International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 13, Issue 2 [February. 2024] PP: 64-72

Channel Pooling Dynamic Activation Unit for Image Feature Learning

Ruipeng Yan[1], and Wenju Wang[1,*]
[1]*College of Communication and Art Design, University of Shanghai for Science and Technology, Shanghai 200093,China*
[*]*Corresponding author*

Abstract:
Activation function is an important factor that affects the performance of deep neural networks, traditional fixed and adaptive activation functions have low stability across different tasks or networks, while the accuracy improvement of dynamic activation function becomes less with increasing network parameters. Therefore, in this paper, a novel activation function called Channel Pooled Dynamic Activation Unit (CPDAU) is proposed to improve the performance of deep neural networks on image classification tasks.The derivative of CPDAU combines trainability and smoothing and achieves superior graph classification performance across different network architectures by using a dynamic hyperfunction based on channel feature pooling inputs. Experimental results show that CPDAU outperforms ReLU on both ShuffleNetV2, a lightweight network, and ResNet-50, a deep network, and effectively reduces the classification error with respect to the state-of-the-art activation function.
Keywords: *deep neural network, image classification, activation function, dynamic hyperfunction*

Date of Submission: 03-02-2024 Date of acceptance: 14-02-2024

I. Introduction

In recent years, deep neural networks have been used in autonomous driving[1] , Intelligent Medical[2] remote sensing[3] and Internet of Things[4] and other fields, and has rapidly become one of the most popular and fastest growing research directions in the related fields of computer vision[5, 6] The activation function of deep neural network is the basic function of deep neural network. The activation function is a basic component of deep neural networks, and the network that obtains nonlinearity is able to approximate arbitrary functions to better learn and express complex patterns of data. Therefore, the activation function is an important factor that affects the performance of the neural network framework. Currently available activation functions do not perform consistently in different task scenarios, so finding activation functions that enable neural networks to better learn and generalize has become a current research hotspot.

Based on the variation of activation functions during neural network training and inference, activation functions can be broadly classified into three categories: fixed invariant activation functions, adaptive parameterized activation functions and dynamic activation functions that vary with input.

Fixed invariant activation functions are often applied to classical neural networks such as tanh and sigmoid[5] This saturation function tends to cause the gradient to vanish and cannot be adapted to neural networks with increasing depth. The linear rectification unit ReLU[7] was proposed to make the training of deep neural networks more feasible, but it tends to lead to the problem of dead neurons. The modified linear unit Leaky ReLU with leakage[8] The exponential linear unit ELU[9] , makes the output of a negative neuron a small negative value, while alleviating the problem of mean shift in the training of models using ReLU. These two nonlinear layers help the neural network to further improve its performance, outperforming ReLU in models for multiple image classification tasks. however, this approach is less efficient in parameter optimization during model training, further increasing the computational cost. Also exponential computation leads to decrease in testing speed and incompatibility with batch normalized BNs. The proportional exponential linear unit SELU in the self-normalizing neural network SNN[10] , and the sigmoid-weighted linear unit SiLU in the deep reinforcement learning network DQN[11] , are both capable of outperforming ReLU in a given network, however the complex nonlinear computations used by both methods lead to a decrease in test speed. The Gaussian error linear unit GELU, which combines the regularization idea[12] GELU, enables the generalization performance of neural networks to outperform ELU and ReLU in a variety of application scenarios; however, the accuracy of the approximation function used to compute the Gaussian distribution in practice is affected by the hyperparameters. Overall, the actual performance of a fixed invariant activation function depends heavily on the choice of hyperparameters.

In order to solve the problem of hyperparameter selection, adaptive parameterized activation function is proposed. Adaptive segmented linear activation function APL[13] The concept of adaptive parameters is introduced, and the significant difference in parameters between the activation functions of each neuron in the experimental results suggests that a fixed classical activation function may not be an optimal solution for neural networks. However, the gradients of the basis functions in APL may not be balanced, which makes the optimization of the adaptive parameters difficult. Padé Approximate Activation Unit PAU[14] The first to use rational functions for the optimal design of neural network parameters, which can approximate any existing activation function with sufficiently large polynomial order, while partitioning ameliorates the drawbacks of polynomial gradient explosion and oscillations. However, suitable initial values of hyperparameters require complex computations, and backpropagation of rational functions is computationally expensive. Segmented linear unit PWLU[15] The potential to find good activation functions is improved by well-designed flexible formulations covering a wide range of scalar functions. Easily integrated into various network architectures with microscopically adaptive parameters that are easily learned via gradients. Linear computation is efficient and practical for real-world applications. Balanced gradients and focus on bounded inputs make PWLU consistently outperform APL and PAU in different architectures. parametric linear rectifier unit PReLU[16] Parameterized exponential linear unit PELU by replacing the hyperparameters in LReLU with trainable parameters.[17] Adding two additional learnable parameters to ELU, the morphology of the negative interval of the optimization function further improves the model fitting ability. The effect of the additional parameters of the activation layer is negligible compared to the computational cost of increasing the depth of the network. Both activation functions are more flexible than LReLU and ELU and alleviate the effort of manually setting the appropriate parameters. However, the performance improvement of individual adaptive parameters is not stable and consistent across different datasets and frameworks, and the flexibility of these two activation functions is still insufficient. In this regard, inspired by the idea of biological neuron research and borrowing the method of PReLU, we have successively proposed the flexible rectified linear unit FReLU[18] and S-shaped rectified linear unit SReLU[19] The FReLU mainly learns the negative value information by adding an adaptive rectification point, and the SReLU mainly contains a three-stage linear function with four learnable parameters for flexible adaptive tuning. These two activation functions outperform ReLU in terms of expressiveness and performance with almost no additional computational cost, enabling deep networks to learn features and mappings better. Based on the existing qualities of excellent activation functions with no upper bound in the positive region and nonlinearity in the negative region, AbsLU for absolute value linear units and IpLU for inverse polynomial linear units[20] neural networks based on different depths and architectures on multiple benchmark datasets enhance the learning rate and accuracy. Although the newly proposed activation functions are robust and have high average accuracy across different networks, the training process of finding the most adapted hyperparameters can be time-consuming. From a collection of basis functions, the combined activation function CAF[21] Constructing new activation functions by linear combination, Swish[22] An automated search technique is used to discover new activation functions in an attempt to solve the problem of inconsistent performance improvement of various variants of ReLU in different tasks. Both methods outperform previous optimal activation functions within the framework of the network being tested and evaluated, but the tree-based search space and sample-based search methods make the training process very complex and inefficient, making it difficult to find specialized activation functions for each dataset or neural architecture. Therefore Hard-Swish[23, 24] uses a ReLU6-based[25] based ReLU6 to replace the sigmoid function in Swish, optimizing the computational speed of Swish while achieving a 'net accuracy-latency efficiency' increase. However, these approaches above still do not address the limitation of inconsistent performance improvement brought by a single activation function when applied to different datasets or network architectures.

Fixed or adaptive activation functions that improve performance by adding hyperparameters raise the risk of overfitting, while network architectures that dynamically change with inputs can effectively reduce this risk and facilitate model generalization. Inspired by this idea, recent research has attempted to construct dynamic activation functions that vary with input. Combining the Maxout network[26] and dynamic convolution[27] ideas, the dynamic rectified linear unit DY-ReLU[28] and the funnel-shaped activation Funnel-ReLU[29] which both have their function shapes determined by the input elements, dramatically improve the performance of lightweight networks without a significant increase in computational cost. However its performance enhancement gradually decreases in deeper networks[30] . Meta-activation function with degree of nonlinearity determined by the input meta-ACON[30] Flexible tuning of different activation function patterns on a pixel-by-pixel, channel-by-channel and layer-by-layer basis is realized. In the experiments, the channel-by-channel meta-ACON shows stable (i.e., does not vary with network depth) performance improvement compared to DY-ReLU and Funnel-ReLU. However the simple substitution of ReLU by channel-by-channel meta-ACON does not fully utilize the diversity of this activation function structure.

To summarize, we find that while improving the accuracy, the fixed and adaptive classes of activation functions increase the training cost and their performance is not stable enough in different datasets or network architectures, while the dynamic class of activation functions is more flexible and stable in different environments but the accuracy improvement becomes less with the increase of network parameters. Aiming at this problem of the current activation function, this paper tries to integrate the characteristics of multi-class activation function to construct a new activation function.The contributions made in this paper are as follows:

1. A trainable channel pooling dynamic activation unit, CPDAU, is proposed by combining two properties, the trainable derivative of PReLU and the smooth derivative of Swish.Experimental results show that the classification accuracy of CPDAU is better than that of ReLU on two network structures, namely, ShuffleNetV2 and ResNet-50, and it effectively improve the accuracy of the image classification task. In addition, CPDAU compares favorably with the recent activation functions ACONC[30] and LAU[31] can effectively reduce the classification error on different network structures compared to the recent activation functions ACONC and LAU, respectively.
2. In this paper, we propose a dynamic input-feature-based hyperfunction consisting of a combination of depth-separable convolutional layers and batch normalization, and investigate the performance comparison of using different hyperfunctions (Conv, DSC, Conv+BN, DSC+BN) in the activation function. The experimental results show that without adding batch normalization, the classification error of the ordinary convolutional hyperfunction is slightly smaller than that of the depth-separable convolutional hyperfunction. However, with the addition of batch normalization, the performance of the depth-separable convolutional hyperfunction is significantly improved, and the batch normalization plays an important role in the enhancement of its nonlinear representation.
3. In this paper, a series of pooling functions based on different statistics are designed, including mixed application mean pooling (AP), L2 paradigm pooling (L2P) and standard deviation pooling (SDP). The experimental results show that it can help the model to better understand and distinguish the feature differences between different categories, while in the practical application, it is necessary to consider the specific task, model architecture and data characteristics to choose the appropriate pooling function.

II. Related work

Smooth maximum Maxout[26] can segment linearly approximate arbitrary convex functions and is a general formula for many current activation functions. There exists a smooth approximation formula for the maximum function α-softmax[32], when the maximum max(x1,x2) is sought between two inputs, α-softmax can be reduced to S(x1,x2)=(x1-x2)-σ[β(x1-x2)]+x2, where β is the activation factor: when β & gt;0, Sβ is a smoothed approximation of the maximum; when β < 0, Sβ is a smoothed approximation of the minimum; when β → 0, Sβ converges to the average of the two inputs. The Smooth Maximum Unit (SMU) from Smooth Approximation of Maximum uses the smooth approximation function of |x| to find a generalized approximation formula for a maximum function that smoothly approximates the general Maxout family, ReLU and its variants, and well-established functions such as GELU and Swish can also be considered as special cases of SMU.

Dynamic modules Dynamic CNNs[27] use dynamic convolutional kernels, widths, or depths, and can achieve significant accuracy gains depending on the conditions of the input samples. Dynamic network implementations include learning dynamic convolutional kernels, using attention-based methods to change the network structure, and focusing on the depth of the dynamic convolutional network. In practice as the depth of the convolutional neural network increases, the memory used for nonlinear activations gradually decreases as the resolution of each layer typically decreases.MobilenetV3[23] has found that most of the benefits of swish are actually achieved by using them only in deeper layers, and therefore only using h-swish in the second half of the model.Even with these optimizations, h swish still introduces some latency costs. However, its net effect on accuracy and latency is also positive, and h-swish can be implemented as a segmented function to reduce the number of driver memory accesses and thus significantly reduce latency costs.

Lazy Neuron The sparsity of activations that commonly occurs in Transformer[33] models is that the hidden layer following the activation function outputs very few non-zero elements, with larger Transformer models (with more layers and wider MLP hidden dimensions) being sparser in terms of the percentage of non-zero entries. Top-k thresholding using small k values to achieve sparser activations can bring a range of desired properties to the model, including lower sensitivity to noisy training data, greater robustness to input corruption, and better calibration of prediction confidence. Thus boosting activation sparsity is a way to significantly reduce FLOP counts and improve Transformer efficiency.

Gated Linear UnitsA gated linear unit[34] is a neural network layer that consists of an element-by-element product of the outputs of two linear transformer layers, one of which undergoes sigmoid activation. Instead of the first linear transform and activation function in the Transformer FFN layer, a GLU or a variant thereof, which also omits the bias term, is used. All these layers have three weight matrices, while the original FFN has only two. In order to keep the number of parameters and computation constant, the number of hidden units in the FFN needs to be reduced.

Channel AttentionSENet [35] adds an attention mechanism to the channel dimension to obtain the importance of each channel of the feature map, and then uses this importance to assign a weight value to each feature, thus allowing the neural network to focus on certain feature channels. Boost the channels of the feature map that are useful for the current task and suppress the feature channels that are not very useful for the current task. Compress the 2D features (H*W) of each channel into 1 real number by global average pooling. Generate a weight value for each feature channel and construct the correlation between the channels by two fully connected layers, weighting the normalized weights obtained earlier to the features of each channel.

III. Method

Our approach is a channel-pooled dynamic activation unit based on input hyperfunctions whose shape is determined by multiple dynamic hyperfunctions based on trainable parameters and input features. These hyperfunctions are composed of a combination of deeply separable convolutional layers and batch normalization that enables custom activation behavior based on the input features. Its input is a channel distribution pooling of the input features, which can make the custom activation behavior more channel-specific.

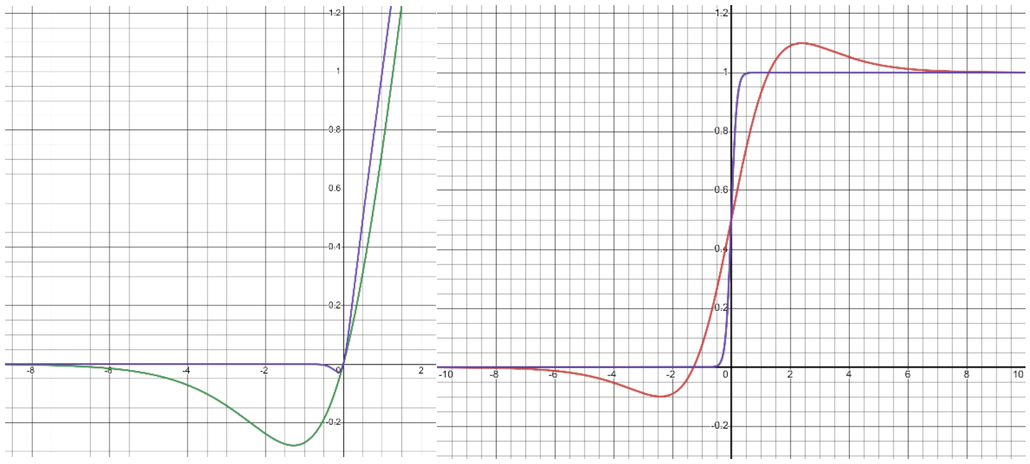
Fig. 2. The channel pooling dynamic activation unit on the left and its first order derivatives on the right. The blue line is parameterized by h1=1, h2=0, h3=10,the red and green lines are parameterized by h1=1, h2=0, h3=1.

3.1 Variant Swish activation function

Among the variants of the classical ReLU activation function, the trainable derivatives of PReLU as well as the smooth derivatives of Swish are shown to be effective improvements. By combining the properties of these two, we propose a trainable, i.e., derivative-continuous, nonlinear function whose expression and first-order derivatives are shown in Eqs. (1) and (2). where · denotes element-by-element multiplication.

$$f\left(x\right) = x \cdot h_1(\sigma(h_3 \cdot x) + h_2) \quad (1)$$

$$f'(x) = x \cdot (h_1 h_3 e^{(-h_3 x)}) / (1 + e^{(-h_3 x)})^2 + h_1 \sigma(h_3 \cdot x) + h_2 h_1 \quad (2)$$

As shown in Figure 2, the first-order derivative of the activation function has the property of smoothing the gradient of the activation function as the inputs get closer together, which can reduce the potential risk of model overfitting. A smooth activation function is also better able to cope with the noise interference present in the data, as small changes in the input do not have a very large impact on the output results. At the same time, the first-order derivatives are very steep near the zero input, which means that the gradient of the activation function varies more near the zero point, which makes it easier for the model to learn the features and improves the speed of convergence of the model. In addition, the existence of negative gradient can avoid the problem of gradient vanishing or gradient explosion to a certain extent, thus improving the training efficiency and accuracy of the model.

The parameters h1,h2,h3 in the activation function formulation can be obtained by generating a set of dynamic hyperfunctions based on the trainable parameters and input features can effectively improve the flexibility of the activation function. If the right-hand side of the element-by-element product symbol · is regarded as a whole, then the activation function formula can also be understood as an operation that multiplies the input with the corresponding element in the threshold. The threshold mechanism is a method of controlling the flow of information, and by introducing thresholds to determine which information should be passed between different layers of the network, the expressive power and predictive accuracy of the model can be improved. A static threshold performs the same operation on all input samples and is unable to accommodate the differences between different inputs. Dynamic thresholds, on the other hand, consisting of hyperfunctions of the input features, adaptively adjust the shape of the activation function according to the input data.

3.2 Dynamic hyperfunctions

For the activation function to be flexible enough to handle changing inputs and thus improve the generalization ability of the model, the dynamic hyperfunction needs to be able to adapt to input data with different features, sizes and distributions. For the activation function to be easily integrated into existing neural network architectures, the implementation of the dynamic hyperfunction needs to be as simple and general as possible. To adapt the activation function to the needs of lightweight neural networks, the additional computational overhead of the dynamic hyperfunction needs to be as small as possible. At the same time, for the activation function to help the model capture important features at different scales and improve the feature extraction capability of the model, the dynamic hyperfunction needs to extract and abstract the input information sufficiently. Therefore, the dynamic hyperfunction should be characterized by strong data adaptability, small computational overhead, sufficient feature extraction, and plug-and-play ease of implementation.

Our proposed dynamic hyperfunction h_i (i=1,2,3) based on input channel features consists of depth-separable convolutional layers[6] and batch normalization[36] combination. Its expression is shown in Eq. (3).

$$h(\mathrm{f_c}) = PC(BN(DC(BN(\mathrm{f_c})))) \quad (3)$$

Where DC is Depthwise Convolution, PC is Pointwise Convolution, BN is Batch Normalization, and f_c is the input channel feature.

Depth Separable Convolution (DSC) consists of two parts, Depthwise Convolution and Pointwise Convolution.The computation of Depthwise Convolution is very simple, it uses one convolution kernel for each channel of the input feature map, and then splices the outputs of all the kernels to get the final output. Pointwise Convolution is actually a 1×1 convolution, which allows DSC to freely change the number of output channels while performing channel fusion on the feature maps output by Depthwise Convolution. The biggest advantage of using DSC instead of normal convolution to extract input features is that it is very computationally efficient and the additional parameters added are negligible to the total network parameters. At the same time the addition of batch normalization helps to improve the nonlinear expressiveness of the hyperfunction, which in turn enables the hyperfunction to better learn more complex input features.

Since each channel represents the same class of features in CNN, by analyzing and extracting different channels of an image or feature map, we are able to capture feature information at different levels and perspectives to better understand and represent the input data. Channel features can help us distinguish between different objects, patterns, or attributes and provide more accurate and robust feature representations. In many vision tasks, such as target detection, image classification, and semantic segmentation, the effective extraction and utilization of channel features is crucial for obtaining good performance. Therefore, hyperfunctions that can fully understand and utilize the channel features can help us improve the model's ability to solve visual tasks, i.e., the inputs to the hyperfunctions should fully reflect the salient features of the channel features.

3.3 Pooling of channel distribution

Inspired by the fact that pooling function plays a crucial role in speech feature learning[37] inspired by the crucial role played by pooling functions in speech feature learning, we design a series of pooling functions based on different statistics with reference to typical pooling methods in speech feature learning. The output of the pooling function serves as the input channel features of the dynamic hyperfunction, reflecting the feature information of each channel of its input. The first statistic is the mean, which is calculated by adding all data values and dividing by the number of data. The mean value is able to measure the trend of channel feature pooling, summarize the overall information of channel features, and quickly compare feature differences between channels. The pooling function based on the mean value is shown in Equation (4).

$$AveragePooling_c(x) = \frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} x_{c,w,h} \quad (4)$$

The second statistic is the sample standard deviation, which is calculated by squaring the difference between each data value and the mean and averaging these squared values and then opening the square root. The standard deviation is a measure of how discrete the distribution of features is and reflects the outlier characteristics of the channel features. The pooling function based on the sample standard deviation is shown in equation (5).

$$StandardDeviationPooling_c(x) = \sqrt{\frac{1}{W-1} \sum_{w=1}^{W} \left(\sqrt{\frac{1}{H-1} \sum_{h=1}^{H} \left(x_{c,w,h} - \mu_h \right)^2} - \mu_w \right)^2} \quad (5)$$

where μ_h and μ_w are the mean values of the input features in the dimensions of height and width, respectively.

The third statistic is the L2 paradigm, which is calculated by adding the 2nd power of the absolute value of the data and then opening the root of the 2nd power.The L2 paradigm can reflect the distribution range of the channel features and measure the feature similarity between the channels. The pooling function based on the L2 paradigm is shown in Equation (6).

$$L2normPooling_c(x) = \sqrt[2]{\sum_{w=1}^{W} \left| \sqrt[2]{\sum_{h=1}^{H} \left| x_{c,w,h} \right|^2} \right|^2} \quad (6)$$

These pooling functions extract stable feature representations through feature degradation and aggregation operations and capture the distribution information of the input feature maps. Pooling functions can retain the main feature information and have translation invariance, which can spatially abstract the input features and extract more stable and robust feature representations. For image classification tasks, the global information extracted by the pooling function can help the model better understand and distinguish feature differences between different categories.

IV. Experiment

4.1 Experimental setup

In order to show the performance of the proposed channel pooling dynamic activation unit more objectively and to compare it with different activation functions, we conduct our experiments on the ImageNet2012 dataset, which is challenging in image classification tasks[38] We conduct experiments on the challenging ImageNet2012 dataset for the image classification task. The experimental results are obtained by replacing the lightweight neural network ShuffleNetV2[39] and the deep neural network ResNet-50[40] in the lightweight neural network ShuffleNetV2 and the deep neural network ResNet-50 to obtain the activation function. Except for changing the activation functions of the models, all models were trained for 20 epochs using the same input size of 224x224, data batch size of 32, and learning rate of 0.1, and all other training settings for the same models were kept consistent with the recommended settings in the corresponding papers for the models. The activation functions such as ReLU, PReLU, Swish, ACONC, LAU are used as baseline functions for the experiments and for the trainable activation functions they are initialized and updated by back propagation algorithm as recommended in the literature. The initial value of the DSC convolution kernel in the dynamic hyperfunction is set to 1. The default pooling inputs for the hyperfunctions h1,h2,h3 are standard deviation pooling, mean pooling and l2-paradigm pooling, respectively. For the performance comparison of different activation functions, the evaluation criterion uses the common Top-1 classification error, and the lowest error value is highlighted in **bold** in the table.

Table 1.Top-1 errors for different activation functions on ImageNet

Method	ShuffleNetV2↓	ResNet-50↓
ReLU	39.4	24.0
PReLU	38.9	23.8
Swish	38.3	23.5
ACONC	37.0	23.2
LAU	36.7	**22.5**
CPDAU (ours)	**36.1**	22.6

4.2 Comparison with other activation functions

Table 1 shows the performance of various activation functions on ShuffleNetV2 and ResNet-50. The Top-1 classification error of CPDAU is 36.1% on ShuffleNetV2 and 22.6% on ResNet-50.CPDAU outperforms ReLU on both networks.CPDAU has a 2.2% improvement over Swish on ShuffleNetV2 and compared to PReLU has a 2.8% improvement. On ResNet-50, CPDAU shows a 0.9% improvement over Swish and a 1.2% improvement over PReLU. CPDAU reduces the Top-1 classification error by 0.9% and 0.6% compared to ACONC on the two network architectures, respectively. error on ShuffleNetV2 is reduced by 0.6% and the Top-1 classification error on ResNet-50 is improved by 0.1%. By comparing the performance of different activation functions, the proposed CPDAU has a higher accuracy on the lightweight network ShuffleNetV2, while the performance on ResNet-50, which has a larger number of parameters, is slightly inferior to that of LAU.The experimental results show that our method CPDAU has an excellent performance on neural network structures of different sizes and complexities, which significantly improves the image classification performance of the network.

4.3 Selection of dynamic hyperfunctions

Table 2.Top-1 errors of different hyperfunctions on ImageNet.
Conv is normal convolution, DSC is depth separable convolution, and BN is batch normalization.

Method	ShuffleNetV2↓	ResNet-50↓
Conv	36.4	22.7
DSC	36.8	23.0
Conv+BN	36.3	22.6
DSC+BN	**36.1**	**22.6**

According to the experimental results in Table 2, the Top-1 errors of directly using Conv and DSC as hyperfunctions in ShuffleNetV2 are 36.4% and 36.8%, respectively, and the Top-1 errors of Conv hyperfunctions in ResNet-50 are 0.3% smaller than those of DSC hyperfunctions. Due to the better expressive power of ordinary convolution with more trainable parameters, the Top-1 error of directly using Conv hyperfunctions in ShuffleNetV2 and ResNet-50 will be slightly smaller than that of DSC hyperfunctions by 0.4% and 0.3%, respectively. With the addition of batch normalization, the Top-1 errors for the Conv+BN hyperfunction are 36.4 (ShuffleNetV2) and 22.6 (ResNet-50), while the Top-1 errors for the DSC+BN hyperfunction are 36.1 and 22.6 in ShuffleNetV2 and ResNet-50, respectively. in ShuffleNetV2 the Top-1 error of the DSC+BN hyperfunction is slightly lower than that of the Conv+BN hyperfunction, while in ResNet-50 the Top-1 errors are equal. Adding batch normalization leads to a slight decrease in Top-1 error in both models compared to using the DSC and Conv hyperfunctions alone. However the Top-1 error of the DSC hyperfunction is improved to a greater extent, suggesting that batch normalization is more helpful in improving the performance of DSC. This is because batch normalization can effectively solve the covariance shift problem, which in turn improves the nonlinear expressiveness of the hyperfunction. In addition, compared with the other three hyperfunctions, the DSC+BN hyperfunction is able to obtain better Top-1 errors in both ShuffleNetV2 and ResNet-50, which indicates that the DSC+BN hyperfunction has better feature extraction and representation capabilities. In conclusion, the experimental results show that DSC+BN is a hyperfunction that performs better on ImageNet2012, and it can be used to improve the accuracy of image classification and other visual tasks.

Table 3. h_1 Top-1 errors for different pooling inputs of the hyperfunction.

Method	ShuffleNetV2↓	ResNet-50↓
AP	36.8	23.0
SDP	**36.1**	**22.6**
L2P	36.4	22.6

Table 4. h_2 Top-1 errors for different pooling inputs of the hyperfunction.

Method	ShuffleNetV2↓	ResNet-50↓
AP	**36.1**	**22.6**
SDP	37.0	22.9
L2P	37.3	22.8

Table 5. h_3 Top-1 errors for different pooling inputs of the hyperfunction.

Method	ShuffleNetV2↓	ResNet-50↓
AP	37.2	22.9
SDP	36.1	22.7
L2P	**36.1**	**22.6**

Different pooling inputs (based on mean, sample standard deviation, and L2 paradigm) were used as pooling operations for the h1, h2, and h3 hyperfunctions, changing the pooling inputs for only one of the three hyperfunctions and leaving the other two hyperfunctions with default pooling inputs. The errors (e.g., Top-1 error) of the ShuffleNetV2 and ResNet-50 models on the test set of ImageNet2012 are computed separately to evaluate the effect of different pooling on the performance, and the experimental results are shown in Tables 3, 4, and 5, where AP is the mean pooling, SDP is the standard deviation pooling, and L2P is the L2-paradigm pooling. According to the data in Table 3, on ShuffleNetV2, the Top-1 errors for mean pooling (AP), L2 paradigm pooling (L2P) and standard deviation pooling (SDP) are 36.8%, 36.1% and 36.4%, respectively. On ResNet-50, the Top-1 errors for mean pooling (AP), standard deviation pooling (SDP) and L2-paradigm pooling (L2P) are 23.0%, 22.6% and 22.6%, respectively. That is, on the h1 hyperfunction in ShuffleNetV2, standard deviation pooling (SDP) performs best, outperforming mean pooling (AP) and L2 paradigm pooling (L2P). On the h1 hyperfunction in ResNet-50, standard deviation pooling (SDP) and L2-paradigm pooling (L2P) have comparable performance and outperform mean pooling (AP). According to the data in Table 4, the Top-1 errors of mean-value pooling (AP) on ShuffleNetV2 and ResNet-50 are 36.1% and 22.6%, respectively, the Top-1 errors of standard deviation pooling (SDP) on ShuffleNetV2 and ResNet-50 are 37.0% and 22.9%, respectively, and the L2 paradigm pooling (L2P) has Top-1 errors of 37.3% and 22.8% on ShuffleNetV2 and ResNet-50, respectively. Thus on the h2 hyperfunction of ShuffleNetV2, standard deviation pooling (SDP) performs best, outperforming mean pooling (AP) and L2 paradigm pooling (L2P). On the h2 hyperfunction of ResNet-50, mean value pooling (AP) has the best performance, slightly outperforming standard deviation pooling (SDP) and L2 paradigm pooling (L2P). According to the data in Table 5, on ShuffleNetV2, the Top-1 error is 37.2% for mean pooling (AP), 36.1% for standard deviation pooling (SDP), and 36.1% for L2 paradigm pooling (L2P). On ResNet-50, the Top-1 error is 22.9% for mean pooling (AP), 22.7% for standard deviation pooling (SDP), and 22.6% for L2 paradigm pooling (L2P). Standard Deviation Pooling (SDP) performs best on the h3 hyperfunction of ShuffleNetV2, slightly outperforming Mean Value Pooling (AP) and L2 Paradigm Pooling (L2P). Mean value pooling (AP) performs best on the h3 hyperfunction of ResNet-50, slightly outperforming standard deviation pooling (SDP) and L2-paradigm pooling (L2P).

Taken together, Standard Difference Pooling (SDP) achieves the best performance on h1 hyperfunctions in both ShuffleNetV2 and ResNet-50, as well as better performance on h1 and h3 hyperfunctions in ShuffleNetV2. L2 Paradigm Pooling (L2P) has a relatively outstanding performance on ResNet-50, especially on h3 hyperfunctions performs better. Average value pooling (AP) performs poorly on both h1 and h3 hyperfunctions, but performs better on h2 hyperfunctions than the other two pooling methods. Standard deviation pooling is universal and stable in models with different hyperfunction structures, while L2-paradigm pooling shows better performance in some cases. The mean pooling function extracts stable feature representations through feature degradation and aggregation operations, which capture the distribution information of the input feature maps well. A mixed application of mean pooling (AP), L2 paradigm pooling (L2P) and standard deviation pooling (SDP) can be designed to help the model better understand and distinguish feature differences between different classes. In practical applications, it is necessary to consider the selection of pooling functions and the performance performance may be affected by the specific task, model architecture and data characteristics to choose the appropriate pooling function.

V. Conclusion

The channel pooling dynamic activation unit CPDAU proposed in this paper effectively improves the performance of two neural networks of different magnitudes, ShuffleNetV2 and ResNet-50, on the task of image classification, and reduces the classification error compared to the latest activation function practice on different network structures, respectively. In addition, performance comparison experiments using different hyperfunctions (Conv, DSC, Conv+BN, DSC+BN) in ShuffleNetV2 and ResNet-50 show that the performance of deeply separable convolutional hyperfunctions can be significantly improved with the addition of batch normalization, and batch normalization plays an important role in its nonlinear representation. Finally, the comparison of the performance of three channel pooling functions (SDP, L2P, and AP) in different hyperfunctions illustrates that the pooling functions with different statistics can extract stable feature representations through feature degradation and aggregation operations. The channel pooling dynamic activation unit can provide a useful reference for feature learning in vision tasks, and the adaptability and effectiveness to different task requirements and network architectures can be further explored in future work.

Reference

[1]. GRIGORESCU S, TRASNEA B, COCIAS T, et al. A survey of deep learning techniques for autonomous driving [J]. Journal of Field Robotics, 2020, 37(3): 362-86.

[2]. ESTEVA A, CHOU K, YEUNG S, et al. Deep learning-enabled medical computer vision [J]. NPJ digital medicine, 2021, 4(1): 1-9.

[3]. MA L, LIU Y, ZHANG X, et al. Deep learning in remote sensing applications: A meta-analysis and review [J]. ISPRS journal of photogrammetry, 2019, 152: 166-77.

[4]. LI Y, ZUO Y, SONG H, et al. Deep learning in security of internet of things [J]. IEEE Internet of Things Journal, 2021.

[5]. GOODFELLOW I, BENGIO Y, COURVILLE A. Deep learning [M]. MIT press, 2016.

[6]. CHOLLET F. Xception: Deep learning with depthwise separable convolutions; proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, F, 2017 [C].

[7]. NAIR V, HINTON G E. Rectified linear units improve restricted boltzmann machines; proceedings of the Icml, F, 2010 [C].

[8]. MAAS A L, HANNUN A Y, NG A Y. Rectifier nonlinearities improve neural network acoustic models; proceedings of the 30th International Conference on Machine Learning, F, 2013 [C]. Citeseer.

[9]. CLEVERT D-A, UNTERTHINER T, HOCHREITER S. Fast and accurate deep network learning by exponential linear units (elus); proceedings of the International Conference on Learning Representations, F, 2016 [C].

[10]. KLAMBAUER G, UNTERTHINER T, MAYR A, et al. Self-normalizing neural networks; proceedings of the 31st Annual Conference on Neural Information Processing Systems, NIPS 2017, December 4, 2017 - December 9, 2017, Long Beach, CA, United states, F, 2017 [C]. Neural information processing systems foundation.

[11]. ELFWING S, UCHIBE E, DOYA K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning [J]. Neural Networks, 2018, 107: 3-11.

[12]. HENDRYCKS D, GIMPEL K. Gaussian error linear units (gelus) [J]. arXiv preprint arXiv:08415, 2016.

[13]. AGOSTINELLI F, HOFFMAN M, SADOWSKI P, et al. Learning activation functions to improve deep neural networks [Z]. ICLR

workshop. 2015

[14]. MOLINA A, SCHRAMOWSKI P, KERSTING K. Pad\'e Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks; proceedings of the International Conference on Learning Representations, F, 2020 [C].

[15]. ZHOU Y A Z, ZEZHOU AND ZHONG, ZHAO. Learning Specialized Activation Functions With the Piecewise Linear Unit; proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), F, 2021 [C].

[16]. HE K, ZHANG X, REN S, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [J]. CoRR, 2015, abs/1502.01852.

[17]. TROTTIER L, GIGUERE P, CHAIB-DRAA B. Parametric exponential linear unit for deep convolutional neural networks; proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), F, 2017 [C]. IEEE.

[18]. QIU S, XU X, CAI B. FReLU: flexible rectified linear units for improving convolutional neural networks; proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), F, 2018 [C]. IEEE.

[19]. JIN X, XU C, FENG J, et al. Deep learning with s-shaped rectified linear activation units; proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, F, 2016 [C].

[20]. ALKHOULY A A, MOHAMMED A, HEFNY H A. Improving the Performance of Deep Neural Networks Using Two Proposed Activation Functions [J]. IEEE Access, 2021, 9: 82249-71.

[21]. MANESSI F, ROZZA A. Learning combinations of activation functions; proceedings of the 2018 24th international conference on pattern recognition (ICPR), F, 2018 [C]. IEEE.

[22]. RAMACHANDRAN P, ZOPH B, LE Q V. Searching for activation functions; proceedings of the 6th International Conference on Learning Representations, ICLR 2018, April 30, 2018 - May 3, 2018, Vancouver, BC, Canada, F, 2018 [C]. International Conference on Learning Representations, ICLR.

[23]. HOWARD A, SANDLER M, CHEN B, et al. Searching for mobileNetV3; proceedings of the 17th IEEE/CVF International Conference on Computer Vision, ICCV 2019, October 27, 2019 - November 2, 2019, Seoul, Korea, Republic of, F, 2019 [C]. Institute of Electrical and Electronics Engineers Inc.

[24]. AVENASH R, VISWANATH P. Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function; proceedings of the VISIGRAPP (4: VISAPP), F, 2019 [C].

[25]. KRIZHEVSKY A, HINTON G J U M. Convolutional deep belief networks on cifar-10 [J]. 2010, 40(7): 1-9.

[26]. GOODFELLOW I, WARDE-FARLEY D, MIRZA M, et al. Maxout Networks [Z]//SANJOY D, DAVID M. Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research; PMLR. 2013: 1319--27

[27]. CHEN Y, DAI X, LIU M, et al. Dynamic convolution: Attention over convolution kernels; proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, F, 2020 [C].

[28]. CHEN Y, DAI X, LIU M, et al. Dynamic relu; proceedings of the European Conference on Computer Vision, F, 2020 [C]. Springer.

[29]. MA N, ZHANG X, SUN J. Funnel activation for visual recognition; proceedings of the European Conference on Computer Vision, F, 2020 [C]. Springer.

[30]. MA N A Z, XIANGYU AND LIU, MING AND SUN, JIAN. Activate or Not: Learning Customized Activation; proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), F, 2021 [C].

[31]. ZHOU X-M, LI L-F, ZHENG X-Z, et al. A two-parameter learnable Logmoid Activation Unit [J]. 2023.

[32]. LANGE M, ZüHLKE D, HOLZ O, et al. Applications of lp-Norms and their Smooth Approximations for Gradient Based Learning Vector Quantization; proceedings of the ESANN, F, 2014 [C]. Citeseer.

[33]. DEVLIN J, CHANG M-W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding [J]. arXiv preprint arXiv:04805, 2018.

[34]. SHAZEER N. Glu variants improve transformer [J]. arXiv preprint arXiv:05202, 2020.

[35]. HU J, SHEN L, SUN G. Squeeze-and-excitation networks; proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, F, 2018 [C].

[36]. IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift; proceedings of the International conference on machine learning, F, 2015 [C]. pmlr.

[37]. WANG S, YANG Y, QIAN Y, et al. Revisiting the statistics pooling layer in deep speaker embedding learning; proceedings of the 2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP), F, 2021 [C]. IEEE.

[38]. DENG J, DONG W, SOCHER R, et al. Imagenet: A large-scale hierarchical image database; proceedings of the 2009 IEEE conference on computer vision and pattern recognition, F, 2009 [C]. Ieee.

[39]. MA N, ZHANG X, ZHENG H-T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design; proceedings of the Proceedings of the European conference on computer vision (ECCV), F, 2018 [C].

[40]. HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition; proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, F, 2016 [C].