Neural Sparse Representation for Image Restoration

Yuchen Fan, Jiahui Yu, Yiqun Mei, Yulun Zhang*, Yun Fu*, Ding Liu**, Thomas S. Huang
University of Illinois at Urbana-Champaign, *Northeastern University, **ByteDance
yuchenf4@illinois.edu

Abstract

Inspired by the robustness and efficiency of sparse representation in sparse coding based image restoration models, we investigate the sparsity of neurons in deep networks. Our method structurally enforces sparsity constraints upon hidden neurons. The sparsity constraints are favorable for gradient-based learning algorithms and attachable to convolution layers in various networks. Sparsity in neurons enables computation saving by only operating on non-zero components without hurting accuracy. Meanwhile, our method can magnify representation dimensionality and model capacity with negligible additional computation cost. Experiments show that sparse representation is crucial in deep neural networks for multiple image restoration tasks, including image super-resolution, image denoising, and image compression artifacts removal. Code is available at [https://github.com/ychfan/nsr](https://github.com/ychfan/nsr).

1 Introduction

Sparse representation plays a critical role in image restoration problems, such as image super-resolution [1, 2, 3], denoising [4], compression artifacts removal [5], and many others [6, 7]. These tasks are inherently ill-posed, where the input signal usually has insufficient information while the output has infinitely many solutions w.r.t. the same input. Thus, it is commonly believed that sparse representation is more robust to handle the considerable diversity of solutions.

Sparse representation in sparse coding is typically high-dimensional but with limited non-zero components. Input signals are represented as sparse linear combinations of tokens from a dictionary. High dimensionality implies larger dictionary size and generally leads to better restoration accuracy, since a more massive dictionary is capable of more thoroughly sampling the underlying signal space, and thus more precisely representing any query signal. Besides, sparsity limits numbers of non-zero elements work as an essential image prior, which has been extensively investigated and exploited to make restoration robust. Sparsity also brings computational efficiency by ignoring zero parts.

Deep convolutional neural networks for image restoration extend the sparse coding based methods with repeatedly cascaded structures. The deep network based approach was firstly introduced to improve the performance in [8] and conceptually connected with previous sparse coding based methods. A simple network, with two convolutional layers bridged by a non-linear activation layer, can be interpreted as: activation denotes sparse representation; non-linearity enforces sparsity and convolutional kernels consist of the dictionary. SRResNet [9] extended the basic structure with skip connection to form a residual block and cascaded a large number of blocks to construct very deep residual networks.

Sparsity of hidden representation in deep neural networks cannot be solved by iterative optimization as sparse coding, since deep networks are feed-forward during inference. Sparsity of neurons is commonly achieved by ReLU activation in [10] by thresholding negative values to zero independently in each neuron. Still, its 50% sparsity on random vectors is far from the sparsity definition on the overall number of non-zero components. Oppositely, sparsity constraints are more actively used in
model parameters to achieve network pruning \[11\]. However, hidden representation dimensionality is reduced in pruned networks, and accuracy may hurt.

In this paper, we propose a method that can structurally enforce sparsity constraints upon hidden neurons in deep networks but also keep representation in high dimensionality. Given high-dimensional neurons, we divide them into groups along channels and allow only one group of neurons to be non-zero each time. The adaptive selection of the non-sparse group is modeled by tiny side networks upon context features. And computation is also saved when only performed on the non-zero group. However, the selecting operation is not differentiable, so it is difficult to embed the side networks for joint training. We relax the sparse constraints to soft and approximately reduce as a sparse linear combination of multiple convolution kernels instead of hard selection. We further introduce additional cardinal dimensions to decompose sparsity prediction into sub-problems by splitting each sparse group and concatenating after cardinal-independent combination of parameters.

To demonstrate the significance of neural sparse representation, we conduct extensive experiments on image restoration tasks, including image super-resolution, denoising, and compression artifacts removal. Our experiments conclude that: (1) dedicated constraints are essential to achieve neural sparsity representation and benefit deep networks; (2) our method can significantly reduce computation cost and improve accuracy, given the same size of model footprint; (3) our method can dramatically enlarge the model capacity and boost accuracy with negligible additional computation cost.

## 2 Related Work

### 2.1 Sparse coding and convolutional networks

Here we briefly review the application of sparsity in image restoration and its relation to convolutional networks. Considering image super-resolution as an example of image restoration, sparse coding based method \[1\] assumes that input image signal $X$ can be represented by a sparse linear combination $\alpha$ over dictionary $D_1$, which typically is learned from training images as

$$X \approx D_1 \alpha, \text{ for some } \alpha \in \mathbb{R}^n \text{ and } ||\alpha||_0 \ll n. \quad (1)$$

In \[3\], a coupled dictionary, $D_2$, for restored image signal $Y$ is jointly learned with $D_1$ as well as its sparse representation $\alpha$ by

$$Y \approx D_2 \alpha. \quad (2)$$

Convolutional networks, which consist of stacked convolutional layers and non-linear activation functions, can be interpreted with the concepts from sparse coding \[8\]. Given for instance a small piece of network with two convolutional layers with kernels $W_1, W_2$ and a non-linear function $F$, the image restoration process can be formalized as

$$Y = W_2 * F(W_1 * X). \quad (3)$$

The convolution operation $*$ with $W_2$ is equivalent to projecting input image signal $X$ onto dictionary $D_1$. The convolution operation $*$ with $W_2$ is corresponding to the projection of the signal representation on dictionary $D_2$. These two convolutional layers structure is widely used as a basic residual block and stacked with multiple blocks to form very deep residual networks in recent advances \[9\] \[12\] of image restoration.

Dimensionality of hidden representation or number of kernels in each convolutional layer determines the size of dictionary memory and learning capacity of models. However, unlike sparse coding, representation dimensionality in deep models is usually restricted by running speed or memory usage.

### 2.2 Sparsity in parameters and pruning

Exploring the sparsity of model parameters can potentially improve robustness \[13\], but sparsity in parameters is not sufficient and necessary to result in sparse representation. Furthermore, group sparsity upon channels and suppression of parameters close to zero can achieve node pruning \[14\] \[11\] \[15\] \[16\] \[17\], which dramatically reduces inference computation cost. Despite efficiency, node pruning reduces representation dimensionality proportionally instead of sparsity, limits representation diversity, and leads to accuracy regression.
Figure 1: Illustration of computation reduction in two-layer neural networks with sparse hidden nodes in a simplified matrix multiplication example. Left: network with sparsity constraints, which only allow one group with \( c \) hidden nodes to be non-zero over \( kc \) nodes in total. Right: reduced computation with only \( W_{1i} \) and \( W_{2i} \), since other activation nodes are zero. (Grayscale reflects magnitude of matrix values. Matrix multiplication is in right to left order.)

2.3 Thresholding and gating

Thresholding function, ReLU [[18]] for example, plays the similar role of imposing the sparsity constraints [[10]] by filtering out negative values to zero, and contributes to significant performing improvement over previous activation functions, i.e., hyperbolic tangent. Although ReLU statistically gives only 50% sparsity over random vectors, there is still a significant gap between sparsity definition in Eq. 1. Gating mechanism, in Squeeze-and-Excitation [[19, 20]], for example, scales hidden neurons with adaptive sigmoid gates and slightly improves sparsity besides noticeable accuracy improvements. Both thresholding and gating are applied independently to hidden neurons and could not inherently guarantee global sparsity in Eq. 1.

3 Methodology

We propose novel sparsity constraints to achieve sparse representation in deep neural networks. Relaxed soft restrictions are more friendly to gradient-based training. Additional cardinal dimension refines the constraints and improves the diversity of sparse representation.

3.1 Sparsity in hidden neurons

Unlike the methods discussed in Section 2.3 only considering local sparsity for each neuron independently, our approach enforces global sparsity between groups. Specifically, the hidden neurons are divided into \( k \) groups with \( c \) nodes in each group, and only one group is allowed to contain non-zero values. Correspondingly, convolution kernels can also be divided upon connected hidden neurons. Then only the kernels connected to non-zero neurons need to be accounted. Formally, for networks structure in Eq. 3, the convolution kernels are divided as

\[
W_1 = [\cdots W_{1i}^{T}, \cdots] \\
W_2 = [\cdots W_{2i}^{T}, \cdots] \\
\]

Then the Eq. 3 can be rewritten as

\[
Y = W_2^i * F(W_1^i * X) \quad \quad \left(4\right)
\]

When sparsity constraints only allow the \( i \)th group of neurons with non-zero components, then Eq. 4 can be reduced, as shown in Figure 1 and formally as

\[
Y = W_2^i * F(W_1^i * X). \quad \quad \left(5\right)
\]

The proposed sparsity is supposed to pick the node group with the largest amplitude and cannot be achieved without computing the values of all the nodes. In our approach, the selection of the only non-zero group is modeled by a multi-layer perceptron (MLP) with respect to the input signal \( X \).

Regular convolution operations need the kernels shared for every pixel. Hence the selection should also be identified through the spatial space. We are inspired by the Squeeze-and-Excitation [[19, 20]] operation and propose to add pooling operation before the MLP and broadcasting operation for group selection. The above procedure can be formalized as

\[
i = \arg\max_{j \in [1, k]} \text{MLP}(\text{Pool}(X), j). \quad \quad \left(6\right)
\]
Figure 2: Illustration of weighted neurons in soft sparsity constraints and reduced counterpart with weighted sum of parameters. Left: network with soft sparsity constraints, weights $\beta_i$ are applied to neurons in $k$ groups. Right: approximate reduction by firstly weighted summing of parameter groups into a small slice then applying it to features.

Note that, as most of patch-based algorithms [1, 20] for image restoration, the pooling operation should be with respect to a specific patch size instead of the whole image.

Comparison to thresholding and gating. The proposed method limits the number of non-zero entities under $1/k$ of all the nodes in hidden-layer representation, which is more closed to sparsity definition in Eq. [1] than thresholding and gating methods discussed in section 2.3. The proposed method also dramatically reduces computation cost by $k$ times by only considering the adaptively selected group, which is not possible with thresholding and gating methods.

Comparison to node pruning. Node pruning is designed to diminish activation nodes by zeroing all the related trainable parameters. The pruned nodes stick to zero no matter how the input signal varies, which substantially reduces representation dimensionality. In our method, the sparsity adaptively depends on input. Although the input inherently keeps the high dimensionality in representation, our method saves computation and memory cost as narrow models.

3.2 Relaxed soft sparsity

Similar as L0-norm in sparse coding, the adaptive sparse group selection in Eq. [6] is not differentiable and feasible to be jointly learned with neural networks. Although Gumbel trick [21] is proposed to re-parameterize the argmax with respect to a conditional probability distribution, it does not achieve convincing results in our experiment settings.

The sparsity constraints are relaxed by substituting selection with softmax as a smooth approximation of max. Instead of predicting index over $k$, the MLP is relaxed to predict probability over groups $\beta = [\beta_1, \beta_2, \ldots, \beta_k] \in \mathbb{R}_{(0,1)}^k$ with softmax function $\sigma(\cdot)$ by

$$
\beta = \sigma(\text{MLP}(\text{Pool}(X))).
$$

Then, the two-layer structure in Eq. [4] is updated to adaptive weighted sum of groups as

$$
Y = \sum_{i=1}^{k} \beta_i W_i^2 \ast F(W_i^1 \ast X).
$$

With weighted summation, Eq. [8] cannot be directly reduced as Eq. [5] since none of group weights is exactly zero. Fortunately, given sparse assumption of softmax outputs, $\exists i$, s.t. $\beta_i \gg \beta_j \rightarrow 0, \forall j \neq i$, and piece-wise linear activation function $F$, ReLU for example, it can be proved that weighted sum of hidden neurons can be approximately reduced to weighted sum of parameters $W^2$, as shown in Figure [2] and formally as

$$
Y \approx \left( \sum_{i=1}^{k} \sqrt{\beta_i} W_i^2 \right) \ast F(\left( \sum_{i=1}^{k} \sqrt{\beta_i} W_i^1 \right) \ast X).
$$

Note that the two $\sqrt{\beta}$ applied to $W_1$ and $W_2$ are not necessary to be identical to achieve the approximation. Our experiments show that independently predicting weights for $W_1$ and $W_2$ has benefits for accuracy.

In this way, networks restricted by soft sparse constraints can be as efficient as those with hard constraints. And the only additional computation cost from the interpolation of convolution kernels is negligible comparing with convolution operations with the image.
Figure 3: Illustration of our method. Features of image patch are firstly spatially pooled and feed in MLP with softmax activation to predict sparsity constraints $\gamma \in \mathbb{R}^{d,k}$. Softmax function is performed along $k$ axis. Convolution kernel $W$ is divided into $k$ sparsity groups and $c$ channels per group $W^i$. Each group is further divided into $d$ cardinal groups and $c/d$ channels per group $W^{j,i}$. The cardinal-independent weighted sum is performed as Eq. 10. Finally, the aggregated kernel $\hat{W}$ convolves with the original features. (Colors reflect sparsity groups and grayscale reflects magnitude of matrix values.)

**Comparison to conditional convolution.** CondConv [22] has similar operation of the adaptive weighed sum of convolution kernels as our relaxed soft sparsity approach. However, CondConv uses the sigmoid function to normalize the weights of kernels instead of softmax function in our method. Hence, no sparsity constraints are explicitly applied in CondConv, and our experiments show that sparsity is very important for model accuracy.

### 3.3 Cardinality over sparsity groups

Modeling sparsity between groups with a simple MLP is challenging, especially when dimensionality $c$ per group grows. Also, bonding channels within pre-defined groups limits diversity of the sparsity patterns. Inspired by group convolution in ResNeXt [23], we split the $c$ nodes per sparsity group into $d$ cardinal groups, and each cardinal group with $c/d$ nodes is independently constrained along $k$ sparsity groups, as shown in Figure 3. Formally, the averaging weights are extended to matrix $\gamma = [\gamma_1, \gamma_2, \ldots, \gamma_d] \in \mathbb{R}^{d,k}_{(0,1)}$ and $\gamma_i = \sigma(\text{MLP}_i(\text{Pool}(X)))$, then weighted averaged convolution kernel becomes

$$\hat{W} = \text{concat}_{j=1}^d \left( \sum_{i=1}^k \gamma_{j,i} W^{j,i} \right),$$

(10)

where $W^i = [W^{1,i}, W^{2,i}, \ldots, W^{d,i}]$ and $W^{j,i}$ is the $j$th cardinal group and $i$th sparsity group. concat is concatenation operation along the axis of output channels. Notably, with cardinal grouping, Squeeze-and-Excitation [19] operation becomes a particular case of our approach when $d = c$, $k = 1$ and the MLP activation is substituted with sigmoid function.

### 4 Experiments

#### 4.1 Settings

**Datasets and benchmarks.** We use multiple datasets for image super-resolution, denoising, and compression artifacts removal separately. For image super-resolution, models are trained with DIV2K [24] dataset which contains 800 high-quality (2K resolution) images. The DIV2K also comes with 100 validation images, which are used for ablation study. The datasets for benchmark evaluation include Set5 [25], Set14 [2], BSD100 [26] and Urban100 [27] with three up-scaling factors: x2, x3 and x4. For image denoising, training set consists of Berkeley Segmentation Dataset (BSD) [26] 200 images from training split and 200 images from testing split, as [28]. The datasets for benchmark evaluation include Set12, BSD64 [26] and Urban100 [27] with additive white Gaussian noise.
Table 1: Comparison of different sparsity constrains.

| Sparsity | N/A | Sigmoid | Gumbel | Softmax($\tau = 10$) | Softmax($\tau = 1$) | Softmax($\tau = 0.1$) |
|----------|-----|---------|--------|---------------------|---------------------|---------------------|
| PSNR     | 34.76 | 34.81   | 34.45  | 34.86               | **34.87**            | 34.83               |

noise (AWGN) of level 15, 25, 50. For compression artifacts removal, training set consists of 91 images in [11] and 200 training images in [26]. The datasets for benchmark evaluation include LIVE1 [29] and Classic5 with JPEG compression quality 10, 20, 30 and 40. Evaluation metrics include PSNR and SSIM [30] for predicted image quality in luminance or grayscale, only DIV2K is evaluated in RGB channels. FLOPs per pixels is used to measure efficiency, because the runtime complexity is proportional input image size for fully convolutional models.

**Training settings.** Models are trained with nature images and their degraded counterparts. Online data augmentation includes random flipping and rotation during training. Training is based on randomly sampled image patches for 100 times per image and epoch. And total training epochs are 30. Models are optimized with L1 distance and ADAM optimizer. Initial learning rate is 0.001 and multiplied by 0.2 at 20 and 25 epochs.

### 4.2 Ablation study

We conduct ablation study to prove the significance of neural sparse representation. The experiments are evaluated on DIV2K validation set for image super-resolution with x2 up-scaling under PSNR. We use WDSR [31] networks with 16 residual blocks, 32 neurons and 4x width multiplier as the baseline, and set $k = 4$ for sparsity groups by default.

**Sparsity constraints.** Sparsity constraints are essential for representation sparsity. We implement the hard sparsity constraints with Gumbel-softmax to simulate the gradient of hardmax and compare it with soft sparsity achieved by softmax function. The temperature in softmax also controls the sharpness of output distribution. When the temperature is small, softmax outputs are sharper and closer to hardmax. Thus gradient will vanish. When the temperature is large, softmax outputs are more smooth, then it will contradict with our sparsity assumption in Eq. 9 for approximation. We also compare them with a similar model with sigmoid function as MLP activation instead of sparsity constraints in CondConv [22]. Results in Table 1 show that Gumbel-based hard-sparsity methods are not feasible and even worse than the baseline without sparsity groups. Temperature is necessary to be initialized with proper value to achieve better results, which coincides with the above analysis. Sigmoid also gets worse results than softmax because sigmoid cannot guarantee sparsity, which also agrees with our comparison in the previous section.

**Cardinality.** Cardinal dimension reduces the actual dimensionality and dependency between channels in sparsity groups and improves the diversity of linear combination weights over convolution kernels. Results of models with different cardinalities in Fig. 4 show that increasing cardinality constantly benefits accuracy. We also compare with Squeeze-and-Excitation (SE) model, which is a special case of our method, under the same FLOPs. And our models significantly outperform the SE model.

**Efficiency.** Our method can approximately save computation by $k$ times with $k$ sparsity groups but remains the same model size or number of parameters. Results in Table 2 have the same model size in columns and show that our method can save at least half of computation without hurting accuracy uniformly for various model sizes.

**Capacity.** Our method can also extend model capacity or number of parameters by $k$ times with $k$ sparsity groups but only with negligible additional computation cost. Results in Table 3 have the same computation cost in columns and show that our method can continually improve accuracy by extending model capacity up to 16 times.

**Visualization of kernel selection.** It is difficult to directly visualize the sparsity of high-dimensional hidden representation, we take the selection of kernels as a surrogate. As Fig. 5 in the first block, the weights are almost binary everywhere and only depend on color and low-level cues. In later blocks, the weights are more smooth and more attentive to high-frequency positions with more complicated
Table 2: Models with the same size. FLOPs is inversely proportional to group size.

| Group size | # of residual blocks |
|------------|----------------------|
| 2          | 33.91 34.29 34.56 34.76 |
| 3          | 33.92 34.30 34.57 34.77 |
| 4          | 33.86 34.23 34.51 34.71 |
| 8          | 33.81 34.21 34.41 34.68 |

# params (M): 0.15 0.30 0.60 1.2

Table 3: Models with the same FLOPs. Model size is proportional to group size.

| Group size | # of residual blocks |
|------------|----------------------|
| 2          | 33.91 34.29 34.56 34.76 |
| 3          | 33.98 34.38 34.65 34.83 |
| 4          | 34.07 34.45 34.70 34.87 |
| 8          | 34.14 34.50 34.74 34.89 |
| 16         | 34.17 34.56 34.77 34.91 |

FLOPs (M): 0.15 0.30 0.60 1.2

Figure 5: Visualization of kernel selection for network with 2x sparsity group and 4 residual blocks. First column shows input images. Later columns shows softmax MLP outputs of the second convolutional layer in each block. Blue and yellow denotes two groups respectively.

texture. And the last layer is more correlated to semantics, for example, tree branches in the first image and lion in the second image.

4.3 Main results

In this section, we compare our method on top of the state-of-the-art methods on image super-resolution, image denoising, and image compression artifact removal.

Table 4: Public image super-resolution benchmark results and DIV2K validation results in PSNR / SSIM. The better result with small and large EDSR are underlined and in bold respectively.

| Dataset | Scale | Bicubic | VDSR | EDSR (S) | Sparse + | EDSR (L) | Sparse - |
|---------|-------|---------|------|----------|----------|----------|----------|
| Set5    | ×2    | 33.66 / 0.9299 37.53 / 0.9587 | 37.99 / 0.9604 38.02 / 0.9610 | 38.11 / 0.9601 | 38.23 / 0.9614 |
|         | ×3    | 30.39 / 0.8682 33.66 / 0.9213 | 34.37 / 0.9270 34.43 / 0.9277 | 34.65 / 0.9282 | 34.62 / 0.9289 |
|         | ×4    | 28.42 / 0.8104 31.35 / 0.8838 | 32.09 / 0.8938 32.75 / 0.8957 | 32.46 / 0.8969 | 32.55 / 0.8987 |
| Set14   | ×2    | 30.24 / 0.8688 33.03 / 0.9124 | 33.57 / 0.9175 33.60 / 0.9191 | 33.92 / 0.9195 | 33.94 / 0.9203 |
|         | ×3    | 27.55 / 0.7742 29.77 / 0.8314 | 30.28 / 0.8418 30.57 / 0.8445 | 30.52 / 0.8462 | 30.57 / 0.8475 |
|         | ×4    | 26.00 / 0.7027 28.01 / 0.7674 | 28.58 / 0.7813 28.66 / 0.7836 | 28.80 / 0.7876 | 28.79 / 0.7876 |
| B100    | ×2    | 29.56 / 0.8431 31.90 / 0.8960 | 32.16 / 0.8994 32.26 / 0.9008 | 32.32 / 0.9013 | 32.34 / 0.9020 |
|         | ×3    | 27.21 / 0.7385 28.82 / 0.7976 | 29.09 / 0.8052 29.15 / 0.8074 | 29.25 / 0.8093 | 29.26 / 0.8100 |
|         | ×4    | 25.96 / 0.6675 27.29 / 0.7251 | 27.57 / 0.7357 27.61 / 0.7372 | 27.71 / 0.7420 | 27.72 / 0.7414 |
| Urban100| ×2    | 26.88 / 0.8403 30.76 / 0.9140 | 31.98 / 0.9272 32.57 / 0.9329 | 32.93 / 0.9351 | 33.02 / 0.9367 |
|         | ×3    | 24.46 / 0.7349 27.14 / 0.8279 | 28.15 / 0.8527 28.43 / 0.8587 | 28.80 / 0.8653 | 28.83 / 0.8663 |
|         | ×4    | 23.14 / 0.6577 25.18 / 0.7524 | 26.04 / 0.7849 26.24 / 0.7919 | 26.64 / 0.8033 | 26.61 / 0.8025 |
| Manga109| ×2    | 30.80 / 0.9339 37.22 / 0.9750 | 38.55 / 0.9769 38.94 / 0.9776 | 39.10 / 0.9773 | 39.31 / 0.9782 |
|         | ×3    | 26.95 / 0.8556 32.01 / 0.9340 | 33.45 / 0.9439 33.77 / 0.9462 | 34.17 / 0.9476 | 34.27 / 0.9484 |
|         | ×4    | 24.89 / 0.7866 28.83 / 0.8870 | 29.35 / 0.9067 30.63 / 0.9210 | 31.02 / 0.9148 | 31.10 / 0.9145 |
| DIV2K   | validation | 31.01 / 0.8923 33.66 / 0.9290 | 34.61 / 0.9372 34.87 / 0.9395 | 35.03 / 0.9407 | 35.07 / 0.9410 |
|         | ×3    | 28.22 / 0.8124 30.09 / 0.8590 | 30.92 / 0.8734 31.10 / 0.8767 | 31.26 / 0.8795 | 31.30 / 0.8797 |
|         | ×4    | 26.66 / 0.7512 28.17 / 0.8000 | 28.95 / 0.8178 29.10 / 0.8223 | 29.25 / 0.8261 | 29.29 / 0.8263 |

FLOPs (M): 0.67 1.4 1.4 43 9.5
Table 5: Benchmark image denoising results of PSNR / SSIM for various noise levels. Training and testing protocols are followed as in [28]. The best results are in bold and the second are underlined.

| Dataset | Noise | BM3D | WNNM | DnCNN | Baseline | Sparse - | Sparse + |
|---------|-------|------|------|-------|----------|----------|----------|
| Set12   | 15    | 32.77 / 0.8995 | 32.70 / 0.8982 | 32.86 / 0.9031 | 32.97 / 0.9044 | 33.00 / 0.9048 | 33.04 / 0.9054 |
|         | 25    | 39.97 / 0.8504 | 30.28 / 0.8557 | 30.44 / 0.8622 | 30.59 / 0.8655 | 30.63 / 0.8667 | 30.68 / 0.8676 |
|         | 50    | 26.72 / 0.7676 | 27.05 / 0.7775 | 27.18 / 0.7829 | 27.40 / 0.7939 | 27.46 / 0.7954 | 27.51 / 0.7969 |
| BSD68   | 15    | 31.07 / 0.8717 | 31.37 / 0.8766 | 31.73 / 0.8907 | 31.79 / 0.8925 | 31.81 / 0.8928 | 31.83 / 0.8931 |
|         | 25    | 28.57 / 0.8013 | 28.83 / 0.8087 | 29.23 / 0.8278 | 29.30 / 0.8311 | 29.33 / 0.8319 | 29.35 / 0.8327 |
|         | 50    | 25.62 / 0.6864 | 25.87 / 0.6982 | 26.23 / 0.7189 | 26.35 / 0.7272 | 26.37 / 0.7285 | 26.39 / 0.7274 |
| Urban100| 15    | 32.35 / 0.9220 | 32.97 / 0.9271 | 32.68 / 0.9255 | 32.94 / 0.9309 | 32.96 / 0.9316 | 33.05 / 0.9324 |
|         | 25    | 29.70 / 0.8777 | 30.39 / 0.8885 | 29.97 / 0.8797 | 30.33 / 0.8930 | 30.36 / 0.8932 | 30.48 / 0.8959 |
|         | 50    | 25.95 / 0.7791 | 26.83 / 0.8047 | 26.28 / 0.7874 | 26.76 / 0.8181 | 26.84 / 0.8113 | 26.95 / 0.8122 |
| FLOPs (M) | -    | -    | 0.55 | 0.59 | 0.30 | 0.60 |

Table 6: Compression artifacts reduction benchmark results of PSNR / SSIM for various compression qualities. Training and testing protocols are followed as in [28]. The best results are in bold.

| Dataset | q | JPEG | SA-DCT | ARCNN | DnCNN | Baseline | Sparse - | Sparse + |
|---------|---|------|--------|-------|-------|----------|----------|----------|
| LIVE1   | 10 | 27.77 / 0.7905 | 28.65 / 0.8093 | 28.98 / 0.8217 | 29.19 / 0.8123 | 29.36 / 0.8179 | 29.39 / 0.8183 |
|         | 20 | 30.07 / 0.8683 | 30.81 / 0.8781 | 31.29 / 0.8871 | 31.59 / 0.8802 | 31.73 / 0.8832 | 31.79 / 0.8839 |
|         | 30 | 31.41 / 0.9000 | 32.08 / 0.9078 | 32.69 / 0.9166 | 32.98 / 0.9090 | 33.17 / 0.9116 | 33.21 / 0.9121 |
|         | 40 | 32.35 / 0.9173 | 32.99 / 0.9240 | 33.63 / 0.9306 | 33.96 / 0.9247 | 34.18 / 0.9273 | 34.23 / 0.9276 |
| Classic5| 10 | 27.82 / 0.7800 | 28.88 / 0.8071 | 29.04 / 0.8111 | 29.40 / 0.8026 | 29.54 / 0.8085 | 29.56 / 0.8087 |
|         | 20 | 30.12 / 0.8541 | 30.92 / 0.8663 | 31.16 / 0.8694 | 31.63 / 0.8610 | 31.72 / 0.8634 | 31.72 / 0.8635 |
|         | 30 | 31.48 / 0.8844 | 32.14 / 0.8914 | 32.52 / 0.8967 | 32.91 / 0.8861 | 33.07 / 0.8885 | 33.08 / 0.8891 |
|         | 40 | 32.43 / 0.9011 | 33.00 / 0.9055 | 33.34 / 0.9101 | 33.77 / 0.9003 | 33.94 / 0.9028 | 33.96 / 0.9031 |
| FLOPs (M) | -  | -    | 0.11   | 0.55  | 0.30  | 0.60   |

Super-resolution. We compare our method on top of EDSR [12], the state-of-the-art single image super-resolution methods, also with bicubic upsampling, VDSR [32]. As shown in Table 4, the small EDSR(S) has 16 residual blocks and 64 neurons per layer, and our sparse+ model extends it to 4 sparsity groups with cardinality 16 and outperforms on all benchmarks with 4x model capacity but negligible additional computation cost. The large EDSR(L) has 32 residual blocks and 256 neurons per layer, and our sparse- model has 32 residual blocks, 128 neurons per layer, 4 sparsity groups with cardinality 16. Then they have a similar model footprint and on-par benchmark accuracy but 4x computation cost difference.

Denoising. We compare our method with state-of-the-art image denoising methods: BM3D [33], WNNM [34] and DnCNN [28]. As shown in Table 5, our baseline model is residual networks with 16 blocks, 32 neurons per layer, 2x width multiplier [31], and has similar footprint as DnCNN but better performance because of residual connections. Our sparse- model with 2 sparsity groups and 1x width multiplier keeps the model size as baseline but gains 2x computation reduction with better performance. Our sparse+ model adds 2 sparsity groups over baseline model, doubles model capacity, and boosts performance with negligible computation cost.

Compression artifact removal. We compare our method with state-of-the-art image compression artifact removal methods: JPEG, SA-DCT [35], ARCNN [36] and DnCNN [28]. As shown in Table 6, baseline and sparse- models have the same structure as the ones in denoising. Our method consistently saves computation and improves performance on all the benchmark datasets and different JPEG compression qualities.

5 Conclusions

In this paper, we have presented a method to structurally enforces sparsity constraints upon hidden neurons to achieve sparse representation in deep neural networks. Our method trade-offs between sparsity and differentiability, and is jointly learnable with deep networks iteratively. Our method is packed as a standalone module and substitutable for convolution layers in various models. Evaluation and visualization both illustrate the importance of sparsity in hidden representation for multiple image restoration tasks. The improved sparsity further enables optimization of model efficiency and capacity simultaneously.
References

[1] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[2] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730, 2010.

[3] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas Huang. Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing*, 21(8):3467–3478, 2012.

[4] Michael Elad and Michal Aharon. Image denoising via learned dictionaries and sparse representation. In *CVPR*, volume 1, pages 895–900, 2006.

[5] Chen Zhao, Jian Zhang, Siwei Ma, Xiaopeng Fan, Yongbing Zhang, and Wen Gao. Reducing image compression artifacts by structural sparse representation and quantization constraint prior. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(10):2057–2071, 2016.

[6] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2007.

[7] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas Huang. Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing*, 21(8):3467–3478, 2012.

[8] Michael Elad and Michal Aharon. Image denoising via learned dictionaries and sparse representation. In *CVPR*, volume 1, pages 895–900, 2006.

[9] Chen Zhao, Jian Zhang, Siwei Ma, Xiaopeng Fan, Yongbing Zhang, and Wen Gao. Reducing image compression artifacts by structural sparse representation and quantization constraint prior. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(10):2057–2071, 2016.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[12] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR workshops*, pages 136–144, 2017.

[13] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. In *Advances in neural information processing systems*, pages 242–251, 2018.

[14] Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu. Reshaping deep neural network for fast decoding by node-pruning. In *ICASSP*, pages 245–249, 2014.

[15] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

[16] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, 2019.

[17] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *ICLR*, 2019.

[18] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[19] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018.

[20] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, pages 286–301, 2018.

[21] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[22] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems*, pages 1305–1316, 2019.

[23] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.

[24] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPR workshops*, pages 114–125, 2017.

[25] Marco Bevilacqua, Aline Roumy, Christine Guilleminot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, pages 135.1–135.10, 2012.
[26] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In ICCV, volume 2, pages 416–423, 2001.
[27] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In CVPR, pages 5197–5206, 2015.
[28] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing, 26(7):3142–3155, 2017.
[29] Hamid R Sheikh, Muhammad F Sabir, and Alan C Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. IEEE Transactions on Image Processing, 15(11):3440–3451, 2006.
[30] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004.
[31] Jiahui Yu, Yuchen Fan, and Thomas Huang. Wide activation for efficient image and video super-resolution. In BMVC, 2019.
[32] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In CVPR, pages 1646–1654, 2016.
[33] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Transactions on Image Processing, 16(8):2080–2095, 2007.
[34] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In CVPR, pages 2862–2869, 2014.
[35] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. IEEE Transactions on Image Processing, 16(5):1395–1411, 2007.
[36] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In ICCV, pages 576–584, 2015.