We thank all the reviewers. Our novelty is appreciated by R1, R3, and R4, the superior performance is recognized by R1, R2, and R4, and the writing clarity is accredited by R1, R3, and R4. Below we addressed all questions and concerns.

R1: shared β. Following the settings in Table 1, the model with shared β drops PSNR from 34.87 dB to 34.85 dB.

R1: cardinality & performance. Increasing cardinality can constantly improve performance when model is under-parameterized. Since our ablation study is based on small models, the cardinality is limited by the number of channels. We do observe performance regression point when reducing the size of training dataset.

R2: cardinality. The most primitive formulation of cardinality groups is that we start from single convolution layer with only c/d ($/$ denotes numeric division) output nodes and build sparse representation with k groups upon it as Section 3.2, then each sparsity group has $c/(dk)$ nodes and the MLP has k outputs. The cardinality groups repeat building this structure for d times and concatenate output nodes of all the d convolutions. In section 3.3, we move the concatenation operation from convolution output nodes to convolution kernels for formulation simplification.

R2: visualization. Dark blue and light yellow reflect the most certain regions of sparsity group selection. In the second column of Fig. 5, dark blue more likely co-occurs with brown and black texture in original RGB images and light yellow co-occurs with green textures. In the last column, dark blue and light yellow occur only within the region of tree and sky, lion, and squirrel, and especially concentrate in foreground in the second and third images. We think it is more related to semantic, since foreground can be automatically identified and receptive field grows as stacking convolution layers. We will remove the statement about semantic if you still think it is not conclusive.

R3: sparse coding & LISTA. We focus on exploring sparsity of hidden representation in deep neural networks instead of implanting deep structures in sparse coding approaches, such as LISTA. Although LISTA variants could be unrolled into similar formulation as deep models, the iterative shrinkage algorithms solve sparse coding with several folds of computation cost than a single convolution layer, which is hard to be applied in efficient deep models without parameter sharing across layers. In contrast, our method improves the efficiency of deep models, meanwhile increases the number of distinct parameters. We will revise the statement in line 33 as "cannot be efficiently solved" to reduce ambiguity.

R3: sparsity by ReLU. As we discussed in Section 2.3, sparsity by ReLU has been addressed in [10]. ReLU thresholds negatives to zeros and has a gap with sparsity definition in Eq. 1, where the number of non-zero values should be much less than representation dimensionality. Also, ReLU can only determine sparsity for given vectors, but our method predicts the sparsity by a side MLP and saves the computation for unnecessary elements. Experiments show our method constantly outperforms networks only with ReLU activations.

R3: group sparse coding. Our method with multi-level sparsity constraints by ReLU and Softmax shares a similar idea with mixed L2 and L1 norms in group sparse coding. Furthermore, enforcing the constraints into model structure instead of additional objective functions is more favorable for efficiency and performance in deep models. We will add more detailed connection and comparison between group sparse coding and our method in the future version.

R3: guarantee of efficiency. First, our method constantly outperforms baselines over multiple benchmarks including hundreds of images with either less computation and the same model size or the same computation and larger model size. Second, there are many design choices in our methods, and results in the ablation studies coincide with our design and analysis. Third, visualization shows non-trivial sparsity group selection.

R3&R2: state-of-the-art baselines. We focus on the importance of sparse representation of deep models and propose a very general approach to achieve sparsity within individual convolution layer, instead of a single state-of-the-art model. Hence, all the baselines are residual networks with only convolutional and activation layers. Other advanced techniques, such as multi-scale (U-net) and self-attention (non-local), are perpendicular to our method and can be easily combined with ours for superior performance.

R4: coupled dictionary. The coupled dictionary is introduced in [3], where encoding dictionary D_1 and decoding dictionary D_2 are independent two dictionaries. In previous method [1], D_1 and D_2 are the same one.

R4: computation time. We follow the protocol as Squeeze-and-Excitation networks using CPU inference time to address the latency for embedded device applications. We evaluate the forward running time of a single convolution layer of our implementation in PyTorch. The running time is the average of 100 times on quad-core Intel Core i5-2500 at 3.30GHz. We set batch size as 100, input and output channels as 64, patch size as 64. The time for additional computation is around 5-10%. Our implementation can be improved, since SE-Net takes 3ms over 164ms and our approach is more efficient by only modifying convolution kernels instead of features.

Computation time (in milliseconds).

Group size	N/A	2	4	8	16
fixed model size	1.96	1.17	0.77	0.56	0.70
fixed FLOPs	1.96	2.12	2.12	2.08	2.08

R4: patch-based approach. We will add the suggested paper and more methods with fair comparisons including patch-based deep models, such as channel attention and self-attention models.