# VERIFICATION OF NEURAL NETWORKS FOR OCCLUSION ROBUSTNESS

**Ananya Alekar**
School of Mathematics and Computer Science
Indian Institute of Technology, Goa
Ponda, Goa 403401
ananya.alekar.21031@iitgoa.ac.in

December 4, 2024

**ABSTRACT**

Occlusion is one of many perturbations that often lead neural networks to give incorrect predictions. It is a type of perturbation where a part or all of the object is hidden from view by some other object in the scene. In safety systems such as maintenance systems and surveillance systems, robustness to such perturbations is critical. This work proposes an approach to encode occlusions through neural networks, increasing the efficiency of further robustness verification.

## 1 Introduction

In recent years, neural networks have become integral to a wide range of applications, including safety-critical systems such as surveillance and maintenance. However, their susceptibility to input perturbations, particularly occlusions, poses significant challenges to their robustness and reliability. Occlusion occurs when parts of an input, such as an image, are blocked, either partially or entirely, by other elements in the scene. This can lead to incorrect predictions, severely affecting the reliability of neural networks.

This report presents an approach to efficiently encode occlusions within neural networks, facilitating their robustness verification. Using encoding techniques such as Random Erasing, Bilinear Interpolation, and neural network-based occlusion encoding, this study aims to provide an effective framework for addressing occlusion-related vulnerabilities. The methods are evaluated on benchmark datasets such as MNIST, CIFAR-10, and GTSRB to assess their efficiency in generate occlusions of different size and color schemes. Through these efforts, the study contributes to advancing the robustness of neural networks in handling occlusion perturbations, enhancing their utility in real-world applications.

## 2 Preliminaries

### 2.1 Robustness

A neural network is said to be **robust** if perturbations to its input do not affect the outcome of prediction [1]. Therefore, a network $F$ is said to be robust to a set of perturbations $\Omega$ to its input $x_0$, if the $F$ predicts the same result for $\Omega$ which it did for $x_0$.

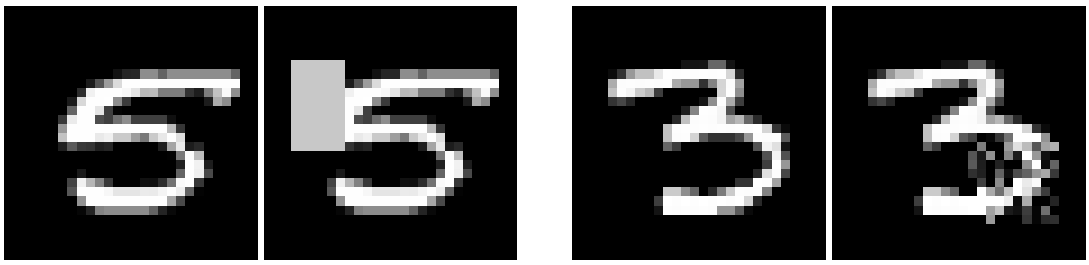
**Local Robustness [2]** A network $F : \mathbb{R}^u \to \mathbb{R}^r$ is said to be locally robust with respect to an input $x_0$ and a set of perturbations $\Omega$ to $x_0$ if

$$\forall x \in \Omega, \phi_F(x_0) = \phi_F(x)$$

where $\phi_F(x)$ is the prediction made by $F$ for an input $x$.

In general, the set $\Omega$ is defined as an $l_p$-norm ball around $x_0$ with radius $\epsilon$ such that

$$\mathbb{B}_p(x_0, \epsilon) := \{x | \, \|x - x_0\| < \epsilon\}$$



(a) Uniform (b) Multiform

Figure 1: Examples of uniform and multiform occlusion

### 2.2 Occlusion

Occlusion is a perturbation where parts of an image are blocked before it is fed to a classification network. An occlusion is defined by its shape, size, color and position. It can take the form of a square, rectangle, triangle or an irregular shape. For the purpose of this study, only rectangular and square shaped occlusions are generated. The size of an occlusion is defined by the number of occluded pixels in the image.

Based on its color, an occlusion can be defined as *(a) uniform*, where all occluded pixels have the same color or *(b) multiform*, where occluded pixel values can vary in the range $[-\epsilon, \epsilon]$, where $\epsilon \in \mathbb{R}$ is the threshold between the original pixel value and occluded pixel value. Fig 1 shows two examples of uniform and multiform occlusion.

An occlusion can be placed directly on the image pixels, superimposing them or between a pixel and its surrounding pixels. Fig 2 shows a representation where black dots represent the image pixels and orange dots represent occluding pixels. An occluding pixel at $(i^{'}, j^{'})$ is said to *surround* an image pixel $(i, j)$ if $|i - i^{'}| < 1$ and $|j - j^{'}| < 1$. Here $i^{'}$ and $j^{'}$ are real numbers and the location of an occluding pixel placed on the image. An image pixel can only be occluded by its surrounding occluding pixels.

As represented in Fig 2, an image pixel can be surrounded by at most four occluding pixels. Let $\mathbb{I}_p$ be the locations of all surrounding pixels of an image pixel, then the value of the image pixel can be computed using bi-linear or nearest neighbor interpolation [3, 4]. However, [5] uses a different interpolation method based on the $L_1$ distance to calculate how much an image pixel can be occluded by its surrounding pixels. Since the distance between two adjacent pixels is 1, an image pixel is not affected by an occluding pixel if the $L_1$ distance between them is greater than 1. The amount an image pixel at location $(i, j)$ is occluded by an occluding pixel at $(i^{'}, j^{'})$ is given by the formula $max(0, (|i-i^{'}+1|+|j-j^{'}+1|)-1)$. Thus the occlusion factor $s_{ij}$ for an image pixel at location $(i, j)$ is given by

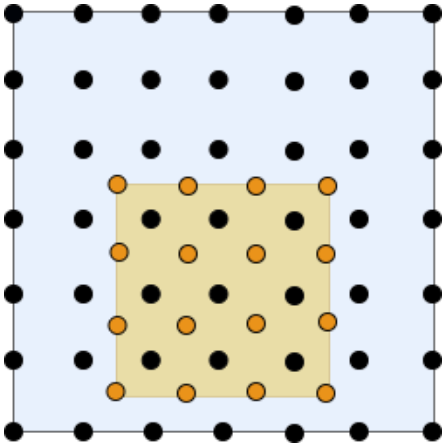


Figure 2: A 4×4 occlusion on a 7×7 image

$$s_{ij} = max(0, \sum_{i^{'} \in \mathbb{I}_p} (|i-i^{'}+1|) + \sum_{j^{'} \in \mathbb{I}_p} (|j-j^{'}+1|) - 1) \tag{1}$$

When the image pixel is completely occluded, $s_{ij}$ is 1. It is 0 when the set $\mathbb{I}_p$ is empty i.e. the image pixel has no surrounding occluding pixels. In all other cases, $s_{ij}$ falls in the interval $(0, 1)$.

**Occlusion function** [5] Let $x$ be an image belonging to $\mathbb{R}^{m \times n}$ which is the set of images with height $m$ and width $n$. A coloring function, $\zeta : \mathbb{R}^{m \times n} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is defined as the mapping of each image pixel to its corresponding color value ranging from $[0, 255]$. Thus, $\zeta(x, i, j)$ is the value of the pixel at $(i, j)$.

Given a coloring function $\zeta$, an occlusion $\vartheta$ of size $w \times h$, the occlusion function $\gamma_{\zeta, w \times h} : \mathbb{R}^{m \times n} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^{m \times n}$ such that, $x^{'} = \gamma_{\zeta, w \times h}(x, a, b)$ where $(a, b)$ is the upper-left corner of the occlusion $\vartheta$, if $\forall i \in [1, n], j \in [1, m]$, there

is

$$x_{ij}^{'} = x_{ij} - s_{ij} \times (x_{ij} - \zeta(x, i, j)) \tag{2}$$

$$\zeta(x, i, j) = \frac{\sum_{(i',j') \in \mathbb{I}_{i,j}} \vartheta_{i',j'} \sqrt{(i - i')^2 + (j - j')^2}}{\sum_{(i',j') \in \mathbb{I}_{i,j}} \sqrt{(i - i')^2 + (j - j')^2}} \tag{3}$$

When $(i^{'}, j^{'})$ is an integer, Equation 3 can be reduced to $x_{ij}^{'} = \vartheta_{i',j'}$. This is because in the integer case, $x_{ij}$ is completely occluded by the occluding pixel and as a result, $s_{ij}$ is 1 and there is only one element in the set $\mathbb{I}_{(i,j)}$ i.e. $(i, j)$ itself. When $s_{ij}$ is 0, $x_{ij}^{'} = x_{ij}$.

**Note** [5] In the case of a uniform occlusion, all occluding pixels have the same value $\mu \in (0, 1)$ and hence the coloring function can be defined as $\zeta(x, i, j) = \mu$. In a multiform occlusion, the coloring function is defined as $\zeta(x, i, j) = x_{ij} + \Delta_p$ with $\Delta_p \in [-\epsilon, \epsilon]$ where $\epsilon \in \mathbb{R}$ is the threshold that a pixel can be altered.

## 3 Occlusion Encoding

There are multiple ways of artificially encoding occlusions. In this study we explore three encoding methods, namely Random Erasing [6], Bilinear Interpolation and a Neural Network encoding [5].

### 3.1 Random Erasing

Random Erasing is a data augmentation technique used to train convolutional neural networks. Given an input image $x$ and the occlusion size $w \times h$, the Random Erasing procedure (Fig 3) randomly selects a rectangle region in the image with probability $p$ and erases its pixels with random values. However, this can only produce uniform occlusions.

**Algorithm 1:** Random Erasing Procedure

**Input :** Input image $I$;
Image size $W$ and $H$;
Area of image $S$;
Erasing probability $p$;
Erasing area ratio range $s_l$ and $s_h$;
Erasing aspect ratio range $r_1$ and $r_2$.
**Output:** Erased image $I^*$.
**Initialization:** $p_1 \leftarrow$ Rand (0, 1).

**if** $p_1 \geq p$ **then**
  $I^* \leftarrow I$;
  **return** $I^*$.
**else**
  **while** *True* **do**
    $S_e \leftarrow$ Rand $(s_l, s_h) \times S$;
    $r_e \leftarrow$ Rand $(r_1, r_2)$;
    $H_e \leftarrow \sqrt{S_e \times r_e}$, $W_e \leftarrow \sqrt{\frac{S_e}{r_e}}$;
    $x_e \leftarrow$ Rand $(0, W)$, $y_e \leftarrow$ Rand $(0, H)$;
    **if** $x_e + W_e \leq W$ *and* $y_e + H_e \leq H$ **then**
      $I_e \leftarrow (x_e, y_e, x_e + W_e, y_e + H_e)$;
      $I(I_e) \leftarrow$ Rand (0, 255);
      $I^* \leftarrow I$;
      **return** $I^*$.
    **end**
  **end**
**end**

Figure 3: The Random Erasing Procedure [6]

For this study, the algorithm in Fig 3 is altered to meet the definitions of multiform occlusions. The threshold value $\epsilon$ is fed as input to the procedure. Further, line 12 in the procedure can be altered as follows:

$$\Delta_p \leftarrow Rand(-\epsilon, \epsilon)$$
$$I(I_e) \leftarrow I(I_e) + \Delta_p$$

However, since the procedure randomly selects a rectangular region (Fig. 4) in the image, the locations of the occluding pixels can only be integers. Thus we proceed to the next method, Bilinear Interpolation.

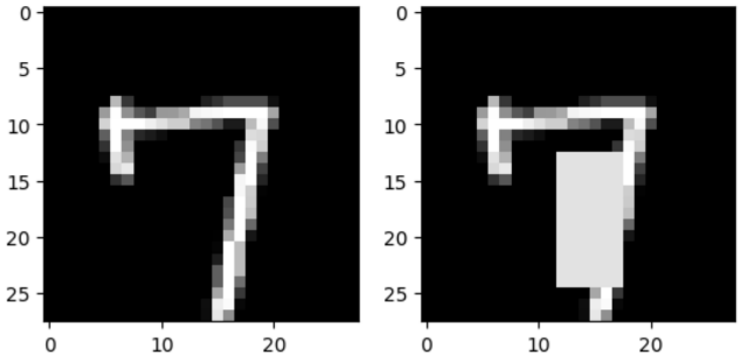
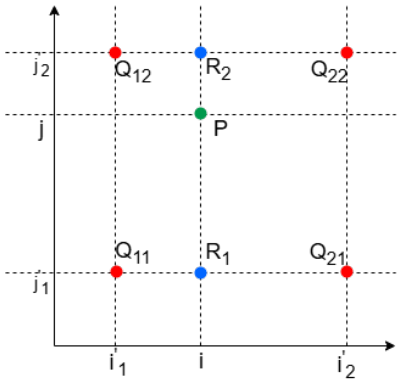


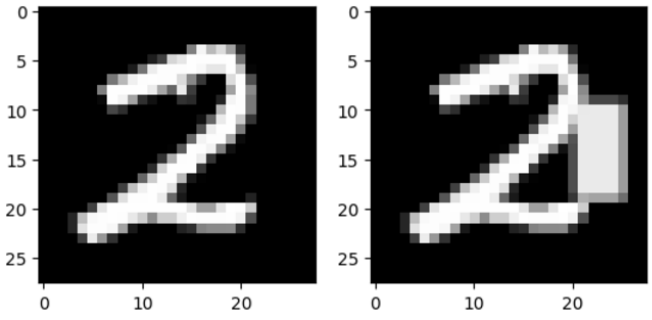

Figure 4: Results of Random Erasing on an MNIST image

### 3.2 Bilinear Interpolation

The coloring function can be defined using bilinear interpolation. As shown in Fig 5b, given the value of pixels at points $Q_{11}, Q_{21}, Q_{22}, Q_{12}$ and the values $i, j, i_1, i_2, j_1, j_2$, we require the value of a image pixel at point $P$. In the context of occlusion, point P can be thought of as the image pixel whose new value needs to be computed and the points $Q_{pq}$ are its surrounding occluding pixels.




(a) Bilinear Interpolation [7]




(b) Results of bilinear interpolation on an MNIST image

The coloring function $\zeta(x, i, j)$ i.e. at point $P$ can be therefore be defined as [7],

$$R_1(i,j) = Q_{11} \cdot \frac{(i_2^{'} - i)}{(i_2^{'} - i_1^{'})} + Q_{21} \cdot \frac{(i - i_1^{'})}{(i_2^{'} - i_1^{'})}$$

$$R_2(i,j) = Q_{12} \cdot \frac{(i_2^{'} - i)}{(i_2^{'} - i_1^{'})} + Q_{22} \cdot \frac{(i - i_1^{'})}{(i_2^{'} - i_1^{'})}$$

$$\zeta(x,i,j) = R_1 \cdot \frac{(j_2^{'} - j)}{(j_2^{'} - j_1^{'})} + R_2 \cdot \frac{(j - j_1^{'})}{(j2^{'} - j_1^{'})} \tag{4}$$

**Algorithm 1** Bilinear Interpolation Procedure

**Input:** Input image $I$; Image shape $m$ and $n$; Occlusion upper left corner $(X_0, Y_0)$; Occlusion Size $W$ and $H$; Occlusion color $\mu$

**Output:** Occluded image $I^{'}$

$X_3, Y_3 \leftarrow min(m, X_0 + W), min(n, Y_0 + H)$
$i, j \leftarrow X_0, Y_0$
**while** $i < X_3$ **do**
  **while** $j < Y_3$ **do**
    $x_1 \leftarrow \lfloor i \rfloor$
    $y_1 \leftarrow \lfloor j \rfloor$
    $x_2 \leftarrow \lceil i \rceil$
    $y_2 \leftarrow \lceil j \rceil$

    $Q_{11} \leftarrow I[x_1, y_1], Q_{12} \leftarrow I[x_1, y_2], Q_{21} \leftarrow I[x_2, y_1], Q_{22} \leftarrow I[x_2, y_2]$

    **if** $x_1 == x_2$ **then** $C_{x_1} \leftarrow 1, C_{x_2} \leftarrow 1$
    **else** $C_{x_1} \leftarrow \frac{(x_2 - i)}{(x_2 - x_1)}, C_{x_2} \leftarrow \frac{(i - x_1)}{(x_2 - x_1)}$
    **end if**

    **if** $y_1 == y_2$ **then** $C_{y_1} \leftarrow 1, C_{y_2} \leftarrow 1$
    **else** $C_{y_1} \leftarrow \frac{(y_2 - i)}{(y_2 - y_1)}, C_{y_2} \leftarrow \frac{(i - y_1)}{(y_2 - y_1)}$
    **end if**

    $C_{x_1 y_1} \leftarrow C_{x_1} \cdot C_{y_1}$,
    $C_{x_1 y_2} \leftarrow C_{x_1} \cdot C_{y_2}$,
    $C_{x_2 y_1} \leftarrow C_{x_2} \cdot C_{y_1}$,
    $C_{x_2 y_2} \leftarrow C_{x_2} \cdot C_{y_2}$

    ▷ In the case of multiform occlusion, the occlusion color $\mu = Q_{ij} + \Delta_p$

    **if** $(x_1 != x_2) \wedge (y_1 != y_2)$ **then**
  $I(x_1, y_1) \leftarrow I(x_1, y_1) - C_{x_1 y_1} \cdot (Q_{11} - \mu)$
  $I(x_1, y_2) \leftarrow I(x_1, y_2) - C_{x_1 y_2} \cdot (Q_{12} - \mu)$
  $I(x_2, y_1) \leftarrow I(x_2, y_2) - C_{x_2 y_1} \cdot (Q_{21} - \mu)$
  $I(x_2, y_2) \leftarrow I(x_2, y_1) - C_{x_2 y_1} \cdot (Q_{22} - \mu)$
    **else if** $(x_1 == x_2) \wedge (y_1 == y_2)$ **then**
  $I(x_1, y_1) \leftarrow I(x_1, y_1) - C_{x_1 y_1} \cdot (Q_{11} - \mu)$
    **else if** $(x_1 == x_2)$ **then**
  $I(x_1, y_1) \leftarrow I(x_1, y_1) - C_{x_1 y_1} \cdot (Q_{11} - \mu)$
  $I(x_1, y_2) \leftarrow I(x_1, y_2) - C_{x_1 y_2} \cdot (Q_{12} - \mu)$
    **else if** $(y_1 == y_2)$ **then**
  $I(x_1, y_1) \leftarrow I(x_1, y_1) - C_{x_1 y_1} \cdot (Q_{11} - \mu)$
  $I(x_2, y_1) \leftarrow I(x_2, y_1) - C_{x_2 y_1} \cdot (Q_{21} - \mu)$
    **end if**
  $j \leftarrow j + 1$
  **end while**
  $i \leftarrow i + 1$
  $j \leftarrow X_0$
**end while**
**return** $I$

### 3.3 Neural Network Encoding

Since the ultimate goal of the experiment is to verify the robustness of neural networks, encoding occlusions using neural networks is a more efficient path, since the output of the occlusion network (Fig. 6) can be directly fed to the network to be verified. Given a coloring function $\zeta$, occlusion size $w \times h$ and input image $x \in \mathbb{R}^{m \times n}$, the occlusion network $O : \mathbb{R}^4 \rightarrow \mathbb{R}^{m \times n}$ [5] encodes all possible occlusions for the image. The network takes the input $(a, w, b, h)$ where $(a, b)$ is the top-left coordinate of the occlusion area.

It should be noted that this is a network with fixed weights and bias. In all layers, the ReLU activation function is used. The following subsections explain the weight and biases used to encode the occlusions.

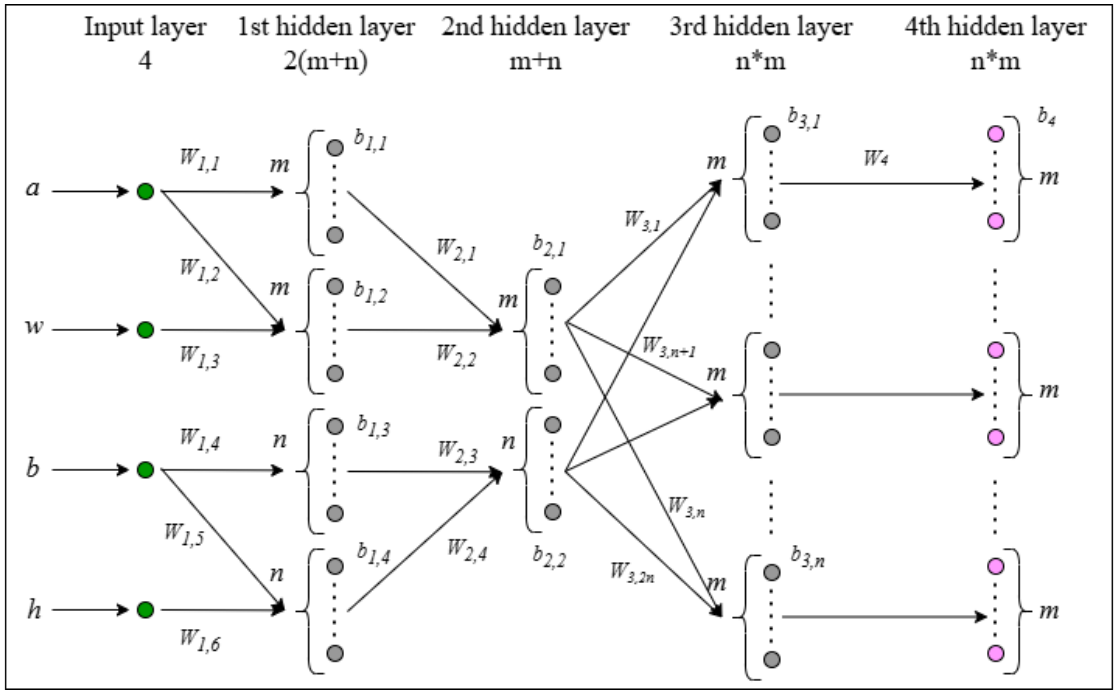



Figure 6: An occlusion neural network for the occlusions on image $x$ with $\zeta$ and $w \times h$

#### 3.3.1 Encoding the occlusion position

The connections between the input layer and first hidden layer, $W_{1,1}, W_{1,2}$ and $W_{1,3}$ with biases $b_{1,1}$ and $b_{1,2}$ are defined as

$$W_{1,1} = \begin{bmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{bmatrix}_{m \times 1}, W_{1,2} = \begin{bmatrix} -1 \\ -1 \\ . \\ . \\ -1 \end{bmatrix}_{m \times 1}, W_{1,3} = \begin{bmatrix} -1 \\ -1 \\ . \\ . \\ -1 \end{bmatrix}_{m \times 1}, b_{1,1} = \begin{bmatrix} -1 \\ -2 \\ . \\ . \\ -m \end{bmatrix}_{m \times 1}, b_{1,2} = \begin{bmatrix} 2 \\ 3 \\ . \\ . \\ m+1 \end{bmatrix}_{m \times 1}$$

The weights $W_{1,4}, W_{1,5}$ and $W_{1,6}$ with biases $b_{1,3}$ and $b_{1,4}$ are defined in the similar way.

In the second layer, the weights $W_{2,1}$ to $W_{2,4}$ are defined as follows

$$W_{2,i} = \begin{bmatrix} -1 & & & & \\ & -1 & & & \\ & & . & & \\ & & & . & \\ & & & & -1 \end{bmatrix}_{m \times m}$$

The biases $b_{2,1}$ and $b_{2,2}$ are set to 1. After propagating to the second layer, a pixel at location $(i, j)$ is occluded if and only if both the outputs of the $i^{th}$ neuron from the first $m$ neurons and the $j^{th}$ neuron from the next $n$ neurons are 1.

The third layer determines the occlusion status of a pixel. It outputs the occlusion factor $s_{ij} \in [0, 1]$ for each pixel. In this layer, each $i^{th}$ group of $m$ neurons is connected to the second layer by the weights $W_{3,i}$ and $W_{3,n+i}$. In $W_{3,i}$, the $i^{th}$ column entries are 1 and rest of the entries are 0. These weights are defined as follows

$$W_{3,i} = \begin{bmatrix} 0 & . & . & 1 & 0 & . & . & 0 \\ 0 & . & . & 1 & 0 & . & . & 0 \\ . & & & . & . & & & . \\ . & & & . & . & & & . \\ 0 & . & . & 1 & 0 & . & . & 0 \end{bmatrix}_{m \times m}, W_{3,n+i} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & . & \\ & & & . \\ & & & 1 \end{bmatrix}_{m \times n}, b_{3,i} = \begin{bmatrix} -1 \\ -1 \\ . \\ . \\ -1 \end{bmatrix}_{1 \times m}$$

After propagating to the third layer, the occlusion factor $s_{ij}$ for the pixel at $(i, j)$ is given by the $(i \times m + j)^{th}$ neuron.

### 3.3.2 Encoding coloring function and input image

The fourth layer encodes the coloring function $\zeta$ and the input image. In the uniform case, $\zeta = \mu$ and the weights $W_4$ are defined as

$$W_4 = \begin{bmatrix} \mu - x_1 & & & & \\ & \mu - x_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \mu - x_{m \times n} \end{bmatrix}_{(mn) \times (mn)}$$

In multiform occlusion, $\zeta = x_i + \Delta_i$, as a result $W_4(i, i) = \zeta - x_i = \Delta_i$

$$W_4 = \begin{bmatrix} \Delta_1 & & & & \\ & \Delta_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \Delta_{mn} \end{bmatrix}_{(mn) \times (mn)}$$

The bias $b_4$ is the flattened vector of the input image.

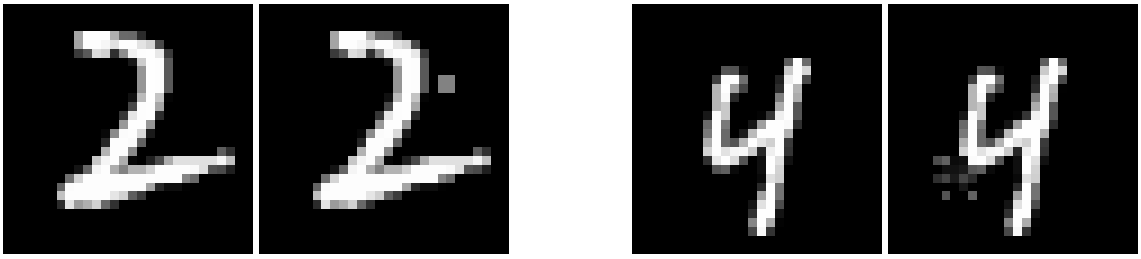
$$b_4 = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{mn} \end{bmatrix}_{mn \times 1}$$

The weights multiplication step in the fourth layer follows the Eq. 4. The layer then outputs the occluded image.

## 4 Results

The above methods were implemented using Python and also using PyTorch for the implementation of the occlusion neural network. The results only depict the performance of the neural network encoding as this is the most efficient approach. The approach was tested in benchmark datasets, including MNIST [8], CIFAR-10 [9] and GTSRB [10] to evaluate the efficiency of this approach in generating occlusions of different shapes and color ranges. The experiment considers two occlusion sizes $2 \times 2$ and $5 \times 5$ with the occlusion threshold $\epsilon$ being from 0.05 to 0.4.

Fig. 7,8 and 9 shows several occlusive adversarial examples that are generated by the neural network encoding under different occlusion settings. These occlusions do not alter the semantics of the original images and should be classified to the same results as those non-occluded ones. However, they are misclassified to other results.



(a) (b)

Figure 7: Results for MNIST images

(a) (b)

Figure 8: Results for CIFAR-10 images

(a) (b)

Figure 9: Results for GTSRB images

## 5 Conclusion

This study explored the challenge of occlusion in neural networks and proposed efficient methods for encoding occlusions to improve robustness verification. Three encoding approaches—Random Erasing, Bilinear Interpolation, and neural network-based encoding—were implemented and evaluated using benchmark datasets such as MNIST, CIFAR-10, and GTSRB. The results demonstrated the efficacy of these methods in generating diverse occlusions. The neural network-based encoding approach, in particular, is the most efficient since it integrates occlusion network with the verification network.

Future work can focus on extending these encoding methods to handle more complex occlusion shapes like parallelograms, triangles and circles. By addressing the vulnerabilities identified in this study, neural networks can become more reliable and robust, generating more trust in their use.

## 6 Acknowledgements

This work was conducted with the guidance and support of Dr. Sudakshina Dutta (Professor, School of Mathematics and Computer Science, IIT Goa) and Ms. Andleeb Zuhra (PhD student, School of Mathematics and Computer Science, IIT Goa).

## References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[2] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks, 2017.

[3] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks, 2016.

[4] Earl J. Kirkland. *Bilinear Interpolation*, pages 261–263. Springer US, Boston, MA, 2010.

[5] Xingwu Guo, Ziwei Zhou, Yueling Zhang, Guy Katz, and Min Zhang. Occrob: Efficient smt-based occlusion robustness verification of deep neural networks, 2023.

[6] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation, 2017.

[7] x engineer. Bilinear interpolation.

[8] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[10] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2013.

$$W_4 \cdot O_3 + b_4 = (\zeta(x, i, j) - x_{ij}) \times s_{ij} + x_{ij} \tag{5}$$