



# Convolutional Neural Networks Hyperparameter Tunning for Classifying Firearms on Images

Isaac Cardoza, Juan P. García-Vázquez, Arnoldo Díaz-Ramírez & Verónica Quintero-Rosas

To cite this article: Isaac Cardoza, Juan P. García-Vázquez, Arnoldo Díaz-Ramírez & Verónica Quintero-Rosas (2022) Convolutional Neural Networks Hyperparameter Tunning for Classifying Firearms on Images, Applied Artificial Intelligence, 36:1, 2058165, DOI: [10.1080/08839514.2022.2058165](https://doi.org/10.1080/08839514.2022.2058165)

To link to this article: <https://doi.org/10.1080/08839514.2022.2058165>



© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.



Published online: 04 Apr 2022.



Submit your article to this journal [↗](#)



Article views: 1360



View related articles [↗](#)







View Crossmark data [↗](#)



Citing articles: 2 View citing articles [↗](#)

# Convolutional Neural Networks Hyperparameter Tuning for Classifying Firearms on Images

Isaac Cardoza , Juan P. García-Vázquez , Arnoldo Díaz-Ramírez ,  
and Verónica Quintero-Rosas 

<sup>a</sup>Department of Computer Systems, Tecnológico Nacional de México/IT Mexicali, Mexicali, México;

<sup>b</sup>Facultad de Ingeniería, Universidad Autónoma de Baja California (UABC), MyDCI, Mexicali, México

## ABSTRACT

In recent years, the rates of firearm-related violent acts worldwide have risen significantly. It is a severe social problem that compromises the safety of every individual to some extent. This situation has motivated researchers to find new ways to improve the current state-of-the-art solutions, such as automatic surveillance systems, to detect and classify the presence of firearms within a specific scene. These systems reduce the drawbacks of using direct human supervision. Among the available solutions for the classification task, the performance of Deep Learning models stands out, especially those based on Convolutional Neural Networks. Since they start learning directly from raw data (e.g., images), their learning process can be improved even further by using Transfer Learning techniques. However, the classification accuracy depends significantly on choosing the optimum set of values for the different hyperparameters composing them. Thus, this paper analyses the improvement in the performance of an image-based handgun classification algorithm when tuning its hyperparameters values instead of using its default values. In this work, we evaluated the performance variation using two benchmarks Convolutional Neural Networks architectures: AlexNet and Inception V3. We obtained a maximum accuracy of 94.11% when using the Inception V3 network and transfer learning. We employed Nadam as the optimizer and a learning rate equal to 0.0001, a batch size equal 256, and a total of 13 epochs. Experimental results suggest an essential relationship between the performance of the classification model and the data set, the specific combinations of values for the selected optimizer, the batch size, and the learning rate. The obtained improvement in the accuracy was up to 10.33% after the tuning process.

## ARTICLE HISTORY

Received 11 October 2021

Revised 15 February 2022

Accepted 16 March 2022

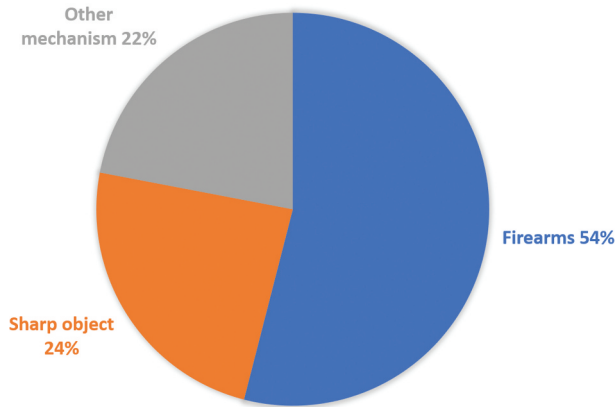
## Introduction

Firearms play an unquestionable role in the occurrence of numerous kinds of violent crimes worldwide, including theft and homicide. According to the Global Study on Homicide UNODC (2019), from the 464,000 estimated total homicides in 2017, roughly 54% (238,804) of the cases involved the utilization

**CONTACT** Arnoldo Díaz-Ramírez.  [adiaz@itmexicali.edu.mx](mailto:adiaz@itmexicali.edu.mx)  Tecnológico Nacional de México/IT Mexicali, Department of Computer Systems, Mexicali, México, 21376

© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



**Figure 1.** Estimated breakdown of intentional homicide worldwide, by a mechanism of perpetration in 2017 UNODC (2019)

of firearms, as it can be seen in [Figure 1](#). Particularly in the Americas, the rate of firearm-related homicides raised to approximately 75%, which is higher than anywhere else in the world UNODC (2019).

Another fundamental indicator of the magnitude of the firearm-related violence problem is the number of firearm seizures by world region. From the UNODC Global Study on Firearms Trafficking 2020 UNODC (2020), it can be concluded that there is a relationship between the number of firearms seized to the number of homicides committed using them. Higher rates of firearm seizures can be observed in countries with low homicide rates, while low rates of firearms seizures are observed in countries experiencing high levels of homicide. This relationship suggests that tighter levels of vigilance over firearms help low homicide levels. In contrast, an increase in the possession rate of weapons corresponds to a rise in the homicide rate. The study also shows that the main types of seized firearms in the Americas are handguns (pistols and revolvers). Particularly in Mexico, violence has risen dramatically over the past few years. In 2018, 36,685 homicides were committed, according to the National Institute of Statistics and Geography (INEGI) INEGI (2019b). It represents a growth of 161% compared to the past decade. In addition, INEGI statistics showed that from the 18.9 million estimated crimes where the victim was present at the moment of the crime, the perpetrator(s) possessed some type of firearm in 32.2% of the cases. The statistic is alarmingly high considering the threat level those artifacts represent. Providing more significant control over firearm usage is a crucial factor in keeping low crimes and homicide rates. Among the possible solutions to address this problem, we may deploy surveillance systems, particularly in urban areas with a high incidence of crimes. Firearms detection systems are attractive for smart cities since statistics data show that metropolitan areas are hot spots for gun violence INEGI (2019a). However, surveillance systems

often have many limitations, including the need for constant human supervision over vast volumes of data. Besides, these systems can be expensive and unfeasible for large-scale deployment. An attractive alternative is the development of automated surveillance systems, where potential criminal activities are detected using artificial intelligence techniques, such as machine learning algorithms. In recent years, Convolutional Neural Networks (CNN) have shown promising results in image classification tasks, outperforming humans Galab et al. (2020). CNN have already been used to address handgun classification with good results Olmos, Tabik, and Herrera (2018a); Kanehisa and Neto (2019); Redmon and Farhadi (2017); Galab et al. (2020). Nevertheless, the existing models still can be improved since most of the proposals for weapons classification based on CNN do not consider one of their most fundamental aspects: their hyperparameters and their tuning process. Different data sets often require different sets of values for the CNN's hyperparameters to yield a good accuracy when classifying images. But the large number of hyperparameters to choose from makes it difficult to decide which ones to prioritize when tuning a classification model. In this process, we first need to select the most determinant hyperparameters to develop a high-performance classifier. Next, we need to adjust its values and evaluate how they behave and correlate with the other hyperparameters. Since this process is exhaustive, time-consuming, and potentially computationally expensive, many developers opt for using only the default values for the training of their CNN. The main reason is that those values are usually set for an ideal performance for general problems. However, for most classification problems, there is no answer to how many network layers provide a better performance, how many neurons per layer are the best, or which optimizer suits the best for every data set and problem. That is why the tuning process results crucial in finding the best possible sets of values for hyperparameters to build an optimum model from a specific data set and problem.

Machine learning algorithms have two kinds of hyperparameters: the model parameters, which are learned directly from the data, and the model hyperparameters, used to control the learning process. Determining the optimal hyperparameters values is imperative to achieve a high ML model performance. It is known as hyperparameter tuning to the task to select the optimal hyperparameters values. Several strategies for hyper-parameter tuning exist. Some of them use automatic optimization techniques Victoria and Maragatham (2021); Kolar et al. (2021), but they show different strengths and drawbacks when applied to different types of problems, such as costly objective function evaluations and complex search space, among others Yang and Shami (2020). For instance, although Bayesian optimization is potentially efficient, is not guaranteed to find better hyperparameters and can get stuck in a local minimum of the objective function Albelwi and Mahmood (2017). We

decided to employ the most used grid-search strategy in this work, which consists of selecting different hyperparameters values and trying all the possible configurations.

Considering the complexity of determining the best values for hyperparameters in a CNN, in this paper, we present an analysis of the effect of the hyperparameters on the performance of a CNN-based handgun classifier. The main contribution of this work is that it explores the efficiency of firearms image classifiers and provides a clue of which combinations of the values of the hyperparameters offer better results. The obtained results give a guide for configuring CNN-based image classifiers to get a greater accuracy using values other than the default ones.

The rest of the document is organized as follows. [Section 2](#) discusses the related work. In [section 3](#), we introduce the materials and methods used in this work. [Section 4](#) introduces the design of the experiments. In [section 5](#), we discuss the results of the analysis of the tuning process of the values of the hyperparameters. [Section 6](#) is for the discussion of the obtained results. Finally, in [section 7](#), the conclusions and future work are presented.

## Related Work

The application of convolutional network architectures has been considered by the scientific community to solve computational vision problems in various fields, such as health Sarvamangala and Kulkarni (2021); agriculture Kamilaris and Prenafeta-Boldú (2018); Pérez-Pérez, Pablo García Vázquez, and Salomón-Torres (2021); biology Tang et al. (2019); transport Rao et al. (2019). Regarding problems related to security in cities, in recent years, computational models have been proposed that allow firearms to be detected and classified in images or videos Kaya, Tuncer, and Baran (2021); Ağdaş, Türkoğlu, and Gülseçen (2021); Dwivedi, Kumar Singh, and Singh Kushwaha (2019); Veranyurt and Okan Sakar (2020); Olmos, Tabik, and Herrera (2018b); Elmir, Ahmed Laouar, and Hamdaoui (2019); Ağdaş, Türkoğlu, and Gülseçen (2021). The main goal is the development of intelligent systems that can prevent the occurrence of violent events.

In [Table 1](#), we present a comparison of the most recent studies that use CNN architectures to detect firearms. It can be observed that those studies considered the variation of one or two more hyperparameters in the evaluation of the performance of the CNN architectures; or applied the transfer of learning as a feature extraction phase. They present models with good performance to classify firearms, some of them with accuracies greater than 95% accuracy Ağdaş, Türkoğlu, and Gülseçen (2021) Dwivedi, Kumar Singh, and Singh Kushwaha (2019) Veranyurt and Okan Sakar (2020). However, most of the works focus on the VGG-16 architecture; so far, we have not identified a study exploring the variation of architectures and hyperparameters related to



**Table 1.** Works that propose CNN architectures for detecting fire guns in images.

Reference	Gun Type	No. of Images	CNN Arch.	Transfer Learning	Hyper parameters	Accuracy
Kaya, Tuncer, and Baran (2021)	assault rifles bazookas, grenades hunting rifles, knives, pistols and revolvers	5214 weapon images from the internet	VGGNet, ResNet-50	No	ReLU activation function 32 mini batch size	<b>VGGNet 98.40%</b> VGG-16
Ağdaş, Türkoğlu, and Gülseçen (2021)	gun and knife images	16000 images containing 9500 knives, 3500 guns, and 3000 ordinary pictures Images from the Internet	AlexNet VGG16, and VGG19	Yes fine tuning	0.25 dropout Adamax optimization alg. 30 epochs Mini-batch size 8–16 Maximum epoch number 5– 20 Weight decay factor 0.01 Initial learning rate 0.0001 Optimization method SGDM (Stochastic Gradient Descent with Momentum) Model A dropout 0.5	89.75% ResNet-101 83.33% Alexnet 97.74 <b>VGG16 99.38</b> VGG19 99.27
Dwivedi, Kumar Singh, and Singh Kushwaha (2019)	gun and knife	Training set contains 1520, 1800 and 1176 images of knives, guns and no-weapons, respectively. Test set contains 344, 368 and 296 images of same classes as the train set Gun Data Set has 3000 images	VGG-16	Yes, fine tuning	Model A dropout 0.5 Model B dropout 0.7	Model A obtains <b>98.41%</b> accuracy (acc.)
Veranyurt and Okan Sakar (2020)	images with and without a hand-gun	Film Gun Data Set has 333 images Coco 2017 Validation Data Set	Basic CNN from scratch and with transfer learning VGG-16 from	Yes, fine tuning	Optimization method SGDM. 12 epochs	VGG16 with transfer learning 88% of accuracy. VGG16 with transfer learning

(Continued)

**Table 1.** (Continued).

Reference	Gun Type	No. of Images	CNN Arch.	Transfer Learning		Hyper parameters	Accuracy
				CNN Arch.	Learning		
Olmos, Tabik, and Herrera (2018a)	pistols, machine guns, rifle,	3000 images of guns taken from a variety	VGG-16	scratch and transfer meaning	Yes, fine tuning	Optimization method SGD	The best obtained accuracy
(Stochastic Gradient Descent)	grenade, rocket launcher, tank	of contexts. Authors created four databases ImageNet for Transfer Learning				(Stochastic Gradient Descent) with back-propagation	was <b>94.44%</b>
Elmir, Ahmed Laouar, and Hamdaoui (2019)	handgun	Gun Data Set consist of 3000 images	MobileNet CNN basic Fast R-CNN		No	Not specified	MobileNet <b>90%</b> acc. CNN basic 55% acc. Fast R-CNN 80% acc.

training, such as learning rate and the number of epochs batch size, and optimizer. In addition, the generated models are trained using a few images related to handguns, which according to the statistics of violent events during the last 50 years, are those that are commonly used in crimes such as murders, assaults, and kidnappings Zimring (2020). In contrast, we used a larger and richer dataset in this work and evaluated more CNN architectures.

## Materials and Methods

### Convolutional Neural Networks

A CNN is a kind of Deep Neural Network specifically designed for image recognition Zaccone and Karim (2018). In a CNN, every input image is represented as a three-dimensional matrix of pixels, consisting of the red (R), green (G) and blue (B) colors, respectively. Every pixel is represented by a tuple of three 8-bit numbers, representing the R, G and B colors. Each image used by the CNN is processed by hidden layers, consisting of convolutional layers, rectified linear units, pooling layers, and fully connected layers. Figure 2 shows the basic architecture of a CNN and its components, which are described as follows:

#### Convolution Layer

It is a special type of layer where each neuron connects to a certain region of the input area called the receptive field Zaccone and Karim (2018). Convolutional layers use several kernel filters of different dimensions on the same receptive fields to recognize images from a different feature. The set of neurons identifying the same feature defines a single feature map.

#### Pooling Layer

A pooling layer consolidates the features learned by the feature map from the previous convolutional layer. It divides a convolutional region into subregions and then selects a single representative value to reduce the computational time of subsequent layers and increase the robustness of the feature concerning its spatial position Zaccone and Karim (2018).

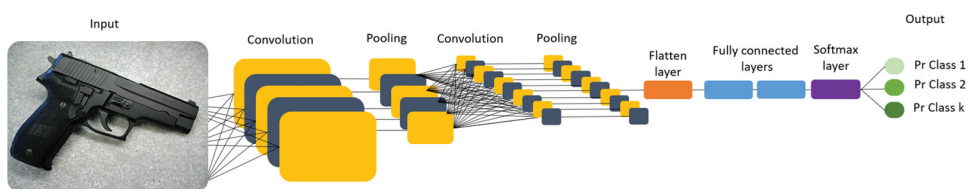


Figure 2. Basic architecture of a CNN.



### **ReLU**

The Rectified Linear Unit is a non-linear function that returns 0 if it receives a negative input, but it returns the same input value if it is positive Alzubaidi et al. (2021). It is the default activation function for many neural networks since its models are easier to train and often yield better performance.

### **Fully Connected Layers**

The fully connected layers form the last few layers in a CNN Aghdam and Heravi (2017). Their input is the output from the final pooling or convolutional layer, which is flattened and then fed into it to perform the mathematical operations, which once passed to the softmax layer will determine the probability for each class.

### **Softmax Layer**

The softmax layer is an output layer for multi-classification tasks operating in conjunction with the cross-entropy loss function. It normalizes the outputs of the previous layer, so they sum up to one. This way, the output could represent the probability for every class Ketkar (2017).

### **Loss Function**

It is the function utilized to evaluate a solution (model). Focused on minimizing the error (loss) and denoted by the difference between the output and the ground truth for a single input Zhao et al. (2017). The loss function to use will depend on the nature of the problem, being the most common mean squared error, binary, categorical, and sparse categorical cross-entropy.

## **CNN Architectures Used in This Work**

In this subsection, a description of the CNN used in this work are briefly discussed.

### **AlexNet**

Its architecture is composed of eight layers with trainable parameters Alex, Sutskever, and Hinton (2017). Five of them are consecutive convolutional layers at the start and three fully connected layers at the end. Every convolutional layer can include a ReLU function and max-pooling optionally. All pooling layers have a  $3 \times 3$  extension region and a step rate of 2. AlexNet requires input images of size  $227 \times 227$ . Summarizing our implementation of AlexNet, it has a total of 200,114,946 parameters, from which 200,112,194 are trainable and 2,752 are non-trainable.

### ***GoogLeNet (Inception V3)***

It is 42 layers deep, and instead of using the  $5 \times 5$  filter, it uses two  $3 \times 3$  in the first inception module Szegedy et al. (2016). In the same way, the other two inception modules also reduce its number of parameters, making it less likely to face an overfitting scenario Smith (2018) and allowing the network to grow deeper than its earlier versions while maintaining most of its features. Inception V3 requires input images of size  $229 \times 229$ .

### ***Transfer Learning***

Transfer learning is an ML technique in which a pre-trained model is reused in a new machine learning model Zaccane and Karim (2018). The advantage of having two models performing similar tasks is that generalized knowledge can be shared between them. Since training new machine learning models can be resource-intensive, the use of transfer learning saves both resources and time, allowing improved performance when modeling the second task. Transfer learning is used in deep learning due to the huge amount of resources required to train deep learning models or the big datasets on which deep learning models are trained. For instance, because computer vision and natural language processing tasks, like image recognition and sentiment analysis, require considerable computational resources, transfer learning is widely used to solve those kinds of problems. It is important to note that transfer learning only works in deep learning if the model features learned from the first task are general. Transfer learning offers several advantages, such as reduced training time, improved neural network performance, or not needing a large amount of data. In this work, we employ transfer learning to speed the training time and improve performance. The Inception V3 structure was modified with the intention of applying transfer learning from the ImageNet data set. An extra trainable layer with unfrozen weights before the last one was added. Thus, summarizing our implementation of Inception V3, we used 43 layers and a total of 23,851,784 parameters. From them, 23,817,352 are trainable and 34,432 are non-trainable.

### ***Hyperparameters and Values***

Hyperparameters are often defined as the configuration of a neural network structure Bochini, Senst, and Sikora (2017). Examples are the number of layers in the network, the number of neurons in each layer, and the activation function. However, the CNN structure is not the only configurable factor. We can also consider as hyperparameters those configurable parameters which define the learning algorithm as it is the case of the optimizer, learning rate, batch size, and the number of epochs. In the same way, when working with CNN, those configurable parameters, particularly of the convolution layers, as

well as stride and padding, can also be considered as hyperparameters. For our experiments, it was necessary to define *a priori* which hyperparameters to configure in order to measure the impact of every one of them in the performance of the generated model. In order of importance, they are:

### **Optimizer**

It is responsible for updating the weights of neurons within a neural network in order to reach the minimum loss function Zewen et al. (2021).

### **Learning Rate**

It represents the magnitude of the step taken on each iteration before updating the weights of the network Alzubaidi et al. (2021). Defining a higher learning rate makes the model learn faster than when using a lower learning rate, but it may miss the minimum loss function and only reach the surroundings of it. On the other hand, a lower learning rate gives a better chance of finding a minimum loss function and therefore may result in a local optimum; however, it needs a higher number of epochs as a trade-off, which ultimately means more time and computational cost.

### **Batch Size**

The batch size is the number of samples that are passed to the network at one time, i.e., on each epoch Feurer and Hutter (2019). Depending on the available computational resources, a larger batch size will speed up the training process. For instance, the use of GPU will allow parallelism. However, larger batch sizes lead to poor generalization, causing the model to not usually achieve high accuracy Kandel and Castelli (2020). In contrast, a small batch size does not guarantee that the model will converge to the global optima.

### **Number of Epochs**

The number of times that the entire training data set is shown to the network during the training process. One epoch means that the training dataset is passed forward and backward through the neural network once Bochinski, Senst, and Sikora (2017). A too-small number of epochs could result in an underfitting model because the neural network has not learned enough to solve the problem efficiently. However, specifying too many epochs could also represent a problem due to the possibility of overfitting, where the model learns from the noise and inaccurate data in the training data set. Consequently, the model cannot generalize accurately on new data and cannot classify new unseen data. That is why the number of epochs must be tuned to gain the optimal result.

The default value for the hyperparameter learning rate is 0.001 for all the available optimizers in Keras (2020b). The exception is the stochastic gradient descent optimizer (SGD), which default value for the learning rate is 0.01. In

the same way, Keras uses a default value for the batch size of 32 Keras (2020a). However, since there is no default value concerning the number of epochs, we defined a standard batch size value of 10 due to the large number of different combinations to test. In our experiments, the ranges considered for each hyperparameter were:

- **Optimizers:** Adadelta, Adagrad, Adam, Adamax, Follow the regularized leader (Ftrl), Nadam, Root Mean Squared propagation (RMSprop), and Stochastic Gradient Descent (SGD).
- **Batch Sizes:** 32, 64, 128 and 256.
- **Learning Rate:** 0.01, 0.001 and 0.0001.
- **Number of Epochs:** 10 and 200.

## Design of Experiments

As mentioned previously, the goal of our experiments is to evaluate the performance AlexNet and GoogLeNet (Inception V3) with transfer learning by changing the hyperparameters associated with the structure and learning. The methodology used in this work consisted of two phases. In the first one, we used the default values for each optimizer on both networks. The goal was to evaluate which hyperparameters have the more significant impact on the performance of the classification model. To accomplish the goal, we conducted a grid-search strategy selecting different hyperparameters values and trying all the possible configurations to determine the hyperparameters values that improve the performance of the model. Thus, we discarded the optimizers whose performance was below the average (Adam, FTRL, and RMSProp), to subsequently explore the impact of both the learning rate and batch size values in the performance of the models. To this extent, we increased the default learning rate tenth fold (0.01) and tested it in every optimizer, each one with a batch size of values 32, 64, 128, and 256, in both networks. Next, we did the same while decreasing the default learning rate tenth fold as well (0.0001). It is important to mention that during this phase, all values for epochs were equal to 10 since we tried to reduce the time spent on each experiment.

In the second phase, we averaged the obtained results to discard every hyperparameter configuration whose results were below the average performance by the learning rate used. Also, we discarded the hyperparameters when the score obtained by the default optimizer values was below the average performance. In this phase, we also tested the default hyperparameters with an epoch value equals to 200 in all configurations to generate a new and more accurate threshold, considering the number of epochs. With this approach, we wanted to explore how the tuning of the hyperparameter values improves the performance, and not that the improvement was obtained only by increasing the number of epochs.

Finally, we discarded those values that caused a performance drop while increasing the epochs and categorized the rest of the results by the best combinations per optimizer. Upon obtaining the higher score of the overall combinations, we tuned the number of epochs even further by running several approximation tests until we reached the best performance.

### ***Data Set and Experiment Setup***

Another key aspect in constructing a highly accurate CNN-based classification model is the image data set employed. For an efficient handgun image classification model, it is important to consider not only images of static weapons but also scenarios of real-life where those firearms were being held by humans in different threatening stances. That is why, looking for a balance between consistency and production environment usefulness, we decided to form our own data set using only some images from other existent data sets. For instance, we included the Internet standard "Gun-Detection" data set Annamraju (2019), from which we took 3,000 images for the true class. Then, we added 2,973 more images from Roboflow.com "Object-Detection/Pistols" data set Webpage (2020). And once collected, the data set was verified image by image taking care to not include any image with watermarks, video game or animated images, toy guns, photo filters, thick frames, or any other kind of visual disturbance to keep it as clean and useful as possible. During this curating process, more than 50% of the images collected were discarded. As a consequence, to reach the desired quantity of images for the true class (3,000), we filled the rest with the results of the istockphoto.com images database using the search query "gun pointed" iStock webpage (2021). As for the false class, we needed a variety of scenes to provide the model with the ability to generalize the background. Also, we needed large quantities of pictures of people holding different kinds of objects to make sure that the model would not learn that people were the main feature for the true class, but guns. Under that criterion, we selected approximately 2,400 images from the Microsoft Common objects in context (COCO) data set Lin et al. (2014), whereas the rest were obtained while doing manual searches in Google Images. Examples of the contents of both classes within our data set can be observed in [Figure 3](#).

Our data set consists of 6,000 total images. Three thousand of them belong to the "true" class, and the other 3,000 to the "false" class. From the total set of images, 70% of them were used as training data, 15% for validation, and 15% for test purposes. The file format for all these images is .jpg, and they come in a variety of sizes ranging from 15 Kb to 5,000 Kb.



**Figure 3.** Examples of the contents of our data set for both categories.

All of our experiments were performed using a Personal Computer using a CPU: Intel(R) Core(TM) i7-10700 F CPU @ 2.90 GHz processor; a GPU NVIDIA GeForce GTX 1660 SUPER GPU @ 1.785 GHz 6.00 GB card; RAM: 31.9 GB Utilizable and Hard disk: 476 GB Utilizable. As for the software specifications, we employed: Windows 10 Home as Operative System; NVIDIA CUDA version 11.0.2\_451.48 with cuDNN 8.0.5 library as GPU-accelerators; Anaconda 3 as environment and package manager; Jupyter as frontend, with Keras 2.4.0 and Tensorflow 2.4.0 as backend. All of our code was written in Python 3.8.8.

## Results

After concluding with the experiments, we obtained enough proof to support our hypothesis that the default hyperparameters values do not always provide the optimum performance to a very specific classification problem. We also found that some hyperparameters alone have an impact on the performance of the classifier, whereas in some cases, the performance improvement is caused by the relationship among the tuning of several hyperparameters.

In [Table 2](#), we can observe the results obtained from the evaluation with the default values for 200 epochs, and in [Table 3](#) we can observe the results obtained with tuned hyperparameters by optimizer and its respective accuracy gain, where it applies.

Through the experiments, we were able to confirm that, for the handgun classification problem, the hyperparameters tuning process is indeed critical. Since the best result was not achieved by following the "by the book" tuning techniques and assumptions but through exhaustive experiments and testing. Moreover, upon obtaining the values of the hyperparameters, which provided the higher accuracy from all the experiments. From the important findings, we

**Table 2.** Results using default hyperparameter values for Keras optimizers for both AlexNet and Inception V3.

CNN	Optimizer	Batch size	Learning rate	Epochs	True pos	True neg	Accuracy
AlexNet	Adadelata	32	0.001	200	240/450	418/450	73.11%
"	Adagrad	32	0.001	200	255/450	409/450	73.78%
"	Adamax	32	0.001	200	285/450	353/450	70.89%
"	Nadam	32	0.001	200	238/450	402/450	71.11%
"	SGD	32	0.01	200	278/450	402/450	75.56%
Inception V3	Adadelata	32	0.001	200	377/450	443/450	91.11%
"	Adagrad	32	0.001	200	375/450	444/450	91.00%
"	Adamax	32	0.001	200	304/450	444/450	83.11%
"	Nadam	32	0.001	200	299/450	415/450	79.33%
"	SGD	32	0.01	200	394/450	442/450	92.89%

**Table 3.** Results using tuned hyperparameter values for Keras optimizers for both AlexNet and Inception V3, with its accuracy gain respect to the default values.

CNN	Optimizer	Batch size	Learning rate	Epochs	True pos	True neg	Accuracy	Change
AlexNet	Adadelata	32	0.01	200	235/450	421/450	72.89%	-0.22%
"	Adagrad	128	0.001	200	239/450	413/450	72.44%	-1.34%
"	Adamax	64	0.0001	200	276/450	409/450	76.11%	+5.22%
"	Nadam	64	0.0001	200	282/450	411/450	77.00%	+5.89%
"	SGD	128	0.01	200	272/450	409/450	75.67%	+0.11
Inception V3	Adadelata	64	0.01	200	386/450	443/450	92.11%	+1.00%
"	Adagrad	64	0.01	200	397/450	444/450	93.44%	+2.44%
"	Adamax	128	0.0001	200	394/450	444/450	93.11%	+10.00%
"	Nadam	256	0.0001	200	389/450	444/450	92.56%	+13.23%
"	SGD	128	0.01	200	383/450	441/450	91.56%	-1.33%

have that 1) the optimizer was not one of the most used in CNN's literature, 2) the learning rate was not the default and presumably best for the optimizer, and 3) the number of epochs was way too low in comparison with the one employed with other optimizers to get similar results. We can conclude that the combination of the hyperparameters favors the optimum performance for our case study. Thus, we cannot obviate a single "winning formula" for a given problem and data set. The optimum performance for the handgun classification model was obtained training Inception V3 with transfer learning using ImageNet data set and the hyperparameters: Nadam optimizer, a learning rate of 0.0001 (ten times less than its default value), a batch size of 256, and a number of epochs equals to 13. The confusion matrix is presented as shown in Table 4.

**Table 4.** Confusion matrix for Inception V3 with optimum parameters.

Actual/Predicted	Positive	Negative
Positive	406	44
Negative	9	441

**Table 5.** Confusion matrix for AlexNet with optimum hyperparameters.

Actual/Predicted	Positive	Negative
Positive	282	168
Negative	39	411

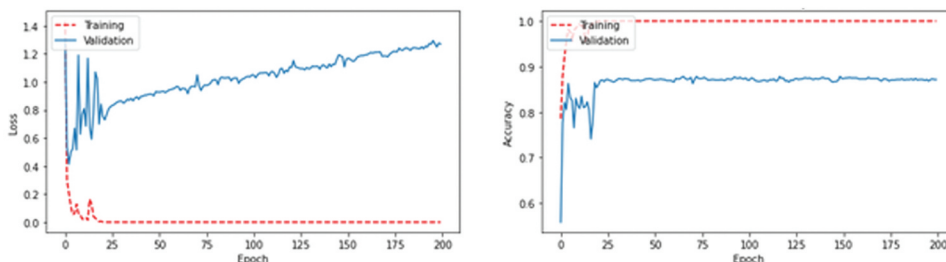
The classifier with the best performance had an accuracy value of 94.11% and a precision value of 92.22%. These results represent a 10.33% accuracy improvement and a 9.34% precision improvement over the default values when using the Nadam optimizer. Meanwhile, for the AlexNet network (without employing transfer learning techniques), using the Nadam optimizer with a learning rate of 0.0001, a batch size equals 64 and 200 epochs, was the configuration with which the best results were obtained, achieving a maximum accuracy of 77% and precision of 71.33%. Compared to the results obtained using the default values, these results represent an improvement of 5.89% in accuracy and 7.78% in precision. Table 5 shows the confusion matrix for this configuration of hyperparameters. Figure 4 shows some examples of images that were classified as containing firearms incorrectly. It can be seen that those images are hard to classify.

In Figures 5 and 6, we can see the drop in validation loss and rise in accuracy across epochs during the training process, for the best performance configurations of both AlexNet and Inception V3. Figures 7 and 8 show the Receiver Operating Characteristic curves (ROC) for the training of both networks, respectively.

## Discussion

Hyperparameter tuning is a process of vital importance in the development of a CNN-based classification model. For instance, choosing one optimizer over another can be the difference between a high accuracy model and a dummy

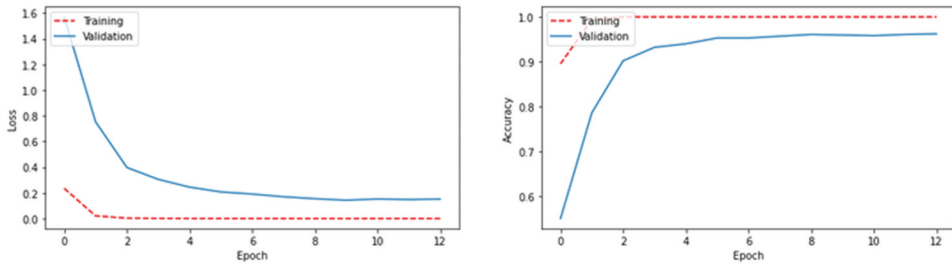
Training and validation accuracy across epochs.



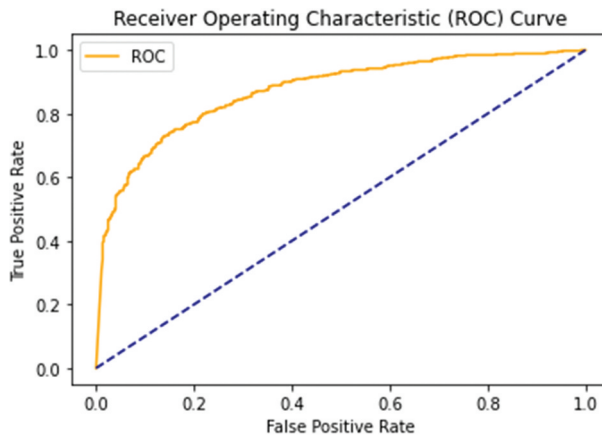
**Figure 4.** Validation loss and accuracy across epochs during the training process of AlexNet network with tuned hyperparameters.



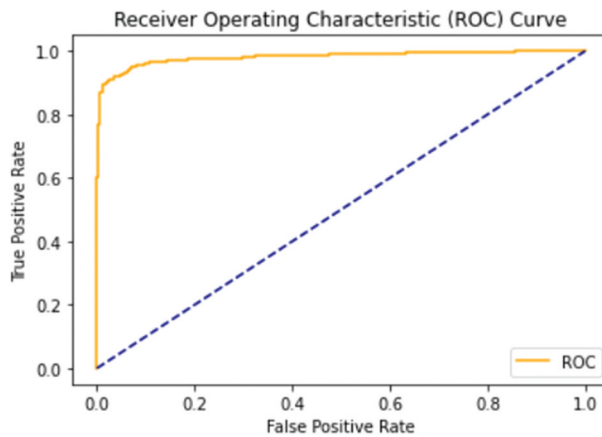
### Training and validation accuracy across epochs.



**Figure 5.** Validation loss and accuracy across epochs during the training process of Inception V3 network with tuned hyperparameters.



**Figure 6.** ROC curve for AlexNet network with tuned hyperparameters. Its area under the curve value is 0.8709.



**Figure 7.** ROC curve for Inception V3 network with tuned hyperparameters. Its area under the curve value is 0.9812.



**Figure 8.** Examples of misclassification.

classifier, as was demonstrated by the experiments results. Additionally, we found that optimizer default values do not always yield the best performance for a particular classification problem. For instance, when we lowered tenth fold the default learning rate for the Nadam optimizer we converted a below-average performing model, which has a test accuracy equals 67.67%, into a high-performance one, with a test accuracy of 94.11%.

For the particular data set used in our experiments, after selecting the right optimizer, the most critical factor in building an optimum classification model was selecting the adequate CNN to use. In our case, GoogLeNet (Inception V3) yielded much better results than AlexNet, averaging 79.81% accuracy against the 64.88% accuracy obtained by AlexNet, when using the default values from optimizers. The better performance of GoogLeNet (Inception V3) is a consequence of two facts: the use of transfer learning from the Imagenet database and the larger number of layers used by GoogLeNet.

The next hyperparameters on our list of importance in the tuning process are the learning rate and the batch size since a combination of a low value for learning rate (0.0001), and a significant value for batch size (256) provided the best results while running tests. However, it is essential to mention that the lowest learning rate values tend to slow down the learning process, requiring more processing time during the training process. In the same way, the largest batch size values consume more memory, which makes the training process computationally more expensive.

At the end of our list, we have the number of epochs because, from the results of our experiments, we can conclude that specifying a large number of epochs does not guarantee better results than a small one due to the possibility of the overfitting scenario. This not significant impact of the number of epochs

usually happens because presenting the data set too many times to the network can help to improve training accuracy. However, it makes it harder for the model to classify correctly data it has never seen before.

Across the different phases of our experiments, we could discern some patterns between the performance of the optimizers employed with one or various specific hyperparameters. For example, Adadelta and Adagrad showed positive effects when increasing the learning rate from 0.001 to 0.01 (ten times its original value), disregarding the batch size employed. On the other hand, the optimizers Adamax and Nadam yielded performance improvement when reducing the learning rate from 0.001 to 0.0001, and this improvement kept growing as the batch size increased. This performance improvement occurred in both networks, so it would be safe to say that, for the most part, what determines these correlations is the dataset. Another interesting finding was that big batch sizes tend to go along quite well with small numbers of epochs. The experiments demonstrated that the validation accuracy ceases to grow before specifying large batch size values compared to small ones. Sometimes, the validation accuracy drops after a certain number of epochs. Nevertheless, again, this happened in both networks.

Improved accuracy in classifying firearms in images has been reported in other works. For instance, in Olmos, Tabik, and Herrera (2018a), the author uses a classifier trained under the VGG-16 architecture, specially designed to minimize prediction loss. VGG-16 has more parameters than both AlexNet and GoogLeNet standard networks, but it is considerably more computationally expensive. His work reaches a maximum precision of 84,21%. Moreover, the final model is then analyzed as an automatic alarm system. In the same way, in Kanehisa and Neto (2019) the authors employ a network architecture based on Redmon and Farhadi (2017) as a classifier, reaching an accuracy of 96.26% and precision of 95.74%. They used the Internet benchmark IMFDb data set for their experiments.

Another example is discussed in Galab et al. (2020), which proposes the application of transfer learning techniques to the well-recognized CNN GoogLeNet and AlexNet. Their approach successfully outperforms its predecessors, reaching an accuracy of 99.2% and precision of 99.5% while using AlexNet for trained network, and a 97.7% accuracy and a 97.3% precision while using GoogLeNet. All their tests were performed on the IMFDb data set as well.

Those results are quite encouraging, but there is always room for improvement when dealing with a subject as important as public safety. With this in mind and looking for a way to understand and improve the actual methods, we inquired deep into the fundamental aspects of CNN. It is important to note that the mentioned related works do not discuss the hyperparameters used while training their networks or their impact on the performance of the classification model.

In contrast to other works, we deeply explore the foundation of CNN. One of the research goals was to evaluate the impact of the use or absence of transfer learning techniques or the number of layers in the networks. For instance, AlexNet has eight layers, while Inception V3 has 42 layers.

Among the limitations of our experiments, we must consider that the inner processes of the CNN include employing randomness to a certain extent, resulting in a slightly different model each time the network is trained, even when using the same hyperparameter settings and data set. This behavior affects the final prediction outcomes from one experiment to another. The most effective solution to this issue would be to repeat the experiments several times until we get a general behavior for each setting, calculated by a statistic function. However, creating an optimum model could require a considerable number of epochs, and we had to test multiple times each combination of hyperparameter values for batch size, optimizer and learning rate, thus, the time and computational cost necessary for such a massive number of experiments would make the attempt for normalization unfeasible.

Also, since the experiments showed that after around 50 epochs, the overall model accuracy ceases to grow (and in some cases, principally for large batch sizes, it diminishes), the epochs value beyond 200 was not explored.

## Conclusions and Future Work

In this paper, we analyzed the benefits of tuning the values for a classification algorithm hyperparameters compared to its default values on the performance of a handgun classification model. Particularly, we evaluated two benchmark CNN architectures: AlexNet and Inception V3. We obtained a 94.11% test accuracy after training our model in the Inception V3 network while employing transfer learning from the ImageNet database. We used Nadam as the optimizer with a custom learning rate of 0.0001, a batch size of 256, and a total of 13 epochs. Experimental results revealed an important relationship between the data set, specific combinations of values for the selected optimizer, batch size, learning rate, and the final performance of the classification model since improvements yielded an accuracy of up to 10.33% after the tuning process.

As future work, once successfully detecting the best optimizer for the image-based firearm classification problem (Nadam), we are interested in investigating the impact on the outcome of its configurable internal parameters (epsilon value, dropout, and momentum). Also, we would evaluate those hyperparameters that define the network's structure, such as the number of network layers, activation functions, and dropout, to determine if there is also a correlation between the tuning process between them and the

performance of the CNN. The obtained results showed that using different values for the hyperparameters of a CNN-based classification model, other than the default values, yielded better performance.

The use of tools for automatic tuning of hyperparameters, such as Bayesian optimization, is an interesting topic for future work. Comparing the obtained results between using grid-search and automatic tuning of hyperparameters using the same dataset of this project will give a clue of the advantages and limitations of both approaches.

Additionally, we consider developing a data set considering specific scenarios, such as a school, college campus, or a particular city spot. This data set will represent more accurately a target application. Once the data set is built, it will be used to develop a classification model using the results obtained in this paper and implement an Internet of Things-based system.

Finally, we consider converting the architecture of a CNN with transfer learning which yields the best accuracy in classifying firearms in images, into a firearms detector in images or video. The main idea is to compare our transfer learning-based detector with traditional object detectors, such as regions with CNN features (R-CNN) Girshick et al. (2014), Fast-R-CNN Girshick et al. (2014), Faster-RCNN Ren et al. (2017), and YOLO Redmon and Farhadi (2017), among others.

## Disclosure Statement

No potential conflict of interest was reported by the author(s).

## ORCID

Isaac Cardoza  <http://orcid.org/0000-0002-9776-2663>

Juan P. García-Vázquez  <http://orcid.org/0000-0003-1787-7223>

Arnoldo Díaz-Ramírez  <http://orcid.org/0000-0002-6188-0756>

Verónica Quintero-Rosas  <http://orcid.org/0000-0002-8508-7429>

## References

- Ağdaş, M. T., M. Türkoğlu, and S. Gülseçen. 2021. Deep neural networks based on transfer learning approaches to classification of gun and knife images. *Sakarya University Journal of Computer and Information Sciences* 4 (1):131–41. doi:10.35377/saucis.04.01.891308.
- Aghdam, H. H., and E. Jahani Heravi. 2017. *Guide to Convolutional Neural Networks*, 1st ed. Switzerland: Springer. doi:10.1007/978-3-319-57550-6.
- Albelwi, S., and A. Mahmood. 2017. A framework for designing the architectures of deep convolutional neural networks. *Entropy* 19 (6):242. <https://www.mdpi.com/1099-4300/19/6/242>.
- Alex, K., I. Sutskever, and G. E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60 (6):84–90. doi:10.1145/3065386.

- Alzubaidi, L., J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, J. S. Omran Al-Shamma, M. A. Fadhel, M. Al-Amidie, and L. Farhan. 2021. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data* 8 (53):1–74. doi:10.1186/s40537-021-00444-8.
- Annamraju, A. 2019. “Weapon Detection Dataset.” Accessed 2 July 2021. <https://www.kaggle.com/abhishek4273/gun-detection-dataset>.
- Bochinski, E., T. Senst, and T. Sikora. 2017. “Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms.” In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Beijing, China, September, pp. 3924–28. doi: 10.1109/ICIP.2017.8297018.
- Dwivedi, N., D. Kumar Singh, and D. Singh Kushwaha. 2019 April. “Weapon Classification using Deep Convolutional Neural Network.” In *2019 IEEE Conference on Information and Communication Technology*, pp.1–5. Manhattan, New York, U.S.: IEEE. doi: 10.1109/CICT48419.2019.9066227.
- Elmir, Y., S. Ahmed Laouar, and L. Hamdaoui. 2019. “Deep learning for automatic detection of handguns in video sequences.” In *JERI*, April. [http://ceur-ws.org/Vol-2351/paper\\_69.pdf](http://ceur-ws.org/Vol-2351/paper_69.pdf).
- Feurer, M., and F. Hutter. 2019. Hyperparameter optimization. *Automated Machine Learning: Methods, Systems, Challenges* The Springer Series on Challenges in Machine Learning, pp. 3–33. Switzerland: Springer International Publishing. doi:10.1007/978-3-030-05318-5\_1.
- Galab, M., M. Mai, A. Taha, and H. Zayed. 2020. Automatic gun detection approach for video surveillance.” *International Journal of Sociotechnology and Knowledge Development* 12 (1):49–66. doi:10.4018/IJSKD.2020010103.
- Girshick, R., J. Donahue, T. Darrell, and J. Malik. 2014. “Rich feature hierarchies for accurate object detection and semantic segmentation.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, USA, June, pp. 580–87. doi: 10.1109/CVPR.2014.81.
- iStock webpage. 2021. “Gun pointed images.” Accessed 2 July 2021. Getty images. <https://www.istockphoto.com/es/search/2/image?page=3&phrase=gun%20pointed>
- INEGI. 2019a September. “Encuesta Nacional de Victimización y Percepción sobre Seguridad Pública (ENVIPE) ENVIPE 2019.” Available online: <https://www.inegi.org.mx/programas/envipe/2019/>.
- INEGI. 2019b. “Mortalidad. Conjunto de datos: Defunciones por homicidios.” October. <https://www.inegi.org.mx/sistemas/olap/proyectos/bd/continuas/mortalidad/defunciones/shom.asp?s=est>.
- Kamilaris, A., and F. X. Prenafeta-Boldú. 2018. A review of the use of convolutional neural networks in agriculture. *The Journal of Agricultural Science* 156 (3):312–22. doi:10.1017/S0021859618000436.
- Kandel, I., and M. Castelli. 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express* 6 (4):312–15. doi:10.1016/j.icte.2020.04.010.
- Kanehisa, R., and A. Neto. February 2019. “Firearm detection using convolutional neural networks.” In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART)*, 707–14. Prague, Czech Republic: 2019). Doi: 10.5220/0007397707070714.
- Kaya, V., S. Tuncer, and A. Baran. 2021. Detection and classification of different weapon types using deep learning. *Applied Sciences* 11 (16):7535. doi:10.3390/app11167535.
- Keras, A. P. I. 2020a. “Keras Image data preprocessing.” Accessed 2 July 2021. <https://keras.io/api/preprocessing/image/>.
- Keras, A. P. I. 2020b. “Keras Optimizers.” Accessed 2 July 2021. <https://keras.io/api/optimizers/>

- Ketkar, Nikhil. 2017. . In *Deep learning with python, a hands-on introduction*, 1st ed. Berkeley, CA: Apress. doi:10.1007/978-1-4842-2766-4.
- Kolar, D., D. Lisjak, M. Pajak, and M. Gudlin. 2021. Intelligent fault diagnosis of rotary machinery by convolutional neural network with automatic hyper-parameters tuning using bayesian optimization. *Sensors* 21 (7):2411. <https://www.mdpi.com/1424-8220/21/7/2411>.
- Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Lecture Notes in Computer Science.- European conference on computer vision* 8693: 740–55. doi:10.1007/978-3-319-10602-1\_48.
- Olmos, R., S. Tabik, and F. Herrera. 2018a. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing* 275:66–72. 2017.05.012. doi:10.1016/j.neucom.
- Olmos, R., S. Tabik, and F. Herrera. 2018b. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing* 275:66–72. 2017.05.012. doi:10.1016/j.neucom.
- Pérez-Pérez, B. D., J. Pablo García Vázquez, and R. Salomón-Torres. 2021. Evaluation of convolutional neural networks' hyperparameters with transfer learning to determine sorting of ripe medjool dates. *Agriculture* Doi: 10.3390/ agriculture11020115. 11 (2):115. doi:10.3390/agriculture11020115.
- Rao, Y., G. Zhang, W. Zhou, C. Wang, and Y. Lv. 2019 April. “Deep convolutional neural network based traffic vehicle detection and recognition. *International Conference on Internet of Things as a Service* Xian, China, pp. 427–38. Switzerland: Springer. doi: 10.1007/978-3-030-44751-9\_36.
- Redmon, J., and A. Farhadi. 2017. “YOLO9000: Better, faster, stronger.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, USA, July, 7263–71. Doi: 10.1109/CVPR.2017.690.
- Ren, S., H. Kaiming, R. Girshick, and J. Sun. 2017. Faster R-CNN: Towards real- time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (6):1137–49. doi:10.1109/TPAMI.2016.2577031.
- Sarvamangala, D. R., and R. V. Kulkarni. 2021. Convolutional neural networks in medical image understanding: A survey. *Evolutionary Intelligence* 1–22. doi:10.1007/s12065-020-00540-3.
- Smith, L. N. 2018. *A disciplined approach to neural network hyper-parameters: Part 1– Learning rate, batch size, momentum, and weight decay*. Technical Report US Naval Research Laboratory Technical Report 5510-026. Cornell University.
- Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2016. “Rethinking the inception architecture for computer vision.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June, pp. 2812–2816. doi: 10.1109/CVPR.2016.308.
- Tang, B., Z. Pan, K. Yin, and A. Khateeb. 2019. Recent advances of deep learning in bioinformatics and computational biology. *Frontiers in Genetics* 10:214. doi:10.3389/fgene.2019.00214.
- UNODC. 2019. “Global study on homicide 2019.” July. Accessed 2019. <https://www.unodc.org/unodc/en/data-and-analysis/global-study-on-homicide.html>.
- UNODC. 2020. “Global study on firearms trafficking 2020.” Mar. Accessed 2020. <https://www.unodc.org/unodc/en/firearms-protocol/firearms-study.html>.
- Veranyurt, O., and C. Okan Sakar. 2020. “Hand-Gun detection in images with transfer learning-based convolutional neural networks.” In *28th Signal Processing and Communications Applications Conference (SIU)*, Gaziantep, Turkey, January, pp. 1–4. Manhattan, New York, U.S.: IEEE. doi: 10.1109/SIU49456.2020.9302394.

- Victoria, A. H., and G. Maragatham. 2021. Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems* 12 (1):271–223. doi:10.1007/s12530-020-09345-2.
- Webpage, R. 2020. “Pistols Dataset.” Accessed 2 July 2021. <https://public.roboflow.com/object-detection/pistols/1>.
- Yang, L., and A. Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (4):295–316. doi:10.1016/j.neucom.2020.07.061.
- Zaccone, G., and M. R. Karim. 2018. *Deep learning with tensorflow: Explore neural networks and build intelligent systems with python*. 2nd ed. Birmingham, United Kingdom: Packt Publishing. <https://books.google.com.mx/books?id=zZlUDwAAQBAJ>.
- Zewen, L., F. Liu, W. Yang, S. Peng, and J. Zhou. 2021. A survey of convolutional neural networks: analysis, applications, and prospects.” *IEEE Transactions on Neural Networks and Learning Systems*:1–21. doi:10.1109/TNNLS.2021.3084827.
- Zhao, H., O. Gallo, I. Frosio, and J. Kautz. 2017. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging* 3 (1):47–57. doi:10.1109/TCI.2016.2644865.
- Zimring, F. E. 2020. Firearms and violence in American life—50 years later. *Criminology & Public Policy* 19 (4):1359–69. doi:10.1111/1745-9133.12523.