Proceedings of ICSP2018

How Functions Evolve in Deep Convolutional Neural Network

Ling Hu[1,2], Shuhui Wang[1], Liang Li[1], Qingming Huang[1,2]
[1]*Key Laboratory of Intelligent Information Processing, ICT, CAS, Beijing 100190, China*
[2]*School of Computer and Control Engineering, UCAS, Beijing 100049, China*
Email: ling.hu@vipl.ict.ac.cn, wangshuhui@ict.ac.cn, liang.li@vipl.ict.ac.cn, qmhuang@ucas.ac.cn

***Abstract*—Deep Convolutional Neural Network (CNN) has been successful in various visual applications. Unfortunately, the mechanism to explain how CNN actually learns and how it works is not clearly revealed and understood. In this paper, we propose analytics for CNN from the functional perspective by constructing three simple yet effective measurements on the convolutional filters. It quantitatively measures the change of convolutional filters in training process, which can be used to explain the learning mechanism of different CNN architectures. By experimental facts on representative VGGNet and ResNet, we find that 1) the change magnitude of lower layer parameters is greater than that of upper layers; 2) lower layers are closer to raw data and wash out redundancy, and higher layers learn more useful information of the data; 3) redundant filters do exist in typical CNN; and 4) the functional behaviors of VGGNet and ResNet can explain the intrinsic difference of plain CNN and residual CNN architectures. Our analytical framework and observations can facilitate future research to understand the learning mechanisms of a wider range of CNN family comprehensively.**

***Keywords*—convolutional neural network, interpretability, quantitative metrics.**

I. Introduction

As an important member of Deep Neural Network (DNN) family, Deep Convolutional Neural Network (CNN) has been successful in many applications, such as image recognition, object detection, image captioning and even in sequence-to-sequence learning [1]. For image/video related tasks, CNN is leveraged as an efficient visual feature extractor, and has shown its strong power in learning features end-to-end. However, the mechanism to explain how DNNs actually learn and how they work is not clearly revealed and understood. It is found that they may imitate the human brains [2], [3], yet the exact mechanism of human brains and whether DNNs really approximate human brains are still unclear. Beyond the above-mentioned visual analysis tasks, understanding how CNN works is of great values for other applications involving consequent decision making, *e.g.*, visual knowledge extraction, medical image diagnosis and automatic driving.

Since modern deep neural network consists of multiple layers, the key to unveil the black box is to understand the behavior inside the network layer by layer, instead of in an end-to-end way. Deconvolution is proposed in [4], [5] to investigate the network by observing images reconstructed from feature maps. It demonstrates that the first two layers capture low-level features such as edges and blobs, and higher layers represent more like semantic objects. However, investigating feature maps depends on the input images and only explores the individual output of each layer. On the other hand, there has been effort in explaining working mechanism for DNN from information theoretic perspectives [17], [18]. The mutual information between layer-wise representations and inputs as well as labels is measured to show how DNN fits the data and removes redundant information during training process.

Different from existing studies, we consider analytics for CNN from the functional perspective by constructing a set of simple yet effective measurements on the convolutional filters. It quantitatively measures the change of convolutional filters in training process, which can be used to explain the learning mechanism of CNN beyond visualization-based qualitative analysis. Moreover, the learning behavior of individual convolution filters in each layer can also be measured and analyzed, which provides a complementary way for existing theoretic analysis methods. Specifically, we propose three quantitative metrics in functional Hilbert space, *i.e.*, measurement of the temporal changes of convolutional filter via Euclidean distance, measurement of the ratio of convolutional filters with changes large than a given threshold in magnitude, and measurement of the proportion of channels in a CNN filter whose changes are all below a predefined threshold. The metrics are capable of demonstrating how convolutional filters evolve during training. Compared with other works on network interpretability, our method is simple and efficient. The evaluation thus can be done offline, without adding much overhead to the training process like [9], while other works (e.g., [19]) have to monitor the network during training by adding probe components to the network.

By quantitatively observing changes of internal parameters in typical CNN models (e.g., VGGNet and ResNet), we have the following observations. **First**, the change magnitude of lower layer parameters is always greater than that of upper layers. This phenomenon might be counter-intuitive, since it is known that lower layers learn more general features that are transferable and stable [4], [6]. But in our observations, their changes are greater than those of higher layers. **Second**, we show that lower layers are closer to raw data and wash out redundancy, and that higher layers learn more useful information of the data. This observation also supports the Information Bottleneck Theory [17], [18]. **Third**, we observe that there do exist redundant filters, *i.e.*, their parameters are rarely updated. This observation provides guidance on how to prune

978-1-5386-4673-1/18/$31.00 ©2018 IEEE

a large CNN layer-by-layer towards commercial scale real-time applications. **Moreover**, we observe that the functional behaviors of VGGNet and ResNet are different, and such difference can be used to explain the intrinsic mechanisms of plain CNN and residual CNN. We believe our analytical framework and observations can facilitate future research to understand the learning mechanisms of CNN family more comprehensively.

II. Related Work

Previous works on network interpretability categorized into visualization [5], theoretical proof and quantitative evaluation standards. The most direct way to facilitate understanding the network is to *visualize the layer-wise feature responses*. Zeiler et al. [5] proposed de-convolution network to reconstruct the input according to the representations inside the network. Then de-convolution network is leveraged as a diagnostic tool to modify CNN architecture and get state-of-the-art result in ImageNet classification task [4]. Two visualization tools are proposed in [7] for features and activations but without explaining the intrinsic mechanism.

The basic theoretical foundation of CNN is not well established. Since the empirical minimization problem is non-convex, it is hard to analyze the convergence property during training process. theoretic analyses are conducted on simplified network structure (e.g., one- or two-layered network structure) by making assumptions on how data is generated from a certain distribution [11]–[13]. For example, based on the Gaussian distribution assumption, how gradient descent works and how the networks converge are investigated.

Criteria for evaluation of the representation learning ability of the network are also investigated. For example, the method proposed in [10] quantifies the interpretability of CNN representations by evaluating the alignment between individual hidden units and semantic concepts. It is a good criteria for representations in CNN but it requires supervision beforehand. Zhang et al. [8] model the relationship between representations of different attributes inside the CNN and compares it with ground-truth attribute relationship to discover the failure modes and blind spots of CNN. A new method is proposed in [9] using linear classifiers referred to as "probes", to better understand the roles and dynamics of the intermediate layers. Koh et al. [19] utilize influence functions from robust statistics, to trace the models prediction process and identify training points that are most responsible for a given prediction. The mutual information between input, layer-wise representation and output is used to explain the learning mechanism of deep neural networks [17], [18].

III. Analytics

A. Visualizing Method

To observe the evolution of the filters, it is intuitive to see its values with the increase of the training step. However, as the filters are three-dimensional tensors, it is difficult to observe interesting patterns from those raw real-valued numbers. Visualization of the filters is more intuitive, but it is problematic that except the first convolutional layer, the number of channel of a filter is always larger than three which cannot be visualized as three-channel RGB picture. An alternative way is to visualize individual channel of a filter as a single channel grey picture. However, it is hard to observe subtle changes on some individual channels of a filter because of the insensitivity of human eyes. Therefore we just visualize the first convolutional layer as RGB pictures. Fig. 1 shows the visualization of two filters in the first convolutional layer of VGGNet.

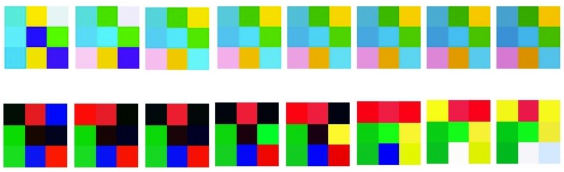


Fig. 1. Evolution of 3x3 filters in the first convolutional layer of VGGNet-16. Each filter per row and depicted from left to right according to the training steps.

Fig. 1 is the initial evolution process of the first convolution layer of VGGNet-16, with respect to the number of training steps as (0, 10, 20, ..., 70). In Fig. 1, darker color indicates larger value. For simplicity, only two filters of some beginning steps are shown here. We can see that some filters such as the one of the first row change gently during training, while other filters like the one of the second row change relatively large, indicating that different filters in one layer have different evolution trends. It is necessary to take the different behaviours of different filters into consideration to define the quantitative measurements to reveal the evolution of filters. Besides, we find that not all the values are updated in a filter, which may reveal some particular phenomenon of filters during training. Therefore, we propose to measure the proportion of values in a filter that change significantly. Visualizing methods only allow us to directly observe the changes of the filter with naked eyes, which is far from being rigorous and effective, and easy to miss some characteristics that are not easy to observe. However, the phenomena observed here can guide us later in the specific design of quantitative metrics.

B. Quantitative Metrics

The convolution operation by the filters can be thought as a function $f_t(\cdot)$ with respect to the input, in which t stands for training step. After a training step, the filter is updated according to some optimization algorithms, and correspondingly becomes another function $f_{t+1}(\cdot)$. To measure the difference between the two filters, we take these filters, or the function itself as the variable, and the difference between them is thus a function with respect to the same filter at different training steps: $D(f_t) = difference(f_t(\cdot), f_{t+1}(\cdot))$. Mathematically speaking, the difference function $D(\cdot)$ is a functional as it maps from functions to certain real value. Now all the filters, or as we note as functions, span a high-dimension Hilbert space, and all the filters functions are embedded as points in

the space. Since this, we measure the difference of filters by the distance of the points in this Hilbert space. Since Hilbert space is a complete inner product space, we naturally consider Euclidean distance to measure the distance of two filters. The Euclidean distance of the same filter at different training steps is calculated as follows:

$$D_t = \frac{1}{N}\sum_{i}^{N} \parallel \boldsymbol{f}_{t+1}^{i} - \boldsymbol{f}_{t}^{i} \parallel_2^2, \tag{1}$$

where $\boldsymbol{f}_t^i \in \mathbb{R}^{W\times H\times C}$ represents i_{th} convolutional filter in a certain layer at the t_{th} training step. $F_t = (\boldsymbol{f}_t^0, \boldsymbol{f}_t^1, \cdots, \boldsymbol{f}_t^N)$ represents filters in a layer. D_t represents the distance between filters F_{t+1} and F_t.

As we can see from Fig. 1, not all values in a filter change significantly, so we design an extra measurement to represent the proportion of values in a filter that change significantly, in another word, larger than a given threshold. We define the metric as follows:

$$S_t = \frac{1}{N}\sum_{i}^{N} \frac{1}{WHC}\sum_{j,k,l} \mathbb{I}(\mid \boldsymbol{f}_{t+1}^{i}(j,k,l) - \boldsymbol{f}_{t}^{i}(j,k,l) \mid > \tau), \tag{2}$$

where $f_t^i \in \mathbb{R}^{W\times H\times C}$, and $\mathbb{I}$ represents the indicator function. This metric S_t can quantitatively represent the severity of the evolution of the filters. The threshold τ is set to the initial learning rate.

At the same time, we would like to see how many channels in a filter do not update significantly - if this occurs at the beginning and through the whole training process, it implies that these channels do not learn much. To be specific, we define Z_t as the ratio of channels in a filter that the total change of the whole channel is less than the chosen threshold:

$$Z_t = \frac{1}{N}\sum_{i}^{N} \frac{1}{C}\sum_{l} \mathbb{I}(R_{i,l} == 0), \tag{3}$$

where $R_{i,l} = \sum_{j,k} \mathbb{I}(\mid \boldsymbol{f}_{t+1}^{i}(j,k,l) - \boldsymbol{f}_{t}^{i}(j,k,l) \mid > \tau)$, Z_t quantitatively represents the ratio of channels whose magnitude of changes are nearly zero in the filters. The threshold τ is also set to the initial learning rate.

IV. Validation and Extension

To evaluate our proposed metrics and further use them to reveal the evolution process of the network, we conduct initial experiments. On the observations we obtain, we make assumptions and do extensive experiments to verify them.

A. Models and Dataset

We use a subset of ImageNet ILSVRC dataset [16], which consists of 100,000 images classified in 1000 categories, for the reason that we have to record changes of the filters every 10 steps for analysis, and that training large dataset from scratch is both time and storage consuming. To explore the evolution of models with obvious and significant different architectures, we experiment on two typical models that ever achieved state-of-the-art results on ImageNet: VGGNet-16 [14] and ResNet-50 [15]. The detailed architectures of the two models we experiment on are shown in Table I and Table II where $3\times3, 64$ means kernel size is 3×3 and output channel is 64.

TABLE I
VGGNet-16 architecture.

block1	block2	block3	block4	block5
$3\times3, 64$	$3\times3, 128$	$3\times3, 256$	$3\times3, 512$	$3\times3, 512$
$3\times3, 64$	$3\times3, 128$	$3\times3, 256$	$3\times3, 512$	$3\times3, 512$
		$3\times3, 256$	$3\times3, 512$	$3\times3, 512$

TABLE II
ResNet-50 architecture.

block1	block2		block3		block4		block5	
$7\times7, 64$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}$	$\times 3$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}$	$\times 4$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}$	$\times 6$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}$	$\times 3$

Instead of depicting all the layers here, we select several layers that are representative. We follow the principal that layers selected should span lower layers, upper layers and those in middle. Specially, we select five convolutional layers from bottom to top in VGGNet-16 and ResNet-50 respectively. The layers we select are shown in Table III.

TABLE III
Layers selected to be analyzed in VGGNet and ResNet

	VGGNet-16	ResNet-50
conv1	block1/conv1_1	block1/conv1
conv2	block2/conv2_2	block2/unit_3/conv2
conv3	block3/conv3_3	block3/unit_4/conv2
conv4	block4/conv4_3	block4/unit_6/conv2
conv5	block5/conv5_3	block5/unit_3/conv2

B. Training Details

We train the models on a single GPU (GeForce GTX 1080Ti). In the training process, batch size and initial learning rate are set to 64 and 2×10^{-4}. We halve the learning rate after 25 epochs, and then halve it every 10 epochs. We use Adam as the optimization algorithm. We save the model parameters every 10 steps for more fine-grained analysis.

C. Filter Evolvement

First, we apply our proposed metrics on these saved model parameters and record the results. On one hand, we compare the evolution process between different layers in one model. On the other hand, we compare the evolution process between the two models: typical plain CNN architecture (e.g. VGGNet) and typical residual CNN architecture (e.g. ResNet). For each model, we select several certain layers (e.g. the 1st, 4th, 7th, 10th and 13th layer of VGGNet-16) as mentioned above, and compute the quantitative changes of the filters in these layers. Then we plot the records with respect to training steps in a poly-line chart so as to make a clear quantitative comparison, as shown in Fig. 2(a).

(a) D_t record of VGGNet and ResNet. (b) I_t record of VGGNet and ResNet. (c) Z_t record of VGGNet and ResNet.

Fig. 2. The three metrics we proposed on VGGNet-16 and ResNet-50. Each chart has five polylines. The five polylines are sampled from low layer to high layer of CNN. There are legends in the chart showing the layer of corresponding curve. For details, see Table III.

The sudden drop in the figures is due to the decay of learning rate, and the oscillation is a common characteristic of every layer, where the difference in amplitude is due to the difference range of value itself. In Fig. 2(a), we can observe obvious stratification in VGGNet that filters of lower layers always change relatively larger than those of upper layers. The stratification in ResNet is not as obvious as VGGNet, but it can still be found that the top layer has the least change. Taking these two models for comparison, we find huge difference between plain CNN and residual CNN: different layers in plain CNN have very different behaviours as each layer performs its own unique duty, while that is not obvious in residual CNN since the information is well spread to all layers due to the residual connection.

We then depict the other two statistic measurements in Fig. 2(b) and Fig. 2(c), where we can also observe the similar phenomena as above. Fig. 2(b) demonstrates the change intensities of every layer that the ratio represents how many parameters in a filter that change significantly. Not all the parameters in a filter are changing since the ratios are not close to 1, which can also be seen in Fig. 1. Comparing the figure of VGGNet and ResNet, stratification is also more obvious in VGGNet than ResNet, similar to the distance measurements discussed above. In Fig. 2(c), the metric Z_t represents the ratio of channels that change slightly. In the figure of VGGNet, we can observe that the metrics of higher layers are found to be closer to 1 through the entire training, which may suggest that higher layers contain larger proportion of redundant parameters. While in ResNet, the Z_t is small at the beginning and gradually increase to 1, suggesting that it has less redundant parameters and is more robust. The Z_t is complementary against I_t, so the figures of the two statistics metrics are against each other.

In summary, by observing the records of three metrics on two models, we obtain two points, and we make reasonable assumptions:

- Typical deep models like VGGNet is more capable of washing out redundant information than models with residual connection like ResNet, so that the changes of different layers in VGGNet behave more differently while residual CNN is more robust.
- Not all the filters are updating or learning so that some filters are redundant or dead.

In section IV-D and section IV-E, we experimentally verify our assumptions on the observations and make extensive discussions.

D. Information Filtering

In VGGNet, we have found that during training, the magnitude of changes of filters in lower layers are always greater than that in upper layers, as discussed in section IV-C. This could be in that raw data input contains much redundant information and noise, so that lower layers are more affected. When raw data is fed into the network, the redundant information is believed to be firstly removed through multiple lower layers. Then upper layers learn the general distribution of the data more effectively. Mathematically speaking, it can be reasoned from the formula that calculates the gradient of convolutional kernel, as shown below:

$$z^l = a^{l-1} * W^l + b, \tag{4}$$

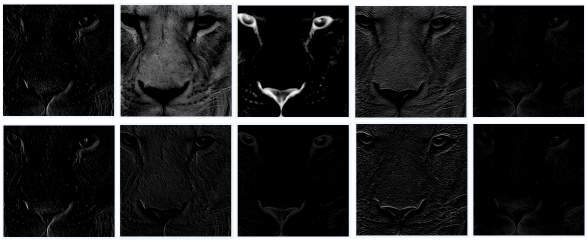
$$\frac{\partial J(W,b)}{\partial W^l} = \frac{\partial J(W,b)}{\partial z^l}\frac{\partial z^l}{\partial W^l} = \delta^l * R_{180}(a^{l-1}), \tag{5}$$

where l is denoted as the layer index, a is the input, z is the output, J is the loss function, $*$ is the convolution operation and R_{180} means rotating 180 degrees.

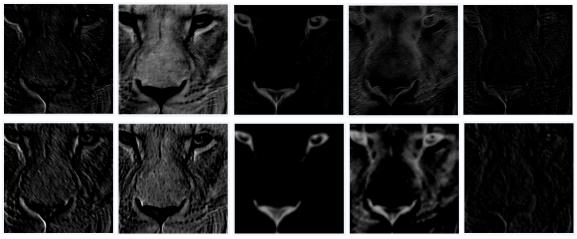
According to the formula above, the update of the convolution kernel is dependent on the gradient. As we can see, the

gradient denoted as is related with the input a. On another hand, the commonly used optimization method SGD also causes a lot of randomness with respect to different input data. However in ResNet, the residual connection enhances the information flow and the gradient flow through the whole network, so that the gap of the update of the filters between different layers is much smaller.

Here we randomly select some images as input, and operate convolution on these input using the trained convolutional kernel of the first layer. We compare the feature maps after the ReLU between different training steps, and show one example in Fig. 3.



(a) Feature map generated by the first layer of VGGNet



(b) Feature map generated by the first layer of ResNet

Fig. 3. Feature maps of different filters in the first convolutional layer in VGGNet and ResNet. From left to right, there are five feature maps generated by five randomly chosen filters operating on a tiger image. From top to bottom, the first row is at training step 0, and the second row is at training step 60000 (40 epochs)

From Fig. 3 (a), we can see that after training for many epochs, the features are refined through the updates of parameters in VGGNet. But there are some filters that learn little, like the last column. From Fig. 3 (b), we can see that features change slightly after many epochs in ResNet, which may due to that the information of feature maps of the first layer are spreading to every higher layer in ResNet through residual connections.

We can draw the conclusions that typical plain CNN is more capable of washing out redundant information than residual CNN while CNN with residual connection is more robust because: 1) residual connections make behaviors of different layers less different, and 2) 1×1 convolutional kernel is not utilized in plain CNN that different feature channels tend to deliver more redundancies that need to be squeezed out, which is in contrast widely used in ResNet. However, CNN with residual connection is more robust since all the information from input data are spread to higher layer.

E. Redundant Filters

Compare the two models, we can observe that many filters change by near zero in VGGNet, which could be that they are not working or learning anything. While in ResNet, there are no filters that change by near zero, which suggests that through the residual connection, all the filters get effective information so that they can always learn something.

We also compare D_t of different filters in each layer in Fig. 4. From the figure, strong inconsistence of the distributions of filters among different layers can be found. In sub-figure of VGGNet, the polyline of conv5 contains more filters that change slightly or even by nearly zero, which may learn nothing. Those filters change a lot are learning effective features. However in ResNet, none of filters changes by nearly zero and all the filters of each layer change in a narrow range. We speculate that more redundant filters are in typical plain CNN and filters in residual CNN are more robust. Based on this observation and our assumption, we design experiments to verify this. In concrete, we set the values of those redundant filters we have observed to 0 by hand and then use the model to predict on the validation set, and we get not significant decline in performance. The results are shown in Table IV.

It can be found from Table IV (a) that in VGGNet, our approach almost does not decline, which indicates that this way of choosing redundant filters is appropriate. For randomly zeroing, filters closer to the bottom drop smaller (e.g. conv1 drops 0.0272%, smaller than conv2, conv3 and con5), suggesting that filters in bottom layers are more robust. In contrast, when randomly zeroing filters in ResNet as Table IV (b), filters closer to the bottom drop larger (0.431% of conv1) while the overall magnitude of drop of the whole network is very small. Taking conv5 for comparison, when randomly zeroing out filters, ResNet drops only 0.0028% while VGGNet drops 0.1988%, suggesting that filters in ResNet are more robust than VGGNet. Since the first layer sees the original data input without any residual connection before itself, it is affected more. Subsequent layers in ResNet drop more slightly due to their residual connections with the first layer, so that the drop of different layers do not have obvious differences, and the gap between randomly zeroing and selectively zeroing is not large.

Through experiments above, we draw the conclusions that there does exist redundant filters in CNN. And in particular architecture as VGGNet, higher layers have more redundant filters than lower layers and in ResNet filters in each layer are more robust. In Table IV, ratio means the ratio of filters in a layer that are set to be zero, which is selected according to the Z_t record. Random represents randomly choosing filters of the ratio. Our method is to find the filters change near zero.

V. Conclusion

We propose analytics for CNN from the functional perspective by constructing three simple yet effective measurements on the convolutional filters. It can be used to explain the learning mechanism of different CNN architectures. We find that 1) the change magnitude of lower layer parameters is

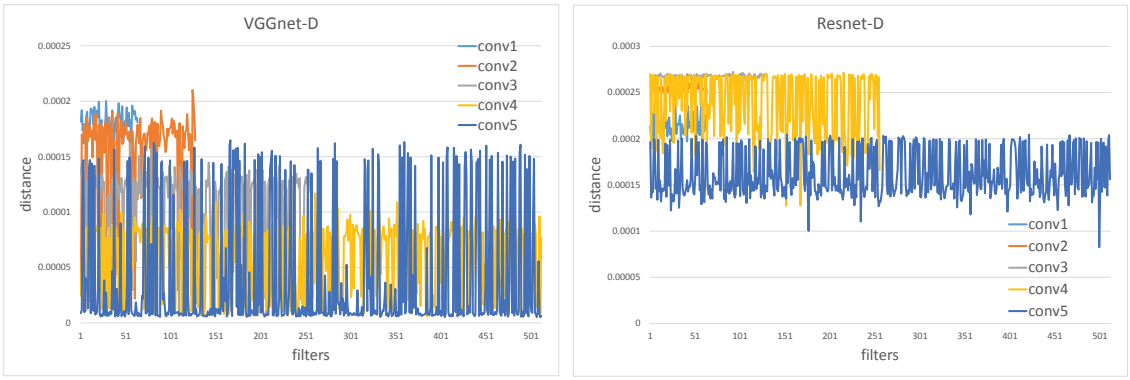


Fig. 4. Left chart is the D_t record of all the filters in a layer of VGGNet by averaging all the training steps, and right chart is the D_t record of the filters in ResNet. Different layers have different numbers of filters

TABLE IV
VALIDATION PERFORMANCE DROP FOR SETTING SOME FILTERS TO ZERO.

(a) Validation performance drop in VGGNet-16

	Ratio	Random	Our method
conv1	7/64	0.0272%	0.0652%
conv2	15/128	0.0292%	0%
conv3	68/256	0.0522%	0%
conv4	45/512	0.0149%	0%
conv5	272/512	0.1988%	0%

(b) Validation performance drop in ResNet-50

	Ratio	Random	Our method
conv1	7/64	0.431%	0.2569%
conv2	15/64	0.0159%	0.0299%
conv3	68/128	0.0317%	0.0729%
conv4	45/256	-0.01%	0.0001%
conv5	272/512	0.0028%	0.0005%

always greater than that of upper layers; 2) lower layers are closer to raw data and wash out redundancy, and higher layers learn more useful information from data; 3) redundant filters do exist in typical CNNs; and 4) the functional behaviors of VGGNet and ResNet can explain the intrinsic difference of plain CNN and residual CNN architectures. In the future work, we consider to construct a complete theoretic foundation for analyzing the functional behavior of CNNs more comprehensively, and extend our analytical framework on more types of DNN models.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China: 61672497, 61332016, 61620106009, 61650202, 61771457, 61732007 and U1636214, in part by National Basic Research Program of China (973 Program): 2015CB351802 and in part by Key Research Program of Frontier Sciences of CAS: QYZDJ-SSW-SYS013.

REFERENCES

[1] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. “Convolutional sequence to sequence learning.” In ICML, pages 1243-1252, 2017.
[2] Hongyin Luo, Jie Fu, and James Glass. “Bidirectional backpropagation: Towards biologically plausible error signal transmission in neural networks.” CoRR, abs/1702.07097, 2017, unpublished.
[3] Chang Le and Doris Y. Tsao. “The code for facial identity in the primate brain.” In Cell, 169(6):1013, 2017.
[4] Matthew D. Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks.” In ECCV, pages 818-833, 2014.
[5] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. “Adaptive deconvolutional networks for mid and high level feature learning.” In ICCV, pages 2018-2025, 2011.
[6] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. “How transferable are features in deep neural networks?” In NIPS, pages 3320-3328, 2014.
[7] Jason Yosinski, Jeff Clune, Anh MaiNguyen, Thomas J. Fuchs, and Hod Lipson. “Understanding neural networks through deep visualization.” In ICML workshop, 2015.
[8] Quanshi Zhang, Wenguan Wang, and Song-Chun Zhu. “Examining CNN representations with respect to dataset bias.” In AAAI, 2018.
[9] Guillaume Alain and Yoshua Bengio. “Understanding intermediate layers using linear classifier probes.” In ICLR, 2016.
[10] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. “Network dissection: Quantifying interpretability of deep visual representations.” In CVPR, pages 33193327, 2017.
[11] Yuandong Tian. “An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis.” In ICML, pages 34043413, 2017.
[12] Simon S. Du, Jason D. Lee, Yuandong Tian, Barnabs Pczos, and Aarti Singh. “Gradient descent learns one-hidden-layer CNN: don't be afraid of spurious local minima.” In ICML, 2018.
[13] Yuanzhi Li and Yang Yuan. “Convergence analysis of two-layer neural networks with relu activation.” In NIPS, pages 597607, 2017.
[14] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” In ICLR, 2015.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Identity mappings in deep residual networks.” In ECCV, pages 630-645, 2016.
[16] Jia Deng, Wei Dong, Richard Socher, LiJia Li, Kai Li, and Fei-Fei Li. “Imagenet: A large-scale hierarchical image database.” In CVPR, pages 248-255, 2009.
[17] Ravid Shwartz-Ziv and Naftali Tishby. “Opening the black box of deep neural networks via information.” CoRR, abs/1703.00810, 2017, unpublished.
[18] Naftali Tishby and Noga Zaslavsky. “Deep learning and the information bottleneck principle.” In ITW, pages 1-5, 2015.
[19] Pang Wei Koh and Percy Liang. “Understanding black-box predictions via influence functions.” In ICML, pages 1885-1894, 2017.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.” In ICCV, pages1026-1034, 2015.