Benchmarcking pre-trained Encoders for real-time Semantic Road Scene Segmentation

Lennart Evers\textsuperscript{1,}*  
\textsuperscript{1} Optimization and Optimal Control, Center for Industrial Mathematics (ZeTeM), University Bremen, Bibliothekstraße 5, 28359 Bremen, Germany

Semantic segmentation, i.e. assigning each pixel in an image a class to which it belongs, can be a part of the implementation for the perception model of autonomous vehicles. Over the last years multiple powerful neural network architectures for solving this task have been developed. In this work, a simple lightweight but modular, fully convolutional encoder-decoder network has been implemented and multiple, publicly available pre-trained classification networks are evaluated on their encoder performance on the Cityscapes \cite{8} benchmark dataset with respect to their accuracy and their inference speed.

1 Introduction

A neural network solving the task of semantic image segmentation for \( N_C \) classes is a map \( F: I \rightarrow O \), where \( I := \mathbb{R}^{c \times h \times w} \) is the space of all images with \( c \) channels and spatial dimensions \( h \) and \( w \), while \( O := [0, 1]^{N_C \times h \times w} \) is the output space of all per pixel probability distributions of \( N_C \) classes. The output \( F(x) \) for an image \( x \) is the predicted class distribution for each pixel across spatial dimensions. Since the inception of the field of learned segmentation models, numerous network architectures have been developed to improve the prediction accuracy. If the network can be decomposed to the form \( F = F_d \circ F_e \), where \( F_e: I \rightarrow Z \) is an encoder that maps images to some latent space \( Z \) and \( F_d: Z \rightarrow O \) decodes variables from the latent space into the per pixel distribution of classes, we say \( F \) is an encoder-decoder network.

Many segmentation architectures such as Deeplab \cite{1}, PSPNet \cite{2} and others utilize the convolutional part of popular image classifiers such as resnets \cite{3} or densenets \cite{4} as their encoder. During the training process the encoder stages are often pre-trained, i.e. initialized to weights given by solving a classification task to get a more meaningful starting point for the optimization. In real time applications, the choice of network architecture and especially the choice of encoder, which is often the most computationally expensive part of the model, is constrained by requirements on the inference speed. In the following, an encoder-decoder model for semantic segmentation is introduced and a range of pre-trained encoder networks that are publicly available within the PyTorch \cite{5} model zoo are evaluated on their effect on prediction accuracy and inference speed.

2 Network Architecture

The proposed network architecture combines ideas from the classical U-net \cite{6} approach with the atrous spatial pyramid pooling (ASPP) introduced with Deeplab \cite{1}. An illustration of the architecture can be seen in Fig. 1. After the last encoder layer, the encoder output is passed to the decoder which applies ASPP and upscales using bilinear upsampling. To equalize decoder complexity for arbitrary encoder networks a channel reduction is performed via \( 1 \times 1 \) convolutions. The ASPP output is upsampled to a quarter of the input size and concatenated with the quarter scale encoder features and the upsampled eighth scale encoder features. Final upsampling back to height and width of the original input image and prediction is done via dense upsampling convolution \cite{7}. For a more detailed insight into the architecture and its implementation see \cite{9}.

3 Training and Evaluation

All models are implemented in Pytorch and trained using two RTX2080Tis on the Cityscapes segmentation benchmark dataset. The encoders are initialized with publicly available weights obtained by training classification on the Imagenet dataset. Models

* Corresponding author: levers@uni-bremen.de

\textsuperscript{©} 2021 The Authors Proceedings in Applied Mathematics & Mechanics published by Wiley-VCH GmbH
are trained with a batch size of 14 on square patches with 768px in width, which are randomly sampled from the finely annotated training subset of the dataset. Prior to selecting patches data augmentation is used, e.g. training images are randomly scaled between 50% and 200% in size, randomly mirrored horizontally, altered in histogram and color as well as having gaussian noise applied. To reduce the memory footprint during the learning phase, the networks are trained using half precision floats. We are employing the Adamax optimizer with a base learning rate of 0.004 for the decoder and 0.001 for the encoder. The learning rate is lowered by a factor of 0.4 at epochs 80, 160, 240, and 280 with 300 training epochs in total.

Table 1 shows the results of all trained models with respect to the encoder network used. The model accuracy is measured by the mean intersection over union achieved on the Cityscapes validation dataset, while the inference speed is measured by the achieved frames per second during prediction on 500 images. To indicate model complexity both the number of multiply-add computations (MACs) and the models total number of parameters is given. All numbers are obtained on full resolution input images of size 2048px by 1024px evaluated on a single RTX2080Ti.

Overall, all encoders behave somewhat similarly in accuracy, falling into a three percentage point window between 74.3% and 77.3% mIOU. Computational complexity varies wildly however, with the encoders mobilenet v2, resnet18 and resnet34 achieving real-time capability, i.e. inference faster than 20 fps. Fig. 2 shows the relationship between inference time and prediction accuracy. Modern networks are either very fast or very accurate, with the resnet34 based model falling somewhere in between. The older VGG classifiers fall short in terms of both accuracy as well as speed.

Acknowledgements This project was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 281474342/GRK2224/1.

Open access funding enabled and organized by Projekt DEAL.

References

[1] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4), pp. 834-848, 2018.
[2] H. Zhao, Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, USA, 2017, pp. 2881-2890.
[3] K. He, Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, USA, 2016, pp. 770-778.
[4] G. Huang, Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, USA, 2017, pp. 4700-4708.
[5] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, Automatic differentiation in pytorch, 2017.
[6] O. Ronneberger, International Conference on Medical image computing and computer-assisted intervention, Munich, Germany, pp. 234-241, 2015.
[7] P. Wang, IEEE winter conference on applications of computer vision (WACV), Lake Tahoe, USA, 2018, pp. 1451-1460.
[8] M. Cordts, Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, USA, 2016, pp. 3213-3223.
[9] L. Evers, GitHub repository, 2020, https://github.com/LennartEvers/Ulab.