A TWO-PARAMETER LEARNABLE LOGMOID ACTIVATION UNIT

Anonymous authors
Paper under double-blind review

ABSTRACT

A novel learnable Logmoid Activation Unit (LAU) is proposed as, $f(x) = x\ln(1 + \alpha\text{sigmoid}(\beta x))$, by parameterizing Logmoid with two hyper-parameters α and β that are optimized by the back-propagation algorithm. The end-to-end learning deep neural networks with learnable Logmoids can increase the predictive performances beyond well-known activation functions for different tasks.

1 INTRODUCTION

Activation functions are generally classified into linear, nonlinear monotonic and nonlinear non-monotonic functions. Although linear functions have been widely used in early results, they are useless in practical applications. These problems are further addressed by using nonlinear monotonic functions such as Sigmoid, Tanh and ReLU families. While small derivatives may cause the gradient to disappear (Klambauer et al., 2017). Other characters such as the educed saturation, sparsity, efficiency, and ease of use will be explored.

Good activation function is still an open question. One is to design new activation functions by combining different units, such as Mish (Misra, 2019) and TanhExp (Liu & Di, 2020). The other is to parameterize some well-known activation functions Biswas et al. (2020); Zhou et al. (2020), which may show better performance beyond parameter-free functions. We proposed a learnable activation unit with new family of activation functions by parameterizing Logmoid with fewer trainable parameters for each network layer. This allows better simulations than others.

2 METHODS

The experiment simulate new activation functions for different tasks based on the Logmoid Activation Units (LAU) as

$$f(x; \alpha, \beta) = x\ln(1 + \alpha\sigma(\beta x)) \tag{1}$$

where α and β are trainable hyper-parameters being optimized end-to-end to find good one at each layer automatically, and σ refers to Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$. Here, β shows how fast the first derivative changes asymptotically. It keeps the gradient property of ReLU for solving the dying problem by $\beta = 1$. The hyper-parameters from $1 \leqslant \alpha \leqslant 5$ and $1 \leqslant \beta \leqslant 5$ are for practical applications. The accuracy decreases faster for $-1 < \beta < 1$ and $\alpha = 1$. This allows special subfamily Logmoid-1 that is a smooth and non-monotonic activation function beyond ReLU. It inherits the self-gated property of Swish where the self-gated refers to the input itself and a function with the input as its argument, i.e., $f(x) = xg(x)$. The output landscapes of a five-layer fully connected network are shown in Fig.1. While ReLU has lot of sharp transitions Logmoid-1 shows a continuous and fluent transition shape.

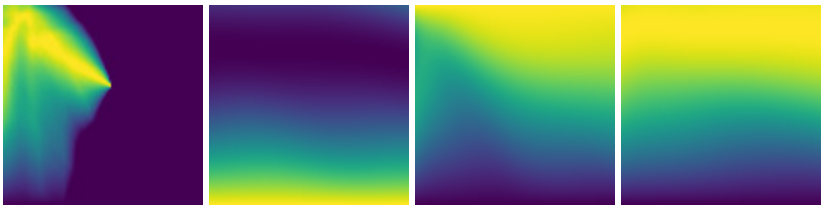

Fig. 1: The output landscapes of ReLU, Swish, TanhExp and Logmoid-1.

LAU networks are feed-forward networks including convolutional and residual architectures with pooling layers. One LAU per layer is inspired by PAUs. All the simulations used a basic network with 15 layers. Each one of the last 12 layers contains a batch normalization, a dropout rate of 0.25, an activation function, and a dense layer with 500 neurons. By training the basic 15 layers network with 15 epoches on MNIST and 30 epoches on KMNIST, Logmoid-1 shows better learning speed than Swish and ReLU both in the convergence speed and final accuracy, and has a similar performance with TanhExp. Meanwhile, Logmoid-1 and TanhExp maintain a high accuracy in large models and then prevent from over-fitting.

3 Experiments and Results

Experiments are performed with Logmoid-1, ReLU, Swish, ACONC (Ma et al., 2021), and TanhExp using different standard networks. Two basic architectures of LeNet and VGG-8 will be used to evaluate Logmoid-1 on FMNIST. LeNet contains 61,706 trainable parameters while LAU adds 8 additional parameters. LAU always matches or outperforms the best performance on FMNIST. Some blocks of VGG-8 will be replaced by dilated convolutions, as Fig.2, named as VGG-8-Dilated network. Some learnable activations are smooth versions of Logmoid-1 by controlling hyper-parameters α and β. This shows LAU is a learnable activation function without requirement of additional suboptimal experiments.

LAU outperforms the baseline activation functions on EfficientNet-B0, ResNet-20, SqueezeNet, MobileNet with CIFAR-100, and on MobileNet-V3 with ImageNet 2012. For the object detection by combing Mask R-CNN (He et al., 2017) with Swin Transformer (Liu et al., 2021), LAU gets better improvements over other activation functions. From Table 1, LAU outperforms the baseline activation functions on EfficientNet-B0, TinyNet, SeNet-18, MobileNet-V2 and ShuffleNet-V2.

Table 1: Performance comparison of activation functions on CIFAR-10.

Model	ReLU	Swish	ACONC	TanhExp	Logmoid-1	LAU
ResNet-20	90.96	91.08	∘91.16	•**91.26**	90.97	∘91.12
MobileNet-V2	84.92	85.87	∘86.54	85.83	86.39	•**87.95**
ShuffleNet-V2	87.42	87.62	∘87.64	87.38	87.18	•**88.97**
SqueezeNet	•**88.50**	87.04	∘88.35	87.86	87.63	∘88.22
SeNet-18	88.81	90.53	90.37	90.19	∘90.95	•**91.43**
EfficientNet-B0	76.17	77.35	∘78.55	77.19	78.23	•**82.52**
TinyNet	76.04	∘79.86	79.04	79.53	78.01	•**80.21**
MicroNet	62.79	67.16	68.48	•**68.79**	67.11	∘68.75

4 Conclusion

The present novel learnable Logmoid Activation Unit is initialized using Logmoid-1, trainable in an end-to-end fashion. One can replace standard activation functions with LAU units in any neural networks. This kind of functions may be potentially applied for approximate any continuous function. This requires more involved theoretical analysis by considering the universal approximation operators, feed-forward neural networks and approximation error. Simulations shows LAU has the best performance across all activation functions and architectures. Further work will be performed to apply LAU to other related tasks.

URM STATEENT

The authors acknowledge that at least one key author of this work meets the URM criteria of the ICLR 2023 Tiny Papers Track.

REFERENCES

Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Tanhsoft–a family of activation functions combining tanh and softplus. *arXiv:2009.03863*, 2020.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Xin-Yu Liu and Xiao-Guang Di. Tanhexp: A smooth activation function with high convergence speed for lightweight neural networks. *arXiv:2003.09855*, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

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*, pp. 8032–8042, 2021.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv:1908.08681*, 2019.

Yuan Zhou, Dandan Li, Shuwei Huo, and Sun-Yuan Kung. Soft-root-sign activation function. *arXiv preprint arXiv:2003.00547*, 2020.

A APPENDIX

Some blocks of VGG-8 will be replaced by dilated convolutions, as Fig.2, named as VGG-8-Dilated network.

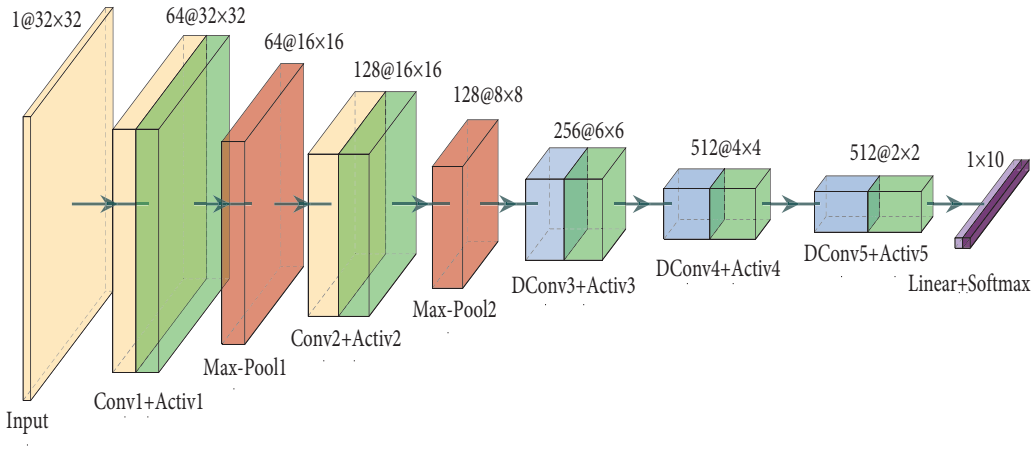

Fig. 2: Schematic VGG-8-Dilated network. DConv denotes the dilated convolutional.