ProtoSeg: Interpretable Semantic Segmentation with Prototypical Parts – Supplementary Materials

Mikołaj Sacha[1] Dawid Rymarczyk[1,2] Łukasz Struski[1] Jacek Tabor[1]
Bartosz Zieliński[1,3]
[1] Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
[2] Ardigen, Kraków, Poland [3] IDEAS NCBR, Warsaw, Poland
{mikolaj.sacha;dawid.rymarczyk}@doctoral.uj.edu.pl
{lukasz.struski;jacek.tabor;bartosz.zielinski}@uj.edu.pl

A. Examples of interpretable semantic segmentation with ProtoSeg

In Figures 1 to 5 we show some more examples of interpretable semantic segmentation using ProtoSeg.

B. Results on EM Segmentation Dataset using U-Net backbone

B.1. Experimental setup

For the EM Segmentation Dataset, we train U-Net as well as ProtoSeg with U-Net as the backbone instead of DeepLab. We use the same model architecture as in the original U-Net, training and evaluating on full 512x512 images. For the U-Net-backed ProtoSeg, we set the output number of classes in U-Net to 64 and use these output features to compare each pixel to one of 20 prototypes (10 per class) of size 64. We also employ the prototype diversity loss with $\lambda_{\text{J}} = 0.25$. For both models, we follow the following training procedure: we randomly initialize all model weights except w_h, skip the *warmup* phase and run joint training phase for 10000 batches of size 2. During *joint* training phase, we use a polynomial learning rate schedule with $power = 0.9$, starting from learning rate equal to 10^{-4}. In both fine-tuning phases, we use a constant learning rate equal to 10^{-5} and train for 10000 batches of size 2. We employ data augmentations during training, such as random rotation, scaling and gaussian blur. The training procedure takes less then 5 hours on a single NVidia GeForce RTX 2080 GPU.

B.2. ProtoSeg Segmentation results

We show the sample results for segmentation on the EM Segmentation Dataset with the U-Net backbone in Figures 6 and 7

Proto per class	Proto vector length	*joint* training length	mIOU
10	64	1x	72.05
10	128	1x	67.13
10	256	1x	65.79
4	64	1x	65.04
20	64	1x	**72.76**
50	64	1x	72.67
10	64	2x	71.63

Table 1: Model accuracy on Pascal VOC 2012 validation set for different training hyperparameter values, with ImageNet pretraining. The model accuracy can be slightly improved by increasing the number of prototypes per class.

C. Additional ablation study on Pascal VOC 2012

In this section we present the results of an ablation study of different hyperparameter values for ProtoSeg, trained on Pascal VOC 2012 with DeepLab backbone and ImageNet pretraining. We present mIOU scores of some hyperparameter variants in Table 1. The variants include different number of prototypes per class, different prototype vector length and 2 times longer *joint* training than the one described in the main text. Accuracy was measured for models after the final *pruning* phase. We see that the accuracy can be slightly improved by increasing the number of prototypes. However, making ProtoSeg as accurate as the baseline model remains challenging.

D. Discussion

In this section we discuss in more details some of the rationale behind ProtoSeg.

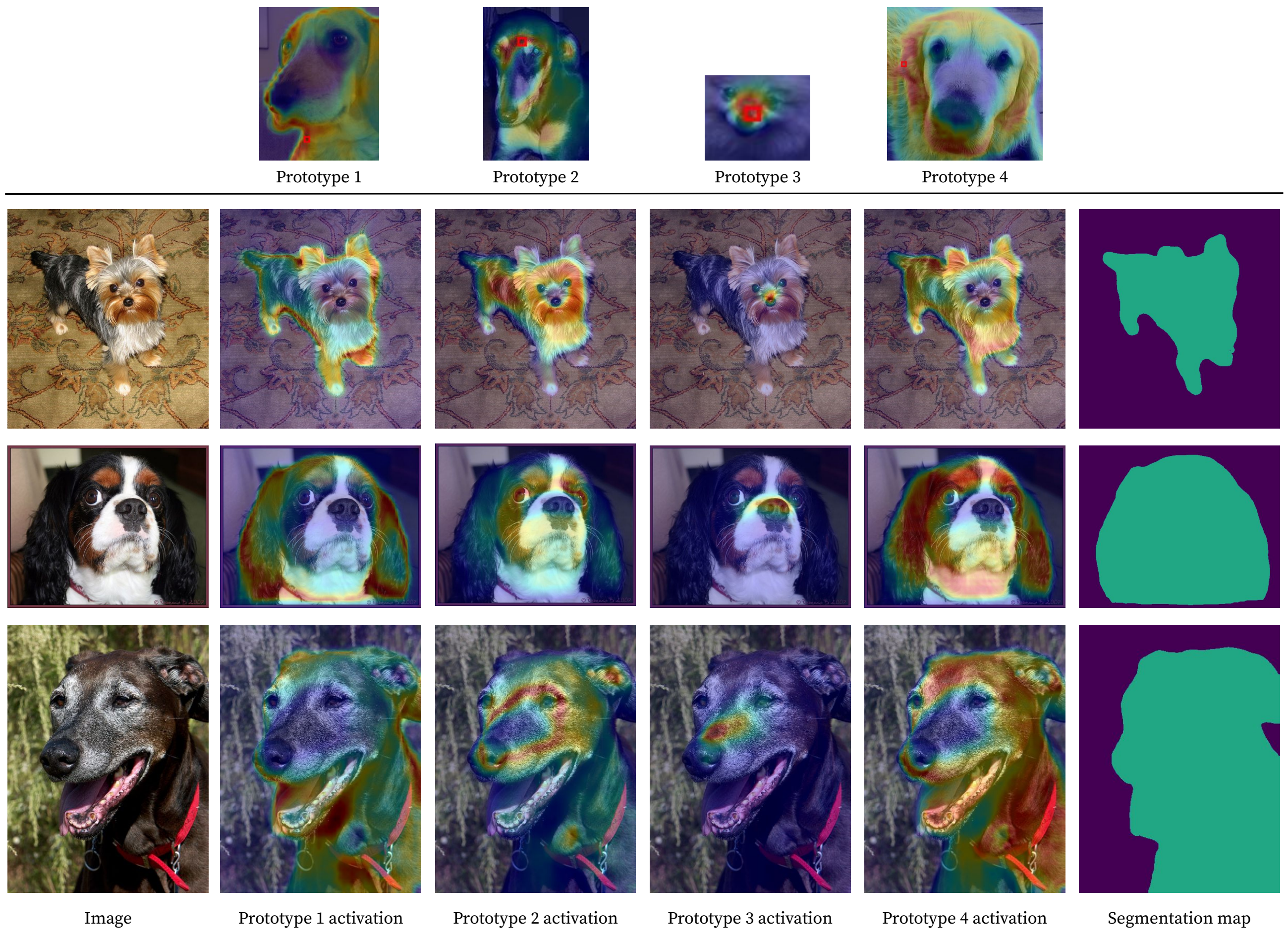

Figure 1: Prototype activation maps generated by ProtoSeg for four prototypes from class *dog* (columns) and three sample images from PASCAL VOC 2012 (rows). We see that prototypes concentrate their activations on different semantic concepts, such as dog's nose.

Why the original ProtoPNet is not directly applicable to segmentation? ProtoSeg is a model suitable for segmentation tasks, inspired by ProtoPNet. However, ProtoPNet is a model addressing fine-grained image classification and cannot serve directly as a backbone for the segmentation task. Here, we integrate the prototypical layer with a segmentation convolutional network, such as DeepLab. Straightforward integration would assume each pixel has a classifier consisting of a prototypical layer and the last layer from ProtoPNet. This type of integration would result in very inefficient training and inference times. For this reason, we propose an architecture that fuses the prototypical layer with DeepLab in a way that does not increase the time needed to train a model or obtain the prediction. We also propose a regularization function which encourages the model to find a diverse set of prototypes for a given class. This way, we make the prototypical representation of an object richer in interpretable features.

What is the purpose of the Prototypical Layer? The goal of this work is to make a step towards more explainable image segmentation, which requires comprehending the nature of the segmented object. Prototypes make the model more transparent by presenting its predictions interpretably (as a reference to the training set examples). As segmentation is a common task in multiple fields, such as medical imaging and autonomous driving, it is important to interpret the model predictions. When the model presents the prediction rationale to an expert in the field, the incorrect decision can be avoided, such as misdiagnosis when the doctor sees a faulty explanation.

Do prototypes always catch semantically meaningful features? ProtoSeg does not base explanations on prede-

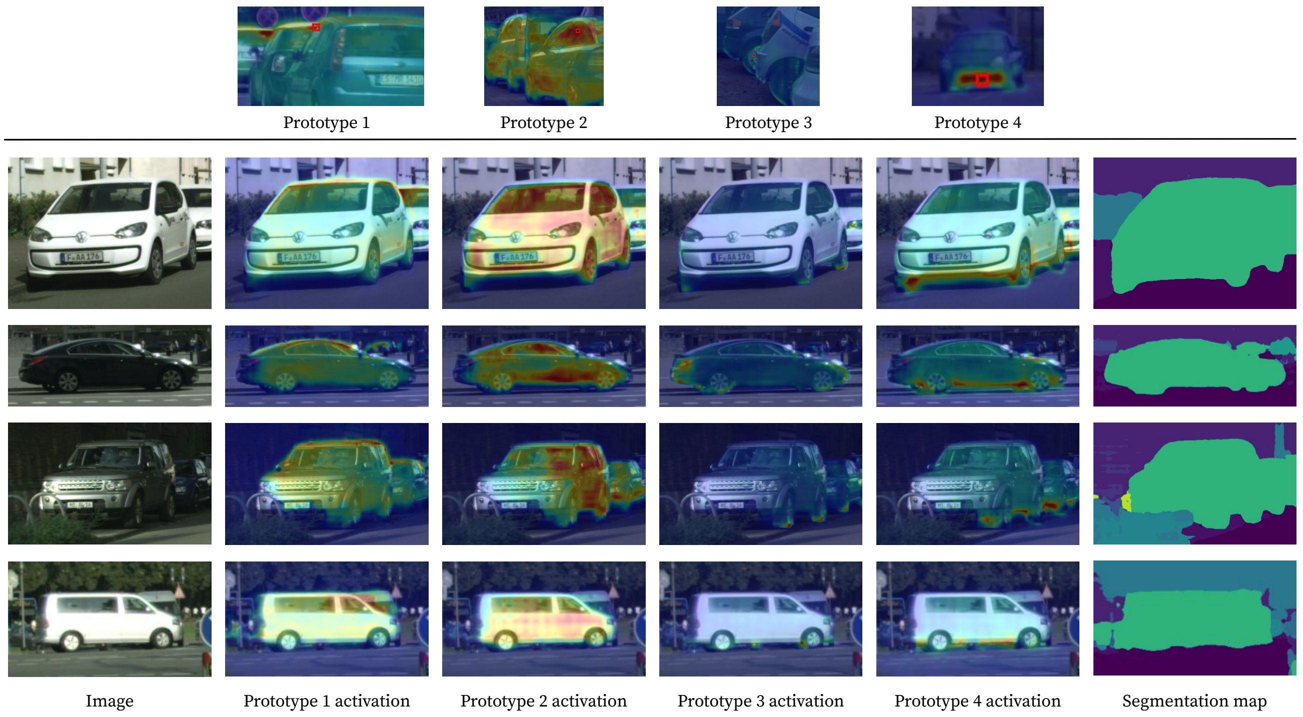

Figure 2: Prototype activation maps generated by ProtoSeg for four prototypes from class *car* (columns) and four sample images from Cityscapes (rows). The original images were cropped for the purpose of visualization. We see that prototypes concentrate their activations on different semantic concepts, such as car tyres or chassis.

fined dictionaries, nor does it use any human labeling of prototypes. Because of this, the model can learn both low and high-level object features. Figure 2 in the main text shows that ProtoSeg learns both low-level features, such as object boundaries, and semantic features, such as object parts (e.g., a cat's nose). We discover that pretraining on a large-scale image classification task, such as ImageNet, helps the model to focus more on semantically meaningful features.

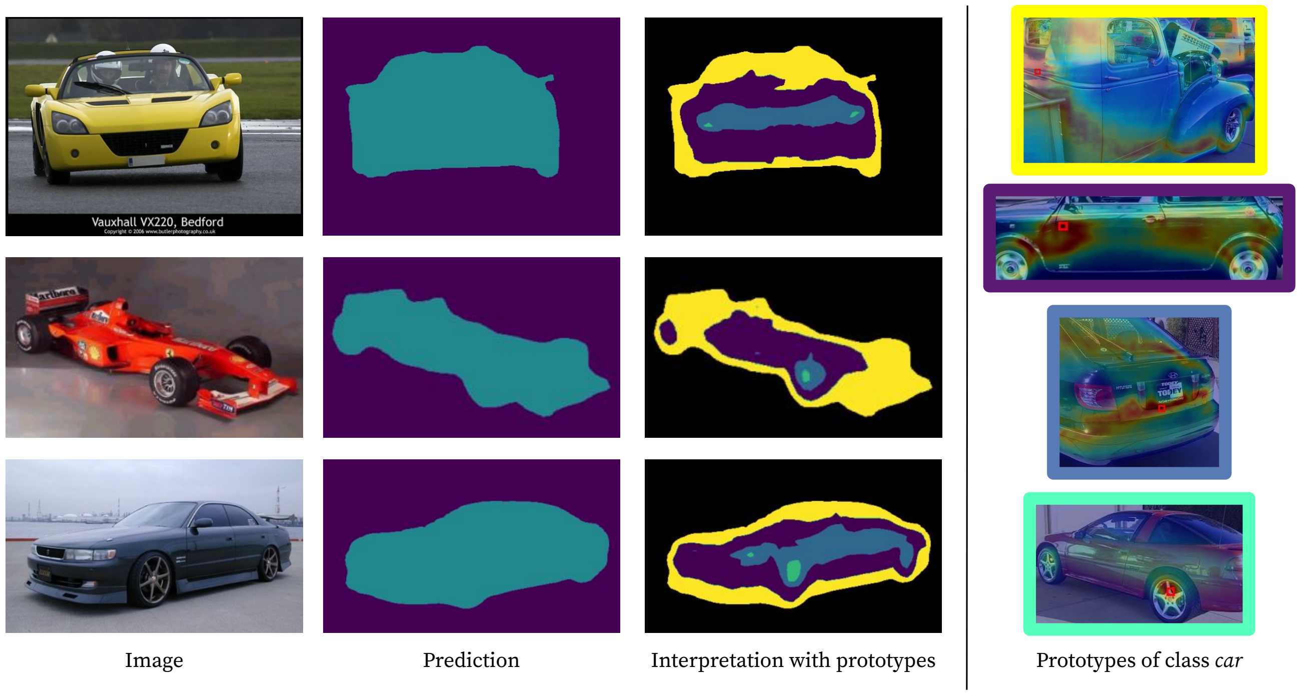

Figure 3: Sample segmentation (second column) of images (first column) and interpretation with four prototypes (third column) from class *car* obtained by ProtoSeg on PASCAL VOC 2012. Interpretation with prototypes is acquired by assigning the prototype with the maximal activation to a considered pixel. Pictures in the right column show the four prototypes (activating on car tyres, bumper or other parts), and their frame colors correspond to the colors from the third column.

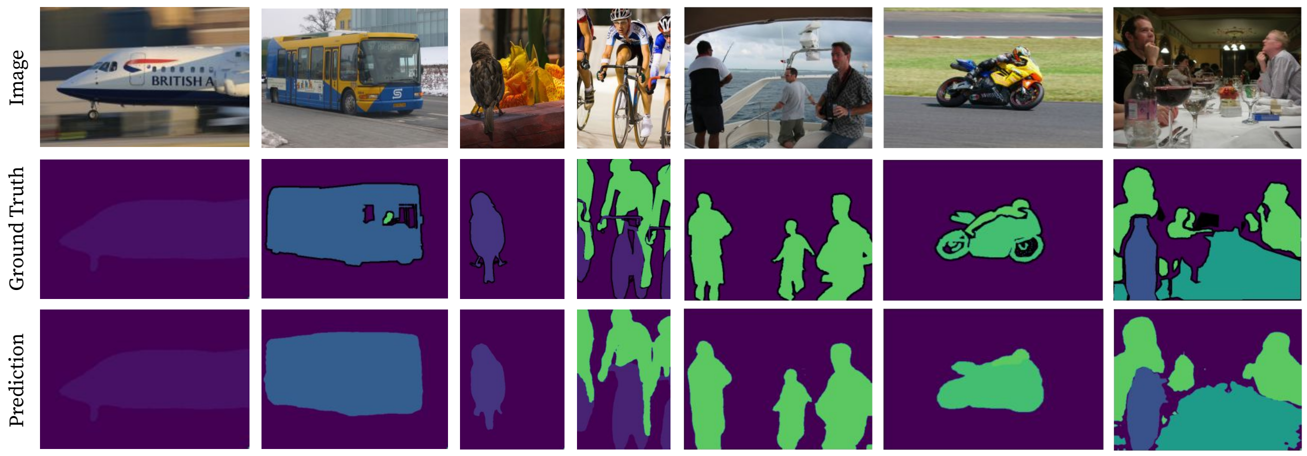

Figure 4: Sample ProtoSeg segmentations on PASCAL VOC 2012. Pixels not considered in the evaluation are masked with black color in ground truth images.

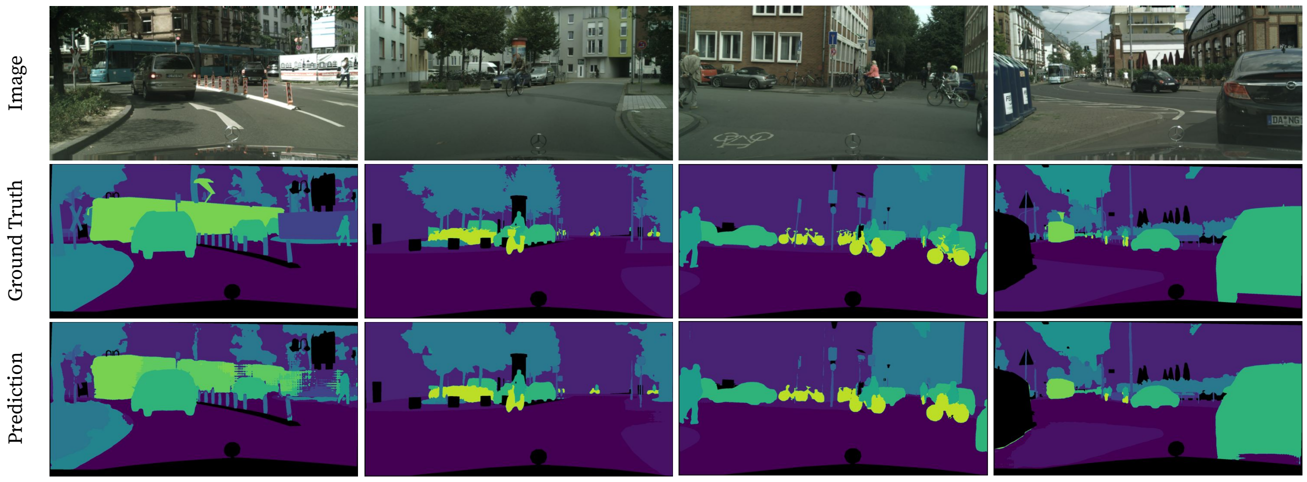

Figure 5: Sample ProtoSeg segmentations on Cityscapes. Pixels not considered in the evaluation are masked with black color in ground truth and prediction images.

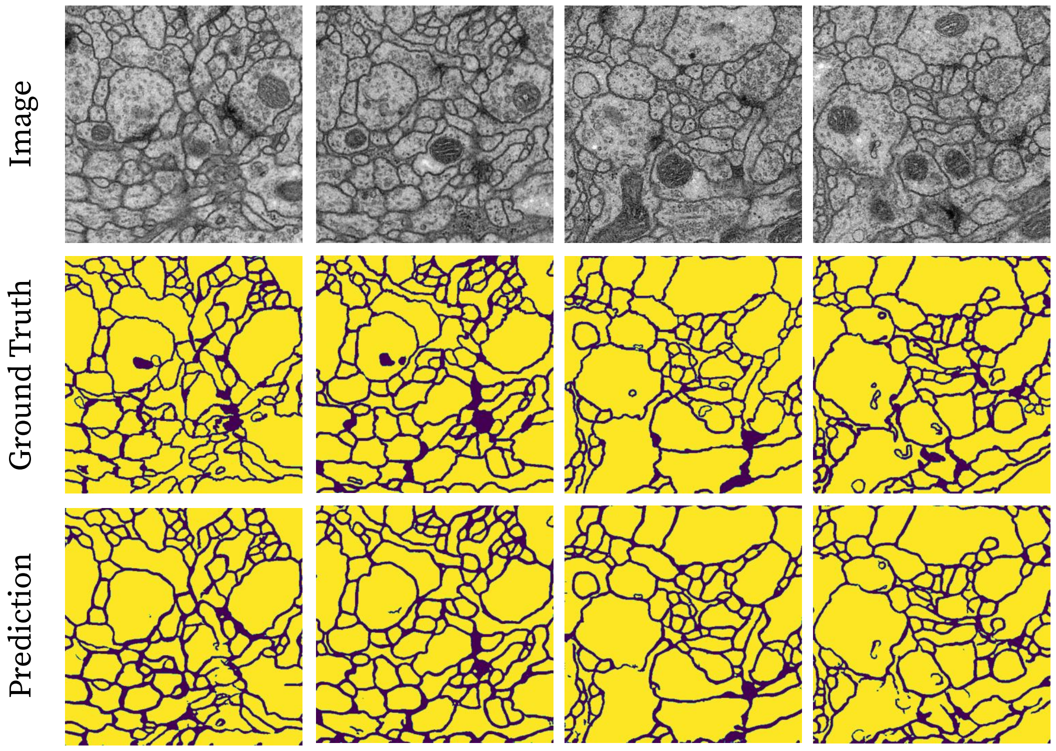

Figure 6: Sample ProtoSeg segmentations on images from EM segmentation challenge dataset, using U-Net as the model backbone

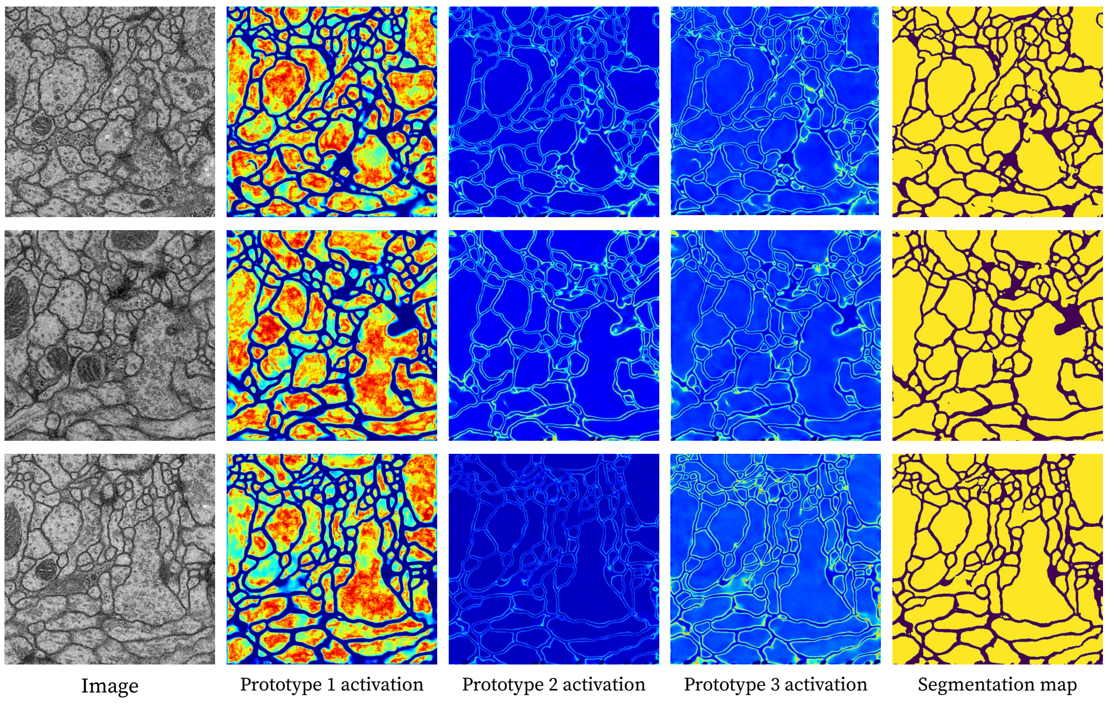

Figure 7: Prototype activations of 3 selected prototypes of ProtoSeg with U-Net as the backbone, on selected images from EM segmentation challenge dataset.