Wide Hidden Expansion Layer for Deep Convolutional Neural Networks

Min Wang
University of Central Florida
mwang@cs.ucf.edu

Baoyuan Liu
Amazon
baoyuan@amazon.com

Hassan Foroosh
University of Central Florida
foroosh@cs.ucf.edu

Abstract

Non-linearity is an essential factor contributing to the success of deep convolutional neural networks. Increasing the non-linearity in the network will enhance the network's learning capability, attributing to better performance. We present a novel Wide Hidden Expansion (WHE) layer that can significantly increase (by an order of magnitude) the number of activation functions in the network, with very little increase of computational complexity and memory consumption. It can be flexibly embedded with different network architectures to boost the performance of the original networks. The WHE layer is composed of a wide hidden layer, in which each channel only connects with two input channels and one output channel. Before connecting to the output channel, each intermediate channel in the WHE layer is followed by one activation function. In this manner, the number of activation functions can grow along with the number of channels in the hidden layer. We apply the WHE layer to ResNet, WideResNet, SENet, and MobileNet architectures and evaluate on ImageNet, CIFAR-100, and Tiny ImageNet dataset. On the ImageNet dataset, models with the WHE layer can achieve up to 2.01% higher Top-1 accuracy than baseline models, with less than 4% computation increase and less than 2% more parameters. On CIFAR-100 and Tiny ImageNet, when applying the WHE layer to ResNet models, it demonstrates consistent improvement in the accuracy of the networks. Applying the WHE layer to ResNet backbone of the CenterNet object detection model can also boost its performance on COCO and Pascal VOC datasets.

1. Introduction

In convolutional neural networks, the convolutional layers perform linear transformations to the feature space, while the activation function layers provide non-linear warping to the feature space. Both linear transformation and non-linear warping are essential to the network's capability of learning complex distribution.

In this work, we propose to increase the non-linearity in the networks by massively increasing the number of activation functions with a novel Wide Hidden Expansion (WHE) layer, which can be flexibly embedded into different network architectures via appending to convolutional layers.

In the WHE layer, while the number of output channels remains the same as that of input channels, there implicitly exists a large number of hidden channels connecting the input and the output. Each hidden channel is followed by one activation function. Therefore, the number of activation functions can grow along with that of hidden channels. Each hidden channel passes the weighted sum of two input channels through an activation function and then aggregates the result to one output channel. Such structured sparse connectivity guarantees that the computational complexity of this layer is bound to be a fraction of the traditional convolutional layer. On the other hand, hidden channels merely exist temporarily in the GPU cache during both the forward and backward phase of the layer. Therefore, the memory consumption (both training and inference) of WHE is aligned with the size of the input tensor rather than that of hidden channels.

The merit of the proposed WHE layer is straightforward: we can increase the number of channels as well as the number of activation functions by more than an order of magnitude, with minimal increase of consumed resources (both computation and memory). Hidden channels here cannot be considered equivalent to the channels in the traditional convolutional layer since they are very sparsely connected to the much narrower input and output. However, they do introduce substantial activation functions, which provide the non-linearity that is essential for a deep network to learn the complex distribution of visual data.

Comparing to methods that utilize channel-wise "bottleneck" structure like ResNet [10] and ResNeXt [31], the fundamental difference between the WHE layer and them is the structured sparse connections in WHE. Because each hidden channel in WHE is only connected to two input channels and one output channel, a substantial increase in hidden channels requires only minor extra computation. While in ResNeXt, each hidden channel is connected to all the input and output channels in the same group. With the same configuration on the input, hidden, and output channels, a module that is built with the "bottleneck" and grouping structure as in ResNeXt consumes more than 10 times computation than WHE. Additionally, the structured sparse connection in WHE makes it possible for the hidden channels to be computed on-the-fly and never stored on the main GPU memory, which is difficult to be achieved by ResNeXt.

We evaluate our WHE layer on the ImageNet classification dataset, CIFAR-100, and Tiny ImageNet datasets with various network architectures. It demonstrates that when adding the WHE layer to existing network architectures, our method can consistently achieve significant improvement with a marginal increase of computation and parameters. We also show that applying WHE to the backbone of the object detection network can boost performance on different datasets.

The rest of this paper is structured as follows. In section 3, we introduce the mechanism of the proposed WHE layer, analyze its computational complexity and memory consumption, and discuss the implementation details; In section 4, we explain the evaluation of our method, and discuss the experimental result; Section 5 concludes the paper.

2. Related Work

Deep Architectures

AlexNet [19] is the first deep CNN architecture applied to large scale image classification problems. ZFNet [35] adjusts the kernel and stride size of the first convolutional layer in AlexNet to achieve better accuracy. VGGNet [24] uses 3×3 convolutional layer exclusively, which is very deep and computational expensive at that time but shows the power of depth in network design. GoogleNet [26], in contrast, takes a very different approach. While having similar depth to VGGNet, its architecture is much more complex and diverse, achieving similar accuracy with much less theoretical computation. Batch Normalization [15] makes it possible to train a deep network without vanishing or exploding gradients. In ResNet [10], the residual connection reduces the length of the back- propagation path and enables effective training of network with more than one hundred layers. ResNeXt [31] improves the "cardinality" of the network with group convolutions. In [34], extremely wide networks are evaluated against deep architectures. The authors show that width can be as effective as depth regarding improving network performance. DenseNet [14] inserts residual connections between each pair of convolutional layers to improve feature propagation and reuse. In SENet [13], a squeeze-and-excitation block is introduced to aggregate the spatial information in each channel. In NASNet [37], an optimized convolutional architecture is learned on a small dataset and then applied to large scale datasets.

Novel layer designs

While most of the state-of-the-art deep architectures are designed by reconfiguring existing layers, in recent years, there is some research focusing on proposing new layers, to which this work also belongs. Several new types of activation functions [3, 5, 9, 16, 28] have been proposed to replace the original ReLU function. In NIN [20], a micro multiple layer perceptron network is used to scan the input feature map to extract better representation. In SICNet [27] and MobileNet[12], the expensive convolutional layer is replaced with a combination of a novel 2D intra-channel convolution and 1×1 convolutional layer, achieving similar representative power with much lower complexity. CNNPack [29] performs the convolution in the frequency domain with Discrete Cosine Transform (DCT), and achieves high compression by discarding low-energy frequency coefficients. In LBCNN [17], the pre-defined sparse binary 2D kernels are convolved with the input, and then linearly combined to replace the traditional convolutional layers.

Activation functions

The predominant approach for a higher non-linearity in the network is to design more powerful activation functions. The most extensively used activation function is Rectified Linear Unit (ReLu) due to its fast convergence. In ReLU, the non-linearity comes only at the linear path's change at the origin. Several ReLU variants are proposed to increase the non-linearity of the networks. PReLU [9] introduces learnable parameters for the slope of the negative linear part to increase the diversity of the activation function. ELU [5] proposes to use the exponential curve for the negative part and makes the activation function differentiable at every location. LuTU [28] proposes to learn the shape of activation function with a look-up table structure and observes the peak and valley patterns widely existing in the shapes of the learned activation function, which can be replaced by Gaussian mixture models.

Sparse connections

The layer proposed in this work can be considered as a hidden one, with structured sparse connections with the input/output layers. Sparsity in convolutional neural networks has been widely studied in recent years. In SCNN [22], the convolutional kernel is decomposed along both the spatial dimensions and the channel dimension to exploit its redundancy and then sparsified by training with l_1 penalty. In [8], the weights in the kernel are pruned, quantized, and encoded with Huffman coding to achieve high compression. In [30], structured sparsity of the convolutional kernel is learned with group Lasso regularization over the filter and channel dimensions. [11] introduces sparsity along the channel dimension by imposing Lasso regression on the channel selection weights.In [33], the authors propose an exclusive sparsity regularization, which enforces the output channels to connect to different input channels in one convolutional layer.

Saving memory

We introduce an implementation technique to reduce the memory consumption of the proposed WHE layer. There are a few related works that aim at the same purpose. In [4], sub-linear memory consumption is achieved by selectively storing a subset of the intermediate feature maps, and recover the rest by rerunning the forward operations during the backward pass. In [7], the authors avoided storing the activation function output by utilizing a reversible activation function.

3. The Proposed WHE Layer

Here, we describe the proposed novel Wide Hidden Expansion (WHE) layer that can be used as a building block embedded in a deep network. The layer's input tensor $\mathbf{X} \in \mathbb{R}^{c\times h\times w}$ and output tensor $\mathbf{Y} \in \mathbb{R}^{c\times h\times w}$ have the same dimension, where c is the number of channels, and h and w are the spatial dimensions. Considering the efficiency of implementing its sparse connections, we first split the input data along the channel dimension into groups of size g so that the weight parameters for each group can be processed in the GPU cache. The implementation details are discussed in Section 3.2.

In each group, the input data is processed along the channel dimension at each spatial location. Therefore, we only need to consider the input vector $\mathbf{x} \in \mathbb{R}^g$ and output vector $\mathbf{y} \in \mathbb{R}^g$ in one group for simpler notation.

A wide hidden layer $\mathbf{h} \in \mathbb{R}^{g\times g}$ that has g^2 neurons is created to connect the input $\mathbf{x}$ and output $\mathbf{y}$. For each $i \in [1, g]$ and $j \in [1, g]$, there is one hidden channel that connects the input $\mathbf{x}[i]$ and $\mathbf{x}[j]$ with the following linear equation:

$$\mathbf{h}(i,j) = \mathbf{A}(i,j) * \mathbf{x}(i) + \mathbf{B}(i,j) * \mathbf{x}(j) + \mathbf{C}(i,j) \quad (1)$$

where $\mathbf{A} \in \mathbb{R}^{g\times g}$, $\mathbf{B} \in \mathbb{R}^{g\times g}$ and $\mathbf{C} \in \mathbb{R}^{g\times g}$ are weight tensors that will be learned.

Each channel in the hidden layer is then passed through an activation function that provides the non-linearity. Finally, the hidden channels that correspond to the same input channel are aggregated to the corresponding output channel as follows:

$$\mathbf{y}(i) = \sum_{j=1}^{g} f(h(i,j))\mathbf{D}(i,j) \quad (2)$$

where $\mathbf{D} \in \mathbb{R}^{g\times g}$ is another weight tensor that is learned.

Figure 1 shows an example of a WHE layer with three input channels and nine hidden channels. In summary, with

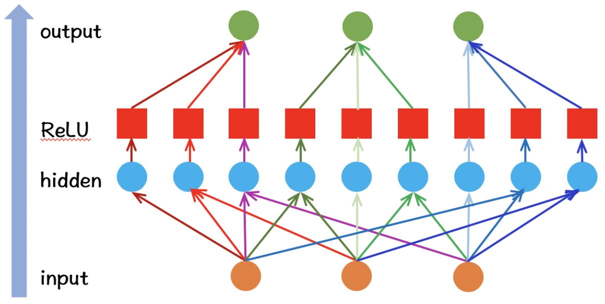


Figure 1. An example of the proposed WHE layer with three input channels. Nine hidden neurons are created to connect with one pair of input channels. The hidden neurons are then passed through an activation function layer and then aggregated to the corresponding output channels.

input in one group that has g channels, WHE layer creates g^2 hidden channels, and yields g^2 corresponding non-linear activation functions. The width of the hidden layer for all the groups is, therefore, g times the width of the input tensor. On the other hand, since each channel in the hidden layer is connected only to two input channels and one output channel, such structured sparse connections benefit the complexity of the whole layer only a fraction of that from a standard convolutional layer.The computational complexity and memory consumption of the WHE layer will be discussed in detail in Section 3.1.

3.1. Complexity, Memory Consumption and Parameters Analysis

Computational Complexity

We analyze the number of Multiply-Accumulate (MAC) operations of the proposed WHE layer. For each input vector $\mathbf{x} \in \mathbb{R}^g$ at each spatial location in one group, g^2 hidden neurons need to be computed. Each hidden neuron involves 3 MAC operations, which include 2 MAC operations in computing the neuron from the input, and 1 MAC operation in accumulating the neuron to the output. An input tensor $\mathbf{X} \in \mathbb{R}^{c\times h\times w}$ is composed of chw/g vectors of length g. Therefore, the total number of MAC operations consumed by the WHE layer is

$$3chwg \quad (3)$$

In comparison, the number of MACs of a standard convolutional layer with a 3×3 kernel and the same input dimensions is

$$9c^2hw \quad (4)$$

The ratio of their complexity is $\frac{g}{3c}$. For instance, consider a typical convolutional layer with 256 channels. If we choose $g = 16$, which means the hidden layer has 4096 channels, then the complexity of the WHE layer is only $\frac{1}{48}$ the convolutional layer.

Memory Consumption

With an input tensor $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$, the full hidden layer $\mathbf{H} \in \mathbb{R}^{c \times g \times h \times w}$ is g times the size of the input, and the output $\mathbf{Y} \in \mathbb{R}^{c \times h \times w}$ is the same size as the input. Therefore, the theoretical overall additional memory consumption of the proposed layer is

$$(g+1)chw \tag{5}$$

In a straightforward implementation for the proposed layer, if all the intermediate data ($\mathbf{H}$ and $\mathbf{Y}$ here) are stored for back-propagation, it could have much higher memory consumption than a standard convolutional layer. However, in our implementation, which will be described in Section 3.2, the hidden layer exists only temporarily and does not consume actual GPU memory. Therefore, the memory consumption of our implementation of the WHE layer is the same as a convolutional layer.

Parameters

Given a WHE layer with input tensor $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$, and group size g, each hidden channel is associated with 4 parameters ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ in Equations 1 and 2). There are gc hidden channels in total. Therefore, the total number of parameters in the proposed WHE layer is $4gc$, compared to $9c^2$ in a standard 3×3 convolutional layer with the same input size. With 256 input channels and group size $g = 16$, the number of parameters of the former is as few as $\frac{1}{36}$ of the latter.

3.2. Implementation Details

The forward and backward operations of the WHE layer are implemented with CUDA kernel functions. Each CUDA block processes the input data in one group. The weights that correspond to this group are preloaded to the shared memory of the block. Each block's shared memory resides in the cache of the Streaming Multiprocessor (SM) of the GPU, which has much higher bandwidth and much lower latency than the main GPU memory. In each iteration, an input vector $x \in \mathbb{R}^g$ that corresponds to one spatial location is processed. Thanks to the sparse connection structure of the WHE layer, the hidden layer $h \in \mathbb{R}^{g \times g}$ can be computed on-the-fly and cached in the shared memory. Then the hidden layer is aggregated to the output vector $y \in \mathbb{R}^g$, which is then stored into the output tensor that resides in the main GPU memory. Since the hidden layer is computed on-the-fly and never enters the main GPU memory, the overall memory consumption is not determined by the width of the hidden layer. This explains the memory consumption saving in Section 3.1. The only requirement that needs to be guaranteed is that the hidden layer can fit into the shared memory of the block.

The backward pass is implemented similarly, except that it involves more computation and consumption of shared memory. First, the output of the hidden layer needs to be recomputed so that the gradient over the aggregation weights $\mathbf{D}$ can be calculated. Second, the gradients over both the weights and the hidden layer also need to be cached in the shared memory, which almost doubles the shared memory consumption.

4. Experiments

We evaluate the performance of the proposed WHE layer on ImageNet [23] LSVRC 2012, CIFAR-100 [18], and Tiny-ImageNet [32] datasets on a computer with Nvidia GTX 1080 Ti GPU , dual E5-2670 CPUs, cuDNN-v7.1 and CUDA-10.0.

ImageNet classification dataset is a large scale dataset that contains 1000 categories. It contains 1.2M training images, 50K validation images, and 100K test images. The average resolution of the images is 482×415. The ImageNet dataset is the most similar to the real-world data and is the most accurate way of measuring the performance of deep CNN models.

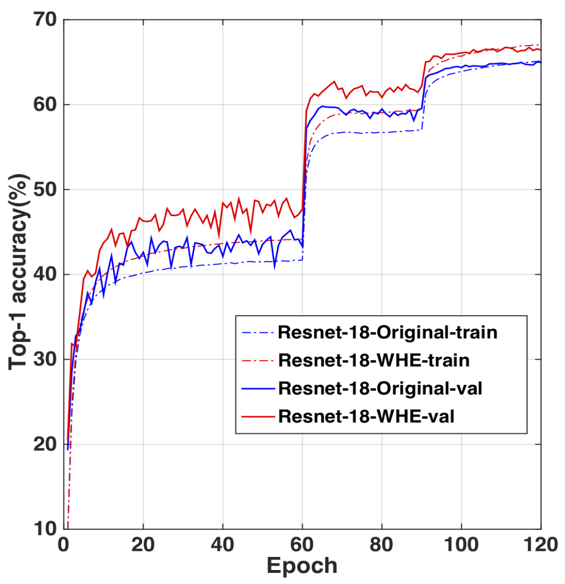
CIFAR-100 is a smaller scale image classification dataset including 100 categories of 32×32 color images. Each category has 500 training images and 100 test images. Due to its small image size, it is suitable for fast prototyping and evaluation.

Tiny ImageNet is a subset of the ImageNet dataset. It has 200 categories. Each category has 500 training images, 50 validation images, and 50 test images. The spatial dimensions of the images are 64×64.

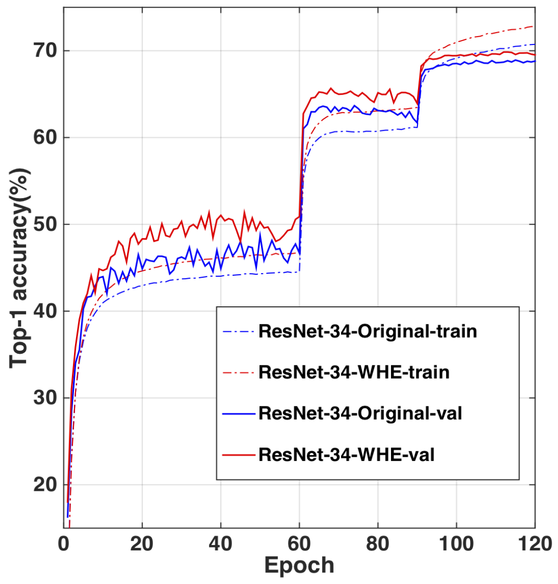
Experimental Settings

We train and evaluate our networks on the PyTorch [1] framework and follow the examples in [2]. We follow the standard data augmentation procedures that randomly flip horizontally and crop each training image. All the models are trained with the "step" learning rate policy that divides the learning rate by 10 after a certain number of epochs. For experiments on ImageNet, Top-1 and Top-5 accuracies with center cropping are reported. For experiments on CIFAR-100 and Tiny-ImageNet, the averages of Top-1 accuracies on 5 runs are reported.

WHE block To embed our WHE layer into different network architectures with minimal modification on original



(a) ResNet-18



(b) ResNet-34

Figure 2. Training curves of ResNet-18 and ResNet-34 models on ImageNet with input image size 150 × 150. The Top-1 accuracies are plotted. Training accuracies are plotted with dashed lines and validation accuracies are plotted with solid lines. Blue curves represent the original ResNet models and red curves represent models with our WHE layer. With the WHE layer, both the training and validation accuracies outperform the original model.

Model	FLOPs	Param	Top-1	Top-5
ResNet18	1800M	11.7M	69.37%	88.96%
ResNet18+WHE	1872M	11.9M	**71.08%(+1.71%)**	**89.93%**
SENet18	1800M	11.7M	70.03%	89.44%
SENet18 + WHE	1872M	11.9M	**71.15%(+1.12%)**	**90.14%**
MobileNet	596M	4.2M	70.6%	-
MobileNet + WHE	640M	4.3M	**71.60% (+1.0%)**	**90.13%**

Table 1. Comparing Top-1 & Top-5 accuracies on ResNet-18, SENet-18 , and MobileNet models on ImageNet with the input image size 224 × 224. With the WHE layer, the Top-1 accuracy on these models are improved by $\mathbf{1.71}\%$,$\mathbf{1.12}\%$ and $\mathbf{1.0}\%$, respectively, with only a slight increase in complexity and parameters.

network structures, we designed a WHE block that contains one WHE layer, one batch normalization layer, and one residual connection.

4.1. ImageNet

We evaluate the performance of the WHE layer based on the ResNet-18, ResNet-34, WideResNet, SENet-18, and MobileNet. One WHE block follows one convolution layer in the original networks.

The networks are trained with two different input image sizes: 150 × 150 cropped from images with shorter edge equal to 164, and 224 × 224 cropped from images with shorter edge equal to 256. We train the networks for 120 epochs. The learning rate is set to 0.1 in the first 60 epochs and divided by 10 every following 30 epochs. The weight decay is set to $1e^{-4}$ for the convolutional layer and the batch normalization layer. For the WHE layer, we apply an adjusted weight decay policy that sets the weight decay to $1e^{-6}$ in the first 80 epochs, $1e^{-5}$ in the subsequent 30 epochs, and $1e^{-4}$ in the last 10 epochs. The group size g for the WHE layer is set to 16. The parameters of the WHE layer are initialized with uniform distribution in [-0.1, 0.1] (In the experiments, adjusting the range from 0.01 to 0.1 resulted in the marginal improvement in accuracy by 0.2%).

Table 2 shows the results of our method compared with the original ResNet , WideResNet , SENet-18 models on 150×150 input image sizes. On the ResNet-18 and ResNet-34 models, adding the WHE layer significantly increases the top-1 accuracy by $\mathbf{2.01}\%$ and $\mathbf{1.25}\%$ respectively, with only a small increase in computational complexity (<**4%**) and in number of parameters (<**2%**) . The WideResNet models that we compare with are WideResNet-18_1.25 and WideResNet-18_1.5, which are widened versions of ResNet-18 that have 1.25× and 1.5× width, respectively.

Model	Depth	FLOPs	Param	Top-1	Top-5
ResNet-18 Original	18	790M	11.7M	64.96%	86.19%
ResNet-18 + WHE	18	**820M**	**11.9M**	**66.97%(+2.01%)**	**87.48%(+1.29%)**
ResNet-34 Original	34	1577M	21.8M	68.84%	88.90%
ResNet-34 + WHE	34	**1632M**	**22.3M**	**70.09%(+1.25%)**	**89.43%(+0.53%)**
WideResNet-18 Original	18_1.25	1200M	17.4M	67.27%	87.79%
WideResNet-18 + WHE	18_1.25	**1238M**	**17.7M**	**68.47%(+1.20%)**	**88.35%(+0.56%)**
WideResNet-18 Original	18_1.5	1690M	25.1M	68.62%	88.46%
SENet-18 Original	18	790M	11.7M	65. 64%	86.51%
SENet-18 + WHE	18	820M	11.9M	**67.05% (+1.41%)**	**87.54% (+1.03%)**

Table 2. Top-1 & Top-5 accuracies, computation complexity (measured with number of FLOPs), and number of parameters of our methods compared with the original ResNet, Wide-ResNet, and SENet models on 150×150 input image size. On ResNet-18, ResNet-34, WideResNet-18, and SENet-18 models, adding our WHE layer to the networks achieves significant improvements on accuracy by **2.01%**, **1.25%**, **1.20%**, and **1.41%** respectively, with <**4%** computational complexity increase and <**2%** more parameters. Also, WideResNet-18_1.25 with WHE layer can achieve an accuracy comparable to WideResNet-18_1.5 model, which has 40% more computational complexity and more parameters than the original WideResNet-18_1.25 .

Model	FLOPs	Param	Top-1	Top-5
ResNet18 Origin	790M	11.7M	64.96%	86.19%
ResNet18 + DepthWise	796M	11.8M	65.12% (+0.16%)	86.14%
ResNet18 + WHE	820M	11.9M	**66.97% (+2.01%)**	**87.48% (+1.29%)**

Table 3. Same depth comparison: Unlike our WHE layer, when low-cost DepthWise convolutional layers are deployed as alternative to the WHE layer, the network performance is almost not improved.

With only a slight increase in complexity ($\mathbf{3.2}\%$) and parameters ($\mathbf{1.7}\%$), WideResNet-18_1.25 with WHE layer can achieve $\mathbf{1.20}\%$ higher Top-1 accuracy than the original WideResNet-18_1.25 model, and matches the accuracy of WideResNet-18_1.5, which has 40% higher complexity and parameters than the original WideResNet-18_1.25 . When compared with the SENet-18 model, with the WHE layer, we can boost the model by $\mathbf{1.41}\%$ higher top-1 accuracy with a slight complexity increase.

Table 1 shows the comparison results based on ResNet-18, SENet-18 ,and MobileNet models with 224×224 input image size. Adding the WHE layer improves the Top-1 accuracies of ResNet-18, SENet-18 , and MobileNet by $\mathbf{1.71}\%$, $\mathbf{1.12}\%$ and $\mathbf{1.0}\%$, respectively, with a slight increase in computational complexity and parameters.

Figure 2 shows their training curves on both the training and validation sets.

4.2. Same Depth comparison

To verify that the performance increase from WHE layer is attributed to the non-linearity increase in the network rather than the depth increase (appending WHE after convolutional layer), we replace the WHE layers with the same number of low-cost DepthWise convolutional layers that are mainly used in the efficient SICNet [27]and MobileNet [12], making a head-to-head comparison based on ResNet-18 model with image size 150×150. Table 3 shows the comparison result. The improvement from only adding DepthWise convolutional layers is trivial, with 0.1% marginally better performance. In contrast, with the WHE layer, we can boost network performance by 2.01% higher accuracy.

Model	Methods	Top-1 Accuracy increase ↑
ResNet18	ReLU	0
	ELU	-2.24%
	PReLU	+0.18%
	APL	+0.27%
	WHE	**+1.71%**

Table 4. Comparison with other non-linearity increasing methods with ResNet-18 model (Input resolution: 224×224) on ImageNet dataset.

4.3. Comparison with other non-linearity increasing methods

To compare our WHE layer with activation function based methods like PReLU, APL , and ELU, we consider their impacts on the network's performance based on the ResNet-18 model (image size: 224×224) with ImageNet dataset. Models with different activation functions are trained under the same setting as our WHE layer. Table 4 shows the comparison result. The activation function based methods improve the performance of ResNet-18 by

Model	Depth	Param		Accuracy					
				CIFAR-100			Tiny-ImageNet		
		Original	Ours	Original	with dp	Ours	Original	with dp	Ours
ResNet	20	0.27M	0.28M	67.63%	68.42%	**69.17%**	48.15%	49.63%	**50.29%**
	32	0.46M	0.47M	69.49%	70.29%	**70.74%**	49.86%	51.66%	**52.61%**
	44	0.66M	0.68M	70.23%	71.02%	**72.05%**	51.24%	52.73%	**53.23%**
	56	0.85M	0.88M	70.71%	71.83%	**72.41%**	51.35%	53.65%	**54.17%**
	110	1.7M	1.76M	71.73%	73.10%	**73.57%**	52.60%	54.37%	**54.90%**

Table 5. Top-1 accuracies and number of parameters of our method compared with original ResNet models on CIFAR-100 and Tiny ImageNet datasets. Since Dropout regularization is adopted in our method with the WHE layer, we also list the accuracies of original ResNet models with Dropout for fair comparison. While adding Dropout boost the accuracies of the ResNet models, our models consistently demonstrate further improvement on all the models for both datasets with a negligible parameters increase.

Model	Top-1 accuracy	Forward Time	Training Memory
Original	69.37%	8.1 ms	1061 MB
WHE	**71.08%(+1.71%)**	8.4 ms	1105 MB

Table 6. Comparing running time and memory consumption based on ResNet-18 for batch size 8.

less than 0.3%, while our WHE layer boosts the accuracy of ResNet-18 by 1.71%.

4.4. Running time and memory consumption

WHE layer is implemented with efficient low level CUDA kernel, as described in Section 3.2. In Table 6, we evaluate the forward running time and training memory consumption based on ResNet-18 model on 1 GPU.

4.5. CIFAR-100 & Tiny ImageNet

We evaluate five different ResNet models on CIFAR-100 and Tiny ImageNet, with depths equal to 20, 32, 44, 56, and 110. Each model is trained for 150 epochs. The learning rate is set to 0.1 for the first 80 epochs and divided by 10 every following 40 epochs. On smaller scale datasets like CIFAR-100 and Tiny ImageNet, deep network models are more prone to the problem of overfitting. To reduce the impact of overfitting, we add dropout regularization [25] between the two convolutional layers in each basic residual block. Since the ResNet models on CIFAR-100 and Tiny ImageNet have much fewer channels than the ones on ImageNet, we also reduce the group size of the WHE layer to 4.

Table 5 shows the Top-1 accuracies of our method compared with the original ResNet models. To make a fair comparison, we also list the performance of the original ResNet models with dropout added (same dropout rate and locations in the residual block as ours). In all of the five network architectures, our method shows consistent improvement over the original ResNet models with dropout on both CIFAR-100 and Tiny ImageNet datasets.

Model	Group size	Param	Top-1	Top-5
Original	-	11.7M	64.96%	86.19%
Ours	4	11.7M	65.38%	86.35%
	8	11.8M	66.28%	87.12%
	16	11.9M	66.97%	87.48%

Table 7. Comparing performance with different group size in WHE layers based on ResNet-18 model on ImageNet (150×150). Accuracy increases consistently with the group size, which determines the width of the hidden layers.

Model	Weight Decay	Top-1	Top-5
Original	-	64.96%	86.19%
Ours	Fixed ($1e^{-4}$)	65.87%	86.81%
	Fixed ($1e^{-6}$)	66.61%	87.32%
	Adjusted	**66.97%**	**87.48%**

Table 8. Performance of different weight decay policies for WHE layers based on ResNet-18 model on ImageNet with 150×150 image size. The weight decay for other layers is set to $1e^{-4}$ as in the original model. Compared with using the same weight decay as other layers, reducing the weight decay of WHE layers to $1e^{-6}$ further improves Top-1 accuracy by 0.74%. Adjusting the weight decay during the training process further improves 0.36%.

4.6. Ablation study

Group Size

In the proposed WHE layer, the input is first split into groups. Inside each group with size g, the width of the hidden layer is g^2. Therefore, the group size determines the width of the whole hidden layer. In Table 7, we analyze the influence of different group sizes on the performance of the models with our WHE layer. When increasing the group size from 4 to 16, a consistent accuracy improvement ($\mathbf{0.42}\%$, $\mathbf{1.32}\%$ and $\mathbf{2.01}\%$) is observed. This result further justifies the benefits of increasing implicit width with our WHE layer.

Input	Model	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv
512	Original	74.9	76.1	84.1	**75.9**	64.9	54.7	81.9	82.2	85.1	57.9	83.1	70.9	81.9	84.8	84.7	78.3	49.5	75.7	70.3	82.8	72.8
	WHE	**76.2**	**76.8**	**84.6**	75.7	**66.1**	**56.4**	**83.0**	**83.4**	**85.2**	**61.6**	**82.6**	**72.7**	**85.1**	**86.1**	**84.9**	**78.8**	**50.2**	**79.3**	**73.3**	**83.2**	**74.9**
384	Original	71.6	73.0	**79.6**	**69.7**	60.8	44.3	78.5	78.2	**86.1**	52.3	79.7	72.6	82.2	81.6	**83.1**	74.3	43.7	71.4	**72.5**	78.3	68.9
	WHE	**72.8**	**73.4**	79.0	69.6	**62.8**	**46.7**	**80.2**	**78.7**	85.2	**55.4**	**81.1**	**73.2**	**84.5**	**84.5**	81.5	**76.6**	**48.6**	**73.9**	71.5	**80.1**	**69.5**

Table 9. Comparison on Pascal VOC 2007 test set for two different input resolutions.

Input	Backbone	AP	AP_{50}	AP_{75}
512	ResNet18	27.7	44.3	29.1
	ResNet18+WHE	**28.9**	**46.1**	**29.8**

Table 10. Comparison on COCO validation dataset for input size of 512×512.

Weight Decay for WHE layers

We studied the impact of different weight decay policies on the WHE layer with group size 16 and found that different weight decays have a noticeable influence on the scale of the parameters in WHE layer. If the same weight decay ($1e^{-4}$) is applied over all layers, a large portion of the parameters in the WHE layers turn to zero. Considering this, we make the weight decay for WHE layers different from other layers. On the other hand, setting the weight decay to a smaller value will introduce slight overfitting , and the accuracy increases less than expected when the learning rate is reduced. Therefore, we adopt an adjustable weight decay policy in our experiments. We set the weight decay to a small value ($1e^{-6}$) in the early phase of the training process, and increase it to ($1e^{-5}$) later. Table 8 compares the accuracy of using such an adjusted weight decay against the fixed one. It demonstrates that the adjustable weight decay can introduce higher performance increase ($+\mathbf{2.01}\%$ Top-1) than the fixed one (+$\mathbf{1.65}\%$ Top-1).

4.7. Object Detection

We evaluate the performance of WHE on the object detection problem by applying it to CenterNet [36], the state-of-the-art one-stage object detection CNN model. We use the ResNet-18 backbone model as a baseline and embed WHE as was described in the ImageNet experiment to measure its improvement. We follow the same training strategy as the original CenterNet paper and source code. Mean Average Precision (mAP) is evaluated on MS COCO [21] and Pascal VOC [6] datasets. We test with 512×512 resolution on COCO, and both 512×512 and 384×384 on Pascal VOC.

On COCO, we list the comparison results with the original model on mean average precision over different IoU thresholds (AP, for all the thresholds, AP_{50}, for IoU threshold 0.5, AP_{75} for IoU threshold 0.75). On the Pascal VOC dataset, we report the average precision for each category and the mean average precision over all the categories.

Table 9 shows that with our proposed WHE layer, the average precisions of CenterNet on most categories of the Pascal VOC dataset are significantly higher than the original model.

Table 10 shows that when applying the WHE layer on the backbone of ResNet-18, CenterNet can achieve more than 1% better mAP on different IoU thresholds on the COCO dataset.

5. Conclusion

In this work, we introduce a novel Wide Hidden Expansion (WHE) layer for deep convolutional neural networks. The WHE layer is composed of a wide hidden layer, which has much larger number of channels than the input and output. Each channel in the hidden layer is only connected to two input channels and one output channel. WHE layers significantly increase the implicit width of the network with a negligible increase of computational complexity and memory consumption. With extensive experiments, we show that adding WHE layers to existing network architectures consistently boosts the accuracy on different datasets and network architectures for both image classification and object detection.

References

[1] `https://pytorch.org`.

[2] `https://github.com/pytorch/examples/tree/master/imagenet`.

[3] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[4] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[7] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, edi-

tors, *Advances in Neural Information Processing Systems 30*, pages 2214–2224. Curran Associates, Inc., 2017.

[8] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, 2017.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[13] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7, 2017.

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[16] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan. Deep learning with s-shaped rectified linear activation units. *arXiv preprint arXiv:1512.07030*, 2015.

[17] F. Juefei-Xu, V. N. Boddeti, and M. Savvides. Local binary convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, volume 1. IEEE, 2017.

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[22] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[27] M. Wang, B. Liu, and H. Foroosh. Design of efficient convolutional layers using single intra-channel convolution, topological subdivisioning and spatial bottleneck structure. *arXiv preprint arXiv:1608.04337*, 2016.

[28] M. Wang, B. Liu, and H. Foroosh. Look-up table unit activation function for deep convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1225–1233. IEEE, 2018.

[29] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu. Cnnpack: Packing convolutional neural networks in the frequency domain. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 253–261. Curran Associates, Inc., 2016.

[30] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[31] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.

[32] L. Yao and J. Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2015.

[33] J. Yoon and S. J. Hwang. Combined group and exclusive sparsity for deep neural networks. In *International Conference on Machine Learning*, pages 3958–3966, 2017.

[34] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[36] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[37] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.