Performance Study of Pre-trained Networks on Multi-Object Tracking and Segmentation

TUNAR MAHMUDOV[1], SAMET DEMIR[1], AND ZULFUGAR VERDIYEV[1]

[1] *Bilkent University, 2020, Ankara, Turkey*

June 14, 2020

Abstract - This paper aims to find the best feature extraction architecture for MOTS challenge. For feature extraction step, the initial architecture proposed in Faster-CNN is ResNet-101. The extensions of Faster R-CNN, which are Mask R-CNN and TrackR-CNN, does not utilize any other model which is proved to be faster and have better performance such as Xception. To find the best model for MOTS, we have tried five different models excluding the initial model, which are VGG-16, ResNet-50, ResNet-152, and Xception and compared their performances using MOTSP, MOTSA and sMOTSA metrics.

1. INTRODUCTION

After the Deep Learning, the computer vision era has made a great improvement in most aspects. One of the aspects in which Deep Learning has been utilized is the challenge of tracking multiple objects in a video. For this challenge, the Deep Learning architecture is used to locate the bounding boxes that surrounds the detected objects on particular scenes in the video. This approach worked better than the previous approaches not having Deep Learning architecture. To enhance the bounding box approach, pixel-level detection has recently been tried and made a significant improvement in MOT challenge. The models started detecting the occluded objects unlike the bounding box approach.

Previously, the segmentation and tracking were handled separately. Since we are utilizing the output of the segmentation stage while tracking the objects, the information of the pixels of the objects that could be useful for tracking purposes cannot be used in the tracking stage. Multi-Object Tracking and Segmentation (MOTS) have proposed to combine segmentation and tracking challenges in a single challenge [1]. They have performed tracking and segmentation in the same model. With help of this approach, the performance of the previous models has been increased.

In segmentation of the objects, the MOTS model is first extracting the features from the given the image. For this purpose, ResNet-101 architecture has been used. The reason why they have used ResNet-101 is that at the time of the first MOT challenge model using Deep Learning, Faster R-CNN, is implemented, the latest and the best performing feature extraction model was ResNet-101. Nowadays, there are better architectures such as Xception. In this paper, for TrackR-CNN model, we have tried 4 different feature extraction networks to compare their performances. We have compared initial model ResNet-101 with ResNet-50, ResNet-152, VGG-16, and Xception.

2. RELATED WORK

To solve the multi-object tracking challenge, there are a vast number of methods available and proposed so far. Before the Deep Learning era, the solutions to MOT task was not as successful as they are expected to be. After the improvement of Deep Learning architectures, one of the most successful systems on MOT task, Faster R-CNN [2], is proposed in 2015. It was based on bounding-box level segmentation. It was considerably more successful than the existing non-Deep Learning architectures in MOT task and it has shown that with the help of Deep Learning, this task can be successfully completed.

Two years later than the Faster R-CNN in 2017, Mask R-CNN [3] is proposed as an extension of Faster R-CNN. They have proposed to use masks rather than the bounding boxes. They have generated masks from the extracted features, which has eased to evaluate obtained results more precisely. On top of Mask R-CNN, TrackR-CNN [1] is proposed in 2019 and has extended the Mask R-CNN with two 3D convolutional layers and with a association of objects with vectors to decrease the ID switches. As a convention, these models have utilized the versions of Residual Network architecture (ResNet). Faster R-CNN, Mask R-CNN and TrackR-CNN have used ResNet-101.

In the implementation of Faster R-CNN, they have

extracted the features using VGG-16 model first. Then, to enhance the performance of the system, they have replaced it with ResNet-101. It is claimed in the paper of Faster R-CNN, by changing only the feature extraction layer from VGG-16 to a 101-layer residual net (ResNet-101), the Faster R-CNN system increases its performance with respect to their metric mAP from 41.5%/21.2% (VGG-16) to 48.4%/27.2% (ResNet-101) on the COCO val set [4]. However, in this paper, we are using different dataset (KITTI MOTS) and different metrics such as MOTSA and sMOTSA, which are introduced two years later than Faster R-CNN. In addition to dataset and metrics difference, we have used TrackR-CNN instead of Faster R-CNN. Therefore, in our experiment, it is likely that we could come up with different results than the results obtained in Faster R-CNN.

After the improvement in Deep Learning era, Deep Learning has started to be used to segment the objects before processing them. Because of the limitations of the memory and GPU, the first architectures were shallow neural networks. Even 14 layers was considered as deep. So, a 16-layer architecture, VGG-16, has been proposed in 2014 [5]. It has detected the objects better than previously implemented networks. However, since it includes 138 million parameters, its size was enormous and its training and evaluation speed was considerably slow.

As an alternative of VGG-16, Google has come up with an architecture called Inception [6]. Inception has 4 versions that have been introduced in between 2015-2016. The most frequently used version is Inception V3. In contrast to VGG-16, it is a much deeper and small sized network. Since it is deeper and has less parameters, it has provided faster and better results than VGG-16.

In theory, the loss of neural networks needs to decrease over time when the number of epochs increase. However, this was not the case in reality. In practice, the loss of the neural networks decreases till a threshold value and eventually starts to increase over time. This problem have been solved by a network called Residual Network (ResNet). This network has been proposed in 2016 [7]. The depth of the network has been increased over years and the version are named with their number of layers. For instance, ResNet-50 has 50 layer depth. ResNet has been utilized in many fields and Faster R-CNN has started using ResNet in MOT challenge.

After realizing the effect of residual networks, Google has come up with another architecture called Xception and they have proposed depthwise convolution followed by a pointwise convolutional blocks [8]. With this approach, they have achieved a better accuracy and less error rate by using a small number parameters.

3. EVALUATION MEASURES

To evaluate the performance of each architecture, we have first trained the model by using each architecture and used the metrics introduced together with TrackR-CNN to calculate how well the model works. These metrics are IDS, MOTSP, MOTSA and sMOTSA. Before giving the definitions of each metric, first we shall give some of the definitions.

All the evaluation measure definitions are taken from MOTS [1]. The ground truth of a video has T time frames. Each frame has height h, width w, and N ground truth pixel mask values $M = \{m_1, ..., m_N\}$ where $m_i \epsilon \{0,1\}^{hxw}$. each m value corresponds to a time frame $t_m \epsilon \{1, ..., T\}$ and has a track id $id_m \epsilon N$. The method outputs a set of K non-empty predicted masks $H = \{h_1, ..., h_K\}$ where $h_i \epsilon \{0,1\}^{hxw}$, each h value corresponds to a predicted track id $id_h \epsilon N$ and a time frame $t_h \epsilon \{1, ..., T\}$.

Intersection-over-Union (IoU) is used to determine how well the prediction is in pixel level. It is the ratio of the union of the ground truth object and the predicted mask with their intersection. So, it takes two masks as a parameter and returns a value between 0 and 1. By using IoU, we define a function c that will return the corresponding ground truth object of the given detected object and will be defined as follows.

$$c(h) = \begin{cases} \underset{m \epsilon M}{argmax}\ \text{IoU(h,m)}, & \text{if } \underset{m \epsilon M}{max}\ \text{IoU(h,m)} > 0.5 \\ \emptyset, & \text{otherwise.} \end{cases} \quad \textbf{(1)}$$

Therefore, we can define true positive as $TP = \{h \epsilon H | c(h) \neq \emptyset\}$ and false positive as $FP = \{h \epsilon H | c(h) = \emptyset\}$. By using the same idea, $FN = \{m \epsilon M | c^{-1}(h) = \emptyset\}$. Finally, we will have a *pred* function that will return the predecessor of a ground truth mask.

Now, we shall define our first metric IDS, which stands for the number of ID switches.

$$IDS = \{m \epsilon M | c^{-1}(m) \neq \emptyset \wedge pred(m) \neq \emptyset \wedge id_{c^{-1}(m)} \neq id_{c^{-1}(pred(m))}\} \quad \textbf{(2)}$$

To determine how well we have detected the objects in pixel-level, we will have a soft TP value. This value is the summation of all the IoU values for each truly detected object and its ground truth value. We want this number to be as high as possible and the maximum value of soft TP value is equal to the number of truly detected objects.

$$\widetilde{TP} = \sum_{h \epsilon TP} IoU(h, c(h)) \quad \textbf{(3)}$$

To evaluate the performance of the architecture, we used MOTSA, MOTSP values which are defined as follows.

$$MOTSA = 1 - \frac{|FN| + |FP| + |IDS|}{|M|} = \frac{|TP| - |FP| - |IDS|}{|M|} \quad \textbf{(4)}$$

$$MOTSP = \frac{\widetilde{TP}}{|TP|} \quad \textbf{(5)}$$

And we will use the sMOTSA value, which is a deviation of MOTSA. The only change is the replacement of TP value with the soft TP value to evaluate the pixel level correctness of the system.

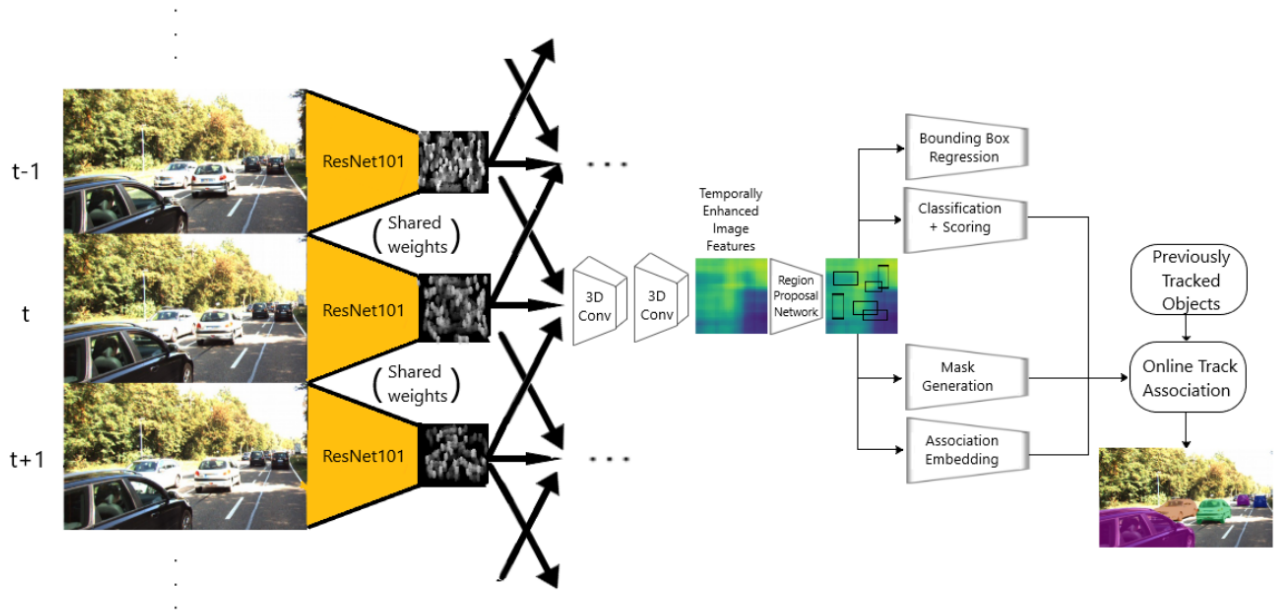

Fig. 1. The Model of TrackR-CNN

$$sMOTSA = \frac{\widetilde{TP} - |FP| - |IDS|}{|M|} \quad (6)$$

4. METHOD

A. Model of TrackR-CNN

TrackR-CNN is a Deep Learning model to track multiple objects. It is the first model proposed as a solution to MOTS challenge. The general flow is shown in Figure 1. Since it is an extension of Mask R-CNN, which is also an extension of Faster R-CNN, they have some additions on top of each other. The general flow of Track-RCNN is that it first extracts the features by using ResNet-101 network for each frame. Then, it has two 3D convolutional layers which combines the extracted features of the frame at time t, t+1 and t-1. These convolutional layers are followed by a Region Proposal Network, which detects the possible positions of the objects. Using the possible positions of the objects, it finds the bounding boxes, classifies the object, generates the mask and calculates a vector value, which has close distance for the same objects in different frames. Detected class, mask, vector and previously tracked objects are used to detect the objects int the current frame.

The reason why they have used ResNet-101 is that when Faster R-CNN has been proposed, ResNet-101 was the best performing network among all the networks. So, Faster R-CNN has decided to use ResNet-101. Also, Mask R-CNN and TrackR-CNN did not work on the feature extraction step to find the best performing algorithm among the available algorithms at their time. So, we have aimed to discover the network which will perform the best. The algorithms we have tried are VGG-16, ResNet-50, ResNet-101, ResNet-152, and Xception. We have used KITTI-MOTS dataset [9], same as TrackR-CNN.

B. VGG-16

VGG-16 is one of the first Deep Learning architectures used in feature extraction. It was proposed in 2014 and was rewarded as the 1^{st} Runner up in ILSVRC 2014 [10]. VGG-16 has and 5 convolution blocks and 1 Fully Connected block. It has 16 layers in total as shown in the Figure 2. Since it has 16 layers, it is called VGG-16.

VGG-16 suggests 5 convolutional blocks and decreases the size of the image with the help of max pooling before each block. They also propose to keep the number of convolutions constant inside the block and double it after each block.

Since it has 3 fully connected layers of 4096 neurons, we have $3 * (4096 * (4096 + 1)) = 50,343,936$ parameters only for the last block. When the parameters coming from the other layers are taken into consideration, VGG-16 has $138,357,544$ parameters. Since in Deep Learning, more parameters result in a slower network, we want a model that has less number of parameters and gives a better performance. Since 138 million parameters are remarkably huge, it is considerably slow compared to other networks we have used.

According to statistics conducted by keras [11] (see Table 1), the VGG-16 has the least Top-1 Accuracy and Top-5 Accuracy among all the networks we tried in this paper.

C. ResNet-50

ResNet has a special place in Deep Learning. It solves the problem of the increase in loss after a threshold value of

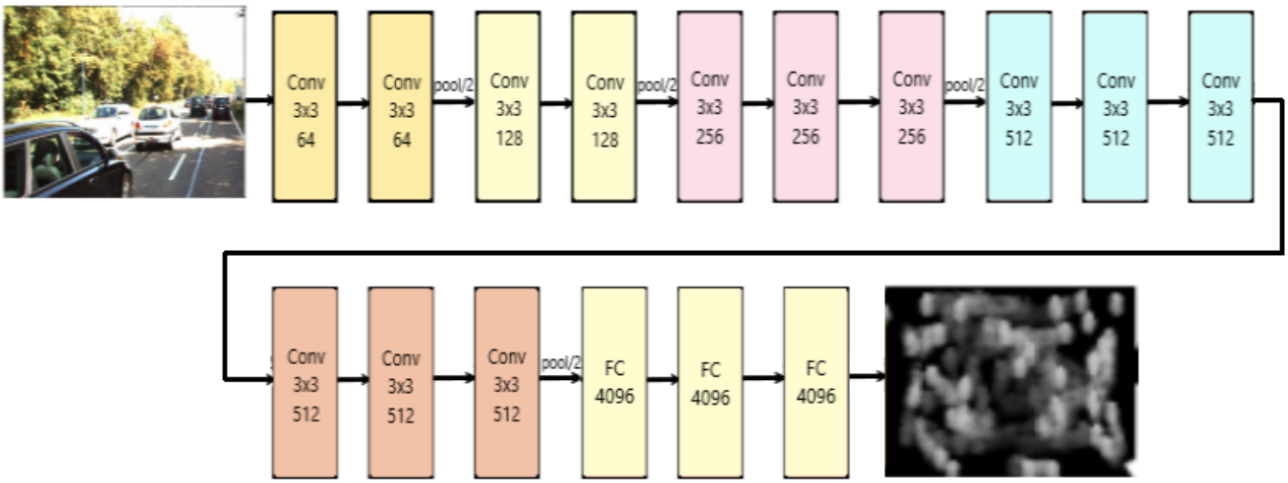

Fig. 2. The Architecture of VGG-16

number of epochs. It solves the problem by using the current layer's parameters together with the previous layer's parameters. Due to this, it is called Residual Network (ResNet). It has several versions. The versions are named with the number of layers. For instance, ResNet-50 has 50 and ResNet-101 has 101 layers. The architecture of ResNet-50 is provided in Figure 3.

ResNet-50 has 4 convolutional layers, each of which has several ResNet blocks and each ResNet block has 3 convolutions. The number of ResNet blocks in convolutional layers are 3, 4, 6, and 3 respectively. To make the smaller, 2x2 stride is applied before each convolutional layer.

D. ResNet-101

ResNet-101 is the network used in TrackR-CNN. It has almost the same architecture as ResNet-50. The only difference is the number of ResNet blocks in each layer. The first, second and last layers has the same number of ResNet blocks whereas third layer has 23 ResNet blocks. They have decided to increase the number of blocks in third block because it increases the number of parameters and it provides considerably more precise results as shown in Table 1. However, there is a trade-off between the precision and the speed, when we increase the depth and the number of parameters, we slow down the model. This decrease in speed can be compensated by the new generation GPUs.

E. ResNet-152

Similar to ResNet-50 and ResNet-101, ResNet-152 has 4 convolutional blocks. The first and last blocks have 3 ResNet blocks as before. The second and third layers have 8 and 36 blocks respectively. ResNet-152 does not make a significant improvement in accuracy as shown in Table 1. In addition, it has 15 million more parameters than ResNet-101. So, it is expected to be slower than ResNet-101. Since our focus in this paper is to increase the accuracy, we expected ResNet-152 to perform slightly better than previous versions of ResNet architecture.

F. Xception

Xception [8] is proposed by Google in 2016. Its full name is Extreme version of Inception and it is called short Xception. So, it is a deviation of Inception model. They have included depthwise separable convolution, they achieved more accurate results than Inception-v3 [6]. One later than VGG-16's ILSVRC award, Xception has been rewarded next year as the winner of ILSVRC 2015 for ImageNet ILSVRC dataset as well as JFT dataset. One difference between the Inception V3 and Xcpetion is the sequence of the convolutions and the blocks. The order of Xception model can be seen in Figure **??**. The other difference is the non-linearity. The ReLu non-linearity after the first operation has been removed. With the help of these two changes, they have achieved better results as shown in Table 1. In addition, they have the least number of parameters among the models we tried. They have nearly 22.9 million parameters.

There are two versions of Xception model. One version uses Residual Network structure whereas the other version does not. The version with Residual Network is proved to be better in accuracy.

Since this model is the performing model in ImageNet ILSVRC dataset and JFT dataset, and since the number of parameters is less than all the models we used, Xception

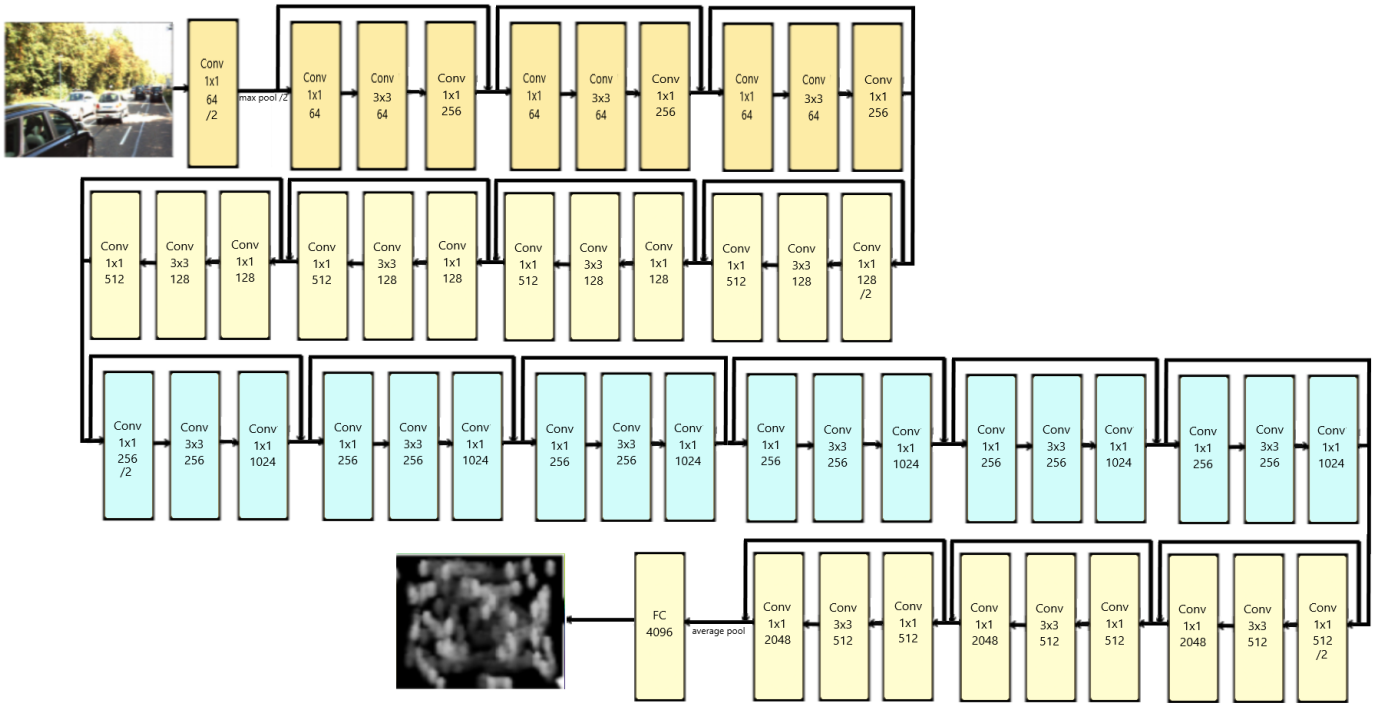

Fig. 3. The Architecture of ResNet-50

model has been expected to be the best model in our experiment.

G. Statistics by keras

Keras is a well-known Python library that is designed for Deep Learning. It has provided a model comparison table in their website [11]. According to their calculations and statistics, the performance of the algorithms in general is as follows. VGG-16, which is the first proposed one among the models we used, is expected to be the slowest and the worst performing model. ResNet architectures are expected to be much faster than VGG-16 and to perform better than VGG-16. When we upgrade the version of the ResNet model, we get slightly better performance according to Table 1. Lastly, since Xception model is the latest model in the models we tried, it works better than the previous models in general. It has less number of parameters and better accuracy. It makes a significant enhancement in accuracy and is remarkably faster than the VGG-16 and ResNet models. Therefore, we expected the best performing model as Xception.

In the Figure 5, we see the error rate vs the number of parameters of each model. Since we want less number parameters and less error rate, we expect a good model to be as close to origin as possible. Therefore, we can claim that among the models we used in this paper, Xception performs the best in general. However, this does not have to be the case in our experiment.

Model	Size	Top-1 Acc.	Top-5 Acc.	Parameters	Depth
VGG-16	528 MB	0.713	0.901	138,357,544	23
ResNet-50	98 MB	0.749	0.921	25,636,712	-
ResNet-101	171 MB	0.764	0.928	44,707,176	-
ResNet-152	232 MB	0.766	0.931	60,419,944	-
XCeption	88 MB	0.790	0.945	22,910,480	126

Table 1. Statistics of Network Models Conducted by keras

5. EXPERIMENTS

Table 2 shows different models on TrackR-CNN. In our case, 5 models - VGG16, ResNet-50, ResNet-101, ResNet-152, and XCeption were tested in order to get a better result.

For the sMOTSA metrics on the Car objects, XCeption shows the best result followed by the original network model ResNet-101. On this account, ResNet-50 performed the least accurate result. On the Pedestrian objects, however, VGG16 has the most accurate results with a significant dominance over the original network ResNet-101, which performed in the least accurate manner. Although in the paper of Faster R-CNN, it is claimed that ResNet-101 made a great improvement in mAP metric, in our experiment, we have obtained the opposite of their claim. There are three reasons behind this situation. One is that we used a model that extends Faster-RCNN. The additional parts might have affected the performance of VGG-16. The second reason is that we used a different metrics such as sMOTSA and MOTSA. They have used mAP metric.

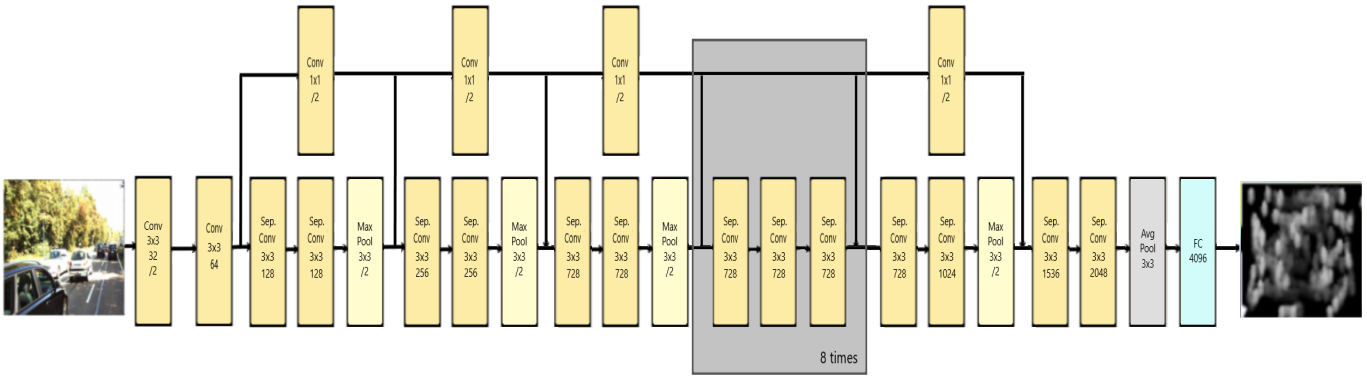

Fig. 4. The Architecture of XCeption

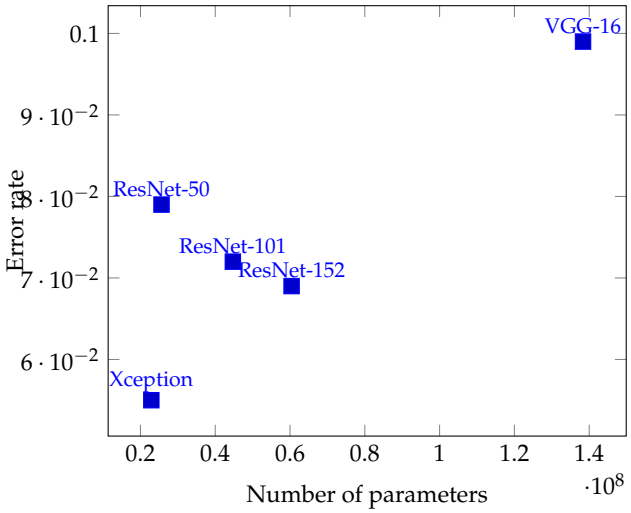

Fig. 5. Plot of Error Rate vs Number of Parameters for Statitcs of Keras

Model	sMOTSA		MOTSA		MOTSP		IDS	
	Car	Ped	Car	Ped	Car	Ped	Car	Ped
VGG-16	74.1	56.6	86.3	74.5	86.9	78.0	317	155
ResNet-50	72.1	52.0	84.5	66.0	86.8	79.6	299	117
ResNet-101	76.2	46.8	87.8	65.1	87.2	75.7	327	145
ResNet-152	73.9	55.0	86.0	72.3	86.9	78.1	370	170
XCeption	76.3	53.2	85.0	72.1	87.4	79.7	353	167

Table 2. Results of Our Experiment on Network Models

And the last reason is that we have worked on KITTI-MOTS dataset whereas they have worked on COCO object detection dataset. When we take these reasons into consideration, it is acceptable to have better results in VGG-16 model.

For the MOTSA metrics on the Car objects, ResNet-101 has the highest value and similar values are obtained for the rest of the models. On the other hand, VGG-16 shows much better results than the other models considering the Pedestrian objects of the same Metrics.

For the MOTSP metrics, XCeption performed best both for Car and Pedestrian objects compared to the rest of the models. While ResNet-50 has the second best result on the Pedestrian objects, second best result for the car objects were obtained from ResNet-101. However, among all, ResNet-101 performed the worst results on the Pedestrian objects.

Figure 6 shows the plot of the number of parameters of the 5 models versus their error rates on sMOTSA car detection results obtained from our experiments. Each square on the graph represents a network model correspondingly. Squares near the origin of the graph has the most accurate results due to their less number of error rates and squares far from the origin has more error rates and thus, less accurate results.

In our experiment on sMOTSA car detection results, XCeption performed the most accurately compared to the other network models followed by ResNet-101.

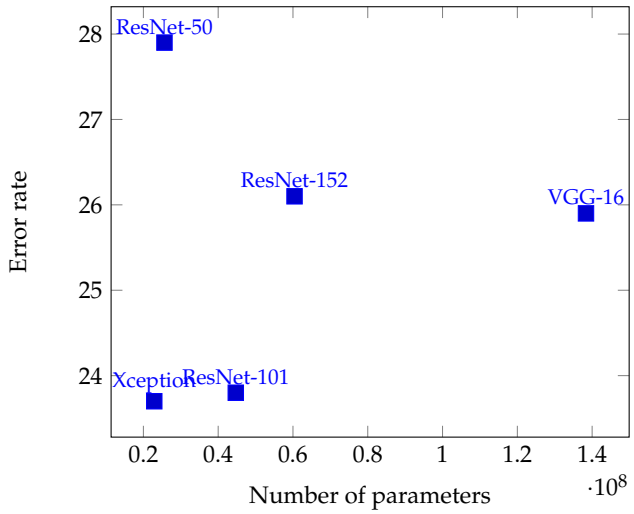

Fig. 6. Plot of Error Rate vs Number of Parameters of Our Experiment on sMOTSA car detection results

6. CONCLUSION

In this paper, we aimed to compare the results of different networks for feature extraction on TrackR-CNN system. We have observed that ResNet-101 was the most preferable choice at the time of Faster R-CNN. Mask R-CNN and TrackR-CNN did not make any change on the existing feature extraction model and it has stayed as ResNet-101. Nowadays, we have a newly introduced and better network named XCeption. It can be claimed by our experiment that that XCeption model has made an great improvement on the overall system. It has decreased the number of parameters and supplied better results on our metrics. Since MOTS challenge is a real-time task, we need to have a faster model. Because Xception model makes the system faster and more accurate, we propose to convert the feature extraction network from RestNet-101 to Xception. For future work, other networks that we did not use such as Inception V3 and DenseNet models can be tested on TrackR-CNN system. Our code is available at https://github.com/tinabd14/TrackR-CNN.

REFERENCES

1. P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "MOTS: multi-object tracking and segmentation," CoRR **abs/1902.03604** (2019).
2. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28,* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds. (Curran Associates, Inc., 2015), pp. 91–99.
3. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV),* (2017), pp. 2980–2988.
4. 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 **abs/1405.0312** (2014).
5. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR **abs/1409.1556** (2014).
6. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," (2016).
7. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR **abs/1512.03385** (2015).
8. F. Chollet, "Xception: Deep learning with depthwise separable convolutions," (2016). Cite arxiv:1610.02357.
9. A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR),* (2012).
10. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis. (IJCV) **115**, 211–252 (2015).
11. K. Team, "Keras documentation: Keras applications," .