LST-Net: Learning a Convolutional Neural Network with a Learnable Sparse Transform

Lida Li[1[0000−0001−9386−194X]⋆], Kun Wang[2,1[0000−0001−5954−8036]⋆], Shuai Li[1,3[0000−0003−0760−5267]], Xiangchu Feng[2[0000−0002−3463−2060]], and Lei Zhang[1,3[0000−0002−2078−4215]⋆⋆]

[1] Dept. of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[2] School of Mathematics and Statistics, Xidian University
[3] DAMO Academy, Alibaba Group
{cslli, csshuaili, cslzhang}@comp.polyu.edu.hk, kwang96@stu.xidian.edu.cn, xcfeng@mail.xidian.edu.cn

Abstract. The 2D convolutional (Conv2d) layer is the fundamental element to a deep convolutional neural network (CNN). Despite the great success of CNN, the conventional Conv2d is still limited in effectively reducing the spatial and channel-wise redundancy of features. In this paper, we propose to mitigate this issue by learning a CNN with a learnable sparse transform (LST), which converts the input features into a more compact and sparser domain so that the spatial and channel-wise redundancy can be more effectively reduced. The proposed LST can be efficiently implemented with existing CNN modules, such as point-wise and depth-wise separable convolutions, and it is portable to existing CNN architectures for seamless training and inference. We further present a hybrid soft thresholding and ReLU (ST-ReLU) activation scheme, making the trained network, namely LST-Net, more robust to image corruptions at the inference stage. Extensive experiments on CIFAR-10/100, ImageNet, ImageNet-C and Places365-Standard datasets validated that the proposed LST-Net can obtain even higher accuracy than its counterpart networks with fewer parameters and less overhead.

Keywords: CNN · network architecture · learnable sparse transform

1 Introduction

The past decade has witnessed a great success of deep convolutional neural netowrk (CNN) in various computer vision problems, such as visual object recognition [34,14], object detection [44,43,35], face recognition [25,30], scene understanding [56,64], etc. The 2D convolutional (Conv2d) layer [34] is one of the key elements in a CNN to extract powerful features from the input image. Despite the great success of CNN, the conventional Conv2d is limited in effectively

⋆ The first two authors contribute equally in this work.
⋆⋆ Corresponding author. This work is supported by HK RGC General Research Fund (PolyU 152216/18E).

reducing the spatial and channel-wise redundancy of features. When image features are propagated through Conv2d, it usually requires a large number of kernels to model the data and hence introduces exaggerated parameters and overhead. Meanwhile, Conv2d simply sums up all convolutional responses along the channel dimension regarding to the same kernel and takes little advantage of inter-channel cues [24,9], which is less effective.

A lot of efforts have been devoted to improving the performance of Conv2d. Recent works can be roughly categorized into two categories. The first category of works aim to enhance what a Conv2d layer sees in the spatial domain. For representative works in this category, dilated convolution [58] effectively expands its receptive field by applying predefined gaps, while deformable convolutional networks [8,65] improve the performance of Conv2d by learning internal parameters to model geometric transformation or variations so as to adaptively focus on some more important areas. Though these methods make better use of spatial information, they fail to take advantage of the channel-wise cues. The second category of works strengthen the performance of Conv2d by combining both spatial and channel-wise attentions. Representative works in this category can be found in [24,54,16,4]. For example, squeeze-and-excitation networks (SENet) [24] re-weights the features along the channel dimension using an efficient squeeze-and-excitation block. Usually, these works rely on an extra network path to adjust spatial and channel-wise attentions after the conventional Conv2d is computed. The redundancy of conventional Conv2d remains but it requires additional network parameters and overhead. It is interesting to investigate whether we can develop a new convolutional module, which can better describe the local features, reduce the spatial and channel-wise feature redundancies, and reduce the parameters and overhead while keeping the accuracy unchanged or even improved.

We propose to mitigate these issues by learning a CNN with a learnable sparse transform (LST). We are motivated by the classical harmonic analysis works such as discrete cosine transform (DCT) [52] and discrete wavelet transform (DWT) [21,45,5], which can convert the given image into a more compact and sparse domain to reduce the spatial and channel redundancy of features. In DCT and DWT, the sparser transforms are manually pre-designed, while in our proposed LST, the sparse transform is learned from training data together with the process of CNN training. The proposed LST learning can be efficiently implemented with existing CNN modules, such as point-wise convolutions [36] (PWConvs) and depth-wise separable convolutions [23] (DWConvs). This makes LST compatible with existing CNN architectures for seamless training and inference without additional operations.

The proposed LST promotes sparser features. In light of the sparsity priors [50,2,3], we further present a hybrid soft thresholding [13] and ReLU [40] (ST-ReLU) activation scheme. Compared with the standard ReLU, the ST-ReLU activation can suppress the noise and trivial features in the learning process, making the trained network more robust to image corruptions, such as noise, blur, digital compression, etc. Overall, the proposed LST module can be applied to existing state-of-the-art network architectures such as ResNet and VGGNet. The obtained new network, namely LST-Net, achieves more robust and accurate performance with fewer parameters and less overhead. Our major contributions are summarized as follows.

- A novel learnable sparse transform based Conv2d module is developed, which can be efficiently implemented and seamlessly integrated into existing CNN learning process, producing sparser features and improving the effectiveness of learned CNN models.
- A new activation function is presented by properly combining soft-thresholding and ReLU operations, which endows the proposed LST-Net better robustness to image trivial features and corruptions.

2 Related Work

2.1 Network bottleneck

To save parameters and overhead of Conv2d layers, group convolution [34] (GConv) and PWConv [36] are popularly employed in the design of bottlenecks. PWConv employs a 1×1 window, performing a linear combination of the input from all channels. It is often used to align a set of feature maps with different number of channels [49]. GConv assumes that the input features can be decomposed into several groups along the channel dimension, where features from different groups are independent. A successful application of GConv is ResNeXt [57]. DWConv [23] is a special case of GConv when there is only one input channel per group. It is widely used to build lightweight models for mobile devices, such as MobileNet [23,47], ShuffleNet [37,62], etc.

Xie *et al*. [57] improved ResNet bottleneck [19] by substituting the conventional 3×3 Conv2d in the middle with a GConv of slightly more channels. One problem of this method is how to set the group number. A larger number of groups can easily cause loss of inter-channel cues while a smaller number of groups can hardly reduce redundancy of Conv2d. Recently, Res2Net [17] was developed by fusing the group with the intermediate results obtained from the latest group in a recursive manner. Though Res2Net demonstrates higher accuracy, it actually sacrifices parallel execution on devices such as GPUs. In this paper, we naturally incorporate DWConvs and PWConvs to facilitate transforms in spatial and channel-wise fields.

2.2 Learning space

The conventional Conv2d layer is less effective in reducing the spatial and channel-wise feature redundancies because each Conv2d kernel interacts with input features locating in a local grid of limited size and cannot take features outside the grid into consideration. To mitigate this issue, dilated convolution [58] applies predefined gaps to enlarge spatial receptive field of Conv2d. Deformable convolutional networks [8,65] learn to adaptively focus on some more important areas by modelling geometric transformation or variations with internal parameters; however, they fail to further consider the channel-wise cues of features and require sophisticated implementation skills. SENet [24] and its variants [54,16,4] focus on designing lightweight network paths to fuse channel-wise and spatial features to improve the attention of the conventional Conv2d. Though these methods is effective to boost accuracy, they remain inefficient as they use more parameters and require extra overhead.

To improve the performance of Conv2d layer, it's more straightforward to perform convolution in a more compact and sparser domain. The classical DCT [52] and DWT [21,45,5] transform the input image into a sparse space for manipulation and they have a wide range of successful applications [52,15,61,1,5,26]. The sparse coding [41] techniques encode the image patches as a sparse linear combination of learned atoms. However, the transformation filters used in DCT and DWT are manually designed and they are not effective enough to represent image structures, while sparse coding is computationally inefficient and is hard to be extended for deep feature extraction. In this paper, we propose to learn a sparse transformation together with the deep CNN learning so that the network can be more efficiently and effectively learned in a sparser space.

2.3 Activation function

Non-linearity introduced by the activation function is among the most critical factors to the success of a CNN model in various computer vision tasks. ReLU [40] is a pioneer and the most popular non-linear activation function in deep CNN. It is a simple yet highly effective segmented function, forcing the input negative valued features to zeros and keeping only the non-negative features. To make use the information of negative features, parametric ReLU [18], leaky ReLU [51], ELU [7] and SELU [31] are proposed to allow adaptive negative activation with learnable parameters. However, negative activation functions need to be carefully designed and they only exhibit better performance in specific applications.

One problem of ReLU and its variants is that they are not very robust to noise or other corruptions in the input image. It is well-known that by soft-thresholding the image features in some sparse domain, such as DWT domain [21,45,5] and sparse coding domain [41], the latent image features can be well recovered. In our proposed LST, we adaptively learn a sparse transform together with the CNN learning, which can make the CNN features sparser. This motivates us to develop a new activation scheme, i.e., hybrid soft-thresholding and ReLU (ST-ReLU), to better exploit the merit of sparser features. The ST-ReLU further enhances the robustness of learned CNN models to various types of corruptions.

3 Proposed Method

3.1 Learnable sparse transform (LST)

Denote by $\mathcal{I} \in \mathcal{R}^{H_{in} \times W_{in} \times C_{in}}$ the input feature and $\mathcal{O} \in \mathcal{R}^{H_{out} \times W_{out} \times C_{out}}$ the output feature of a Conv2D layer, where H_{in}/H_{out}, W_{in}/W_{out}, C_{in}/C_{out} denote the height, the width, and the channel number of the input/output feature, respectively. The sliding window Ω of the Conv2D can be parameterized by the kernel size $s_H \times s_W$ (for simplicity of expression, we omit the subscripts H and W in the remaining of this paper), number of kernels C_{out}, stride, as well as padding. We denote the k^{th} kernel by $\mathcal{K}^{(k)}$.

The Conv2d output feature is redundant in both spatial and channel dimensions. When the sliding window Ω is centered at spatial location (i,j) of $\mathcal{I}$, the output $\mathcal{O}_{i,j,k}$ by convolving $\mathcal{I}$ with kernel $\mathcal{K}^{(k)}$ is computed as

$$\mathcal{O}_{i,j,k} = \sum_{x=1}^{s}\sum_{y=1}^{s}\sum_{z=1}^{C_{in}} \Omega(\mathcal{I};i,j)_{x,y,z} \cdot \mathcal{K}_{x,y}^{(k)}, \tag{1}$$

where $\Omega(\mathcal{I};i,j)_{x,y,z}$ is the pixel at (x,y,z) of the tensor extracted from $\mathcal{I}$ by Ω, and $\mathcal{K}_{x,y}^{(k)}$ means the pixel at (x,y) of $\mathcal{K}^{(k)}$.

We have two observations from Eq. 1. First, all feature pixels in the local neighborhood at spatial location (i,j) are involved in the computation. While this is helpful to extract the high frequency features, it is redundant for extracting the low frequency features, which usually occupy most of the pixels in a feature map. Second, the subscript z does not follow $\mathcal{K}$ but only comes up with Ω. That is to say, all pixels in the same channel are equally weighted to produce $\mathcal{O}_{i,j,k}$. It has been found that the input features have strong similarities along the channel dimension [53,20]. Therefore, there exists much redundant channel-wise computations. All these motivate us to develop a learnable sparse transform (LST), with which the redundancy of conventional Conv2D can be reduced and hence a more efficient CNN can be learned.

Overview of LST. Our LST consists of three transforms: a spatial transform T_s, a channel-wise transform T_c, and a resize transform T_r. T_s and T_c strive to reduce the spatial and channel-wise redundancies by transforming the corresponding field into a more compact domain, while T_r aims resize the input to obtain the desired shape of output. T_r can be placed either before or after T_s and T_c. The LST, denoted by T_{LST}, can be implemented as

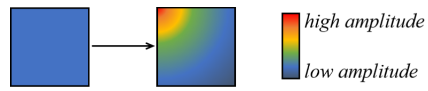
$$T_{LST} \circ \mathcal{I} = T_r \circ T_s \circ T_c \circ \mathcal{I}, \tag{2}$$

or in the form of

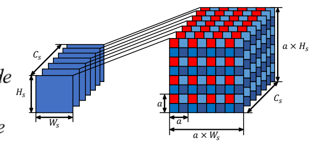
$$T_{LST} \circ \mathcal{I} = T_s \circ T_c \circ T_r \circ \mathcal{I}. \tag{3}$$

The spatial transform T_s. We propose to reduce the spatial redundancies of local features by using a learnable spatial transform T_s with associated weights $\mathcal{W}_s \in \mathcal{R}^{a^2\times 1\times s\times s}$ (dimensions are organized in PyTorch [42] style). Inspired by the success of classical 2D-DCT [39], which decomposes the image local region into different frequency bands by using sequential column and row transforms, we can implement T_s by applying column and row transforms, denoted by T_{column} and T_{row}, respectively. Mathematically, the corresponding weights W_s can be expressed as:

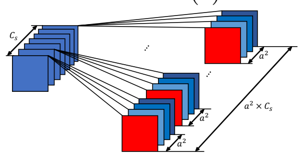
$$\mathcal{W}_s = \mathcal{W}_{column} \otimes \mathcal{W}_{row}, \tag{4}$$



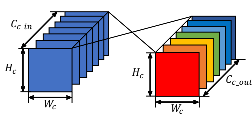
(a) 2D-DCT



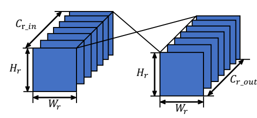
(b) A tiled spatial transform



(c) T_s



(d) T_c



(e) T_r

Fig. 1: Illustration of different transforms.

where $\otimes$ means the Kronecker product with necessary dimension insertion and removal, $\mathcal{W}_{column} \in \mathcal{R}^{a\times 1\times s\times 1}$ and $\mathcal{W}_{row} \in \mathcal{R}^{a\times 1\times 1\times s}$ are the weights of T_{column} and T_{row}, respectively, and a is a hyper parameter specifying the number of coefficients to keep.

As illustrated in Fig. 1a, a 2D-DCT transforms a local region into different frequencies. The low frequency coefficients concentrate at the top left corner and they dominate the energy (high amplitude), while many high frequency coefficients are close to zero (low amplitude) and can be neglected. Based on this fact, to save unnecessary parameters and computation, we set $1 \leq a < s$ so that the low amplitude trivial features can be excluded from calculation. Since for almost all existing CNN architectures, it is true that the kernel size $s \geq 3$, we set $a = \lceil \frac{s}{2} \rceil$ by default in this paper.

Fig. 1c depicts our implementation of T_s. One can see that the output of T_s is arranged along the channel dimension (this will ease much the implementation of our resize transform T_r). For each $s \times s$ local region, by convolving it with W_s, we obtain an a^2-dim output vector. Thus, for each input feature map, it is transformed into a number of a^2 feature maps by aggregating the a^2-dim output vectors. That is, T_s maps $\mathcal{R}^{H_{s\_in}\times W_{s\_in}\times C_s}$ to $\mathcal{R}^{H_{s\_out}\times W_{s\_out}\times (a^2\times C_s)}$, where C_s is the channel number of the input argument of T_s, and H_{s_in}/H_{s_out} and W_{s_in}/W_{s_out} are the input/output height and width, respectively. In contrast, the conventional spatial transform organizes the output in the height and width fields, instead of the channel domain. We term the conventional spatial transform as tiled spatial transform, which maps $\mathcal{R}^{H_{s\_in}\times W_{s\_in}\times C_s}$ to $\mathcal{R}^{(a\times H_{s\_in})\times(a\times W_{s\_in})\times C_s}$, as illustrated in Fig. 1b. Comparing our T_s with tiled spatial transform, we can obtain three findings.

First, T_s is simpler to implement than tiled spatial transform. In practice, we can adopt a DWConv operation to implement it. Second, T_s only affects the channel dimension, which allows us to easily use the existing efficient implementations for the resize transform T_r. (Please see the following section of resize transform for details.) In contrast, a tiled spatial transform increases both the height and width of feature maps so that T_r must be changed to deal with the enlarged spatial dimensions. Third, our T_s always holds memory continuity, making it faster in both training and inference. In contrast, a tiled spatial transform needs channel shuffle, which requires extra memory alignment.

Owe to the physical meaning of T_s (*i.e.*, to reduce feature spatial redundancy), 2D-DCT can be effectively used to initialize $\mathcal{W}_s$ with Eq. 4. This makes the training of LST converge efficiently to a good local minimum. In Section 4.2, we will show that initialization of W_s by 2D-DCT exhibits much better performance than random initialization.

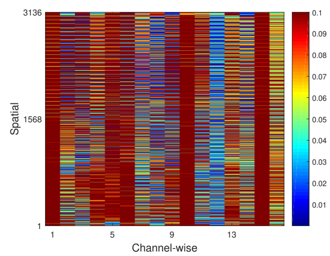
Channel-wise transform T_c. T_c is used to reduce the redundancy along channel dimension. It is a $\mathcal{R}^{H_c \times W_c \times C_{c\_in}} \rightarrow \mathcal{R}^{H_c \times W_c \times C_{c\_out}}$ mapping, where C_{c_in}/C_{c_out} is the channel number of the input/output of T_c, and H_c and W_c denote the height and width of the input, respectively. T_c encourages features to be more separable along the channel and simplifies the resize transform T_r in reweighting data.

A PWConv operation can be naturally leveraged for T_c with its associated weights $\mathcal{W}_c \in \mathcal{R}^{C_{c\_out} \times C_{c\_in} \times 1 \times 1}$. Similar to T_s, 2D-DCT can be used to initialize T_c for compact features. We fill $\mathcal{W}_c$ with the 2D-DCT basis functions shaped as $C_{c\_in} \times C_{c\_out}$ and expand its dimensions where necessary. Fig. 1d illustrates the implementation of T_c. One can see that the output of T_c is organized in order by the expected feature amplitude like 2D-DCT. It should be noted that T_c is similar to the resize transform T_r since both of them adopt PWConv for implementation. However, they are initialized in different ways.

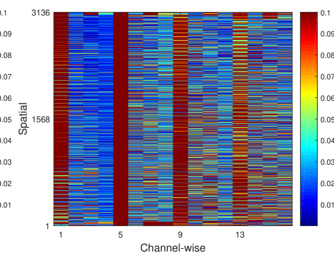
The resize transform T_r. A conventional Conv2d equally treats all the samples in the window without considering their importance. To fill this gap, the resize transform T_r is designed as a $\mathcal{R}^{H_r \times W_r \times C_{r\_in}} \rightarrow \mathcal{R}^{H_r \times W_r \times C_{r\_out}}$ mapping, where H_r/W_r is the height/width of the input, and C_{r_in} and C_{r_out} are the channel number of the input and output, respectively. T_r is learned to adaptively reweight the input features with its weights $\mathcal{W}_r \in \mathcal{R}^{C_{r\_out} \times C_{r\_in} \times 1 \times 1}$. Fig. 1e illustrates how T_r works. With the help of our design of T_s and T_c, T_r can be implemented by directly leveraging a normal PWConv operation in our paper.

Discussions. To better understand the role of LST, in Fig. 2 we visualize the learned features by the standard ResNet50 (with conventional Conv2d) and our LST-Net with a ResNet50 architecture on ImageNet [11]. (The details of the model can be found in our supplementary material.) Once trained, a validation image is randomly selected and its center crop is fed into the two models. Fig. 2 visualizes 16 channels of the features (other channels are similar) from the first bottleneck (the features after the T_s transform are visualized for our LST-Net). We clip the amplitude of the feature in the range of [0, 0.1] and stretch the features in each channel as a vector. Each column in Fig. 2 represents the vectorized features of a channel.

One can see that the output features of conventional Conv2d in ResNet50 are mixed up along the channel dimension. In contrast, the features output by LST-Net are sparser (with lower amplitude) and well-structured along channel dimension. Specifically, every $a^2 = 2^2 = 4$ channels form a unit where the four



(a)



(b)

Fig. 2: Visualization of output features obtained by (a) a conventional Conv2d layer in ResNet50 and (b) our LST-Net after T_s. One can see that the features output by LST-Net are sparser and well-structured along the channel dimension.

channel features are de-correlated into different frequency bands (please also refer to Fig. 1c). Such kind of sparser and structured features are highly suited to the successive channel-wise operations such as PWConv (used by resize transform T_r) or a sequential of global average pooling (GAP) [36] plus a dense layer.

3.2 Hybrid ReLU-ST activation scheme

By using the proposed LST introduced in Section 3.1, we are able to generate more compact and sparser features than the conventional Conv2D layers in a CNN, as illustrated in Fig. 2. It has been shown in the many works of WT [13,12] and sparse coding [41] that a soft-thresholding (ST) operation in the sparse feature domain can increase the robustness to noise and trivial features. The ST operation for the input feature x can be written as

$$y = \begin{cases} sgn(x)(|x| - \tau), & |x| \geq \tau, \\ 0, & otherwise. \end{cases} \quad (5)$$

where τ is a hyper parameter for the threshold. To exploit the merit of sparser features brought by LST, we propose a new activation scheme for our LST-Net by jointly using ST and ReLU, namely ST-ReLU.

Specifically, ST is adopted at two places in LST-Net; otherwise, ReLU is used. First, ST is inserted in the middle of T_c and T_s. It not only reduces the noises along the channel dimension but also further forces sparsity and suppresses trivial features in the spatial domain. Second, ST is used as the last activation function for an LST to allow adaptive negative activation. Unlike existing methods such as parametric ReLU [18], leaky ReLU [51], ELU [7] and SELU [31], ST is a natural selection of activation in the sparse feature domain, and it accords with the findings on spiking states of neurons in neuroscience [27,28,60,46,10].

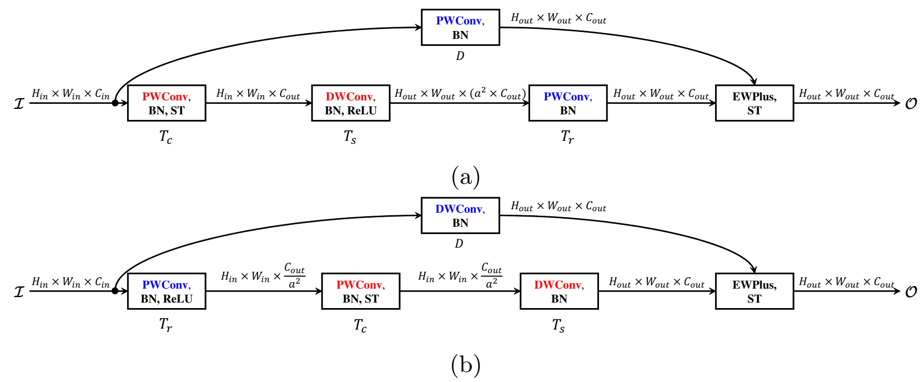


Fig. 3: Illustration of the two LST bottlenecks with downsample operators. (a): LST-I; (b): LST-II. EWPlus means element-wise plus. Red font indicates initialization with 2D-DCT while blue font suggests random initialization.

3.3 The bottleneck

We construct a novel bottleneck, namely LST bottleneck, to wrap LST and the hybrid ST-ReLU activation scheme. A shortcut path is introduced in our LST bottleneck to avoid gradient exploding or vanishing when a model goes deeper. As a result, an LST bottleneck can be written as follows when the shape of input feature $\mathcal{I}$ is the same as that of output $\mathcal{O}$:

$$\mathcal{O} = T_{LST} \circ \mathcal{I} + \mathcal{I} \tag{6}$$

If the input shape is different from the output shape, the bottleneck becomes

$$\mathcal{O} = T_{LST} \circ \mathcal{I} + D \circ \mathcal{I}, \tag{7}$$

where D is a downsample operator to adjust the shape of features. D is adopted when the stride of Ω is greater than 1 or $C_{in} \neq C_{out}$.

According to the arrangement of T_s, T_c and T_r defined in Eq. 2 and Eq. 3, we design two bottleneck structures, namely LST-I and LST-II, as illustrated in Fig. 3. One difference between LST-I and LST-II lies in how the bottleneck expands. LST-I is similar to the basic bottleneck in [19]. It first expands the number of channels by a^2 times with T_s; then, it reduces the number of channels back to C_{out} with T_r. The expansion factor of LST-I is 1. In contrast, LST-II adopts a similar ideology to the ResNet bottleneck [19]. It starts to reduce the channel number to $\frac{C_{out}}{a^2}$ with T_r and then increases it to C_{out} with T_s. Like ResNet bottleneck [19], we refer the planes (core number of channels) of an LST-II bottleneck to C_{ch_out}, *i.e.*, $\frac{C_{out}}{a^2}$. Meanwhile, the expansion factor of LST-II is determined by T_s, which equals to a^2.

Another difference between the two bottlenecks lies in the implementation of D. LST-I adopts the widely used structure, *i.e.*, a PWConv followed by a

BN. In contrast, we propose to leverage a 1×1 DWConv followed by BN for the downsample operator D of LST-II by assuming that C_{out} is divisible by C_{in}. Such an assumption usually holds in many modern architectures, e.g., VGG [48], ResNet [19], ResNeXt [57], etc. It shifts the original definition of "identity" in such cases to a group-wise mapping by expanding one channel to $\frac{C_{out}}{C_{in}}$ channels. Each input feature map only interacts with its $\frac{C_{out}}{C_{in}}$ associated output feature maps regarding to the DWConv, making it very efficient to handle hundreds or even thousands of feature maps. With LST-I or LST-II, one can easily build an LST-Net by using existing network architectures with fewer parameters and less overhead. Code is available at: https://github.com/lld533/LST-Net.

4 Experiments

4.1 Experiment setup and datasets

To evaluate our method, we build up LST-Nets by replacing conventional Conv2d operations with our proposed LST bottlenecks *w.r.t.* some widely used CNN architectures. The datasets used include CIFAR-10/100 [33] and ImageNet [11]. Besides, ImageNet-C [22] dataset is used to demonstrate the robustness of LST-Net to common image corruptions. Ablation studies are performed to discuss the initialization, the selection of parameter τ in ST-ReLU, the difference between LST-I and LST-II and comparison of ReLU-ST to other activations. Results on Places365-Standard [64] can be found in the supplementary material.

4.2 Ablation study

Initialization. As discussed in Section 3, 2D-DCT is used to initialize our spatial and channel-wise transforms W_s and W_c to reduce the feature redundancy. It is wondering whether random initialization (R.I.) can achieve similar results. We build LST-Nets of 20∼164 layers in depth using the vanilla ResNet architecture [19] to test this (LST-II bottleneck is used). We use the uniform distribution within $[-\sqrt{u}, \sqrt{u}]$ to randomly initialize W_s and W_c, where $u = \frac{1}{C_{in} \times a^2 \times s^2}$ for W_s and $u = \frac{1}{C_{in} \times a \times s}$ for W_c.

Table 1a summarizes the error rates on CIFAR-100 (similar conclusions can be obtained on CIFAR-10). One can see that 2D-DCT initialization obtains much better performance than R.I., which lags behind the former by 2.7% $\sim$ 7.0%. Besides, an LST-Net with R.I. is even worse than the baseline vanilla ResNet. This is because LST-Net will drop a certain amount of trivial frequencies after 2D-DCT initialization, while R.I. is difficult to transform the channel and spatial fields of the input feature into different frequencies with PWConv and DWConv operations, resulting in unnecessary loss of some crucial information.

The selection of parameter τ. We study the effect of parameter τ (refer to Eq. 5) on LST-Net. We built a 20-layer LST-Net in favor of ResNet architecture, and tested on CIFAR100. We search for the optimal value of τ in the range of $\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The error rates are {28.92%, 28.32%,

Table 1: Comparison (error rates, %) on CIFAR-100.

(a) Different initialization methods.

Method / Depth	20	56	110	164
ResNet [19] (R.I.)	30.88	27.62	26.23	26.07
LST-Net (R.I.)	31.12	29.92	28.95	28.94
LST-Net (2D-DCT)	**28.21**	**24.09**	**22.66**	**21.94**

(b) Different bottlenecks.

Method / Depth	20	56	110	164
ResNet [19]	30.88	27.62	26.23	26.07
LST-I	**27.64**	25.08	23.76	23.15
LST-II	28.21	**24.09**	**22.66**	**21.94**

(c) Different activation methods.

Depth	ReLU-ST	LReLU [38]	SELU [31]	ReLU [40]	PReLU [18]	ELU [7]
20	**28.21**	28.37	28.69	28.92	29.35	31.60
164	**21.94**	22.44	22.84	22.86	23.91	29.94

28.28%, 28.86%, 28.21%, 28.43%, 28.70%}, where the best result is 28.21% when $\tau = 10^{-4}$. Thus, we set $\tau = 10^{-4}$ by default in all experiments of this paper. Note that when $\tau = 0$, ST-ReLU is reduced to standard ReLU, but its error rate is larger than other values of τ. This validates that our hybrid ReLU-ST activation scheme works better than ReLU for LST-Net.

LST-I vs LST-II. We discuss the pros and cons of our proposed LST-I and LST-II bottlenecks in building up a LST-Net. For LST-I, one is free to replace a conventional Conv2d with it in many existing architectures, such as ResNet [19], AlexNet [32], VGG [48], etc. For example, a basic ResNet bottleneck can be replaced by a pair of LST-I bottlenecks as it has two Conv2d operations. For LST-II, due to the expansion factor of LST-II, parameters and overhead of the associated PWConv operation in the shortcut path are increased by a^2 times compared to LST-I. Thus, LST-II is not suitable to architectures with larger spatial size at earlier layers, such as AlexNet and VGG. LST-II will also increase the output channel number of the last bottleneck, but this issue can be easily solved with an extra PWConv operation, which is cheap compared to the entire CNN model in terms of number of parameters and computational cost. When building a deeper CNN model, such as ResNet-50 or ResNet-101, it is more suitable to use LST-II than LST-I. In Table 1b, we construct LST-Nets with LST-I and LST-II bottlenecks *w.r.t.* ResNet architecture and compare them on CIFAR-100. The vanilla ResNet is included as the baseline. Both LST-I and LST-II outperform the baseline by a large margin, while LST-II performs better than its LST-I counterpart. In the remaining experiments of this paper, if not specified, we adopt LST-II bottleneck to build ResNet models by default.

Comparison of ST-ReLU to other activations. We use a 20-layer and a 164-layer LST-Net to compare our ReLU-ST with ReLU [40], leaky ReLU (LReLU) [38], parametric ReLU (PReLU) [18], ELU [7] and SELU [31] on CIFAR-100. For comparison, we remove ST operations in LST bottleneck and replace ReLU by other activations. Table 1c presents the Top-1 error rates achieved by different activations. One can see that ReLU-ST outperforms other activations for both 20- and 164-layer LST-Nets. The gain is higher for deeper models.

Table 2: Results (error rates, %) by different networks on CIFAR-10/100.

(a) ResNet family.

Depth	Model	Param/FLOPs	C10/C100
20	ResNet [19]	0.27M/40.8M	7.7/30.9
	PreactResNet [19]	0.27M/40.8M	7.7/30.8
	ShiftResNet [55]	**0.16M/27M**	9.0/31.4
	FE-Net [6]	**0.16M/27M**	8.3/30.8
	SENet [24]	0.28M/40.8M	7.6/30.5
	CBAM [54]	0.28M/40.8M	7.3/30.3
	LST-Net	0.20M/34M	**6.7/28.2**
56	ResNet [19]	0.86M/126M	6.6/27.6
	PreactResNet [19]	0.86M/126M	6.5/27.6
	ShiftResNet [55]	**0.55M/84M**	7.3/27.9
	FE-Net [6]	**0.55M/84M**	8.3/30.8
	SENet [24]	0.87M/126M	6.4/27.5
	CBAM [54]	0.87M/126M	6.0/27.1
	LST-Net	0.59M/94M	**5.6/24.1**
110	ResNet [19]	1.73M/253M	6.6/25.2
	PreactResNet [19]	1.73M/253M	6.2/24.1
	ShiftResNet [55]	1.18M/187M	6.8/27.4
	FE-Net [6]	N.A.	N.A.
	SENet [24]	1.74M/253M	5.2/23.9
	CBAM [54]	1.74M/253M	5.1/23.5
	LST-Net	**1.17M/183M**	**5.0/22.7**

(b) WRN.

Depth	Multiplier	Model	Param/FLOPs	C10/C100
16	8	WRN [59]	10.96M/2.00G	4.80/22.03
		LST-Net	**6.03M/0.98G**	**4.70/20.88**
	10	WRN [59]	17.12M/3.12G	4.49/21.52
		LST-Net	**9.36M/1.52G**	**4.46/20.21**
22	8	WRN [59]	17.16M/2.91G	4.56/21.21
		LST-Net	**8.87M/1.40G**	**4.40/19.33**
	10	WRN [59]	26.80M/4.54G	4.44/20.75
		LST-Net	**16.99M/2.79G**	**4.31/18.57**
28	10	WRN [59]	36.48M/5.95G	4.17/20.50
		LST-Net	**22.47M/3.60G**	**4.03/18.23**
	12	WRN [59]	43.42M/8.56G	4.33/20.41
		LST-Net	**26.06M/4.03G**	**3.94/17.93**
40	4	WRN [59]	8.91M/1.41G	4.97/22.89
		LST-Net	**4.98M/0.72G**	**4.31/19.14**
	8	WRN [59]	35.75M/5.63G	4.66/19.38
		LST-Net	**19.88M/3.16G**	**3.76/18.56**

4.3 Evaluation on CIFAR-10 and CIFAR-100

We build our LST-Net models *w.r.t.* the popular architectures, including ResNet [19] and Wide Residual Networks (WRN) [59], and compare LST-Net with state-of-the-art CNNs in those families, e.g. Pre-activation ResNet [19], SENet [24], CBAM [54], and two other models, i.e., ShiftResNet [55] and FE-Net [6].

Table 2 presents the results on CIFAR-10/100. We can have the following findings. First, LST-Net achieves the lowest error rates under different network depths with almost the least number of parameters and FLOPs (very close to ShiftResNet and FE-Net). This validates its effectiveness and efficiency. LST-Net outperforms ResNet and PreactResNet while reducing over 40% parameters and 35% overhead. Compared to SENet and CBAM, LST-Net does not need extra paths while it achieves even better results. For instance, a 110-layer LST-Net improves SENet/CBAM of the same depth by 0.2%/0.1% and 1.2%/0.8% on CIFAR-10 and CIFAR-100, respectively. Besides, LST-Net outperforms both ShiftResNet and FENet by a large margin with comparable parameters and overhead. For example, a 20-layer LST-Net reduces the error rates of ShiftResNet and FE-Net by 2.3/3.2% and 1.6/2.6% on CIFAR-10/100, respectively.

Second, when we switch to wider CNN models, our bottleneck can save more parameters and computational cost because the computation of PWConv dominates an entire LST bottleneck when it is wide enough (the cost of DWConv can be neglected). We can obtain consistent performance boost of our LST-Net with the increase of width and/or depth. In contrast, the corresponding WRN architecture is less effective to improve its results with more channels and/or layers. For example, for a 28-layer WRN, the error rates will rise by 4.17% $\sim$ 4.33% on CIFAR-10 when the width multiplier is increased from 10 to 12.

Table 3: Results (error rates, %) by different networks on ImageNet.

(a) ResNet family.

Depth	Model	Param/FLOPs	Top-1/Top-5
18	ResNet [19]	11.69M/1.81G	30.24/10.92
	SENet [24]	11.78M/1.81G	29.41/10.22
	CBAM [54]	11.78M/1.82G	29.31/10.17
	LST-Net	**8.03M/1.48G**	**26.55/8.59**
34	ResNet [19]	21.79M/3.66G	26.70/8.58
	SENet [24]	21.96M/3.66G	26.13/8.35
	CBAM [54]	21.96M/3.67G	26.01/8.40
	LST-Net	**13.82M/2.56G**	**23.92/7.24**
50	ResNet [19]	25.56M/4.09G	23.85/7.13
	SENet [24]	28.09M/4.09G	23.14/6.70
	CBAM [54]	28.09M/4.10G	22.98/6.68
	LST-Net	**23.33M/4.05G**	**22.78/6.66**
101	ResNet [19]	44.55M/7.80G	22.63/6.44
	SENet [24]	49.29M/7.81G	22.35/6.19
	CBAM [54]	49.29M/7.81G	21.65/5.95
	LST-Net	**42.36M/7.75G**	**21.63/5.94**

(b) WRN.

Depth	Mulp.	Model	Param/FLOPs	Top-1/Top-5
18	1	WRN [59]	11.69M/1.81G	30.24/10.92
		LST-Net	**8.03M/1.48G**	**26.55/8.59**
	1.5	WRN [59]	25.88M/3.87G	27.06/9.00
		LST-Net	**14.40M/2.49G**	**24.44/7.51**
	2	WRN [59]	45.62M/6.70G	25.58/8.06
		LST-Net	**25.12M/4.31G**	**23.49/6.93**
	3	WRN [59]	101.78M/14.72G	24.06/7.33
		LST-Net	**55.44M/9.43G**	**22.33/6.52**
34	1	WRN [59]	21.79M/3.66G	26.70/8.58
		LST-Net	**13.82M/2.56G**	**23.92/7.24**
	1.5	WRN [59]	48.61M/8.03G	24.50/7.58
		LST-Net	**24.78M/4.41G**	**22.29/6.30**
	2	WRN [59]	86.04M/14.09G	23.39/7.00
		LST-Net	**43.44M/7.69G**	**21.44/6.11**
50	1	WRN [59]	25.56M/4.09G	23.85/7.13
		LST-Net	**23.33M/4.05G**	**22.78/6.66**
	2	WRN [59]	68.88M/11.40G	21.90/6.03
		LST-Net	**66.10M/11.09G**	**20.89/5.76**

(c) Other CNNs.

Model	Param/FLOPs	Top-1/Top-5
AlexNet [32]	61.10M/0.71G	43.45/20.91
AlexNet (BN)	61.10M/0.71G	41.93/20.02
AlexNet (GAP)	2.73M/0.66G	51.13/26.33
LST-Net (FC)	**60.30M/0.62G**	**39.32/17.40**
LST-Net (GAP)	**2.25M/0.60G**	**39.91/17.86**
VGG [48]	132.86M/7.61G	30.98/11.37
VGG (BN)	132.86M/7.61G	29.62/10.19
VGG (GAP)	9.73M/7.49G	33.40/12.20
LST-Net (FC)	**128.63M/5.89G**	**28.56/9.79**
LST-Net (GAP)	**6.63M/5.04G**	**29.23/10.26**
ShiftNet-A [55]	**4.1M**/1.4G	29.9/10.3
ShiftNet-B [55]	**1.1M**/N.A.	38.8/16.4
ShiftNet-C [55]	**0.78M**/N.A.	41.2/18.0
LST-Net (A)	4.3M/**1.2G**	**29.3/10.0**
LST-Net (B)	1.2M/389.5M	**36.9/14.8**
LST-Net (C)	0.84M/342.5M	**38.9/16.3**
MobileNet V2 [47]	**3.4M/300M**	28.1%/N.A.
LST-Net (M-V2)	**3.4M/300M**	**27.7%/9.4%**

4.4 Evaluation on ImageNet

We then evaluate LST-Net on ImageNet [11] for large-scale image classification. We construct LST-Nets regarding to the widely used network architectures, including ResNet [19], WRN [59], AlexNet [32] and VGG (with 11 layers) [48]. We also build LST-Nets *w.r.t.* ShiftNet [55] and MobileNet V2 [47].

Specifically, for ResNet or WRN architecture, we construct LST-Net using LST-II bottleneck, and for AlexNet/VGG, we build LST-Net (FC) by replacing Conv2d layers with LST-I bottlenecks. We also change the original classifier layer in AlexNet/VGG into GAP [36] plus a dense layer following [63], resulting in LST-Net (GAP). Similarly, the standard AlexNet/VGG can be modified in the same way, resulting in AlexNet (GAP)/VGG (GAP). Since BN [29] is used in our bottleneck, we further insert a BN layer after each Conv2d of AlexNet/VGG, termed as AlexNet/VGG (BN). For ShiftNet architecture, we build the LST-Nets by adjusting the stride, kernel size, number of stages, etc., according to its variants A, B, and C with different depth and width. For MobileNet V2, we build LST-Net (M-V2) by replacing Inverted Residual bottlenecks with our modified LST-I bottlenecks. Details can be found in our supplementary material.

Table 3 summarizes the results. One can see that LST-Net consistently surpasses ResNet, SENet and CBAM of the same depth with fewer parameters and less overhead. An 18-layer LST-Net even achieves lower Top-1 error rates than the standard ResNet-34 on ImageNet. Despite of different depth, increasing width of LST-Net with WRN architecture steadily increases its accuracy. Meanwhile, LST-Net saves larger proportion of parameters and overhead compared to WRN. LST-Net with AlexNet or VGG architecture is much more robust to different classifier structures than the standard AlexNet or VGG because LST-Net learns structured features, which are well suited for channel-wise operations (see our discussion in Section 3.1). Meanwhile, LST-Net (FC) can reduce Top-1/Top-5 error rates of AlexNet (BN) and VGG (BN) by 2.61%/2.62% and 1.06%/0.40%, respectively. LST-Net also shows better performance under the ShiftNet architecture. Compared with all the three variants, our LST-Net reduces the Top-1

Table 4: Comparison of robustness to common corruptions on ImageNet-C.

Network	mCE	Noise			Blur				Weather				Digital				Extra			
		Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	Saturate	Spatter	Gaus. Blur	Speckle
ResNet-18 [19]	85.29	87	89	89	88	93	90	88	88	87	81	73	81	93	81	89	72	81	86	86
SENet-18 [24]	83.97	85	86	87	87	93	88	88	85	85	79	73	82	92	84	92	71	81	86	82
CBAM-18 [54]	84.97	86	88	87	88	93	89	90	85	86	80	74	82	92	81	89	71	81	87	85
LST-Net-18 (w/o ST)	80.34	81	82	85	85	**91**	**83**	**85**	**82**	83	**75**	**68**	79	**90**	74	**85**	**66**	75	84	78
LST-Net-18 (w/ ST)	**79.89**	**80**	**81**	**83**	**84**	**91**	**83**	**85**	**82**	**82**	**75**	**68**	**78**	**90**	**73**	**85**	**66**	**74**	**83**	**76**
ResNet-50 [19]	77.01	78	80	80	79	90	81	80	80	78	69	62	75	88	76	78	62	74	78	76
SENet-50 [24]	74.47	76	77	76	77	89	79	82	75	76	70	59	75	85	71	74	58	69	76	71
CBAM-50 [54]	72.56	**69**	**71**	**71**	80	86	**77**	78	75	76	69	61	74	85	63	**70**	58	68	78	**66**
LST-Net-50 (w/o ST)	70.85	71	72	**71**	77	85	**77**	**75**	**73**	**72**	66	**58**	**70**	82	**61**	72	**56**	**65**	76	67
LST-Net-50 (w/ ST)	**70.54**	71	72	**71**	**76**	**84**	**77**	**75**	**73**	**72**	**65**	**58**	**70**	**81**	**61**	72	**56**	**65**	**75**	67

error rate of its corresponding counterpart by 0.6% ∼2.3% with similar number of parameters. LST-Net (M-V2) achieves a 72.3% Top-1 accuracy, outperforming MobileNet V2 by 0.4% using the same number of parameters and computational cost. This again validates the generality and superiority of our LST method.

4.5 Evaluation on ImageNet-C

We study the robustness of LST-Net to common corruptions in input by using the ImageNet-C dataset [22]. The mean corruption error (mCE) defined in [22] is used as our criteria. We construct LST-Net according to the ResNet architecture and compare it with the vanilla ResNet [19], SENet [24] and CBAM [54]. To examine the role of ST (please refer to Section 3.2) in improving the robustness of LST-Net, we also test LST-Net without ST in activation.

Table 4 lists the mCE and corruption errors for each type of corruption. One can see that LST-Net achieves lower mCE than its competitors of the same depth. It significantly reduces the mCE of the vanilla ResNet by 3.69% (18-layer) / 6.47% (50-layer), and also improves SENet and CBAM by at least 2.76% (18-layer) / 2.02% (50-layer). Though SENet and CBAM use extra paths which work well on clean images, the pooling operations in these paths may produce biased results in the existence of corruptions when the model is shallow. In contrast, LST does not need such extra paths and its robustness comes from the compact and sparser features. In addition, the ST operation in our ST-ReLU activation function can strengthen the robustness of LST-Net to most types of corruptions. With ST, the mCE of LST-Net-18/50 is reduced by 0.45%/0.31%.

5 Conclusion

In this paper, we proposed to train deep CNNs with a learnable sparse transform (LST), which learns to convert the input features into a more compact and sparser domain together with the CNN training process. LST can more effectively reduce the spatial and channel-wise feature redundancies than the conventional Conv2d. It can be efficiently implemented with existing CNN modules, and is portable to existing CNN architectures for seamless training and inference. We further presented a hybrid ST-ReLU activation to enhance the robustness of the learned CNN models to common types of corruptions in the input. Extensive experiments validated that the proposed LST-Net achieves even higher accuracy than its counterpart networks of the same family with lower cost.

References

1. Cai, J.F., Dong, B., Osher, S., Shen, Z.: Image restoration: Total variation, wavelet frames, and beyond. JAMS **25**(4), 1033–1089 (2012)
2. Candes, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**(2), 489–509 (2006)
3. Candes, E.J., Wakin, M.B., Boyd, S.: Enhancing sparsity by reweighted ℓ_1 minimization. J. Fourier Anal. Appl. **14**, 877–905 (2008)
4. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: GCNet: Non-local networks meet squeeze-excitation networks and beyond. arXiv preprint arXiv:1904.11492 (2019)
5. Chang, T., Kuo, C.C.: Texture analysis and classification with tree-structured wavelet transform. IEEE Trans. Image Process. **2**(4), 429–441 (1993)
6. Chen, W., Xie, D., Zhang, Y., Pu, S.: All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In: Proc. CVPR (2019)
7. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). In: Proc. ICLR (2016)
8. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proc. ICCV (2017)
9. Dai, T., Cai, J., Zhang, Y., Xia, S.T., Zhang, X.P.: Second-order attention network for single image super-resolution. In: Proc. CVPR (2019)
10. Denève, S., Alemi, A., Bourdoukan, R.: The brain as an efficient and robust adaptive learner. Neuron **94**(5), 969–977 (2017)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. CVPR. IEEE (2009)
12. Donoho, D.L.: De-noising by soft-thresholding. IEEE Trans. Inf. Theory **41**(3), 613–627 (1995)
13. Donoho, D.L., Johnstone, J.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika **81**(3), 425–455 (1994)
14. Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vis. **111**(1), 98–136 (Jan 2015)
15. Fracastoro, G., Fosson, S.M., Magli, E.: Steerable discrete cosine transform. IEEE Trans. Image Process. **26**(1), 303–314 (Jan 2017)
16. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual attention network for scene segmentation. In: Proc. CVPR (2019)
17. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2Net: A new multi-scale backbone architecture. IEEE Trans. Pattern Anal. Mach. Intell. (2020). https://doi.org/10.1109/TPAMI.2019.2938758
18. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. ICCV (2015)
19. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Proc. ECCV. Springer (2016)
20. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proc. ICCV (2017)
21. Heil, C., Walnut, D.F.: Continuous and discrete wavelet transforms. SIREV **31**(4), 628–666 (1989)
22. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: Proc. ICLR (2019)

23. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: Proc. CVPR (2017)
24. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proc. CVPR (2018)
25. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst (October 2007)
26. Huang, K., Aviyente, S.: Wavelet feature selection for image classification. IEEE Trans. Image Process. **17**(9), 1709–1720 (2008)
27. Hubel, D.H., Wiesel, T.N.: Receptive fields of single neurones in the cat's striate cortex. J. Physiol. **148**(3), 574–591 (1959)
28. Huys, R., Jirsa, V.K., Darokhan, Z., Valentiniene, S., Roland, P.E.: Visually evoked spiking evolves while spontaneous ongoing dynamics persist. Front. Syst. Neurosci. **9**, 183 (2016)
29. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. ICML (2015)
30. Kemelmacher-Shlizerman, I., Seitz, S.M., Miller, D., Brossard, E.: The megaface benchmark: 1 million faces for recognition at scale. In: Proc. CVPR (2016)
31. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Proc. NeurIPS (2017)
32. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997 (2014)
33. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
34. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. NeurIPS (2012)
35. Li, S., Yang, L., Huang, J., Hua, X.S., Zhang, L.: Dynamic anchor feature selection for single-shot object detection. In: Proc. ICCV (2019)
36. Lin, M., Chen, Q., Yan, S.: Network in network. In: Proc. ICLR (2014)
37. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proc. ECCV (2018)
38. Maas, A., Hannun, A., Ng, A.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. ICML (2013)
39. Makhoul, J.: A fast cosine transform in one and two dimensions. IEEE Trans. Acoust., Speech, Signal Process **28**(1), 27–34 (1980)
40. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proc. ICML (2010)
41. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607–609 (1996)
42. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: Proc. NeurIPS-W (2017)
43. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proc. CVPR (2016)
44. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Proc. NeurIPS (2015)
45. Rioul, O., Duhamel, P.: Fast algorithms for discrete and continuous wavelet transforms. IEEE Trans. Inform. Theory **38**(2), 569–586 (1992)
46. Roland, P.E.: Space-time dynamics of membrane currents evolve to shape excitation, spiking, and inhibition in the cortex at small and large scales. Neuron **94**(5), 934–942 (2017)

47. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proc. CVPR (2018)
48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. ICLR (2015)
49. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI (2017)
50. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996)
51. Wang, S.H., Phillips, P., Sui, Y., Liu, B., Yang, M., Cheng, H.: Classification of alzheimer's disease based on eight-layer convolutional neural network with leaky rectified linear unit and max pooling. J. Med. Syst. **42**(5), 85 (2018)
52. Watson, A.B.: Image compression using the discrete cosine transform. Mathematica Journal **4**(1), 81 (1994)
53. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Proc. NeurIPS (2016)
54. Woo, S., Park, J., Lee, J.Y., So Kweon, I.: CBAM: Convolutional block attention module. In: Proc. ECCV (2018)
55. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K.: Shift: A zero flop, zero parameter alternative to spatial convolutions. In: Proc. CVPR (2018)
56. Xiao, J., Ehinger, K.A., Hays, J., Torralba, A., Oliva, A.: Sun database: Exploring a large collection of scene categories. Int. J. Comput. Vis. **119**(1), 3–22 (2016)
57. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proc. CVPR. IEEE (2017)
58. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: Proc. ICLR (2016)
59. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proc. BMVC (2016)
60. Zerlaut, Y., Destexhe, A.: Enhanced responsiveness and low-level awareness in stochastic network states. Neuron **94**(5), 1002–1009 (2017)
61. Zhang, L., Bao, P., Wu, X.: Multiscale lmmse-based image denoising with optimal wavelet selection. IEEE Trans. Circuits Syst. Video Technol. **15**(4), 469–481 (April 2005)
62. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proc. CVPR (2018)
63. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proc. CVPR (2016)
64. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. **40**(6), 1452–1464 (2018)
65. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable ConvNets v2: More deformable, better results. In: Proc. CVPR (2019)