Better Convolutional Networks: Foundations & Improvements

Bo Pang
Hangzhou, China
3170110152@zju.edu.cn

Jieting Chen
Hangzhou, China
Jieting.17@intl.zju.edu.cn

***Abstract*—Convolutional neural networks have been a very powerful tool in computer vision. The fields such as image classification, image segmentation, and object detection deeply rely on the help of convolutional networks. This paper analyses four cutting-edge essays in the field of convolutional neural networks and computer vision. Firstly, we will focus on the papers, and analyse the challenges these papers face and how they solve the problems by introducing new methods or combining existing methods. Secondly, we will compare these papers, find the uniqueness and commonalities between these papers. By analysing these papers, we can to some extend explore the essence of convolutional networks, and propose some potential improvements to further ameliorate the DNN configuration when we are designing our networks.**

***Index Terms*—convolution, CNN, computer vision, semantic segmentation**

I. Introduction

Convolutional neural networks have been widely used in the field of computer vision since AlexNet, the forefather of CNN, is invented. From the perspective of today, AlexNet is rather primitive, but it is exactly AlexNet that opens the door of convolutional layers. It ignites researches' interests in the mighty of convolutional layers. We can see that, the traditional convolution has evolved to things like stacked convolution, inception, dilated convolution, depthwise separable convolution, deformable convolution, etc. The four papers we will discuss in this paper reveal some of these concepts and open a door of the power of convolution. We will analyze the ideas concerning convolution in these four papers and give some of our opinions about these techniques.

II. Comparison and Evaluation of Methods Used in the Four Essays

A. Fully Convolutional Networks for Semantic Segmentation

That paper explores the application of CNN in the semantic segmentation field. Before the paper, there are methods such as patch-wise training and pre- and post-processing, but Fully Convolutional Network illustrates a more efficient and practical way to solve semantic segmentation problems.

The main idea the paper puts forward is that the typical fully-connected layers have fixed dimensions and throw away spatial coordinates. However, these fully-connected layers can also be viewed as convolutions with kernels that cover their entire input regions, which casts them into fully convolutional

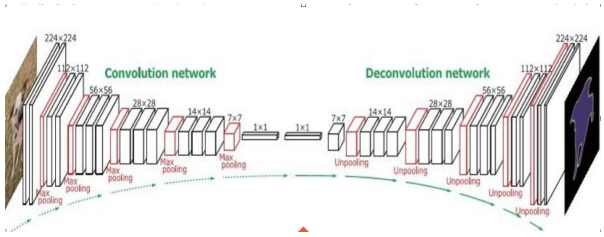

Fig. 1. The whole fully convolutional network, including the convolution network process and the deconvolution network process.

Networks that take input of any size and output classification maps.

Basically, the whole fully convolutional network includes the convolution network and deconvolution network. The process is illustrated in Fig. 1. In detail, since the fully-connected layer and convolution layer can be viewed as equal in practice, we can substitute all fully-connected layers in traditional CNN by convolution layers. After convolution, we get a feature map, or equivalently, the heat-map (Fig. 2), and that is the down-sampling process.

Then, we take up-sampling and generates different segmentation results of the original size in Fig. 3. The point is that the paper uses backwards strided convolution method, which combines predictions from both the final layer and intermediate layer generated from down-sampling process, to predict finer details (see Fig. 3). The purpose of this method

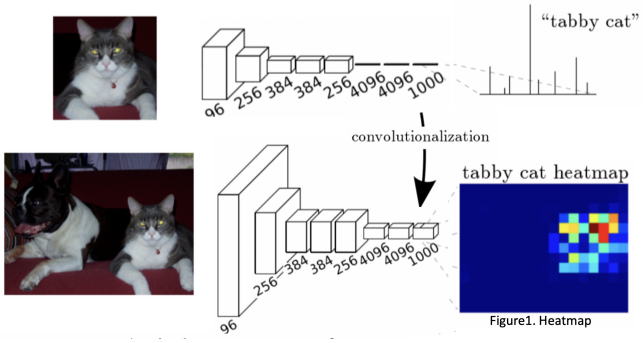

Fig. 2. Example of how we transfer the fully-connected layer to convolution layer. The right bottom image is the heat-map that we get after down-sampling process.

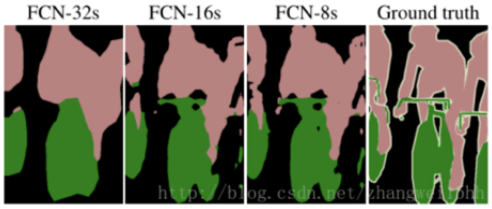

Fig. 3. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Fig. 4).

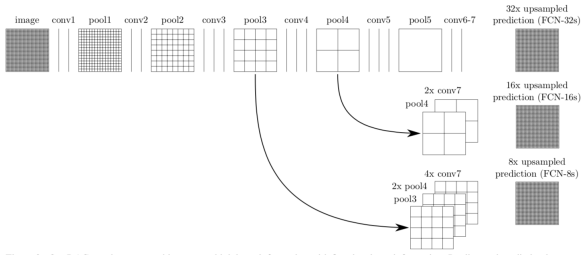

Fig. 4. The process of combine information from the final layer and different higher pooling layers to get different predictions. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. For instance, the second row (FCN-16s) combines predictions from both the final layer and the pool4 layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information.

is that though fully convolutionalized classifiers can be fine-tuned to segmentation, the output is still unsatisfactory. The 32-pixel stride at the final prediction layer limits the scale of detail in the up-sampling output (as shown in Fig. 3.1, FCN-32s). To solve it, the paper combines information in coarse, high layers with low layer, or in other words, combines where and what the feature is. As we see, refining fully convolution nets by fusing information from layers with different strides improves segmentation details (FCN-16s and FCN-8s in Fig. 4), making the segmentation closer to the ground truth.

B. Rethinking Atrous Convolution for Semantic Segmentation

Note that paper is based on the FCN method in the first paper. It uses Atrous Convolution to optimize the process of convolution and deconvolution. See Fig. 5 to understand how it works basically. Briefly, Atrous Convolution is the convolution kernel with "holes" inside it. It tries to include

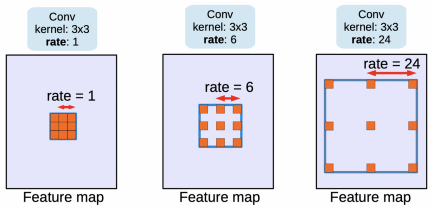

Fig. 5. Atrous convolution with kernel size 3×3 and different rates. Standard convolution corresponds to atrous convolution with $rate = 1$. Employing large value of atrous rate enlarges the model's field-of-view, enabling object encoding at multiple scales.

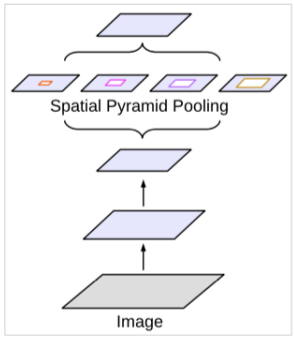

Fig. 6. Spatial Pyramid Pooling model, with the ability to capture multi-scale context.

broader information around the pixel and solve the multi-scale problem of objects. With different rates, we get information in different scales. Besides, Atrous Convolution settles the problem in original FCN method that feature map resolution is reduced and a lot of spatial information is lost when down-sampling.

Based on the existing Spatial Pyramid Pooling model (Fig. 6), the paper puts forward Atrous Spatial Pyramid Pooling (ASPP) model by adopting the kernels with different rates. In traditional way of pooling such as max-pooling, we simply select the max value in $N \times N$ pixels and lost information of the other pixels. ASPP uses different convolution strides and rates to get information in different scales and then concatenates them together, as illustrated in Fig. 7.

ASPP with different atrous rates effectively captures multi-scale information. However, as the sampling rate becomes larger, the number of valid filter weights becomes smaller. That is, the 3×3 filter degenerates to a simple 1×1 filter when the rate value is close to the feature map size. To overcome the problem, the paper applies global average pooling on the last feature map of the model, feeds the resulting image-level features to a 1×1 convolution with 256 filters, and then bilinearly upsamples the feature to the desired spatial dimension to obtain the optimized kernel group for ASPP algorithm. Finally, the kernel group of improved ASPP is consisted of one 1×1 convolution and three 3×3 convolutions with $rates = (6, 12, 18)$ when output $stride = 16$, as shown in Fig. 7.

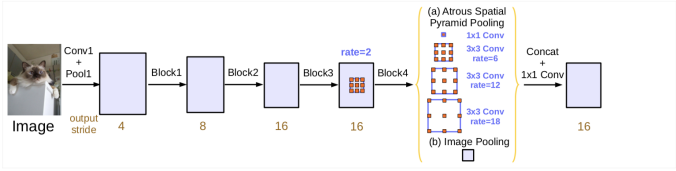

Fig. 7. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

C. YOLO9000: Better, Faster, Stronger

Concerning convolutional layers, this paper is more like an integration of existing methods rather than an essay trying to propose a brand-new technique. This paper combines and utilizes some existing techniques which, have been proved by practice, could speed up the computation or increase the accuracy (IOU in the context of object detection).

Note that YOLO9000 paper is not all about better methods concerning convolutional layers. It also contains some content about other topics such as jointly training. However, here we give them a miss as they are not really the interesting point of this paper.

The authors give a detailed table in their paper, which we cited below. This table vividly describes the process of evolution from original YOLO to YOLOv2.

- Batch Normalization. Batch normalization leads to significant improvements in convergence while eliminating the need for other forms of regularization.
- High Resolution Classifier. The original YOLO trains the classifier network at 224×224 and increases the resolution to 448 for detection. This means the network has to simultaneously switch to learning object detection and adjust to new input resolution. For YOLOv2, the author first fine tune the classification network at the full 448×448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. Then they fine tune the resulting network on detection. This high resolution classification network gives us an increase of almost 4% mAP.
- Convolutional With Anchor Boxes. Instead the fully connected layers used in YOLO, that paper uses anchor boxes to predict bounding boxes. This slightly decreases the accuracy (but increase in recall), so even though the mAP decreases, the increase in recall means that this model has more room to improve.
- Multi-Scale Training. Instead of fixing the input image size, during the training, the researchers change the network every few iterations. Every 10 batches this network randomly chooses new image dimensions. Since this model downsamples by a factor of 32, the authors pull from the following multiples of 32: $\{320, 352, ..., 608\}$. Thus the smallest option is 320×320 and the largest is 608×608. The network is resized to that dimension, then training continues. This regime forces the network to learn to predict well across a variety of input dimensions. This means the same network can predict detection at different resolutions. The network runs faster at smaller sizes so YOLOv2 offers an easy trade-off between speed and accuracy.

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Fig. 8. Example of a figure caption.

Fig. 9. One of the author, Bo's experience of using YOLO: Bad outcome for small objects.

Above modifications makes YOLOv2 a good model in various situations. At low resolutions YOLOv2 operates as a cheap, fairly accurate detector. At 288 × 288 it runs at more than 90 FPS with mAP almost as good as Fast R-CNN. This makes it ideal for smaller GPUs, high framerate video, or multiple video streams. At high resolution YOLOv2 is a state-of-the-art detector with 78.6 mAP on VOC 2007 while still operating above real-time speeds.

However, we have to mention that YOLO is not very powerful in small object detection. One of the authors of this paper, Bo, once tried to use YOLOv2 to detect batteries and bottles in the wild environment. From the left part of Fig. 9, it can be seen that YOLO works (though not very well). However we can see on the right of Fig. 9 that, if we took the picture from another angle, the YOLO can't recognize the object. This somehow reflects the weakness of YOLO: it is not very sensitive in small objects detection and is not accurate especially when rotations are involved. This might be a result of the algorithm YOLO used. YOLO only detects the output of the last convolutional layer, and for small objects, after layers and layers of convolution, its information almost gets totally lost in that layer.

D. MobileNetV2: Light-weight Solution for Mobile Devices

For the fourth essay, the thing interested us most is the depthwise separable convolutions. As we know, the limit of hardware of mobile device restricts the size of a neural network. A promising DNN on server and PC may not operate very well on an outdated iPhone, as we can't expect a mobile ARM SoC will have the same computation capability as big servers. Thus, the implementation of a light-weight network emerges, at the prerequisite that there won't be much loss in accuracy. Some commonly used techniques of DNN compression are pruning, quantization and Huffman coding. But these are sometimes not enough. Researchers are trying to invent a small-device-friendly network that can have at least

similar accuracy as the big and complex networks. That is, MobileNetV2.

For MobileNetV2, the author of this paper utilizes a method called Depthwise Separable Convolution to help to reduce the network size. The outcome is rather amazing: while depthwise separable convolution layers significantly reduced the number of parameters and computation compared to traditional convolution layers, the accuracy doesn't seem to have been influenced.

The basic idea is to replace a full convolutional layer with a factorized version that splits convolution into two separate layers. The first layer is called a depthwise convolution, it performs lightweight filtering by applying a single convolutional filter per input channel. The second layer is a 1×1 convolution, called a Pointwise Convolution, which is responsible for building new features through computing linear combinations of the input channels. Fig. 10 and Fig. 11 might help when trying to understand the mechanism of depthwise Separable Convolution and traditional convolution:

Standard convolution takes an $h_i \times w_i \times d_i$ input tensor L_i, and applies convolutional kernel $K \in R^{k \times k \times d_i \times d_j}$ to produce an $h_i \times w_i \times d_j$ output tensor L_j. Standard convolutional layers have the computational cost of $h_i \cdot w_i \cdot d_i \cdot d_j \cdot k \cdot k$. Depthwise separable convolutions are a drop-in replacement for standard convolutional layers. Empirically, they work almost as well as regular convolutions but only cost: $h_i \cdot w_i \cdot d_i(k^2 + d_j)$.

This significantly reduces the amount of parameters and the multiplication and adding operations. For an efficient depthwise separable convolution, roughly speaking a reduction of factor of k^2 can be achieved. So, no wonder that, compared with its ancestor MobileNetV1, or the famous light-weight network ShuffleNet, MobileNetV2 is actually much smaller in size.

As the test suggests, actually although the number of parameters reduced by a great amount, the accuracy doesn't fall. It achieves almost the same accuracy as VGG16 in ImageSet

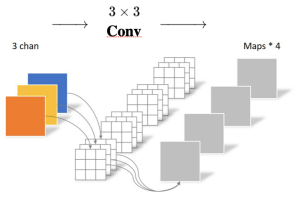

Fig. 10. An intuitive illustration of traditional convolution layer.

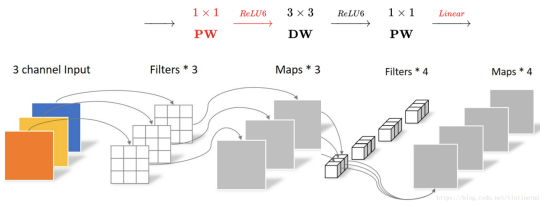

Fig. 11. An intuitive illustration of depthwise separable convolution layers.

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	1/O(1)	1/O(1)	1/O(1)
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	800K	**200K**	600K

Fig. 12. Comparison between MobileNetV1, MobileNetV2, and ShuffleNet

and as Inception in Stanford dog.

Reading this essay really amazes us. The usual belief is a stereotype that convolutional layers are fragile and more sensitive than fully connected ones, and thus researchers should try to remain the intact of convolutional layers when slimming down the model. However, this essay proves that, with appropriate method, we can also slimming down the convolutional layers without harming the accuracy.

E. The Connection between Four Papers

All these papers explore some subdivisions of computer vision. For the first two papers, the commonality between them is that they both try to combine the information we get when we "look" into and "look" outside from the certain region. Paper 1 uses strided convolution method to combine down-sampling pooling results from high and low layers to get more accurate semantic segmentation result, while paper 2 uses ASPP, the kernels with different strides. In both ways, we will not lose too much information when we go deeply in convolution process. Specifically, the first paper, as the foundation of semantic segmentation, opened the door of semantic segmentation. But this doesn't mean the second essay is more trivial, as it uses innovative convolution kernels of different types/sizes and does far better in getting information of objects in different scales.

The focuses of the third and fourth paper are not totally about inventing new techniques. Instead, they put much of their attention at combining and integrate existing methods to improve existing model (YOLOv1 for paper 3, MobileNetV2 for paper 4). The improved version get boosts in either applicability on smaller device, or the number of objects the model can detect simultaneously, i.e., they get either quicker or stronger.

III. Our Opinions in Building Networks for Computer Vision

Computer vision is one of the hottest topics, if not the hottest, in the field of machine learning and artificial intelligence. So far, this paper has discussed some of the most famous papers in this field, and we have covered some common computer vision missions including image classification, image detection, and semantic segmentation.

Here, we would like to give some general, high-level evaluations when the readers and researchers intend to apply methods in the 4 papers we have already discussed above:

Fig. 13. Up: Comparison: The details of the horse are lost during up-sampling.

- FCN, as the foundation of semantic segmentation, does have some flaws. We can see in Fig. 13 that, the details of the leg of the horse are not very clear. The outline of semantic segmentation is a little too smooth. The details of legs were lost during up-sampling. In a nutshell, due to the internal designing flaws, FCN can't recognize details very well, even if we use the most outstanding model (FCN-8s) presented in that paper.
- There are also some potential problems in Atrous convolution, one among them is the gridding effect. That is, since the kernel has "holes" in it, after stacking multiple layers of atrous kernel, we may find there are some pixels not used by the kernel, that is, the information of these omitted pixels is not used in the training the network. In Fig.14 we can see that if the parameters of Atrous convolution are not carefully chosen, we might get a very bad output.
- For the third essay, YOLOv2 is actually a very powerful model, but still has some flaws such as the incompetence in small object detection, which was not solved until YOLOv3. Please refer to Bo's experience in previous parts to learn more.
- The fourth essay represents another direction of the improvement: quicker and lighter. It provides an idea that can be generally applied to networks on light-weight devices to reduce the amount of calculation and parameters, and just as the author illustrated, in practice this won't result in any notable accuracy lost.
- The effectiveness of depthwise separable convolution depends on the size of convolution kernel. As illustrated before, the reduction of computation is at a factor of k^2. Therefore, if we are using big kernels such as 7×7 (which is rarely used after VGG), we can achieve a reduction of approximately $\times 49$. If the kernel size is 3×3 (a much common size nowadays), we can still achieve a reduction of $\times 9$. However, say we are applying a fully convolutional network which consists of many 1×1 kernels (just as in the first essay), such depthwise separable convolution technique will be of no use at all, if not of negative effect.

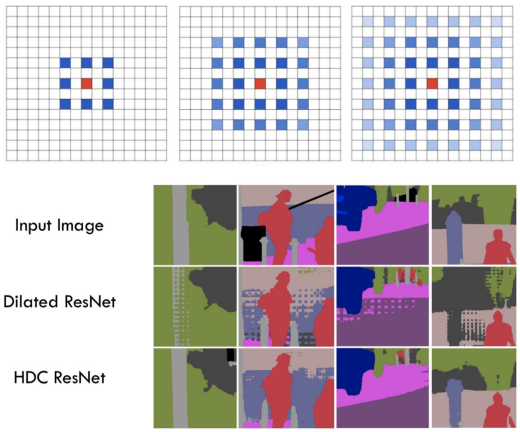

Fig. 14. The gridding effect.

IV. Conclusion

The improvement of neural networks can be roughly divided into two parts: Better & Stronger and Quicker & Lighter. As we can see in the above four papers, the challenges the authors meet or the problems they try to solve is related to how they want to improve the existing model for better adoption of fundamental neural networks in different environmental conditions or limited resources. Let us conclude how improvements are related to purposes in these four papers:

- FCN model – Better & Stronger. It is based on the original CNN model. In order to find the optimal way for usage of CNN in semantic segmentation field, it cases the fully-connected layer in down-sampling to the convolution layer and puts forward up-sampling process, creating a nearly new model for solving semantic segmentation problems.
- ASPP model – Better & Stronger. It is based on the FCN model. To solve multi-scale, resolution-reduced and spatial-information-lost flaws in FCN model, it puts forward the idea of changing the rate of kernels, which is also called Atrous Convolution. Its purpose is to improve the accuracy of semantic analysis results.
- YOLO9000 model – Better & Stronger. Firstly, YOLOv2 is developed by YOLO through the application of a variety of ideas from past works such as anchor boxes. Then, combined with the jointly optimizing detection and classification training method, YOLO9000 model is trained as a real-time framework for detection more than 9000 object categories. It is much stronger than the original YOLO model.
- MobileNetV2 model – Quicker & Lighter. It is based on the existing MobileNet model. The purpose of this series of models is applying CNN on mobile devices with limited resources. By applying concepts such as depthwise separable convolution, MobileNetV2 model successfully slims down the original model while the accuracy does not go down too much.

In conclusion, the improvements of convolutional networks in a specific field are based on existing models, and when the standpoint or modification is approved in this field, it becomes the new foundation. Thus, the development of convolutional

networks is a process of continuous innovation and amelioration. It attempts to identify and correct imperfections in existing models, for better implementation in different contexts and environments.

ACKNOWLEDGMENT

The authors of this paper, Bo Pang and Jieting Chen, would like to express our heartfelt thanks to the instructor of this course, Volodymyr Kindratenko. Thank you for your instruction during the COVID-19.

REFERENCES

[1] Jonathan Long, Evan Shelhamer, Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation" arXiv preprint (2017).

[2] Liang-Chieh Chen George Papandreou Florian Schroff Hartwig Adam. "Rethinking Atrous Convolution for Semantic Image Segmentation." arXiv preprint (2017)

[3] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." arXiv preprint (2017).

[4] Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks" arXiv preprint (2018).

[5] Han, S., Mao, H. and Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[6] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).