What are the Receptive, Effective Receptive, and Projective Fields of Neurons in Convolutional Neural Networks?

Hung Le
University of Central Florida
hungle@knights.ucf.edu

Ali Borji
Center for Research in Computer Vision
University of Central Florida
aborji@crcv.ucf.edu

Abstract

In this work, we explain in detail how receptive fields, effective receptive fields, and projective fields of neurons in different layers, convolution or pooling, of a Convolutional Neural Network (CNN) are calculated. While our focus here is on CNNs, the same operations, but in the reverse order, can be used to calculate these quantities for deconvolutional neural networks. These are important concepts, not only for better understanding and analyzing convolutional and deconvolutional networks, but also for optimizing their performance in real-world applications.

1. Definitions

Receptive Field (RF): is a local region (including its depth) on the output volume of the previous layer that a neuron is connected to. This term has been prevalent in Neurosciences since the study of Hubel and Wiesel [1] in which they suggested local features are detected in early visual layers of the visual cortex and are then progressively combined to create more complex patterns in a hierarchical manner.

As an example, assume that the input RGB image to a CNN has size $[32 \times 32 \times 3]$. For a filter size of 5×5, then each neuron in the first convolutional layer will be connected to a $[5 \times 5 \times 3]$ region in the input volume. Thus, a total of $5 \times 5 \times 3 = 75$ weights (+1 bias parameter) needs to be learned. Notice that RF is a 3D tensor with its depth being equal to the depth of the volume in the previous layer. Here, for simplicity, we discard the depth in our calculation.

Effective Receptive Field (ERF): is the area of the original image that can possibly influence the activation of a neuron. One important point to notice here is that RF and ERF are the same for the first convolutional layer. However, they differ as we moves along the CNN hierarchy. The RF is simply equal to filter size over the previous layer but ERF traces the hierarchy back to the input image and indicates
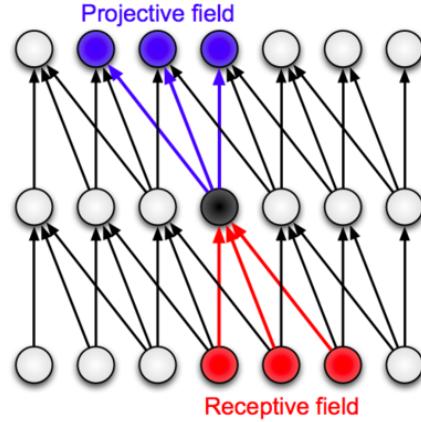

Figure 1. Schematic plot demonstrating receptive and projective fields of a neuron (Borrowed from http://www.scholarpedia.org/article/Projective_field).

the extent of the input image which can modulate the activity of a neuron. Here, we focus on ERF calculation. It is worth noting that ERF and RF are sometimes used interchangeably (and hence confused) in the computer vision community.

Projective Field (PF): is the set of neurons to which a neuron projects its output [3].

Figure 1 illustrated these definitions.

2. Calculating the ERF

In convolutional neural networks [2], the ERF of a neuron indicates which area of the input image is considered by the filter. Calculating the size of the ERF would help choosing suitable filter sizes using domain knowledge for enhancing the performance of CNNs.

There are two ways to calculate the ERF size: 1) Bottom-Up, and 2) Top-Down. Both ways produce the same result. The intermediate values of each approach, however, have different meanings in each case.
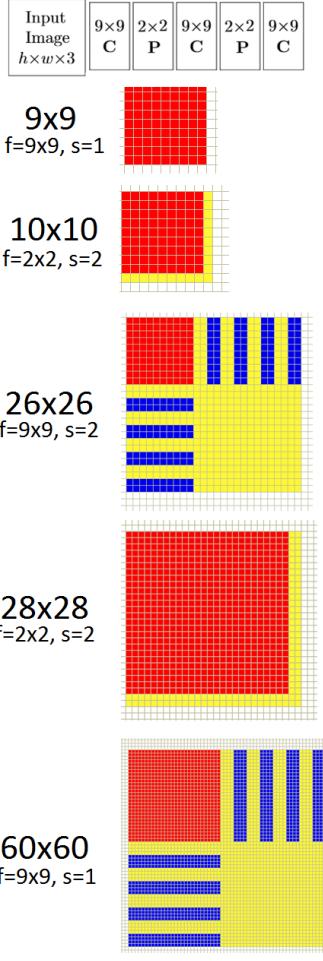
Figure 2. Example of Bottom-up approach for ERF calculation for the network shown in the top row. Red area is the ERF of the lower layer. Yellow and blue are non-overlapped areas, used to indicate how stride affects the calculation of the additional area. In this example, after the first pooling, each additional filter adds 2 pixels to the ERF. After the second pooling, each additional filter adds 4 pixels to the ERF.

2.1. Bottom-Up Approach

The bottom-up approach is the method to calculate the ERF of a neuron at layer k projected on the input image. Let R_k be the ERF of a neuron at layer k. Given the ERF of the previous layer R_{k-1}, where $R_0 = 1$ is the ERF at input image layer, the ERF for a neuron at current layer R_k can be computed by adding the non-overlapped-area A to R_{k-1}:

$$R_k = R_{k-1} + A \tag{1}$$

Let f_k represent the filter size of layer k. There are $(f_k - 1)$ filters overlap with each other. Since a filter can be convolved with a stride greater than one, it can significantly increase the non-overlapped area. Thus, it is necessary to
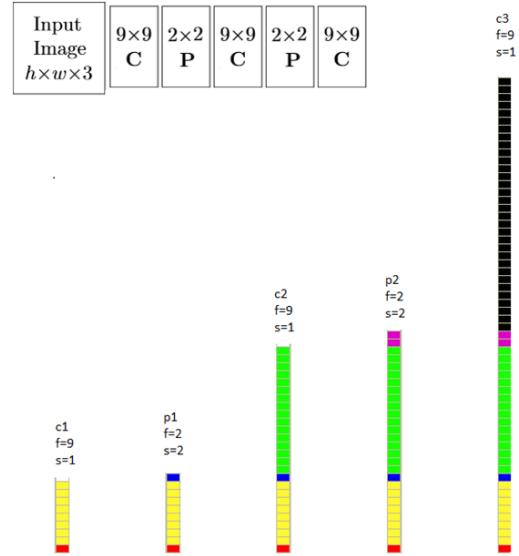

Figure 3. 1 dimensional example illustrating how each layer expands the ERF.

account for the number of pixels each extra filter contributes to the ERF. Since the stride of the lower layer also affects the ERF of the higher layer, the pixel contributions of all layers must be accumulated. Therefore the non-overlapped area is calculated as:

$$A = (f_k - 1) \sum_{i=1}^{k-1} s_i \tag{2}$$

where s_i is the stride of the layer i. Combining equation 1 and equation 2, the ERF can be computed as:

$$R_k = R_{k-1} + (f_k - 1) \sum_{i=1}^{k-1} s_i \tag{3}$$

Figures 2 and 3 illustrate ERF calculation for a sample architecture. The advantage of bottom-up approach is that it produces the ERF for all layers in one feed-forward pass.

2.2. Top-Down Approach

The computation of ERF in this approach is done via calculating the RF a neuron at layer k projected on the lower layer j, where the RF of the last layer would be the ERF. Given the RF of a neuron at higher layer $R_{k,j+1}$, if there is no overlap (i.e., stride equal filter size) then the RF of the current layer is:

$$R_{k,j} = R_{k,j+1} f_{j+1} \tag{4}$$

where f_{j+1} is the filter size of higher layer. The RF is 1 when $j = k$. When the filters are overlapped with

each other, the overlapped area must be subtracted from the value. Imagine placing down a filter, then every filter being placed after it would have area overlap with the previously placed filter. Since the RF of the higher layer is the one being projected down, the number of filters that overlap with each other would simply be the RF of higher layer minus one:

$$O = R_{k,j+1} - 1 \qquad (5)$$

The subsequent filter being placed down would be shifted by the stride, thus the overlapped area is dependent on the size of the filter and the stride. Larger strides would yield to less overlap. Larger filters would result in more overlap. The overlapped area of each filter would be the difference of filter size and stride of the higher layer s_{j+1}:

$$A = f_{j+1} - s_{j+1} \qquad (6)$$

Having the number of overlapped filters and the area that each filter overlapped, the RF at the current layer can be computed by combining the equations 4, 5, and 6:

$$R_{k,j} = R_{k,j+1}f_{j+1} - (R_{k,j+1} - 1)(f_{j+1} - s_{j+1}) \quad (7)$$

Expanding and simplifying the above equation gives the final top-down equation:

$$R_{k,j} = (R_{k,j+1} - 1)s_{j+1} + f_{j+1} \qquad (8)$$

The top-down approach is helpful during the analysis as it can be computed relatively quickly. Also, given a point on a filter, it is possible to speculate the nodes that contributed to its output. For the deconvolutional networks, the top-down approach can be used to control the resolution of the output image. Thus instead of using upside down CNN layers, the deconvolutional layers can be designed to incorporate any domain knowledge about the problem. Figures 4 and 5 show examples of the progression of RF being projected back to lower layers.

2.3. Case Study

Here, we calculate the ERF of neurons for an the CNN from Wei et al., [4]. In their paper, Wei et al. proposed a method for pose estimation (known as Convolutional Pose Machine). Figure 6 shows the original architecture. Here, we focus on calculating ERF for part of the network shown in Figure 7, with the 1x1 filters being omitted.

Bottom-Up approach: The ERF of each layer is computed progressively while skipping the 1×1 filters as they do not have any effect on the ERF size. The process of computing ERF of the architecture in Figure 7, according to equation 3, is shown below:
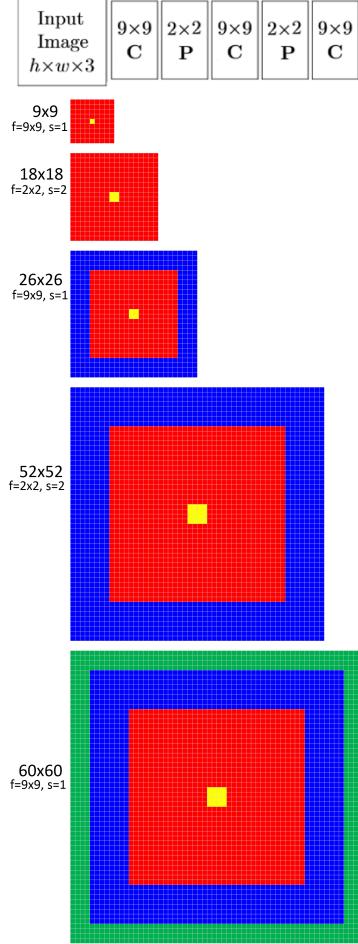

Figure 4. Example of Top-Down approach showing the RF of the last layer (9×9 Convolution) being projected back to the input image. With stride of 2 and filter size of 2×2, the RF is simply doubled in size.

$$
\begin{aligned}
R_0 &= 1 \\
R_1 &= 1 + (9-1)(1) = 9 \\
R_2 &= 9 + (2-1)(1) = 10 \\
R_3 &= 10 + (9-1)(2) = 26 \\
R_4 &= 26 + (2-1)(2) = 28 \\
R_5 &= 28 + (9-1)(2*2) = 60 \\
R_6 &= 60 + (9-1)(2*2) = 64 \\
R_7 &= 64 + (5-1)(2*2*2) = 96 \\
R_8 &= 96 + (9-1)(2*2*2) = 160 \\
R_9 &= 160 + (11-1)(2*2*2) = 240 \\
R_{10} &= 240 + (11-1)(2*2*2) = 320 \\
R_{11} &= 320 + (11-1)(2*2*2) = 400
\end{aligned} \qquad (9)
$$

Top-Down approach: Here, the ERF is calculated for each layer separately. So for a network with n layers, n passes back to the image are needed. In other words, intermediate numbers can not be reused. The process of computing ERF
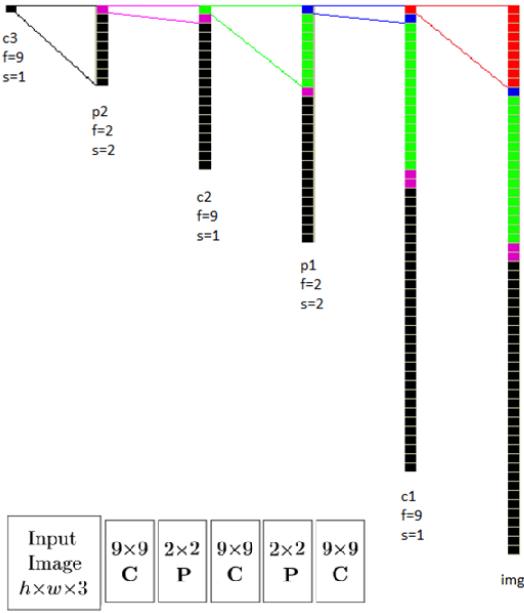
Figure 5. 1 dimensional example illustrating how the top-down approach expands the RF through each lower layer.

for the architecture in Figure 6, according to equation 8, is shown below. $R_{11,0}$ is the ERF of the $11th$ layer in the network. Notice that a separate computation is needed to compute $R_{12,0}$ or $R_{10,0}$.

$$
\begin{aligned}
R_{11,11} &= 1 \\
R_{11,10} &= (1-1)(1) + 11 = 11 \\
R_{11,9} &= (11-1)(1) + 11 = 21 \\
R_{11,8} &= (20-1)(1) + 11 = 31 \\
R_{11,7} &= (31-1)(1) + 9 = 39 \\
R_{11,6} &= (39-1)(1) + 5 = 43 \\
R_{11,5} &= (43-1)(2) + 2 = 86 \\
R_{11,4} &= (86-1)(1) + 9 = 94 \\
R_{11,3} &= (94-1)(2) + 2 = 188 \\
R_{11,2} &= (188-1)(1) + 9 = 196 \\
R_{11,1} &= (196-1)(2) + 2 = 392 \\
R_{11,0} &= (392-1)(1) + 9 = 400
\end{aligned}
\tag{10}
$$

3. Projective field size

In this section, we discuss the calculation of the PF size of a neuron. For the example in Section 1, assuming 10 filters of size 5×5, and stride equal 5, in the first convolutional layer, the PF of each image pixel (i.e., input neuron and in each R, G, or B channels), would be $1 \times 1 \times 10$. Notice that this calculation is independent of the filter size but as we will show below depends on the stride size. Further, notice that as in calculation of the RF, there is a depth component involved as well. For simplicity, we discard the depth in what follows.

The size of the projective field can be calculated by sliding the filter over an area and update the counter of each neuron when it overlaps with the filter (Figure 10, 11). However, sliding the filter is prone to error and difficult to keep track of the value in the x and y directions. A simpler method to determine the projective field was is shown in Figures 8 and 9.

With a stride of 1, the immediate PF is the same size as the filter size of the next layer. For example, if the filter size is 3×3, then a neuron will influence 3×3 nodes in the output filter map. For the above example, assuming 10 filters in the first convolutional layer, the PF of each image pixel (i.e., input neuron and in each R, G, or B channel), would be $3 \times 3 \times 10$. The pixels at the corners and the edges would have slightly smaller PFs. Here, for simplicity we assume that the input image has been zero padded.

With a stride greater than 1, then some neurons will have bigger PFs than other neurons. For example, for a filter size of 5×5 and stride of 2, the center neuron (see figure 8) has the PF of 3×3. The neurons on the x-axis and y-axis of the center neuron would have PFs of 3×2 or 2×3, respectively. The neurons diagonal to the center neurons would have PF sizes of 2×2.

From the above analysis (See Figures 10, 11, and 9), the projective field of a node at layer k of a CNN is bounded with a set of four pairs of values:

$$
P_k = \left\{ \left\lfloor \frac{f_{k+1}}{s_{k+1}} \right\rfloor \times \left\lfloor \frac{f_{k+1}}{s_{k+1}} \right\rfloor, \left\lfloor \frac{f_{k+1}}{s_{k+1}} \right\rfloor \times \left\lceil \frac{f_{k+1}}{s_{k+1}} \right\rceil, \right.
$$
$$
\left. \left\lceil \frac{f_{k+1}}{s_{k+1}} \right\rceil \times \left\lfloor \frac{f_{k+1}}{s_{k+1}} \right\rfloor, \left\lceil \frac{f_{k+1}}{s_{k+1}} \right\rceil \times \left\lceil \frac{f_{k+1}}{s_{k+1}} \right\rceil \right\}
\tag{11}
$$

where f_{k+1} and s_{k+1} are the filter size and stride in the next layer. According to the equation 11, if the remainder of the fraction is zero, then all the nodes have equal projective field sizes. Otherwise, depending on the location of the nodes, projective field sizes would be different. In other words, when the fraction does not yield an integer value, there is disparity in the influence of the nodes in the next layer. This is perhaps why researchers tend not to use strides greater than 1 in convolution layers (or use strides equal the filter size in pooling layers). Nonetheless, it is unclear whether such disparity can cause any practical problems.

Deconv nets are inverted versions of CNNs. Therefore, their projective field can be calculated using the ERF formulas. Similarly, their ERF is the same as the projective field in CNNs.

4. Discussion

Here, we discussed how receptive, effective receptive and projective fields of neurons in CNNs are calculated. Understanding these quantities has important implications
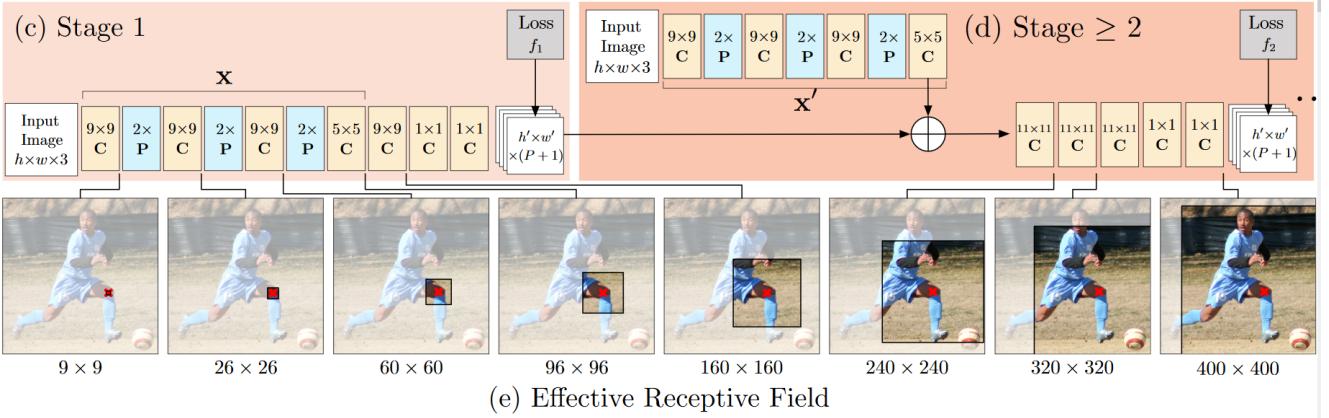
Figure 6. Convolutional Pose Machine by Wei et al. [4] and the ERF of neurons in different layers (Figure taken from [4]).
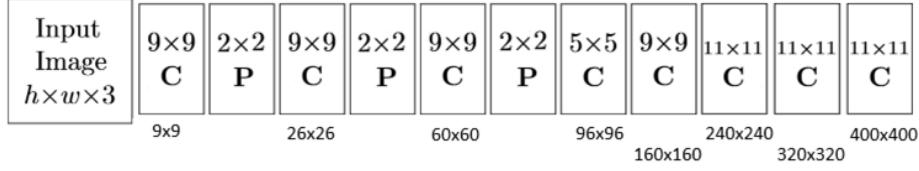

Figure 7. Sample CNN architecture with filter sizes and effective receptive fields shown, reproduced from [4].
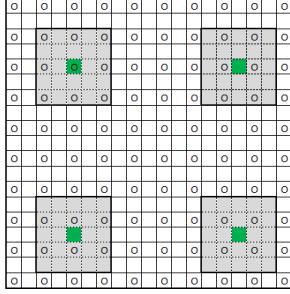

Figure 8. Projective field size. The gray box is the filter with size of 5×5. The small circle is the output node of the filter map with the stride of two. The PF of a neuron (green box) is calculated by counting how many filters (circles) being applied within the bounding box. Depend on the location, the PF of a neuron would be different from other.

Figure 9. Illustration of projective field size calculation. In this image, the blue box is where the filter is being applied. To calculate the PF of a neuron (shown in green), the filter (gray area) is applied around the neuron to determine the PF. For the showed neuron, the PF is 2x2. The same action can be done with other cell. The values can be verified with more complex process showed in Figure 10, and Figure 11.

in deep learning research. First, it helps setting the parameters such as filter size, number of filters, and stride more effectively. Second, it allows analyzing how objects are represented by CNNs and investigate whether some image information is lost along the CNN hierarchy.

5. Acknowledgment

References

[1] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] S. R. Lehky and T. Sejnowski. Network model of shape-from-shading: neural function arises from both receptive and projective fields. *Nature*, 333(6172):452–454, 1988.

[4] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4732, June 2016.
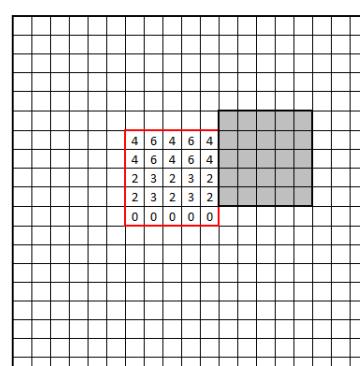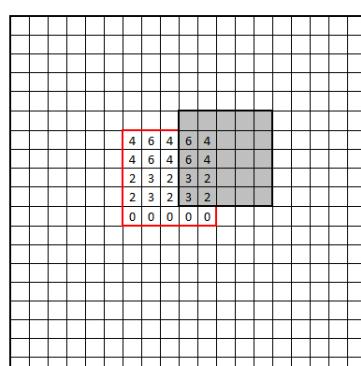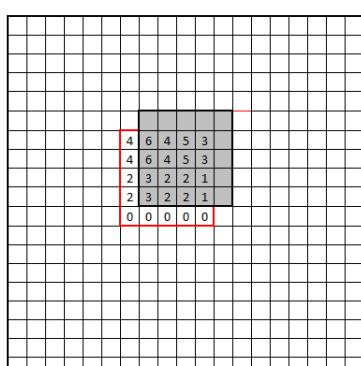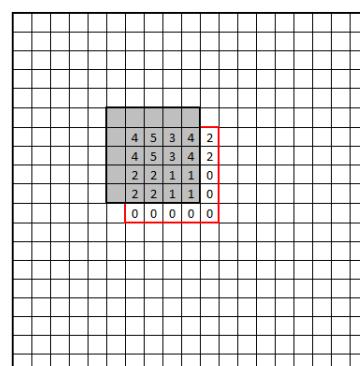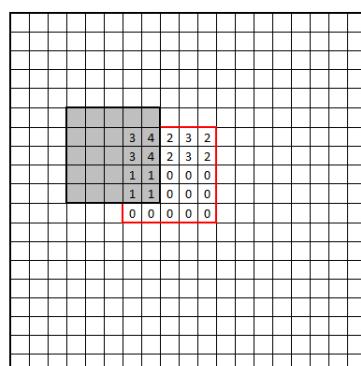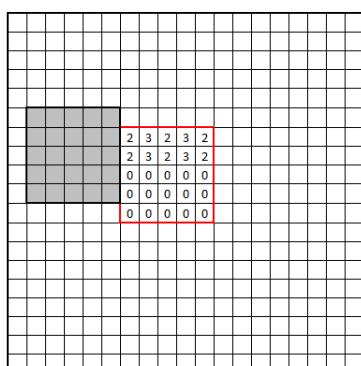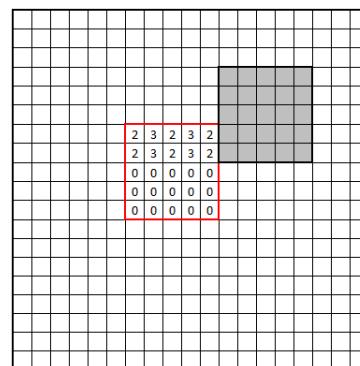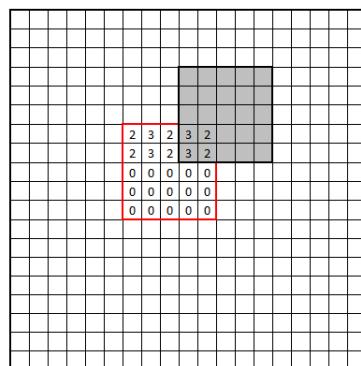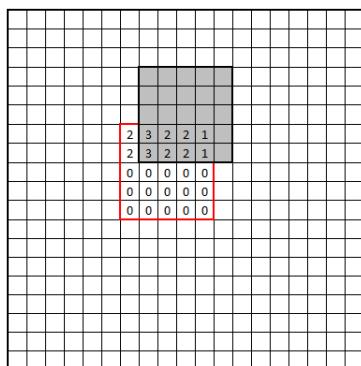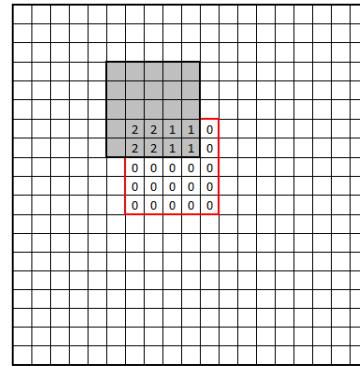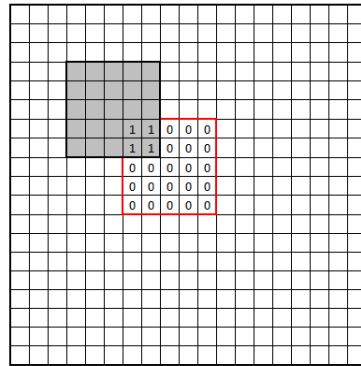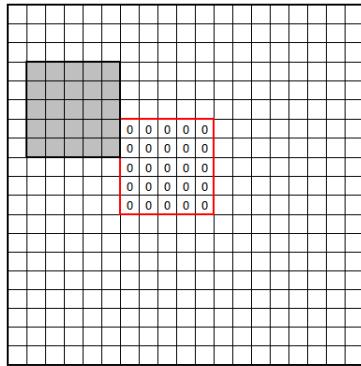
Figure 10. Sequence of sliding a filter for calculating the projective field of neurons. Here filter size is 5×5 and stride is 2.

Figure 11. Sequence of sliding a filter for calculating the projective field of neurons (continued).