Deep Block Transform for Autoencoders

Kyong Hwan Jin

***Abstract*—We discover that a trainable convolution layer with a stride over 1 and kernel $\geq$ stride is identical to a trainable block transform. A block transform is performed when we use a convolution layer with a stride ≥ 2 and a kernel $\geq$ the stride. For instance, if we use the same widths, such as a 2×2 convolution kernel and stride-2, there are no overlaps between sliding windows, so this layer operates a block transform on the partitioned 2×2 blocks. A block transform reduces the computational complexity due to a stride ≥ 2. To keep the original size, we apply a transposed convolution (stride $=$ kernel ≥ 2), an adjoint operator of a forward block transform. Based on this relationship, we propose a trainable multi-scale block transform for autoencoders. The proposed method has an encoder consisting of two sequential convolutions with stride-2, a 2×2 kernel, and a decoder consisting of the encoder's two adjoint operators (transposed convolution). Clipping is used for nonlinear activations. Inspired by the zero-frequency element in the dictionary learning method, the proposed method uses DC values for residual learning. The proposed method shows high-resolution representations, whereas the stride-1 convolutional autoencoder with 3×3 kernels generates blurry images.**

***Index Terms*—Block Transform, Image Representation, Autoencoder, Convolutional Neural Network**

I. Introduction

RECENT convolutional neural networks [1], [2], [3] have an odd-sized kernel convolution (ex. 3×3) with an integer stride number as a basic unit. Convolutions with stride ≥ 2 have natural sampling aliasing, resulting from the Shannon-Nyquist sampling theorem [4]. However, aliasing is reduced after nonlinear activations that take place after convolution layers in common convolutional neural networks. Here, we reinterpret convolutions with a stride ≥ 2 and kernel $\geq$ stride as a trainable block transform.

One well-known block transform is discrete cosine transform (DCT) in JPEG image compression [5]. An 8×8 non-overlapped patch is a coding unit for DCT transform [6]. This approach is advantageous for light computations, but it frequently suffers from blocking artifacts. Such artifacts are reduced by a deblocking filter [7]. Recently, block transformations have been exploited in the neural network researches [8], [9]. However, these approaches require legacy transformations, such as DCT and wavelet transform [10], to leverage block-based operations.

In this letter, we discover that a trainable convolution layer with a stride over 1 and kernel over stride number becomes a trainable block transform. This implies that arbitrary multi-scale convolutional neural networks using a stride over 1 are generalized as block transforms, which have a unit block approximated to the receptive field. Based on this property, we propose an efficient block transform network for autoencoders with the same width to stride and kernel to reduce padding errors from image boundaries. DC (zero-frequency) terms calculated from receptive block size are subtracted from the input and then given for the last layer as a skip-connection.

Our contributions are summarized as follows: (1) we discover a block transform from a convolutional layer of a stride ≥ 2 and kernel size $\geq$ stride, (2) we propose an autoencoder based on a trainable block transform, and (3) we demonstrate that our autoencoder outperforms a stride-1 convolutional autoencoder in terms of both quality and computational complexity.

K. H. Jin is with the Department of Information & Communication Engineering, DGIST, South Korea (e-mail: kyong.jin@dgist.ac.kr). This work was supported by the DGIST Start-up Fund Program of the Ministry of Science and ICT(2021030012).

II. Observation

A. A Block Transform from a Convolution with a Stride Greater than 2

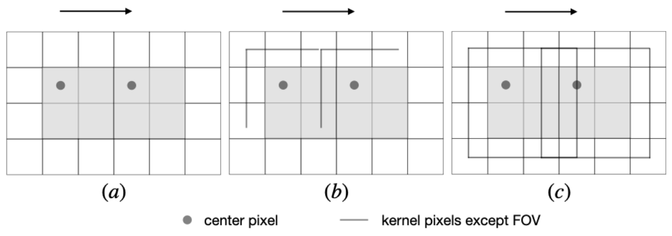

Fig. 1. Examples are stride-2 convolutions. Each example has different kernel sizes: (a) 2×2, (b) 3×3 and (c) 4×4. Grey area means a processing unit. FOV : field of view.

For simplicity, we explain stride-2 convolutions with single-channel input and output. A linear convolution with stride 2 slides its field of view by an amount of 2 pixels as shown in Fig.1. A convolution in Fig.1(a) has a 2×2 kernel, so an output after the convolution is halved in both width and height:

$$f[u,v] = \sum_{n=0}^{s-1}\sum_{m=0}^{s-1} k[n,m]x[s\cdot u+1-n, s\cdot v+1-m], \quad (1)$$

where x is a discrete input signal, k is a 2×2 kernel, s is a stride (2 in this case), and f is a discrete output. The output, f, comes from each 2×2 block without overlaps. This implies that a convolution with the same kernel size ($k \times k$) and stride (k) becomes a block transform on $k \times k$ blocks, which do not need to be orthogonal bases as in DCT [6]. Such equivalence between a block transform and a convolution with the same kernel size ($k \times k$) and stride (k) is still valid with multiple output channels or a transposed convolution because non-overlap sliding is still maintained in a multi-channel or transposed operator. When a given kernel is larger than a stride, such as in Fig.1(b) and (c), the block transform is operated with surrounding pixels at the block's edges. Surrounding pixels compensate for blocking artifacts.

A receptive field in a block transform is calculated as a multiplication between the stride and the kernel's pitch.

B. Error Back-propagation from Boundaries

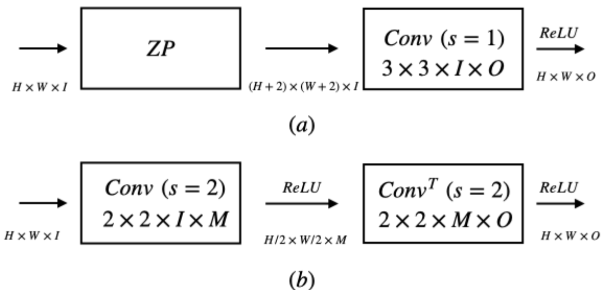

Fig. 2. (a) Conventional convolution with kernel size of 3 after zero-padding and (b) a convolution with stride of 2 and kernel size of 2; this becomes a block transform. ZP : zero-padding, Conv : 2D convolution layer, s : stride.

We review the back-propagation of the trainable block transform. We analyze the two cases shown in Fig. 2: an odd-sized convolution of 3×3, and an even-sized block transform with a 2×2 kernel followed by an adjoint block transform. In matrix-vector form, a 2-D convolution operation becomes a block Toeplitz matrix (row-reversed block Hankel matrix [11]). When a signal $s \in \mathbb{R}^{H \times W \times I}$ passes through a convolution layer, as in Fig. 2 (a), this is described in matrix-vector format as

$$\vec{o} = \mathcal{C}\mathbf{Z}\vec{s}, \tag{2}$$

where $\mathcal{C} \in \mathbb{R}^{HWO \times (H+2)(W+2)I}$ is a block Toeplitz matrix, $\vec{\cdot}$ is a vectorization of a multi-dimensional signal, $\mathbf{Z}^{(H+2)(W+2)I \times HWI}$ is a matrix producing zero-padded vector, and $\vec{o}$ is a corresponding vectorized output signal. When this convolutional layer is updated through back-propagation, $\delta \in \mathbb{R}^{H \times W \times O}$ from a post layer is applied by the chain rule to calculate $\frac{\partial L}{\partial \mathcal{C}}$ (L is the loss function) given by

$$\frac{\partial L}{\partial \mathcal{C}} = \frac{\partial L}{\partial \vec{o}} \frac{\partial \vec{o}}{\partial \mathcal{C}} = \vec{\delta}(\mathbf{Z}\vec{s})^T = \vec{\delta}\vec{s}^T\mathbf{Z}^T. \tag{3}$$

This is an updating term for a convolutional layer, but it undergoes a transposed zero-padding operation ($\mathbf{Z}$) which makes additional errors by inserting zero-rows ($4I + 2HI + 2WI$).

On the other hand, if a signal $s \in \mathbb{R}^{H \times W \times I}$ goes through a proposed block transform layer, as in Fig. 2 (b), this is given as

$$\vec{o} = \mathcal{B}_b \underbrace{\mathcal{R}(\overbrace{\mathcal{B}_a \vec{s}}^{\vec{a}})}_{\vec{z}}, \tag{4}$$

where $\mathcal{B}_a \in \mathbb{R}^{HWM/4 \times HWI}$ is a block Toeplitz matrix, $\mathcal{B}_b \in \mathbb{R}^{HWO \times HWM/4}$ is a transposed block Toeplitz matrix, and $\mathcal{R}$ is a ReLU. Similar to Eq.3, when we update our block transform using the chain rule, the updating term is given by

$$\frac{\partial L}{\partial \mathcal{B}_a} = \frac{\partial L}{\partial \vec{o}} \frac{\partial \vec{o}}{\partial \vec{z}} \frac{\partial \vec{z}}{\partial \vec{a}} \frac{\partial \vec{a}}{\partial \mathcal{B}_a} = \mathcal{B}_b^T \vec{\delta} \left(\mathcal{R}\left(\vec{s}\,\right)\right)^T, \tag{5}$$

$$\frac{\partial L}{\partial \mathcal{B}_b} = \frac{\partial L}{\partial \vec{o}} \frac{\partial \vec{o}}{\partial \mathcal{B}_b} = \vec{\delta}\vec{z}^T = \vec{\delta}\left(\mathcal{R}\left(\mathcal{B}_a \vec{s}\,\right)\right)^T, \tag{6}$$

where $\vec{b}$ is $\mathcal{B}\vec{s}$. ReLU has itself as the derivative ($\frac{\partial \vec{z}}{\partial \vec{a}} = R(\cdot)$). Here, $\mathcal{B}_a$ and $\mathcal{B}_b$ are non-overlapped convolution/transposed convolution matrices, so it is full-rank leading to no additional errors coming from a rank-deficient matrix, such as $\mathbf{Z}$. Thus, trainable block transforms with non-overlapped kernels operate back-propagations without errors on gradients related to zero-padding.

C. Computational Complexity

Given $x \in \mathbb{R}^{H \times W \times I}$, a convolution with a kernel size of 3 in Fig. 2(a) takes $\mathcal{O}(3^2 \cdot HWIO)$. On the other hand, the block transform with a kernel size of 2 in Fig. 2(b) spends $\mathcal{O}(HWM(I+O))$. With the same input, a block transform shows less computational complexity than a convolution approximately 4.5 times faster when $I = O$.

III. Method

A. DC (zero-frequency) Map as Residual Skip

To keep the energy of zero-frequency of a patch, K-SVD [12] uses constant dictionary elements. Inspired by the DC basis of K-SVD, DC values with respect to the receptive block size are subtracted from the input, and we pass it over to the last layer for restoring the original energy of the processing blocks. In practice, we calculate DC values by 2D average pooling with the stride of the receptive block size. The nearest neighborhood interpolation fits DC values into the output size.

Layers	Channel	Kernel (XYC)	Strides (XY)	Output (XYC)
Input				$H \cdot W \cdot 1$
$-$DC$_{S^2}$(Input)				$H \cdot W \cdot 1$
Conv+clip	C	$S \times S \times 1$	$S \times S$	$H/S \cdot W/S \cdot C$
Conv+clip	LC	$S \times S \times C$	$S \times S$	$H/S^2 \cdot W/S^2 \cdot LC$
Conv^T+clip	C	$S \times S \times LC$	$S \times S$	$H/S \cdot W/S \cdot C$
Conv^T	1	$S \times S \times C$	$S \times S$	$H \cdot W \cdot 1$
$+$DC$_{S^2}$(Input)				$H \cdot W \cdot 1$

TABLE I
PROPOSED MULTI-SCALE BLOCK TRANSFORM. WE DENOTE THE PROPOSED METHOD AS BTN/DC-K-S. K : KERNEL, S : STRIDE, LC : NUMBER OF CHANNELS FOR LATENT SPACE, CLIP : CLIPPING IN EQ. (7)

B. Non-overlap Multi-Scale Block Transform

Recall that we construct a trainable block transform by using a stride of more than 2 and a kernel size that is greather than the stride. Here, we propose a trainable multi-scale block transform using non-overlapped block transforms (Table. I). An encoder calculates DC values for each receptive block size and subtracts these from the input. Then, two convolution layers of (stride = kernel) ≥ 2 without zero-padding follow, as in Table. I. The clipping function followed by a convolution is defined as

$$c(x) = \begin{cases} |x| \cdot \text{sign}(x) & \text{if } |x| < 1, \\ \text{sign}(x) & \text{if } |x| \geq 1. \end{cases} \tag{7}$$

A decoder transforms latent features with two sequential transposed convolutions with (stride = kernel) ≥ 2. Again, clipping in Eq. (7) follows transposed convolution. Our network has no padding, leading to error-free backpropagation, as explained in Sec.II-B. At the last layer, DC values are added.

A trainable block transform becomes a fully connected layer when a kernel size and a stride size are the same as the input size. Thus, a multi-scale block transform is limited by the performance of a full-size fully connected layer.

IV. Experiments

We used a stride-1 convolutional autoencoder (denoted as 'AE') as a baseline: Cv(st:1,kr:3), M(st:2), Cv(st:1,kr:3), M(st:2), Cv(st:1,kr:3), B(st:2), Cv(st:1,kr:3), B(st:2), Cv(st:1,kr:3,c:1), where st is the stride, kr is the kernel size, c is the output channel size, Cv is the 2D convolution layer, M is the a 2D max-pooling layer, and B is bilinear interpolation. ReLU followed convolution except for the last layer. Block transform networks without both DC subtraction and skip-connection were also compared: **[BTN-3-2]** Cv(st:2,kr:3), Cv(st:2,kr:3), Cv^T(st:2,kr:3), Cv^T(st:2,kr:3,c:1), and **[BTN-4-2]** Cv(st:2,kr:4), Cv(st:2,kr:4), Cv^T(st:2,kr:4), Cv^T(st:2,kr:4,c:1). ReLU followed after convolution except for the last layer. **BTN-3-2** and **BTN-4-2** cropped boundaries after convolution or transposed convolution to keep the original size.

A. Study1: Numerical Simulation

To verify the benefit of the padding-free property, we assessed our approach ('BTN/DC-2-2', C:12, LC:12) and baselines (C:12, LC:12) with 64×64 numerical images which had 4 lines with different widths at every boundary. Six generated images were ternary (0, 0.5, 1), and a Euclidean norm was used for a loss. The training/test set was the same in this case (overfitting). The number of epochs was 50, and an Adam optimizer was used with a learning rate of 10^{-3}.

B. Study2: Convolutional Autoencoder

We studied the different autoencoder architectures between a stride-1 convolution and a block transform using overlapped/non-overlapped convolution. The training loss was a Euclidean norm. An Adam optimizer was used with a learning rate of 10^{-3}. The total number of elements in the latent space was smaller than the original input size [3]. We calculated the peak signal-to-noise ratio (PSNR) on the testset and simulated 10 experiments to obtain the average PSNR.

1) Fashion MNIST: We used the Fashion MNIST dataset [13] to compare the performance of block transforms and a convolutional autoencoder. We used 12 channels in both the latent space and feature space. The number of epochs was 10, and the batch size was 32.

2) BSDS500: The BSDS500 dataset [14] was used for training. The splitting ratio between the training and validating sets was 4:1. We used 8 channels in both the latent space and feature space. The batch size was 8, the training patch size was 256×256, and the number of epochs was 100. Only the Luma component (Y) was used as a feeding input. The output of 'BTN', Y, was merged with the original Chroma channels (U,V). An ablation study according to the total number of trainable parameters versus PSNRs was conducted with various numbers of channels ($C = 4, 8, 16, 32$) and is presented in Fig.5.

V. Results

A. Study1: Numerical Simulation

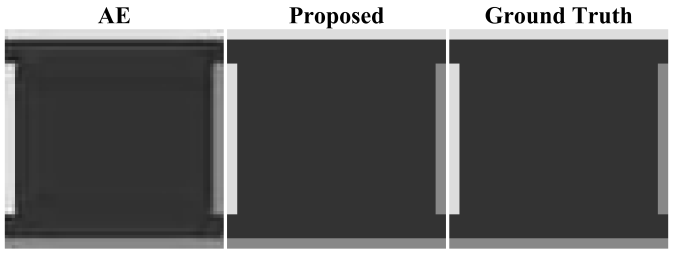

Fig. 3. Numerical simulation results.

As seen in Fig. 3, the proposed method achieved almost perfect reconstruction, whereas the convolutional autoencoder with stride 1 generated ringing artifacts around lines and boundaries. Table II shows the numerical results. In terms of the total number of parameters (Table. II), the proposed method has the smallest numbers.

C:12, LC:12	AE	BTN-3-2	BTN-4-2	BTN/DC-2-2 (**Proposed**)
# of Param.	4,153	2,845	5,029	1,285
Study1 (dB)	31.64	56.65	52.36	105.24
Fashion (dB)	21.38	31.54	31.97	33.67
Time (msec)	75.8	29.9	35.9	26.0

TABLE II
PSNR SUMMARIES FROM NUMERICAL PHANTOMS AND FASHION MNIST. # OF PARAM. IS THE TOTAL NUMBER OF TRAINABLE PARAMETERS.

B. Study2: Convoultional Autoencoder

1) Fashion MNIST: Table II shows the results of the Fashion MNIST experiments. As seen in Fig. 4, AE decoded blurred images; however, the proposed method showed detailed textures with better PSNRs than the baselines in Table II. Computational times for 10^4 samples are reported at the last row in Table II (Tesla T4).

BTN-2-2	+ReLU	+Tanh	+DC	+DC+clip
C:8, LC:8	+Leaky	+clip	+RelU	(**Proposed**)
BSDS400	30.47	32.01	32.42	33.28
(dB)	32.10	32.20		

TABLE III
ABLATION STUDIES WITH BSDS500 DATASET. LEAKY : LEAKY RELU,TANH : HYPERBOLIC TANGENT.

2) BSDS500: Table III shows the results of ablation studies on the combinations of nonlinear activation and skip-connection. The proposed method achieved the best reconstruction. The graph in Fig. 5 shows the PNSRs of the BSDS500 test images on various numbers of channels ($C = 4, 8, 16, 32$) for baselines. As seen in the figure, the proposed method achieved the best performance even with a smaller number of trainable parameters than other baselines. Fig. 6 shows that AE decoded blurry images again; however, the proposed method showed the face of a tiger clearly, in an image that is close to the ground-truth.

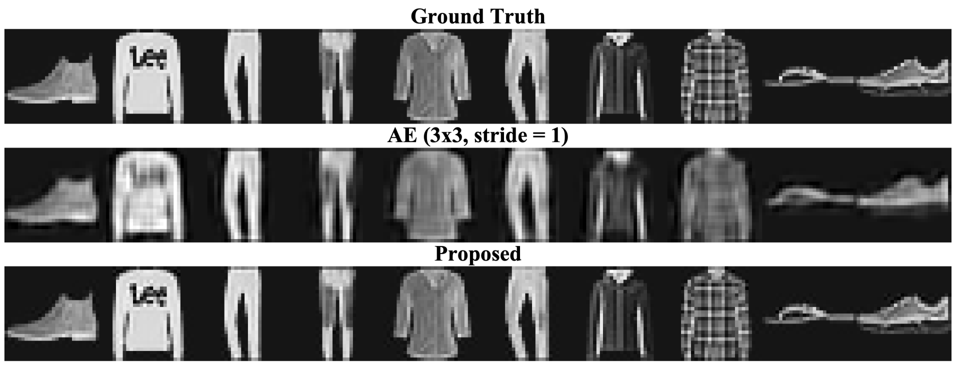

Fig. 4. Fashion MNIST results. Second row shows images from autoencoder(AE) comprised of 3×3 convolutional layers of stride 1. Third row shows images from the proposed block transform network consisting of 2×2 convolution/convolution-transpose layers of stride 2.

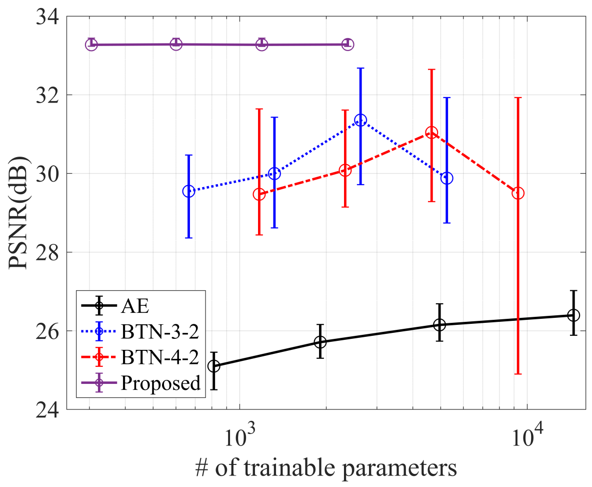

Fig. 5. Error plots for representation of BSDS500 test images (LC : 8). Channel C is $(4, 8, 16, 32)$. Circles mean averaged values for each channel. Error bars mean minimum and maximum PSNRs around mean values.

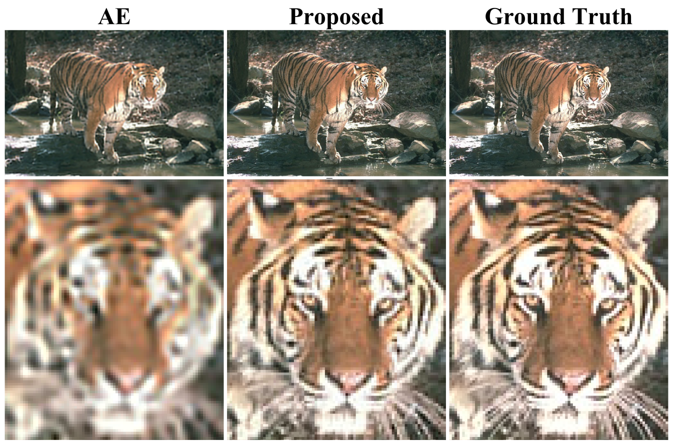

Fig. 6. Representation of a test image in BSDS500 (LC : 8, C:16)

VI. Conclusion

We discovered that a convolution layer with a stride ≥ 2 and a kernel size $\geq$ stride becomes a block transform. We proposed an autoencoder based on the trainable block transform instead of using stride-1 convolution layers. By using the same size for both the stride and kernel, the proposed network is free from padding operations, leading to error-free backpropagation. Clipping is followed by each convolution. We subtract block DC values from the input and add them to the output through a skip-connection to restore the zero-frequency energy of blocks. We demonstrated that the proposed method outperforms a stride-1 convolutional autoencoder. We believe that the trainable block transform could be used for solving inverse problems such as denoising, inpainting, etc.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[2] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
[3] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International conference on artificial neural networks*. Springer, 2011, pp. 52–59.
[4] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.
[5] G. K. Wallace, "The JPEG still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
[6] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.
[7] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera, "HEVC deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1746–1754, 2012.
[8] M. Ehrlich and L. S. Davis, "Deep residual learning in the JPEG transform domain," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3484–3493.
[9] H. Huang, R. He, Z. Sun, and T. Tan, "Wavelet-SRnet: A wavelet-based CNN for multi-scale face super resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1689–1697.
[10] S. Mallat, *A wavelet tour of signal processing*. Elsevier, 1999.
[11] K. H. Jin and J. C. Ye, "Annihilating filter-based low-rank Hankel matrix approach for image inpainting," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3498–3511, 2015.
[12] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.
[13] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[14] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2010.161