]

This research paper focuses on the evolution of instance segmentation with special regards to the techniques that helped shape the current state-of-the-art instance segmentation technique. The first part of the paper highlights an in-depth analysis and understanding of the architectures represented by the researchers in this field whilst the second part of the paper experiments with the most notable architectures performing an evaluation on the results achieved. This paper is then concluded with a summary of the main points and findings of this paper.

II. Literature Review

The inception of instance segmentation is gratified to Hariharan *et al.* [13] with their proposal of generating and classifying mask proposals of a given image. Since that day, deep learning has become increasingly popular, resulting in the origin of several instance segmentation frameworks, some of which will be discussed here-under.

A. Mask Regional Convolutional Neural Network (Mask R-CNN)

The Mask R-CNN architecture proposed by He *et al.* [4] in 2017 remains to this date referred to by many as the current state-of-the-art instance segmentation technique. This architecture is based on the Faster R-CNN architecture proposed by Ren *et al.* [14] which in turn is based on the Fast R-CNN architecture implemented by the researchers Girshick *et al.* [15] which is based on their previous work [16] on the RCNN architecture.

To gather an understanding of this architecture, therefore, the RCNN architecture will be first discussed. This architecture proposed by Girshick *et al.* [16] was one of the first architectures that analysed the use of convolution neural networks (CNN) for object detection [7] after the success of the CNN for image classification. This architecture utilises selective search to extract 2000 regions from an image. Selective search makes use of varying scales, colours, textures and enclosure as indicative tools to determine regions within an image that contain objects referred to as regions of interest. These regions are then resized and inputted into a pre-trained CNN to perform feature extraction. This pre-trained CNN, also referred to as the backbone of the architecture, makes use of a chosen renowned CNN model. In this architecture, the researchers opted to make use of the AlexNet model as the backbone. The regions extracted from this CNN are then inputted to a Support Vector Machine (SVM) to perform the classification of these regions to their respective classes. Finally, a bounding box regression is used to determine the bounding boxes coordinates for each identified region.

This technique, although showcasing high quality is relatively slow due to the number of models involved. Furthermore, performing feature extraction on every image region is computationally expensive. To address these issues, Girshick *et al.* [15] introduced the Fast R-CNN with the main aim of tackling the challenges presented in the previous model. Differently, this architecture receives an image as input and passes it directly to the pre-trained CNN with a VGG16 architecture which in turn generates the regions of interest. A Region Of Interest (ROI) pooling layer is then introduced and applied on all of these generated regions reshaping them like the input of the CNN. Each region is then inputted to a fully connected network. On top of the fully connected network, a softmax layer is used to output the classes and in parallel, a linear regression layer is used to output the coordinates of the bounding box for the given predicted classes.

This architecture addresses the previously mentioned challenges of the R-CNN, however, this architecture still utilises a selective search to locate the regions of interest which is time expensive. This challenge was addressed by Ren *et al.* [14] with the implementation of the Faster R-CNN. The main difference between its precedent is that instead of utilizing a selective search, this architecture utilises a Region Proposal Network (RPN). Therefore, this model accepts an image as input and passes this image to a CNN with a VGG16 architecture to generate the feature map of the image. This map is inputted to the RPN which returns the object proposals along with a score. This is achieved by making use of a sliding window over the feature map generated by the CNN and at each window generate k number of anchor boxes of different shapes and sizes. For each anchor, the RPN predicts the probability that the anchor contains an object and a bounding box regressor adjusts the anchors to better fit the object. Similar to the previous model, an RoI pooling layer is then applied to these proposals to reshape them and passes them to the fully connected layer which has a softmax layer that outputs the classes and a linear regression layer that outputs the coordinates of the bounding boxes.

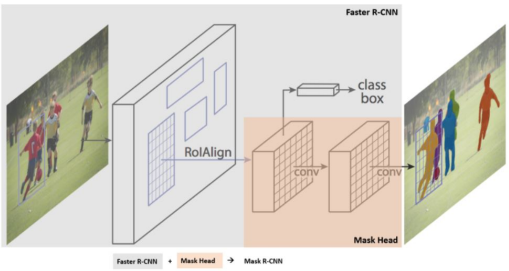


Fig. 2: Mask R-CNN Architecture [4]

Although this model performed the best out of the RCNN family, this model is still prone to some challenges due to the fact the architecture is structured in such a way that it does not look at the complete image but rather focuses on parts of the image sequentially. Therefore, the model is required to perform multiple passes through a single image to extract all the objects. Nonetheless, this architecture inspired He *et al.* [4] with the implementation of the Mask R-CNN architecture. This architecture extends on the Faster R-CNN as shown in Figure 2 by introducing an object mask prediction on each Region of Interest (RoI) that works in parallel to the object bounding box recognition, therefore adding a third output to the Faster R-CNN architecture outputting an object's mask. This branch is essentially a small Fully Connected Network (FCN) predicting a segmentation mask for every ROI in a pixel-to-pixel manner. To achieve this, ROIAlign was implemented instead of the ROI pooling technique used in the Faster RCNN to preserve the exact spatial location allowing for better and more precise mapping of the regions. Similar to its precedent, the researchers made use of the RPN as the backbone of the architecture but instead of utilising the VGG16 with the RPN, the ResNet101 model was used.

B. Yolact (You Only Look At CoefficienTs)

The Yolact was introduced by Bolya *et al.* [17] and it consists of a simple and fast instance segmentation model with a fully convolutional topology. This model breaks the main segmentation problem into two parallel sub-tasks. The first task involves generating a set of prototype masks and the second task consists of predicting per-instance mask coefficients. Instance masks were generated by linearly combining the prototype masks to the mask coefficients.

As shown in Figure 3, this architecture is based on the RetinaNet [18] by using the ResNet-101 model and a Feature Pyramid Network [19]. A Feature Pyramid Network (FPN) is a network architecture that contains a bottom-up pathway, a top-down pathway, and lateral connections that works independently of the backbone that output proportionally sized feature maps at multiple levels, in a fully convolutional fashion. The architecture of the Yolact *et al* [17] then separates into two branches that work in parallel. The first branch as discussed

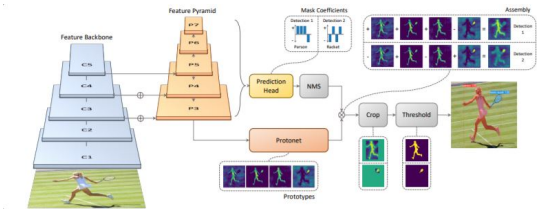


Fig. 3: Yolact Architecture [17]

before produces a set of image-sized prototype masks by utilising a fully convolution network (FCN). In parallel, the second branch generates mask coefficients for each anchor by adding an extra prediction head to the FPN. Those instances that pass through the Non-Maximum Suppression (NMS) successfully are then linearly combined with the output of the first branch constructing a mask for that instance.

This architecture was improved upon by the same researchers with the implementation of the Yolact++ [20]. The Yolact++ is based on its precedent the Yolact with improvements targeted to increase the model's performance while retaining its real-time applications. The first improvement consisted of a fast mask re-scoring network that performs re-ranking of mask predictions according to their mask quality based on the mask IoU with the ground truth. This was inspired by the work of Huang *et al.* [21] with their research on the Mask Scoring R-CNN architecture. The second alteration to the Yolact architecture involves the use of deformable convolutions [22] within the backbone network to generate more precise mask prototypes. Deformable convolutions make use of free-form sampling in replacement to grid sampling allowing for feature sampling to align better with instances. Finally, the researchers also experimented with detection anchors to create a more optimized prediction head opting to make use of multi-scale anchors per FPN level.

Both architectures can perform real-time instance segmentation due to their parallel structure and lightweight assembly.

C. TensorMask

Chen *et al.* [23] present fairly new research in the under-explored area of dense sliding-window for instance segmentation through their TensorMask architecture. The sliding-window technique which this work is based on can be described as locating objects by analysing each window placed over a dense set of image locations. Their work highlights how popular instance segmentation techniques inspired by the Mask R-CNN generally perform object bounding box detection followed by segmentation to generate masks, therefore through this work, the researchers propose a technique that utilizes structured high-dimensional tensors to represent image content in a set of densely sliding windows.

The ideology behind this architecture follows this technique to analyse instance segmentation as a prediction task over 4D structured tensors that captures its geometry and enables novel operators. These structured tensors used to represent masks over a spatial domain are being described by the following notation *(V, U, H, W)* where *(H, W)* show-cast the object position and *(V, U)* shows the relative mask position representing windows sliding over *(H, W)* which are the height and width of a feature map. Chen *et al.* then implemented a pyramid structure which is being referred to a *tensor bipyramid* over a scale-indexed list of the before-mentioned 4D tensors. The tensor bypyramid consists of a pyramidal shape in both *(V, U)* and *(H, W)* which grows in opposite direction capturing the fact that large objects have high-resolution masks with coarse spatial localisation while contrarily small objects have low-resolution masks with fine spatial localization. The TensorMask architecture consists of a mask prediction head to generate masks in sliding windows and a classification head in the sliding window to perform object detection. To perform the predictions, the architecture utilises an underlying FPN backbone with Resnet-50 similarly to the training process proposed in RetinaNet [18] to generate a pyramid of feature maps.

This research hopes to inspire other researchers to experiment with dense mask prediction for instance segmentation and act as a foundation for such research.

D. CenterMask

CenterMask is an architecture introduced by Lee and Park [24] in 2020 that is inspired by the high accuracies achieved by the Mask R-CNN [4] and the high-speed performance of the Yolact [17] with the aim of creating a system that show-casts improvements to both speed and accuracy.

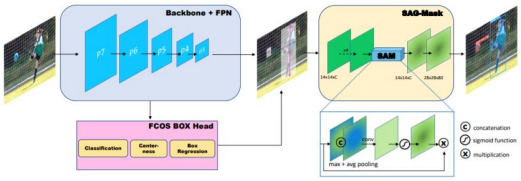


Fig. 4: CenterMask Architecture [24]

To achieve this aim, the researchers proposed an anchor-free one stage instance segmentation technique presented in Figure 4. The CenterMask architecture consists of three main parts: the backbone, the fully convolutional one-stage object (FCOS) [25] detection head and the mask head. The backbone of the architecture is used for feature extraction and for its implementation the researchers experimented with different networks such as the ResNet, the DenseNet and eventually the VoVNetv2 which showed better performance and faster speeds than the before-mentioned networks along with an FPN. The VoVNetV2 network is an improvement on the VovNet [26] model with the addition of a residual connection and a Squeeze and Excitation (eSE) attention module which learns a channel's specific descriptor and rescales the input feature map to highlight only the useful channels.

The second part of the architecture consists of the FCOS detection head. The FCOS [25] is an anchor-free and proposal-free object detector that predicts a 4D vector in addition

to the class label at each spatial location on a level of feature maps. Given its anchor-less properties, this network avoids complex computations hence reducing computational costs thus increasing its efficiency. In addition, this network also proved to outperform anchor box based object detectors making it an ideal network for the CenterMask.

The objects detected are then used by the mask head to predict segmentation masks. To achieve this, the researchers designed the SAG-Mask head which is a spatial attention module that predicts a segmentation mask for each Region of Interest (RoI). The SAG-Mask consists of extracting features using ROI Align and using these features as input to four convolutional layers and spatial attention module (SAM) sequentially, followed by a convolutional layer and a sigmoid function for normalisation.

The research presented claims to be the first anchor-free one-stage instance segmentation technique proposed and aims to pave the way for real-time instance segmentation by offering its architecture as a baseline. In addition through this research, the researchers aim to provide attention to the VoVNet2 network and its efficient contribution as a backbone for other related computer vision tasks.

III. Methodology

To evaluate the before-mentioned architectures, a process needed to be adhered to to ensure a fair and equal comparison. This section targets the details of the evaluation starting first by giving an overview of the datasets being used and the metrics which the evaluation will utilise. Furthermore, the setup and methodology are discussed in further details giving information regarding the pre-trained models being used and the hardware being used for this experimentation.

A. Datasets

The Microsoft Common Objects in Context [27] (MsCoco) dataset is a well-renowned dataset that contains 330,000 images with 1.5 million object instances and 80 object categories. The applications of this dataset vary from object detection to segmentation as well as image captioning. This dataset is also one of the most commonly used benchmarking datasets and therefore researchers utilise the validation or the testing set of this dataset to compare their architectures with other works within the same field. The 2017 Coco dataset consists of 118,000 images for training and 5,000 images for validation. This dataset also contains a test set in which annotations are not publicly available and the performance of the model on this dataset can only be verified through the Coco Evaluation Server.

The Tiny Pascal VOC is a subset of the Pascal VOC dataset [28] which contains 1349 training images and 100 test images with 20 common object classes out of the 11,530 images found in the original dataset. Similar to the MsCoco dataset, this dataset is famous for object recognition and detection tasks in the fields of computer vision and machine learning. Previous to the MsCoco dataset, the Pascal VOC dataset was widely accepted as a benchmark for object detection and similarly contains an Evaluation Server.

B. Metrics

Instance Segmentation algorithms are generally evaluated through the use of the Average Precision (AP) and mean Average Precision (mAP) [29]. When calculating the AP, one needs to consider a further two metrics: precision and recall. Precision is defined as the percentage of correct positive predictions made amongst all the predictions performed while recall is defined as the percentage of correct positive predictions made among all positive instances. The average precision is then calculated by the area under the precision-recall curve (PR curve).

The intersection over union (IoU) is another important metric when it comes to the evaluation of instance segmentation architectures to calculate the before-mentioned metrics. The IoU is determined by the area of overlap between two segmentation masks and is calculated by the following equation:

$$IoU = \frac{Area\,of\,Overlap}{Area\,of\,Union}$$

If a prediction is perfectly placed on the reference, a result of 1 is generated while contrarily if the prediction performed is completely off a value of 0 is generated. Using a pre-established threshold of this value, the evaluation metric will be able to determine if a prediction performed by the model should be classified as a positive prediction or else a negative prediction. This will establish the true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn) which are then used to calculate the precision and recall using the following equations:

$$Precision = \frac{tp}{tp + fp} \quad (1) \qquad Recall = \frac{tp}{tp + fn} \quad (2)$$

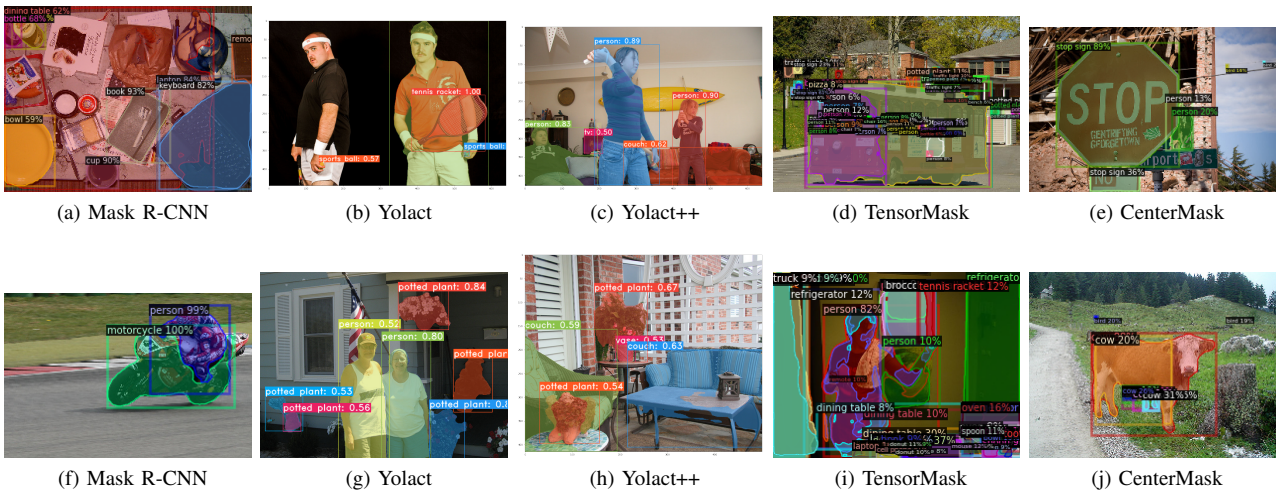
The threshold is established based on the researcher's preferences and the evaluation technique implemented. The MsCoco researchers offer an evaluation technique [27], [29] in which a range of IoU threshold values are taken into consideration and the mAP is calculated for each IoU, taking a final average to generate a final mAP value. Here the researchers make use of IoU thresholds ranging between 0.5 to 0.95 with a step size of 0.05 generating 10 precision-recall pairs. The mAP is then calculated by averaging over the IoU at each threshold in the following way:

$$mAP_{Coco} = \frac{mAP_{0.5} + mAP_{0.55} + ... + mAP_{0.95}}{10}$$

Using such a technique, threshold bias is eliminated since different thresholds are considered and given a different weighting.

C. Setup and Methodology

The architectures being evaluated for this research are those previously mentioned and therefore consist of the Mask R-CNN, Yolact, Yolact++, TensorMask and CenterMask. Each



(a) Mask R-CNN (b) Yolact (c) Yolact++ (d) TensorMask (e) CenterMask

(f) Mask R-CNN (g) Yolact (h) Yolact++ (i) TensorMask (j) CenterMask

Fig. 5: Sample inference images on the MsCoco17 validation set (top) and the Tiny Pascal Voc training set (bottom)

model was pre-trained on the MsCoco17 training dataset and as part of this research are being evaluated on two different datasets: the MsCoco17 validation dataset and the Tiny Pascal Voc training dataset. The Tiny Pascal Voc dataset performs the classification on 20 objects which are found in the MsCoco dataset, therefore transfer learning was not required. However, prepossessing on the labels had to be performed given that the labels varied from the two datasets. Such an example is the label airplane in which in the Tiny Voc dataset is referred to as an aeroplane. To handle such cases, a natural language processing library, the nltk was used to generate synonyms of the labels and perform the necessary mappings.

The pre-trained models and the architectures were retrieved from two different resources: Detectron2 [30] and Github[1]. Detectron2 [30] is a Facebook AI Research Project that consists of state-of-the-art object detection and segmentation models and it also offers a baseline for researchers to utilise their library and build research projects based on it. The Mask R-CNN model being used is available on Detectron2 through ModelZoo[2] and consists of the Resnet-101 with an FPN as the backbone. Differently, TensorMask is a separate project that is made available in the Detectron2 library section dedicated to projects. The model being evaluated contains a Resnet-50 with an FPN backbone. CenterMask on the other hand is a research project that is based on and requires the Detectron2 library to work and it is made available through the researchers Github [31]. The CenterMask model evaluated for this research utilizes the VoVNetV2 backbone with an FPN. The Yolact and the Yolact++ are not dependable on this library, and these models were also retrieved directly from the researchers Github [32]. The Yolact model being used is the base model that utilizes the Resnet-101 with an FPN backbone while similarly, the Yolact++ model used is the base model utilizing a Resnet-101 and an FPN backbone.

The evaluation of the models was carried out based on the Coco evaluation metric and for the Mask R-CNN, TensorMask and CenterMask the evaluation was carried out through the Detectron2 library. Differently, the Yolact and the Yolact++ was evaluated through the script implemented by the researchers of the architecture. The images were not pre-processed in any way and used directly. The evaluation of the discussed architectures was carried out on individual Google Colab notebooks utilising an Nvidia Tesla T4 GPU with 16GB GPU Memory and 12GB of RAM.

IV. Evaluation

The first analysis that was carried out prior to performing a thorough evaluation on each model was to perform inference on a number of sample images to verify that the model being used is correctly loaded and configured. Some of these sample images for the MsCoco17 validation dataset and the Tiny Pascal Voc training dataset can be analysed in Figure 5. From the inference of these images, one can analyse that the Mask R-CNN, the Yolact and the Yolact++ performed the cleanest inference while the TensorMask and CenterMask although generated accurate predictions were not as clean.

The architectures were then evaluated using the before-discussed evaluation metrics on the validation set of the MsCoco17 and the training set of the Tiny Pascal Voc. The classes and the distribution of both these datasets can be analysed in Figure 6 and Figure 7. From both these distributions it could be analysed that the class *person* is the most common occurring class by a large margin, while the other classes appear with a frequency that is relatively similar to each other.

The first analysis to be discussed is the evaluation of the architectures on the validation dataset of the MsCoco17. As shown in Table I the best performing model was the CenterMask with an AP score of 38.89. This is followed by TensorMask and Mask R-CNN that performed similarly to each other with an AP score of 35.82 and 35.59 respectively.

[1]GitHub is a code hosting platform used for version control and collaboration.

[2]Model Zoo is a machine learning model deployment platform.

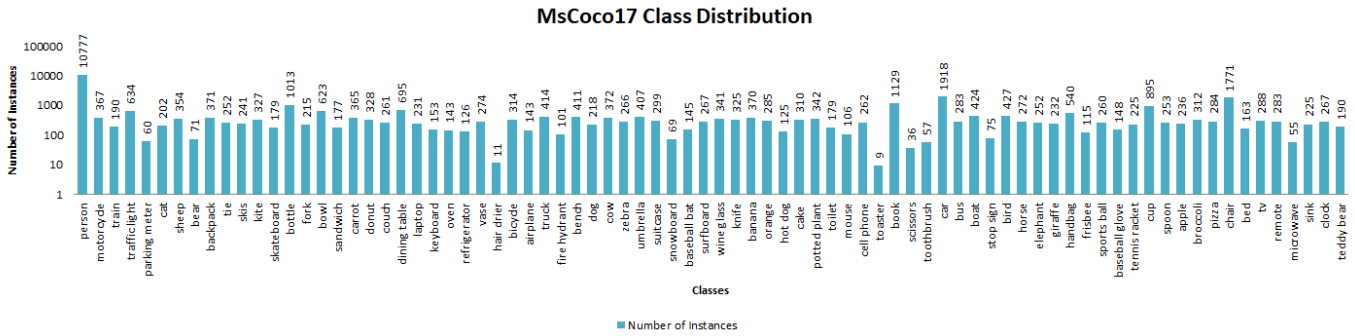


Fig. 6: Class Distribution of the MsCoco17

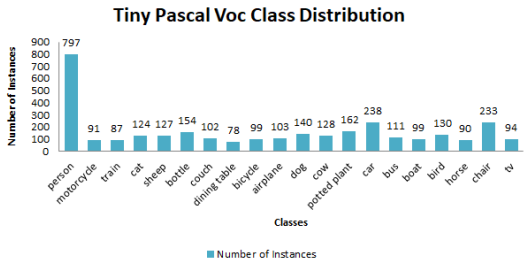


Fig. 7: Class Distribution of the Tiny Pascal Voc Dataset

The worst performance on this dataset was that of the Yolact and Yolact++ which achieved an AP score of 24.89 and 30.57 respectively. Although the performance of the Yolact++ was better than that of its precedent the Yolact, this was still not as good as those achieved by the other architectures. Bolya *et al.* [17] in the Yolact architecture focused on real-time inference rather than the performance of the model with the research conducted on the Yolact++ focusing on increasing the performance of the AP score. Although a distinguishable increase in performance was noticed, this was not sufficient to compete with the famous Mask R-CNN and the newer architectures of the TensorMask and CenterMask.

TABLE I: Evaluation on the MsCoco17 Validation Set

	AP	*AP50*	*AP75*
Mask R-CNN	35.59	54.31	38.67
Yolact	24.89	37.60	27.15
Yolact++	30.57	45.14	33.46
TensorMask	35.82	56.90	37.85
CenterMask	38.89	60.12	41.88

The architectures were then analysed on the Tiny Pascal Voc training set which is a total different dataset than that the architectures were exposed to during training. The performance of the architectures can be analysed in Table II and similar to what was established before, the CenterMask achieved the best AP score out of all the architectures considered with a value of 45.85. This architecture was closely followed by the Mask R-CNN with an AP score of 43.93 and TensorMask with an AP score of 41.98. The Yolact and the Yolact++ achieved the worst performance with an AP score of 12.82 and 14.87 respectively which is distinguishably lower than that of the other architectures evaluated.

TABLE II: Evaluation on the Tiny Pascal Voc Training Set

	AP	*AP50*	*AP75*
Mask R-CNN	43.93	72.62	46.73
Yolact	12.82	20.35	13.71
Yolact++	14.87	22.68	16.15
TensorMask	41.98	73.11	42.23
CenterMask	45.85	76.07	48.36

The performances of the architectures on both datasets were then compared for further evaluation as shown in Figure 8. From this analysis, it was concluded that for the Mask R-CNN, TensorMask and CenterMask the performance was similar on both datasets, with the AP being slightly higher when evaluating the Tiny Pascal Voc dataset. Contrarily, the Yolact and Yolact++ performance quite varied between the datasets with the architectures performing better on the MsCoco17 which is opposite to what was analysed from the other architectures.

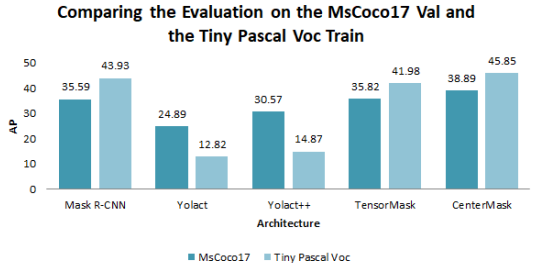


Fig. 8: Comparing the evaluation on the MsCoco17 Val with that on the Tiny Pascal Voc

The evaluation technique being used through the Detectron2 library allows for a deeper evaluation of the performance of the architectures on separate classes. Therefore the Mask R-CNN, TensorMask and CenterMask, the best three performing architectures, were furtherly evaluated with respect to separate classes and their contribution to the final evaluation result. Starting with the MsCoco dataset, an evaluation of the 80 classes present in this dataset shows that the AP score for every class is relatively similar between the different architectures

with the CenterMask being slightly higher than the other discussed architectures. The Mask R-CNN and the TensorMask as evaluated from the previous evaluation are rather neck and neck with some classes like *train* having a higher AP score through TensorMask while other classes such as *laptop* having a higher AP score through Mask R-CNN. This is reflected in the overall evaluation of these architectures. The analysis of each class for each architecture can be analysed in Figure 10.

A similar analysis was carried out on the Tiny Pascal Voc training dataset containing 20 different classes. Similar to what was concluded in the overall analysis of the architectures on this dataset, the CenterMask gave the best performance for each and every class. Between the Mask R-CNN and the TensorMask, it's not always the case that all the Mask R-CNN gave better AP scores than TensorMask. However, it can be analysed that for the majority the Mask R-CNN gave better results hence coming second to CenterMask in the overall analysis.

The final evaluation conducted takes into consideration the inference time of the architectures. The inference time per image for each architecture should not vary depending on the dataset, however, an evaluation was carried on each dataset nonetheless. Starting with the MsCoco17 validation, it could be identified that the longest inference time was when evaluating the TensorMask while the shortest was when evaluating the Mask R-CNN followed closely by the CenterMask. It was surprising to note that the Yolact and Yolact++ were not competitive architectures when it came to faster inference times given their real-time inference properties.

TABLE III: Inference Time on the MsCoco17 val dataset

	Inference Time (img/s)	*Total Inference Time*
Mask R-CNN	0.1615	0:13:26
Yolact	0.3425	0:28:15
Yolact++	0.8197	01:07:39
TensorMask	1.0531	1:27:40
CenterMask	0.1786	0:14:52

As expected a similar pattern follows for the Tiny Pascal Voc Train dataset. The CenterMask here performed the fastest inference on images followed by the Mask R-CNN. As before, the TensorMask gave the poorest inference times. The Yolact and Yolact++ once again did not give optimal inference times excluding them from being considered one of the best current architectures for fast inference time.

TABLE IV: Inference Time on the Tiny Pascal Voc train dataset

	Inference Time (img/s)	*Total Inference Time*
Mask R-CNN	0.2013	0:04:30
Yolact	0.3937	0:08:51
Yolact++	0.3636	0:08:10
TensorMask	0.9977	0:22:20
CenterMask	0.1705	0:03:49

From this evaluation, we have analysed the performance of the Mask R-CNN, Yolact, Yolact++, TensorMask and CenterMask both from an AP score perspective as well from an inference time perspective. Overall it was analysed that the CenterMask was the best performing architecture outputting the greatest AP score while also retaining fast inference time.

V. Conclusion

In this research study, we have analysed current state-of-the-art instance segmentation techniques and performed a thorough evaluation of these architectures on the MsCoco17 validation dataset and the Tiny Pascal Voc train dataset. We start firstly by considering the Mask R-CNN [4] which is referred to by any as the current state-of-the-art and progress to discuss the Yolact [17] and Yolact++ [20] which are famously known for being the first real-time instance segmentation techniques. The TensorMask [23] and the CenterMask [24], two relatively new architectures introducing new concepts, are also being considered for this research. From the evaluation conducted it could be concluded that the CenterMask gave the best overall AP score in addition to also having a competitive inference time surpassing overall all the other architectures including the famous Mask R-CNN. Therefore the CenterMask can be considered to be the current state-of-the-art instance segmentation technique.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

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

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278 – 2324, 12 1998.

[4] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[5] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–21, 01 2019.

[6] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *CoRR*, vol. abs/1907.09408, 2019.

[7] A. Hafiz and G. Bhat, "A survey on instance segmentation: state of the art," *International Journal of Multimedia Information Retrieval*, vol. 9, 09 2020.

[8] B. Li, Y. Shi, Z. Qi, and Z. Chen, "A survey on semantic segmentation," pp. 1233–1240, 11 2018.

[9] J. Wilson, "Could you explain me how instance segmentation works?." https://ai-pool.com/d/could-you-explain-me-how-instance-segmentation-works, 2019.

[10] E. Dayangac and G. Hirtz, "Object recognition for human behavior analysis," in *2014 IEEE Fourth International Conference on Consumer Electronics Berlin (ICCE-Berlin)*, pp. 64–68, 2014.

[11] Z. Yang and R. Nevatia, "A multi-scale cascade fully convolutional network face detector," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 633–638, 2016.

[12] S. Howal, A. Jadhav, C. Arthshi, S. Nalavade, and S. Shinde, *Object Detection for Autonomous Vehicle Using TensorFlow*, pp. 86–93. 01 2020.

[13] B. Hariharan, P. Arbeláez, R. B. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *ECCV*, 2014.

[14] S. Ren, K. He, R. Girshick, and S. J, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015.

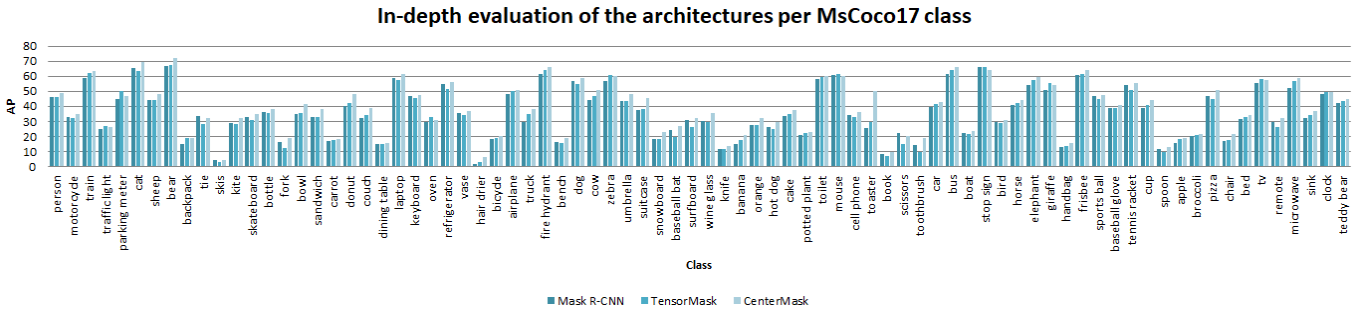


Fig. 9: In-depth evaluation of the architectures per class using the MsCoco17 val dataset

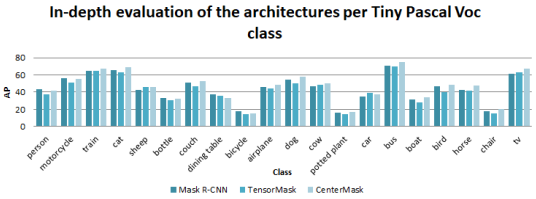


Fig. 10: In-depth evaluation of the architectures per class using the Pascal Tiny Voc train dataset

[15] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[17] D. Bolya, C. Zhou, F. Xiao, and Y. Lee, "Yolact: Real-time instance segmentation," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9156–9165, 2019.

[18] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017.

[19] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[20] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT++: better real-time instance segmentation," *CoRR*, vol. abs/1912.06218, 2019.

[21] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring r-cnn," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6402–6411, 2019.

[22] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," *CoRR*, vol. abs/1703.06211, 2017.

[23] X. Chen, R. B. Girshick, K. He, and P. Dollár, "Tensormask: A foundation for dense object segmentation," *CoRR*, vol. abs/1903.12174, 2019.

[24] Y. Lee and J. Park, "Centermask : Real-time anchor-free instance segmentation," *CoRR*, vol. abs/1911.06667, 2019.

[25] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: fully convolutional one-stage object detection," *CoRR*, vol. abs/1904.01355, 2019.

[26] Y. Lee, J. Hwang, S. Lee, Y. Bae, and J. Park, "An energy and gpu-computation efficient backbone network for real-time object detection," *CoRR*, vol. abs/1904.09730, 2019.

[27] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[29] R. Padilla, S. Netto, and E. da Silva, "A survey on performance metrics for object-detection algorithms," 07 2020.

[30] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[31] Y. Lee and J. Park, "Centermask." https://github.com/youngwanLEE/CenterMask, 2020.

[32] D. Bolya, C. Zhou, F. Xiao, and Y. Lee, "Yolact." https://github.com/dbolya/yolact, 2020.