IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2023–06–07.
Submitted on 2022–11–24, accepted on 2022–11–25.
ISSN 2105–1232 © 2023 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2023.447

Semantic Segmentation: A Zoology of Deep Architectures

Aitor Artola

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
aitor.artola@ens-paris-saclay.fr

Communicated by Gabriele Facciolo *Demo edited by* Aitor Artola

Abstract

In this paper we review the evolution of deep architectures for semantic segmentation. The first successful model was fully convolutional network (FCN) published in CVPR in 2015. Since then, the subject has become very popular and many methods have been published, mainly proposing improvements of FCN. We describe in detail the Pyramid Scene Parsing Network (PSPnet) and DeepLabV3, in addition to FCN, which provide a multi-scale description and increase the resolution of segmentation. In recent years, convolutional architectures have reached a bottleneck and have been surpassed by transformers from natural language processing (NLP), even though these models are generally larger and slower. We have chosen to discuss about the Segmentation Transformer (SETR), a first architecture with a transformer backbone. We also discuss SegFormer, that includes a multi-scale interpretation and tricks to decrease the size and inference time of the network. The networks presented in the demo come from the MM-Segmentation library, an open source semantic segmentation toolbox based on PyTorch. We propose to compare these methods qualitatively on individual images, and not on global metrics on databases as is usually the case. We compare these architectures on images outside of their training set. We also invite the readers to make their own comparison and derive their own conclusions.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README` file of the archive. The original versions of the source codes used by the online demo are available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: deep learning; semantic segmentation; CNN; Transformer

[1] https://doi.org/10.5201/ipol.2023.447
[2] https://github.com/open-mmlab/mmsegmentation

1 Introduction

Semantic segmentation consists in a classification at the pixel level of what composes the scene of an image. Since a mathematical modeling of these semantic classes is an arduous task, a solution is to learn on a training base filters that capture the meaning of the image and that are able to reproduce this segmentation on new data outside this training set. This domain has been growing rapidly in the last few years due to the progress in deep learning and the emergence of databases with rich and varied annotations.

The first successful architectures for semantic segmentation derived from image classification architectures. These are convolutional neural networks (CNN) that convolve the image with learned kernels and activation layers. These convolutions create a feature image that describes an area of the input image. The deep layers usually describe areas larger than the input layers and are often downsampled for efficiency. In classification, this descriptor image is then processed and produces probabilities for each class at an image level. In segmentation, this descriptor image is upsampled so that the prediction is performed at a pixel level.

CNN-based methods often use a classification backbone like Resnet [5]. This backbone produces the concentrated descriptors of the image to be processed to assign each area to a class. These methods, such as FCN [8], PSPNet [11] or DeepLabV3 [1], were until recently, the state of the art in semantic segmentation.

The problem with most classic CNNs is that the scale of analysis is proportional to the number of layers, thus leading to heavy networks. A solution to this problem is downsampling. However, this solution decreases the resolution of the analysis. Other solutions such as atrous convolutions can also mitigate this problem.

The self-attention mechanism [9] used in transformer blocks, inherited from natural language processing (NLP), allows for a global analysis of the image in a single layer. Transformer based methods now dominate the state of the art in semantic segmentation. Evolution in the architecture of the transformer block enabled to reach a new level of performance. We can mention the SWin Transformer [7], which applies the attention mechanics on sliding windows instead of the whole image, leading to performance gains in detection quality and in computation time.

2 Convolutional Neural Networks (CNN)

2.1 Fully Convolutional Network (FCN)[8]

A backbone first produces dense semantic descriptors of the image. These descriptors are then convolved to predict scores per classes and then upsampled to the pixel level. FCN [8], described in Figure 1, uses a Resnet [5] backbone.

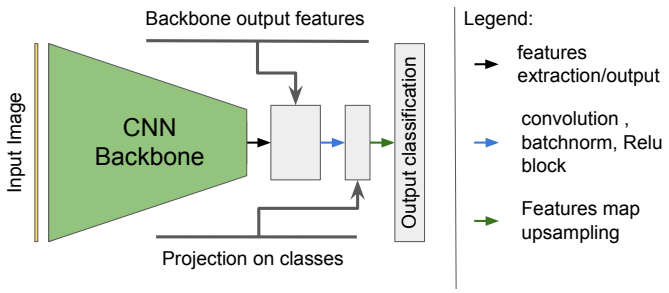

Figure 1: Description of FCN [8]. An image is given to a backbone (in green) to generate semantic features of lower spatial resolution. These features are then convolved to predict a score per class and upsampled to the pixel level.

2.2 Pyramid Scene Parsing Network (PSPnet) [11]

The main problem with FCN is that the output features of Resnet represent a single scale of the image. This can cause problems in getting a global context. PSPnet [11] proposes an additional head to refine its features.

The idea is to make several pooling of these output features to merge them locally. These merged features form a pooling pyramid as shown in red in Figure 2. Pooling allows to aggregate a patch of features by average or max, therefore increasing the network's receptive field and improving robustness to translation. Each pooled feature of the pyramid is convolved and upsampled to be stacked and merged with the backbone features. The goal of this stack of features is to aggregate local information of the resnet output and a multi-scale information of the scene.

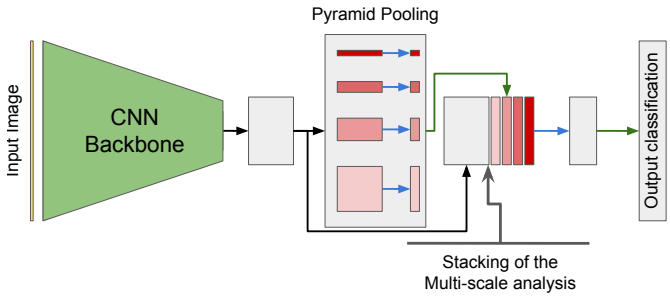

Figure 2: PSPnet [11] principle. The improvement compared to FCN [8] is the pyramid pooling (in red) performing a multi-scale analysis. Each shade of red corresponds to a different pooling size and therefore a different scale. These scales are then upsampled and stacked with the features to compute the class score. See Figure 1 for the signification of the different arrows.

2.3 DeepLabV3 and DeepLabV3+ [1]

Deeplabv3 [1] adapts backbones used in segmentation methods in a way that avoids the resolution loss caused by striding. Convolutions with a large striding are replaced by dilated convolutions with an atrous rate equal to the striding it replaces. Following convolutions are modified to take into account the removal of the striding. This means that the receptive field of the network stays the same.

Dilated convolutions are convolutions with holes in their kernel. They are defined as

$$f(x,y) = \sum_k \sum_l \sum_m I(x+rk, y+rl, m) W(k,l,m) + B(m), \tag{1}$$

where I is the image, (x,y) is the spatial position where the convolution is evaluated, (W,B) is the weights of the kernel and r is the atrous rate. The case $r=1$ corresponds to a normal convolution.

Similarly to PSPnet, a pyramid head, Atrous Spatial Pyramid Pooling (ASPP) in red in Figure 3, is used to perform a multi-scale analysis. This module is composed of several parallel convolutions with different atrous rates that act as a multi-scale analysis of the features. The use of atrous convolutions to replace the pooling of PSPnet allows the network to have a pyramid that does not decrease the spatial dimension. These features are then concatenated and merged with another convolution to give a compact multi-scale feature map.

The improvement of deeplabv3+ over deeplabv3 comes from the use of intermediate backbone features. Intermediate features with higher resolution are extracted from the backbone, in blue in Figure 3. They are set to the same size, concatenated and merged, and then concatenated with the features from the ASPP module and merged again. This final descriptor map is upsampled to be used

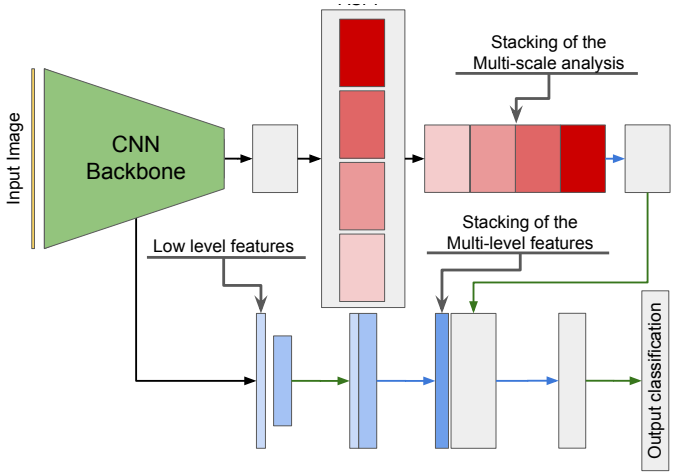

Figure 3: Description of Deeplabv3+ [1]. The atrous pyramid pooling (in red) allows for a multi-scale analysis without loss of resolution. Each shade of red corresponds to a different atrous rate. These scales are then upsampled and stacked with the intermediate features (in blue) of the backbone to give a segmentation with a better resolution. See Figure 1 for the signification of the different arrows.

for segmentation. The use of these intermediate layers allows to increase the quality of segmentation on the details.

3 Transformers

The problem with CNNs is that a convolution is a local analysis layer. This means that in order to have a global description of the image, it is necessary to increase the number of convolutions or use striding so that the receptive filter of the network is at the scale of the image.

Inherited from NLP, the transformer block [9] allows for a global analysis of the image in one single layer, i.e. all pixels of the input image are involved in the creation of a feature.

3.1 The Visual Transformer (ViT) Block [3]

We first describe the ViT [3] encoder block before presenting possible modifications. In the context of NLP, words of a text are vectorized and then given to the transformer. In image processing, the concept of words is replaced by image patches. An image $I \in \mathbb{R}^{H\times W\times 3}$ is cut into several patches of dimension $d \times d \times 3 = D$. Before being used in transformers blocks, patches go through several embedding steps. The first one is patch embedding: the patches $x \in \mathbb{R}^D$ are flattened and projected onto another space with a linear layer $W_e \in \mathbb{R}^{D\times D}$. Then a trainable embedding patch $z_{\text{class}} \in \mathbb{R}^D$ is concatenated at the beginning of the patch sequence. The last embedding is a position embedding: a learnable position vector $b_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$ is added to the patches sequence. These steps form the initial vectors sequence z_0 as described below

$$z_0 = (z_{\text{class}}, W_e x_1, \dots, W_e x_N) + b_{\text{pos}}. \tag{2}$$

ViT aggregates several transformer blocks in a sequential manner with the k^{th} transformer encoding z_{k-1} into z_k. A Transformer block is composed of a multi-head self-attention layer (MSA) and a feed forward network (FFN) with normalization layers (NL), and residual connection such that

$$\begin{aligned} \hat{z}_k &= \text{MSA}_k(\text{NL}(z_{k-1})) + z_{k-1}, \\ z_k &= \text{FFN}_k(\text{NL}(\hat{z}_k)) + \hat{z}_k. \end{aligned} \tag{3}$$

In a self-attention layer (SA), each patch is first transformed into a query, a key and a value $(q, k, v) \in \mathbb{R}^{N \times 3C_{SA}}$ thanks to a linear operation of weight $U_{qkv} \in \mathbb{R}^{C \times 3C_{SA}}$. The keys and queries of all patches are then compared one by one and given to a softmax to produce an attention array $A \in \mathbb{R}^{N \times N}$, which is used as a weighting to merge the values v. This corresponds to

$$\begin{aligned} (q, k, v) &= zU_{qkv}, \\ A &= \text{softmax}(\tfrac{qk^T}{\sqrt{C_{SA}}}), \\ SA(z) &= Av. \end{aligned} \tag{4}$$

Equation (4) describes the operation for a single head. This operation is performed in parallel on n_h independent heads and aggregated by a linear operation with the weight $U_{MH} \in \mathbb{R}^{n_h C_{SA} \times C}$.

$$MSA(z) = [SA_1(z), \ldots, SA_{n_h}(z)]U_{MH}. \tag{5}$$

The FFN is composed of series of linear layers followed by activation layers. In practice, it is often two linear layers with an activation layer in between like Equation (6), where $W_1 \in \mathbb{R}^{C \times \gamma C}$ and $b_1 \in \mathbb{R}^{\gamma C}$ are the weights and bias of the first linear layer and $W_2 \in \mathbb{R}^{\gamma C \times C}$ and $b_2 \in \mathbb{R}^C$ are those of the second one. The parameter $\gamma > 1$ is the expansion rate of the FFN. That is,

$$FFN(z) = \max(0, zW_1 + b_1)W_2 + b_2 \tag{6}$$

The first advantage of ViT compared to CNN is the non-local analysis mechanism, that requires only one block. However, it is often more expensive in parameters and computation time because it implies using the entire image to produce the features of each patch. While this architecture avoids downsampling to be global, it does not benefit from the computational cost reduction induced by the downsampling.

3.2 Segmentation Transformer (SETR) [12]

SETR [12] is one of the first segmentation architectures to use mainly transformers. Previously, transformers were mostly used as heads in CNN/transformer hybrid methods. SETR, presented in Figure 4, uses ViT as backbone.

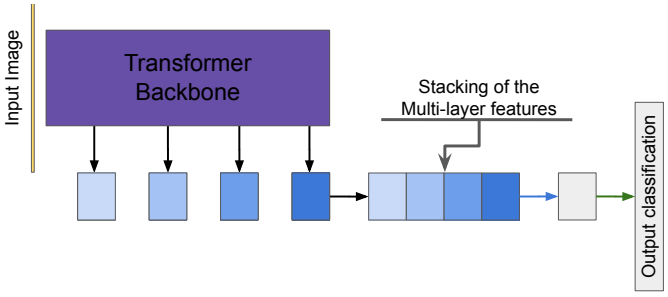

Figure 4: Description of SETR [12]. A ViT backbone [3] (in purple) is composed of several transformer blocks. Features (in blue) are extracted from several intermediate blocks, stacked and convolved to get the scores of each class. See Figure 1 for the signification of the different arrows.

3.3 Segformer [10]

SegFormer [10], shown in Figure 5, proposes a modification of the backbone of SETR to address various issues. It includes a hierarchical mechanism and downsampling like in CNNs. Patch embedding is done by convolutions with overlap. The passage from one scale to another is done by a new embedding with a new convolution that decreases the spatial size of the features map.

The first embedding of the image $I \in \mathbb{R}^{H\times W\times 3}$ is performed using a convolution layer with kernels of size 7×7 and striding of 4. This gives an embedding $z_0 \in \mathbb{R}^{H/4\times W/4\times C_1}$ of overlapping patches, with c_1 the number of channels. The next embedding works in the same way as the initial embedding. The following embeddings are done with a 3×3 convolution and a downsampling with a stride of 2. This convolution reduces the spatial resolution but increases the number of features, similarly to CNN encoders.

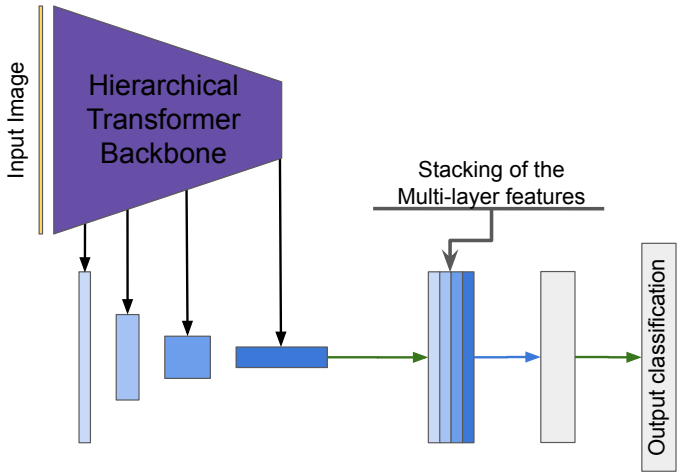

Figure 5: Description of SegFormer [10]. In Segformer, the transformer backbone is multi-scale. Features (in blue) are extracted from several intermediate blocks, stacked and convolved to get the scores of each class. See Figure 1 for the signification of the different arrows.

Absolute position encoding is problematic because images do not have a fixed size for inference. It is better to use the relative patches position. This is done here by modifying FFN and by merging neighboring patches with a 3×3 convolution before the non linearity..

The MSA operation is computationally expensive. For $q, k, v \in \mathbb{R}^{H\times W\times 3C_{SA}}$ the complexity is in $O((HW)^2)$. To reduce the complexity, MSA is modified so that the keys and values are downsampled by a factor R by applying a convolution of size $R \times R$ with a striding of R while keeping the number of channels unchanged. At the output, as the query stays the same and the resolution remains unchanged, the complexity becomes $O((HW/R)^2)$.

At each scale k, the modified transformer block is applied n_k times before re-embedding. Each scale returns an encoding of the image that is then upsampled and merged using a final convolution with a 1×1 kernel size to produce features of dimension C. These features are then provided to a softmax layer to generate class scores.

4 Demo's Architecture Details

Implementations of these architectures is available in the demo. It comes from the MM-Segmentation library [2], which is an open source semantic segmentation toolbox based on PyTorch. It is a part of the OpenMMLab project. Implementation details are given in Table 1.

All the models are trained on the same dataset with the same classes. The dataset that was used is ADE20K [13], a regular image dataset for semantic segmentation. These images are fully annotated with 150 classes and split into 20K training images and 2K images for validation. Training images are randomly cropped to 512×512 to create batches. The objective of the training is to minimize the cross-entropy between the per-pixel model prediction and the ground truth.

All CNNs presented in the demo use Resnet-101 as backbone, with an output stride of 8. SETR uses ViT-L as a backbone and SegFormer a MiT-B4 backbone [4].

Models	Backbone	BackBone Out Stride	mIoU	Mem (GB)	Param Mem(MB)	fps
FCN	Resnet-101	8	39.91	12	268	14.78
PSPnet	Resnet-101	8	44.39	12	266	15.3
DeepLabV3+	Resnet-101	8	45.47	14.1	333	14.16
SETR	ViT-L	1	48.28	18.4	1193	4.72
SegFormer	MiT-B4	32	48.46	6.1	240	15.45

Table 1: Segmentation models implementation details and training results. The mIoU are also given by the documentation of the library of MM-Segmentation [2].

Table 1 evaluates the models according to their mean Intersection over Union (mIoU) whose formula for one class i is $IoU_i = \frac{TP_i}{TP_i+FP_i+FN_i}$. The IoU of a class is defined as the number of true positives (intersection of detected and actual positives) over the number of true positives plus the number of false positives, plus the number of false negatives (union of detected and actual positives). This metric is one of the most popular to measure the quality of semantic segmentation.

Transformer methods have a better mIoU than CNN methods explaining why they have replaced them. Table 1 also presents other indicators than the detection quality: it gives the fps of the model, the memory used during inference and the number of parameters.

5 Experiments

The overall statistical results on the test base show that transformers perform better. In this part of the experimentation we analyze some qualitative example. We encourage the users to do the same using the demo and derive their own conclusions.

We also show the entropy of predictions alongside the segmentation results. Since the prediction of the network is probabilities for each class, the entropy of the prediction can be estimated

$$H(X) = \mathbb{E}[-\log(p(X))] = \sum_{c=1}^{C} p(x_c), \log(p(x_c)), \tag{7}$$

with $X = (x_c)_{c\in\{1,\dots,C\}}$ the prediction of the network and C the number of classes. The entropy represents the confidence of the network: a low entropy means that the model has chosen a class with a high confidence. This is particularly interesting to analyze failures, since it allows to discriminate between low confidence error and high confidence errors that might be problematic.

5.1 Architectures Comparison

We first evaluate the different methods on an example from the ADE20K test base. Figures 6 and 7 show an image of a house and the segmentation by the five methods alongside their entropy. We notice at first that the entropy is always high at the edge between two classes. This is to be expected because they are transition zones, and thus are difficult to assign.

All the architectures classify well the sky and grass with low entropy. There are however other areas that are more problematic: the house has a high entropy for all three CNN based networks. Moreover FCN has classified half of it as a “building” while Transformers classifies it as a “house” with high confidence. This show that FCN can confuse semantically close classes.

Figure 6: Image of a house from ADE20K use to evaluate the five networks in the Figure 7.

All methods seem to also struggle with the small path in the bottom right of the image. Networks are uncertain whether it is a path, a road or a sidewalk. While these are still semantically similar classes and thus are not major mistakes, we can note that CNNs are more likely to predict many small areas with similar classes whereas transformers are more likely to produce a single large prediction. This can also be seen in the middle-left region of the image with the trees and bushes: CNNs struggle to delimit the tree from the plants.

It seems that transformers make a more global decision, leading to more uniform predictions. However, they might miss some more important local details. For example, only the CNNs manage to see the staircase at the entrance of the house. This staircase being of the same material and texture than the path, it might have influenced the transformers to think that both are the same objects.

Overall, all methods work well despite some confusion between similar concepts. On the resolution of the segmentation, the chimneys of the house are detected as sky, which is probably due to the low spatial resolution of these architectures. We notice however that DeeplabV3+, one of the methods that gives more importance to the resolution and details, delimits the transition between the bushes and the wall in the center left better than the other methods.

5.2 Training Biases

Regardless the architecture, these methods are very dependent on the training database. We tested images from COCO [6], another natural image dataset to verify the generalization capabilities of the networks.

First, it is difficult to fool the network on classes that it has seen in large quantities during its training. It generalizes rather well to people, vehicles, buildings and landscapes. So we chose the class animals of which there are only 533 examples in ADE20K. They are often a small part of a landscape with a wide angle of view. We can find many examples of various animals in the COCO database which are often the main element of the image.

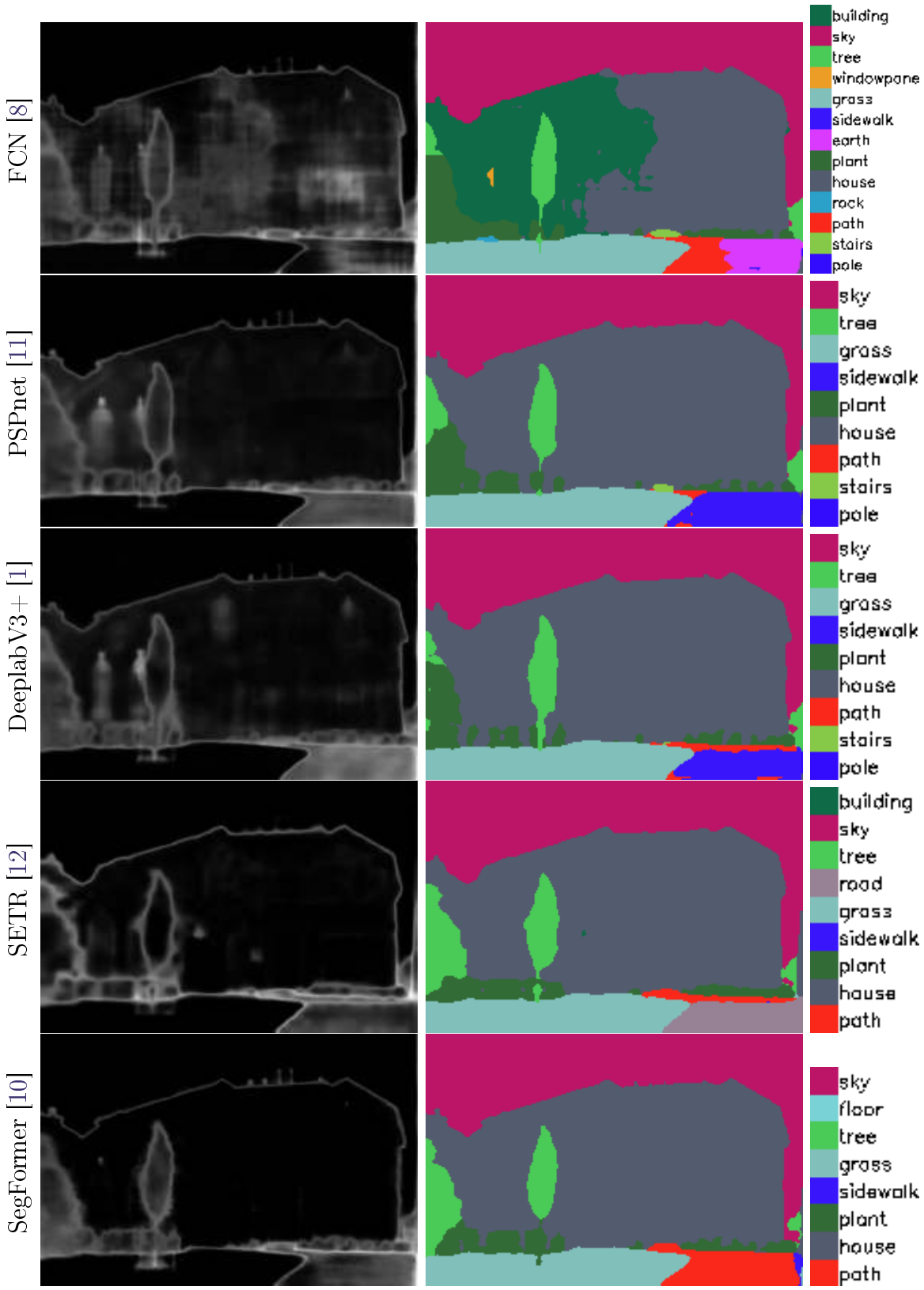

Figure 7: Segmentation predictions using the five architectures on a house image from the ADE20K validation set. The input image is shown in Figure 6. Each row represents a different network. The left column gives the entropy of the detection (a white pixel representing a high entropy), while the right column corresponds to the segmentation alongside the name and color of the detected classes.

Figure 8 shows two different images of sheeps from COCO. The first one shows them from far away in a landscape, while in the second one they are much closer. The results of semantic segmentation for the five networks are shown in Figures 10 and 11. In both cases, the networks seem to segment the background and the landscape well. The transitions between classes are less successful in these examples, where we see that some of them overstep. The results are more mixed with the sheeps, that the CNNs confuse with people in the first image and with water in the second. Segformer also makes mistakes on the animals in the second image. The results on SETR are more impressive because it detects the animals well, without making big mistakes, while it has seen a small number of sheeps in training and rarely zoomed in so much.

Figure 8: Two Images of sheeps from COCO, the one on the left is used in Figure 10, and the one on the right is used in Figure 11.

Figure 9: Two Images of animals from COCO, the one on the left is a cat used in Figure 12, and the one on the right is a bird used in Figure 13.

The errors become much more extreme when we take rarer animals in ADE20K, photographed from very close. Figures 9, 12 and 13 show a cat head and a bird from COCO dataset, and the results of their segmentation with the five networks. The transformers manage to detect a part of the bird as an animal, where the CNNs see mainly a mountain. The cat is much more complicated, probably because it is not entirely in the picture. Only SETR shows animal label on a few pixels, no network can really find the animal.

Entropy is rarely low where there are errors. The exception is FCN, which gives high confidence to bad labels where it should detect animals. In general the networks are less confident on COCO images, even on good classifications, which shows a difficulty to generalize. Transformers are more

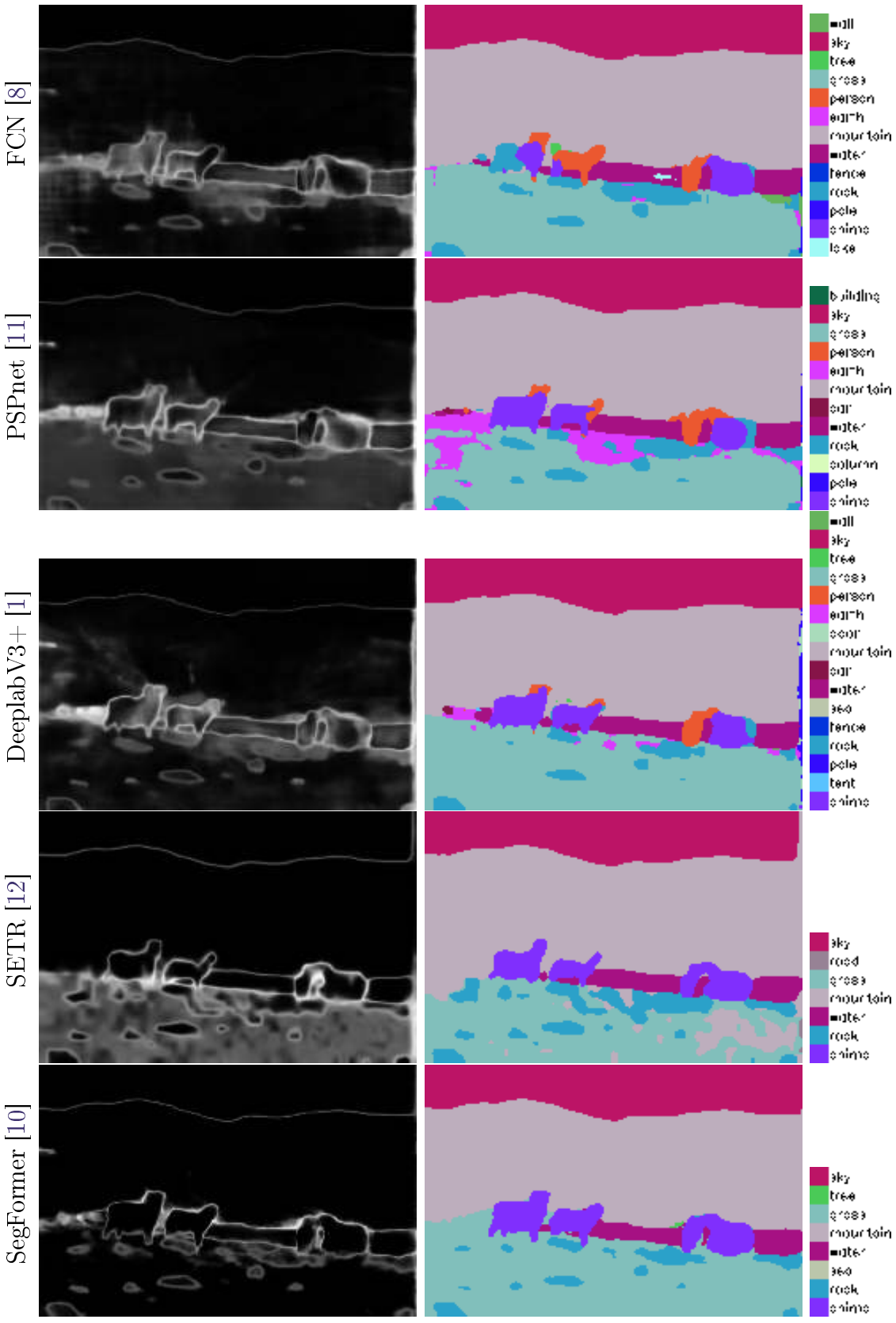

Figure 10: Segmentation predictions using the five architectures on a landscape with sheeps from the COCO validation set. The input image is shown in Figure 8-left. Each row represents a different network. The left column gives the entropy of the detection (a white pixel representing a high entropy), while the right column corresponds to the segmentation alongside the name and color of the detected classes.

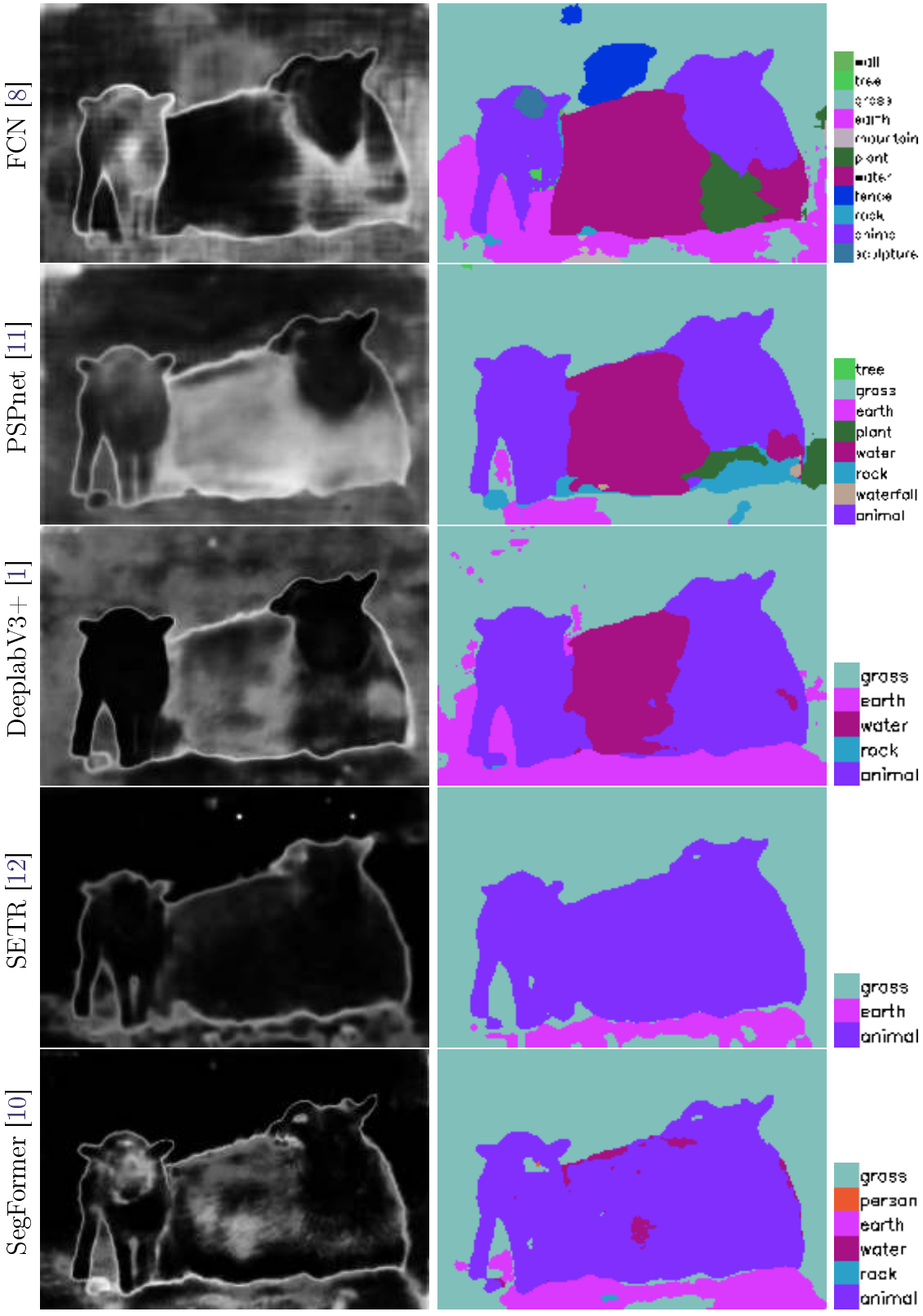

Figure 11: Segmentation predictions using the five architectures on an image of two sheeps from the COCO validation set. The input image is shown in Figure 8-right. Each line represents a different network. The left column gives the entropy of the detection (a white pixel representing a high entropy), while the right column corresponds to the segmentation alongside the name and color of the detected classes.

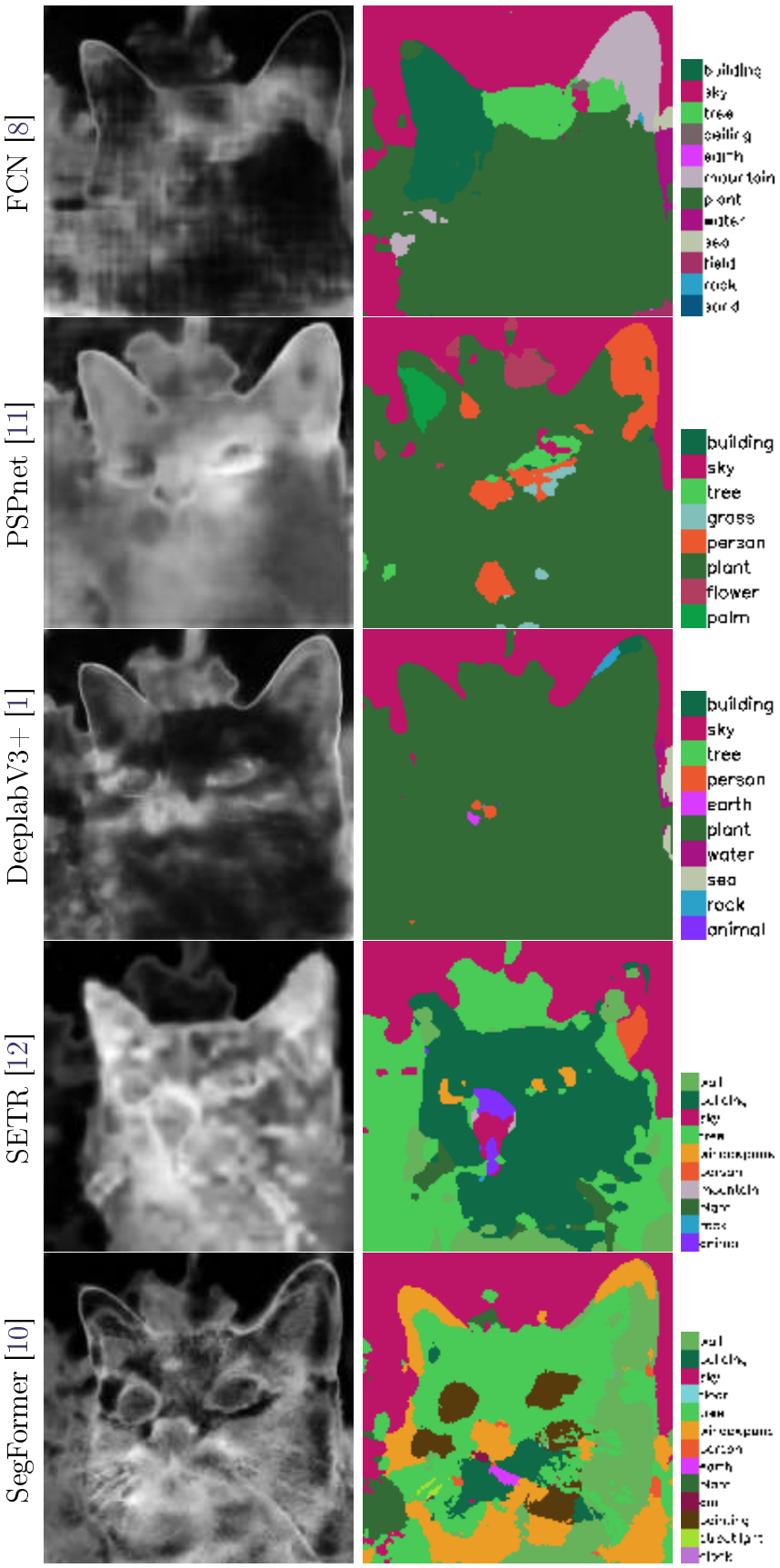

Figure 12: Segmentation predictions using the five architectures on a cat image from the ADE20K validation set. The input image is shown in Figure 9. Each row represents a different network. The left column gives the entropy of the detection (a white pixel representing a high entropy), while the right column corresponds to the segmentation alongside the name and color of the detected classes.

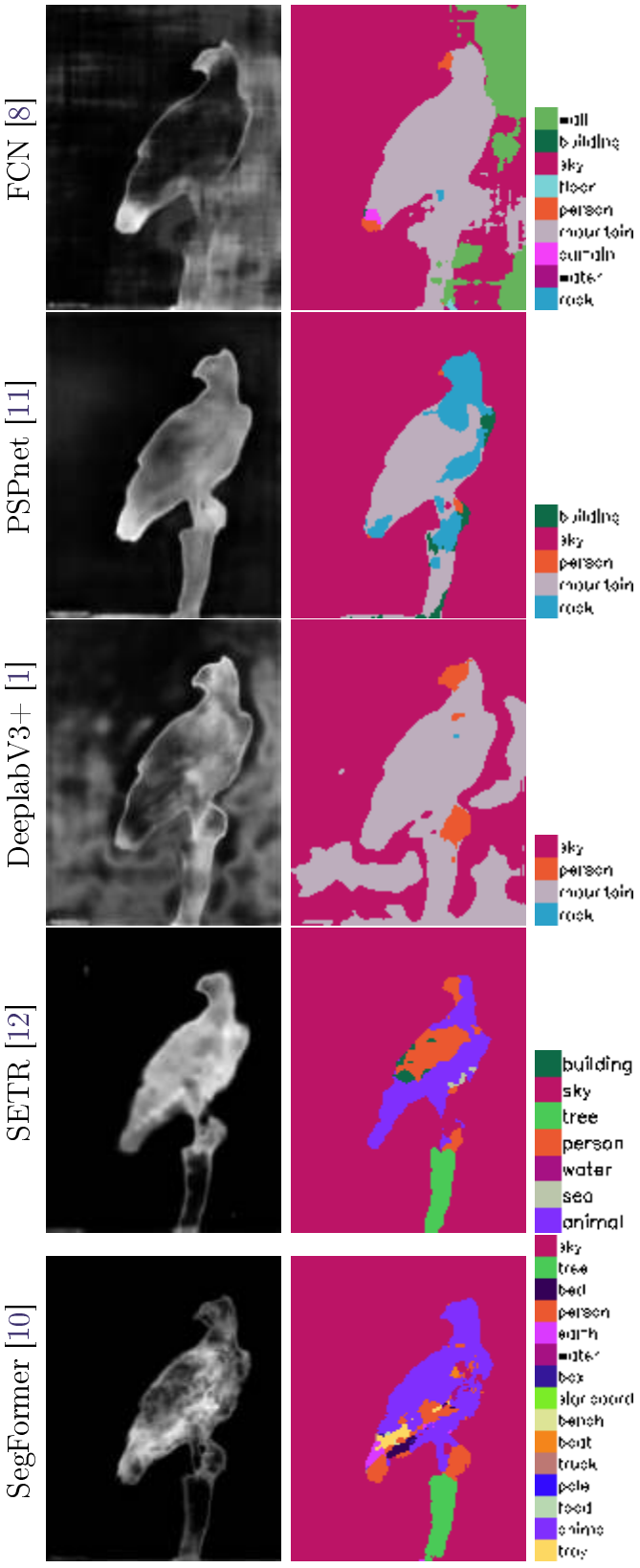

Figure 13: Segmentation predictions using the five architectures on a bird image from the ADE20K validation set. The input image is shown in Figure 9. Each row represents a different network. The left column gives the entropy of the detection (a white pixel representing a high entropy), while the right column corresponds to the segmentation alongside the name and color of the detected classes.

likely to be confident on their good answers and unconfident on the bad ones. They also make less errors on these new images which show, on these examples, a better robustness to variability.

Image Credits

 ADE20K dataset [13]

 COCO dataset [6]

References

[1] L-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, in European Conference on Computer Vision (ECCV), 2018. https://doi.org/10.1007/978-3-030-01234-2_49.

[2] MMSegmentation Contributors, *MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark*. https://github.com/open-mmlab/mmsegmentation, 2020.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, in International Conference on Learning Representations, 2021.

[4] J. Gu, H. Kwon, D. Wang, W. Ye, M. Li, Y-H. Chen, L. Lai, V. Chandra, and D.Z. Pan, *Multi-scale high-resolution vision transformer for semantic segmentation*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 12094–12103. https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01178.

[5] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

[6] T-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick, *Microsoft COCO: Common Objects in Context*, in Computer Vision – ECCV 2014, Springer International Publishing, 2014, pp. 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, *Swin transformer: Hierarchical vision transformer using shifted windows*, in IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 10012–10022. https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00986.

[8] E. Shelhamer, J. Long, and T. Darrell, *Fully convolutional networks for semantic segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2017), pp. 640–651. https://doi.org/10.1109/TPAMI.2016.2572683.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[10] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J.M. Alvarez, and P. Luo, *SegFormer: Simple and efficient design for semantic segmentation with transformers*, in Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc., 2021, pp. 12077–12090. https://proceedings.neurips.cc/paper/2021/file/64f1f27bf1b4ec22924fd0acb550c235-Paper.pdf.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, *Pyramid scene parsing network*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. https://doi.org/10.1109/CVPR.2017.660.

[12] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P.H.S. Torr, and L. Zhang, *Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 6881–6890. https://doi.org/10.1109/CVPR46437.2021.00681.

[13] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, *Semantic understanding of scenes through the ADE20K dataset*, International Journal of Computer Vision, 127 (2019), pp. 302–321. https://doi.org/10.1007/s11263-018-1140-0.