EXPLORING LINEAR FEATURE DISENTANGLEMENT FOR NEURAL NETWORKS

Tiantian He¹, Zhibin Li², Yongshun Gong^{1*}, Yazhou Yao³, Xiushan Nie⁴, Yilong Yin^{1*}

¹Shandong University, ²CSIRO, ³Nanjing University of Science and Technology ⁴Shandong Jianzhu University

ABSTRACT

Non-linear activation functions, e.g., Sigmoid, ReLU, and Tanh, have achieved great success in neural networks (NNs). Due to the complex non-linear characteristic of samples, the objective of those activation functions is to project samples from their original feature space to a linear separable feature space. This phenomenon ignites our interest in exploring whether all features need to be transformed by all nonlinear functions in current typical NNs, i.e., whether there exists a part of features arriving at the linear separable feature space in the intermediate layers, that does not require further non-linear variation but an affine transformation instead. To validate the above hypothesis, we explore the problem of linear feature disentanglement for neural networks in this paper. Specifically, we devise a learnable mask module to distinguish between linear and non-linear features. Through our designed experiments we found that some features reach the linearly separable space earlier than the others and can be detached partly from the NNs. The explored method also provides a readily feasible pruning strategy which barely affects the performance of the original model. We conduct our experiments on four datasets and present promising results.

Index Terms— Linear feature disentanglement, Neutral networks, Network pruning

1. INTRODUCTION

Neutral networks (NNs) have achieved great success in a large variety of fields, e.g., multi-media data processing, natural language processing, and speech recognition [1]. Non-linear activation functions, e.g., Sigmoid, ReLU [2], and Tanh, play important roles in NNs. The accumulation of multiple activation functions enables the NN to fit any continuous function, as well as projects samples with complex non-linear characteristics from their original feature space to a linear separable feature space [3].

In this study, we aim to explore that whether all features need to be transformed by non-linear activation functions in current typical NNs, i.e., whether there exists a part of features have reached a linearly separable space earlier than the



Fig. 1. Illustration of two samples.

others during the learning process. Accordingly, such features do not need to pass the non-linear transformation while using an affine transformation instead. As illustrated in (a) of Fig. 1, two samples are presented into three dimensions. It is apparent that these two samples can be distinguished by their features in z axis via a linear interface, but linearly indistinguishable in the x and y axes because they are projected to the same point (see (b) in Fig. 1). In this case, we illustrate that the features in the third dimension (z) are linear separable, and features in the first two dimensions (x and y) are required further non-linear variation. Note that, these linear distinguishable features do not mean that samples can be classified based solely on them, it is a complex learning process.

To verify our hypothesis, we explore the problem of linear feature disentanglement in neural networks. Specifically, we design a learnable mask module to distinguish between linear features and non-linear features. The mask module generates two complementary binary masks for disentangling linear features. To overcome the non-differentiable problem of the binary mask in the back-propagation, we adopt the straight-through estimator [4] to estimate the gradient. From the results of experiments, we discovered some phenomena as we expect: there exist the linear features which appear in the intermediate layers, and the proportions of them can converge at the end of the training process. Moreover, our method barely deteriorates the model performance.

Furthermore, we also explore the potential of the pro-

^{*}Corresponding authors.

posed network structure in network pruning. The intuition behind this design is to leverage the linear property of features to build a fast track, where linear features can be connected to the last layer of neural networks directly. The main contributions of this paper are summarized as follows:

- We provide the first attempt on distinguishing linear features during the learning process of NNs. A module using learnable masks is proposed according to disentangle linear features.
- We include the binarization of the vector generated by the mask module into the training, and use the straightthrough estimator [4] to alleviate the non-differentiable problem of the binarization function.
- We conduct extensive experiments on convolutional neural network (CNN). Experimental results confirm that a part of features reached the linearly separable space earlier than others. Accordingly using our disentanglement strategy will barely affect the model performance.
- We apply the proposed method in network pruning and show some promising results.

2. RELATED WORK

In this section, we briefly review some related works on feature transformation in NNs. In the early studies, the Logistic Sigmoid and Tanh have been widely used in the traditional NNs. However, they suffer from vanishing gradient especially when models go deeper. Hinton et al. [2] propose the Rectified Linear Unit (ReLU), which makes the training of deep NNs quicker and overcomes the problem of vanishing gradient. It is still one of the most popular activation functions and is extensively applied in deep models due to its simplicity, generality and effectiveness. Subsequently, various variants of ReLU have emerged, e.g., LeakyReLU [5], MTLU [6]. These variants make up for the dead neuron issue of ReLU. Besides, some adaptive activation functions have been investigated recently. They contain some trainable parameters tuning the non-linearity according to the data. For example, Ramachandran et al. [7] propose an adaptive function called Swish. It introduces a learnable parameter which controls the degree of interpolation between the linear function and the ReLU function. The recently proposed general activation function, ACON [8] is in fact an extension of Swish, whose switch factor allows every neuron can be activated with different degrees. With the expansion of the non-linear search space, these trainable activation functions also increase the time and space complexity.

In prior applications of activation functions, all features are fed to the non-linear activation functions. To our best knowledge, there is limited work on investigating whether it



Fig. 2. Overall pipeline of our method. In the mask module, we set the threshold τ to 0 in our implementation.

is necessary to pass all of features through non-linear activation functions. This paper provides the first attempt on the linear feature disentanglement problem during the NN learning process.

3. PROPOSED METHOD

In this section, we provide a feasible scheme to identify the non-linear and linear features. We present the linear feature disentanglement for CNN models in the paper.

Overall Structure. The overall structure of our method is illustrated in Fig. 2. The mask module in it generates two complementary binary masks for selecting the linear and nonlinear features from the output features of a layer. The mask modules divide the features into linear and non-linear part. While the linear features no longer undergo any non-linear transformation because they have been mapped into the linearly separable space well, and just need some appropriate affine transformations. Finally, the two types of features are concatenated as the final features as the input of the final decision layer.

Mask Module. Specifically, the mask module as shown in the green box of Figure 2 learns a function as below:

$$\mathbf{z} = Tanh(G(\mathbf{m})) \in \mathbb{R}^c.$$
(1)

Here, $\mathbf{m} \in \mathbb{R}^c$ is a randomly initialized vector, where c

is the channel number of the input features for the mask module in CNN. G is a function defined as $G : \mathbf{m} \rightarrow \mathbf{g}, \mathbf{g} \in \mathbb{R}^c$. It consists of two fully-connected layers, $\mathbf{g} = FC_2(ReLU(FC_1(\mathbf{m})))$, where FC_1 and FC_2 are two linear transformations, ReLU [2] is the non-linear activation function. And there is another non-linear activation function Tanh, which normalizes the output of the function G to [-1, 1] for stabilizing the training process of the whole network.

Notice that z is a real-valued vector, we need to transform it to a binary vector to indicate which feature is linearly separable. The final two masks are expressed as below:

$$\mathbf{mask_1}[i] = \begin{cases} 1, & \mathbf{z}[i] > 0\\ 0, & \mathbf{z}[i] \le 0, i \in [1, c], \end{cases}$$
(2)

$$mask_2 = 1 - mask_1, \tag{3}$$

where $mask_1$ and $mask_2$ correspond to the non-linear features and linear features respectively. The purpose of mask1 and $mask_2$ is to distinguish which channels (or convolution kernels) in CNN models can be recognized as the linear channels. Given a input image x, we define the non-linear feature and linear feature as: $F_{nonlinear}(\mathbf{x}) = \mathbf{mask_1} \odot F^l(\mathbf{x})$ and $F_{linear}(\mathbf{x}) = \mathbf{mask}_2 \odot F^l(\mathbf{x})$, where $F^l(\mathbf{x})$ denotes the output feature of the l-th convolutional layer and \odot denotes that each element of $mask_1$ (or $mask_2$) is multiplied by the corresponding channel of feature $F_{nonlinear}(\mathbf{x})$ (or $F_{linear}(\mathbf{x})$). And the non-zero channels in $F_{linear}(\mathbf{x})$ correspond exactly to the linear kernels in the *l*-th convolutional layer, while the rest correspond to the non-linear kernels. Obviously, Eq.(2) and Eq.(3) are non-differentiable. In this paper, we leverage the method of straight-through estimator [4] and define the gradients of both $mask_1$ w.r.t z and $mask_2$ w.r.t z to 1. So the well-defined $\frac{\partial F_{nonlinear}(\mathbf{x})}{\partial \mathbf{mask_1}}$ and $\frac{\partial F_{linear}(\mathbf{x})}{\partial \mathbf{mask_2}}$ can be used as approximations for $\frac{\partial F_{nonlinear}(\mathbf{x})}{\partial \mathbf{z}}$ and $\frac{\partial F_{linear}(\mathbf{x})}{\partial \mathbf{z}}$, respectively.

We set every element in z > 0 in the network initialization, because we suppose all features need to undergo nonlinear transformations at first and then observe changes in the proportion of non-linear and linear features. The initialization conduces to optimize better the network at the beginning of the training process. And as the training phase progresses, the mask modules will be able to map the corresponding features well to the linearly separable space.

The mask module is a relatively straightforward component, so it is easy to be plugged into various CNN architectures, as well as MLPs and RNNs without special design. And the whole network can be trained in an end-to-end manner via back-propagation thanks to the straight-through estimator.

4. EXPERIMENTS

In this section, the proposed method is empirically investigated on seven common benchmark datasets relevant to



Fig. 3. (a) The schema of the residual module in ResNet-56_M. (b) The convolutional block in VGG-16_M.

classification tasks, CIFAR-10, CIFAR-100 [9], SVHN [10], STL-10 [11]. And we study the performance of our method on two typical CNN models (ResNet-56 [12] and VGG-16 described in [13]) with the above four datasets. Finally, we combine our method with network pruning and conduct relevant exploratory experiments.

4.1. Experiments setting

Datasets. CIFAR-10 and CIFAR-100 consist of 50k training images and 10k testing images in 10 and 100 classes respectively. For these two datasets, we hold out 5k images from training images for validation. The SVHN dataset has 73,257 images for training and 26,032 images for testing from 10 categories. In addition, there are 531,131 additional images in SVHN, but we did not use the images due to restricted computational resources. STL-10 contains 96×96 natural images belonging to 10 classes. And most of the images in the dataset are unlabeled, so we use only the labeled images. There are 5k training and 8k testing labeled images.

Architecture setting. We use ResNet-56 and VGG-16 which are widely used for image classification as the backbone networks. Fig. 3 depicts the structures of ResNet-56_M and VGG-16_M. Indeed, we add a mask module before the non-linear activation function at the end of each residual block on ResNet-56. while on VGG-16, we do the same before each max-pooling layer. According to the actual situation, the number and location of mask modules can be changed flexibly. Note that, ResNet-56 has three stages of residual block, so we add the mask modules to all three stages on ResNet-56_M_A, last two stages on ResNet-56_M_B and last stage on ResNet-56_M_D. And VGG-16_M has five mask modules.

Training setting. To get the baseline accuracies for ResNet-56 and VGG-16, we follow the same training schedule as [12]. For a fair comparison, we train our models with mask modules utilizing the same training scheme as baseline models, except that we optimize the mask modules via Adam with an initial learning rate of 0.001 and a weight decay of 0.0001.



Fig. 4. Proportion curves of some non-linear features for ResNet-56_M_D on CIFAR-10. The four subfigures above show the results of non-linear features extracted by the first two and the last two mask modules.

4.2. Results and Analysis

The purpose of the experiments is finding a part of the features that have been mapped to the linearly separable space early. And then we can verify the rationality of our method by top-1 accuracy and the trend of the proportion of features requiring non-linear activation functions. The proportion of the non-linear features in the *l*-th mask module could be formulated as $P_{nonlinear} = \frac{\sum_{i=1}^{c} \max_{i=1}^{l} \max_{i} l(i)}{c}$, where $\max_{i=1}^{l} l(i)$ denotes the *i*-th element of the mask corresponding to the non-linear features in the *l*-th layer.

We analyse the performance on CIFAR-10/100, SVHN, STL-10, comparing against two popular CNNs, including ResNet-56 and VGG-16. Firstly, as shown in Fig. 4, the proportions of non-linear features have converged eventually in all layers. The phenomenon proves that some intermediate layers in the network already have the ability to map the corresponding features to the linearly separable space. And in terms of the change curves from different layers, we find that if the proportion of non-linear features extracted in the first



Fig. 5. Proportion curves of some non-linear features for VGG-16_M on CIFAR-10. The four subfigures above show the results of non-linear features extracted by the first two and the last two mask modules.

few layers are low, the proportion in the last few layers tends to be higher. After linear features are extracted in the first few layers, in order to maintain the performance of the network, more non-linear features need non-linear transformations. On the other hand, we observe the contrary phenomenon from Fig. 5. When few or even no linear features are extracted, the representation power of layers closer to the output are relatively stronger during training. So the proportion of nonlinear features of last few layers can decrease and finally converge. The difference between ResNet-56_M_D and VGG-16_M might be caused by the depth of models. In particular, ResNet-56_M_D is deeper then VGG-16_M. Moreover, the mask modules of ResNet-56_M_D are equipped in the last stage, and the network parameters in the stage have greater ability to map corresponding features to the linear separable space. So the first mask module of ResNet-56_M_D can extract more than 20% linear features at last. These observations firmly support our proposed assumption.

The experiment results are reported in Table 1. For the CIFAR-10 dataset, it is evident that our models ResNet-



Fig. 6. Removing linear feature maps results in pruning of corresponding filters.

56_M_D and VGG-16_M are performing better than the baseline models. It seems that the number and location of mask modules can affect the final performance. And for the CIFAR-100 dataset, the performance of our models is somewhat inferior to the baseline models, but the gap between them is marginal. For the SVHN and STL-10 datasets, our models based on ResNet-56 (ResNet-56_M_B on SVHN and ResNet-56_M_A on STL-10) outperform the baseline models, while two models with VGG-16_M achieve worse performance than original VGG-16, especially on SVHN. From the results of experiments, our method is not perfect in terms of accuracy. Nonetheless, we provide some insights on the evolution of linear/non-linear features and the disentanglement problem.

4.3. Exploration on Network Pruning

Considering the linear features that have been mapped to the linearly separable space, we could set up a fast track to export the linear features which are disentangled by the first mask module to the last decision layer of the network. As for the linear features and non-linear features extracted from subsequent layers, we use only non-linear features, while ignore the linear features. That is, we leave out the network parameters associated with the linear features, except for the linear features extracted from the first mask module (see Fig. 6). We conduct the experiments on CIFAR-10 and SVHN with our pre-trained networks: ResNet-56_M and VGG-16_M. The masks generated by the mask modules in above two models

 Table 1.
 Classification accuracy on the CIFAR-10/100,

 SVHN and STL-10 test sets. The best results are in bold.

Model	CIFAR-10	CIFAR-100	SVHN	STL-10
ResNet-56(base)	93.07	70.73	96.66	76.70
ResNet-56_M_A	93.14	70.06	96.55	77.01
ResNet-56_M_B	92.22	70.25	96.80	75.71
ResNet-56_M_D	93.28	70.65	96.57	76.79
VGG-16(base)	93.04	72.54	96.53	80.64
VGG-16_M	93.40	71.92	96.41	79.69

are made use of to guide pruning. For retraining, we fine-tune the networks for 40 epochs after pruning, with the learning rate of 0.001 and divided by 10 at epochs 10 and 20.

We compare our pruning algorithm with some existing frameworks, e.g., L1 [13], KSE [14], SFP [15], GAL [16], HRank [17], Variational [18], FPGM [19]. Table 2 shows the experimental results. The top-1 accuracy drop between pruned model and the baseline model and the reduction ratios of parameters are reported.

CIFAR-10. For the CIFAR-10 dataset, we prune the network based on pre-trained ResNet-56_M_D and VGG-16_M. As shown in Table 2, we observe a gap between our method and others in parameters reduction. But it is noteworthy that the accuracies of our method significantly increase with parameters reduction compared to the original ResNet-56 and VGG-16. The results suggest a great potential of the feature disentanglement.

SVHN. For the SVHN dataset, we prune the network based on pre-trained ResNet-56_M_B. Compared with original ResNet-56, our method achieves 18.7% parameters reduction with a 0.21% accuracy increase. Significantly, our accuracy drop is the smallest, and the reduction of parameters is more than the classic channel pruning method, L1 [13]. The results show that our pruning algorithm achieves competitive performance.

Although our pruning method is inferior to the others in parameter reduction, it is simple yet effective, and the accuracy of ours is competitive. In short, our study shows the great potential in feature disentanglement. This is a pioneering and enlightening research, and provides a wholly new perspective for network pruning.

5. CONCLUSION

In this paper, we make the first exploration to disentangle the linear features from the output features of a network layer. We devise a simple yet efficient module: the learnable mask module that distinguishes between linear features and non-linear

Table 2. Comparison of the pruned ResNet-56 and VGG-16 on CIFAR-10 and SVHN. The "Acc. \downarrow " is the accuracy drop between pruned model and the baseline model, the smaller, the better. The best results are in bold and the second best are underlined.

Model	Method	Acc.↓%	Params.↓%
	Variational [18]	0.78	20.49
	KSE [14]	-0.20	54.73
ResNet-56	HRank [17]	0.09	42.4
(CIFAR-10)	GAL-0.6 [16]	0.28	11.8
	L1 [13]	-0.06	9.4
	Ours	<u>-0.11</u>	5.4
	L1 [13]	0.13	64.0
VGG-16	Variational [18]	0.07	73.34
(CIFAR-10)	GAL-0.05 [16]	0.19	77.6
	Ours	-0.11	11.6
ResNet-56 (SVHN)	L1 [13]	-0.01	9.1
	SFP [15]	-0.12	25.6
	FPGM [19]	0.20	38.8
	Ours	-0.21	18.7

features. We also overcome the non-differentiable problem with the help of the straight-through estimator. Extensive experiments on several datasets demonstrate that our method has a great value in feature disentanglement and provides some insights in the evolution of linear or non-linear features. Besides, our exploration experiments in network pruning have also achieved promising results. Our future work will be on improving the feature disentanglement modules for better parameter efficiency.

6. REFERENCES

- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436– 444, 2015.
- [2] Vinod Nair and Geoffrey E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *In*ternational Conference on International Conference on Machine Learning, 2010.
- [3] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [4] Y. Bengio, Nicholas Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *Computer Science*, 2013.
- [5] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al., "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml.* Citeseer, 2013, vol. 30, p. 3.
- [6] Shuhang Gu, Wen Li, Luc Van Gool, and Radu Timofte, "Fast image restoration with multi-bin trainable linear

units," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4190–4199.

- [7] Prajit Ramachandran, Barret Zoph, and Quoc V Le, "Searching for activation functions," arXiv preprint arXiv:1710.05941, 2017.
- [8] Ningning Ma, Xiangyu Zhang, Ming Liu, and Jian Sun, "Activate or not: Learning customized activation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8032–8042.
- [9] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.
- [10] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, and Andrew Y Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop*, 2011.
- [11] Adam Coates, Honglak Lee, Andrew Y Ng, Adam Coates, Honglak Lee, and Andrew Y Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Aistats*, 2011.
- [12] 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, 2016, pp. 770–778.
- [13] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient convnets," in *International Conference of Learning Representation (ICLR)*, 2017.
- [14] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji, "Exploiting kernel sparsity and entropy for interpretable cnn compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2800–2809.
- [15] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*, 2018.
- [16] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- [17] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao, "Hrank: Filter pruning using high-rank feature map," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1529–1538.
- [18] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian, "Variational convolutional neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2780–2789.

[19] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.