THE REPRODUCIBILITY OF ENET

YUEFU LIU, YANG HONG, HANG SU

ABSTRACT

Architecture, parameters of ENet as well as some related models are explored in this paper as the supplemental experiments of the ENet paper (Adam et al.).

1 INTRODUCTION

This paper analyses the segmentation model, ENet, which was originally proposed in ICLR 2017 but received a rejection from the committee due to a lack of detailed experimental results. To evaluate more experimental results, we reproduce the main results of their papers by debugging existing code git (c), extensively analyze the ENet architecture and compare it with CGNet and DeepLab V3.

2 EXPERIMENTAL RESULTS AND ANALYSIS

2.1 ANALYSIS OF ENET STRUCTURE

This section explores the performance of the three most representative design choices in ENet. Experiment accuracy was measured by Mean IoU (Intersection over Union) and obtained from the corresponding model trained in three rounds on CamVid dataset with epoch 300.

Dilated convolution The main reason for ENet adopts dilated convolution to boost accuracy is that dilated convolution helps in gaining wider receptive field so that some segmentation tasks that require large context, such as detecting riders or pedestrians in a road scene, would benefit. It is found that except down-sampling bottleneck, most of the convolution layers are applied in the regular bottlenecks of the encoder, and the dilation factor k for every regular bottleneck is doubled for every layer up, giving $k = 1, 2, 4, 8, 16$. We are interested in understanding how this design would affect performance. Accuracy was compared in both validation and test set between the original architecture and the architecture with consistent $k = 1$. It turns out that the best accuracy was not significantly affected in both validation set and test set but such consistent dilation factor took more epoch in achieving its highest accuracy (averagely 178 epoch in original model and 240 epoch in modified model), which suggests that exponentially increased dilated convolution layers can be more efficient in extracting useful features from the inputs.

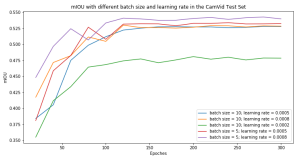
Early down-sampling There are two down-sampling blocks in ENet serving as an encoder to reduce the input size. For CamVid dataset, the dimension of the input is reduced from early $3 \times 360 \times 480$ down to $128 \times 45 \times 60$ after the final stage of the encoder. As claimed by the author, this dimension reduction is designed not only for easing computational load but also for removing redundant representation in visual information. However, it is noticed that each down-sampling bottleneck is followed by a bunch of regular bottlenecks (4 in the first stage and 8 in the second stage), inside which the input channels are reduced into a quarter of its size and then restored back afterwards. We are interested in evaluating how this design affects the model's performance. After several rounds of experiments, it shows that the absence of regular bottlenecks brought adverse impact on its predictive accuracy in both validation set and test set, giving decrease of 3.2% and 3.0% respectively in mean accuracy and damage in the efficiency in searching its best performance (averagely 178 epoch in original model and 253 epoch in modified model). Therefore such experimental evidence proves that deeper convolution layers are beneficial in extracting more useful features for different objects.

Information-preserving dimensionality changes Max pooling layer and max Unpooling layer are found to be paralleled with a sequence of convolution layers in down-sampling and up-sampling

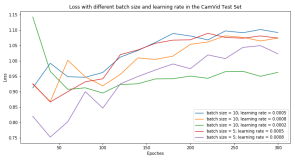
bottleneck respectively doing element-wise addition with their outputs. This design, cited by the author, is aimed to alleviate the aggressive dimensionality reduction brought by convolution layers as too much convolution would potentially remove some spatial information and hinder information flow. This section compares the performance between the original architecture and the architecture without this information-preserving design. The result shows that the removal of the max-pooling layer in single down-sampling and the up-sampling bottleneck is more detrimental than the removal of a bunch of regular bottlenecks. The predictive accuracy decreased by 4.4% and 4.8% in the validation set and test set in this experiment, giving 1.2% and 1.8% worse than that regarding regular bottleneck. Such results demonstrate that max pooling layer plays a crucial role in enhancing predictive accuracy by preserving sub-regional information from input representation.

2.2 ENET PARAMETER EXPERIMENTS

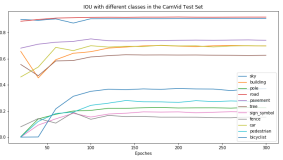
In the original paper, the author directly proposed their parameters. In order to verify whether these parameters are optimum, we evaluated their performances by conducting extensive experiments. The Cityscapes dataset is around 15.1GB large and was converted into 19 categories before training. After this preprocessing, since the Citiescapes dataset has no test set but only the validation set, we compared its scores in the validation set for all training models in Table 1. Performances are recorded with the same running machine. Fig. 1(a) 1(b), show the Mean IoU and loss in the testset of CamVid, and Fig. 1(d), 1(e) show the Mean IoU and loss in the validation dataset of Cityscapes.



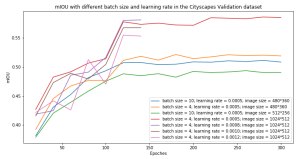
(a) mIOU in CamVid



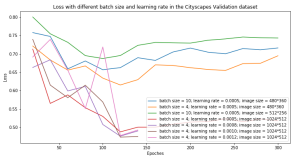
(b) Loss in CamVid



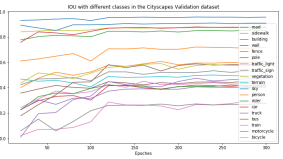
(c) IOU in CamVid



(d) mIOU in Cityscapes



(e) Loss in Cityscapes



(f) IOU in Cityscapes

Figure 1: ENet in CamVid and Cityscapes

Learning rate This parameter works with the decay rate. Although the author pointed out in the paper that the best learning rate is 0.0005 for CamVid, we found that 0.0008 can achieve better results. For Cityscapes, 0.0005 is a suitable choice as a larger learning rate would cause oscillating performance.

Batch size Although the author pointed out in the paper that the best batch size is 10 for CamVid, we found that 5 is a better choice. It is also found that batch size = 5 can improve the performance of CGNet as well. For Cityscapes, the batch size is not as important as that in CamVid. If the image size is lower, small batch size like 5 can achieve better performance.

Image size Judging from Fig. 1 and Table 1, it is noted that the image size has a crucial impact on the final result in terms of Mean IOU and the training time, which we analyzed to be the reason for introducing random scaling in the training process of the CGNet. As seen in Table 1, the result is 0.656 in the validation dataset as we used the same parameters such as decaying rate and batch size of the original paper, where the authors get a result of 0.683 in their paper. Besides, we get the 0.6088 in the baseline model (SegNet), while the authors get 0.629.

Different categories Fig. 1(c) shows the IOU result of ENet in CamVid. The categories of bicycles and cars are the two categories receiving the most improved recognition during training. As for Cityscapes, fig. 1(f) shows that truck and pole are the two categories having the most improved

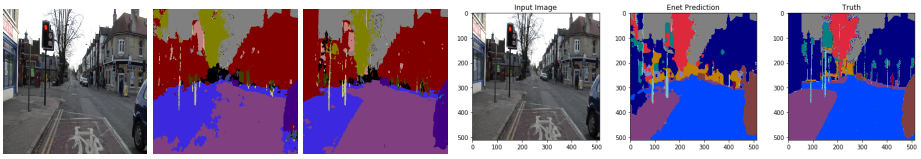
recognition during training. In both two datasets, the recognition of big objects such as sky and road is very high, which is more than 90%. However, the recognition of small objects like traffic sign and fence is relatively low in both dataset. These performances can be reasonable.

2.3 ENET AND SEGNET (GITHUB: GIT (D))

SegNet is the baseline model for performance comparison in the original paper. By observing the results of these two models under the same epoch and running machine, we can see that these two models are different in many aspects. As shown in Table 1, with the same batch size of 5, the SegNet got a poor result and longer training time compared with ENet. Besides, we found that the initial weights play an important role in optimizing the speed and mIOU. The initial weights of SegNet are from the VGG model, which gave longer training time for the SegNet model to learn from the segmentation task. These pre-train weights are foundations of other improvements.

The segmentation result of 20 epoch for SegNet is shown in Fig. 2(b), and the result of 300 epoch in Fig. 2(c). The segmentation result of 300 epoch for ENet is displayed in Fig. 2(e), and the actual target in Fig. 2(f). Compared with the same epoch of 300, ENet can make a more accurate prediction than SegNet, especially for small objects. The running time is summarized in Table 1. As the original paper claimed $18\times$ faster run-time than SegNet, we only got $5\times$ faster run-time, which may be the limitation of our running machine.

In our experiments, the parameter number of SegNet is 29.5M, whilst ENet is only around 0.4M. As the original paper claimed that $79\times$ fewer parameters than SegNet, we verified its authenticity.



(a) Input (1) (b) SegNet(1) (c) SegNet(2) (d) Input (2) (e) ENet (f) Truth

Figure 2: SegNet and ENet

Model	Dataset	Image size	Batch size	GPU memory	Training time	Validation mIOU
SegNet	CamVid	480×360	5	5648MB	6h 11min	0.6088
ENet	CamVid	480×360	5	2725MB	1h 15min	0.6751
ENet	CamVid	480×360	10	4687MB	1h 4min	0.6560
CGNet	CamVid	480×360	5	2171MB	53min	0.7402
CGNet	CamVid	480×360	10	3483MB	1h 10min	0.7174
ENet	Cityscapes	480×360	10	4977MB	11h 28min	0.5082
ENet	Cityscapes	512×256	10	3599MB	9h 10min	0.4903
ENet	Cityscapes	1024×512	4	5421MB	32h 5min	0.5846
CGNet	Cityscapes	1024×512	4	4835MB	41h 36min	0.5990

Table 1: Model Comparison when Epoch = 300 (Pytorch; Hardware: Nvidia GTX 1060)

2.4 ENET AND CGNET (GITHUB: GIT (A))

Since this paper was proposed several years ago, there have been many developments in the optimization. In order to compare the performance gap between this paper and the latest model proposed for the same purpose, we use the new model CGNet published in a few months ago as the performance benchmark. During the training and testing in CGNet model, we tested the suggested value from the CGNet paper for M = 3 and N = 21 and the default value for other parameters such as scale ratio.

Number of parameters CGNet is 0.49M as opposed 0.4M in ENet, so these two models are very similar in this aspect. Compared with SegNet, both these two lightweight models can greatly reduce the number of parameters during training and save the training time and GPU usage accordingly as demonstrated in Table 1, and both have achieved better mean IOU results than SegNet.

Training and testing As shown in Table 1, CGNet can save more GPU memories during training and always recorded better validation results compared with ENet. Furthermore, CGNet achieved 0.632 Mean IoU on the test set of CamVid, while ENet only achieved 0.5023 (batch size = 10). This reflected the influence of improved approaches in CGNet, such as random scaling and random mirror in the training images. Since the size of CamVid dataset is relatively small, these approaches of automatic randomly generated training data can greatly improve the recognition results.

2.5 ENET AND DEEPLAB

In the original conference paper, the authors set SegNet as the baseline model and then benchmarked the performance of Enet and SegNet. However, SegNet was not the state-of-art model at that time. Thus in our experiment, we tried to compare the performance of ENet with Deeplab v3, one of the state-of-art model in 2016 $\sim$ 2017, on the CamVid dataset.

When implementing the Deeplab v3 on CamVid, we used the source code from the official tensorflow repositories git (b). Following the same step of Cityscapes, it is necessary to preprocess the image before the training. We used the script provided by the CamVid website to generate the index of training, testing and validation images, which is used to convert the datasets to TFRecord files as a preparation for training. Next, we added description information of the CamVid dataset into the code and registered the dataset to make the mode support training. Lastly, we modified some parameters and the structure of the model. Some key parameters and results are shown below.

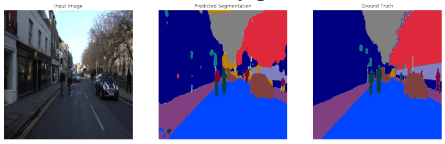


Figure 3: output

Model	Dataset	GPU memory	Training time	Test set Mean IoU
Enet	CamVid	2.8GB	1.1h	0.523
Deeplab v3	CamVid	4.2GB	3.8h	0.614

Table 2: CamVid test set result under the batch size = 4

Using the parameters with learning rate = 0.0008, learning rate decay factor = 0.1, batch size = 4, epoch = 300, Deeplab can be trained in a longer time and occupied more resources as the table 2 shows. Moreover, Deeplab achieved 0.614 Mean IoU and Enet achieved a relatively low score, 0.523. All these evidences reflect the importance of capturing the multi-scale context. It is also reasonable to speculate that Deeplab can make more accurate classification result when increasing epochs according to the optimal parameter provided by an official document.

3 CONCLUSION

In this paper we further explored the design choices, the selection of parameters such as batch size, learning rate, and image size, and compared the typical models of the original baseline (SegNet), the new lightweight (CGNet) and the state-of-art model in 2016 $\sim$ 2017 (DeepLab V3).

REFERENCES

Cgnet. `https://github.com/wutianyiRosun/CGNet`, a.

models. `https://github.com/tensorflow/models/tree/master/research`, b.

Pytorch-enet. `https://github.com/davidtvs/PyTorch-ENet`, c.

pytorch-semseg. `https://github.com/meetshah1995/pytorch-semseg`, d.

Paszke Adam, Chaurasia Abhishek, Kim Sangpil, and Culurciello Eugenio. *arXiv preprint arXiv:1606.02147*.