Appendix

LayerAct: Advanced Activation Mechanism for Robust Inference of CNNs

A Important Properties of Activation

One-side saturation and zero-like activation mean stand out as crucial properties of activation for effective and efficient network training. Activation functions that exhibit one-side saturation, such as the ReLU, are favored within deep networks for their potential to provide more informative signal propagation during the backward pass by allowing for larger derivatives [3]. Furthermore, the presence of a saturation state contributes to robustness against input shifts, maintaining consistent activation outputs for elements in this state and thereby stabilizing network training [1, 19].

Achieving a zero-like activation mean is another important property of activation functions, facilitating efficient network training. The activation mean refers to the average activation output of an individual element across samples. Functions such as the Exponential Linear Unit (ELU) [1] and Parametric ReLU (PReLU) [5] push the activation mean toward zero by allowing negative outputs. Activation functions with this property can correct the bias shift that occurs between layers and lead to efficient network training [1, 19].

The activation output of the i^{th} unit of m^{th} sample ($m \in \{1, 2, ..., M\}$) is defined as $a_{m,i} = f(y_{m,i})$, where f, $y_{m,i}$, and M are activation function, the i^{th} activation input of the m^{th} sample, and the number of samples in the batch, respectively. Ideally, a "zero-like activation mean" occurs when the activation mean of a single unit, a_i, approximates zero across the samples. Mathematically, this can be represented as:

$$\frac{1}{M}\sum_{m=1}^{M} a_{m,i} \approx 0.$$

However, approximating the activation mean to zero is challenging for activation functions that saturate at (large) negative outputs, such as ELU, SiLU, or FReLU. Due to this saturation, previous studies have defined the "zero-like activation mean" property of an activation function as its ability to "push" the activation mean towards zero. In mathematical terms, this can be presented as $|\mu_{a_i}| \ll c$, where c is a small positive constant [1, 19].

B Detailed Derivation Process of Activation Fluctuation with LN

In this section, we present the detailed derivation process of the equations in Definition 4 in the main manuscript as follows:

$$\begin{aligned}
\|f(\hat{n}) - f(n)\| &= \sum_{i=1}^{D} |\hat{n}_i s(\hat{n}_i) - n_i s(n_i)| \\
&= \sum_{i=1}^{D} \left| \frac{\hat{y}_i + \epsilon_i - \mu_y - \mu_\epsilon}{\sqrt{\sigma_y^2 + \sigma_\epsilon^2 + \alpha}} s(\hat{n}_i) - \frac{y_i - \mu_y}{\sqrt{\sigma_y^2 + \alpha}} s(n_i) \right| \\
&\approx \sum_{i=1}^{D} \left| n_i (s(\hat{n}_i) - s(n_i)) + \frac{(\epsilon - \mu_\epsilon)\, s(\hat{n}_i)}{\sqrt{\sigma_y^2 + \alpha}} \right|
\end{aligned}$$

$$\leq \sum_{i=1}^{D} \left(|n_i| \cdot |s(\hat{n}_i) - s(n_i)| + \frac{|\epsilon - \mu_\epsilon| \cdot s(\hat{n}_i)}{\sqrt{\sigma_y^2 + \alpha}} \right),$$

where $\sqrt{\sigma_y^2 + \alpha + \sigma_\epsilon^2} \approx \sqrt{\sigma_y^2 + \alpha} > 1$ when $\sigma_y \gg \sigma_\epsilon$ and α is sufficiently large.

C Difference between LayerAct and Activation with LN

In this section, we provide detailed comparison between the activation outputs of LayerAct and those of element-level activation functions paired with LN. When a LN layer is placed right before an activation layer, the i^{th} activation output is $a_i = n_i^{LN} s\left(n_i^{LN}\right)$, where n_i^{LN} is the i^{th} normalized output of LN. Conversely, the i^{th} activation output of a LayerAct function is $a_i = y_i s(n_i)$, as defined in Equation 12 in the main manuscript.

The critical distinction between activation with LN and LayerAct lies in the preservation of input mean and variance statistical information in the activation output. *LayerAct can preserve the statistical information, but activation with LN cannot.* With LN, the activation function takes a normalized input in layer-direction, resulting in activation outputs that exhibit similar mean and variance across samples (as shown in the activation output equation for LN above). This homogenization of statistical information across samples, a characteristic of LN, is a reason why BN often outperforms LN in non-sequential models such as CNNs [11, 14]. In the main manuscript, we demonstrated that this homogeneity limitation of LN leads the representation of the samples with different labels to be similar (see Figure 1 in the main manuscript).

LayerAct, on the other hand, produces more distinguishable activation outputs between samples by preserving statistical variation between samples. This is due to the fact that only the activation scale function of LayerAct uses the layer-normalized input, not the LayerAct function itself (as shown in Equation 12 in the main manuscript). Based on this unique output of LayerAct, networks with LayerAct can produce much more distinguishable representation of samples compared to those with activation and LN (see Figure 3 in the main manuscript).

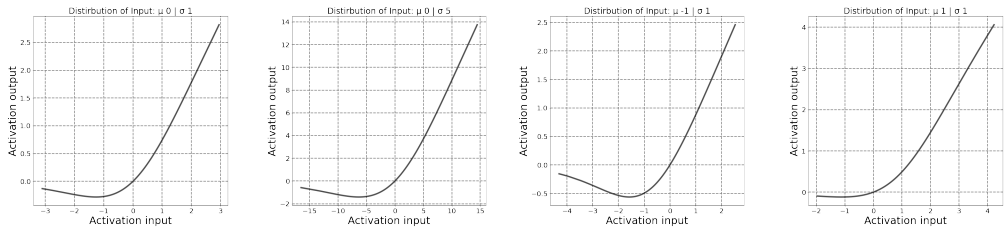

Figure 1: LA-SiLU with different mean and variance values in the input. The distribution of the activation input is: (1) $\mu_y = 0$, $\sigma_y = 1$, (2) $\mu_y = 0$, $\sigma_y = 5$, (3) $\mu_y = -5$, $\sigma_y = 1$, and (4) $\mu_y = 5$, $\sigma_y = 1$ from the left to right.

D Activation Output of LayerAct Functions

In this section, we provide detailed comparison between the activation outputs of LayerAct and those of element-level activation functions paired with LN. When a LN layer is placed right before an activation layer, the i^{th} activation output is $a_i = n_i^{LN} s\left(n_i^{LN}\right)$, where n_i^{LN} is the i^{th} normalized output of LN. Conversely, the i^{th} activation output of a LayerAct function is $a_i = y_i s(n_i)$, as defined in Equation 12 in the main manuscript.

The critical distinction between activation with LN and LayerAct lies in the preservation of input mean and variance statistical information in the activation output. *LayerAct can preserve the statistical information, but activation with LN cannot.* With LN, the activation function takes

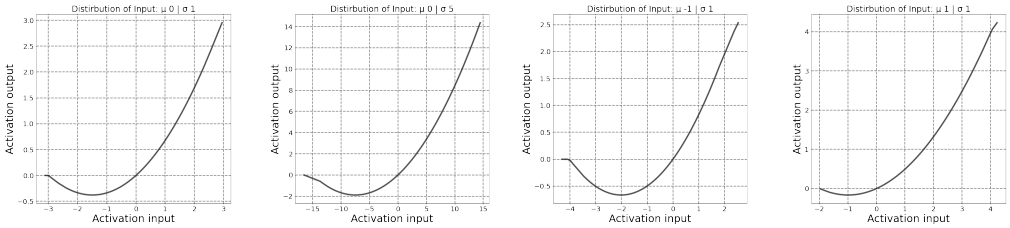

Figure 2: LA-HardSiLU with different mean and variance values in the input. The distribution of the activation input is: (1) $\mu_y = 0$, $\sigma_y = 1$, (2) $\mu_y = 0$, $\sigma_y = 5$, (3) $\mu_y = -5$, $\sigma_y = 1$, and (4) $\mu_y = 5$, $\sigma_y = 1$ from the left to right.

a normalized input in layer-direction, resulting in activation outputs that exhibit similar mean and variance across samples (as shown in the activation output equation for LN above). This homogenization of statistical information across samples, a characteristic of LN, is a reason why BN often outperforms LN in non-sequential models such as CNNs [11, 14].

LayerAct, on the other hand, produces more distinguishable activation outputs between samples by preserving statistical variation between samples. This is due to the fact that only the activation scale function of LayerAct uses the layer-normalized input, not the LayerAct function itself (as shown in Equation 12 in the main manuscript).

E Trade-off between Two Properties

A fundamental trade-off exists between the important properties of activation, one-side saturation and zero-like mean activation. Activation functions that have a saturation state typically limit negative outputs, consequently driving the activation mean away from zero. On the other hand, to achieve a zero-like activation mean, certain functions like PReLU allow large negative outputs. However, functions with this allowance are not robust due to the absence of saturation.

To address this trade-off issue, various activation functions, such as Sigmoid Linear Unit (SiLU, also known as SWISH) [2, 20], Gaussian Error Linear Unit (GELU) [7], and Mish [16], saturate the large negative outputs only. These activation functions can achieve a zero-like activation mean with small negative outputs. However, a trade-off still exists because the restriction of negative outputs, designed to ensure saturation, prevents the allowance of large negative outputs, thereby restraining the mean of activation from being more zero-like.

Additionally, this trade-off cannot be addressed by LN. Even when a LN layer is placed before the activation layer, the large negative output is restricted with element-level activation. Therefore, mitigating the trade-off between activation properties is a distinct benefit of LayerAct functions that element-level activation functions cannot attain.

F Experimental Setting

We implemented LayerAct functions and networks for experiment with PyTorch [18]. All networks used in our experiments were trained on NVIDIA A100. We used multiple devices to train the networks on ImageNet, and a single device for the other experiments. The versions of Python and the packages were (1) Python 3.9.12, (2) numpy 1.19.5 (3) PyTorch 1.11.0, and (4) torchvision 0.12.0. We used cross entropy loss functions for all the experiments. The random seeds of the experiments were $11 \times i$ where $i \in \{1, 2, ..., 30\}$ on CIFAR10 and CIFAR100 and 11 on ImageNet.

To train networks on MNIST for experimental analysis, we applied batch gradient descent for 80 epochs with the weight decay and momentum fixed to 0.0001 and 0.9, respectively. The learning rate started from 0.01, and was multiplied 0.1 at epochs 40 and 60 as scheduled. We used ResNet [6] with BN right before activation for experiments on CIFAR10, CIFAR100 and ImageNet. We

initialized the weights following the methods proposed by He et al. [5]. For all experiments, the weight decay, momentum, and initial learning rate were 0.0001, 0.9 and 0.1, respectively.

For CIFAR10 and CIFAR100, we trained ResNet20, ResNet32, and ResNet44 with a basic block using the stochastic gradient descent with a batch size of 128 for about 64000 iterations. We randomly selected 10% of the training dataset as the validation set. The learning rate was scheduled to decrease by the factor of 10 at 32000 and 48000 iterations. For the data augmentation of CIFAR10 and CIFAR100, we followed [12]. We rescaled the data between 0 and 1, padded 4 pixels on each side, and randomly sampled a 32×32 crop from the padded image or its horizontal flip. The data was normalized after augmentation. For testing, we did not apply data augmentation, only normalized the data. The hyper-parameter α of LayerAct functions for the experiments was set to 0.00001.

For the experiment with ImageNet, we trained ResNet50 and ResNet101 with the bottleneck block using stochastic gradient descent, and the batch size was 256 for about 600000 iterations. The learning rate was scheduled to decrease by a factor of 10 at 180000, 360000, and 540000 iterations. For the data augmentation on ImageNet, we rescaled the data between 0 and 1, resized it to 224×244, and randomly sampled a 224×224 crop from an image or its horizontal flip [10]. We normalized the data after data augmentation. For testing, we resized the data to be 256×256 and applied 10-crop. Afterward, the data was normalized. To ensure stable learning, we set the hyper-parameter α of LayerAct functions to 0.1 which is larger than those for CIFAR10 and CIFAR100.

For the reproducibility, we have attached the code that used for the experiments in supplementary materials.

G Relationship between LayerAct and BN

In our main manuscript, we emphasized the importance of selecting an appropriate normalization method for employing LayerAct functions effectively. This section delves into the relationship between LayerAct and BN, the most dominant normalization method for CNNs. In this section, our objective is to discuss the benefits of BN in CNNs and demonstrate how LayerAct functions collaborate well with BN, in contrast to LN.

BN operates along three dimensions, batch, height, and width, particularly for image datasets, where the input matrix $y \in \mathbb{R}^{B \times C \times H \times W}$. To simplify, we consider a flattened dimension that combines height and width, denoted by size D. The mean, variance, and output of BN are then defined as follows:

$$\mu_j^{BN} = \frac{1}{BD} \sum_{i=1}^{B} \sum_{k=1}^{D} y_{i,j,k}, \tag{1}$$

$$\sigma_j^{BN} = \frac{1}{BD} \sum_{i=1}^{B} \sum_{k=1}^{D} \left(y_{i,j,k} - \mu_j^{BN} \right), \tag{2}$$

$$n_{i,j,k}^{BN} = \gamma_j \cdot \frac{y_{i,j,k} - \mu_j^{BN}}{\sqrt{\sigma_j^{BN\,2} + \alpha}} + \beta_j. \tag{3}$$

BN enjoys the common advantages of normalization methods, re-scaling and re-centering operations that significantly enhance the efficiency and effectiveness of network training, promote stable training, and allow the use of a larger learning rate [14, 11, 9].

Additionally, BN offers the unique benefit of avoiding channel collapse, a phenomenon where a channel exhibits a linear activation (i.e. a channel loses its non-linear activation), leading to inefficient network training [11]. For instance, when a channel's activation inputs are consistently positive or negative across all samples, the activation output of a ReLU layer becomes linear, $a = y$ or $a = 0$, respectively. BN addresses this by normalizing across the batch dimension, ensuring a diverse distribution of channel activations among samples. Moreover, it does not have the homogenization issue present in LN.

Table 1: Classification performance on CIFAR and CIFAR-C datasets without a normalization method. C10 and C100 denotes CIFAR10 and CIFAR100, respectively.

			CIFAR		**CIFAR-C**				
Data	Model	Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
C10	ResNet20	ReLU	88.51	67.43	53.27	62.12	70.93	75.91	71.38
C10	ResNet20	SiLU	88.29	68.94	54.66	**65.84**	71.6	76.54	72.48
C10	ResNet20	LA-SiLU	**88.95**	**69.36**	**54.71**	**65.58**	**71.93**	**77.59**	**73.34**
C10	ResNet32	ReLU	**89.56**	68.94	54.48	63.90	72.45	77.52	72.73
C10	ResNet32	SiLU	46.51	38.69	33.21	37.52	39.62	41.6	40.11
C10	ResNet32	LA-SiLU	89.12	**70.59**	**56.29**	**66.81**	**73.11**	**78.94**	**74.22**
C10	ResNet44	ReLU	**90.03**	69.97	55.75	65.08	73.38	78.45	73.62
C10	ResNet44	SiLU	15.25	14.18	13.5	14.01	14.26	14.55	14.39
C10	ResNet44	LA-SiLU	88.77	**71.65**	**58.15**	**68.32**	**74.01**	**79.49**	**74.88**
C100	ResNet20	ReLU	59.46	37.69	21.17	38.76	41.36	44.15	38.89
C100	ResNet20	SiLU	59.74	40.39	24.21	42.72	43.51	46.45	41.01
C100	ResNet20	LA-SiLU	**61.01**	**42.44**	**26.07**	**44.50**	**45.74**	**48.8**	**42.98**
C100	ResNet32	ReLU	**60.71**	39.56	23.05	40.78	43.23	45.93	40.67
C100	ResNet32	SiLU	20.36	14.99	10.05	16.27	15.89	16.47	15.02
C100	ResNet32	LA-SiLU	60.08	**44.01**	**28.77**	**46.73**	**47.02**	**49.51**	**44.19**
C100	ResNet44	ReLU	**61.59**	40.88	24.57	42.08	44.60	47.16	41.90
C100	ResNet44	SiLU	2.91	2.43	1.97	2.58	2.53	2.54	2.42
C100	ResNet44	LA-SiLU	60.30	**44.26**	**29.22**	**46.74**	**47.45**	**49.65**	**44.50**

When utilized with LayerAct, BN does not compromise the benefits of LayerAct because they operate on different normalization dimensions. Given that the output of LayerAct functions is the product of the activation input (i.e. the normalization output) and the activation scale, BN's benefits extend to the activation output, enhancing overall network performance as when utilized with element-level activation function. Therefore, utilizing LayerAct with BN preserves the advantages of both, enhancing the efficiency of network training and robustness of network inference.

In summary, the choice of normalization method for CNNs with LayerAct functions should be made carefully, taking into account the interplay between LayerAct and the normalization. While LN may diminish LayerAct's benefits, as discussed in Appendix H, BN, with its beneficial properties, would be a suitable choice for CNNs with LayerAct functions on image datasets.

H LayerAct with no normalization or LN

In this section, we explore the cases when LA-SiLU is utilized with LN. For the further investigation, we first examine the relationship between LN and LayerAct, subsequently presenting experimental results on CIFAR datasets for two scenarios: (1) networks without any normalization methods, and (2) networks with LN. ReLU and SiLU were selected as baseline element-level activation functions, with ReLU being the most popular choice for CNNs, and SiLU being as the corresponding element-level activation function of LA-SiLU.

For the networks incoporating LN, the normalization output of a LN layer is expressed as $n^{\hat{LN}} = \gamma \cdot \frac{y-\mu_y^2}{\sqrt{\sigma_y^2+\alpha}} + \beta$. This equation results in the output of SiLU $n^{\hat{LN}} s\left(n^{\hat{LN}}\right)$ and LA-SiLU $n^{\hat{LN}} s\left(n\right)$ becoming similar. When the learnable parameter γ and β are 1 and 0, respectively, the outputs $n^{\hat{LN}} s\left(n^{\hat{LN}}\right)$ and $n^{\hat{LN}} s\left(n\right)$ become identical, nullifying any distinct advantage offered by the LayerAct mechanism. Thus, the more the output of the LN layer approximates the normalized

Table 2: Classification performance on CIFAR and CIFAR-C datasets with LN. C10 and C100 denote CIFAR10 and CIFAR100, respectively.

			CIFAR		**CIFAR-C**				
Data	Model	Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
C10	ResNet20	ReLU	88.24	73.77	62.05	71.26	76.48	79.69	76.46
C10	ResNet20	SiLU	87.53	74.3	63.32	71.74	77.33	79.77	76.58
C10	ResNet20	LA-SiLU	**88.52**	**75.31**	**63.81**	**73.11**	**78.23**	**80.92**	**77.6**
C10	ResNet32	ReLU	**88.55**	74.48	63.33	71.92	76.94	80.33	77.11
C10	ResNet32	SiLU	87.27	75.3	64.78	72.92	78.55	80.3	77.32
C10	ResNet32	LA-SiLU	87.85	**76.39**	**67.35**	**74.22**	**78.80**	**80.99**	**78.32**
C10	ResNet44	ReLU	**88.58**	75.2	64.32	72.75	77.87	**80.67**	77.67
C10	ResNet44	SiLU	86.65	75.11	65.67	72.73	77.84	79.85	77.1
C10	ResNet44	LA-SiLU	86.88	**76.73**	**69.53**	**74.98**	**78.51**	80.43	**78.43**
C100	ResNet20	ReLU	61.30	44.04	31.82	44.39	47.03	48.86	45.07
C100	ResNet20	SiLU	60.45	**45.93**	**35.66**	**46.46**	48.46	49.91	**46.59**
C100	ResNet20	LA-SiLU	**62.30**	45.30	31.63	45.90	**48.72**	**50.65**	46.17
C100	ResNet32	ReLU	**63.21**	45.92	33.23	46.17	49.26	50.94	46.84
C100	ResNet32	SiLU	60.04	47.14	**37.37**	**48.60**	49.57	50.33	47.41
C100	ResNet32	LA-SiLU	60.76	**47.26**	36.63	48.51	**49.97**	**51.03**	**47.51**
C100	ResNet44	ReLU	**64.18**	46.63	33.92	46.89	**49.68**	**51.72**	**47.75**
C100	ResNet44	SiLU	59.58	**47.41**	**38.28**	48.69	49.45	50.61	47.73
C100	ResNet44	LA-SiLU	60.17	47.31	38.10	**48.82**	49.63	50.26	47.46

input of LayerAct's activation scale function (i.e., the more gain and bias approximate to 1 and 0, respectively), the more the benefits of LayerAct are reduced.

Nevertheless, the benefits of LayerAct functions, addressing the trade-off between two important activation properties and potentially exhibiting lower variance in robustness across samples, can still be partially retained when combined with LN that the learnable parameters γ and β are far from 1 and 0, respectively. This is because the layer-direction normalized input of LayerAct's activation scale function does not utilize an affine function. By not utilizing an affine function, we can ensure that $\|s(\hat{n})\|$ of LayerAct functions remains small enough for robustness. When an affine function is used, the mean of the normalized vector n can become large if the parameter β is large. Such large n leads to a large $\|s(\hat{n})\|$, which does not ensure robustness.

Additionally, LayerAct functions can allow a larger negative output compared to element-level activation functions, as they do not restrict large negative outputs. Furthermore, the robustness benefit of LayerAct becomes significant when the normalized outputs of LN (i.e. activation input) are mostly outside the saturation state.

To investigate this, we conducted experiments on ResNets both without normalization and with LN. We followed the same experimental setting of the experiments in our main manuscript. The results reported in the tables are the average accuracies over 30 runs, each with different seeds for weight initialization.

Table 1 presents the performance of ResNets without any normalization methods on both clean and noisy CIFAR datasets, with a learning rate set at 0.01 for stable training. The results demonstrate that networks with LA-SiLU outperformed those with ReLU and SiLU on noisy datasets. This indicates that the robustness of LayerAct functions is independent from the presence of normalization layers. Additionally, while networks with SiLU, the corresponding element-level activation function, experienced training instability and often exploded, those with LA-SiLU maintained stable training. This suggests that the LayerAct mechanism can contribute to more stable network training. However, in deeper networks such as ResNet32 and ResNet44, the performance

Table 3: Classification performance on CIFAR and CIFAR-C datasets with SN. We report the average mean accuracy in the table. C10 and C100 denotes CIFAR10 and CIFAR100, respectively.

			CIFAR		**CIFAR-C**				
Data	Model	Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
C10	ResNet20	ReLU	89.65	72.40	56.07	71.35	75.24	80.43	74.81
C10	ResNet20	SiLU	**90.60**	**74.17**	**57.70**	**72.94**	**77.57**	**82.21**	**76.33**
C10	ResNet20	LA-SiLU	89.56	72.43	55.9	71.05	75.47	80.87	74.73
C10	ResNet32	ReLU	90.70	73.90	58.24	72.84	76.3	81.79	76.42
C10	ResNet32	SiLU	**90.79**	**74.65**	**58.78**	**73.39**	**77.64**	**82.57**	**76.91**
C10	ResNet32	LA-SiLU	89.94	72.72	56.15	71.07	75.52	81.48	75.23
C10	ResNet44	ReLU	**91.40**	**74.65**	**58.10**	**74.01**	**77.18**	**82.81**	**76.99**
C10	ResNet44	SiLU	74.48	61.70	49.50	60.58	64.11	67.87	63.41
C10	ResNet44	LA-SiLU	89.36	72.02	54.85	70.59	75.07	81.05	74.24
C100	ResNet20	ReLU	57.36	37.01	20.61	38.43	39.75	43.56	38.59
C100	ResNet20	SiLU	**64.55**	**42.96**	**25.13**	**43.91**	**46.35**	**50.42**	**44.55**
C100	ResNet20	LA-SiLU	63.88	41.76	23.23	42.83	45.28	49.43	43.4
C100	ResNet32	ReLU	60.19	38.81	21.89	40.02	41.30	46.00	40.61
C100	ResNet32	SiLU	**64.35**	**42.45**	**25.00**	43.08	**45.86**	49.92	**44.00**
C100	ResNet32	LA-SiLU	64.05	42.27	23.72	**43.46**	45.50	**50.16**	43.89
C100	ResNet44	ReLU	61.64	39.93	22.64	41.03	42.47	47.54	41.67
C100	ResNet44	SiLU	44.8	29.57	18.11	29.96	31.98	34.21	30.73
C100	ResNet44	LA-SiLU	**62.99**	**41.91**	**22.86**	**43.06**	**46.02**	**49.79**	**43.06**

with ReLU surpassed those with LA-SiLU. This outcome may be because of the more complex representation of LA-SiLU's output, pointing towards the necessity of a normalization layer to mitigate potential overfitting issues.

Table 2 demonstrates the performance of ResNets with LN on both clean and noisy CIFAR10 and CIFAR100 datasets. It validated our concerns about the relationship between LN and LayerAct. On noisy CIFAR100, ResNet20 and ResNet44 with SiLU demonstrated similar or greater robustness than those with LA-SiLU. This suggests that the robustness benefits of LayerAct are diminished when utilized with LN. However, it is important to note that the benefits of LayerAct may still be partially preserved even when LN is applied. For clean and noisy CIFAR10, the results where similar with those in Table 1, the networks with LA-SiLU demonstrated more robust inference compared to those with other activation functions on noisy datasets, while the performance of the deeper networks (ResNet32 and ResNet44) with ReLU where better on clean datasets than those with LA-SiLU. Furthermore, on clean datasets, the networks with LA-SiLU outperformed those with SiLU, which employs the same function (Logistic Sigmoid) for the activation scale with LA-SiLU. This indicates that the advantage of LA-SiLU, in addressing the trade-off, can enhance network performance.

Additionally, networks employing LayerAct functions with BN, which normalizes in a different dimension from that of LayerAct functions and remains the dominant normalization method for CNNs, perform effectively. In our experiments reported in the main manuscript, BN was employed for all networks, except for UNets, which do not utilize any normalization layer. The networks with LayerAct functions exhibited remarkable performance on both clean and noisy datasets. Consequently, despite the necessity for careful selection of the normalization method when using LayerAct functions, their robustness and improved performance when combined with BN make them an attractive choice for CNNs.

I LayerAct with Other Normalization Methods.

To investigate the relationship between LayerAct and various normalization methods, we conducted experiments on ResNets with Switchable Normalization (SN) [15], Instance enhancement batch normalization (IEBN) [13], and Decorrelated Batch Normalization (DBN) [8]. We used the same experiment setting with those of our main manuscript. We report the average accuracy over 10 runs for the experiments.

Analysis of the experimental results revealed that LayerAct functions are not effectively compatible with normalizations that can cause a large variance in the channel means, such as SN and IEBN. This incompatibility arises because LayerAct is more sensitive to such large variance between channel means, which causes channels with smaller means to be more likely to become inactivated compared to those with larger means.

Table 3 demonstrates that utilizing LayerAct with SN is not effective. This ineffectiveness arises because SN is a combination of BN, LN, and Instance normalization (IN) [21], while the benefit of LayerAct functions diminish when preceded by LN. Additionally, LayerAct functions are more sensitive to the presence of similar channel characteristics across samples compared to element-level activation functions because the channel dimension is incorporated into normalizing dimension of LayerAct's activation scale input. IN aggressively normalizes the mean and variance across channels towards uniformity. Therefore, given the composite normalization strategy of SN, we expect that the effect of IN will lead to a homogenization of channel characteristics, thus undermining the efficiency of LayerAct functions.

Table 4: Classification performance on CIFAR and CIFAR-C datasets with IEBN. We report the average mean accuracy in the table. C10 and C100 denotes CIFAR10 and CIFAR100, respectively.

			CIFAR	**CIFAR-C**					
Data	Model	Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
C10	ResNet20	ReLU	91.50	70.12	52.49	66.73	73.44	79.64	73.89
C10	ResNet20	SiLU	91.44	68.98	49.39	66.26	72.29	79.00	73.06
C10	ResNet20	LA-SiLU	**91.63**	**70.64**	**52.70**	**67.30**	**74.20**	**79.87**	**74.65**
C10	ResNet32	ReLU	**92.54**	71.67	54.06	**68.82**	74.82	81.12	75.11
C10	ResNet32	SiLU	91.85	71.07	53.39	68.26	73.90	80.50	74.90
C10	ResNet32	LA-SiLU	92.36	**71.77**	**54.19**	68.19	**75.32**	**81.15**	**75.60**
C10	ResNet44	ReLU	**92.78**	72.18	55.20	68.83	75.61	81.59	75.43
C10	ResNet44	SiLU	92.08	71.23	52.79	68.54	74.27	80.95	74.99
C10	ResNet44	LA-SiLU	92.50	**73.01**	**56.97**	**69.18**	**76.03**	**82.34**	**76.5**
C100	ResNet20	ReLU	66.62	**41.97**	**22.81**	**42.51**	45.07	50.30	44.37
C100	ResNet20	SiLU	66.19	40.88	21.14	41.08	44.29	49.68	43.28
C100	ResNet20	LA-SiLU	**66.73**	41.59	21.31	41.98	**45.26**	**50.53**	**43.82**
C100	ResNet32	ReLU	**68.17**	43.49	23.81	**43.89**	46.76	52.30	**45.78**
C100	ResNet32	SiLU	67.43	42.30	23.15	42.38	45.33	51.14	44.68
C100	ResNet32	LA-SiLU	67.97	**43.56**	**24.49**	43.39	**46.94**	**52.47**	45.76
C100	ResNet44	ReLU	**69.47**	44.73	25.74	44.77	47.77	53.51	47.12
C100	ResNet44	SiLU	68.18	43.47	24.76	43.37	46.46	52.22	45.89
C100	ResNet44	LA-SiLU	68.41	**45.29**	**26.89**	**45.15**	**48.67**	**54.03**	**47.13**

Table 4 demonstrates the performance of networks with IEBN. With the exception of ResNet20 on CIFAR100, networks with LA-SiLU demonstrated enhanced performance on noisy datasets when compared to their counterparts utilizing ReLU and SiLU. Conversely, ReLU outperformed LA-SiLU in ResNet32 and ResNet44 models. Nonetheless, it is important to highlight that LA-SiLU consistently surpassed SiLU, which utilizes the same activation scale function, across all

tested scenarios. These results imply that the mechanism of LayerAct holds promise for enhancing efficiency. It is also important to consider careful consideration and integration of network complexity, particularly due to the interplay between normalization and activation functions, are essential, given that the scale function of ReLU is considerably simpler compared to sigmoid, the scale function of LA-SiLU and SiLU.

Table 5: Classification performance on CIFAR and CIFAR-C with DBN. We report the average mean accuracy in the table. C10 and C100 denotes CIFAR10 and CIFAR100, respectively.

			CIFAR	**CIFAR-C**					
Data	Model	Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
C10	ResNet20	ReLU	90.8	65.52	35.87	65.43	71.27	78.68	68.93
C10	ResNet20	SiLU	87.69	63.55	38.16	61.21	70.24	74.77	67.01
C10	ResNet20	LA-SiLU	**91.72**	**69.36**	**45.68**	**67.81**	**73.41**	**80.6**	**73.37**
C10	ResNet32	ReLU	91.85	68.1	37.15	68.38	**75.07**	81.04	71.1
C10	ResNet32	SiLU	92.22	68.3	40.01	67.24	73.98	81.44	71.77
C10	ResNet32	LA-SiLU	**92.27**	**70.36**	**45.62**	**69.32**	74.58	**81.96**	**74.14**
C10	ResNet44	ReLU	92.35	69.79	39.25	**70.48**	76.61	**82.31**	72.67
C10	ResNet44	SiLU	**92.39**	69.48	42.19	68.97	74.63	81.63	73.13
C10	ResNet44	LA-SiLU	92.36	**70.17**	**44.0**	69.35	**74.63**	82.22	**74.11**
C100	ResNet20	ReLU	**58.38**	**34.19**	**13.48**	**34.26**	**38.28**	**43.67**	**36.07**
C100	ResNet20	SiLU	57.82	32.13	13.41	31.57	35.78	40.99	34.22
C100	ResNet20	LA-SiLU	58.19	32.24	14.32	30.95	35.11	41.4	34.96
C100	ResNet32	ReLU	54.06	30.29	14.16	28.67	34.09	38.34	32.14
C100	ResNet32	SiLU	62.0	35.26	14.99	34.41	39.35	45.01	37.46
C100	ResNet32	LA-SiLU	**65.02**	**38.75**	**19.18**	**37.84**	**41.81**	**48.66**	**41.36**
C100	ResNet44	ReLU	60.32	34.64	14.96	33.68	39.03	44.1	36.51
C100	ResNet44	SiLU	64.67	37.58	15.81	37.63	42.01	47.2	39.83
C100	ResNet44	LA-SiLU	**67.59**	**42.44**	**21.03**	**42.54**	**46.52**	**52.15**	**44.6**

Table 5 demonstrates the performance of networks with DBN. Except for ResNet20 on CIFAR100, networks with LA-SiLU showed similar or improved performance on clean datasets, and outstanding performance on noisy datasets compared to those employing ReLU and SiLU. The results of these experiments highlight the potential applicability of LayerAct functions in conjunction with advanced batch-direction normalization methods.

Table 6: Classification performance on the clean CIFAR10 and CIFAR100

	CIFAR10			**CIFAR100**		
Activation	ResNet20	ResNet32	ResNet44	ResNet20	ResNet32	ResNet44
ReLU	91.29	92.03	92.03	65.92	67.04	68.02
LReLU	91.31	92.03	92.03	65.88	**67.37**	67.96
PReLU	90.82	92.03	-	64.00	66.35	67.68
SiLU	91.45	92.17	92.18	65.89	67.22	67.71
HardSiLU	91.09	91.77	91.42	65.19	66.49	66.38
Mish	91.48	**92.21**	92.30	65.85	67.18	68.06
GELU	**91.50**	**92.25**	**92.22**	65.84	67.30	**68.19**
ELU	91.04	91.61	91.68	**66.24**	67.01	67.55
LA-SiLU	**91.60**	92.20	**92.36**	**66.39**	**67.74**	**68.07**
LA-HardSiLU	91.21	91.68	91.36	66.16	66.63	65.51

J Additional Tables

Table 6 shows the classification performance of networks on clean CIFAR10 and CIFAR100 datasets. LA-SiLU outperformed element-level activation functions in four out of six cases. Specifically, LA-SiLU always demonstrated superior performance compared to SiLU.

Table 7: Classification performance of ResNet32 and ResNet44 on noisy CIFAR10 and CIFAR100. We report the average mean accuracy over 30 runs.

Data	Model	Activation	Total	Noise	Blur	Digital	Weather	Extra
CIFAR10-C	ResNet32	ReLU	72.00	53.07	67.62	74.75	80.44	74.41
CIFAR10-C	ResNet32	LReLU	72.01	52.66	**67.86**	**74.77**	80.42	74.5
CIFAR10-C	ResNet32	PReLU	71.7	52.82	67.06	74.72	79.90	74.17
CIFAR10-C	ResNet32	SiLU	71.7	52.37	67.21	74.08	80.51	74.38
CIFAR10-C	ResNet32	HardSiLU	71.32	52.75	66.75	73.39	79.66	74.28
CIFAR10-C	ResNet32	Mish	71.96	53.09	67.42	74.3	80.57	74.64
CIFAR10-C	ResNet32	GELU	71.64	52.44	67.18	74.11	80.26	74.26
CIFAR10-C	ResNet32	LA-SiLU	**72.8**	**54.02**	**68.42**	**75.13**	**81.36**	**75.51**
CIFAR10-C	ResNet32	LA-HardSiLU	**72.6**	**55.36**	67.71	74.70	**80.53**	**75.62**
CIFAR10-C	ResNet44	ReLU	**73.71**	**56.39**	**70.03**	**76.05**	81.27	75.92
CIFAR10-C	ResNet44	LReLU	**73.69**	56.03	**70.10**	**76.09**	**81.43**	75.81
CIFAR10-C	ResNet44	SiLU	72.45	53.12	68.64	74.80	80.88	75.06
CIFAR10-C	ResNet44	HardSiLU	72.63	55.51	68.72	74.73	79.86	75.34
CIFAR10-C	ResNet44	Mish	72.79	53.74	68.86	75.22	81.18	75.3
CIFAR10-C	ResNet44	GELU	72.82	54.65	68.69	75.26	80.85	75.24
CIFAR10-C	ResNet44	LA-SiLU	73.5	55.29	69.14	75.73	**81.91**	**76.19**
CIFAR10-C	ResNet44	LA-HardSiLU	73.33	**57.45**	68.48	75.30	80.64	**76.30**
CIFAR100-C	ResNet32	ReLU	43.51	24.10	41.99	45.81	50.58	44.32
CIFAR100-C	ResNet32	LReLU	43.58	23.8	42.12	45.94	50.73	44.42
CIFAR100-C	ResNet32	PReLU	42.44	23.72	40.31	44.81	49.42	43.27
CIFAR100-C	ResNet32	SiLU	42.94	23.06	41.27	45.01	50.31	44.02
CIFAR100-C	ResNet32	HardSiLU	42.67	23.64	40.87	44.87	49.54	43.71
CIFAR100-C	ResNet32	Mish	42.95	22.69	41.53	45.05	50.37	43.97
CIFAR100-C	ResNet32	GELU	43.00	23.15	41.45	45.21	50.21	43.94
CIFAR100-C	ResNet32	LA-SiLU	**44.6**	**24.16**	**43.58**	**46.74**	**52.11**	**45.51**
CIFAR100-C	ResNet32	LA-HardSiLU	**44.86**	**25.98**	**43.83**	**47.02**	**51.50**	**45.81**
CIFAR100-C	ResNet44	ReLU	44.77	25.45	43.05	47.33	51.94	45.44
CIFAR100-C	ResNet44	LReLU	44.8	25.57	43.26	47.17	51.91	45.5
CIFAR100-C	ResNet44	PReLU	44.31	25.78	42.19	46.7	51.44	44.97
CIFAR100-C	ResNet44	SiLU	44.04	24.52	42.61	46.16	51.08	45.01
CIFAR100-C	ResNet44	HardSiLU	44.11	26.22	42.77	46.29	50.32	44.9
CIFAR100-C	ResNet44	Mish	44.14	24.18	42.69	46.36	51.37	45.11
CIFAR100-C	ResNet44	GELU	43.93	24.39	42.29	46.15	51.02	44.85
CIFAR100-C	ResNet44	LA-SiLU	**46.12**	**26.68**	**44.94**	**48.32**	**53.44**	**46.89**
CIFAR100-C	ResNet44	LA-HardSiLU	**46.83**	**30.85**	**45.83**	**48.93**	**52.5**	**47.35**

Table 7 illustrates the classification performance of ResNet32 and ResNet44 on CIFAR10-C and CIFAR100-C datasets. We do not report the results for ResNet44 with PReLU on CIFAR10 due to network instability during training. LayerAct functions outperformed the element-level activation functions in all but one instance, specifically ResNet44 on CIFAR10-C. However, it is noteworthy that LA-SiLU and LA-HardSiLU demonstrated much robust inference compared to their corresponding element-level activation functions, SiLU and HardSiLU, in all cases. Tables

Table 8: Standard deviation of ResNet20s' accuracy on CIFAR10 and CIFAR10-C with different activation functions. We average in the table.

Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
ReLU	0.2120	1.3171	2.4428	1.5145	0.9736	0.7231	1.2127
LReLU	0.2635	1.2630	2.2750	1.4947	0.8948	0.6938	1.2099
PReLU	0.2323	1.4176	2.4859	1.4106	1.3038	0.8178	1.3369
SiLU	0.2057	1.1067	1.7695	1.3471	0.8738	0.7012	1.0076
HardSiLU	0.2287	1.2052	2.3675	1.4039	0.8074	0.6465	1.0914
MISH	0.2302	1.1507	2.1373	1.3116	0.7921	0.6335	1.1255
GELU	0.1868	1.1358	1.8216	1.5201	0.8537	0.5251	1.1298
ELU	0.1742	1.1673	2.2787	1.3572	0.7776	0.5961	1.1047
LA-SiLU	0.1645	1.1800	1.8781	1.4681	0.9738	0.7475	1.0069
LA-HardSiLU	0.2051	1.1765	2.3889	1.1128	0.9142	0.7081	1.0616

Table 9: Standard deviation of ResNet32s' accuracy on CIFAR10 and CIFAR10-C with different activation functions. We average in the table.

Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
ReLU	0.3880	1.5955	2.8978	1.7971	1.3270	0.9254	1.3556
LReLU	0.3356	1.7599	3.0043	2.1394	1.4202	1.0200	1.5270
PReLU	0.2809	1.6875	2.6927	1.8571	1.6496	0.9261	1.5634
SiLU	0.2188	1.3848	2.7176	1.7485	0.8207	0.7220	1.2483
HardSiLU	0.2200	1.2341	2.0973	1.5508	0.8625	0.8104	1.0653
MISH	0.2497	1.2869	2.1864	1.7262	0.8797	0.6475	1.2194
GELU	0.1568	1.2598	2.3925	1.4798	0.8745	0.6164	1.2188
ELU	0.1800	1.2041	1.8703	1.5525	0.9373	0.7095	1.1177
LA-SiLU	0.1898	1.2232	1.9876	1.4006	1.1328	0.6804	1.1056
LA-HardSiLU	0.3023	1.4008	2.2405	1.7160	1.1416	0.8028	1.3129

8, 9, 10, 11, 12, and 13 demonstrate the standard deviation of the networks on the CIFAR and CIFAR-C dataset.

Tables 14 and 15 present the results of a statistical significance test between the accuracy of networks with element-level activation functions and those with LA-SiLU functions on clean CIFAR10 and CIFAR100. Tables 16 and 17 present the corresponding results of networks with LA-SiLU and LA-HardSiLU on CIFAR10-C and CIFAR100-C. RN20, RN32, and RN44 denotes ResNet20, ResNet32, and ResNet44. We do not report the experiments of ResNet44 with PReLU on CIFAR10 and CIFAR10-C as a network exploded during training. When the accuracies of both functions were normally distributed, we performed a T-test. In cases where at least one of them are not, we performed a Wilconxon signed-rank test otherwise. The notation '> 0.05' indicates that the p-value from either a T-test or a Wilcoxon signed-rank test is larger than the standard significance level of 0.05 (i.e. p-value > 0.05).

K Additional Figures

In this section, we present additional tables and figures extracted from the experiments. Figure 3 presents the distribution of the mean activation input. As observed in the mean of activation input at epoch 40 (right), LayerAct functions promote the training of parameter W such that the output of the linear projection $y = W^T x$, which is also activation input, gets closer to zero compared to other functions. This helps the activation output to exhibit a 'zero-like' behaviour.

Table 10: Standard deviation of ResNet44s' accuracy on CIFAR10 and CIFAR10-C with different activation functions. We average in the table.

Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
ReLU	0.6249	2.1269	3.5833	2.4709	1.8301	1.2399	1.8742
LReLU	0.4437	1.9929	3.3751	2.3684	1.6750	1.1650	1.7264
PReLU	14.5207	11.3037	8.8671	10.8069	11.8174	12.5987	11.8192
SiLU	0.2435	1.4765	2.6747	1.7969	1.0507	0.8144	1.3453
HardSiLU	0.4192	1.7214	3.2993	1.8791	1.3047	1.0844	1.4341
MISH	0.4472	2.1393	3.7470	2.5737	1.6741	1.2354	1.8683
GELU	0.4696	1.6769	3.1032	1.8889	1.2754	0.8846	1.5892
ELU	0.2073	1.4907	2.7899	1.7389	1.1132	0.7598	1.3765
LA-SiLU	0.2719	1.3634	2.5109	1.5678	1.0625	0.7445	1.2180
LA-HardSiLU	0.2033	1.5738	2.8731	1.7762	1.3225	0.8906	1.3316

Table 11: Standard deviation of ResNet20s' accuracy on CIFAR100 and CIFAR100-C with different activation functions. We average in the table.

Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
ReLU	0.2937	0.8120	1.0754	0.9131	0.7956	0.5861	0.7558
LReLU	0.3233	0.8633	1.0476	0.8994	0.7943	0.7862	0.8353
PReLU	0.4435	0.9101	1.2044	1.0421	0.8121	0.7970	0.7682
SiLU	0.4147	0.7665	1.0922	0.8286	0.6642	0.5197	0.8095
HardSiLU	0.3903	0.8104	1.1921	0.8172	0.6649	0.6375	0.8356
MISH	0.3269	0.7400	1.0435	0.6064	0.7377	0.6462	0.7420
GELU	0.3866	0.8059	0.9036	0.9290	0.8281	0.6220	0.7714
ELU	0.3618	0.7367	1.0898	0.7222	0.6868	0.5689	0.7044
LA-SiLU	0.3493	0.8537	1.3183	0.8409	0.7137	0.6806	0.8313
LA-HardSiLU	0.4056	0.8751	1.2555	0.8963	0.7188	0.7466	0.8536

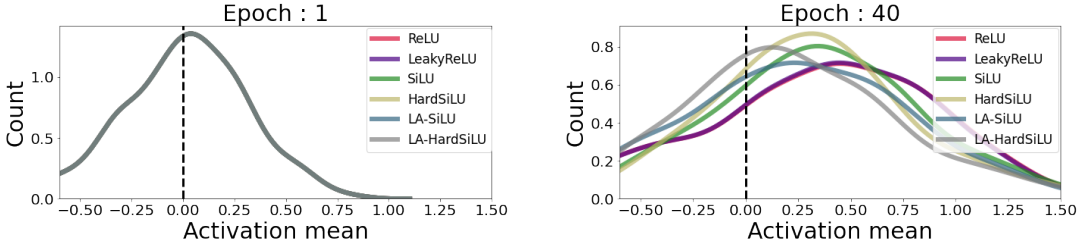

Figure 3: Distribution of the activation **input** means of the elements in a trained network on MNIST at 1^{st} and 40^{th} epochs.

LayerAct functions exhibit a significantly lower mean and variance of activation fluctuation among the samples compared to any other element-level activation function (see Figure 3 in the main manuscript). Figure 4 demonstrates that the distribution of mean fluctuation in activation input appears similar across all functions. This observation confirms that the lower mean and variance of activation output fluctuation of LayerAct functions is not due to a smaller fluctuation in activation input, but is a result of the inherent mechanism of LayerAct.

Table 12: Standard deviation of ResNet32s' accuracy on CIFAR100 and CIFAR100-C with different activation functions. We average in the table.

Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
ReLU	0.5187	1.0671	1.3930	1.1459	1.0240	0.8663	0.9876
LReLU	0.3364	0.9092	1.3616	0.8028	0.9881	0.6881	0.8184
PReLU	0.5045	0.9674	1.2581	1.0727	0.8316	0.8351	0.9123
SiLU	0.5053	1.0087	1.5275	1.0608	0.8957	0.6965	0.9926
HardSiLU	0.5325	0.9114	0.9530	1.0173	0.9229	0.9203	0.7538
MISH	0.4820	0.9148	1.2750	0.9058	0.8194	0.7269	0.9369
GELU	0.4418	0.8121	1.0506	0.9158	0.7070	0.6675	0.7790
ELU	1.4877	1.3180	1.1907	1.3131	1.3876	1.4028	1.2640
LA-SiLU	0.3776	0.8765	1.3721	0.8750	0.6403	0.6864	0.9326
LA-HardSiLU	0.4486	0.9763	1.5039	1.0204	0.8062	0.7838	0.8988

Table 13: Standard deviation of ResNet44s' accuracy on CIFAR100 and CIFAR100-C with different activation functions. We average in the table.

Activation	Clean	Total	Noise	Blur	Digital	Weather	Extra
ReLU	0.5652	1.1011	1.5926	1.1677	0.9634	0.9098	0.9947
LReLU	0.6872	1.2173	1.8513	1.1183	1.1814	0.9895	1.1046
PReLU	0.7084	1.3226	1.8320	1.4481	1.0939	1.2049	1.1616
SiLU	0.5204	0.9563	1.2746	0.9791	0.8475	0.8444	0.9157
HardSiLU	0.9417	1.4808	1.6740	1.5590	1.5011	1.3818	1.3362
MISH	0.5119	0.9386	1.3795	0.9206	0.7863	0.7879	0.9291
GELU	0.6182	0.9809	1.3545	0.9600	0.9044	0.8583	0.9207
ELU	0.4118	1.1195	1.6396	1.1471	0.9781	0.8948	1.0679
LA-SiLU	0.4366	1.1648	1.5511	1.2528	1.1577	0.9353	1.0238
LA-HardSiLU	0.7955	1.8629	2.9063	1.8458	1.6924	1.5680	1.5630

Table 14: Statistical significance test of LA-SiLU on CIFAR10 dataset.

	LA-SiLU							
	ReLU	LReLU	PReLU	SiLU	HardSiLU	Mish	GELU	ELU
RN20	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	0.002	$< 1e^{-3}$	0.011	0.016	$< 1e^{-3}$
RN32	0.02	0.011	0.005	0.331	$< 1e^{-3}$	0.422	0.125	$< 1e^{-3}$
RN44	0.014	0.001	-	0.007	$< 1e^{-3}$	0.479	0.094	$< 1e^{-3}$

Table 15: Statistical significance test of LA-SiLU on CIFAR100 dataset.

	LA-SiLU							
	ReLU	LReLU	PReLU	SiLU	HardSiLU	Mish	GELU	ELU
RN20	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	0.065
RN32	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN44	0.357	0.244	0.008	0.006	$< 1e^{-3}$	0.476	0.207	$< 1e^{-3}$

L Medical Image

In this section, we present the setting of experimental result from U-net [17] and Unet++ [23] for segmentation task on a nuclei image dataset from Data Science Bowl 2018 [4]. Detailed ex-

Table 16: Statistical significance test of LayerAct functions on CIFAR10-C dataset.

	LA-SiLU							
	ReLU	LReLU	PReLU	SiLU	HardSiLU	Mish	GELU	ELU
RN20	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN32	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN44	0.399	0.233	-	$< 1e^{-3}$	0.001	0.002	0.003	$< 1e^{-3}$

	LA-HardSiLU							
	ReLU	LReLU	PReLU	SiLU	HardSiLU	Mish	GELU	ELU
RN20	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN32	0.007	0.014	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN44	0.18	0.138	-	$< 1e^{-3}$	0.008	0.01	0.018	$< 1e^{-3}$

Table 17: Statistical significance test of LayerAct functions on CIFAR100-C dataset.

	LA-SiLU							
	ReLU	LReLU	PReLU	SiLU	HardSiLU	Mish	GELU	ELU
RN20	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN32	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN44	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$

	LA-HardSiLU							
	ReLU	LReLU	PReLU	SiLU	HardSiLU	Mish	GELU	ELU
RN20	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN32	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$
RN44	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$	$< 1e^{-3}$

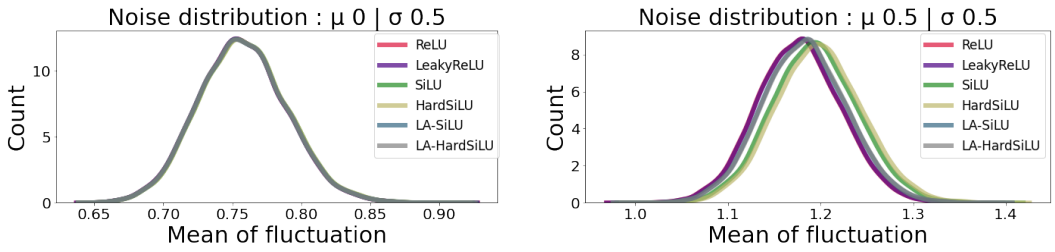

Figure 4: Distribution of activation **input** fluctuation due to noise with different noise distribution.

perimental setting is as follows: (1) Adam optimizer with $3e^{-4}$ learning rate, and $1e^{-4}$ weight decay, (2) training 100 epoches with cosine annealing scheduler, and (3) BCE-Dice Loss as the loss function. We report the average IoU (Intersection over Union; %) over 10 trials with different weight initialization in Table 4 of the main manuscript.

M Adversarial Robustness

We conducted experiments to investigate the adversarial robustness of LayerAct. We utilized ReLU, SiLU, and LA-SiLU as the activation of ResNet18. The experimental setting was based

on [22]. The experimental results in Table 4 in the main manuscript show that the ResNet20 with LA-SiLU is more robust against adversarial attacks compared to those with other activations. These results indicate that LayerAct can enhance adversarial robustness as well.

References

[1] Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). In: International Conference on Learning Representations (ICLR) (2016)

[2] Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks **107**, 3–11 (2018)

[3] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics (AISTATS). vol. 15, pp. 315–323 (11–13 Apr 2011)

[4] Goodman, A., Carpenter, A., Park, E., jlefman nvidia, Josette_BoozAllen, Kyle, Maggie, Nilofer, Sedivec, P., Cukierski, W.: 2018 data science bowl. https://kaggle.com/competitions/data-science-bowl-2018 (2018), accessed: 2024-12-17

[5] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (December 2015)

[6] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)

[7] Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus) (2023)

[8] Huang, L., Yang, D., Lang, B., Deng, J.: Decorrelated batch normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)

[9] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML). vol. 37, pp. 448–456. PMLR (2015)

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 25 (2012)

[11] Labatie, A., Masters, D., Eaton-Rosen, Z., Luschi, C.: Proxy-normalizing activations to match batch normalization while removing batch dependence. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 34, pp. 16990–17006. Curran Associates, Inc. (2021)

[12] Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS. vol. 38, pp. 562–570 (2015)

[13] Liang, S., Huang, Z., Liang, M., Yang, H.: Instance enhancement batch normalization: An adaptive regulator of batch noise. In: AAAI. vol. 34, pp. 4819–4827 (2020)

[14] Lubana, E.S., Dick, R., Tanaka, H.: Beyond batchnorm: Towards a unified understanding of normalization in deep learning. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 34, pp. 4778–4791. Curran Associates, Inc. (2021)

[15] Luo, P., Zhang, R., Ren, J., Peng, Z., Li, J.: Switchable normalization for learning-to-normalize deep representation. IEEE transactions on pattern analysis and machine intelligence **43**(2), 712–728 (2019)

[16] Misra, D.: Mish: A self regularized non-monotonic activation function (2020)

[17] Olaf Ronneberger, Philipp Fischer, T.B.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). pp. 234–241 (2015)

[18] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (NeruIPS). vol. 32 (2019)

[19] Qiu, S., Xu, X., Cai, B.: Frelu: Flexible rectified linear units for improving convolutional neural networks. In: International Conference on Pattern Recognition (ICPR). pp. 1223–1228 (2018)

[20] Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. In: International Conference on Learning Representations (ICLR) Workshop (2018)

[21] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization (2017)

[22] Zhou, N., Wang, N., Liu, D., Zhou, D., Gao, X.: Enhancing robust representation in adversarial training: Alignment and exclusion criteria (2023)

[23] Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: Unet++: A nested u-net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA). pp. 3–11 (2018)