

# Salt and Pepper Noise Estimation in a Digital Image Using Convolutional Neural Network

Carlos Guerrero-Mendez<sup>1</sup>, Tonatiuh Saucedo Anaya<sup>1</sup>,  
Daniela Lopez-Betancur<sup>2</sup>

<sup>1</sup> Universidad Autónoma de Zacatecas,  
Unidad Académica de Ciencia y Tecnología de la Luz y la Materia,  
Mexico

<sup>2</sup> Universidad Politécnica de Aguascalientes,  
Dirección de Posgrado e Investigación,  
México

guerrero\_mendez@uaz.edu.mx, tsaucedo@uaz.edu.mx,  
daniela.betancur@upa.edu.mx

**Abstract.** In this research work, a new technique for estimating impulsive, or salt and pepper noise in digital images is proposed. The estimation of the noise level in an image will serve as a diagnostic, and then facilitate actions to reconstruct an image with salt and pepper noise in digital image processing (i.e., using a filter of correct size). The technique estimates the noise factor in an image using the output of a convolutional neural network (CNN) and a multidimensional linear regression (MLR). The CNN is able to classify the noise factor and provides an output vector, and the MLR is able to use the output vector to estimate the level of noise in the images. For this study, different groups of random images were created with specific noise levels to train the CNN and to fit the MLR. A second dataset with intermediate noise values was used to evaluate the performance of the proposed technique. The proposed method achieves a repeatability of 0.87 and a mean error of 0.71 in salt and pepper noise values in the new images.

**Keywords:** Estimation of impulsive noise, salt and pepper, convolutional neural network.

## 1 Introduction

Digital images are prone to degradations and aberrations due to the occurrence of impulsive noise, also called “salt and pepper”. This type of noise is visualized as random variations in intensity levels in a digital image and appears in the process of sending and receiving information. The causes of noise in an image (in real world) can be due to a malfunction or any damage (existence of dead pixels) in the image sensor or failures in the acquisition, hardware, and transmission [1]. Digital image processing techniques are useful as an easy and efficient way to deal with and remove unwanted noise or aberrations present in digital images. These techniques represent a digital

**Table 1.** Original images dataset. Image name: a) Cameraman, b) Eight, c) Trees, d) Moon, e) Coins, f) Lenna, g) Barbara, h) Mandrill, i) Honeybadger, j) Rice.

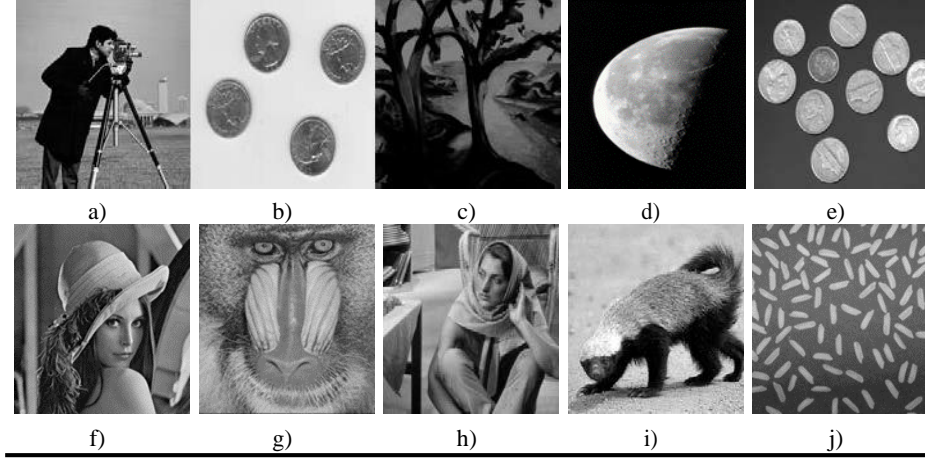


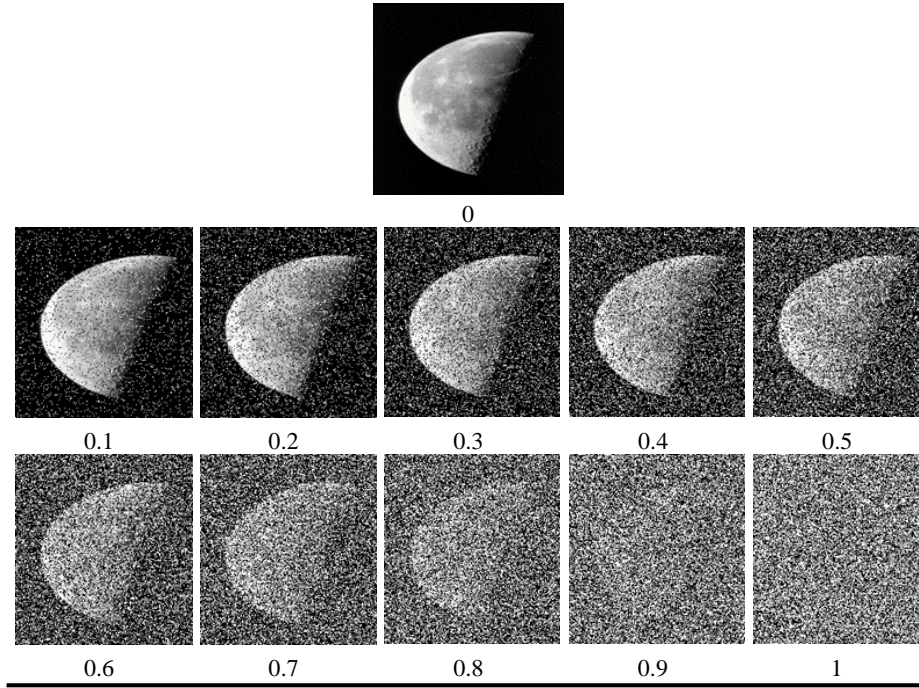
image as a matrix of numerical values and can operate mathematical transformations to treat digital images. There are many methods and techniques to remove impulsive noise in digital images, however, the most common resources to eliminate salt and pepper noise are the Median-type filters [2]

A promising new technology is convolutional neural networks, briefly called CNN. CNN tries to simulate the visual cortex in living beings, though using advanced operation blocks, and several layers of neurons [3]. Popularly, CNNs are used in object classification [4- 7] and detection [8- 11] tasks and semantic segmentation [12]. However, their versatility and power allow them to be implemented in an infinite number of applications [13- 16].

Since digital images can be expressed by numerical values and CNNs can decipher patterns within a large number of numerical values, then it is possible to find and estimate some value of interest within a pattern in an image. Previously, the use of a CNN and a multivariate linear regression (MLR) was introduced by Carlos Guerrero-Mendez et al [17] to estimate the phase difference between digital holograms. Therefore, this research is significant because by applying the correct median filter size to an image, we can reconstruct it correctly and avoid processing an image using too large a median filter, which degrades or removes important information in the image.

## 2 Materials and Methods

This research report focuses mainly on: the generation of data, the training of a CNN using the transfer learning technique, the fitting of an MLR to approximate the noise value, and finally the evaluation of the results obtained.

**Table 2.** Sample images of each class from the training dataset using the "Moon" image.

## 2.1 Image Generation

The training dataset was created from a common set of images used for image processing tasks. The set of seeds or sample images consists of 10 images that can be viewed in Table 1.

## 2.2 Training Dataset

In the training dataset, different noise factors of 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1 salt and pepper were inserted in each image in the seed set. Based on the values of the noise in the images, 11 categories or training classes were created to train the CNN architecture. In total, the training dataset consists of 11,000 images divided into 11 classes. Each class of the training dataset has 1,000 images, of which 100 images are generated using a single noise factor per image sample. Table 2 shows some examples of the training classes using the noise factors in the image "Moon".

### 2.3 Data Augmentation

Training a CNN architecture requires a large number of images and its labels, in the case of supervised learning. Generally, depending on the number of images in the training datasets a CNN architecture can learn and detect more patterns in an image. In addition, overfitting in the CNN model is avoided. The lack of images in the training dataset can be solved using the "data augmentation" technique, which creates new images only for the training process based on some random digital transformations [18]. As part of this research, new images were created using the flipped horizontal transformation.

### 2.4 Validation Dataset

Once the CCN has been trained, the performance for the classification task is validated. In the validation step, similar or new images of the same classes are implemented in the training process to assess the learning of the model. In this research, 250 images were generated for each class of salt and pepper noise factor for the validation dataset. The 10 sample images were used in each class and we added 25 times the same noise factor in each image. So, the total number of images in the validation dataset is 2750.

### 2.5 CNN Architecture

The CNN model used in this research is called AlexNet, which has a simple CNN architecture and is easy to train. The AlexNet model has 60 million parameters and 650,000 neurons. The feature extraction process is done using five convolutional layers, while the classifier process uses three fully-connected layers. AlexNet uses a ReLU. Nonlinearity activation function that allows the training process to be performed much faster than using the activation functions of Tanh and Sigmoid [19]. The output of a CNN is an array of numbers that represent the probabilities that the CNN will assign the input data to an output category. There are probabilities raw values between  $-\infty$  to  $+\infty$  of the input data, called Logits, which can be obtained from the last layer of the CNN before the Softmax layer.

### 2.6 Transfer Learning

Training a neural network from random initial values (weights and bias) is a task that requires a large amount of computational resources and training time. Furthermore, to consider the performance of a neural network as acceptable, it must be trained with a large number of labeled images. There are large databases of labeled images that are used to train neural networks architectures; these databases are usually composed of a large number of common objects, as in ImageNet [20]. When a neural network is trained to classify images in some task, it is possible to use this knowledge acquired by the neural network to classify other types of images in another task, i.e., we use the weights and biases of a neural network to classify a task, and from this acquired knowledge, adjust a second training, with new images, the weights, and biases for a

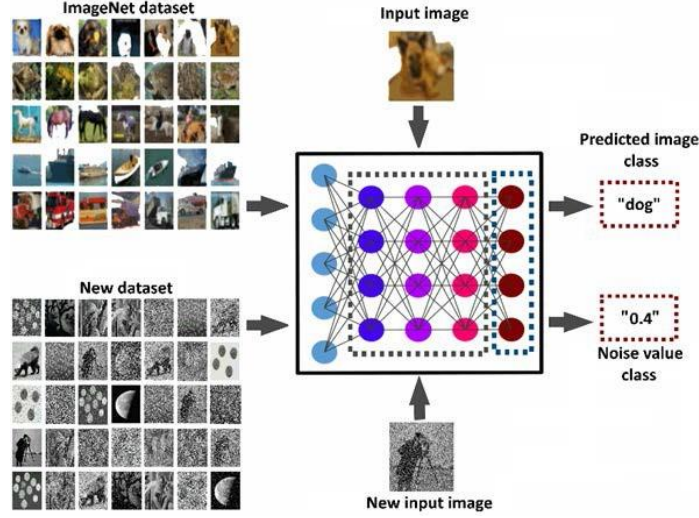


Fig. 1. Transfer learning using new images.

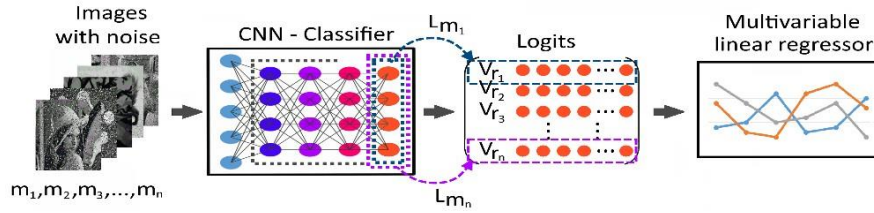
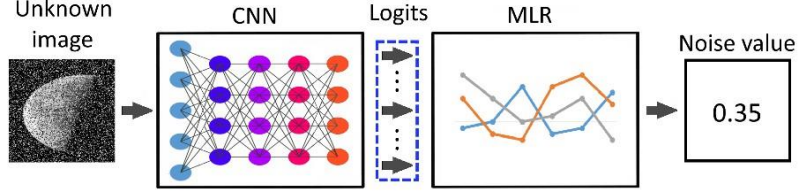


Fig. 2. Diagram of creation of a Logits dataset to fit an MLR.

new classification. In this paper, the AlexNet CNN model, trained with ImageNet, was implemented using transfer learning to classify the noise images (see Fig. 1).

Once the classifier has been trained, the next step is to create a database using the Logits vector extracted from the last layer of the CNN (before the Softmax function) using each image of the training dataset. The new database will contain the logits that will be used to fit the MLR. Fig. 2 shows a diagram of the creation of a dataset with the logits of the images.

The proposed technique uses an algorithm in which we enter an image with an unknown noise value, later, the value of the noise factor in the image will be given as an output. That is, the algorithm receives an image with noise as input, then the image is processed by the CCN, and the Logits vector passes to the MLR as input, and finally, we get the noise factor in the image. This entire process is illustrated in Fig. 3.



**Fig. 3.** Noise estimation in an unknown image.

## 2.7 CNN and MLR Performance Metrics.

One of the most used tools to evaluate the training performance of a CCN is the confusion matrix. Using the confusion matrix, we can graphically observe the results of classifying the images in the validation dataset. In addition, we can also observe where our classifier has classification problems, and change the training options to obtain better results. By implementing the confusion matrix, we can extract some basic parameters or data, and using them we can develop metrics more powerful. The basic parameters are: The true positive (TP) which are the number of predicted cases that belong to a class and that were correctly classified to belong to a class. False positive (FP) are the number of predicted cases that belong to a class, but they really do not belong to the class. True negative (TN) refers to the predicted cases that do not belong to a class and they do not belong to the class. and False Negative (FN) which are the elements classified that do not belong to a class but they really belong to a class. Using TP, FP, TN and FP we can develop the metrics of: accuracy, precision, recall, specificity, f-score. For evaluation of the MLR, the metrics were used: Coefficient of determination ( $R^2$ ), mean absolute error (MAE), mean square error (MSE) [17].

## 3 Experimental Results



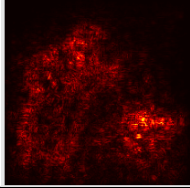
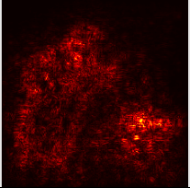
### 3.1 Saliency Maps

A useful tool to analyze the learned features during the training process is the saliency maps [21]. By analyzing the saliency maps of a CNN, we can observe which regions of an image provide a greater degree of activation in the artificial neurons of a CNN model. In other words, a Saliency map shows the areas in an image in which a CNN concentrates to perform a classification.




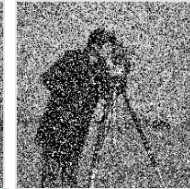
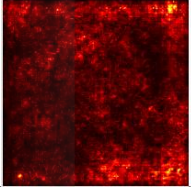
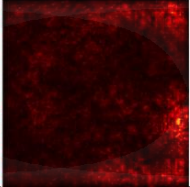
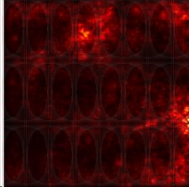
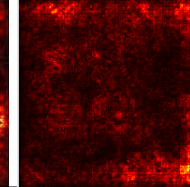
Although the predicted class is not the main interest in this research, it is important to analyze what the CCN classifier defines as important and to understand some of the patterns found in the image to be classified. When analyzing the image samples (clean images), we observe that there is a pattern of activations in the areas where the silhouettes are defined. In Table 3 we can view the activation regions in original images.



**Table 3.** Saliency maps of “Cameraman” images with a noise level of 0.0.

<b>Cameraman image</b>		
<b>Saliency maps</b>		

**Table 4.** Saliency maps of “Cameraman” images with a noise level of 0.4.

<b>Cameraman image</b>				
<b>Saliency maps</b>				

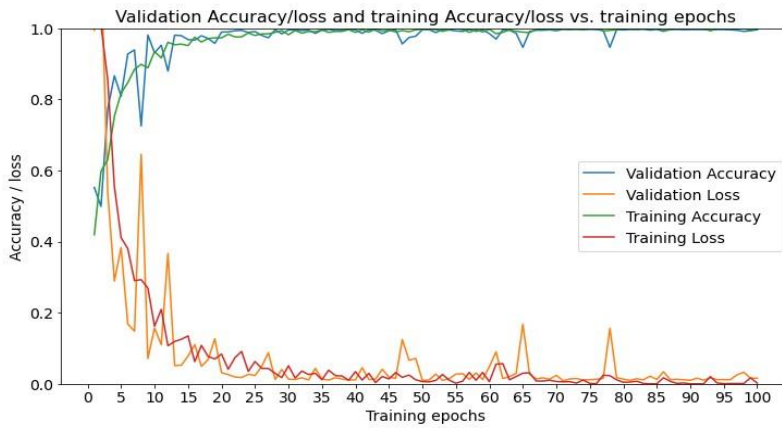
However, if you want to classify the same image where only the noise factor was added (see Table 4), the CNN will not be able to find a single or similar pattern, and it won't even try to find the patterns found in the clean image, but it bases its classification on the use of patterns generated by pixel variations caused by noise in the image.

### 3.2 CNN Training Process

The training process algorithm was executed using the free Google Colab cloud service, which is a research tool for machine learning education and research. Colab allows us to write and execute Python code through a Jupyter notebook environment that runs on servers of Google. One of the great qualities of this cloud service is that we use for 12 hours a virtual machine that has specialized hardware such as a graphics processing unit (GPU). A GPU improves computing performance like matrix multiplication, which is a common and time-consuming computation operation in Deep Learning [22]. In this research, the CNN training was performed on a Tesla T4 GPU,

**Table 5.** Hyperparameters used in the training process.

Hyperparameter	Value
Optimization algorithm	Mini-batch gradient descent
Epochs	100
Learning rate	0.001
Batch size	32

**Fig. 4.** Accuracy/loss evolution in the training process.

also, the Python code was developed using the Pytorch 1.9.0+cu102 library. The CNN model and other tools necessary for CCN training were obtained from Torchvision 0.10.0+cu102. According to the training parameters, we used the Gradient descent as the optimization algorithm and the loss was calculated using CrossEntropyLoss, a total of 100 training epochs were implemented, with a batch size of 32 and a learning rate of 0.001. Table 5 lists the main

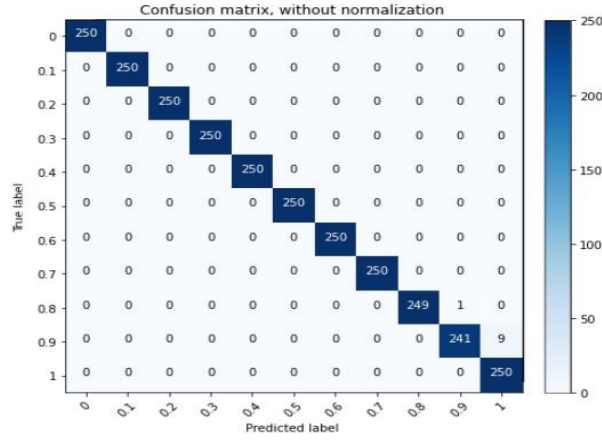
Hyperparameters implemented.

According to the evolution and behavior of the accuracy and loss values, indications of overfitting in the training process are discarded (see Fig 4).

On the other hand, we observe that the confusion matrix has high accuracy to predict the true classes in the training images (see Fig 5).

The CCN training process was evaluated using the validation dataset with the metrics of accuracy, precision, recall, specificity. Using 100 training epochs, it was found that the best training epoch according to its accuracy and loss was epoch 91. Table 6 lists the validation metrics in the training process at epoch 91.





**Fig. 5.** Confusion matrix obtained in the training process.

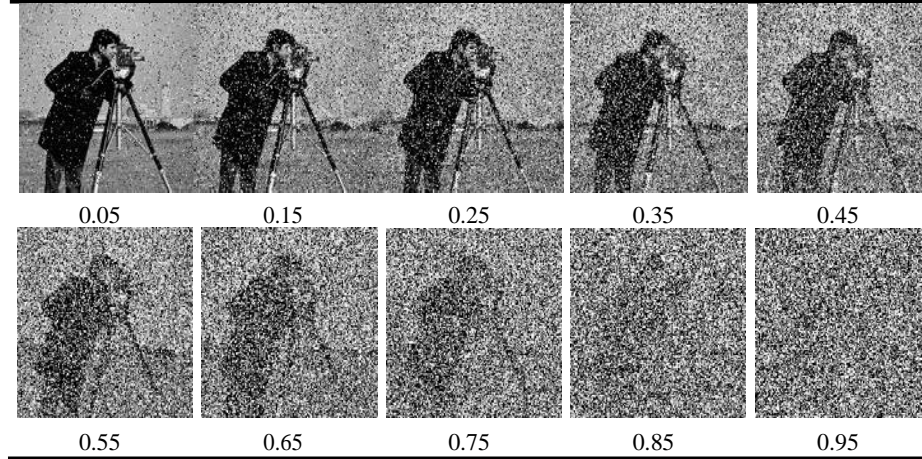
**Table 6.** CNN training validation metrics.

Metric	Value
Accuracy	0.9967
Precision	0.9968
Recall	0.9967
Specificity	0.9997
Training time	116m 50s

### 3.3 Validation of the Noise Estimation Technique

Training a CNN to classify images based on their noise value is a trivial task that does not require much complexity. The use of the training and validation datasets to validate the estimation process may not be entirely appropriate. Using an MLR would make it possible to predict the magnitude of the intermediate noise value in a new image. To validate the operation of the proposed method, classes with intermediate values were created to estimate and compare the noise value in the image. The noise factor values for the intermediate images are: 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95.

As a result, 1,000 images were created to validate the proposed technique. Table 7 shows some examples of the different noise intermediate values applied in the "Cameraman" sample image.

**Table 7.** “Cameraman” with intermediate noise values.**Table 8.** MLR performance metrics.

Metric	Value
$R^2$	0.8703
MAE	0.0716
MSE	0.0106
Standard deviation	0.2631

The values of the performance reached by the proposed technique are shown in Table 8. The values reached demonstrate excellent behavior, and a good repeatability. This indicate that the CNN could be used as technique for image denoising process without the need to implement arduous digital image processing techniques.

## 4 Conclusions

In this article, a novel method for estimating salt and pepper noise in digital images was presented. The noise value is performed using a CNN and a regression model. This method can be the first step in implementing an image denoising process. Given that, knowing the correct amount of noise, an accurate image filter can be applied. According to the results obtained, a CNN can focus on the number and distribution of black and white pixels in an image. In addition, we have demonstrated that the output of a CNN can be used to link to other parameters or aspects of interest through an MLR. The results showed high accuracy in the estimation of salt and pepper noise added to an image, even in images with a noise factor with which the CNN was not trained. As future work, a complete analysis of the use of different CNN models will be developed. In addition, different regression models will be tested and the dropout process will be

analyzed using different values. Furthermore, an analysis is required for the implementation of median filters for an optimal size according to the noise level of the salt and pepper in the input image.

## References

1. Djurović, I.: BM3D Filter in Salt-and-Pepper Noise Removal. *EURASIP Journal on Image and Video Processing*, 13 (2016) doi: 10.1186/s13640-016-0113-x.
2. Liang, H., Li, N., Zhao, S.: Salt and Pepper Noise Removal Method Based on a Detail-Aware Filter. *Symmetry*, 13, pp. 515 (2021) doi: 10.3390/sym13030515.
3. Qin, Z., Yu, F., Liu, C.: How Convolutional Neural Network See the World-A Survey of Convolutional Neural Network Visualization Methods, *arXiv* (2018) doi: 10.48550/arXiv.1804.11191.
4. Khan, A., Sohail, A., Zahoor, U.: A Survey of the Recent Architectures of Deep Convolutional Neural Networks. *Artificial Intelligence Review*, 53, pp. 5455–5516 (2020) doi: 10.1007/s10462-020-09825-6.
5. Lopez-Betancur, D., Bosco Duran, R., Guerrero-Mendez, C.: Comparación de arquitecturas de redes neuronales convolucionales para el diagnóstico de COVID-19. *Computación y Sistemas*, 25 (2021) doi: 10.13053/cys-25-3-3453.
6. Ortiz-Preciado, A.A., Vega-Fernández, J.A., Ochoa-Ruiz, G.: Clasificación de enfermedades del tórax usando aprendizaje profundo y aumento de datos de calidad en el conjunto de datos ChestX-ray8. *RCS*, 148, pp. 41–53 (2019) doi: 10.13053/rcs-148-8-3.
7. Colmenares Guillen, L.E., Torres López, R.G., Carrillo Ruiz, M.: Clasificador de edad en imágenes digitales usando métodos estadísticos. *RCS*. 140, pp. 91–104 (2017) doi: 10.13053/rcs-140-1-8.
8. Liu, L., Ouyang, W., Wang, X.: Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*. 128, pp. 261–318 (2020) doi: 10.1007/s11263-019-01247-4.
9. Zou, Z., Shi, Z., Guo, Y.: Object Detection in 20 years: A Survey, *arXiv* (2019) doi: 10.48550/arXiv.1905.05055
10. González Frayre, G.F., Olvera Olvera, C.A., López Monteagudo, F.E.: Estudio y comparativa de algoritmos de detección de objetos con redes neuronales artificiales convolucionales para la detección de enfermedades en hojas, *Universidad Autónoma de Zacatecas* (2019) doi: 10.48779/p41k-fx69.
11. Takieddine Seddik, M., Kadri, O., Bouarouguene, C.: Detection of Flooding Attack on OBS Network Using ant Colony Optimization and Machine Learning, *Computación y Sistemas*, 25 (2021) doi: 10.13053/cys-25-2-3939.
12. Garcia-Garcia, A., Orts-Escolano, S., Oprea, S.: A Survey on Deep Learning Techniques for Image and Video Semantic Segmentation. *Applied Soft Computing*, 70, pp. 41–65 (2018) doi: 10.1016/j.asoc.2018.05.018.
13. Mamoshina, P., Vieira, A., Putin, E.: Applications of Deep Learning in Biomedicine. *Molecular Pharmaceutics*, 13, pp. 1445–1454 (2016) doi: 10.1021/acs.molpharmaceut.5b00982.

14. Luong, N.C., Hoang, D.T., Gong, S.: Applications of Deep Reinforcement Learning in Communications and Networking: A Survey, *IEEE Communications Surveys & Tutorials*, 21, pp. 3133–3174 (2019) doi: 10.1109/COMST.2019.2916583.
15. Zhao, R., Yan, R., Chen, Z.: Deep Learning and its Applications to Machine Health Monitoring. *Mechanical Systems and Signal Processing*, 115, pp. 213–237 (2019) doi: 10.1016/j.ymssp.2018.05.050.
16. Ortiz-Rodriguez, J.M., Guerrero-Mendez, C., Martinez-Blanco, M. del R.: Breast Cancer Detection by Means of Artificial Neural Networks. *Advanced Applications for Artificial Neural Networks*, pp. 161–179 (2018) doi: 10.5772/intechopen.71256.
17. Guerrero-Mendez, C., Saucedo-Anaya, T., Moreno, I.: Digital Holographic Interferometry without Phase Unwrapping by a Convolutional Neural Network for Concentration Measurements in Liquid Samples. *Applied Sciences*, 10, pp. 4974 (2020) doi: 10.3390/app10144974.
18. Wang, J., Perez, L.: The Effectiveness of Data Augmentation in Image Classification Using Deep Learning. *Convolutional Neural Networks Vis. Recognit*, 11, pp. 1–8 (2017) doi: 10.48550/arXiv.1712.04621.
19. Maeda-Gutierrez, V., Galvan-Tejada, C.E., Zanella-Calzada, L.A.: Comparison of Convolutional Neural Network Architectures for Classification of Tomato Plant Diseases. *Applied Sciences*, 10, pp. 1245 (2020) doi: 10.3390/app10041245.
20. Deng, J., Dong, W., Socher, R.: Imagenet: A Large-Scale Hierarchical Image Database. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255 (2009) doi: 10.1109/CVPR.2009.5206848.
21. Parkhurst, D., Law, K., Niebur, E.: Modeling the Role of Saliency in the Allocation of Overt Visual Attention. *Vision research*, 42, pp. 107–123 (2002) doi: 10.1016/s0042-6989(01)00250-4.
22. Huang, Z., Ma, N., Wang, S.: GPU Computing Performance Analysis on Matrix Multiplication. *The Journal of Engineering*. 2019, pp. 9043–9048 (2019) doi: 10.1049/joe.2018.9178.