Convolutional Neural Networks without Any Checkerboard Artifacts

Yusuke Sugawara, Sayaka Shiota, and Hitoshi Kiya
Tokyo Metropolitan University, 6-6 Asahigaoka, Hino-shi, Tokyo, JAPAN

***Abstract*—It is well-known that a number of convolutional neural networks (CNNs) generate checkerboard artifacts in both of two processes: forward-propagation of upsampling layers and backpropagation of convolutional layers. A condition to avoid the checkerboard artifacts is proposed in this paper. So far, checkerboard artifacts have been mainly studied for linear multirate systems, but the condition to avoid checkerboard artifacts can not be applied to CNNs due to the non-linearity of CNNs. We extend the avoiding condition for CNNs, and apply the proposed structure to some typical CNNs to confirm the effectiveness of the new scheme. Experiment results demonstrate that the proposed structure can perfectly avoid to generate checkerboard artifacts, while keeping excellent properties that the CNNs have.**

***Index Terms*—Convolutional Neural Networks, Checkerboard Artifacts**

I. Introduction

This paper addresses the problem of checkerboard artifacts in convolutional neural networks (CNNs). Recently, CNNs have been widely studying in a variety of computer vision tasks such as image classification [1], [2], semantic segmentation [3], [4], super-resolution [5]–[7] and image generation [8], and have achieved state-of-the-art performances. However, the CNNs often generate periodic artifacts, referred to as checkerboard artifacts, in both of two processes: forward-propagation of upsampling layers and backpropagation of convolutional layers [9].

In CNNs, it is well-known that checkerboard artifacts are generated by operations of deconvolution [10], sub-pixel convolution [11] layers. To overcome these artifacts, smoothness constraint [12], post-processing [13], initialization scheme [14] and different upsampling layer designs [9], [15], [16] have been proposed. Most of them can not avoid checkerboard artifacts perfectly, although they reduce the artifacts. Among them, Odena et al. [9] have demonstrated that checkerboard artifacts can be perfectly avoided by using resize convolution layers instead of deconvolution ones. However, the resize convolution layers can not be directly applied to upsampling layers such as deconvolution and sub-pixel convolution ones, so this method needs not only large memory but also high computational costs. In addition, this method can not be applied to the backpropagation of convolutional layers.

On the other hand, checkerboard artifacts have been studied to design linear multirate systems including filter banks and wavelets [17]–[20]. In addition, it is well-known that checkerboard artifacts are caused by the time-variant property of interpolators in multirate systems, and the condition for avoiding these artifacts have been given [17]–[19]. However, the condition to avoid checkerboard artifacts for linear systems can not be applied to CNNs due to the non-linearity of CNNs.

Because of such a situation, in this paper, we extend the avoiding condition for CNNs, and apply the proposed structure to some typical CNNs to confirm the effectiveness of the new scheme. Experiment results demonstrate that the proposed structure can perfectly avoid to generate checkerboard artifacts caused by both of the two processes, while keeping excellent properties that the CNNs have. As a result, it is confirmed that the proposed structure allows us to offer CNNs without any checkerboard artifacts.

II. Preparation

Checkerboard artifacts in CNNs and works related to checkerboard artifacts are reviewed, here.

A. Checkerboard Artifacts in CNNs

In CNNs, it is well-known that checkerboard artifacts are caused by two processes: forward-propagation of upsampling layers and backpropagation of convolutional layers. This paper focuses on these two issues in CNNs.

When CNNs include upsampling layers, there is a possibility that the CNNs generate some checkerboard artifacts that is the first issue, referred to as issue A. Deconvolution [10], sub-pixel convolution [11] and resize convolution [9] layers are well-known as upsampling layers, respectively.

Checkerboard artifacts are also generated by the backward pass of convolutional layers that is the second issue, referred to as issue B. We will mainly consider issue A in the following discussion, since issue B is reduced to issue A under some conditions.

CNNs are illustrated in Fig. 1 for an SR problem, as in [11], where the CNNs consist of two convolutional layers and one upsampling layer. I_{LR} and $f_c^{(l)}(I_{LR})$ are an low-resolution (LR) image and a c-th channel feature map at layer l, and $f(I_{LR})$ is an output of the network. The two convolutional layers have learnable weights, biases, and ReLU as an activation function, respectively, where the weight at layer l has $K_l \times K_l$ as a spatial size and N_l as the number of feature maps.

There are numerous algorithms for computing upsampling layers, such as deconvolution [10], sub-pixel convolution [11] and resize convolution [9] ones, which are widely used as typical CNNs.

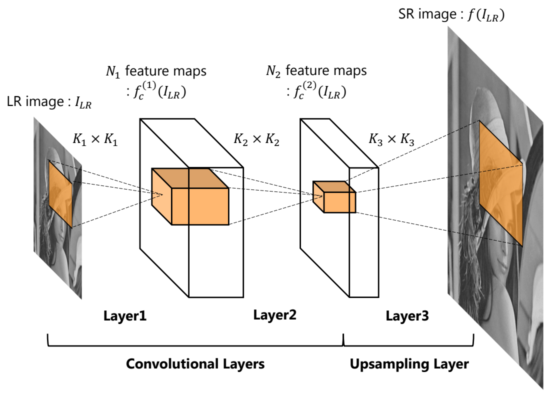


Fig. 1: CNNs with an upsampling Layer

B. Works Related to Checkerboard Artifacts

Checkerboard artifacts have been discussed to design multirate systems including filter banks and wavelets by researchers [17]–[20]. However, most of the works have been limited to in case of using linear systems, so they can not be directly applied to CNNs due to the non-linearity. Some works related to checkerboard artifacts for linear systems are summarized, here.

It is known that linear interpolators which consist of up-samplers and linear time-invariant systems cause checkerboard artifacts due to the periodic time-variant property [17]–[19]. Figure 2 illustrates a linear interpolator with an up-sampler $\uparrow U$ and a linear time-invariant system $H(z)$, where positive integer U is an upscaling factor and $H(z)$ is the z transformation of an impulse response. The interpolator in Fig. 2(a) can be equivalently represented as a polyphase structure as shown in Fig. 2(b). The relationship between $H(z)$ and $R_i(z)$ is given by

$$H(z) = \sum_{i=1}^{U} R_i(z^U) z^{-(U-i)}, \tag{1}$$

where $R_i(z)$ are often referred to as a polyphase filter of the filter $H(z)$.

The necessary and sufficient condition for avoiding the checkerboard artifacts in the system is shown as

$$R_1(1) = R_2(1) = \cdots = R_U(1) = G. \tag{2}$$

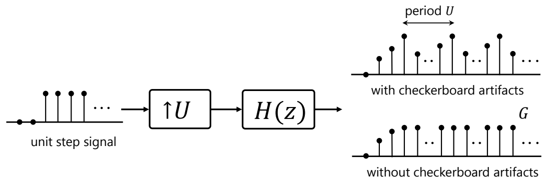
This condition means that all polyphase filters have the same DC value i.e. a constant G [17]–[19]. Note that each DC value $R_i(1)$ corresponds to the steady-state value of the unit step response in each polyphase filter $R_i(z)$. In addition, the condition eq.(2) can be also expressed as

$$H(z) = P(z)H_0(z), \tag{3}$$

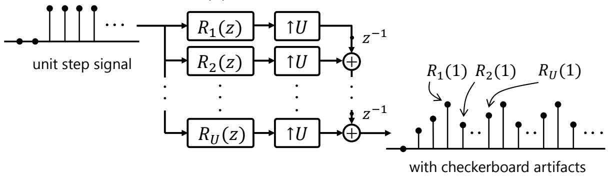
where,

$$H_0(z) = \sum_{i=0}^{U-1} z^{-i}, \tag{4}$$

$H_0(z)$ and $P(z)$ are an interpolation kernel of the zero-order hold with factor U and a time-invariant filter, respectively. Therefore, the linear interpolator with factor U does not generate any checkerboard artifacts, when $H(z)$ includes $H_0(z)$. In the case without checkerboard artifacts, the step response



(a) General structure



(b) Polyphase structure

Fig. 2: Linear interpolators with upscaling factor U

of the linear system has a steady-state value G as shown in Fig. 2(a). Meanwhile, the step response of the linear system has a periodic steady-state signal with the period of U, such as $R_1(1)$, ..., $R_U(1)$, if eq.(3) is not satisfied.

III. Proposed method

CNNs are non-linear systems, so conventional works related to checkerboard artifacts can not be directly applied to CNNs. A condition to avoid checkerboard artifacts in CNNs is proposed, here.

A. CNNs with Upsampling Layers

We focus on upsampling layers in CNNs, for which there are numerous algorithms such as deconvolution [10], sub-pixel convolution [11] and resize convolution [9]. For simplicity, one-dimensional CNNs will be considered in the following discussion.

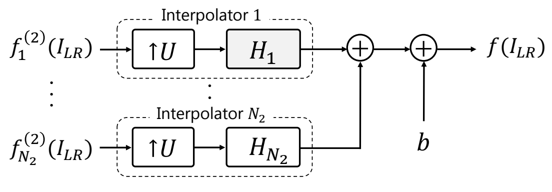
It is well-known that deconvolution layers with non-unit strides cause checkerboard artifacts [9]. Figure 3 illustrates a system representation of deconvolution layers [10] which consist of some interpolators, where H_c and b are a weight and a bias in which c is a channel index, respectively. The deconvolution layer in Fig. 3(a) can be equivalently represented as a polyphase structure in Fig. 3(b), where $R_{c,n}$ is a polyphase filter of the filter H_c in which n is a filter index. This is a non-linear system due to the bias b.

Figure 4 illustrates a representation of sub-pixel convolution layers [11], where $R_{c,n}$ and b_n are a weight and a bias, and $f'_n(I_{LR})$ is an intermediate feature map in channel n. Compared Fig.3(b) with Fig.4, we can see that the polyphase structure in Fig. 3(b) is a special case of sub-pixel convolution layers in Fig. 4. In other words, Fig. 4 is reduced to Fig. 3(b), when satisfying $b_1 = b_2 = ... = b_U$. Therefore, we will focus on sub-pixel convolution layers as the general case of upsampling layers to discuss checkerboard artifacts in CNNs.

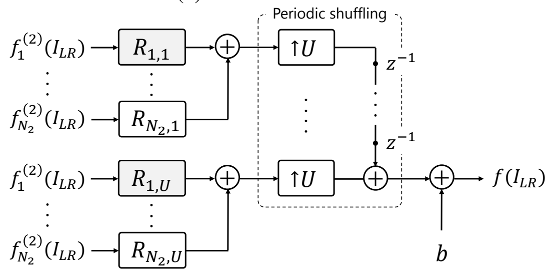
B. Checkerboard Artifacts in Upsampling Layers

Let us consider the unit step response in CNNs. In Fig. 1, when the input I_{LR} is the unit step signal I_{step}, the steady-state value of the c-th channel feature map in layer 2 is given as

$$\hat{f}_c^{(2)}(I_{step}) = A_c, \tag{5}$$



(a) General structure



(b) Polyphase structure

Fig. 3: Deconvolution layer [10]

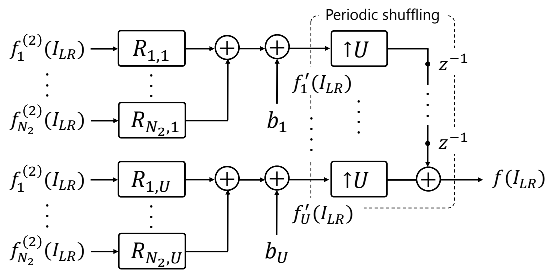


Fig. 4: Sub-pixel convolution layer [11]

where A_c is a positive constant value, which is decided by filters, biases and ReLU. Therefore, from Fig. 4, the steady-state value of the n-th channel intermediate feature map is given by, for sub-pixel convolution layers,

$$\hat{f}'_n(I_{step}) = \sum_{c=1}^{N_2} A_c \overline{R}_{c,n} + b_n, \tag{6}$$

where $\overline{R}_{c,n}$ is the DC value of the filter $R_{c,n}$.

Generally, the condition,

$$\hat{f}'_1(I_{step}) = \hat{f}'_2(I_{step}) = ... = \hat{f}'_U(I_{step}), \tag{7}$$

is not satisfied, so the unit step response $f(I_{step})$ has a periodic steady-state signal with the period of U. To avoid checkerboard artifacts, eq.(7) has to be satisfied, as well as for linear multirate systems.

C. Upsampling Layers without Checkerboard Artifacts

To avoid checkerboard artifacts, CNNs must have the non-periodic steady-state value of the unit step response. From eq.(6), eq.(7) is satisfied, if

$$\overline{R}_{c,1} = \overline{R}_{c,2} = \cdots = \overline{R}_{c,U},\ c = 1, 2, ..., N_2 \tag{8}$$

$$b_1 = b_2 = \cdots = b_U, \tag{9}$$

Note that, in this case,

$$\hat{f}'_1(K \cdot I_{step}) = \hat{f}'_2(K \cdot I_{step}) = ... = \hat{f}'_U(K \cdot I_{step}), \tag{10}$$

is also satisfied as for linear systems, where K is an arbitrary constant value. However, even when each filter H_c in Fig.4 satisfies eq.(3), eq.(9) is not met, but eq.(8) is met. Therefore, we have to seek for a new insight to avoid checkerboard artifacts in CNNs.

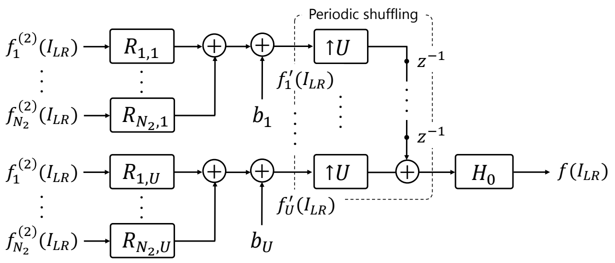


Fig. 5: Proposed upsampling layer structure without checkerboard artifacts

In this paper, we propose to add the kernel of the zero-order hold with factor U, i.e. H_0 in eq.(4), after upsampling layers as shown in Fig. 5. In this structure, the output signal from H_0 can be a constant value, even when an arbitrary periodic signal is inputted to H_0. As a result, Fig. 5 can satisfy eq.(7).

There are two approaches to use H_0 in CNNs as follows.

1) Training CNNs with H_0

The first approach for avoiding checkerboard artifacts, called approach 1, is to add H_0 to CNNs as shown in Fig. 5, and then the CNNs with H_0 are trained. This approach allows us to perfectly avoid checkerboard artifacts generated by CNNs.

2) Training CNNs with H_0 inside upsampling layers

Approach 2 is applicable to only deconvolution layers, although approach 1 is available for both of deconvolution layers and sub-pixel convolution ones. Deconvolution layers always satisfy eq.(9), so eq.(8) only has to be considered. Therefore, CNNs do not generate any checkerboard artifacts when each filter H_c in Fig.5 satisfies eq.(3). In approach 2, checkerboard artifacts are avoided by convolving each filter H_c with the kernel H_0 inside upsampling layers.

D. Checkerboard Artifacts in Gradients

It is well-known that checkerboard artifacts are also generated in gradients of convolutional layers, since operations of deconvolution ones are carried out on the backward pass to compute the gradients. Therefore, both of approaches 1 and 2 are available to avoid the checkerboard artifacts, as well as for deconvolution layers. Note that, for approach 1, we have to add the kernel of the zero-order hold before convolutional layers to avoid checkerboard artifacts on the backward pass.

It is also well-known that max-pooling layers cause high-frequency artifacts in gradients [9]. However, these artifacts are generally different from checkerboard artifacts, so this paper does not consider these high-frequency artifacts.

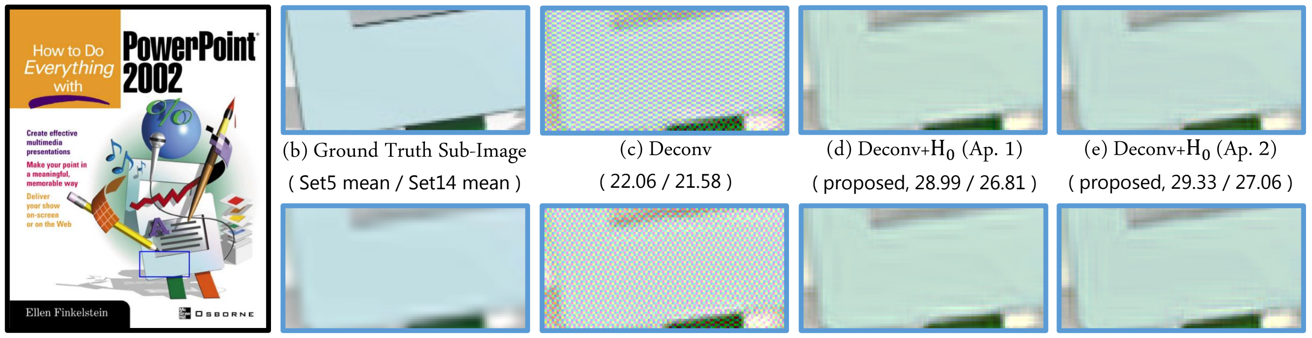
IV. Experiments and results

The proposed structure without checkerboard artifacts was applied to some typical CNNs to demonstrate the effectiveness. In the experiments, two tasks: super-resolution and image classification were carried out.

A. Super-Resolution

1) Datasets for Training and Testing

We employed 91-image set from Yang et al. [21] as our training dataset. In addition, the same data augmentation



(a) Ground Truth Image (Set5 mean / Set14 mean)
(b) Ground Truth Sub-Image (Set5 mean / Set14 mean)
(c) Deconv (22.06 / 21.58)
(d) Deconv+H_0 (Ap. 1) (proposed, 28.99 / 26.81)
(e) Deconv+H_0 (Ap. 2) (proposed, 29.33 / 27.06)
(f) Bicubic (28.42 / 26.00)
(g) Sub-pixel (22.10 / 21.57)
(h) Sub-pixel+H_0 (Ap. 1) (proposed, 29.03 / 26.84)
(i) ResizeConv (29.26 / 27.03)

Fig. 6: Experimental results of super-resolution under perceptual loss (PSNR(dB))

TABLE I: CNNs used for super-resolution tasks

Network Name	Upsampling Layer	$K_3 \times K_3$
Deconv	Deconvolution [10]	9×9
Sub-pixel	Sub-pixel Convolution [11]	3×3
ResizeConv	Resize Convolution [9]	9×9
Deconv+H_0 (Ap. 1)	Deconvolution with H_0 (Approach 1)	9×9
Deconv+H_0 (Ap. 2)	Deconvolution with H_0 (Approach 2)	9×9
Sub-pixel+H_0	Sub-pixel Convolution with H_0 (Approach 1)	3×3

(rotation and downscaling) as in [22] was used. As a result, the training dataset consisting of 1820 images was created for our experiments. Besides, we used two datasets, Set5 [23] and Set14 [24], which are often used for benchmark, as test datasets.

To prepare a training set, we first downscaled the ground truth images I_{HR} with a bicubic kernel to create the LR images I_{LR}, where the factor $U = 4$ was used. The ground truth images I_{HR} were cropped into 72×72 pixel patches and the LR images were also cropped 18×18 pixel ones, where the total number of extracted patches was $8,000$. In the experiments, the three channels of RGB images were used.

2) Training Details

Table I illustrates CNNs used in the experiments, which were carried out based on CNNs in Fig. 1. For other two layers in Fig. 1, we set $(K_1, N_1) = (5, 64)$, $(K_2, N_2) = (3, 32)$ as in [11]. In addition, the training of all networks was carried out to minimize the perceptual loss $\frac{1}{2}\|\phi(I_{HR}) - \phi(f(I_{LR}))\|^2$ averaged over the training set, where ϕ calculates feature maps at the fourth layer of the pre-trained VGG-16 model as in [13]. It is well-known that the perceptual loss results in sharper SR images despite lower PSNR values, and generates checkerboard artifacts more frequently than under the mean squared error (MSE) loss. Note that Deconv+H_0 (Ap. 1), Deconv+H_0 (Ap. 2) and Sub-pixel+H_0 in Table I use the proposed structure.

For training, Adam [25] with $\beta_1 = 0.9, \beta_2 = 0.999$ was employed as an optimizer. Besides, we set the batch size to 4 and the learning rate to 0.0001. The weights were initialized with the method described in He et al. [26]. We trained all models for 200K iterations. All models were implemented by using the tensorflow framework [27].

TABLE II: Execution time of super-resolution (sec)

Resolution of Input Image	Deconv	Deconv+H_0 (Ap. 1)	Deconv+H_0 (Ap. 2)
69×69	0.00871	0.0115	0.0100
125×90	0.0185	0.0270	0.0227
128×128	0.0244	0.0348	0.0295
132×164	0.0291	0.0393	0.0377
180×144	0.0343	0.0476	0.0421

Resolution of Input Image	Sub-pixel	Sub-pixel+H_0 (Ap. 1)	ResizeConv
69×69	0.0159	0.0242	0.107
125×90	0.0398	0.0558	0.224
128×128	0.0437	0.0619	0.299
132×164	0.0696	0.0806	0.383
180×144	0.0647	0.102	0.450

TABLE III: CNNs used for image classification tasks

Network Name	Downsampling Layer	Stride
StridedConv	Convolution	2
StridedConv+H_0 (Ap. 1)	Convolution with H_0 (Approach 1)	2
StridedConv+H_0 (Ap. 2)	Convolution with H_0 (Approach 2)	2

3) Experimental Results

Figure 6 shows examples of SR images, where mean PSNR values for each dataset are also illustrated. In this figure, (c) and (g) include checkerboard artifacts, although (d), (e), (f), (h) and (i) do not include any ones. Moreover, it is shown that the quality of SR images was significantly improved by avoiding checkerboard artifacts. Note that ResizeConv does not generate any checkerboard artifacts, because it uses a pre-defined interpolation like in [5].

Table II illustrates the average executing time when each CNNs were carried out 10 times for some images in Set14. ResizeConv needs the highest computational cost in this table, although it does not generate any checkerboard artifacts. From this table, the proposed structures have much lower computational costs than with resize convolution layers. Note that the result was tested on PC with a 3.30 GHz CPU and the main memory of 16GB.

B. Image Classification

1) Datasets for Training and Testing

We employed two datasets, CIFAR10 and CIFAR100, which contain 32×32 pixel color images and consist of $50,000$ training images and $10,000$ test images [28]. Besides, the standard data augmentation (mirroring and shifting) was used.

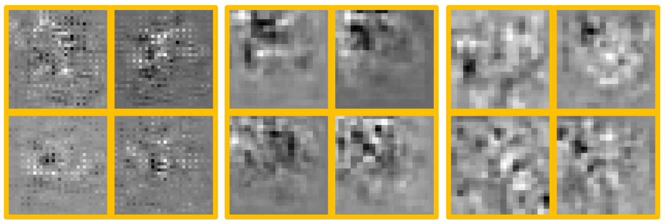

(a) StridedConv (b) StridedConv+H_0 (Ap. 1) (c) StridedConv+H_0 (Ap. 2)

Fig. 7: Gradients computed in the first downsampling layer

TABLE IV: Error rates on CIFAR10, CIFAR100 datasets (%)

Network	CIFAR10	CIFAR10+	CIFAR100+
StridedConv	12.75	6.13	32.72
StridedConv+H_0 (Ap. 1)	16.44	10.08	34.91
StridedConv+H_0 (Ap. 2)	11.21	5.85	29.34

For the preprocessing, the images were normalized by using the channel means and standard deviations.

2) *Training Details*

Table III illustrates CNNs used in the experiments, which were carried out based on ResNet-110 [2]. Note that the projection shortcut [2] was used only for increasing dimensions, and all convolutional layers with stride 2 in ResNet-110 were replaced by downsampling layers in Table III.

All the networks were trained using stochastic gradient descent (SGD) with momentum for 300 epochs. The learning rate was initially set to 0.1, and decreased by a factor of 10 at 150 and 225 epochs. The weights were initialized by the method introduced in [26]. We used the weight decay of 0.0001, the momentum of 0.9 and the batch size of 64.

3) *Experimental Results*

Figure 7 shows examples of gradients, which were computed on the backward pass of the first downsampling layer, for each CNNs. In this figure, (a) includes checkerboard artifacts, although (b) and (c) do not include any ones.

The results on CIFAR10 and CIFAR100 are illustrated in Table IV, where "+" indicates the use of the standard data augmentation. It is shown that approach 2 provided the best performance in this table. This trend is almost the same as for super-resolution tasks.

V. Conclusion

This paper has addressed a condition to avoid checkerboard artifacts in CNNs. The experimental results have demonstrated that the proposed structure can perfectly avoid to generate checkerboard artifacts caused by both of two processes: forward-propagation of upsampling layers and backpropagation of convolutional layers, while keeping excellent properties that the CNNs have. As a result, the proposed structure allows us to offer CNNs without any checkerboard artifacts.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE ICCV*, 2015, pp. 1520–1528.

[4] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[6] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE CVPR*, 2017, pp. 105–114.

[7] Y. Sugawara, S. Shiota, and H. Kiya, "Super-resolution using convolutional neural networks without any checkerboard artifacts," in *Proc. IEEE ICIP*, 2018 (to be accepted).

[8] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[9] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[10] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. IEEE ICCV*, 2011, pp. 2018–2025.

[11] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE CVPR*, 2016, pp. 1874–1883.

[12] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE ICCV*, 2015, pp. 2758–2766.

[13] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. ECCV*, 2016, pp. 694–711.

[14] A. P. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," *arXiv preprint arXiv:1707.02937*, 2017.

[15] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L. C. Chen, A. Fathi, and J. Uijlings, "The devil is in the decoder," in *Proc. BMVC*, 2017.

[16] H. Gao, H. Yuan, Z. Wang, and S. Ji, "Pixel deconvolutional networks," *arXiv preprint arXiv:1705.06820*, 2017.

[17] Y. Harada, S. Muramatsu, and H. Kiya, "Multidimensional multirate filter without checkerboard effects," in *Proc. EUSIPCO*, 1998, pp. 1881–1884.

[18] T. Tamura, M. Kato, T. Yoshida, and A. Nishihara, "Design of checkerboard-distortion-free multidimensional multirate filters," *IEICE Trans. Fundamentals*, vol. E81-A, no. 8, pp. 1598–1606, 1998.

[19] Y. Harada, S. Muramatu, and H. Kiya, "Multidimensional multirate filter and filter bank without checkerboard effect," *IEICE Trans. Fundamentals*, vol. E81-A, no. 8, pp. 1607–1615, 1998.

[20] H. Iwai, M. Iwahashi, and H. Kiya, "Methods for avoiding the checkerboard distortion caused by finite word length error in multirate system," *IEICE Trans. Fundamentals*, vol. E93-A, no. 3, pp. 631–635, 2010.

[21] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[22] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. ECCV*, 2016, pp. 391–407.

[23] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. BMVC*, 2012.

[24] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Curves and Surfaces*, 2010, pp. 711–730.

[25] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE ICCV*, 2015, pp. 1026–1034.

[27] M. Abadi, A. Agarwal, P. Barham, et al, "Tensorflow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[28] A. Krizhevsky, and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.