Skip-SegFormer
Efficient Semantic Segmentation for urban driving

Andrea Lombardi[1,*], Emanuel Di Nardo[1] and Angelo Ciaramella[1]

[1]University of Naples Parthenope, Italy

Abstract

Environmental perception is a crucial aspect within the field of autonomous urban driving that provides information about the environment, identifying clear driving areas and possible surrounding obstacles. Semantic segmentation is a widely used perception method for self-driving cars. The predicted image pixels can be used to bias the vehicle's behaviour and avoid collisions. In this work a Semantic Segmentation model based on an architecture called SegFormer is proposed, made more efficient by using what our Skip-Decoder module. The model is fine-tuned on urban driving datasets and produces accurate segmentation masks in a short time, making the architecture perfectly adaptable to an autonomous driving car system.

Keywords

Autonomous driving, Image Segmentation, Computer Vision, Transformer

1. Introduction

The autonomous driving cars need to be equipped with the necessary perception to understand the nearby situation so that they can safely integrate into our existing roads and have enough information about the environment, clear driving areas and possible surrounding obstacles. One of the many sensor involved in autonomous driving is usually a camera, which allows the system to process the rich visual signal information using, for example, semantic segmentation, that allows the system to recognize possible obstacles and avoid collisions. This work proposes an accurate real-time semantic segmentation model for self-driving cars based on SegFormer [1], a Transformer-based architecture with a lightweight decoder and an efficient multi-head attention. This method guarantees a fast inference time and good performances. The model is fine-tuned and tested using urban driving datasets such as Cityscapes [2] or ApolloScape [3], showing the capability of SegFormer to be easily adaptable to downstream tasks. The proposed architecture is a variation of the original SegFormer implementation, which uses a so called Skip-Decoder which simulates the U-Net "expanding path" and expands the hidden states using different local size information at each iteration.

By using the term "autonomous driving" in this paper, we are referring to all the different levels and possible automation systems with different percentages of human control [1]. Deductions will follow regarding the level of autonomous driving to which the proposed work refers.

2. Related work

One of the most important work in deep semantic segmentation was U-Net [4], a model that strongly relies on data augmentation and uses some sort of skip connection between the encoder and the decoder to "preserve" the features during the up-sampling step. Years later ViT [5] shows that the reliance on CNNs is not necessary, in fact it is the first work to prove that a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. Recent methods such as T2T ViT [6], ViT ADP [7] introduce tailored changes to ViT to further improve image classification performance. Other recent works like Swin Transformer [8] and CvT [9] enhance the local continuity of features in the image removing fixed size position embedding to improve the performance of Transformers in dense prediction tasks. For semantic segmentation in particular, SETR [10] provides an alternative perspective by treating semantic segmentation as a sequence-to-sequence prediction task. A relevant work, using transformer architecture, has been proposed by SegFormer [1], a powerful segmentation framework made by two main ultra-efficient modules. We will further elaborate more about SegFormer in the , exploring the modules and their approaches. However, some aspects for semantic segmentation such as computational efficiency has not been thoroughly studied in the literature and, in fact, these Transformer-based methods have very low efficiency and, thus, difficult to deploy

[1]https://www.synopsys.com/automotive/autonomous-driving-levels.html

Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy

*Corresponding author.

emanuel.dinardo@uniparthenope.it (E. Di Nardo); angelo.ciaramella@uniparthenope.it (A. Ciaramella)

0000-0002-6589-9323 (E. Di Nardo); 0000-0001-5592-7995 (A. Ciaramella)

in real-time applications. Crucial applications such as road scene understanding in autonomous vehicles need much more segmentation accuracy without affecting the efficiency.

3. Datasets

Thanks to the recent works and the state-of-the-art methods in semantic segmentation, a variety of datasets such as ADE20k [11], COCO-Stuff [12] and PASCAL VOC [13] have been proposed, but none of them is precisely developed from a urban driving environment. Recently, significant research efforts have gone into new vision technologies for understanding complex traffic scene and driving scenario. In this paper, we use two challenging datasets:

- **Cityscapes** [2]: is a benchmark suite with a corresponding dataset specifically tailored for autonomous driving in an urban environment and involving a much wider range of highly complex inner-city street scenes that were recorded in 50 different cities with different sizes, geographic position and different time of the year. The base dataset consists of 5000 fine pixel-level annotations layered polygons and realized in-house to guarantee highest quality levels.
- **ApolloScape** [3]: ApolloScape is a dataset used to prove the learning strength of the model, and was choosen for its stronger challenging environments. For instance, high contrast regions due to sun light and large area of shadows from the overpass. The specifications of ApolloScape for the semantic scene parsing are the following: 143906 video frames and their corresponding pixel-level semantic labelling. The number of given samples is large and, furthermore, the dataset is more complex due to the variety of the environments and image features.

3.1. Metrics

In this work, the **Mean IoU** has been used, a particular version of the famous Jaccard index calculated by taking the IoU of each class and averaging them, giving in output a single value. The Jaccard index measures the similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. Another important metric referencing to the inference time is **FLOPs** (Floating Point Operations), that represents the total number of floating point operations required for a single forward pass. The higher the FLOPs, the slower the model and hence low throughput. We will use this metric to measure the efficiency of the proposed model.

4. Proposed method

This section introduces the basic model used and the proposed version of the architecture.

4.1. Segformer in detail

SegFormer [1] is an efficient, robust and powerful segmentation framework without hand-crafted and computationally demanding modules. The architecture consists of two main modules: a *hierarchical Transformer encoder* to generate high-resolution coarse features and low-resolution fine features; and a *lightweight All-MLP decoder* to fuse these multi-level features to produce the final semantic segmentation mask. Image patches of size 4 x 4 (smaller patches favors the dense prediction task) are used as input to the hierarchical Transformer encoder to obtain multi-level features at 1/4, 1/8, 1/16, 1/32 of the original image resolution, that are then passed to the All-MLP decoder to predict the segmentation mask at $\frac{H}{4}$ x $\frac{W}{4}$ x $N_{classes}$ resolution. More in detail, the two main modules can be explained as follows:

- **Mix Transformer Encoder**: the goal of the entire encoder module, is to generate CNN-like multi-level features and reduce the original multi-head self-attention computational complexity of $O(N^6)$ which, for large image resolutions, becomes really prohibitive. Instead, a sequence reduction process is used. Given a *reduction ratio* R, it is computed as:

$$K_1 = Reshape(\frac{N}{R}, C * R)(K) \quad (1)$$

$$K = Linear(C * R, C)(K_1) \quad (2)$$

  As a result, using this *Efficient Self-Attention* the complexity of the self-attention mechanism is reduced from $O(N^2)$ to $O(\frac{N^2}{R})$.
- **Lightweight All-MLP Decoder**: The encoder incorporates various high-performance submodules, but the most of the work is done by the lightweight All-MLP decoder. This simple decoder consists of four steps: first, multi-level features from the MiT encoder go through an MLP lyer to unify the channel dimension; then, these features are then up-sampled to 1/4th of the input size and concatenated together; next all the features are fused using an MLP layer and are finally given in input to another MLP layer to produce the segmentation mask. By aggregating the information from different layers, the MLP decoder combines both local and global attention.

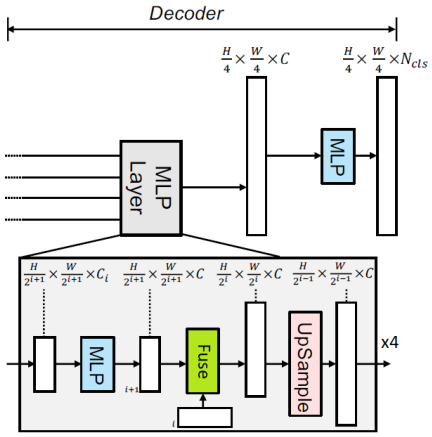

Figure 1: Architecture of the proposed version of the SegFormer decoder, called Skip-Decoder.

4.2. SegFormer with Skip Connections

The All-MLP decoder is what makes the SegFormer architecture so fast and lightweight. The original up-sampling stage is basically performed with a *bilinear interpolation* which makes a large number of the features to be estimated using just a portion of the $\frac{H}{4}$ x $\frac{W}{4}$ x C features. This work proposes a variant of the previously mentioned decoder, in which the up-sampling stage uses more features to build the feature map with size $\frac{H}{4}$ x $\frac{W}{4}$ x C that is eventually given in input to the MLP classifier layer. Taking inspiration from the work proposed by U-Net [4] authors, this decoder uses some sort of *skip-connections*. Considering the decoder as an *expansive path*, each decoder step doubles the feature map size and concatenates two encoder hidden state outputs at each iteration. As shown in Figure 1, there are few simple steps in this decoder. First, the i-th hidden state is fused with the previous one (assuming that the first has already been up-sampled) using a 1x1 convolution that keeps the same size of the features. Then, this $fusedState$, is up-sampled to match the size of the hidden state of the next iteration. The up-sampling is performed as a bilinear interpolation. This loop is performed 4 times, because in our experiments the encoder is made by 4 transformer encoder blocks. Finally, another MLP classifies these fused encoder hidden states to produce the segmentation map with size $\frac{H}{4}$ x $\frac{W}{4}$ x $N_{classes}$.

During the decoding stage, in the classic SegFormer [1], each encoder hidden state is up-sampled to to $\frac{1}{4}$th of the image to be fused with the other ones. Consider for example an input image of 1024 x 1024 and let's reason in terms of pixels. With the hidden states with size $\frac{1}{32}$th, $\frac{1}{16}$th and $\frac{1}{8}$th of the image, the amount of pixels to estimate with the interpolation is $28672 + 49152 + 65536 = 143460$ pixels. dUsing the decoder described in this section, the amount of pixels to estimate (given in input an image with size 1024 x 1024) is 86016. Hence, this method estimates 40% fewer pixels using, for the interpolation, the features of the previous block which should give a better understanding of the data.

5. Experiments

In this section the tests made on the proposed methodologies are compared and discussed to figure out the limitations of the propounded methods and eventually motivate how any modification in the architecture increases or decreases the performances.

5.1. Expertimental Settings

The encoder is pretrained on Imagenet-1k [14] dataset and the decoder is randomly initialized. During training, data augmentation was applied on Cityscapes dataset through random cropping to 1024 x 1024 and random horizontal flipping. The ApolloScape dataset, instaed, was first resized to 2048 x 1024 and then passed through the same data augmentation as the Cityscapes dataset. AdamW optimizer [15] has been used with an initial learning rate value of 0.0006, combined with a StepLR schedule with a factor of 0.5 and a patience of 5 epochs. During the training, test and evaluation, before any measurement, the output of the model has been restored to the full image size using a bilinear interpolation.

5.2. Results

The proposed SegFormer model and its variant are tested on both datasets presented in section 3, and then compared to the state-of-the-art architectures. A discussion about limits, performances and model features is provided in this section.

Method	FLOPs	mIoU
FCN - MobileNet v2	317.1	61.5
DeeplabV3 - MobileNet v2	556.2	75.2
EncNet - ResNet101	1748	76.6
FCN - ResNet101	2203.3	76.9
DeeplabV3 - ResNet101	2032.4	80.9
MiT-B0	125.5	56.9
MiT-B1	247.9	61.9
MiT-B0 - SkipDecoder (**Ours**)	107.3	57.6
MiT-B1 - SkipDecoder (**Ours**)	230.7	61.8

Table 1
Results and comparison on Cityscapes dataset. The FLOPs are also provided for each model to show the performance/efficiency trade-off.

	IoU	
Cityscapes Class	**MiT-B1 CD**	**MiT-B1 SD**
road	96.86	96.51
sidewalk	77.36	75.29
building	87.72	87.52
wall	40.55	35.04
fence	39.17	39.62
pole	34.53	34.11
traffic light	44.37	44.63
traffic sign	53.01	52.69
vegetation	89.46	89.44
terrain	65.35	66.73
sky	93.00	93.42
person	66.93	65.62
rider	40.94	37.65
car	91.28	90.95
truck	50.62	51.26
bus	57.61	64.90
train	56.89	66.48
motorcycle	35.18	27.25
bicycle	55.92	55.24

Table 2
Class IoU of SegFormer MiT-B1 using the Classic Decoder (CD) and the Skip Decoder (SD).

5.2.1. Results on Cityscapes

As it can be seen from Table 2, the most important classes are well detected and segmented. Summarily, the model detects all the flat surfaces and the constructions, respectively with a 96.93 and 87.58 IoU. On the flip side, there are also different irrelevant misclassified classes, strictly related to occlusion problems, and it is a common issue that different systems have. Having a look at the image *b* in Figure 2: it is evident that the model has problems in distinguishing the man's leg from the bicycle, due to the uncommon position taken by the rider, who is usually seen as a pedestrian in an upright position. In fact, this problem also leads to a medium/low accuracy on the *object* category classes. Without these less relevant classes, the mean IoU would be about **87.3%**, which is largely comparable with the state-of-the-art methods and outperforms the methods listed in Table 1.

The same situation was replicated with the proposed SegFormer variant which uses the decoder described in subsection 4.2. As expected, the model is faster despite having the exact same number of layers and the same performance. On the other hand, this version brings up the same issues: the up-scaling technique used in the Skip decoder does not resolve the low IoU encountered over the *object* classes. The MiT-B0 model, being a lighter version of the MiT-B1 model, shows a lower mIoU, but brings with it all the advantages and disadvantages. In this case the model is very fast and portable, and the accuracy makes the model suitable for a driver assistance system under the 3rd level, not sufficient to present it as a valid proposal for an autonomous driving system with high automation.

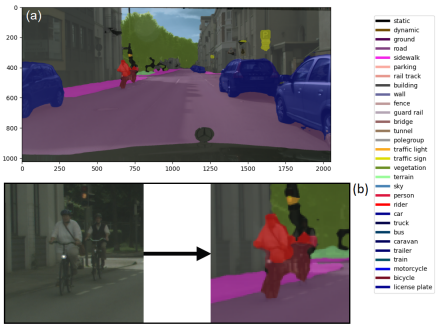

Figure 2: SegFormer MiT-B1 segmentation mask on Cityscapes image test. The image *a* represents the segmentation mask on the original image. The image *b* shows a particular segmentation issue.

5.3. Results on ApolloScape

Despite ApolloScape is a very demanding and challenging dataset, the results are partially comparable to those of Cityscapes. This dataset has multiple classes that refer to particular elements of the environment that cloud be considered less relevant to the autonomous driving scene understanding, such as road piles, dustbins, tunnels and bridges. Despite the good mIoU, several relevant classes are not well segmented by the MiT-B1 model, such as *person*, *wall* and *motorcycle*. The model is not capable to generalize on a very challenging dataset, and these results shows us the handicaps of very lightweight architectures in real-life applications. Anyway, it is possible to appreciate the MiT-B1 architecture performance compared to the MiT-B0 architecture that was not able to reach the same level of accuracy despite having a very large number of samples to be trained on. This could be caused by the small size of the hidden layers which is not enough to guarantee a good generalization capability but, on the other hand, it is also the reason why the model has a low inference time. More parameters are needed to truly appreciate the model's performance on such demanding data.

On ApolloScape, the proposed Skip-decoder SegFormer performed as good as on Cityscapes without any meaningful accuracy improvement. The skip-decoder also shows how the proposed architecture is capable of keeping the same results regardless of the features of the dataset which, in ApolloScape case, are very challenging and tough. Given the results, the model shows a good robustness even on such demanding datasets.

Method	FLOPs	mIoU
ResNet-38	175.4	43.07
ERFNet-IntRA-KD	-	43.02
MiT-B0	125.5	58.7
MiT-B1	247.9	66.4
MiT-B0 - SkipDecoder (**Ours**)	107.3	57.5
MiT-B1 - SkipDecoder (**Ours**)	230.7	66.1

Table 3
Results and comparison on Apolloscape dataset. The FLOPs are also provided for each model to show the performance/efficiency trade-off.

Group	Classes	MiT-B1	MiT-B0
sky	sky	99.75	99.60
movable object	car, motoricycle, bicycle, person, rider, truck, bus	65.59	63.16
flat	road, sidewalk	87.75	86.67
road obstacles	traffic cone, road pile, fence	45.97	42.90
roadside objects	traffic light, sign, pole, wall, dustbin, billboard	56.38	54.63
building	building, bridge, tunnel, overpass	82.93	81.48
natural	vegetation	95.34	95.03

Table 4
Category/Class IoU of SegFormer MiT-B1 (on the left) and MiT-B0 (on the right) with classic decoder on ApolloScape dataset

6. Inference Time

In this work, the *inference time* application is a crucial aspect, because the real-time requirement of the model should be enough to make the driving automation system able to take the correct decision, e.g. warn the driver to turn in a particular direction to avoid an obstacle that the driver was not able to see in time. The GPU used for inference time tests is a *NVIDIA Tesla V100* with 32GB GDDR6, and the tests are made for two different image sizes (2048x2048, 3480x3480).

	inference time (ms)	
Model type	*2048*	*3480*
MiT-B0	41ms	69ms
MiT-B0 + SkipDecoder (**Ours**)	40ms	68ms
MiT-B1	44ms	72ms
MiT-B1 + SkipDecoder (**Ours**)	41ms	69ms

Table 5
Inference time computed for every different model type (computed in milliseconds).

The FLOPs difference between the model with and without the skip decoder is noticeable, but the inference time difference shown in Table 5 is just slightly lower. Hence, even if the number of floating point operations is lower, the difference is not very significant. Now, it is analyzed the inference time presented above by considering a hypothetical situation for a car going 60km/h (16m/s) in a urban environment. On 2048x2048 images the inference time is better and the model is able to perform at about 25 frames per second and it is potentially capable to give a remarkable support to almost all driving automation levels. At 60km/h, the system is able to give a result every 0.6 meters (every 40ms). And at 100km/h, working at 40ms, the model gives a result every single meter, which is acceptable considering a highly automated driving system that modifies the behavior of the car (such as speed, trajectory etc.) without the human control that would also include human reaction times in the calculation.

7. Conclusions

The work presented herein is a study of real-time semantic segmentation for autonomous driving purposes and new techniques that may or not may be useful for new methodologies. The Skip-decoder SegFormer has been advanced, an efficient model composed by a transformer encoder that manages to be used of difficult segmentation tasks, and a lightweight all-MLP decoder which, in the proposed version, uses a faster and efficient up-sampling technique inspired by U-Net. Finally, this has allowed to find an effective method both from the point of view of metrics and from that of computational resources, balancing these two aspects into a light model powerful enough to be ran in real-time without compromising too much with performance. In addition, such solution can be compared with a variety of techniques in terms of speed and accuracy trade-off, being capable of increase its performance just by modifying some architectures parameters at the expense of efficiency.

Acknowledgments

This work was completed in part with resources provided by the University of Naples "Parthenope", Department of Science and Technologies, Research Computing Facilities (https://rcf.uniparthenope.it)

References

[1] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, P. Luo, Segformer: Simple and efficient design for semantic segmentation with transformers, CoRR abs/2105.15203 (2021). URL: https://arxiv.org/abs/2105.15203. arXiv:2105.15203.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[3] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, R. Yang, The apolloscape dataset for autonomous driving., in: CVPR Workshops, IEEE Computer Society, 2018, pp. 954–960. URL: http://dblp.uni-trier.de/db/conf/cvpr/cvprw2018.html#HuangCGCZWLY18.

[4] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, 2015. URL: https://arxiv.org/abs/1505.04597. doi:10.48550/ARXIV.1505.04597.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, ICLR (2021).

[6] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. H. Tay, J. Feng, S. Yan, Tokens-to-token vit: Training vision transformers from scratch on imagenet, CoRR abs/2101.11986 (2021). URL: https://arxiv.org/abs/2101.11986. arXiv:2101.11986.

[7] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, Y. Qiao, Vision transformer adapter for dense predictions, 2022. URL: https://arxiv.org/abs/2205.08534. doi:10.48550/ARXIV.2205.08534.

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, CoRR abs/2103.14030 (2021). URL: https://arxiv.org/abs/2103.14030. arXiv:2103.14030.

[9] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, L. Zhang, Cvt: Introducing convolutions to vision transformers, CoRR abs/2103.15808 (2021). URL: https://arxiv.org/abs/2103.15808. arXiv:2103.15808.

[10] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, L. Zhang, Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 6877–6886. doi:10.1109/CVPR46437.2021.00681.

[11] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, A. Torralba, Semantic understanding of scenes through the ade20k dataset, International Journal of Computer Vision 127 (2019) 302–321.

[12] H. Caesar, J. R. R. Uijlings, V. Ferrari, Cocostuff: Thing and stuff classes in context, CoRR abs/1612.03716 (2016). URL: http://arxiv.org/abs/1612.03716. arXiv:1612.03716.

[13] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, A. Zisserman, The pascal visual object classes (voc) challenge., Int. J. Comput. Vis. 88 (2010) 303–338. URL: http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[15] I. Loshchilov, F. Hutter, Fixing weight decay regularization in adam., CoRR abs/1711.05101 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1711.html#abs-1711-05101.