Pattern Recognition Letters 167 (2023) 149–156

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Dead pixel test using effective receptive field☆

Bum Jun Kim[a], Hyeyeon Choi[a], Hyeonah Jang[a], Dong Gu Lee[a], Wonseok Jeong[b], Sang Woo Kim[a,b,*]

[a] Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 790-784, Republic of Korea
[b] Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang 790-784, Republic of Korea

ARTICLE INFO

Article history:
Received 8 June 2022
Revised 28 October 2022
Accepted 14 February 2023
Available online 16 February 2023

Edited by Jiwen Lu

Keywords:
Convolutional neural networks
Receptive field
Convolutional kernel
Model interpretation
Computer vision
Deep learning

ABSTRACT

Deep neural networks have been used in various fields, but their internal behavior in how they understand images is not well known. In this study, we discuss two counterintuitive properties of convolutional neural networks (CNNs). First, we evaluated the size of the receptive field of CNNs with their classification accuracy. Previous studies have attempted to increase the size of the receptive field for performance gain. However, we observed that some CNNs with a smaller receptive field can achieve higher classification accuracy. In this regard, we claim that a larger receptive field does not guarantee improved classification accuracy. Second, using the effective receptive field, we examined the contribution of each pixel to the output of CNN. Intuitively, each pixel is expected to equally contribute to the final output, but we found that there exist pixels in a partially dead state with little contribution to the output. We reveal that the reason for dead pixels lies in even stride operations with odd-sized kernels in CNN and propose a kernel padding method to remove the dead pixels. We demonstrated the vulnerability of CNNs with dead pixels when we detect a noise or small box that is on dead pixels. Our findings on dead pixels should be understood and considered in practical applications of CNN.

1. Introduction

Deep neural networks have demonstrated remarkable performance in various fields such as object detection, semantic segmentation, and image classification [1–5]. The performance of deep neural networks varies depending on the architecture. State-of-the-art results have been obtained by designing large neural networks with increased depth [6], width [7], and cardinality [8].

In architecture design, one of the factors often considered is the receptive field [9]. For example, if two 3×3 convolutions are applied, a resulting feature covers a 5×5 area. As such, the pixel-level area covered by a specific feature is called the theoretical receptive field. Meanwhile, effective receptive field was proposed by Luo et al. [10]. Contrary to the previous square-shaped theoretical receptive field, the effective receptive field illuminates the actually activated pixels through gradients, whose shape appears as a 2D Gaussian.

Many studies have preferred enlarging the receptive fields to obtain performance gain [11–14]. Further, a paper [9] conjectured the relationship between the size of the receptive field and the classification accuracy. However, our study points out that these common practices should be reconsidered. For modern convolutional neural networks (CNNs), we measured the size of the receptive field. We observed that the large receptive field cannot guarantee the performance superiority of the neural network. For example, some neural networks exhibit high accuracy but have a smaller receptive field. This is because the size of the receptive field reflects only the depth or kernel size and does not reflect the width or cardinality.

In addition to examining the size of the effective receptive field, we investigate the shape of the effective receptive field. We further obtained the effective receptive field of the final output. Conventionally, for a CNN, every pixel, or at least an adjacent pixel, is expected to contribute almost equally to the final output. In other words, it would be strange if a pixel at a specific location is partially dead, and the dead pixel has little effect on the output for any data. Surprisingly, we found that the partially dead pixels exist. In modern CNNs such as ResNet, strong pixels and weak pixels exist for any data. Here, the contribution to the output is significantly different for strong pixels and weak pixels. We will show that this pixel sensitivity imbalance is significant even for adjacent

☆ Editor: Jiwen Lu

* Corresponding author at: Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 790-784, Republic of Korea.
E-mail addresses: kmbmjn@postech.edu (B.J. Kim), hyeyeon@postech.edu (H. Choi), hajang@postech.edu (H. Jang), dgleee@postech.edu (D.G. Lee), wonseok.jeong@postech.edu (W. Jeong), swkim@postech.edu (S.W. Kim).

https://doi.org/10.1016/j.patrec.2023.02.018
0167-8655/

pixels. This pixel sensitivity imbalance occurs when an operation with an odd-sized kernel is applied with an even stride. A solution to this problem is provided in Section 4.

Is pixel sensitivity imbalance a bug or a feature? We compared the performance of CNNs after reducing the pixel sensitivity imbalance. Interestingly, pixel sensitivity imbalance does not degrade but rather enhances the neural network's performance. In this respect, pixel sensitivity imbalance is a feature for general vision tasks. However, when pixel sensitivity imbalance exists, it is difficult to capture small perturbations in images. In contrast, when the pixel sensitivity imbalance is reduced, the neural networks easily capture small perturbations in images. In this regard, pixel sensitivity imbalance is a bug for some special tasks.

Here, we summarize the main contributions of this study as follows:

- We observed that the large receptive field does not guarantee the improvement of CNN's accuracy. For example, we observed a case in which a CNN with a large receptive field has rather a low accuracy. We claim that existing practices to improve performance by enlarging the size of the receptive field should be reconsidered.
- By obtaining the effective receptive field for modern CNNs, we observed that there is an imbalance in the sensitivity of pixels. We reveal that the cause lies in an even stride operation with an odd-sized kernel. As a solution to this, kernel padding is proposed. After applying kernel padding, we observed alleviations of pixel sensitivity imbalance in various CNNs.
- We discuss potential problems that may arise due to the different sensitivity of each pixel. First, we theoretically and experimentally verified that sensitive pixels provide large output differences, which can be exploited for adversarial attacks. Furthermore, we discuss another problem of pixel sensitivity imbalance when we apply random perturbation to an image. To understand this problem, we designed a micro-object classification task and observed the vulnerability of the existing CNNs.

2. Preliminaries: receptive field

In the theoretical receptive field, the largest pixel-level area covered by the target feature is investigated by tracing backward operations in the CNN. For example, if three 3×3 convolutions are applied, one target feature has a 7×7 theoretical receptive field. If any of these operations have a stride greater than 1, the target feature will cover a larger area, resulting in a wider theoretical receptive field [9].

However, as the theoretical receptive field is the theoretical maximum area covered by a target feature, it is far from the practical behavior of the neural network. In the effective receptive field, gradients are used to examine the actual pixels that affect the target feature. Contrary to the theoretical receptive field that appears as a square, the effective receptive field appears as a 2D Gaussian.

Here we provide a detailed formulation of our trick to obtain the effective receptive field (Algorithm 1). Suppose an image $I_{xyz} \in \mathbb{R}^{224\times224\times3}$ is given. The image is passed through given CNN, resulting in a target feature map $A_{ijk} \in \mathbb{R}^{7\times7\times N_c}$. For effective receptive field, the goal is to represent the spatial relationship between pixel-level (x, y) and feature-level (i, j). Therefore, the channel of the image and feature map should be ignored and averaged. First, we define $F = \frac{1}{N_c}\sum_k A_{44k}$, which is the averaged feature over channel k for the spatial center $(4, 4)$ in the target feature map. Then we compute the gradient w.r.t image, $\frac{\partial F}{\partial I_{xyz}}$. By averaging the gradient over the channel z, we obtain $G_{xy} = \frac{1}{3}\sum_z \frac{\partial F}{\partial I_{xyz}}$, which represents how pixel (x, y) affects the central feature for the given image. However, the G_{xy} from a single image is sparse and depends on the image. By averaging G_{xy} over a sufficiently large number of data, the nature of the neural network can be obtained.

Algorithm 1: Obtain effective receptive field.

Require: Image set $\{I\}$, CNN
Return: Effective receptive field R_{xy}
Initialize R_{xy} as a zero matrix;
for $I_{xyz} \in \{I\}$ **do**
  Pass image I_{xyz} to CNN;
  Obtain the target feature map: A_{ijk};
  Aggregate central feature: $F \leftarrow \frac{1}{N_c}\sum_k A_{44k}$;
  Compute gradient: $\frac{\partial F}{\partial I_{xyz}}$; `/* Batch size 1 */`
  Aggregate the gradient over the z-channel: $G_{xy} \leftarrow \frac{1}{3}\sum_z \frac{\partial F}{\partial I_{xyz}}$;
  Accumulate positive contributions: $R_{xy} \leftarrow R_{xy} + \frac{1}{N} ReLU(G_{xy})$;
end
return R_{xy};

However, if some G_{xy} has a negative value, it cancels out with a G_{xy} that has a positive value. For example, averaging $\{1, -3, 2\}$ would result in a zero contribution. As we want to obtain the accumulation of pixel contributions, we ignore negative importance, following the practice in Selvaraju et al. [15]. Thus, we pass G_{xy} through the *ReLU* [16]:

$$R_{xy} = \frac{1}{N}\sum_n ReLU(G_{xy}). \tag{1}$$

Now, R_{xy} represents the general contribution property of pixel (x, y) to the target feature, i.e., the effective receptive field. The effective receptive field R_{xy} is expected to appear as 2D Gaussian:

$$R_{xy} \approx \frac{1}{2\pi\hat{\sigma}_X\hat{\sigma}_Y}\exp\left[-\left\{\frac{(x-\frac{w}{u}i)^2}{2\hat{\sigma}_X^2} + \frac{(y-\frac{h}{v}j)^2}{2\hat{\sigma}_Y^2}\right\}\right], \tag{2}$$

where $u, v = 7$, $w, h = 224$, $i, j = 4$, and $\hat{\sigma}_X$, and $\hat{\sigma}_Y$ are the standard deviations for the 2D Gaussian. In other words, the 2D Gaussian centered at $(\frac{w}{y}i, \frac{h}{v}j)$ spreads at 224×224 pixel-level.

In summary, we need first to calculate G_{xy} for each image, pass it through *ReLU*, and then average it over a sufficiently large dataset. In the modern deep learning environment using mini-batch, applying *ReLU* to the gradient for each image can be difficult. We recommend using batch size 1 to correctly accumulate $ReLU(G_{xy})$ for each image. Accumulating $ReLU(G_{xy})$ over a sufficiently large amount of data yields a clean, high-quality effective receptive field R_{xy} that well describes the internal behavior of a neural network. For more details, see our code on https://github.com/kmbmjn/deadpixeltest_code.

3. Size test on the effective receptive field

Here, we investigate the size of the effective receptive field of modern CNNs. Target CNNs are ResNet and its variants [6–8], which are widely used in various vision tasks. We used `torchvision.models` [17] that were pre-trained on ImageNet. For each model, we summarized the top-5 accuracy reported. We also computed the size of the theoretical receptive field for each model. Note that since ResNet differs in detailed architecture for each implementation, the sizes of the theoretical receptive field for `torchvision.models` are different from that of the `TensorFlow` models [9].

For each pre-trained model, we obtained an effective receptive field using the test dataset from the CUB-200-2011 dataset [18]. Here, we set the target feature map as the last feature map of 7×7

Table 1
For ResNet and its variants, we summarize top-5 classification accuracy (%), theoretical receptive field (TRF) size, effective receptive field size ($\hat{\sigma}_X$, $\hat{\sigma}_Y$), and R^2 from fitting. Contrary to previous studies, we observed that the classification accuracy was not proportional to the size of the theoretical receptive field. The size of the effective receptive field also did not show a tendency consistent with the classification accuracy.

Model	Acc@5	TRF	$\hat{\sigma}_X$	$\hat{\sigma}_Y$	R^2
ResNet-18	89.078	435	76.534	73.662	0.908
ResNet-34	91.420	899	96.148	97.242	0.785
ResNet-50	92.862	427	64.820	62.208	0.945
ResNet-101	93.546	971	60.061	60.679	0.937
ResNeXt-50-32x4d	93.698	427	70.921	67.679	0.908
ResNet-152	94.046	1451	60.589	58.738	0.930
Wide-ResNet-50-2	94.086	427	58.415	60.676	0.954
Wide-ResNet-101-2	94.284	971	56.900	59.888	0.944
ResNeXt-101-32x8d	94.526	971	77.664	77.047	0.889

stage-4. The target feature map can be selected from a feature map at an arbitrary location, such as 7×7 stage-4 or 14×14 stage-3 in CNN. However, if we select an intermediate feature map in CNN, only partial properties of CNN are obtained, so we selected the last feature map of 7×7 stage-4 to see the full properties of CNN. The obtained effective receptive field was fitted with 2D Gaussian using the `Lmfit` library. These processes require 87 s for ResNeXt-101-32x8d. The resulting $\hat{\sigma}_X$ and $\hat{\sigma}_Y$ indicate the standard deviation of the 2D Gaussian, i.e., how large the effective receptive field is. We also provide the coefficient of determination R^2, which indicates how well the effective receptive field matched to the 2D Gaussian. These results are summarized in Table 1. Our major observations are summarized as follows.

Observation 1 (A larger theoretical receptive field does not guarantee improved classification accuracy)**.** We observed that the classification accuracy of CNNs is not proportional to the size of the theoretical receptive field. For example, Wide-ResNet-50-2 and ResNet-152 show similar classification accuracy, but the size of their theoretical receptive field is 427 and 1451 pixels, respectively. This is because the theoretical receptive field reflects only depth or kernel size and cannot reflect width or cardinality. On the other hand, ResNet-34 has a large theoretical receptive field of 899 pixels because it uses early convolution with stride 2 in residual blocks, unlike ResNet-50. As such, ResNet-34 has a wider theoretical receptive field than ResNet-50, but its classification accuracy is lower. These observations are inconsistent with the conjecture [9] that the classification accuracy tends to be proportional to the size of the theoretical receptive field.

Observation 2 (A larger effective receptive field does not guarantee improved classification accuracy)**.** We observed that the classification accuracy of CNNs is also not proportional to the size of the effective receptive field. In other words, even from the viewpoint of the effective receptive field, the large receptive field does not guarantee superiority in performance. Meanwhile, when the depth increases within these ResNets, the size of the effective receptive field does not increase further and saturates to a certain size. These results are different from the study of Luo et al. [10], which reported that the size of the effective receptive field tends to be proportional to $\sqrt{\text{depth}}$.

The same experiment was performed once more. First, each pre-trained ResNet was fine-tuned on the Caltech-101 dataset [19]. The Caltech-101 dataset includes 9 K object images of 101 classes with a background category. The resolution of each image varies and is approximately 200×300 pixels. We replaced the last fully connected layer to output for 102 classes. For training, stochastic gradient descent with momentum 0.9, learning rate 0.01, weight decay 0.0005, batch size 64, epochs 200, and cosine annealing schedule with 200 iterations [20] was used. These hyperparameters are obtained by tuning based on the accuracy on the validation set. For data augmentation, random resized crop with size 256, random rotation with degree 15, color jitter, random horizontal flip, center crop with size 224, and mean-std normalization was applied. The train/val/test set was split at a ratio of 70:15:15. Within 200 epochs, the model with the best validation accuracy was obtained and evaluated. The training takes 4 h and 12 min for ResNeXt-101-32x8d on an RTX 3090 GPU.

Table 2
After fine-tuning each model on the Caltech-101 dataset, the same measurement was performed. Similarly, the classification accuracy (%) did not show a tendency consistent with the size of the receptive field.

Model	Test Acc	TRF	$\hat{\sigma}_X$	$\hat{\sigma}_Y$	R^2
ResNet-18	91.83	435	68.439	71.996	0.953
ResNet-34	92.71	899	78.291	81.658	0.907
ResNet-101	92.71	971	57.916	60.300	0.933
Wide-ResNet-50-2	94.31	427	57.862	63.025	0.963
ResNet-50	94.38	427	56.528	59.245	0.962
ResNet-152	94.60	1451	58.907	60.345	0.916
ResNeXt-50-32x4d	94.97	427	61.416	63.959	0.951
Wide-ResNet-101-2	94.97	971	60.325	64.184	0.935
ResNeXt-101-32x8d	95.40	971	64.720	67.788	0.912

For each fine-tuned model, an effective receptive field was obtained using the test dataset, and its size was investigated (Table 2). Similarly, the size of the theoretical receptive field and the effective receptive field do not agree with the trends in classification accuracy. Therefore, we conclude that the size of the receptive field is not a representative indicator of classification accuracy, nor architectural superiority.

4. Shape test on the effective receptive field

In the previous section, we fitted each effective receptive field to a 2D Gaussian. Although R^2 showed near 0.9, those effective receptive fields did not perfectly match the 2D Gaussian. To understand this behavior, we visualize the obtained effective receptive field.

For the ResNeXt-101-32x8d, we plotted the effective receptive field (Fig. 1). Although the effective receptive field appears as 2D Gaussian, a checkboard pattern exists inside. Therefore, the effective receptive field imperfectly matched the 2D Gaussian because of the internal checkboard pattern.

Additionally, we accumulated $\frac{\partial y}{\partial I}$ to obtain an effective receptive field of output y. This is what we call the dead pixel test. In general, the entire pixels are expected to contribute almost equally to output y. However, even for the effective receptive field of y, we discovered that the checkboard pattern exists.

The existence of this checkboard pattern implies that modern CNNs recognize images in a highly counterintuitive way. Some pixels are weak, partially dead, and hardly contribute to the output. Conversely, some pixels are strong and more sensitive to output. We call this phenomenon *pixel sensitivity imbalance*. As the checkboard pattern appears locally, even in adjacent pixels, the pixel sensitivity differs significantly.

Why does the checkboard pattern appear? We found that it occurs when an odd-sized kernel is applied with an even stride (Fig. 2). For example, when a 3×3 convolution is applied with stride 2, overlapping regions appear. Pixels within the overlapping regions are referenced more in operation, while other pixels are not. As this phenomenon accumulates, some pixels become more influential while others do not. When viewed in 2D, a checkboard pattern appears. This phenomenon is highly similar to the checkboard pattern when using deconvolution in image generation tasks

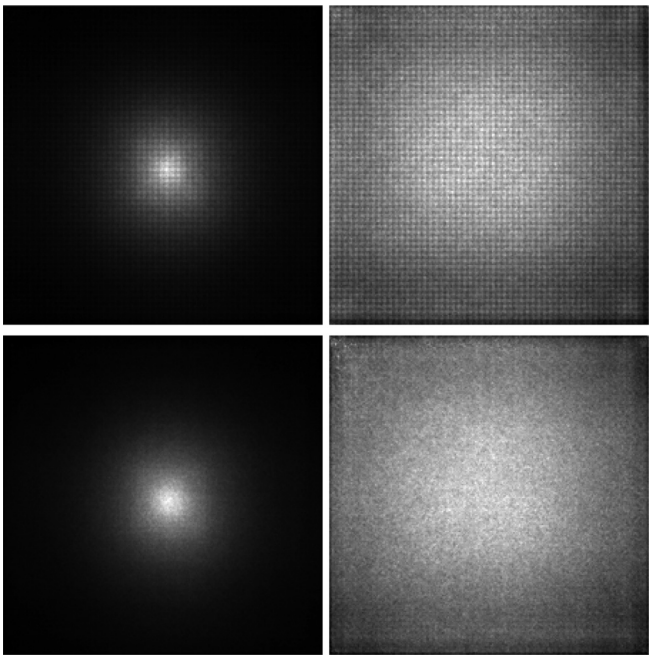

Fig. 1. Effective receptive fields were obtained for ResNeXt-101-32x8d. (Left) Effective receptive field of the 14×14 feature map. (Right) Effective receptive field of the output. The top row is the effective receptive fields for the existing ResNeXt-101-32x8d before kernel padding, showing the checkboard pattern. The bottom row is the effective receptive fields after kernel padding, showing no checkboard pattern. Best viewed electronically with zoom.

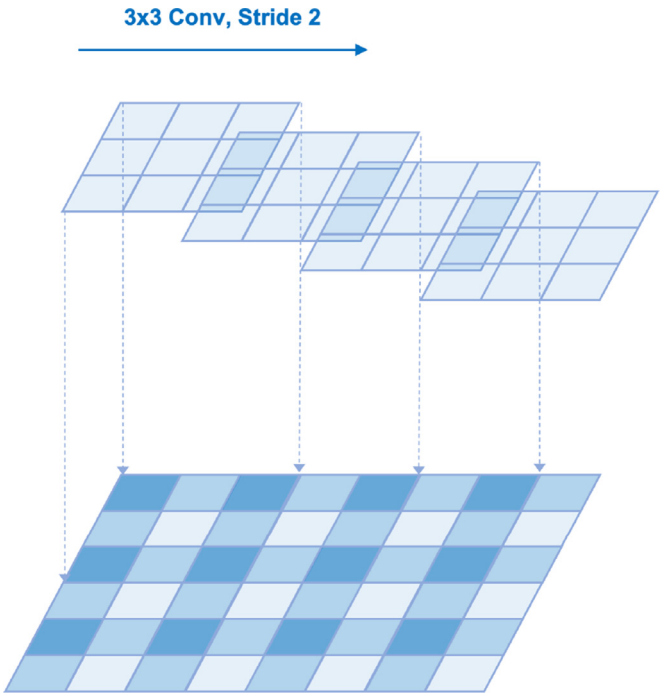

Fig. 2. When an odd-sized kernel is applied with stride 2, a checkboard pattern appears.

[21]. Extending this, we emphasize that the checkboard pattern exists from the perspective of gradient even when using convolution.

Despite these potential problems, odd-sized kernels with stride 2 are widely used in modern CNNs [22–24]. For example, in the early stage of ResNets, 7×7 Conv with stride 2 and 3×3 Pool with stride 2 are used. Further, in the downsampling operation in the

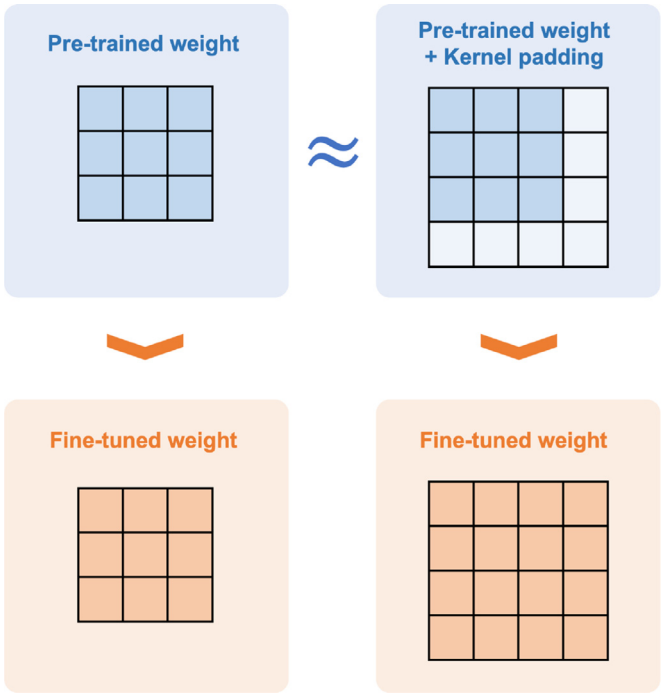

Fig. 3. To construct an even-sized kernel while using pre-trained weights, we propose a kernel padding method. An even-sized kernel is constructed by applying zero-padding to the bottom and right sides of the kernel. After fine-tuning, the added weights along with the existing weights can be properly trained.

residual block, 1×1 Conv is used with stride 2, which subsamples only the specific input and shuts off the flow in other locations.

Here, we would like to modify those problematic odd-sized kernels with stride 2. As ResNet and its variants have similar architectures, most can be modified with similar rules. Not all layers need to be modified. The operations to be modified are as follows: 7×7 Conv with stride 2 and 3×3 Pool with stride 2 in early stage, and 1×1 Conv with stride 2 and 3×3 Conv with stride 2 across all residual blocks. We replace those kernels with even sizes such as 8×8 or 4×4.

However, when replacing with a new kernel, the existing pre-trained weights are discarded. To construct an even-sized kernel while boosting training through pre-trained weights, we propose *kernel padding* method. For the target odd-sized pre-trained weight, zero-padding is applied to the bottom and right sides to obtain an even-sized kernel (Fig. 3). As the kernel is zero-padded, the operation is equivalent to the previous one. Accordingly, pre-trained weights can be enjoyed. Moreover, as the new zero-padded weights are trainable, during fine-tuning, they can be merged into the existing weights.

We applied kernel padding to the ResNets pre-trained on the ImageNet and then fine-tuned them on the Caltech-101 dataset. The training details used in fine-tuning are the same as the experiments in Section 3. Now the effective receptive field of our architecture has no checkboard pattern (Fig. 1).

The degree of the pixel sensitivity imbalance can be measured through the smoothness of the effective receptive field R_{xy} of output y. Here, we define two indices, first-order imbalance index L_1 and second-order imbalance index L_2:

$$L_1 = \frac{1}{224 \times 223 \times 2} \sum_{x,y} (|\partial_x R_{xy}| + |\partial_y R_{xy}|), \tag{3}$$

$$L_2 = \frac{1}{224 \times 222 \times 2} \sum_{x,y} (|\partial_x^2 R_{xy}| + |\partial_y^2 R_{xy}|). \tag{4}$$

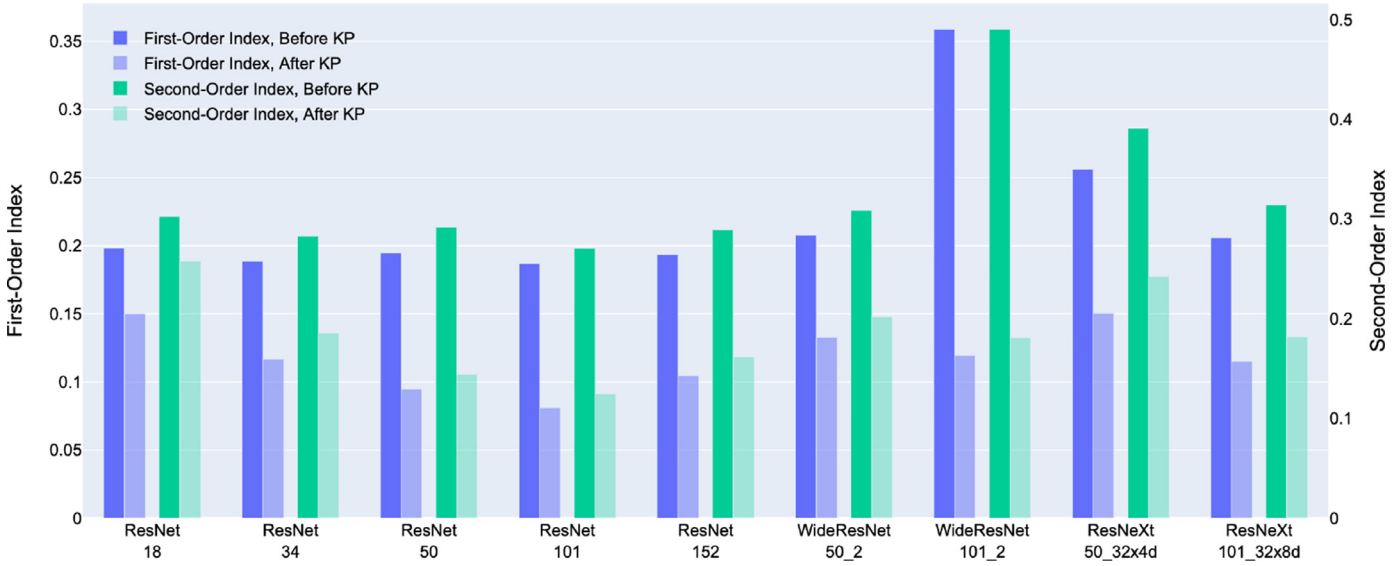

Fig. 4. To quantitatively evaluate pixel sensitivity imbalance, we measured the two indices. In all the target architectures, pixel sensitivity imbalance is reduced after kernel padding.

Table 3
Comparison with other methods on the smoothness of effective receptive field using L_1 index. We denote "R" for ResNet, "W" for Wide-ResNet of 2× width, and "X" for ResNeXt of 32 cardinality.

Model	R-50	R-101	W-50	W-101	X-50	X-101
Vanilla	0.195	0.187	0.207	0.359	0.256	0.205
Gaussian noise	0.164	0.119	0.149	0.270	0.238	0.179
Anti-aliasing	0.120	0.090	0.156	0.193	0.168	0.238
Kernel padding	0.095	0.081	0.134	0.119	0.150	0.116

In other words, we pass the effective receptive field through the difference filters and compute spatial average to evaluate its local variation and curvature. The smaller these values are, the more locally smooth the effective receptive field is. Conversely, the larger the value, the greater the imbalance.

Using these two indicators, we evaluated the degree of pixel sensitivity imbalance before and after applying kernel padding (Fig. 4). Existing ResNets show large L_1 and L_2, which indicates that pixel sensitivity imbalance is significant even in adjacent pixels. After applying the kernel padding, the imbalance decreased across all ResNets.

Furthermore, we found that kernel padding provides a smoother effective receptive field compared to other methods. In fact, the existence of dead pixels is not well known, but several methods can be considered to have an effect similar to kernel padding. First, as a simple method of alleviating pixel sensitivity imbalance, adding Gaussian noise to the image in CNN training can be considered. This applies uniform corruption to both strong and weak pixels, thereby mitigating pixel sensitivity imbalance. Second, one study [25] pointed out the problem with the strided operation of CNN in terms of translation invariance. They improved translation invariance by adding an anti-aliasing operation to the strided operation. We found that the anti-aliasing calibrates the pixel sensitivity imbalance, making the effective receptive field smooth.

Targeting three methods of kernel padding, Gaussian noise, and anti-aliasing, we measured and compared the imbalance index L_1 of the effective receptive field (Table 3). For Gaussian noise and anti-aliasing, training was performed with the same recipe as in Section 3 using ImageNet pre-trained weights. Gaussian noise slightly alleviated the pixel sensitivity imbalance but did not solve the fundamental architectural flaw. Anti-aliasing produced effective receptive fields with improved smoothness, but in the case of ResNeXt-101-32x8d, it produced worse results. Kernel padding showed the smoothest result compared to other methods, and also showed consistent alleviation of pixel sensitivity imbalance in six ResNets.

5. Discussion

Is pixel sensitivity imbalance a bug or a feature? In other words, if the pixel sensitivity imbalance is reduced, can the superiority of the architecture be guaranteed? For both perspectives, we provide some conjectures.

5.1. Pros: pixel sensitivity imbalance is a feature

Even with pixel sensitivity imbalance, ResNets have been widely used in various vision tasks so far. Although some pixels are partially dead, they are not entirely dead. The difference between strong and weak pixels is a matter of contribution degree, and they are all involved in the output.

The phenomenon of blocking the flow of certain inputs is not so unfamiliar. For example, consider Dropout [26] or DropBlock [27]. They improve the performance of neural networks by dropping some neurons or inputs. For understanding the global context of an image, it is fine if some trivial input is missing. Furthermore, dropping some input induces the CNN to understand the image in a different way, introducing a regularization effect.

Further, pixel sensitivity imbalance can be interpreted as rescaling a given image according to strong and weak pixels. As the image is rescaled pixel-wise, when a translated image is given, it is recognized as a completely different image. Accordingly, pixel sensitivity imbalance increases image diversity, thereby boosting the effect of data augmentation.

Here, we examined how pixel sensitivity imbalances affect the performance in a general vision task. As kernel padding reduces pixel sensitivity imbalance, we compared the performance of ResNet and its variants before and after applying kernel padding. We performed fine-tuning on the Caltech-101 dataset, and the experimental details such as the training method and data augmentation are the same as in Section 3.

For each model, we measured the average test accuracy from three experiments (Table 4). We observed that the performance is

Table 4
To investigate whether pixel sensitivity imbalance helps training or not, we compared the test accuracy (%) before and after applying kernel padding. After kernel padding, the performance is rather decreased.

Model	Before KP	After KP	Diff
ResNet-101	**95.28**	94.07	1.22 ↓
Wide-ResNet-101-2	**95.14**	94.09	1.05 ↓
ResNeXt-101-32x8d	**95.58**	94.75	0.83 ↓

rather decreased after kernel padding. This means that pixel sensitivity imbalance is not a bug for a general image classification task but is a feature that improves performance.

5.2. Cons: pixel sensitivity imbalance is a bug

Nevertheless, pixel sensitivity imbalance gives rise to several potential problems.

As mentioned earlier, since pixel sensitivity imbalance introduces pixel-wise rescaling, the translated image is perceived as a completely different image. This increases the data augmentation effect but worsens the translation invariance of CNN [25]. In a practical application, for example, if 1-pixel translated image produces a different result, the vision system would be considered unreliable and unstable.

Moreover, pixel sensitivity imbalance implies a positional difference for capturing a perturbation. Consider one-pixel attack [28], which attempts an adversarial attack to invert the output by perturbing a certain pixel. Here we can additionally exploit the fact that strong pixels are generally more sensitive. If we construct an attack strategy that focuses more on strong pixels, we can attack the neural network more easily. Here, we provide a mathematical formulation.

Theorem 1. *Assume pixel sensitivity imbalance exists by R_{xy} in a CNN with ReLU-like activations that outputs y from an image I. If we put a perturbation ϵ on the image I at (X, Y, Z) pixel, we have $\Delta y \approx \epsilon R_{XY}$, which implies that output is affected by both the magnitude and position of the perturbation.*

Proof. Consider output y, the logit before softmax layer. For a CNN with *ReLU*-like activations, we can represent the output using piece-wise linear function [29,30]:

$$y = \frac{\partial y}{\partial I_{111}} I_{111} + \ldots + \frac{\partial y}{\partial I_{224,224,3}} I_{224,224,3} + C \tag{5}$$

$$= \sum_{x,y,z} \frac{\partial y}{\partial I_{xyz}} I_{xyz} + C. \tag{6}$$

$\frac{\partial y}{\partial I_{xyz}}$ and C come from a combination of weights and biases of the CNN and are evaluated at specific image I. Here, from the mean over various images, we approximate the two terms using $\frac{\partial y}{\partial I_{xyz}} \approx R_{xy}$ and $C \approx E(C)$, which results in a fixed linear model. We define $\tilde{y}(I) = \sum_{x,y} R_{xy}(I_{xy1} + I_{xy2} + I_{xy3}) + E(C)$, which is an approximation of the output y from Eq. 6 and is the output from the fixed linear model.

Now, assume that we put a perturbation ϵ on (X, Y, Z) pixel of image I. Let e_{XYZ} be the standard unit vector for (X, Y, Z). Then,

$$\Delta\tilde{y} = \tilde{y}(I + \epsilon e_{XYZ}) - \tilde{y}(I) \tag{7}$$

$$= \epsilon R_{XY}. \tag{8}$$

If pixel sensitivity imbalance exists, R_{XY} differs depending on the (X, Y). Thus, even if the same amount of ϵ is applied, $\Delta\tilde{y}$ varies depending on where the perturbation is applied. For example, if we put perturbation to a strong pixel, the output can be significantly affected. □

Table 5
Standard deviations for R_{XY} and MD before and after applying kernel padding. After kernel padding, the variations in R_{XY} and MD become smaller, which is desirable.

Item	Model	Before KP	After KP
$\hat{\sigma}(R_{XY})$	R-101	0.3700	**0.1937**
	W-101	0.5713	**0.2485**
	X-101	0.3898	**0.1619**
$\hat{\sigma}(MD)$	R-101	0.0736	**0.0240**
	W-101	0.0575	**0.0164**
	X-101	0.0770	**0.0304**

Here, we verify Theorem 1 empirically. First, ResNet and its variants were trained on the Oxford-IIIT Pet dataset using the same training method in Section 3. Afterward, perturbation ϵ was added to 3 channels of single coordinate (X, Y) for the images of the test dataset. We applied perturbation to the image after ImageNet mean-std normalization and set ϵ to 0.1 accordingly. We measured the mean absolute difference of the output before and after applying perturbation over the test dataset from $MD = \frac{1}{N}\sum_n \|y_\epsilon - y\|_1$.

In general, MD is not expected to change with the coordinates of perturbation. However, we will claim that the MD varies with the location of the perturbation, as described in Theorem 1. Specifically, we will experimentally show that large MD appears in strong pixels.

We measured R_{XY} and MD when perturbation was added to each coordinate for 25 pixels from (100, 100) to (104, 104) in the 224×224 area (Fig. 5). We observed that MD not only differs according to the coordinates but that the change pattern has a strong correlation with R_{XY}. In other words, even when perturbation of the same magnitude is applied, a larger change in output appears when the perturbation is on a strong pixel. Therefore, for the existing ResNet, it is possible to make a larger output change by exploiting the sensitive pixels. This property can be exploited for the adversarial attack on the neural network, and thus pixel sensitivity imbalance is an undesirable property.

Meanwhile, in CNN to which kernel padding is applied, since pixel sensitivity imbalance is alleviated, it is expected that the influence of the location of perturbation will be reduced. Again, we measured R_{XY} and MD for ResNet and its variants to which kernel padding is applied. First, we observed smaller changes in R_{XY} and MD in the model after kernel padding through their standard deviations on 25 pixels (Table 5). Further, a small correlation between R_{XY} and MD was observed after kernel padding (Fig. 5). Ideally, the model after kernel padding does not have a checkboard pattern. The variation of R_{XY} observed here can be understood as an observation close to noise that appears when CNN is trained with a specific weight, rather than indicating a difference between strong and weak pixels. Therefore, the model after kernel padding is hardly affected by the location of perturbation.

In addition, Eq. (8) implies that when pixel sensitivity imbalance exists, it may be difficult to distinguish whether the change in output is due to the magnitude of the perturbation or the position of the perturbation. Then, if perturbations with different magnitudes are applied at random locations, can CNN distinguish the magnitudes of perturbations? Further, if the perturbation magnitude and the position are also randomly varied every time, and only the average of the perturbation magnitude has a difference, it will be quite a challenging problem. However, these problems are commonly encountered in practical vision tasks.

We verify our claim that CNN with pixel sensitivity imbalance has difficulty classifying perturbations of different magnitudes when the position of the perturbation changes every time. To describe this scenario as a practical vision task, we place small-sized

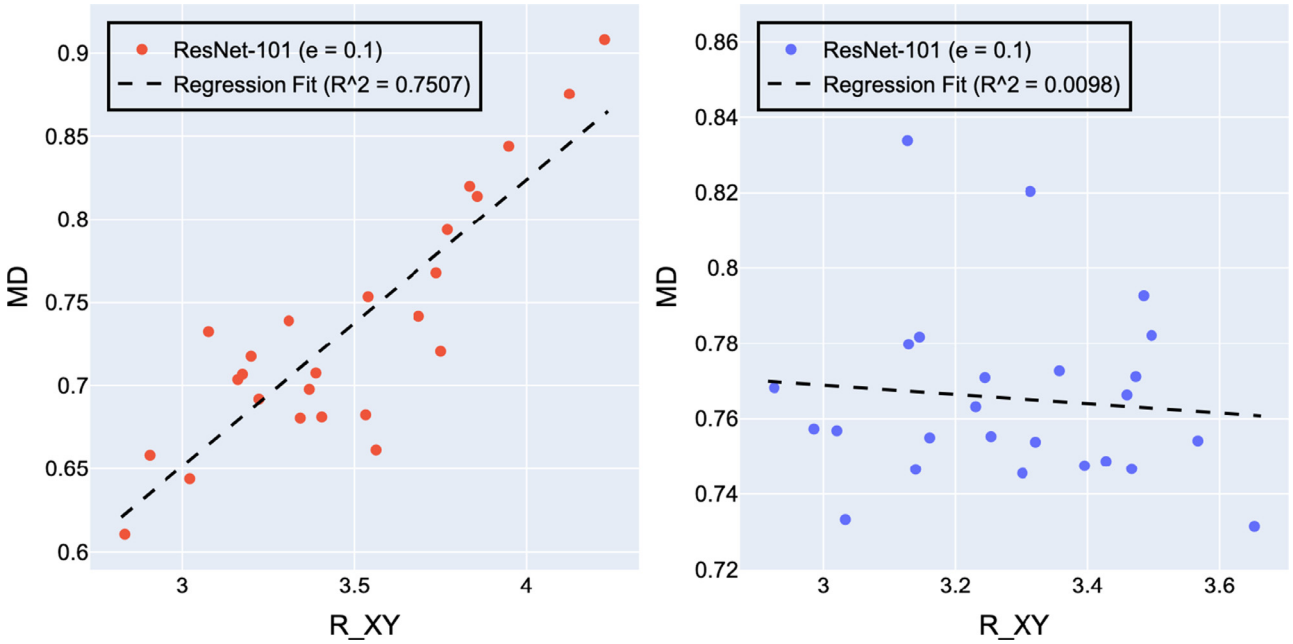

Fig. 5. (Left) Before kernel padding, R_{XY} and *MD* show strong correlations ($R^2 = 0.7507$), which indicates that sensitive pixels exhibit large difference. This property can be exploited for an adversarial attack on the neural network and is an undesirable property. (Right) After kernel padding, the variations in R_{XY} and *MD* become smaller.

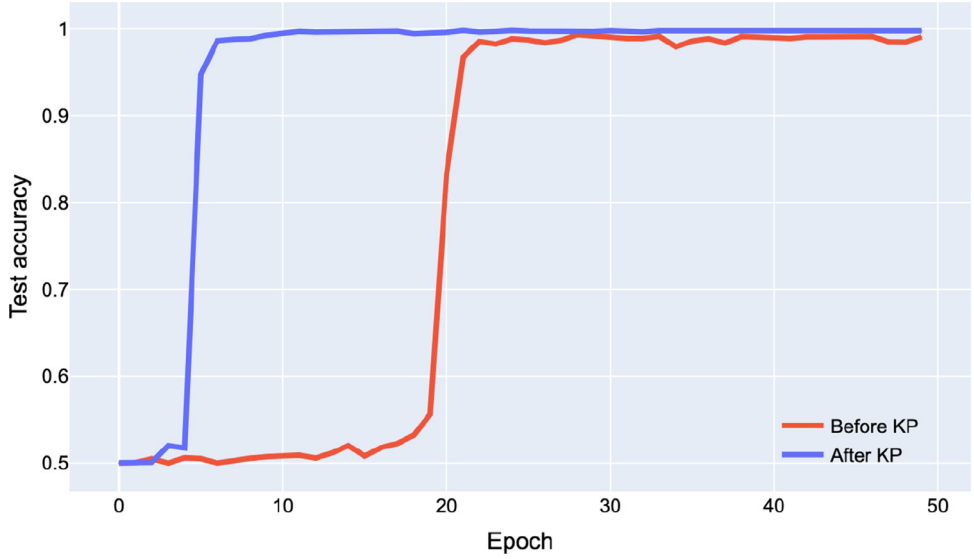

Fig. 6. Training curve before and after applying kernel padding for the micro-object classification task.

white and red boxes on the image and think of them as perturbations of different magnitudes. We randomly change the position of the box every time to describe the position of the perturbation changing every time. We call this *micro-object classification task*. The templates are images from the Caltech-101 dataset. First, we select a random 8×8 region within the template. After changing the color of the selected area to RGB = (0, 0, 0), we label the image as class A. In the same way, for class B, select a random area, but replace it with RGB = (255, 0, 0). One may choose different choices in color or the size of the box. As such, we put a micro-object at a random location in the image to perform a binary classification task. In this task, not only the position of the perturbation but also the magnitude changes every time. Here, when pixel sensitivity imbalance exists, it may be difficult for CNN to capture the difference in perturbation magnitude between the two classes. In contrast, in the model to which kernel padding is applied, the effective receptive field is smoothed and pixel sensitivity imbalance is reduced. In this case, because the change of R_{XY} by (X, Y) is small, the influence of the randomness of the perturbation location decreases (Eq. (8)). For this scenario, the change in the magnitude of perturbation can be more easily captured.

Experimental details such as training method and data augmentation are almost the same as in Section 3. Here, to better see the intrinsic architectural differences, we did not use pre-trained weights. The number of epochs was set to 50, not 200, because it converges faster than in previous experiments. The observed training curve is shown in Fig. 6. Initially, the test accuracy was around 50%, and the difference in perturbation was not captured. After a certain epoch, the test accuracy increased rapidly, and the difference in the micro-objects was captured with an accuracy of more than 99%. Here, the existing model without kernel padding required more epochs to capture the perturbation difference. In contrast, the model to which kernel padding is applied captured the perturbation difference faster.

Table 6
From five experiments, we computed the average number of epochs where the test accuracy exceeds 90% for the micro-object classification task.

Model	Before KP	After KP	Diff
ResNet-101	28.2	**10.2**	18.0 ↓
Wide-ResNet-101-2	28.0	**14.2**	13.8 ↓
ResNeXt-101-32x8d	21.8	**8.8**	13.0 ↓

To verify this more strictly, we measured the number of epoch where the test accuracy first exceeded 90%. If it did not exceed 90% within 50 epochs, it was evaluated as 50 epochs. For ResNet-101 and its variant, five experiments were performed, and the average of the measured number of epochs was summarized (Table 6). Even in the same training environment, after kernel padding, the difference in the micro-objects was captured 13–18 epochs faster. Thus, for some special tasks, pixel sensitivity imbalance is harmful to training.

6. Conclusion

Summary of our findings In this study, we investigated the behaviors of CNNs using effective receptive fields. First, we investigated the size of the receptive field. Contrary to popular belief, we found that the classification accuracy is not proportional to the size of the receptive field. In addition, we observed that the size of the effective receptive field saturates to a certain level even if the CNN becomes deeper. These observations suggest that we need to reconsider when controlling the size of the receptive field. Second, the pixels contributing to the output were investigated through the effective receptive field of the output. We discovered that in modern ResNets, the contribution to the output is different for each pixel. It was identified that the cause of this pixel sensitivity imbalance lies in the use of an odd-sized kernel with stride 2. To solve this, kernel padding was proposed. We quantitatively evaluated pixel sensitivity imbalance through two indices and found that pixel sensitivity imbalance decreases after kernel padding.

Limitation However, for the general vision task, the pixel sensitivity imbalance is found to be a feature of CNNs, not a bug. We observed that CNNs with pixel sensitivity imbalance exhibit higher classification accuracy than CNNs without pixel sensitivity imbalance, which is the major limitation of this study. Nevertheless, we claim that CNNs with pixel sensitivity imbalance have vulnerabilities in several scenarios. First, we theoretically and experimentally confirmed that pixel sensitivity imbalance causes a different change in CNN output depending on the location of perturbation, which can be exploited for an adversarial attack on the neural network. We extended this weakness of CNNs by proposing a micro-object classification task where we observed the improved performance after applying kernel padding. Thus, pixel sensitivity imbalance is a harmful bug for some tasks. We encourage practitioners to consider and understand our findings on CNNs.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the GitHub link to my code in our manuscript.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd (IO201210-08019-01).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patrec.2023.02.018.

References

[1] Y. Wu, R. Li, Y. Yu, X. Li, Reparameterized attention for convolutional neural networks, Pattern Recognit. Lett. 164 (2022) 89–95.
[2] Z. Zhang, X. Sun, J. Li, M. Wang, MAN: mining ambiguity and noise for facial expression recognition in the wild, Pattern Recognit. Lett. 164 (2022) 23-29.
[3] Y. Zhu, L. Wei, C. Lang, S. Li, S. Feng, Y. Li, Fine-grained facial expression recognition via relational reasoning and hierarchical relation optimization, Pattern Recognit. Lett. 164 (2022) 67–73.
[4] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, A.M. Torres, J.J. Lopez, F.J. Ferri, M. Cobos, An open-set recognition and few-shot learning dataset for audio event classification in domestic environments, Pattern Recognit. Lett. 164 (2022) 40–45.
[5] F. Wan, R. Zhi, Gaussian distribution-based facial expression feature extraction network, Pattern Recognit. Lett. 164 (2022) 104–111.
[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CVPR, 2016.
[7] S. Zagoruyko, N. Komodakis, Wide residual networks, BMVC, 2016.
[8] S. Xie, R.B. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, CVPR, 2017.
[9] A. Araujo, W. Norris, J. Sim, Computing Receptive Fields of Convolutional Neural Networks, Distill, 2019.
[10] W. Luo, Y. Li, R. Urtasun, R.S. Zemel, Understanding the effective receptive field in deep convolutional neural networks, NIPS, 2016.
[11] Y. Tsai, W. Hung, S. Schulter, K. Sohn, M. Yang, M. Chandraker, Learning to adapt structured output space for semantic segmentation, CVPR, 2018.
[12] H. Fu, M. Gong, C. Wang, K. Batmanghelich, D. Tao, Deep ordinal regression network for monocular depth estimation, CVPR, 2018.
[13] B. Singh, L.S. Davis, An analysis of scale invariance in object detection SNIP, CVPR, 2018.
[14] J. Kim, J.K. Lee, K.M. Lee, Accurate image super-resolution using very deep convolutional networks, CVPR, 2016.
[15] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2020) 336–359.
[16] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, AISTATS, 2011.
[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E.Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, NeurIPS, 2019.
[18] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, California Institute of Technology, 2011.
[19] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories, Comput. Vis. Image Underst. 106 (2007) 59–70.
[20] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, ICLR, 2017.
[21] A. Odena, V. Dumoulin, C. Olah, Deconvolution and Checkerboard Artifacts, Distill, 2016.
[22] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely connected convolutional networks, CVPR, 2017.
[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S.E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, CVPR, 2015.
[24] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90.
[25] R. Zhang, Making convolutional networks shift-invariant again, ICML, 2019.
[26] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[27] G. Ghiasi, T. Lin, Q.V. Le, DropBlock: a regularization method for convolutional networks, NeurIPS, 2018.
[28] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Comput. 23 (2019) 828–841.
[29] S. Srinivas, F. Fleuret, Knowledge transfer with Jacobian matching, ICML, 2018.
[30] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps, ICLR, 2014.