School of Computer Science and Engineering, Nanyang Technological University, Singapore
Semester 2018-2019 I, November 18, 2018

DM6190 Special Topic Image Segmentation Coursework Project

He Wei, G1802216K

***Review Paper*—ICNet for Real-Time Semantic Segmentation on High-Resolution Images, Zhao, Hengshuang, et al, ECCV 2018**

I. Abstract

THIS paper proposed an image cascade network to tackle the high resolution image semantic segmentation problem in real time.

Firstly, ICNet model[1] focus on establishing a fast semantic segmentation system with decent accuracy. How to balance the trade-off between model performance and model consumption (E.g. Model size, Inference speed) has always been addressed in the deep learning community. To my best knowledge and the paper saying, ICNet is the first method in its kind to accomplish the better above-mentioned trade-off in semantic segmentation application. (See Fig. 1)

Comparing to other state-of-the-art high performance models like Deeplab v2[2], which is with large model size (about 400MB) and high inference latency (4 second per high resolution image), the customized architecture of ICNet is simple and efficient. Light model size and fast inference time is especially meaningful on the real-world application and low computational power device.(E.g. mobile device and embedded systems)

According to the original paper, compared to the backbone network, ICNet achieves 5 times speed up of inference time and memory consumption respectively.

A. Network Architecture & Loss function

After analyzing the computational cost in each layer of PSPNet, the author found that the cost increases squarely regarding to the image resolution size doubling (say comparing the 512×1024 to 1024×2048 input for the sample image).

The speedup principle lays in the cascade branches architecture. Unlike the current popular Encoder-decoder or UNet-based architecture, this model consist of three branches, whose input is the full resolution image and its downsample version by factor 2 and 4 (See Fig. 2). Only the top branch with the low resolution input will go through the whole semantic segmentation backbone network (here is the PSPNet 50 [4]), and it results in a coarse prediction map. Since the most computation burden is in the top branch (dilated convolution, pyramid pooling module, etc.), the way to input low resolution input will be efficient.

In the bottom two branches, a CFF (cascade feature fusion) module is introduced respectively to gradually refine the low-resolution prediction with higher resolution features, and to compute each branch loss with the ground-truth label in corresponding resolution. When calculating the loss, the three

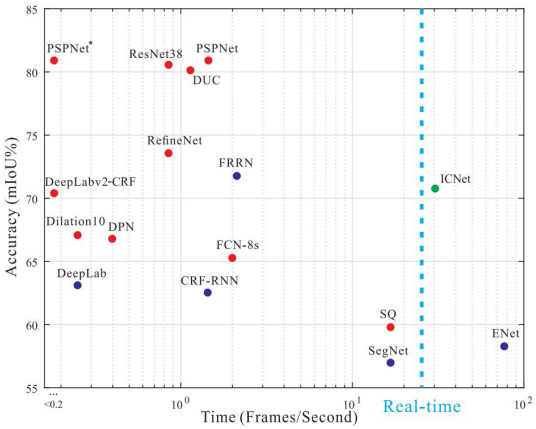

Fig. 1. Inference speed and mIoU performance comparison of different models on Cicyscapes test set.[1][3] The dataset is with high resolution 1024×2048 images in urban scene. ICNet is located in the top-right area.

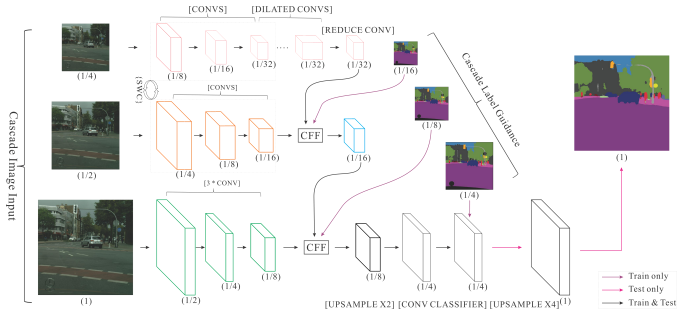

Fig. 2. Image Cascade Network (ICNet) architecture[1]. Each fraction represents the feature maps size comparing the original high resolution image size.

branches loss are weighted summed by three hyper-parameters λ_1, λ_2, λ_3. The loss function L is defined as follows:

$$L = -\sum_{t=1}^{T} \lambda_t \frac{1}{Y_t X_t} \sum_{y=1}^{Y_t} \sum_{x=1}^{X_t} \log \frac{e^{\Gamma^t_{\bar{n},y,x}}}{\sum_{n=1}^{N} e^{\Gamma^t_{n,y,x}}} \qquad (1)$$

Where T is the number of branches, each branch loss is the pixel-wise softxmax cross entropy loss. In addition, during test time the low and medium branch guidance operations are abandoned. (See Fig.2)

B. Limitation

1) Information Loss

Although the idea try to restrict the input size can truly reduce the computation complexity of the network, it also

loses the spatial information a lot, for instance, the boundaries nearby information and the many small but important instances, which is crucial for segmentation. Consequently, the downsampling rate for the branch input should be well decided before implementation. On one hand, if the downsample rate is too high, the information loss become larger. On the other hand, if it is too low, the running time will not satisfy the real-time requirement.

2) *Generalization*
The experiments of ICNet only focus on high resolution dataset, and did not specify the model performance on low resolution cases. In fact, when dealing with low resolution segmentation, there is no need to downsample the input in this way. Also, the original paper did not compare the FLOPs with other models.

3) *Backbone Network*
The model performance and capacity much rely on the backbone network. Although the paper stated that some current model compression technique failed in the current backbone PSPNet50, it is possible to replace other state-of-the-art model with well compression. Also, as the backbone network has been well trained and evaluated in several dataset, we can also make use of the information from the pretrained network, such as utilized the pretrained weigh for initialization, teacher-student network training, knowledge distillation[5] and so on.

II. Implementation

In order to validate ICNet, in here, the implementation is on other dataset from a Kaggle competition, Carvana Image Masking Challenge[1]. The objective is to automatically removes the photo studio background. To conclude, it is a binary semantic segmentation challenge in high resolution images.

In my implementation, I also compare the ICNet result with the baseline KMeans method. Except for the hyper-parameters tuning, the implementation also try something new, including to fuse the DICE coefficient with the vanilla cross entropy loss, to introduce a ensemble model and to scrap real-world car images to evaluate the model generalization. Different experiment is conducted and the final result is submitted to the website.

The implementation code is also available on my Github[2], which is based on the ICNet Tensorflow implementation[3] and the other above-mentioned novelty modification are implemented by myself.

A. Dataset Details

The kaggle dataset contains a large number of car images (as .jpg files). Each car has exactly 16 images, each one taken at different angles. Each car has a unique id and images are named according to id_01.jpg, id_02.jpg ... id_16.jpg. Specifically, in this project, we are given a training data set which has **5,088** high resolution images (each image is 1918×1280), and a test set that contains **100,064** images with the same size. As we can only have the score on the test set after the result submission, the training data will be partitioned into 4,064 for training and 1,024 as the validation set.

B. Dataset Exploration

To better explore the data, the mean and variance of RGB channel, and the car (foreground) area percentage on each datase partition respectively are visualized and compared. Additionally, data augmentation (e.g. random mirroring and random scaling) and normalization with image mean is completed in the pre-preocessing step.

Among all given images, the car area retains around 15-25 percent, and RGB value would be around 174-175. The background screen is almost the same, but challenging cases like similar background-foreground exists.

C. Hyper-Parameters Tuning

The implementation experiments include to tune up the hyper-parameters (epoch, branch loss weight λ_1 , λ_2 , λ_3 , and the learning rate) with the metric MIoU(*Mean of class-wise intersection over union*) on validation set. When tuning one specific hyper-parameters, the other ones stay in the default setting (See. Table VI).

The result is shown in Table I - III in Appendix.

D. Loss Function Modification

The vanilla loss (1) is the pixel-wise cross entropy loss. In this implementation, DICE coefficient, which is able to reflect the similarity between the ground truth map and prediction map and commonly seen in binary semantic segmentation case, is included. The binary cross entropy loss mixing DICE loss is defined as follow:

$$\mathrm{BCE} = -\sum_{y=1}^{Y_t}\sum_{x=1}^{X_t}(f^t_{n,y,x}\cdot\log f^t_{n,y,x} + (1-f^t_{n,y,x})\cdot\log f^t_{n,y,x}) \quad (2)$$

$$DICE = 2\frac{|X\cap Y|}{|X|+|Y|} \quad (3)$$

$$NewLoss = BCE - In(DICE) \quad (4)$$

The result is shown in Table IV in Appendix, the training loss versus validation loss doesn't show obvious overfitting (See Fig. 3).

E. Train from scratch vs restore pre-trained weights

The experiments of randomly initialized the network weights and inherited from pre-trained weights from other dataset has also been conducted (Cityscapes). In here, due to time limitation, the comparison is done with the default epoch 20 only.

The result is shown in Table V in Appendix. The pre-trained mode outperforms the scratch training mode, which can be attributed to the similarity between the dataset and Cityscapes. Moreover, it may also because the training epochs is not enough to find the optimal solution. (Due the time limitation, I didn't try more epochs in this experiments)

[1] https://www.kaggle.com/c/carvana-image-masking-challenge
[2] https://github.com/erichhhhho/DM6190_Coursework
[3] https://github.com/hellochick/ICNet-tensorflow

F. Different models comparison

Different models result are compared, including the K-Means clustering (See Fig. 3), the well-trained ICNet model and the ensemble model (See Fig. 4 & Fig. 5), which fused the TernausNet, a Unet+VGG11 based model [6], with our implemented ICNet on the last activation layer outputs.

The ensemble process also include various weight pair experiments, i.e. (0.4,0.6), (0.5,0.4), (0.6,0.4). And the 0.4 for ICNet model perform the best among them. The baseline K-Means yield at around MIou 0.23, while the ICNet achieve MIou more than 0.92.

G. Inference evaluation

The inference time comparison result is shown on Table. V, including the above-mentioned methods. The real-world data includes the 100,064 Kaggle test set images and 200 scraped images from different websites.

The result shows that the generalization performance is not perfect, though the inference time is the shortest. In cases that ICNet cannot well classify (See Fig. 8), the compensation of ensembling TernausNet works better (See Fig. 6).

H. Experiments environment

All the training session is with an NVIDIA 1080ti GPU and tensorflow-gpu v1.11 in Ubuntu 16.04, and each of them takes about 4 to 5 hours. The outcomes in Kaggle is in Fig. 7.

III. Conclusion

In this segmentation coursework project, different experiments of ICNet in the Kaggle dataset have conducted. As a result, the ICNet can accomplish near real-time inference in the novel dataset with decent accuracy (not the top one though). The well tuned trained model size is around 25MB.

However, the generalization result is not satisfying, and the scratch training should take more times to optimize. From my perspective, as mentioned before in Section I, the way to make use of the information from pretrained backbone network and to compress them efficiently will be a possible future work. Moreover, some recent papers in this similar idea also appear[7].

References

[1]Zhao, H., Qi, X., Shen, X., Shi, J., & Jia, J. (2017). Icnet for real-time semantic segmentation on high-resolution images. arXiv preprint arXiv:1704.08545.

[2]Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062.

[3]Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3213-3223).

[4]Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017, July). Pyramid scene parsing network. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (pp. 2881-2890).

[5]Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[6]Iglovikov, V., & Shvets, A. (2018). TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. arXiv preprint arXiv:1801.05746.

[7]Spek, A., Dharmasiri, T., & Drummond, T. (2018). CReaM: Condensed Real-time Models for Depth Prediction using Convolutional Neural Networks. arXiv preprint arXiv:1807.08931.

Appendix

TABLE I
Learning rate Tuning

Learning Rate	MIoU(%)
0.05	0.20590943098068237
0.01	0.9010630249977112
0.005	0.8684223890304565
0.001	0.8903300762176514
0.0001	**0.9285993576049805**

TABLE II
Branches Loss weights Lamda Tuning

λ_1 , λ_2 , λ_3	MIoU(%)
0.16, 0.40, 1.0	0.9004056453704834
0.16, 0.50, 1.0	0.8718999624252319
0.16, 0.30, 1.0	**0.9143388271331787**
0.20, 0.40, 1.0	0.9022904634475708
0.12, 0.40, 1.0	0.8527867794036865
0.16, 0.40, 1.2	0.9092841148376465
0.16, 0.40, 0.8	0.8243488073348999

TABLE III
Epoch Tuning

Epoch	MIoU(%)
200	**0.9181809425354004**
100	0.8876086473464966
80	0.4844314455986023
40	0.7748814225196838
20	0.6844799518585205
10	0.8811067342758179
5	0.6430301666259766

TABLE IV
Loss Comparison

Loss Type	MIoU(%)
Vanilla Loss	0.906308650970459
BCE+DICE	**0.9207372665405273**

TABLE IV
Pre-trained vs Train from scratch

Initialization Type	MIoU(%)
Pre-trained	**0.9207372665405273**
Train from scratch	0.10197679698467255

TABLE V
AVERAGE INFERENCE TIME ON REAL WORLD DATA

Methods	Inference Time (Second)
K-Means	4.18
ICNet	**0.130**
Ensemble with TernausNet	0.632

TABLE VI
DEFAULT PARAMETERS CONFIGURATION

Parameters	Value
BATCH_SIZE	16
IMG_MEAN	[177.68,175.84,174.21]
INFER_SIZE	[1280,1918,3]
λ_1 , λ_2 , λ_3	0.16, 0.4, 1.0
TRAINING_EPOCHS	20
TRAINING_SIZE	[720,720]
MOMENTUM	0.9
LEARNING_RATE	0.0005
num_classes	2
random_mirror	false
random_scale	false
eval_steps	1024

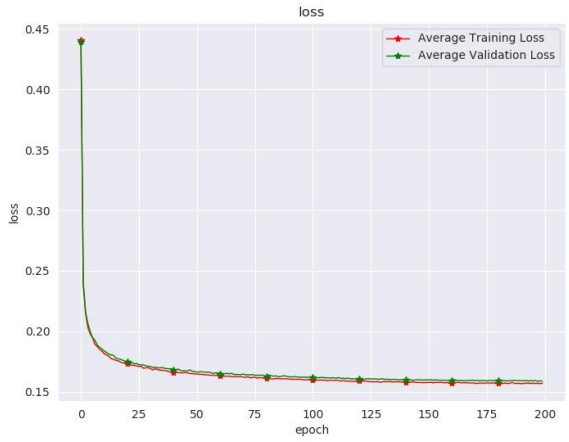

Fig. 3 Training loss and validation loss on 200 epochs training

Fig. 4 K-Means clustering baseline result with visualization

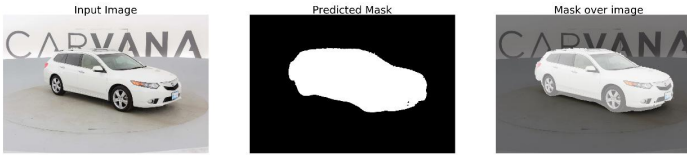

Fig. 5 ICNet well predicted result

Fig. 6 Comparison on ICNet bad predicted case with TernausNet and ensemble result

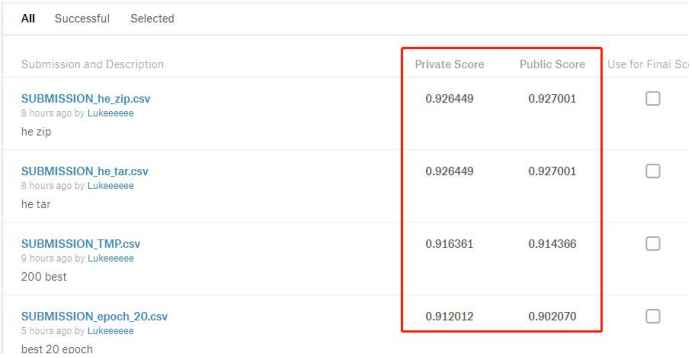

Fig. 7 Submission on Kaggle Competition

Fig. 8 Inference result on real-world data