Scene Segmentation with Reweighting, Multi-scale Feature, and Dual Attention

Fan Yang
yangf@mit.edu

Zhuo Liu
zhuol@mit.edu

Abstract

Scene segmentation is a challenging problem in computer vision that aims to produce a pixel-wise semantic classification for images. In this course project, we start from the MIT Scene Parsing Benchmark and explore several approaches that potentially improve the performance in scene segmentation in various conditions. We find that re-weighting is beneficial for most scene segmentation tasks that have imbalances number of pixels among different categories. We also analyze the respective strength of multi-scale feature fusion and attention mechanisms in improving the decoder performance in different scenarios. Finally, we propose a novel decoder network that potentially integrates the advantage of multi-scale feature fusion and dual attention mechanisms.

1. Introduction

Scene segmentation, also called pixel-level classification, is one fundamental and challenging problem in computer vision, which aims to cluster parts of a scene image with the same semantics together. Scene segmentation has a broad range of applications, including autonomous driving, precision agriculture, traffic management, and facial segmentation.

Since the rising of the deep neural network, the ability to extract the features of images has greatly promoted. Compared with typical image-level classification tasks, where each image is treated as an identical category, scene segmentation usually requires the network to produce correspondingly-sized output with efficient inference and learning. Scene segmentation thus requires an additional "decoder" network instead of a fully connected layer on the basis of "encoder" network to transfer the down-sampled feature extracted by the backbone convolution network to a prediction mask with the same size as the original image. This encoder-decoder structure is now widely accepted as a typical approach for scene segmentation.

Most state-of-the-art models for scene segmentation are based on the Fully Convolution Network (FCN) [6]. The original FCN simply turns fully connected layers to 1x1 convolution layers and applies upsampling (backward strided convolution) to learn a prediction for the per-pixel task. Many recent improvements on FCN tend to utilize the multi-scale context fusion [5]. For example, Pyramid Scene Parsing Network (PSPNet) [10] aggregates the output of the "encoder" network with Pyramid Pooling Module (PPM), where feature maps of different levels generated by pyramid pooling were concatenated for final classification. The recent Unified Perceptual Parsing Network (UPerNet) [8] combines Feature Pyramid Network (FPN) and Pyramid Pooling Module (PPM), and achieves better performance than PSPNet. Apart from utilizing multi-scale features, some studies try to approach this challenge from other aspects. For example, the Dual Attention Network (DANet) [3] appends a position attention module and a channel attention module to the global presentation generated by the encoder. This method enhances effectively the spatial/channel dependencies between positions of the feature maps and achieves even higher performance.

In this course project, we explore some potential improvements on the studies reported by the MIT Scene Parsing Benchmark [1]. The UPerNet is the state-of-art decoder that achieves the best performance in the benchmark. The configuration of Resnet-50 being the encoder and the UPerNet being the decoder is already implemented and trained on Ade20k dataset in the benchmark, and we treat this result as the baseline. For improvements, first, we study the performance of UPerNet on a new dataset – COCO10K, which is a subset of the Coco-Stuff dataset [2] [1]. We make revisions to the original model to accommodate the dataset difference and improve the performance of the new dataset. This revision implementation is general, and we foresee that it can be applied to other datasets which face a similar issue. Second, we implement and train another decoder network, DANet, which seeks to improve the decoder from a totally different aspect compared to UPerNet. We identify the respective advantages of these two decoders by comparing their predictions. Finally, based on our experiments in the second part, we take the advantage of both the networks and propose a novel decoder network for image segmenta-

[1] http://sceneparsing.csail.mit.edu
[2] https://github.com/nightrome/cocostuff

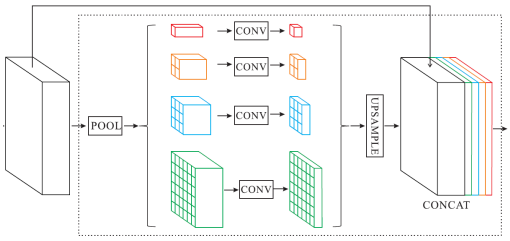

Figure 1. Pyramid Pooling Module (PPM). This figure is adapted from Zhao *et al.* [10]

tion. All our implementations are based on the open repository of the benchmark [3].

2. Related work

2.1. Pyramid Pooling Module and Feature Pyramid Network

In a deep neural network, the size of receptive field roughly indicates how much we use context information. The empirical receptive field is usually much smaller than the theoretical one, especially in the deep layers. This phenomenon impairs the network ability to correctly capture the global context when producing predictions. Pyramid pooling module (PPM) was firstly proposed as a module in the PSPNet [10] and was designed to counter this problem. As shown in Fig. 1, a PPM fuses four levels of feature maps created by a pooling layer with different pooling scales. PPM acts effectively as a global contextual prior, and it's widely used in semantic segmentation.

Feature pyramids are heavily used component in detecting objects at different scales in recognition systems. However, feature pyramids are both compute and memory intensive. The inference time brought by generating feature pyramids makes it infeasible for real time applications, and the extra memory induced by storing feature pyramids also makes adding image pyramids impractical during the training process. Feature pyramid network is designed to address the above issues by generating multi-scale feature representation through a deep neural network [4]. FPN leverages the pyramidal shape of the convolution network and creates feature representation in each scale with the combination of each layer of convolution network and a bottom-to-top up-sampling structure as shown in Fig. 2.

2.2. Attention Modules

Attention modules can model long-range dependencies and have been widely applied in many tasks. For example, self-attention mechanism can help to draw global dependencies of inputs, and it can be applied in machine translation.

[3] https://github.com/CSAILVision/semantic-segmentation-pytorch

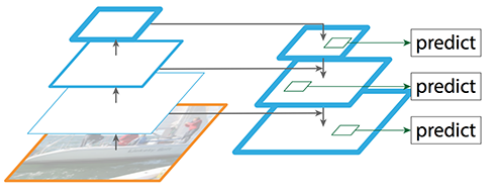

Figure 2. Feature Pyramid Network (FPN). This figure is adapted from Lin *et al.* [4]

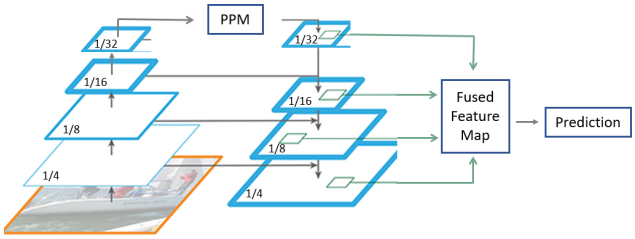

Figure 3. Unified Perceptual Parsing Network (UPerNet). This figure is adapted from Zhou *et al.* [8]

Woo, *et. al* (2018) [7] introduces self attention mechanism to learn a better image generator. Their work mainly explored the effectiveness of non-local operation in space time dimension for videos and images. Different from previous works, they extend the self attention mechanism in the task of scene segmentation, and carefully design two types of attention modules to capture rich contextual relationships for better feature representations with intra-class compactness. Their comprehensive empirical results verify the effectiveness of the attention modules and directly inspires the dual attention network (DANet).

3. Methods

3.1. Baseline

Our baseline is adapted from the MIT Scene Parsing Benchmark. The encoder network is Resnet-50, the decoder network is UPerNet, and the dataset used to train the model is ADE20k [11, 12].

The ADE20K dataset contains more than 20K scene-centric images annotated with objects and object parts. The entire dataset is divided into 20K images for training, and 2K images for validation. There are a total of 150 semantic categories for evaluation. These categories include stuff like scenes such as ground, sky, and also discrete objects like humans, houses, etc. Apart from the 20K original images, it also provides gray-scale annotated images for accuracy evaluation. The grayscale of the pixel between 0-149 stands for different categories. To give a more visualized sense of the network performance, 150 different RGB combinations are assigned to different categories for the evaluation dataset

Figure 4. Sample comparison images from baseline ADE20K dataset. Left: original image; middle: manually annotated image; right: annotated image by the model.

instead of gray-scale values. The example of the generated images for evaluation shown in Fig. 4 contains three parts with the original image on the left, the pixel-wise annotated RGB images in the middle, and the predicted images by the network on the right.

For the decoder network, the framework of UPerNet is straightforwardly derived from PPM and FPN, which is shown in Fig. 3. A PPM is applied to the last layer of the encoder network before feeding it into the top-down branch in FPN. UPerNet thus can do a better job in unifying parsing of visual attributes at multiple levels.

Our baseline directly originates from the benchmark's implementation, but we train the model on our own machines for better comparison. The encoder network is relatively independent of the research for scene segmentation, so we will fix the encoder network and explore alternative decoder networks and datasets.

3.2. Datasets

We then extend the network to the other dataset - the COCO-Stuff and evaluate its performance on different categories. The entire dataset has a total of 164K images, among them there are 118K images for training, 5K images for validation, and 20K images for testing. However, to have a reasonable training time for this project, we only utilize a subset of it, which is the COCO-Stuff 10K (COCO10K) dataset. It includes 10K images for training and 1K images for validation.

Similar to the ADE20K dataset, it has pixel-wise annotations with 182 categories for all the images. Among these categories, there are 91 thing classes and 91 stuff classes. The annotations, different from the ADE20K dataset, are stored with '.mat' data files. We transformed these data into gray-scale images to be consistent with the current networks. These images have a grayscale from 0-181 which stands for different categories. Finally, we also extended the RGB categories to a total of 182 different combinations to provide a visual evaluation of the network performance on the new dataset.

We first plot the histogram of all the 181 categories in the COCO10K dataset, both respect to the number of pixels and to the number of images. The results are shown in Fig. 5. It's obvious that there's a large imbalance between different categories. Roughly 20 categories occupy a large amount of pixels whereas other categories occupy less than 10% of the entire dataset.

Therefore, the major categories get well-trained, and those minor categories don't have enough chance to show up in the training process. This hypothesis can be further evidenced by the prediction image of the COCO10K dataset shown in Fig. 6. The model successfully predicts the major categories such as sky and trees, but fail on minor categories such as the red brick.

To solve this issue, we assign different categories with different weights. Specifically, the weight of a certain category is inversely proportional to its "frequency".

$$\begin{aligned} W_c &= \frac{1}{F_c}\ (F_c > \alpha) \\ W_c &= 0\ (F_c \leq \alpha) \end{aligned} \tag{1}$$

Here, the 'frequency' can be either defined as how many pixels that category appears or how many figures that category appears. We empirically find that defining 'frequency' for images yields a better result, and we'll adopt this method afterward. $\alpha = 130$ is an empirical value that prevents categories with too large weights. This means we neglect extremely small categories and set their weights to zero. The number of these categories is small compared to the entire category size. Also, since these categories almost don't have the chance to get trained in the original model, ignoring them wouldn't reduce the performance in the revised model.

We don't apply different weights of the categories to the revised model directly. From the histogram, we see that different categories have almost one order of magnitude difference. Imposing these large different weights, though improving the performance of the minor categories to some extent, would significantly reduce the performance of the major categories, and cause the performance reduction of the overall dataset. Instead, we define the image weights as a function of category weights.

$$W_I = norm\left(\frac{\sum_{C\,in\,I} W_c}{N_c}\right) \tag{2}$$

The weight of a certain image W_I is the sum of the weights of all the categories W_c that appear in the image and is divided by the total number of categories in that image. Finally, the weights of all the images are normalized so that the sum of the weights is equal to unity. This normalization is not necessary for the current step but is useful in the next step shown below.

The performance of this revised model (model 1) turns out to be slightly worse than the original model, which will be shown in the next section. We attribute this to the overweight of the minor categories in our model.

To solve this issue, we propose the new model (model 2) to weight images to be the average of the proposed weights

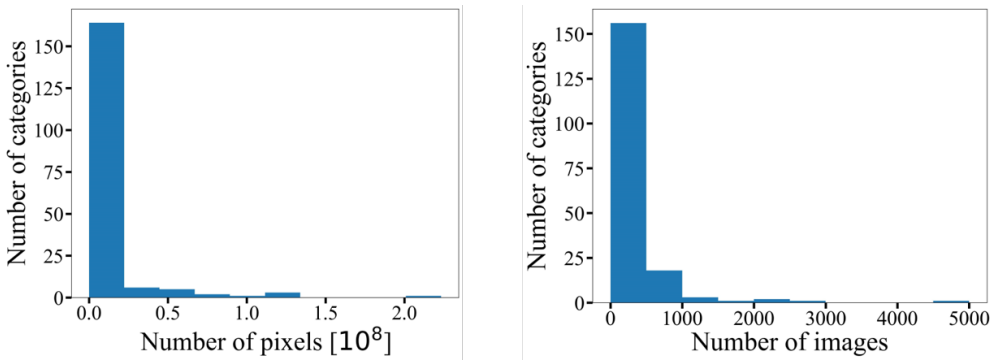

Figure 5. Histograms of COCO10K dataset categories with respect to number of pixels (left) and number of images (right)

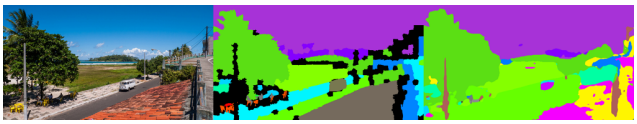

Figure 6. Sample of the predicted image of COCO10K dataset with the original model

and the original weights.

$$W_{I(new)} = 0.5W_I + 0.5\frac{1}{N_I} \tag{3}$$

Here, $\frac{1}{N_I}$ is the original equal weight of all the images, it's averaged with the proposed weights W_I to get the final new weights $W_{I(new)}$. This way, the weights of the major categories are smaller than the original equal weights but larger than the previous proposed weights, and the weights of the minor categories are vice versa. All the weights of the images are ensured to be at least larger than half of the original weights, which helps to improve the performance of the minor categories while maintaining the performance of the major categories. The performance evaluation will also be presented in the next section.

3.3. DANet

In this part, we seek to investigate on a decoder network – DANet, which approaches the problem from a totally different aspect. To our knowledge, the study of DANet on ADE20k dataset has never been reported before.

In DANet, the feature map extracted by the encoder network is fed separately into a position attention module (PAM) and a channel attention module (CAM). The sum of the output of these two modules is for final prediction. In PAM, to obtain the attention map, we do the following: given a local feature $\mathbf{A} \in \mathbb{R}^{C\times H\times W}$, we first feed it into a convolution layers to generate two new feature maps $\mathbf{B}$ and $\mathbf{C}$ of the same dimension as $\mathbf{A}$. Then we reshape them to $\mathbb{R}^{C\times N}, N = H \times W$. The spatial attention map $\mathbf{S}$ can be

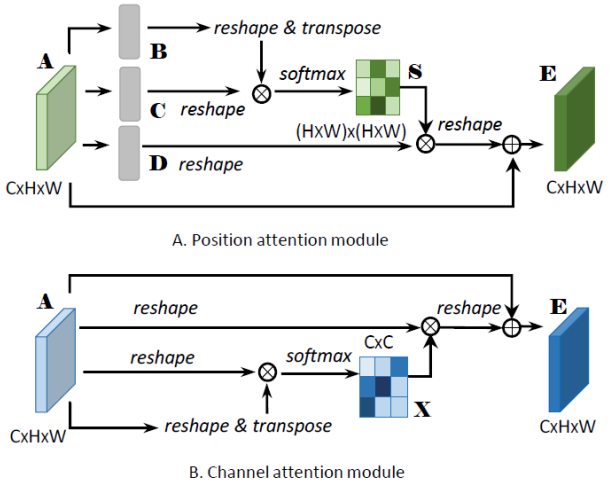

Figure 7. The details of Position Attention Module and Channel Attention Module are illustrated in (A) and (B). This figure is adapted from Fu *et al.* [3]

obtained as

$$S_{ji} = \frac{\exp\left(B_i \cdot C_j\right)}{\sum_{i=1}^{N} \exp\left(B_i \cdot C_j\right)} \tag{4}$$

$S \in \mathbb{R}^{NXN}$, thus S_{ji} measures the i^{th} position's impact on j^{th} position. In CAM, to obtain the attention map, we directly calculate the channel attention map $\mathbf{X} \in \mathbb{R}^{C\times C}$ from the original features $\mathbf{A} \in \mathbb{R}^{C\times H\times W}$:

$$X_{ji} = \frac{\exp\left(A_i \cdot A_j\right)}{\sum_{i=1}^{C} \exp\left(A_i \cdot A_j\right)} \tag{5}$$

$X \in \mathbb{R}^{CXC}$, thus X_{ji} measures the i^{th} channel's impact on j^{th} channel. The details of the two attention modules are illustrated in the Fig. 7. The self-attention mechanism potentially enhances dependencies of different parts of the input to help better segment images.

3.4. PPM-DANet

Inspired by UPerNet and DANet, we propose a novel decoder network that combines the idea of fusing multi-scale features and applying dual attention. We apply a PPM before feeding the feature map into PAM and CAM. We call it PPM-DANet. The structure of PPM-DANet is shown in Fig. 8. We perform some preliminary investigations on this novel model.

4. Experiments

4.1. implementation details

Our implementation is based on MIT Scene Parsing Benchmark [4]. We use the "poly" learning rate policy, where current learning rate equals to the initial learning rate multiplying $(1 - \frac{\text{iter}}{\text{total\_iter}})^{0.9}$. We set base learning rate for both the encoder and decoder to be 0.02, and we use a weight decay of 0.0001 and a momentum of 0.9. During training the input image is resized such that the length of its shorter side is randomly chosen from the set {300, 375, 450, 525, 600}. The maximum length of the longer side is set to 1200 in avoidance of GPU memory overflow. For data augmentation, we adopt random mirror and random resize between 0.5 and 2 for all datasets, and additionally add random rotation between -10 to 10 degrees. The layers in the backbone network are initialized with weights pre-trained on ImageNet [2]. For DANet, dilated convolution is applied and striding is removed in the last two groups of the Resnet-50 (Dilated ResNet [9]) in the backbone network (Dilation is not needed for UPerNet). Due to physical memory limitations a mini-batch on each GPU involves only 2 images. Synchronized SGD is appied when training with multiple GPUs. However, some of our results will be trained with only 1 GPU. Noting that batch size has proven to be important to generate accurate statistics [8, 10], training with only 1 GPU may lead to worse performance.

4.2. Impact of limited resource

Before we go into discuss the result, we have to acknowledge that most of our models are poorly trained. For the baseline method, 8 GPUs were used to train for 40 epochs according to the original study in the MIT Scene Parsing Benchmark. In this course project, most of our results are produced by only one GPU, and some results leverage 4 GPUs. We are also not able to train each model for enough epochs due to limited time and computational resource, especially for the results with 4 GPUs.

4.3. Results on the new dataset

The performance of the baseline approach on both the ADE20K dataset and COCO10K dataset, as well as the new models on the COCO10K dataset are listed in Tab. 1.

We tested the original model on ADE20K dataset with 1 GPU and 10 epochs, which yields a training accuracy of 61.1% and validation accuracy of 53.8%. The same settings were then applied to the COCO10K dataset, and a training accuracy of 57.1% and validation accuracy of 49.4% was obtained, which shows worse performance than the original dataset. This is consistent with our hypothesis that COCO10K dataset has more imbalanced categories. In the following, we use 4 GPUs and 4 epochs instead of 1 GPU and 10 epochs to reduce computation time. We found that 4 GPUs help to improve the training performance - 4 GPUs and 4 epochs show a better result than 1 GPU and 16 epochs. Therefore, in order to give a fair evaluation, all the comparisons are made with 4 GPUs and 4 epochs in the following. We'll not compare the results from 4 GPUs to the results from 1 GPU. With the same model and the COCO10K dataset, 4 GPUs show a training accuracy of 80.5% and validation accuracy of 60.1%. We anticipate further improvement of the model with more epochs. However, we are limited by the computational resources we can obtain, which explains why we only use 4 epochs in the project.

The performance of model 1 is evaluated using the same settings - 4 GPUs and 4 epochs and shows the training accuracy of 80.0% and validation accuracy of 58.6%, which is slightly worse than the original model. A sample predicted image is shown in Fig. 9. It shows a worse prediction of the major categories such as the road compared to the original model, which is a sign that we over-weighted the minor categories in our model.

The performance of model 2 - the final revised model is shown in the last row in Tab. 1. It has a training accuracy of 87.6% and validation accuracy of 64.4%. Compared to the original equal weights, it shows the improvement of the training accuracy by 7.1% and validation accuracy by 4.3%.

The improvement may not seem to be much by looking at the overall accuracy. The reason behind it is that the accuracy is mainly controlled by the major categories, which occupy most of the pixels in the dataset. These categories are not the ones that we aim to improve. Therefore, it makes more sense to look at the category-specific intersect over union (IoU) in order to show the influence of imaging weighting. Fig. 10 shows the comparison of the original model and revised model in the major category - human prediction. The predicted parts on the images are denoted by the red rectangle. The two models show a limited difference in this category with the IoU increasing a bit from 0.74 to 0.78. On the other side, Fig. 11 shows the comparison of the two models in the minor category - car prediction. It's obvious that the two models show a much larger difference. The prediction of the car, which is denoted by the red rectangle, fails in the original model but succeeds

[4] https://github.com/CSAILVision/semantic-segmentation-pytorch

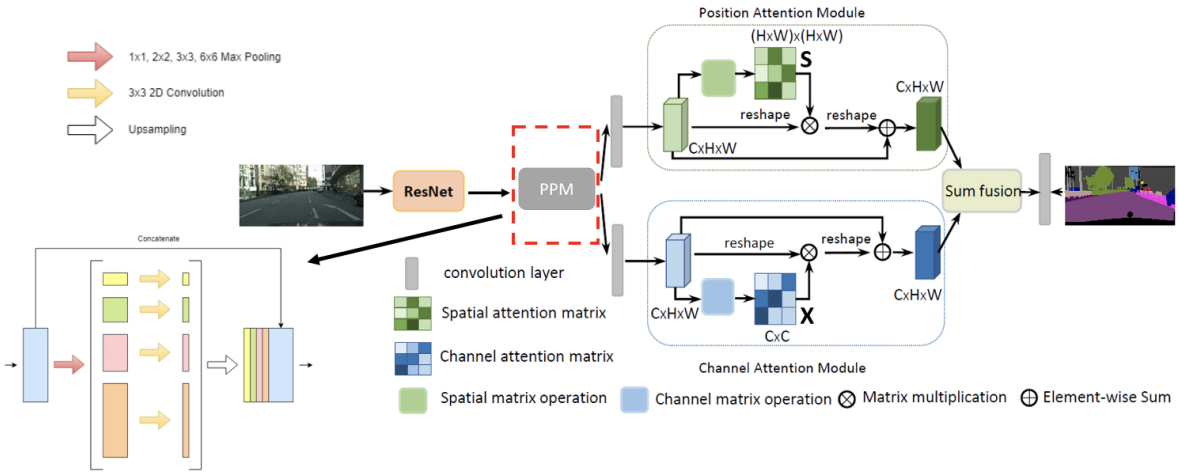

Figure 8. Pyramid Dual Attention Network framework (PPM-DANet).

Model	Dateset	nGPU	Epoch	Train Acc.	Val Acc.
Resnet50-UPerNet	ADE-20K	1	10	61.6%	53.8%
Resnet50-UPerNet	COCO10K	1	10	57.1%	49.4%
Resnet50-UPerNet	COCO10K	4	4	80.5%	60.1%
Revised Resnet50-UPerNet (model 1)	COCO10K	4	4	80.0%	58.6%
Revised Resnet50-UPerNet (model 2)	COCO10K	4	4	87.6%	64.4%

Table 1. Different datasets performance evaluation

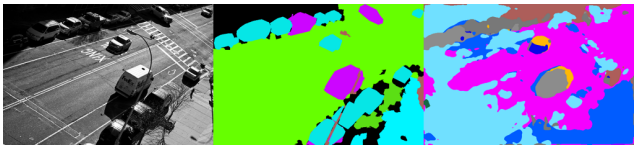

Figure 9. Sample of the predicted image of COCO10K dataset with the initial revised model

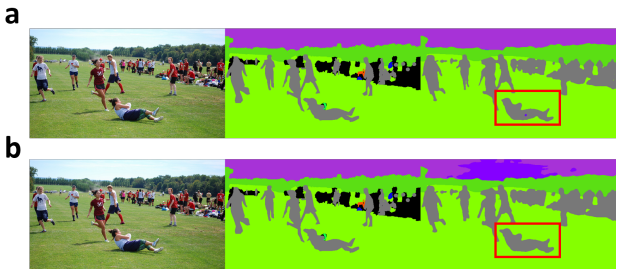

Figure 10. Comparison of the model on major category. (a) Original model with equal weight. (b) Revised model with different weights.

in the revised model. The IoU increased from 0.39 to 0.51 in this case. Therefore, the major categories have marginal improvements while the minor categories have more significant improvements, which is reasonable considering the larger weights of the minor categories in the revised model.

4.4. Results of the DANet

Starting from this section, all the models will be trained on ADE20k, the same as the baseline method. The experimental results are concluded in Tab. 2. Note that the averaging IoU of all the categories is not a good metric when the model is poorly trained for the reason mentioned above. Some categories with a small number of pixels will have IoU of zero, so we are using the averaging IoU of 17 categories that have the most number of pixels in the training dataset. The validation pixel-wise accuracy and IoU are relatively low compared with the performance reported in the benchmark and previous papers. As shown in Table 2, we trained the DANet with 4 GPUs for only 10 epochs, and the results greatly surpass the original one, which confirms that the weak performance is mainly due to the limited computational resource. Therefore, we may not directly compare the performance between UPerNet and DANet with metrics

Model	nGPU	Epoch	Train Acc. (%)	Val Acc. (%)	Mean Val IoU
UPerNet	1	20	67.39	63.04	0.456
DANet	1	20	68.30	63.52	0.439
DANet w/o attention	1	20	68.68	58.30	0.405
DANet	4	10	82.32	72.44	0.548

Table 2. Results on the new decoder network – DANet

Figure 11. Comparison of the model on minor category. (a) Original model with equal weight. (b) Revised model with different weights.

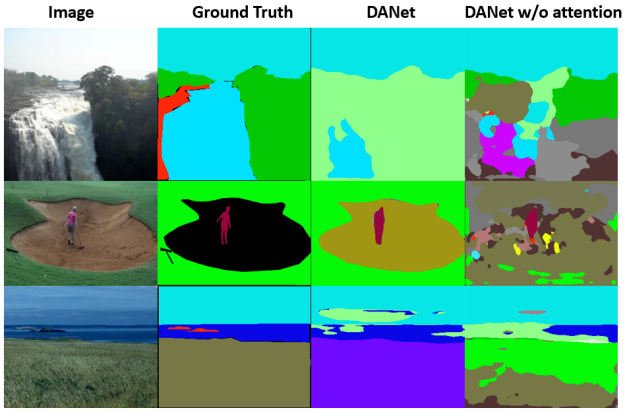

Figure 12. More examples for the ablation study for DANet. These examples demonstrate the poorly trained situation and explain the similar low pixel-accuracies of the two models. However, even though DANet cannot segment the images with correct classification, the segmentation makes much more sense thanks to the attention modules.

such as pixel-wise accuracy and mean IoU because they are not trained to demonstrate their capabilities to the greatest extent.

4.4.1 Ablation study for DANet

Both of our models did not achieve very high pixel-wise accuracies. They also achieved very similar training accuracies with limited training. Therefore, we cannot rule out the possibility that our newly implemented decoder network is not trained to an extent where the inner mechanism is influential and the advantage of this decoder is obvious enough to notice. If this is the case, we are not able to do a further comparison. We thus perform an ablation study for DANet to rule out this probability: we removed the attention modules in DANet but retained the other convolution layers during the training.

We visualize the attention modules to check their effectiveness. For position attention module, the self-attention map is in size of $(\mathrm{H} \times \mathrm{W}) \times (\mathrm{H} \times \mathrm{W})$ (H is the height of the image and W is the width of the image). For a specific point in the image, there is a corresponding sub-attention map representing the attention mask for this point. In Fig. 13, we select one point on a car and show its corresponding sub-attention map in the fourth column. It can be seen that its attention map highlights some nearby cars and some other sharp transitions in the images in the original DANet, while the DANet trained without attention modules fails to produce a semantically reasonable sub-attention map. In other words, the position attention module in our original experiment can indeed capture clear semantic similarity even with limited training. For channel attention, we show the attention strength of channel 12 of the post-CAM feature map. As shown in the last column of Fig. 13, channel 12 responds mostly to the "car" class in our example. We find that the response to "car" is enhanced with trained CAM.

However, the prediction results do not differ too much in this example. Even though DANet trained with attention modules have overall better attention, it still fails to segment some details, such as street lamps, in the image. In the prediction produced by the DANet trained without attention modules, some mistakes in predicting the "road" happen at the bottom of the image, which is not observed in the DANet trained with attention modules. More examples that are more representative of the current training situation are shown in Fig. 12. From these examples, we can further confirm that our models are not well trained. However, even though DANet cannot produce correct classifications

for the images, it usually can segment the images relatively well due to attention mechanisms. For example, the second row shows a man standing in a pit and playing golf. DANet segments the image perfectly but produces the wrong classification for the pit. This also explains the low pixel-wise accuracy of DANet. In contrast, DANet trained without attention modules fails to take the advantage of DANet and tends to predict little fragments with little semantics. Compared to DANet that makes large area prediction errors, DANet trained without attention modules can temporarily achieve comparable pixel-wise accuracy because some fragments are still correctly classified. But we can predict that, if both models are well trained, DANet will surely outperform.

In conclusion, by performing the ablation study, we not only demonstrate that our model are trained to a extend where comparison between different decoder models is feasible, but we justify the cruciality of attention modules in the DANet as well.

4.4.2 Comparison with UPerNet

To compare the performance between DANet and UPerNet, some examples are shown in Fig. 14. We do not have a well-defined metric due to the training condition, but some evidence can be found in these examples to show the respective strength of each decoder network. In the first row, the image shows the scene of a beach with a blue sky and some buildings far away and small. As shown in the red rectangle, UPerNet correctly captures the building even though it is small. In contrast, DANet fails to capture the building and even predicts the "sky" as "sea" due to their similarity in color. On the other hand, the second example presents the advantage of DANet. The scene in the second image is relatively simple. DANet tends to classify similar pixels into the same category through attention mechanisms and avoid deceiving by unimportant details. As shown in the red circle, DANet successfully produce the correct prediction for the grassland, but UPerNet made some mistakes because of the difference in the texture and the brightness of the grass. In conclusion, we make a non-trivial statement here:

1. DANet is relatively better for images with big scenes where details are relatively not important because DANet tends to produce clearer segmentation boundaries.

2. UPerNet is relatively better for images where different features are distributed at different scales thanks to the fusion of multi-scale context.

4.5. Results with PPM-DANet

Due to the fact that a PPM module is applied to the feature map extracted by the encoder network, the feature map feeding into the DANet doubles in the channel (from 512 channels to 1024). As a consequence PPM-DANet is 2-3 times slower in training than DANet. Therefore, we train it with 4GPUs for 10 epochs (even though it is still not enough, but this is what we could afford), and the results are summarized in Tab. 3.

Model	Train Acc.	Val Acc.	Mean Val IoU
PPM-DANet	83.63	68.82	0.587
DANet	82.32	72.44	0.548

Table 3. Results on the new decoder network – DANet

Again, we are not able to justify if PPM-DANet is a better model or not because we have limited computational power and we did not have time to carefully tune the hyperparameters when training. But we can present some preliminary results here, and provide some insights. We observe that PPM-DANet has higher averaging IoU even though the pixel-wise accuracy is lower. This can be due to the fact that PPM-DANet has more parameters and is more difficult to train, but it may produce predictions with better semantics.

Some evidence that PPM-DANet may indeed set off the disadvantages of DANet can be found from the prediction examples as shown in Fig. 15. As circled in red, PPM-DANet can capture more details in the images than DANet (circled in red) while maintaining relatively good semantic segmentation. However, as shown in the last row of Fig. 15, PPM-DANet has worse performance in identifying the big objects and tends to predict more shattered structures at the current stage of training, which leads to a little bit lower pixel-wise accuracy compared to DANet.But this can be due to the fact that the PPM-DANet is more complicated in parameters and more efforts are required to train it properly.

5. Conclusions

We implement scene segmentation analysis by improving models on the existing algorithm. The baseline approach is applied on the ADE20K dataset, it consists of ResNet50 as the encoder and UPerNet as the decoder.

We first implement the model on the new Dataset - the COCO10K. The performance degrades due to the imbalance of the dataset categories. Therefore, we come up with the revised model by assigning larger weights to images containing minor categories, and smaller weights to images with major categories. An improvement of the training accuracy by 7.1% and validation accuracy by 4.3% is achieved. This process is generic, and we foresee that it can be applied to other datasets with similar imbalance issues to

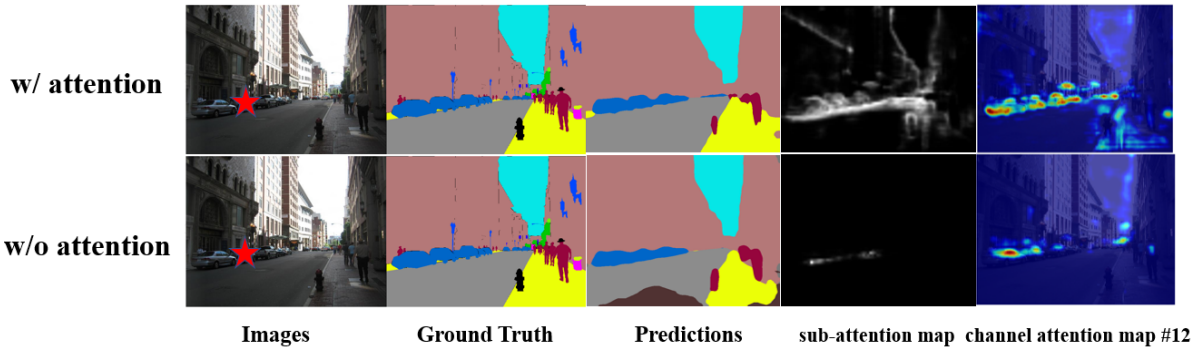

Figure 13. Visualization results of position attention module and channel attention module for DANet trained with and without attention modules. For position attention module (PAM), the sub-attention map of the point marked in red star is shown. For channel attention (CAM) module, the channel 12 of the post-CAM feature map is shown.

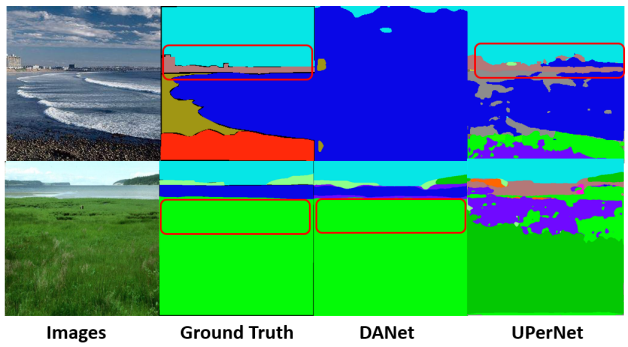

Figure 14. Examples for comparison between DANet and UPerNet. UperNet can better capture details in the image compared to DANet. DANet is more likely to classify similar pixels into the same category through attention mechanisms, and avoid deceiving by unimportant details.

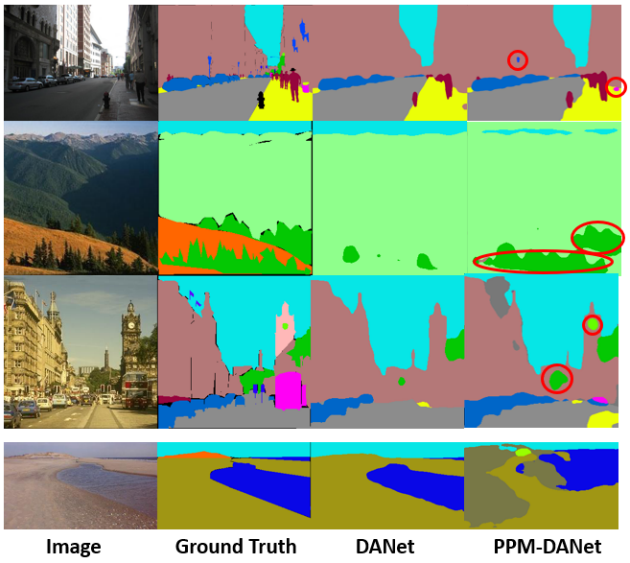

Figure 15. Examples for comparison between DANet and PPM-DANet. Some details circles in red are captured by PPM-DANet but not captured by DANet.

improve the performance of the network.

We then investigate the strength and weaknesses of different types of mechanisms used in the decoder for scene segmentation tasks. It was found that using a multi-scale context fusion can indeed enhance the network's capability in capturing details while applying attention modules can effectively strengthen the semantic connections between different parts of the image and produce clearer segmentation boundaries. We, therefore, propose the more applicable model in different scenarios.

Finally, we propose a novel decoder network that potentially combines the advantages of UPerNet and DANet. Some pieces of evidence are found in our preliminary results to support our idea.

6. Author contributions

Fan Yang contributed to combining the model with COCO10K dataset, analyzing its performance and the potential problems, coming up with the solutions to solve the issue, and finally implementing the dataset with the revised models and analyzing its improvements.

Zhuo Liu worked on the implementation and experiments of the new decoder network, conducted the ablation study, plotted the attention maps, and analyzed the strength and weaknesses of different decoder models. He also proposed the novel PPM-DANet and carried out relevant studies.

All the authors contributed to the general discussions of the project contents.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218, 2018. 1

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[3] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation, 2018. 1, 4

[4] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016. 2

[5] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52(2):1089–1106, jun 2018. 1

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014. 1

[7] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018. 2

[8] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding, 2018. 1, 2, 5

[9] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks, 2017. 5

[10] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2016. 1, 2, 5

[11] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 2

[12] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 2