Interpreting and Disentangling Feature Components of Various Complexity from DNNs: Supplementary Material

Jie Ren [*1] **Mingjie Li** [*1] **Zexu Liu** [1] **Quanshi Zhang** [1 2]

1. Architecture of the decomposer net

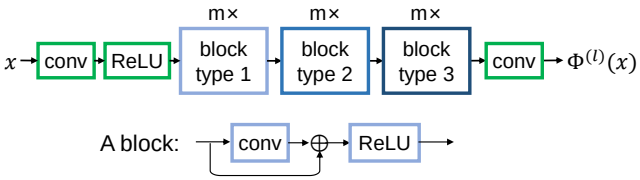

Figure 1. The architecture of the decomposer net.

2. Visualization of feature components

This section shows more visualization results in Section 4 in the paper by visualizing feature components decomposed from the target feature in Figure 2. Given a pre-trained VGG-16 and input images in the CUB200-2011 dataset, we decomposed and visualized feature components of different orders in Figure 5 in the paper. We took the feature in the `conv4-3` layer (with the size of $28 \times 28 \times 512$) as the target feature $f(x)$. Then, we decomposed the target feature and visualized the feature map of a random channel in $f(x)$, and the corresponding channel in $c^{(l)}(x)$ and $\Phi^{(l)}(x)$. We found that low-complexity feature components usually represented the general shape of objects, while high-complexity feature components corresponded to detailed shape and noises.

3. Details of network compression and knowledge distillation

This section introduces more details about the network compression and knowledge distillation in Exp. 3.

Network compression based on (Han et al., 2015): We learned a compressed DNN by pruning and then quantization. In the pruning phase, we pruned the DNN with the sensitivity rate 1.0 for all convolutional and fully connected layers. We iteratively pruned the DNN and retrained the weights. The number of this iteration was 300, and the weights were retrained for 20 epochs in our experiments. The weights in the pruned DNN was retrained for 100 epochs. For example, as a result, ResNet-32 trained on CIFAR10-1000 had an overall pruning rate of 5.88× without affecting the accuracy significantly. To compress further, we quantized weights in the DNN. For weights in convolutional layers, we quantized them to 8 bits, while for weights in fully connected layers, we quantized them to 5 bits. In this way, we obtained the compressed DNN.

Knowledge distillation based on (Hinton et al., 2015): To obtain the distilled DNN, we used a shallower DNN to mimic the intermediate-layer feature of the original DNN. For simplicity, we let the distilled DNN have the same architecture with the decomposer net with seven ReLU layers. The distilled DNN usually addressed the problem of over-fitting. For example, the distilled DNN based on ResNet-32 on CIFAR10-1000 had a 3.48% decrease in testing accuracy, without affecting the

[*]Equal contribution [1]Shanghai Jiao Tong University. [2]Quanshi Zhang is the corresponding author. He is with the John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, China. Correspondence to: Quanshi Zhang <zqs1022@sjtu.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

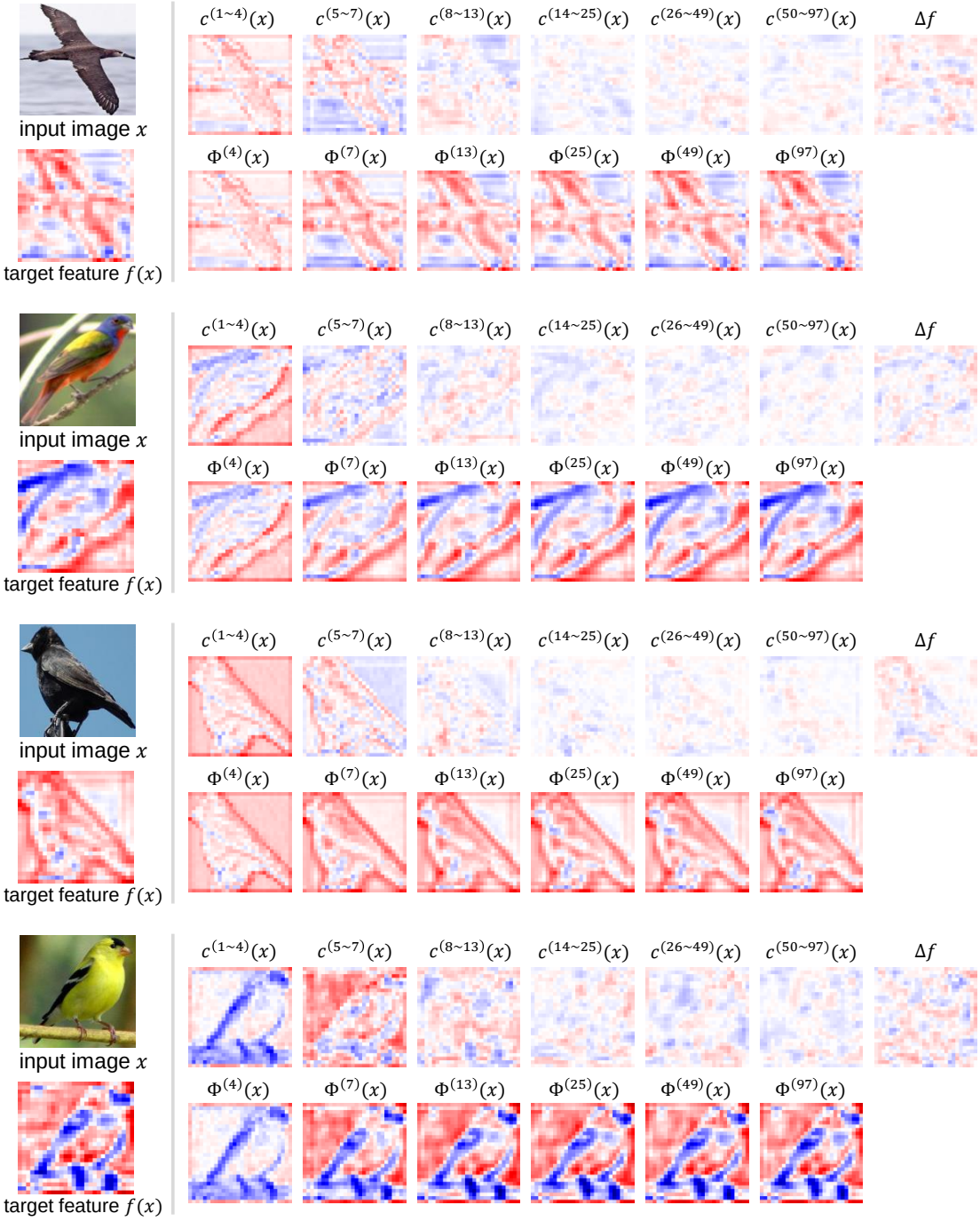

Figure 2. Visualization of feature components of different complexity orders on the CUB200-2011 dataset.

training accuracy significantly.

4. Strong relationship between feature complexity and performance of DNNs

This section introduces more details about the regression task in Exp. 4.

To investigate the relationship between the feature complexity and the performance of DNNs, we learned a regression model, which used the distribution of feature components of different complexity orders to predict the performance of DNNs. For each DNN, we used decomposer nets with $l_1 = 4, l_2 = 7, l_3 = 13, l_4 = 25$ to decompose $\Phi^{(l),\text{reli}}(x)$ and $\Phi^{(l),\text{unreli}}(x)$. Then, we calculated $Var[\Phi^{(l_i),\text{reli}}(x)-\Phi^{(l_{i-1}),\text{reli}}(x)]/Var[f(x)]$ and $Var[\Phi^{(l_i),\text{unreli}}(x)-\Phi^{(l_{i-1}),\text{unreli}}(x)]/Var[f(x)]$, thereby obtaining an 8-dimensional feature to represent the distribution of different feature components. In this way, we learned a linear regressor to use the 8-dimensional feature to predict the testing loss or the classification accuracy, as follows.

$$\text{result} = \sum_{i=1}^{4} \alpha_i \times \frac{Var[\Phi^{(l_i),\text{reli}}(x) - \Phi^{(l_{i-1}),\text{reli}}(x)]}{Var[f(x)]} + \sum_{i=1}^{4} \beta_i \times \frac{Var[\Phi^{(l_i),\text{unreli}}(x) - \Phi^{(l_{i-1}),\text{unreli}}(x)]}{Var[f(x)]} + b$$

where $l_i \in \{4, 7, 13, 25\}$.

For the CIFAR-10 dataset, we applied cross validation: we randomly selected 20 DNNs from 25 pre-trained ResNet-8/14/20/32/44 models on different training sets in Exp. 2 to learn the regressor and used the other 5 DNNs for testing.[1] These 25 DNNs were learned using 200-5000 samples, which were randomly sampled from the CIFAR-10 dataset to boost the model diversity. We repeated such experiments for 1000 times for cross validation.

Table 1 reports the mean absolute value of prediction error for the classification accuracy and the task loss over 1000 repeated experiments. The prediction error was much less than the value gap of the testing accuracy and the value gap of the task loss, which indicated the strong relationship between the distribution of feature complexity and the performance of DNNs.

	Accuracy		Task loss	
	Prediction error	Range of value	Prediction error	Range of value
CIFAR-10	**2.73%**	28.73%-72.83%	**0.49**	1.59-6.42
CUB200-2011	**5.66%**	28.18%-56.18%	**0.47**	2.94-5.76
Stanford Dogs	**3.26%**	9.37%-37.95%	**0.34**	4.34-7.97

Table 1. The mean error of using the feature complexity to predict the classification accuracy and the task loss. The prediction error was much less than the range of the testing accuracy and the range of the task loss.

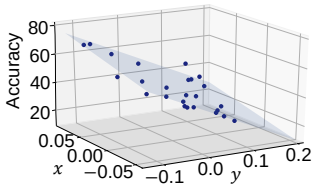

Figure 3. Relationship between the feature complexity and the accuracy on the CIFAR-10 dataset.

Figure 3 further visualizes the plane of the linear regressor learned on the CIFAR-10 dataset. The visualization was conducted by using PCA (Wold et al., 1987) to reduce the 8-dimensional feature into a 2-dimensional space, *i.e.* (x, y) in Figure 3. There was a close relationship between the distribution of feature complexity and the performance of a DNN.

References

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

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2 (1-3):37–52, 1987.

[1]For the CUB200-2011 dataset and the Stanford Dogs dataset, we randomly selected 11 models from 12 pre-trained ResNet-18/34 and VGG-16 models to learn the regressor. One model was used for testing.