Learning continuous piecewise non-linear activation functions for deep neural networks

1st Xinchen Gao
UESTC
Chengdu, China
gxc0327@gmail.com

2nd Yawei Li†
ETH Zurich
Zurich, Switzerland
yawei.li@vision.ee.ethz.ch

3rd Wen Li*
UESTC
Chengdu, China
liwenbnu@gmail.com

4th Lixin Duan
UESTC
Chengdu, China
lxduan@gmail.com

5th Luc Van Gool
ETH Zurich
Zurich, Switzerland
vangool@vision.ee.ethz.ch

6th Luca Benini
ETH Zurich
Zurich, Switzerland
lbenini@iis.ee.ethz.ch

7th Michele Magno
ETH Zurich
Zurich, Switzerland
michele.magno@pbl.ee.ethz.ch

2023 IEEE International Conference on Multimedia and Expo (ICME) | 978-1-6654-6891-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICME55011.2023.00315

***Abstract*—Activation functions provide the non-linearity to deep neural networks, which are crucial for the optimization and performance improvement. In this paper, we propose a learnable continuous piece-wise nonlinear activation function (or CPN in short), which improves the widely used ReLU from three directions, *i.e.*, finer pieces, non-linear terms and learnable parameterization. CPN is a continuous activation function with multiple pieces and incorporates non-linear terms in every interval. We give a general formulation of CPN and provide different implementations according to three key factors: whether the activation space is divided uniformly or not, whether the non-linear terms exist or not, and whether the activation function is continuous or not. We demonstrate the effectiveness of our method on image classification and single image super-resolution tasks by simply changing the activation function. For example, CPN improves 4.78% / 4.52% top-1 accuracy over ReLU on MobileNetV2_0.25 / MobileNetV2_0.35 for ImageNet classification and achieves better PSNR on several benchmarks for super-resolution. Our implementation is available at https://github.com/xc-G/CPN.**

***Index Terms*—Deep neural networks, piecewise activation functions, non-linear terms**

I. Introduction

Advances in deep neural networks have triggered vigorous development in many fields, especially computer vision. During the past decade, many excellent works on model design have emerged for computer vision [1]–[8]. When we concentrate on the details of model architecture, non-linear activation functions are important functional building blocks for deep neural networks. They provide generality to deep neural networks by introducing the capability of modeling non-linearity between inputs and outputs of the network. The rectified linear unit (ReLU) [9], one of the most popular non-linear activation functions, is simple and effective. ReLU can be regarded as two linear units with different slopes and intercepts. ReLU helps to solve the gradient vanishing problem in deep neural networks with Tanh or Sigmoid activation functions by avoiding the saturation area.

Recently, some works [10]–[13] generalize ReLU to piecewise activation functions. For example, MTLU [10] uniformly divides the activation space into multiple pieces and parameterizes the slope and intercept of the linear unit in each section independently. Yet, since each small section of MTLU is learned independently from the others, the function is not continuous. In addition, MTLU only introduces linear terms into each section, which restricts the expressiveness of the deep neural networks. Piecewise activation functions like SELU [14], ELU [15], and Hard-swish [5] introduce differentiable non-linear terms such as $\sigma(x)$, x^2. However, these functions either tend to have too few pieces, or no learnable parameters, which limits their non-linear capacity.

Thus, in this paper, we try to solve the aforementioned problems and further improve the nonlinear capacity of deep neural networks. We propose a learnable **c**ontinuous **p**iecewise **n**on-linear (CPN) activation function to comprehensively optimize ReLU from three aspects: finer pieces, non-linear terms and learnable parameterization. Concretely, the activation space of CPN is divided into multiple sections. In each section, based on previous piecewise activation functions only with linear units, we incorporate a non-linear term to construct a hybrid unit in every interval. Both linear and non-linear terms are associated with learnable parameters. The introduced non-linear terms further improve the expressiveness of the networks. In addition, we enforce continuity constraints on the proposed piecewise non-linear activation functions for

†Equal contribution, *corresponding author.

better optimization. Beyond that, a generalized formulation of CPN is proposed. Previous piecewise activation functions like MTLU and Hard-swish can be considered as special cases of the generalized formulation. For the implementation, we explore different versions of CPN according to three key factors: continuity, uniform division of activation space, and the existence of the non-linear terms.

We evaluate the proposed CPN for image classification and single image super-resolution tasks. For image classification, by replacing the original ReLU in MobileNetV2_0.25 / MobileNetV2_0.35 with CPN, the top-1 accuracy is improved by 4.78% / 4.52%. For the super-resolution task, the evaluation metric PSNR is improved on several standard benchmarks, which demonstrates the effectiveness of our method.

In summary, our major contributions are as follows:

1) We propose a continuous piecewise non-linear activation function to further improve the non-linear capacity.
2) The proposed CPN can be regarded as the generalized version of existing piecewise activation functions.
3) The results on different vision tasks demonstrate the effectiveness of our proposed method.

II. Related Work

Non-linear activation functions are important building blocks of deep neural networks. Without them, deep neural networks can only capture linear relationships between the input and output. Sigmoid function and Tanh are first applied to multilayer perceptrons. The simple and effective ReLU is proposed to solve the gradient vanishing problem of Sigmoid and Tanh and becomes the most popular activation function.

Recently, there are plenty of works to improve ReLU from different perspectives. Leaky ReLU [16] uses a fixed nonzero slope for the negative input, and PReLU [17] parameterizes this slope to adaptively learn the parameters of the rectified units. MTLU [10], PWLU [11], Maxout [12] and DyReLU [13] generalize the two-piece ReLU to activation functions with multiple pieces. MTLU learns the slope and intercept of the linear unit in each section independently [10]. PWLU enforces a continuity constraint between different pieces of the activation function [11]. Besides learning the shape parameters such as the boundary points, slopes, and mapping value of the division points, a statistical realignment strategy is also proposed for PWLU to align the effective region with the input data. Maxout calculates multiple learnable linear mappings for every input value and takes the maximum as the activated value [12]. DyReLU [13] is the dynamic version of Maxout. It utilizes a hypernet to adaptively learn dynamic parameters for the multiple linear units. However, these piecewise activation functions only involve linear units within every single interval. In this paper, we propose to introduce non-linear terms for every interval.

Some works [5], [14], [15], [18] introduce differential non-linear terms to activation functions. For example, the second piece of the three-piece Hardswish is a non-linear term $x \times (x+3)/6$. However, these functions either have too few pieces, or have no learnable parameters.

DyReLU [13] and xUnit [19] are dynamic functions whose coefficients are calculated over inputs. Squeeze-and-Excitation units also strengthen networks from the same perspective [20]. Yet, these methods resort to additional convolution neural networks to achieve the dynamic mechanism, which introduces much more parameters into the network compared with the proposed method.

Aiming at solving the problems mentioned above, we propose our continuous learnable piecewise non-linear activation function. We comprehensively optimize ReLU from the aspects of more pieces, non-linear terms, and learnable parameterization.

III. Methodology

In this paper, we propose a learnable continuous piecewise non-linear activation function (CPN) that introduces non-linear terms in each small section under continuity constraint. We can formulate the CPN as follows:

$$\mathrm{CPN}(x) = \begin{cases} a_1 * x + b_1 * f(x) + c_1, x \le x_1, \\ a_2 * x + b_2 * f(x) + c_2, x_1 < x \le x_2, \\ \quad ... \\ a_k * x + b_k * f(x) + c_k, x_{k-1} < x. \end{cases} \tag{1}$$

where $\{a_i, b_i, c_i\}_{i=1,2,...,k}$ are learnable parameters, $f(x)$ is the non-linear term such as Sigmoid function. The activation space $(-\infty, +\infty)$ are divided into k sections $(-\infty, x_1], (x_1, x_2], (x_2, x_3], ..., (x_{k-1}, +\infty)$ by $k-1$ endpoints $x_1, x_2, ..., x_{k-1}$. Interestingly, activation functions like ReLU and Hard-Swish can be regarded as special cases of CPN. The proposed CPN has three key factors determining the shape of activation:

- Whether the activation space is divided uniformly or not, that is, whether $x_2 - x_1 = ... = x_{k-1} - x_{k-2} = d$.
- Whether the non-linear terms exit or not, that is, whether $\{b_i = 0\}_{i=1,...,k}$.
- Whether the activation function is continuous or not, that is, whether $\mathrm{CPN}(x_i^-) = \mathrm{CPN}(x_i^+)$.

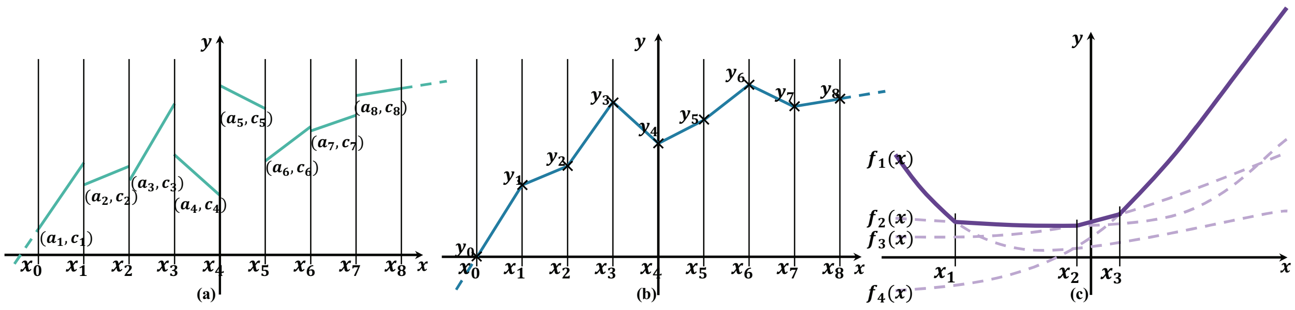
Based on these three key factors, we get different versions of CPN, and we will introduce them in turn.

A. *CPN$_m$*

When we divide the activation space uniformly, remove the non-linear terms and get rid of the continuity constraint, the CPN degenerates to the MTLU which we also call CPN$_m$. Therefore, the formulation of MTLU (CPN$_m$) is:

$$\mathrm{CPN}_m(x) = \begin{cases} a_1 * x + c_1, x \le x_1, \\ a_2 * x + c_2, x_1 < x \le x_2, \\ \quad ... \\ a_k * x + c_k, x_{k-1} < x. \end{cases} \tag{2}$$

In MTLU, the activation space is divided uniformly. Specifically, we usually divide the interval $[-1, 1]$ into k intervals by $k-1$ endpoints points $x_1, ..., x_{k-1}$. We additionally define $x_0 = -1, x_k = 1$, so the k intervals' range are $(x_0, x_1], (x_1, x_2], ..., (x_{k-1}, x_k]$, respectively. In each interval,



(a)
(b)
(c)
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x y
(a_1,c_1) (a_2,c_2) (a_3,c_3) (a_4,c_4) (a_5,c_5) (a_6,c_6) (a_7,c_7) (a_8,c_8)
y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
$f_1(x)$ $f_2(x)$ $f_3(x)$ $f_4(x)$

Fig. 1. The sketch of MTLU, CPN$_{mc}$ and CPN$_{nl}$. **(a) MTLU of $k = 8$.** In MTLU, the linear unit of each small interval is learned independently, so the shape of MTLU is usually not continuous. **(b) CPN$_{mc}$ of $k = 8$.** In CPN$_{mc}$, we learn the mapping values of endpoints of all intervals, so the shape is continuous. **(c) CPN$_{nl}$ of $k = 4$.** In CPN$_{nl}$, we learn the coefficients of k functions and take the maximum as the activated value, so the shape is continuous. However, the activation space is not divided uniformly.

MTLU constructs a linear unit with the learnable parameters of slope a_i and intercept c_i. Therefore, the shape of MTLU is usually not continuous. Note that for the inputs less than or equal to x_0, they share the same parameters with the section $(x_0, x_1]$. Similarly, the inputs greater than x_k share the same parameters with the section $(x_{k-1}, x_k]$. It is worth noting that the MTLU (CPN$_m$) is channel-wise. Thus, for the feature map of C channels, the number of parameters of MTLU is $C*k*2$. We present an example of MTLU (CPN$_m$) of 8 pieces in Fig. 1 (a).

B. CPN$_{mc}$

When we impose continuity constraint on MTLU (CPN$_m$), we can obtain a continuous version of MTLU termed CPN$_{mc}$. In MTLU, since the linear unit of each section is learned independently, the resulting shape of the MTLU is not continuous, that is MTLU(x_i^-) $\neq$ MTLU(x_i^+). The formulation of CPN$_{mc}$ is the same as MTLU. However, in CPN$_{mc}$, we force the function to be continuous at the endpoints of all intervals, *i.e.* CPN(x_i^-) = CPN(x_i^+).

For simplicity, we adopt a similar implementation to PWLU. Specifically, instead of learning the slope and intercept of each small interval independently, we parameterize the mapping value of the endpoints of each small interval in CPN$_{mc}$. Specifically, for the k intervals $(x_0, x_1], (x_1, x_2], ..., (x_{k-1}, x_k]$, where we fix $x_0 = -1, x_k = 1$ like MTLU, we learn the mapping value of the right endpoint of each small interval. That is, we fix $y_0 = 0$, and define k learnable parameters $\{y_i\}_{i=1,2,...,k}$. Then we force the CPN$_{mc}$ to pass by the $k+1$ coordinates $[(x_0, y_0), (x_1, y_1), ..., (x_k, y_k)]$, which determine the shape of the linear unit of each interval. For each small interval, the slope and intercept can be calculated by:

$$\begin{aligned} a_i &= y_i - y_{i-1}/x_i - x_{i-1}, \\ c_i &= y_i - x_i(y_i - y_{i-1})/(x_i - x_{i-1}). \end{aligned} \tag{3}$$

For the inputs out of the range of $[-1, 1]$, we adopt a similar strategy to MTLU. In addition, CPN$_{mc}$ is applied channel-wise like MTLU. Thus, for the feature map of C channels, the number of parameters of MTLU is $C * k * 1$, only half of MTLU. When we train CPN$_{mc}$ with back-propagation algorithm, the gradient of CPN$_{mc}(x)$ with respect to y_i can be calculated by Eqn. 4 simply according to the forward pass.

$$\partial \text{CPN}_{mc}(x)/\partial y_i = (x - x_{i-1})/(x_i - x_{i-1}). \tag{4}$$

Since the slope and intercept of the linear unit within each small interval are determined by the mapping values of endpoints, the shape of CPN$_{mc}$ is surely continuous. In Fig. 1 (b), we show the sketch of CPN$_{mc}$.

C. CPN$_{nl}$

Further, when we introduce the non-linear term for each small interval and remove the constraint on the uniform division of activation space, we obtain the piecewise non-linear activation function denoted by CPN$_{nl}$.

For the non-linear term $f(x)$ introduced in CPN$_{nl}$, we resort to the Sigmoid linear unit (SiLU), a simple and effective non-linear activation function, which is formulated as:

$$f(x) = \text{SiLU}(x) = x * \sigma(x). \tag{5}$$

Where σ is the Sigmoid function.

During training, to implement the continuous CPN$_{nl}$ simply, we adopt a similar strategy to ReLU. ReLU can be viewed as calculating two linear mappings for each input value and taking the maximum one as the activated value, which can be written as:

$$\begin{aligned} \text{ReLU}(x) &= max(w_1 x + b_1, w_2 x + b_2), \\ s.t.(w_1, b_1) &= (0, 0), (w_2, b_2) = (1, 0). \end{aligned} \tag{6}$$

Therefore, based on ReLU, we resort to k functions to calculate k mapping values for each input value and the maximum is taken as the activated value. We formulate CPN$_{nl}$ as:

$$\begin{aligned} \text{CPN}_{nl}(x) &= \max\{p_1(x), p_2(x), ..., p_k(x)\}, \\ p_i(x) &= a_i * x + b_i * x * \sigma(x) + c_i, (i = 1, ..., k). \end{aligned} \tag{7}$$

where k is the number of functions, $\{a_i, b_i, c_i\}_{i=1,..,k}$ are learnable coefficients. Since the intersections of the k activation functions are not evenly distributed, the lengths of the intervals are not equal. During inference, there are two ways to implement the CPN$_{nl}$. One is to adopt the same implementation as the training phase, and the other is to compute the intersection points of the k functions in advance, then use the Eqn. 1 to do forward-propagation. Specifically, after training, we can obtain the division points of the activation space $d_1, ..., d_m$. Then specific activation function $u(x)$ of each interval can be determined. Therefore, we can compute the activated values by the specific activation function directly instead of repeating k calculations in each interval.

Similar to the previous two activation functions, CPN$_{nl}$ is also applied channel-wise. Thus, for the feature map of C channels, the number of parameters of MTLU is $C * k * 3$. We plot the skecth of CPN$_{nl}$ in Fig. 1 (c).

IV. Experimental Results

We verify the effectiveness of our proposed CPN for the image classification task and transfer to the single image super-resolution task on several standard datasets.

A. Datasets and evaluation metrics

Image classification. We conduct image classification experiments on the ImageNet1K dataset, which is the most popular dataset in the field of computer vision. ImageNet1K contains a total of 1,000 categories, consisting of about 1.28M training images, and 50,000 validation images. The evaluation metrics for image classification we use are Top-1 accuracy and Top-5 accuracy on the validation dataset.

Single image super-resolution. For the super-resolution task, we follow the routine, taking 800 training images from DIV2K [21] as the training set. Then we evaluate our methods on the DIV2k validation set containing 100 images, Set5 [22], Set14 [23], B100 [24], and Urban100 [25], [26]. As for the evaluation metric, We evaluate the results with peak signal-to-noise ratio (PSNR).

B. Experimental setup

We implement all experiments using PyTorch 1.7.1 on Nvidia Geforce RTX 2080Ti GPUs.

Implementation details. As mentioned in Sec. III, the hyper-parameter k represents the number of pieces (or the number of functions) in our CPN. In our experiments, we set k to 20 for MTLU and CPN$_{mc}$, 4 for CPN$_{nl}$, respectively. Therefore, for a feature map of C channels, the numbers of parameters of MTLU, CPN$_{mc}$ and CPN$_{nl}$ are $40 * C$, $20 * C$ and $12 * C$, respectively. Compared to a standard 3×3 convolution with the number of parameters of $9 * C^2$, the number of parameters of our proposed CPN is quite little.

Image classification. We train the models for 150 epochs with the batch size set to 512. The initial learning rate is set to 0.1, the SGD optimizer with a momentum of 0.9 is used and the cosine learning rate scheduler is used to decay the learning rate to 0.

Single image super-resolution. We train models from scratch for $\times 2$ SR, and for $\times 3$ and $\times 4$ SR, we train models from corresponding pretrained $\times 2$ model models. As for training details, the models are trained for 300 epochs with a batch size of 16, using Adam optimizer with $\beta = (0.9, 0.999)$. We set the initial learning rate to 1×10^{-4} and utilize the MultiStep scheduler to decay the learning rate to half every 200 epochs.

TABLE I
Image classification accuracy on the ImageNet validation set of models with different activation functions. The best results are **bolded**.

model	activation	top1-acc	top5-acc
MobileNetV2_0.25	ReLU	52.75%	76.43%
	MTLU	55.13%	77.69%
	CPN$_{mc}$	55.52%	78.43%
	CPN$_{nl}$	**57.53%**	**80.10%**
MobileNetV2_0.35	ReLU	59.15%	81.84%
	MTLU	60.73%	82.26%
	CPN$_{mc}$	61.30%	83.19%
	CPN$_{nl}$	**63.67%**	**84.81%**

TABLE II
Super-resolution results for $\times 2$, $\times 3$, $\times 4$ scales on DIV2K, Set5, Set14, B100 and Urban100 benchmarks. PSNR (in dB) is used as the evaluation metric, and the best results are **bolded**.

scale	activation	training strategy	DIV2K	Set5	Set14	B100	Urban100
$\times 2$	ReLU	from scratch	34.54	37.92	33.50	32.14	31.88
	MTLU	from scratch	34.56	37.95	**33.54**	32.15	31.93
	CPN$_{nl}$	from scratch	**34.58**	**38.00**	33.51	**32.15**	**31.94**
$\times 3$	ReLU	from x2	30.90	34.32	30.28	29.08	28.09
	MTLU	from x2	30.90	34.35	30.29	29.08	28.10
	CPN$_{nl}$	from x2	**30.95**	**34.40**	**30.31**	**29.10**	**28.14**
$\times 4$	ReLU	from x2	28.94	**32.15**	28.58	27.56	26.02
	MTLU	from x2	28.94	32.05	28.55	27.57	26.01
	CPN$_{nl}$	from x2	**28.98**	32.11	**28.59**	**27.58**	**26.07**

C. Experiment results

Image classification. For image classification, we take the lightweight MobileNetV2_0.25 (0.35) with the width multiplier of 0.25 (0.35) as the baseline model. Similarly, we compare our proposed CPN with other activation functions by replacing the original ReLU6 activation function. We denote the models by MobileNetV2_"w"_"af" ("w" is the width multiplier and "af" is the specific activation function). The results shown in Tabel I demonstrate that our CPN gains significant performance improvement over ReLU and MTLU. Our MobileNetV2_0.25_CPN$_{mc}$/CPN$_{nl}$ achieve 55.52%/57.53% Top-1 accuracy on ImageNet, being 2.77%/4.78% higher than MobileNetV2_0.25_ReLU and 0.39%/2.40% higher than MobileNetV2_0.25_MTLU, respectively. When the width multiplier is set to 0.35, our CPN$_{nl}$ improves Top-1 accuracy on the ImageNet by 4.52% and 2.94% over ReLU and MTLU, respectively. When compared to DyReLU on MobileNetV2_0.35, the improvement achieved by CPN_$_{nl}$ is 1.56% higher than that of DyReLU-A, and the number of parameters is less in our method. We visualize the Top-1 ImageNet validation accuracy curve and the training loss curve during training phase in Fig. 2. It obvious that our proposed CPN leads to a higher validation accuracy and a lower training loss than other activation functions during

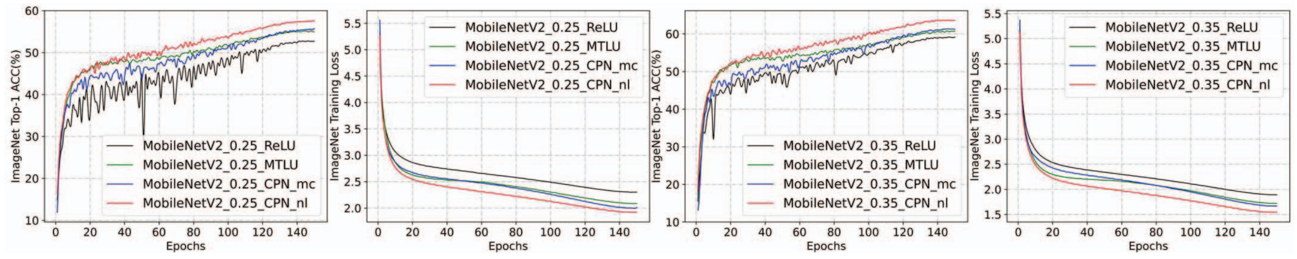


Fig. 2. The comparison of Top-1 validation accuracy and training loss between our proposed CPN and other activation functions on MobileNetV2_0.25 and MobileNetV2_0.35. The red line is our CPN$_{nl}$ and the blue line is our CPN$_{mc}$.

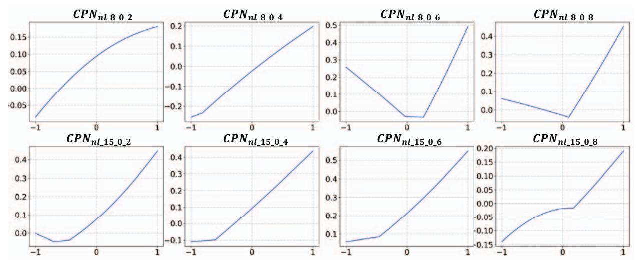


Fig. 3. Some examples of learned CPN$_{nl}$ in MobileNetV2_0.25. We present CPN$_{nl\_f\_g\_h}$ for the h-th channel of the g-th layer in the f-th block.

TABLE III
THE COMPARISONS OF DIFFERENT TRAINING STRATEGY FOR $\times 2$ SUPER-RESOLUTION TASK.

activation	training strategy	DIV2K	Set5	Set14	B100	Urban100
ReLU	from scratch	34.54	37.92	33.50	32.14	31.88
MTLU	from scratch	34.56	37.95	33.54	32.15	31.93
	finetune	34.64	37.90	33.56	32.18	32.05
CPN$_{nl}$	from scratch	34.58	38.00	33.51	32.15	31.94
	finetune	34.64	37.98	33.56	32.18	32.06

training. In Fig. 3, we present some examples of the CPN$_{nl}$ in MobileNetV2_0.25_CPN$_{nl}$. All the activation functions show non-linearity.

Single image super-resolution. The EDSR baseline single-scale model with 64 feature maps in each layer and 16 basic blocks is taken as our baseline model. All the models are denoted by CPN_“af” (“af” is the specific activation function). We construct different models by replacing the original ReLU with our CPN and other activation functions for comparison. The $\times 2$, $\times 3$ and $\times 4$ SR results are shown in Tab. II. It is obvious that our method outperforms the original ReLU and MTLU specially designed for super-resolution. Specifically, we surpass ReLU by 0.04 to 0.05dB on DIV2K, about 0.08dB on Set5, 0.01dB on Set14, 0.01 to 0.02dB on B100 and 0.05 to 0.06dB on Urban100, respectively.

D. Ablation Study

Training strategy. Considering that we only replace the activation functions in the model instead of changing the model architecture. We also adopt a finetuning strategy in which we only train the learnable parameters of introduced activation functions and freeze other model parameters from the pretrained baseline model. The experiments on $\times 2$ SR demonstrate the superiority of the finetuning strategy. The results are shown in Tab. III. Specifically, when trained with the fineutuning strategy from pretrained baseline model, both MTLU and our proposed CPN$_{nl}$ outperform the corresponding models and baseline model that are trained from scratch significantly.

TABLE IV
THE COMPARISON OF DIFFERENT IMPLEMENTATIONS FOR $\times 2$ SUPER-RESOLUTION.

training strategy	activation	DIV2K	Set5	Set14	B100	Urban100
from scratch	CPN$_{nl-1}$	34.56	37.96	33.48	32.14	31.91
	CPN$_{nl-2}$	34.54	37.92	33.47	32.13	31.91
	CPN$_{nl}$	34.58	38.00	33.51	32.15	31.94
finetune	CPN$_{nl-1}$	34.63	37.96	33.56	32.17	32.03
	CPN$_{nl-2}$	34.60	37.95	33.55	32.16	32.03
	CPN$_{nl}$	34.64	37.98	33.56	32.18	32.06

TABLE V
COMPARISON OF DIFFERENT VALUES OF k FOR $\times 2$ SUPER-RESOLUTION.

k	DIV2K	Set5	Set14	B100	Urban100	average
2	34.57	37.96	33.49	32.15	31.98	34.03
4	34.58	38.00	33.51	32.15	31.94	34.04
8	34.59	37.99	33.50	32.16	31.99	34.05

Implementations. We adopt different implementations of CPN introducing the non-linear term. Instead of the simple implementation introduced in Sec. III-C, we also provide another two versions of CPN$_{nl}$, termed CPN$_{nl-1}$ and CPN$_{nl-2}$, respectively. For CPN$_{nl-1}$, we adopt a similar implementation to MTLU, parameterizing the coefficients of each small interval. For CPN$_{nl-2}$, we learn the mapping value of the endpoints of intervals and force the activation function to pass the points, then the calculation of coefficients is converted to the simple equation-solving problem. We compare different ways of implementation for $\times 2$ super-resolution. The result is shown in Tab. IV, demonstrating the effectiveness of our adopted implementation.

Value of k. k in Formulation 1 determines the number of pieces in CPN. We compare different values of k of CPN$_{nl}$, in which k is the number of functions. The results in Tab. V demonstrate that with the increase of the number of pieces, the expression capacity of models improves slightly.

V. Conclusion

In this paper, we propose a general learnable continuous piecewise non-linear activation function called CPN. We provide a general formulation of piecewise activation functions, introduce the non-linear term to each small interval and impose a continuity constraint. The extensive experiments on several benchmarks verify the effectiveness and superiority of our proposed activation function for different vision tasks including image classification and single-image super-resolution.

Acknowledgment

This work is supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, the National Natural Science Foundation of China (No. 62176047, 82121003), Shenzhen Fundamental Research Program (No. JCYJ20220530164812027), Sichuan Science and Technology Program (No. 2021YFS0374, 2022YFS0600), and the Fundamental Research Funds for the Central Universities Grant (No. ZYGX2021YGLH208).

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. CVPR*. IEEE, 2016, pp. 770–778.
[2] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *IEEE TPAMI*, vol. 38, no. 2, pp. 295–307, 2015.
[3] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*. IEEE, 2018, pp. 4510–4520.
[4] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. CVPRW*. IEEE, 2017, pp. 136–144.
[5] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al., "Searching for mobilenetv3," in *Proc. ICCV*. IEEE, 2019, pp. 1314–1324.
[6] 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 *Proc. ICCV*. IEEE, 2021, pp. 10012–10022.
[7] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool, "Localvit: Bringing locality to vision transformers," *arXiv preprint arXiv:2104.05707*, 2021.
[8] Yawei Li, Yuchen Fan, Xiaoyu Xiang, Denis Demandolx, Rakesh Ranjan, Radu Timofte, Luc Van Gool, "Efficient and Explicit Modelling of Image Hierarchies for Image Restoration," in *Proc. CVPR*. IEEE, 2023.
[9] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814
[10] Shuhang Gu, Wen Li, Luc Van Gool, and Radu Timofte, "Fast image restoration with multi-bin trainable linear units," in *Proc. ICCV*. IEEE, 2019, pp. 4190–4199.
[11] Yucong Zhou, Zezhou Zhu, and Zhao Zhong, "Learning specialized activation functions with the piecewise linear unit," in *Proc. ICCV*. IEEE, 2021, pp. 12095–12104.
[12] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," in *ICML*. PMLR, 2013, pp. 1319–1327.
[13] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu, "Dynamic relu," in *Proc. ECCV*. Springer, 2020, pp. 351–367.
[14] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter, "Self-normalizing neural networks," in *NIPS*, vol. 30, 2017.
[15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
[16] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al., "Rectifier nonlinearities improve neural network acoustic models," in *ICML*. Atlanta, Georgia, USA, 2013, vol. 30, p. 3.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*. IEEE, 2015, pp. 1026–1034.
[18] Prajit Ramachandran, Barret Zoph, and Quoc V Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.
[19] Idan Kligvasser, Tamar Rott Shaham, and Tomer Michaeli, "xunit: Learning a spatial activation function for efficient image restoration," in *Proc. CVPR*. IEEE, 2018, pp. 2433–2442.
[20] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*. IEEE, 2018, pp. 7132–7141.
[21] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *Proc. CVPRW*. IEEE, 2017, pp. 114–125.
[22] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.
[23] Roman Zeyde, Michael Elad, and Matan Protter, "On single image scale-up using sparse-representations," in *Proc. Curves and Surfaces*. Springer, 2010, pp. 711–730.
[24] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. ICCV*. IEEE, 2001, vol. 2, pp. 416–423.
[25] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. CVPR*. IEEE, 2015, pp. 5197–5206.
[26] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*. PMLR, 2015, pp. 448–456.