Week3 Pre-Report: Overview of Convolutional Neural Networks; VGGNet and ResNet

Euijin Hong[1],
[1]Department of Electrical and Electronic Engineering, Yonsei University.

1 Introduction

Throughout the history of evolution of Deep Neural Networks(DNNs) including Deep Convolutional Neural Network(DCNN), the performance turns out to be closely related to the depth of weight layers. This is due to the increase of the size of the information per pixel. Where the layer gets deeper, the more precise the output would be. Therefore, increasing the number of layers was a deep concern to enhance the performance of a network model. Both VGGNet and ResNet achieved greater performance in terms of accuracy compared to other contempory models by succeeding to increase the depth of layers. Each model were able to solve the existing problems that constrained adding new layers to the model. In this paper, we will show the methodology VGGNet and ResNet implemented in order to solve these problems and increase the depth of layers, and eventually obtained better performance, respectively.

2 VGGNet

2.1 Background

Prior to the advent of VGGNet[7], the deepest CNN was AlexNet[3], which had a depth of 8 layers. However, joining additional layers to it was challenging because the number of free parameters increases as the layer goes deeper. This can cause an overfitting problem or slower the training process. In order to solve this problem, VGGNet suggests a new architecture; the use of small convolution filters.

2.2 Theory and Architecture

The main contribution of VGGNet is the introduction of convolution filters with small receptive field: 3×3 (the minimum size of filter that recieves the notion of location). It starts with the idea that by using multiple small convolution filters instead of using one large convolution filter(5×5 or 7×7), less number of parameters are used while going into deeper layers. Thereby, the network achieved to increase the number of layers without introducing additional model structures such as auxiliary classifier in GoogleNet[8].

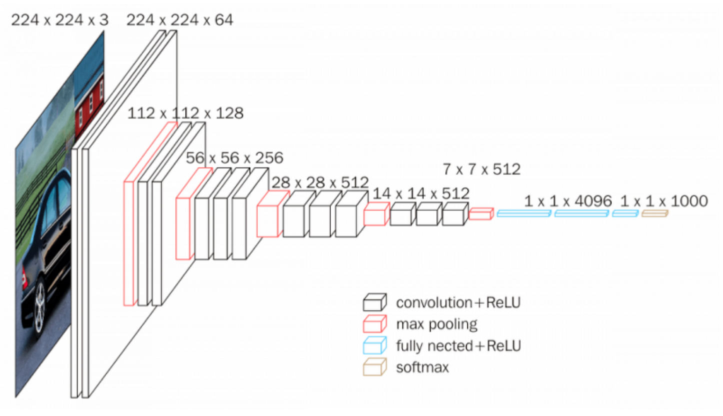

Figure 1: The architecture of VGGNet is comprised of contatenation of following layers: (a) convolution + ReLU (b) max-pooling, which are later followed by fully-connected layers with ReLU and softmax layer at last.

As shown in Figure 1, they used the stack of 3×3 conv. layers(each activated by ReLU function) with stride 1 and padding 1, thus preserving the spatial resolution of the output. The input image passes through this stack of filters, and then spatial-pooled by five max-pooling layers with size 2×2 and stride 2, following some conv. layers.

These combination of convolution and pooling layers are then followed by three Fully-Connected(FC) layers; the first two having 4096 channels and the last having 1000 channels as an output channel(each corresponding to the output class). The final layer is a soft-max layer.

2.3 Configuration and Evaluation

By differing the depth of layers while following the general architecture of VGGNet, the VGGNet research team configured networks (A) - (E). The depth of layers differs from 11(8 conv. + 3 FC layers) to 19(16 conv. + 3 FC layers). The number of channels in conv. layers starts from 64 and reaches until 512. Although the depth has been larger, the number of parameters(144M weights maximum) are not greater than shallower network having larger conv. layer widths and receptive fields[6], which has 144M weights.

The training and evaluation process was based on ILSVRC-2012 dataset. The single model of network (E), which is comprised of 19 weight layers(16 conv. + 3 FC layers), showed the best performance with top-5 validation set error of 8.0% at a single-scale evaluation, 7.5% at a multi-scale evaluation, and 7.1% at a dense and multi-crop evaluation. The VGG team could further decrease the test error to 6.8% by implementing an ensemble of models (D) and (E), combining dense and multi-crop evaluation.

2.4 Discussion

Instead of using the first conv. layer having relatively large receptive field, VGGNet researchers used conv. layers with 3×3 receptive fields on the entire network, which are convolved with the input at every pixel. Here, the stack of multiple 3×3 conv. layers (without max-pool layer between them) has an identical size of effective receptive field to that of larger conv. layers; two stacks of 3×3 layers have 5×5 effective receptive field, and three stacks of these layers have 7×7 effective receptive field. The concept is illustrated in Figure 2.

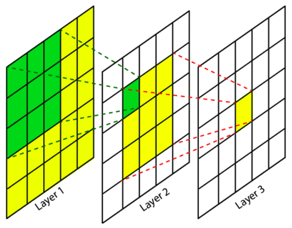

Figure 2: Two stacks of 3×3 convolutional layers have 5×5 effective receptive field.

By doing so, VGGNet achieved two improvements: (1) Making a decision function more discriminative by replacing one non-linear rectification layer into three non-linear rectification layers. (2) Decreasing the number of parameters (e.g. three 3×3 conv. layer stack has $3(3^2C^2) = 27C^2$ weights, while one 7×7 conv. layer requires $7^2C^2 = 49C^2$). These improvements gave significant advantage to VGGNet, thus allowing to stack more layers which lead to better performance.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

3 ResNet

3.1 Background

Although stacking more layers became possible due to VGGNet[7], neural networks that have large number of layers were still difficult to train. This is due to gradient vanishing or exploding problems which hinders convergence of the model. Even if the model began to converge, degradation problem arises with no regard to overfitting. This indicates that optimizing deeper layer models is not an easy matter, which makes our expectation hard to accomplish; to get comparably good or better results by adding new layers in the existing model. Here, ResNet[1] deals with the existing problem by introducing a concept of *deep residual learning* framework.

3.2 Theory and Architecture

The key idea of deep residual learning is, we can train the model more easily by fitting a residual mapping to the stacks of layer instead of directly fitting a desired underlying mapping. While the underlying mapping can be denoted as $\mathcal{H}(x)$, the model lets additional stack of nonlinear layers to fit the mapping of $\mathcal{F}(x) := \mathcal{H}(x) - x$. By doing so, the original mapping can be set into $\mathcal{F}(x) + x$, which is expected to be easily optimized than the unreferenced original mapping. Even though the identity mapping is optimal, it will still be easier for us to set the residual by zero than fitting an identity mapping when introducing additional stack of layers.

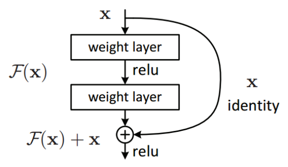
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
x
identity
weight layer
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3: A building block of residual learning.

The formulation of mapping $\mathcal{F}(x) + x$ can be implemented as feed forward neural network by *shortcut connections*, which can be depicted as Figure 3. The structure is simply designed by skipping one or more layers. Here, shortcut connections perform identity mapping where their outputs are added to the output of the original stacked layers. The identity shortcut connection does not add extra complexity or parameters, since it is a linear addition. End-to-end training is possible via SGD with backpropagation, and can be easily implemented.

The building block shown in Figure 3 can be defined as follows:

$$y = \mathcal{F}(x, \{W_i\}) + x \qquad (1)$$

While x and y are input and output vectors of the considered stack of layers respectively, the function $\mathcal{F}(x)$ is a residual mapping to be learned. The form of function $\mathcal{F}$ is flexible, in which can represent multiple linear or convolutional layers. For instance, in Figure 3 two layers are connected, thus mapping would be $\mathcal{F}(x) = W_2\delta(W_1x)$ in which δ indicates a ReLU[4] function while biases are omitted for simplification. Shortcut is connected for implementing $\mathcal{F} + x$ and added element-wise. After $\mathcal{F} + x$, second linearity(ReLU) is adopted.

To generalize the expression to the case where dimension of x is different from that of $\mathcal{F}$, performing linear projection is necessary:

$$y = \mathcal{F}(x, \{W_i\}) + W_s x \qquad (2)$$

W_s is used only when matching dimensions, since the experiment is focusing on the sufficiency of identity mapping for dealing the degradation problem.

For deeper nets, deeper bottleneck design is implemented because of limited training time as shown in Figure 4.

By stacking three layers, each corresponding to 1×1, 3×3 and 1×1 convolution, respectively. 1×1 layers are responsible for reducing and then increasing dimensions, leaving the 3×3 layer a bottleneck with smaller

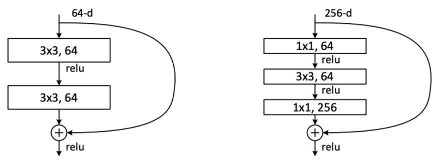

Figure 4: A deeper residual function. Left: a building block of 5×5 feature map for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

input and output dimensions. As Figure 4 shows, both designs have similar time complexity. Here, the parameter-free shortcut plays a crucial role because if linear projection is performed instead, the model size and time complexity will be doubled.

3.3 Configuration

By comprehensive experiment, the ResNet team found that the extremely deep ResNets are easy to optimize while showing less training errors as the depth increases, thus showing better accuracy gains. In order to assess the effectiveness of ResNet, the team defined two models for ImageNet as follows; Plain network and Residual network.

Plain network is based on the basic idea of VGGNet[7], which mostly has 3×3 conv. layers following two rules: (i) the layers have the same number of filters for the same output feature map size (ii) the number of filters is doubled in order to preserve the time complexity per layer, when the feature map size reduced into half. Here, downsampling is performed by conv. layers that have 2 strides. At the end of the network, average pooling to 1000 FC layers is implemented.

Residual network is based on the plain network, but additionally inserting shortcut connections thus turns the network into residual counterpart. Identitiy shortcuts can directly be used when the input and output dimensions are the same. When dimensions increase, the shortcut can be introduced by (a) using shortcut that still perform identity mapping while extra zero padding added (b) the projection shortcut in Equation(2) is performed by 1×1 convolution. For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

3.4 Evaluation

The team then evaluated the ResNet model by using ImageNet 2012 classification dataset[5] as validation and test set, also obtaining top-1 and top-5 error rates. At the plain network, 34-layer net showed higher training and validation error by degradation problem, which the team suggests that it is not due to gradient vanishing, because the models were trained with BN[2]. However, 34-layer ResNet showed better performance(by 2.8%) in validation set and lower training error as well. This shows that degradation problem is well addressed by residual learning. Also, compared to the plain counterparts, the 34-layer ResNet reduced top-1 error by 3.5%. Lastly, the 18-layers plain/residual net shows comparably accurate results, but 18-layer ResNets converge in a faster rate. This case shows that ResNet provides faster convergence thus easing the optimization.

The team also constructed a deeper layer ResNet(50, 101, 152 and so on), showing more powerful performance(minimum top-5 error: 4.49%) with no degradation problems. They could further combine six models of different depth to form an ensemble, and obtained optimal error: 3.57%.

4 Conclusion

To sum up, VGGNet and ResNet contributed to build deeper neural networks by introducing fundamental structural adjustments. For VGGNet, it was multiple concatenation of smaller size convolutional layers, while for ResNet it was proposal of deep residual learning. The ideas has been influential in the field of deep learning, affecting numerous subsequent researches.

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[4] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[6] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.