ACTA UNIVERSITARIA
Multidisciplinary Scientific Journal

I VOLUMEN 34 I 2024 I
ISSN online 2007-9621
http://doi.org/10.15174/au.2024.3919

Design of an application to detect covid-19 using convolutional neural networks and X-ray images

Diseño de una aplicación para detectar covid-19 mediante redes neuronales convolucionales e imágenes de rayos X

Carlos Eduardo Belman López[1*]

[1*] Departamento de Ingeniería Industrial, Tecnológico Nacional de México. C.P. 38010. Celaya, Guanajuato, México.
carlosbelman@gmail.com. https://orcid.org/0000-0003-1305-6778
*Autor de correspondencia

Abstract

This research presents the design of an application to detect covid-19 using convolutional neural networks and X-ray images in two scenarios (covid/Non-covid and covid/Normal/Pneumonia). To avoid overfitting online data augmentation, dropout, batch normalization, and Adam optimizer was used. The three-class network was used as a pre-trained model, tuning only the dense and output layers to obtain the binary model. Additionally, hyper-parameter optimization was used to get dropout probabilities, activation functions, and neurons. The learning rate was adjusted using callbacks to avoid local optimums. Networks were converted to TensorFlow.js format and embedded locally in a hybrid application using Ionic and Capacitor and were deployed through Firebase to help provide diagnostics. The application obtained an accuracy of 98.61% and 96.48% for two and three classes, respectively, achieving higher performance when compared to other proposals (offline models) in the literature and using fewer training parameters.

Keywords: covid-19; artificial vision; convolutional neural networks.

Resumen

Este documento presenta el diseño de una aplicación para detectar covid-19 utilizando redes neuronales convolucionales e imágenes de rayos X en dos escenarios (covid/No-covid y covid/Normal/Neumonía). Para evitar el sobreajuste, se utilizó aumento de datos, dropout, normalización por lotes y optimizador Adam. La red para tres clases se utilizó como modelo pre-entrenado ajustando solo la capa densa y de salida para obtener el modelo binario. Además, se realizó una optimización automatizada de hiper-parámetros como dropout, funciones de activación y número de neuronas. La tasa de aprendizaje se ajustó mediante *callbacks* para evadir óptimos locales. Las redes fueron convertidas al formato TensorFlow.js para integrarse en una aplicación híbrida utilizando Ionic y Capacitor, y se desplegaron mediante Firebase para brindar asistencia y soporte al generar diagnósticos. La aplicación obtuvo una exactitud del 98.61% y 96.48% para dos y tres clases, respectivamente, logrando mayor rendimiento que otras propuestas y utilizando menos parámetros de entrenamiento.

Palabras clave: covid-19; visión artificial; redes neuronales convolucionales.

Recibido: 22 de junio de 2023 Aceptado: 27 de mayo de 2024 Publicado: 24 de julio de 2024

Cómo citar: Belman López, C. E. (2024). Design of an application to detect covid-19 using convolutional neural networks and X-ray images. *Acta Universitaria 34*, e3919. doi: http://doi.org/10.15174/au.2024.3919

Introduction

In December 2019, the World Health Organization (WHO) alerted about cases of respiratory illness of unknown origin coming from Wuhan, China. The virus was named covid-19 and is caused by the new coronavirus SARS-CoV-2 (WHO, 2022). Most people infected with covid-19 recover without requiring any specialized treatment, but older people or those with associated medical conditions such as diabetes, cancer, chronic respiratory, or cardiovascular diseases have a high probability of severe illness.

Today, the scientific community still plays a fundamental role in the development of tools that allow the diagnosis and monitoring of this disease. In radiology, many works in the literature have focused on the use of computed tomography (CT) scans for the detection of covid-19 (Zhou *et al.*, 2020). However, CT scans have drawbacks related to patient transportation, waiting time between services to sanitize the facilities, inefficient sanitizations, and shortage of CT in certain places. Due to these drawbacks, chest X-ray images are more widely used as a tool for the identification of respiratory and lung abnormalities such as covid-19 (American College of Radiology, 2020). Additionally, the X-ray images for the detection of this disease have a key role given the limited or insufficient access to RT-PCR (reverse transcriptase polymerase chain reaction) tests in some parts of the world (Jacobi *et al.*, 2020).

Millions of confirmed cases of covid-19 and thousands of deaths were reported (WHO, 2020). To stop the increase in the number of infections, it is crucial to detect positive cases as soon as possible, isolating and treating patients instantly. This leads to the need to develop auxiliary tools for the diagnosis, where recent research indicates that X-ray images contain significant information about covid-19 (Chung *et al.*, 2020). Artificial intelligence techniques, especially deep learning (Beysolow II, 2017), together with X-ray images can be useful for the diagnosis of covid-19 and compensate for the lack of specialists in remote locations (Ozturk *et al.*, 2020).

Deep learning (DL) has already been used for image and signal processing in healthcare, improving diagnostics based on medical, multidimensional, or thermal images. Recently, X-ray images and DL models have been used for the detection of covid-19. Loey *et al.* (2020) used a DL model based on GAN and transfer learning to reach an accuracy of 85.2% in the prediction of three classes (covid-19, Normal and Pneumonia) but using only 69 covid-19 images and 306 total images for all the classes. For transfer learning, AlexNet, GoogleNet, and ResNet were selected in this study. Rahimzadeh & Attar (2020) trained several deep convolutional networks into three classes: normal, pneumonia, and covid-19 using 180 X-ray images belonging to persons infected with covid-19 and an unbalanced dataset (fewer cases of covid-19 along with more cases from other classes). For the networks, they proposed a neural network that is a concatenation of the Xception and ResNet50V2 pre-trained models. The overall average accuracy for all classes was 91.4%.

Wang *et al.* (2020) developed a DL model based on a pre-trained model from ImageNet to achieve an accuracy of 93.3% in the diagnostic of three classes (covid-19, Normal and Pneumonia) but using only 358 covid-19 images and 11.95 million parameters. This model makes heavy use of a lightweight residual projection-expansion projection-extension (PEPX) design pattern. Ioannis & Tzani (2020) developed a DL model using only 224 covid-19 images based on VGG-19 model, obtaining an accuracy of 98.75% and 93.48% in the detection of two and three classes, respectively.

VGG-19 is a deep neural network architecture that does not leverage residual design principles and lightweight design patterns, and it has very low architectural diversity. Ozturk *et al.* (2020) proposed a DL model consisting of 17 convolutional layers and different filtering on each layer. This approach reached an accuracy of 98.08% and 87.02% for the detection of two and three classes, respectively, but using only 125 covid-19 images and without using techniques for avoiding overfitting such as dropout or data augmentation. Narin *et al.* (2020) reached an accuracy of 98% in detecting covid-19, using only 100 X-ray images (50 covid-19 + 50 Non-covid) in conjunction with the ResNet50 model. ResNet-50 is a deep neural network architecture that leverages residual design principles and lightweight design patterns (e.g., bottleneck design patterns), and it has moderate architectural diversity, but it does not leverage lightweight PEPX sign patterns or selective long-range connectivity. Sethy & Behera (2020) achieved an efficiency of 95.3% in detecting covid-19 using only 50 X-ray images (25 covid-19 + 25 Non-covid) in conjunction with the ResNet50 DL model and support vector machines (SVM). Mahmud *et al.* (2020) used a convolutional network with multi-dilation for the detection of covid-19, reaching an accuracy of 90% in predicting several cases of pneumonia and covid-19.

Every previous research mentioned above is limited by the small number of images used in comparison to the large network capacity (number of trainable parameters) they are using, and the results can also be significantly improved. Moreover, transfer learning models are more useful when the pre-trained models are developed using images with some similarity; otherwise, they will have to be adjusted to be relevant to the problem in question, requiring expensive computing power. Furthermore, the most complex networks do not always produce the best results.

Springenberg *et al.* (2015) has already mentioned that the simplest models are less likely to overfit and may produce better results. Also, pretrained models -such as ImageNet-based models, among others- used to end in very heavy models, making it difficult to deploy them locally in devices with limited capacities. Finally, a platform or application that makes models available to final users in the form of services (software as a service) is essential to help in the detection process.

For these reasons, the contribution of this research is the design of a convolutional neural networks-based application that can be an assistance support to diagnose covid-19 using X-ray images. The proposed application and networks provide accuracy results in two (covid*vs.* Non-covid) and three classes (covid*vs.* Normal vs. Pneumonia), achieving competitive results (and even better) when compared to other proposals (offline models) in the literature and using significantly fewer training parameters. This allowed embedding the models in an application that can be executed without the need for Internet access.

To avoid overfitting, online data augmentation with normalization was used during training execution. The networks also included dropout and batch normalization layers in addition to Adam optimizer. The three-class network was used as a pre-trained model, preserving the convolutional layers, and tuning only the dense and output layers to obtain the binary network (this represents cost savings if time from a cloud computing platform such as AWS is used for training). In both scenarios, automated hyper-parameter optimization (HPO) was used to optimize hyper-parameters such as dropout probabilities, activation functions, and neurons in the dense layer. Additionally, the learning rate was adjusted using callbacks to escape from local optimums. The datasets (training, validation, and testing) were increased compared to most studies.

The proposed networks obtained an accuracy of 98.61% and 96.48% for two and three classes in the validation sample and 99.07% and 93.15% in the testing sample, achieving higher performance when compared to other proposals in the literature and using significantly fewer training parameters. Finally, these network sizes allowed to convert the models into TensorFlow JS format and to embed them locally in a hybrid application (using Ionic and Capacitor) that can be installed and executed without the need for Internet access (which is not always available in remote or poor areas). Additionally, the application was deployed into Internet through Firebase to help provide diagnostics at any time and place.

Deep learning and convolutional neural networks

Deep learning (DL) is a method of data analysis and a branch of artificial intelligence that automates analytical model building (SAS Institute Inc., 2023). The deep in DL is not a reference to any kind of deeper understanding achieved by the approach; rather, it stands for the idea of successive layers of representations, including automatic feature learning and high-volume modelling capabilities (Wang *et al.*, 2018; Wang *et al.*, 2020; Wuest *et al.*, 2016).

DL has several properties that justify its importance within artificial intelligence, such as simplicity (no feature engineering), scalability (within GPU or TPU), and versatility and reusability (continuous learning). The key in DL solutions is the balance between optimization and generalization. Optimization refers to adjusting a model to obtain the best performance on the training data, whereas generalization measures how well the trained model performs on data it has never seen before. When generalization stops improving on the training data, and validation metrics begin to degrade, then the model is starting to overfit. It means the model is learning patterns that are specific to the training data but irrelevant for new observations. It should be highlighted that, the bigger network will likely overfit, meanwhile simpler models are less likely to overfit than complex ones (Springenberg *et al.*, 2015).

Convolutional neural networks (CNN) are a DL area primarily used for image recognition and classification (Loey *et al.*, 2020), although they have also been investigated in other areas such as natural language processing and speech recognition (Wang *et al.*, 2018). In traditional artificial neural networks (ANN), each neuron in the input layer is connected to each neuron in the next layer, which is known as a dense or fully connected layer. However, in CNN, dense layers are used until the last part of the network (Rosebrock, 2017). Patterns learned by CNN are not limited to a particular position; for example, a pattern learned in the lower left corner may be recognized in the upper right corner or anywhere.

CNN achieve pattern recognition using a set of convolutional, pooling, and dense layers. Convolutional layers are combined with the input data using multiple filters (Rong *et al.*, 2020). A nonlinear activation function is applied to the output of the convolution layers (Rosebrock, 2017). The subsequent pooling layers extract the most significant features with a fixed length over sliding windows of the raw input data by pooling operations such as max pooling or average pooling. Max pooling selects the maximum value of one region in the feature map as the most significant feature. Average pooling calculates the mean value and takes it as the pooling value in this region. After multi-layer feature learning, fully connected layers convert a two-dimensional feature map into a one-dimensional vector and then feed it into a softmax function for model construction (Rong *et al.*, 2020).

Recently, several advances have emerged in the development of CNN. These advances include pre-trained models, model assembly, new training algorithms, activation functions, and regularization techniques. A pre-trained model is a model previously trained on a large dataset. If this original dataset is large and general enough, then the pre-trained model can act as a generic model, useful for many different computer vision problems, even though these new problems may involve different classes than those of the original task, such as VGGNet, GoogleNet, ResNet, Xception, and Inception-V3.

Additionally, new efficient training algorithms have been proposed, such as momentum-based SGD, AdaGrad, AdaDelta, Adam, and RMSProp. Another powerful technique to get the best possible results is model assembly. This technique consists of using predictions from a set of models to produce a better prediction. Ensembling relies on the assumption that different good models trained independently are likely to be good for different reasons: each model looks at slightly different aspects of the data to make its predictions, getting part of the "truth" but not all of it (Chollet, 2018).

ANN with a large number of parameters are powerful DL systems; however, overfitting is a serious problem. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural networks at test time. Dropout is a technique for addressing this problem. Dropout is a very effective way to prevent overfitting. The key idea is to randomly select and remove a set of neurons and their connections during training to prevent neurons from correlating. Dropout significantly reduces overfit, providing greater improvements than other regularization methods. The authors propose to use "dropout" in ranges from 0.5 to 0.8 for hidden layers and greater than 0.8 for visible layers (input and output). Standard backpropagation learning usually builds up brittle co-adaptations that work for the training data, but do not generalize to unseen data. Random dropout breaks up these co-adaptations by making the presence of any hidden unit unreliable. It has been tested that Dropout is a general technique and is not specific to any domain (Srivastava *et al.*, 2014).

In addition, new and improved activation functions have been introduced, such as the rectifier linear unit (ReLU) function that significantly improves older activation functions such as sigmoid or tangent (Pedamonti, 2018). Different ReLU variants have been introduced such as Leaky ReLU, ELU, and SELU. But, particularly, the exponential linear units (ELU) activation function can speed up training and lead to higher ranking performances. Like ReLU, leaky ReLU, and parametrized ReLU, ELU alleviate the vanishing gradient problem via the identity for positive values. However, ELU have improved learning characteristics compared to the units with other activation functions. In contrast to ReLU, ELU have negative values which allow them to push mean unit activations closer to zero, like batch normalization, but with lower computational complexity. Mean shifts toward zero speed up learning by bringing the normal gradient closer to the unit natural gradient because of a reduced bias shift effect (Clevert *et al.*, 2016).

Finally, batch normalization (BN) allows to use higher learning rates and be less careful about initialization. It may also regularize, eliminating in some cases the need for dropout. BN allows to speed up the training phase using higher training rates, while it regularizes the network parameters. Merely adding BN to a state-of-the-art image classification model yields a substantial speed up in training. By further increasing the learning rates, and applying other modifications afforded by BN, it is possible to reach state-of-the-art results with only a small fraction of training steps (Zhou *et al.*, 2020).

Materials and methods

This research presents the design and development of a convolutional neural networks-based application that can be an assistance support to diagnose covid-19 using chest X-ray images. The proposed networks were developed to provide accurate results in two (covid*vs.* Non-covid) and three classes (covid*vs.* Normal *vs.* Pneumonia).

The proposed method (Figure 1) benefits from the fact that both scenarios are very similar, and it uses transfer learning from one scenario to the other. In order to avoid overfitting, online data augmentation with normalization was used during training execution. The architecture also included dropout and batch normalization layers in addition to Adam optimizer. The three-class network was the model used as a pre-trained model, preserving the convolutional layers and retraining (continuous learning) only the dense and output layers in order to obtain the binary network. Taking advantage of the fact that the three-class model was trained on a similar dataset, training time is saved by using it as a pre-trained model compared to training from scratch. This can represent cost savings if time from a cloud computing platform such as AWS is used for training. Therefore, a transfer learning strategy from one scenario (in this case the three-class scenario) to the other is a better option.

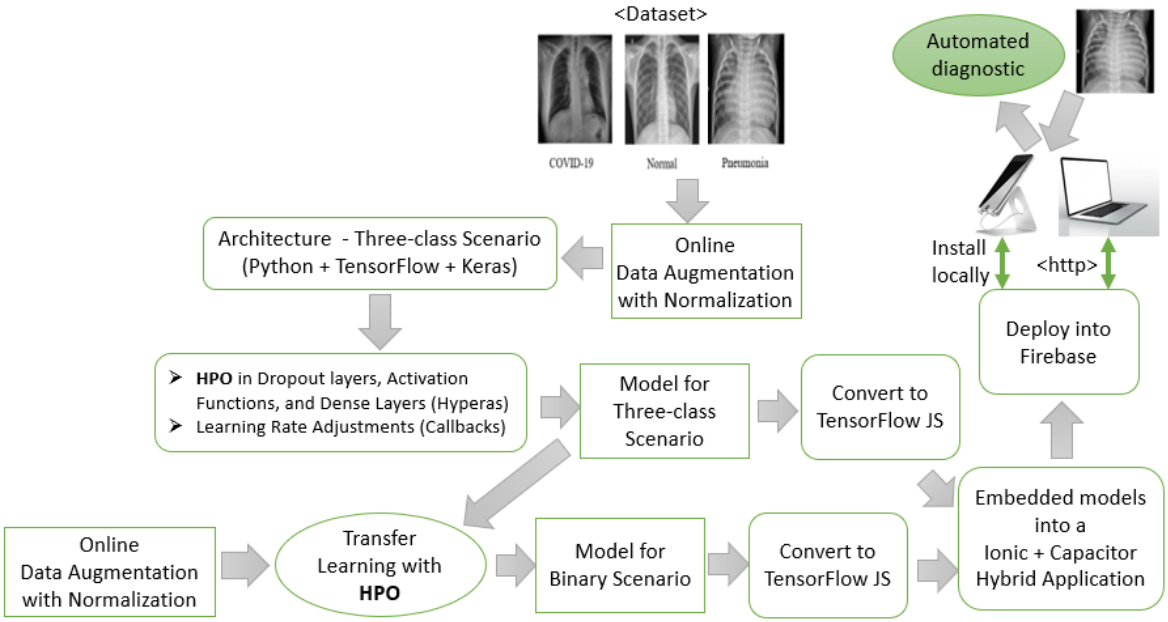

Figure 1. Proposed method.
Source: Author's own elaboration.

In both cases, HPO was used to optimize hyper-parameters such as dropout probabilities, activation functions (such as ReLU or ELU), and neurons in the dense layer. Additionally, the learning rate was adjusted using callbacks to escape from local optimums. Finally, the proposed models were converted to TensorFlow JS format and embedded locally in a hybrid application (using Ionic and Capacitor), and they were deployed through Firebase to help provide diagnostics at any time.

ISSN online 2007-9621

Dataset

In this investigation, X-ray images were acquired from various public data sources. For the design of the proposed models, the number of images for training, validation, and testing samples was considerably increased compared to most studies. For the covid-19 class, 489 images were obtained from Cohen *et al.* (2020) and the Italian Society of Medical and Interventional Radiology (2020); 35 images were obtained from the covid-19 Chest X-ray Dataset Initiative (Github, 2020); 56 images were obtained from Chung (2020); 31 images were obtained from European Society of Radiology (2021); and 31 images were obtained from Radiopaedia (2021). For the Non-coviddata set (Normal and Pneumonia), 5840 images were obtained from Kaggle Inc. (2023).

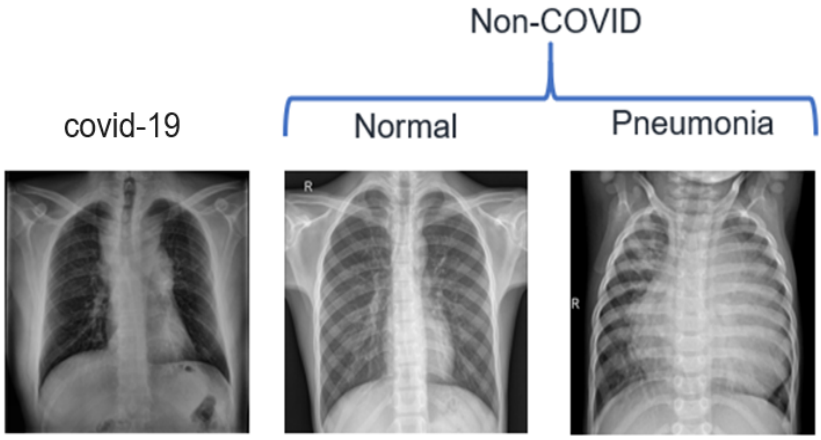

Figure 2. Categories involved in this investigation.
Source: Author's own elaboration.

Figure 2 illustrates some sample images used in this investigation. The number of images used for each scenario is presented in Table 1. The three-class scenario (covid-19, Normal and Pneumonia) analyzed a total of 6482 images, 642 for covid-19, 1580 for the Normal class and 4260 for Pneumonia class. Pneumonia class is internally composed of two types (bacterial and viral pneumonia). The two-class scenario (covid-19 and non-covid) analyzed a total of 6302 images, 642 for covid-19, and 5660 for the Non-covidclass (Normal and Pneumonia). The training and test samples use the same amount of data in both scenarios and the testing sample is never seen during the training phase. It should be noted that although the training and testing samples are unbalanced, the size of the validation sample (used during training process) is balanced using 180 images for each class (360 and 540 for 2 and 3 classes, respectively), although this caused a small difference in the total number of images used in both scenarios.

Table 1. Number of images and classes used for this investigation.

Scenario	Sample	covid-19	Non-covid			
			Normal	Pneumonia		
				Bacterial	Viral	Total
3 classes (6482 images)	Training	400	1 000	2 480	1 200	5 080
	Validation	180	180	90	90	540
	Testing	62	400	200	200	862
2 classes (6302 images)	Training	400	1 000	2 480	1 200	5 080
	Validation	180	60	60	60	360
	Testing	62	400	200	200	862

Source: Author's own elaboration.

Configuration and online data augmentation with normalization

CNN were developed using Python in conjunction with TensorFlow and Keras. The training was carried out using a computer equipped with an Intel Core i5-6200U processor (2.40 GHz), 16 GB of RAM, 222 GB of hard disk and Windows 10. The X-ray images passed through a preprocessing step in order to carry out an easier and faster training. All images were resized to 200x200, followed by a value normalization (dividing by 255), resulting in floating point inputs in a range between 0 and 1. The supported image types are all 3-channel, types such as DICOM are not supported right now.

In addition, online data augmentation was used to avoid overfitting. Data augmentation takes the approach of generating more training data from existing training samples by augmenting the samples via a number of random transformations that yield believable-looking images. The goal is that at training time, the model will never see the exact same picture twice (Chollet, 2018).

Data augmentation techniques can be divided into two types depending on the execution time. The first type consists of running desired transformations before training process. This is known as offline augmentation and is normally applied in small datasets. Here the dataset will be increased before training by a factor equal to the number of transformations. The second type is called online augmentation. Transformations are performed at the time of training the model, applying techniques such as rotation, zoom, crop, among others. Online augmentation increases the variability of the input images, so that the model will never see the exact same picture twice during the training phase providing higher robustness to the model (Gutierrez *et al.*, 2019). This also helps expose the model to more aspects of the data and generalize better. This research used online data augmentation approach applying the following techniques and parameters: rotation range in 35 degrees, width shift and height shift in 5%, zoom range in 10%, shear range in 10%, brightness range in 15%, and fill mode constant in black color.

Architecture and network optimization

The proposed base architecture is illustrated in Figure 3. The resolution used in the images allowed a maximum of six layers. Therefore, the architecture consists of six convolutional layers with 32, 32, 64, 64, 128, and 128 neurons, respectively. The convolutional layers used 3 x 3 filters and padding to conserve the size of the inputs. This padding allowed to extend the network without having to increase the size of the resolution. Additionally, six max pooling layers with 2x2 windows and no stride (stride 1) were utilized. To improve the optimization of the network, two batch normalization layers were added in the positions illustrated in Figure 2. Additionally, researchers must direct efforts to carefully coping with model design and corresponding hyper-parameter selection. This usually involves high costs, time, and effort. HPO consists of searching for optimum hyper-parameters to be used during the training process. Hyper-parameter refers to parameters that cannot be updated during the training of DL models, such as the structure of the model (layers), layer sizes, activation functions, among others (Yu & Zhu, 2020).

In this research, HPO was done using Hyperas and Hyperopt. Hyperas allows to use the power of Hyperopt without having to learn the syntax of it. Instead, just define the model and use a simple template notation to define hyper-parameter ranges to tune. The following hyper-parameters were tested using Hyperas during 100 epochs and five automatic evaluations (trials):

- Dropout probabilities in the dense and output layer in a uniform distribution over the interval [0,1], in order to avoid overfitting.
- Neurons in the dense layer, choosing between 256, 512, and 1024.
- Activation function in the different layers, choosing between ReLU and ELU.

ISSN online 2007-9621
Belman López, C. E.
Design of an application to detect covid-19 using convolutional neural networks and X-ray images | 1-16

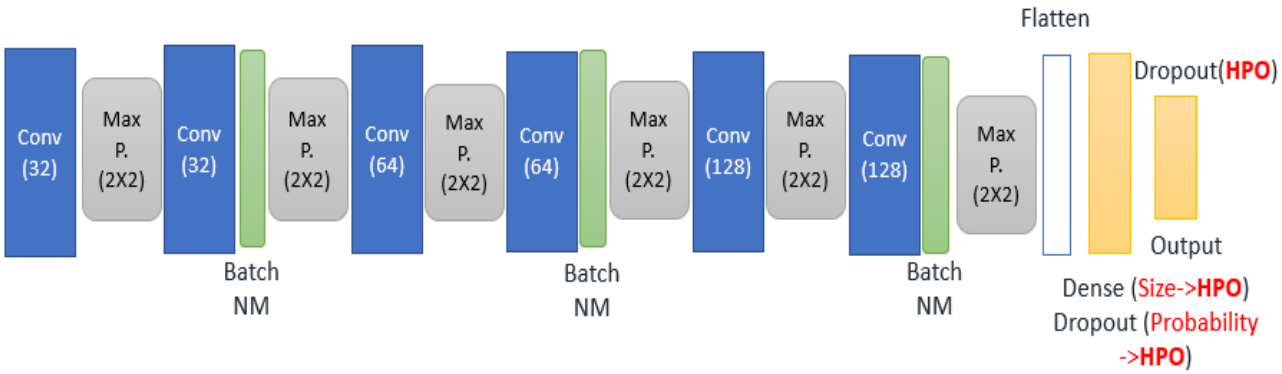

Figure 3. Base architecture.
Source: Author's own elaboration.

CNN were trained using Adam to minimize the loss function (binary and categorical crossentropy, respectively). The batch size (set to 20) was used for calculating the steps per epoch, dividing the training samples by the batch size. Accuracy was the metric used for monitoring the training, which was carried out over 100 epochs. In the output layer, the "softmax" function was used for the three-class scenario, while the "sigmoid" activation function was used for the binary case.

The initial training rate was set to 1.5e-3 but using "callbacks"; the training rate was adjusted by a factor of 0.8 with patience of 2 epochs in order to escape from local optimums. A "callback" is an object used to perform actions at various phases of training (at the start or end of an epoch, before or after a single batch). Callbacks may periodically save your model to disk, do early stopping, reduce learning rate when a metric has stopped improving, get a view on internal statistics of a model during training, among other actions. When reducing the learning rate, patient refers to the number of epochs with no improvement in the metrics after which learning rate will be reduced. The proposed networks were validated using cross-validation, confusion matrix, and accuracy, precision, sensitivity, and F1-score metrics.

Deployment

Once the models were trained and tested, they were embedded into a hybrid application using Ionic and Capacitor frameworks. Ionic is an open-source mobile framework for building high quality, cross-platform native, and web applications (hybrid apps), and Capacitor provides progressive web app (PWA) support so that users can use the exact same API when running on different targets such as iOS or Android (native apps), including elements of both web apps and native apps (Ionic, 2023). Therefore, this hybrid mobile application can be installed on the device or run via a web browser. Ionic uses a single code base and runs everywhere with JavaScript and the Web. To embed the models into Ionic, TensorFlow.js was used. TensorFlow.js is a ML library that uses out-of-the-box models in JavaScript. It can also convert TensorFlow and Python models to run in the browser. In this way, being a hybrid application, it can be installed inside the devices (including the embedded models) and executed without the need for Internet access (which is not always available in remote or poor areas) (Tensorflow, 2023). Finally, Firebase was used to deploy the Ionic hybrid application in the Web. Firebase is a powerful tool that helps you build apps fast, without managing infrastructure. Firebase is built on Google infrastructure and scales automatically, even for largest apps, so you do not need to worry about scaling your servers when necessary (Google, 2023).

Results and discussion

This research proposed a CNN-based application to provide accurate results in two (covidvs. Non-covid) and three classes (covidvs. Normal vs. Pneumonia). In both cases, the networks used a larger number of covid-19 images for training, validation, and testing. Additionally, although the training samples are unbalanced, the size of the validation sample is balanced in both scenarios. Additionally, online data augmentation and dropout in the dense and output layer were used to avoid overfitting. The number of trainable parameters was kept as small as possible, the learning rate was adjusted through callbacks to escape local optimums, and automated hyper-parameter was used to define dropout probabilities (in dense and output layers), activation functions (ELU or ReLU), and number of neurons in dense layer. Adam's algorithm was used for improving the network optimization, and the binary network was obtained through transfer learning using the three-class network as a pre-trained model.

The images used for covid-19 class are still limited, and a larger number of images could help increase performance, especially in the three-class scenario. The proposed models obtained higher performance when compared to other studies reported in the literature, requiring significantly fewer parameters and processing more images (Table 2).

Other very recent alternatives, such as that of Bosowski *et al.* (2021) (based on deep ensembles), can produce competitive results but are computationally much more expensive to be embedded in devices with limited capabilities; as well, their implementation within a platform must have a different approach, using a cloud computing platform or on-promise, which will imply high costs and Internet access. Additionally, the latency of the entire system (including the physical network) must be analyzed to provide responses in real time.

Table 2. Performance comparison of these networks with other proposals.

Reference	Number of images	Architecture/ parameters	Accuracy (%)	
			Binary	Three-classes
Sethy & Behera (2020)	25 covid-19 + 25 non-covid	ReNet-50/SVM 24.97 million	95.38	
Narin *et al.* (2020)	50 covid -19 + 50 non-covid	ResNet-50 24.97 million	98	
Loey *et al.* (2020)	69 covid -19 + 70 Normal	Alexnet/GAN 61 million Googlenet /GAN	99	85.2
	69 covid -19 + 79 Normal + 79 Viral Pneumonia + 79 Bacterial Pneumonia			
Ozturk *et al.* (2020)	125 covid -19 + 500 Normal	DarkCovidNet / 1 164 434	98.08	87.02
	125 covid -19 + 500 Pneumonia + 500 Normal			
Ioannis & Tzani (2020)	224 covid -19 + 700 Pneumonia + 504 Normal	VGG-19/ 20.37 million		93.48
Rahimzadeh & Attar (2020)	180 covid -19 +6 054 Pneumonia + 8 851 Normal	Xception and ResNet50V2 / *Unspecified*		91.4
Mahmud *et al.* (2020)	305 covid -19 + 305 Normal	Multi-resolution CovXNet / *Unspecified*	97.4	90.2
	305 covid -19 + 305 Normal + 305 Viral Pneumonia + 305 Bacterial Pneumonia			
This research	642 covid -19 + 682 NO-covid (1 460 Normal + 2 740 Viral Pneumonia + 1 460 Bacterial Pneumonia)	Convolutional neural networks 435 617 / 879 779	**98.61**	**96.48**
	642 covid -19 + **1 580** Normal + **4 260** Pneumonia (1 493 Viral + 2 767 Bacterial)			

Source: Author's own elaboration.

In addition, previous studies working with multiple scenarios, train each scenario separately. This can represent higher costs if time from a cloud computing platform such as AWS is used for training. Therefore, a transfer learning approach from the three-class scenario to binary scenario is a better option. Also, HPO process that usually involves high costs, time, and effort was automated using novel DL libraries such as Hyperas and Hyperopt, and learning rate was adjusted using callback functions to escape from local optimums. Although the number of images used in this type of research usually has bias in some category, for example, negative cases for covid-19, this is usually evaluated using the confusion matrix. Furthermore, analyzing the confusion matrix from previous works, they usually have poor performance in one category, generally in pneumonia (Ioannis & Tzani, 2020; Loey *et al.*, 2020; Mahmud *et al.*, 2020; Ozturk *et al.*, 2020). The proposed CNN have a very good performance in every category involved. Figure 4 shows the confusion matrix for both proposed models (in the validation and testing samples), and Table 3 shows the values for the selected metrics (accuracy, precision, sensitivity, and F1-score). Processing a larger number of images allowed the proposed networks to learn a greater number of patterns; online data augmentation avoided overfitting, dropout and batch normalization layers provided better generalization (Figure 5), while HPO in combination with Adam algorithm and new activation functions (such as ReLU or ELU) led to further optimization.

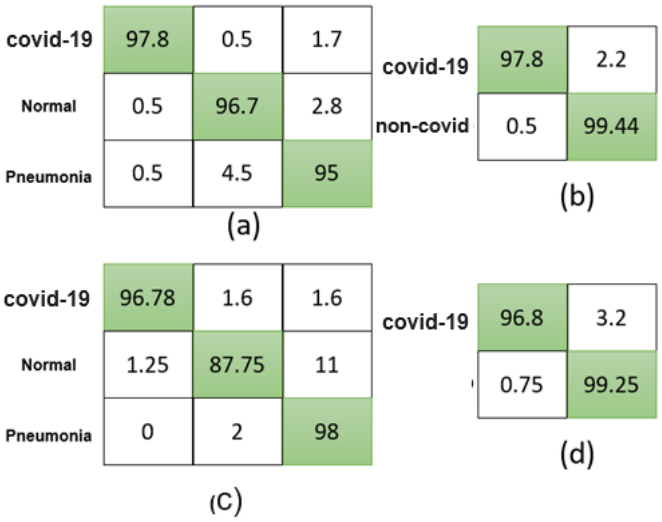

Figure 4. Confusion matrices: (a) 3-class validation scenario, (b) binary validation scenario, (c) 3-class testing scenario, and (d) binary testing scenario.
Source: Author's own elaboration.

Table 3. Performance metrics.

Dataset	Scenario	Class	Precision	Recall	F1-score	Accuracy
Validation	Binary	covid -19	0.99	0.98	0.99	98.61
		Non- covid	0.98	0.99	0.99	
	Three-class	covid -19	0.99	0.98	0.98	96.48
		Normal	0.95	0.97	0.96	
		Pneumonia	0.96	0.95	0.95	
Testing	Binary	covid -19	0.91	0.97	0.94	99.07
		Non-COVID	1.0	0.99	0.99	
	Three-class	covid -19	0.92	0.97	0.94	93.15
		Normal	0.97	0.88	0.92	
		Pneumonia	0.90	0.98	0.94	

Source: Author's own elaboration.

ISSN online 2007-9621

Table 4 shows the selected hyper-parameters using Hyperas and Hyperopt. For the binary classification scenario, the network was adjusted to only 435 617 parameters, while accuracy increased to 98.61% and 99.07% in the validation and testing samples. In the three-class scenario, the network was adjusted to only 879 779 parameters, and the accuracy increased to 96.48% and 93.15% into the validation and testing samples. These network sizes allowed to convert the models to TensorFlow JS format and embed them locally in a hybrid application (using Ionic) that can be installed and executed without the need for Internet access. Additionally, the application was deployed into Internet through Firebase to provide support in diagnostics, without managing infrastructure, saving infrastructure costs, and scaling automatically. Figure 6 presents the proposed application being executed in the three-class scenario. The application supports two languages (English and Spanish) and is available at https://xrays-covid.web.app/.

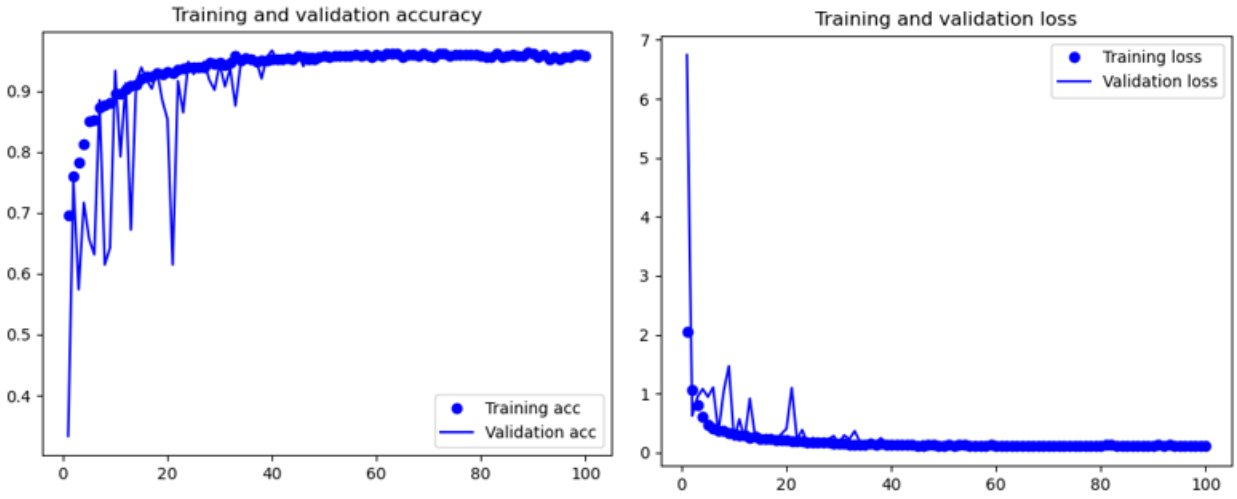

Figure 5. Three-class training and validation accuracy (left) and training and validation loss (right).
Source: Author's own elaboration.

ISSN online 2007-9621

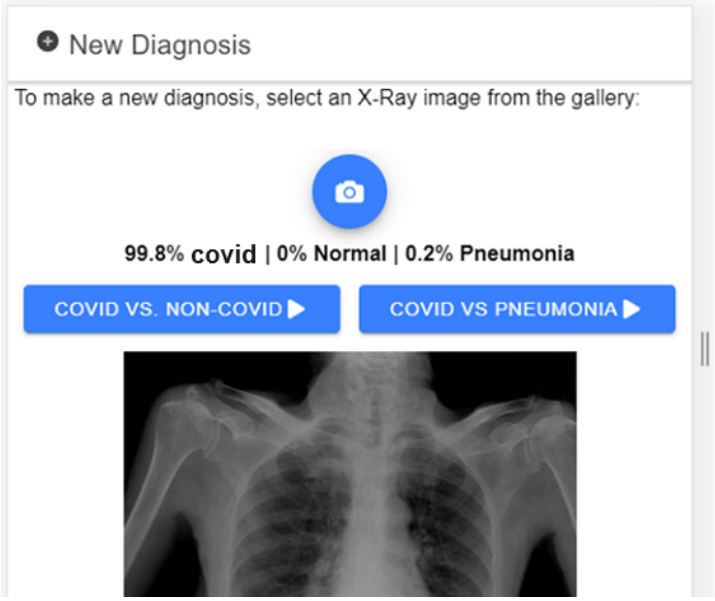

Figure 6. The proposed application.
Source: Author's own elaboration.

Table 4. Hyper-parameters obtained through Hyperas and Hyperopt.

Three-class Scenario	Binary Scenario
Activation Conv. Layer 1: **elu**	Dense Layer Size: **128**
Activation Conv. Layer 2: **elu**	Dropout (Dense Layer): **0.7628**
Activation Conv. Layer 3: **elu**	Activation (Dense Layer): **elu**
Activation Conv. Layer 4: **relu**	
Activation Conv. Layer 5: **relu**	
Activation Conv. Layer 6: **elu**	
Dense Layer Size: **512**	
Dropout (Dense Layer): **0.1124**	
Activation (Dense Layer): **relu**	
Dropout (Output Layer): **0.32**	

Source: Author's own elaboration.

Cloud computing tools (such as Firebase) offer infrastructure built using regions made up of different availability zones that include different computing centers. Widely used services like Amazon S3 offer guaranteed availability of 99.99% and durability of 99.999999999%. Using AWS CloudWatch, an event was scheduled to invoke the proposed application every three minutes. This event ran for one month (14 880 invocations) and found 100% availability.

Additionally, applications previously used redundancy to ensure sufficient capacity to respond to the maximum level of activity. The cloud allows you to achieve high flexibility and scalability levels, automatically providing the amount of resources needed, reducing costs and improving capacity to meet the demand and workload. Concurrency refers to the number of instances serving requests. When requests arrive faster compared to scalability, requests fail due to a throttling error (http code 429). To test the scalability of this application, the artillery.io tool was used, carrying out a simulation of five new users accessing the application (load test) every second for 10 min, resulting in 3000 invocations and 100% successful executions.

Conclusions

DL models can be widely used in many areas, being dependent on the existence of enough data for training. When the amount of data available is scarce, there are some techniques that allow improving the performance of the models such as data augmentation, the use of pre-trained models, regularization techniques (such as dropout and batch normalization) to avoid overfitting, and new and improved training algorithms (like Adam and RMSProp) and activation functions (such as ReLU and ELU). This research proposed a CNN-based application to provide assistance support in diagnosing two (covidvs. Non-covid) and three classes (covidvs. Normal vs. Pneumonia), reaching an accuracy of 98.61% and 96.48% for two and three classes in the validation dataset and 99.07% and 93.15% in the testing dataset, obtaining higher performance when compared to previous studies and using a significantly lower number of parameters. The architecture used into these networks increased the classification performance using state-of-the-art concepts such as dropout, batch normalization, and online data augmentation to avoid overfitting, transfer learning, automated hyper-parameter optimization, learning rate adjustment through callbacks to escape local optimums, new activation functions (such as ELU or ReLU), and Adam optimizer. In addition, the amount of data for training, validation, and testing samples was considerably increased, compared to other works reported in the literature. This application can be widely used to help diagnose since X-ray images are widely used for the diagnosis of respiratory diseases and lung abnormalities such as covid-19. In the future, CNN used in this research can be validated against a greater number of images and include other categories (other lung diseases such as tuberculosis, lung cancer, ARDS, among others). They can even be used as a base architecture or pre-trained models in future investigations. The size of these CNN allowed them to be converted to TensorFlow JS format and be embedded locally in a hybrid application that can be installed and executed without the need for Internet access (which is not always available in remote or poor areas). Additionally, the application was deployed into the Internet through Firebase to provide assistance to health personnel, without managing infrastructure, saving infrastructure costs, and scaling automatically. This application is online at https://xrays-covid.web.app/ and two languages are available (English and Spanish).

Conflict of interest

The authors declare that there is no conflict of interest with the preparation of the article.

References

American College of Radiology (ACR). (March 11, 2020). *ACR Recommendations for the use of chest radiography and computed tomography (CT) for suspected covid-19 infection*. https://www.acr.org/Advocacy-and-Economics/ACR-Position-Statements/Recommendations-for-Chest-Radiography-and-CT-for-Suspected-covid19-Infection

Beysolow II, T. (2017). *Introduction to deep learning using R. A step-by-step guide to learning and implementing deep learning models using R*. Apress.

Bosowski, P., Bosowska, J., & Nalepa, J. (2021). Evolving deep ensembles for detecting covid-19 in chest X-Rays. *IEEE International Conference on Image Processing (ICIP),* 3772-3776. https://doi.org/10.1109/ICIP42928.2021.9506119

Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co.

Chung, M., Bernheim, A., Mei, X., Zhang, N., Huang, M., Zeng, X., Cui, J., Xu, W., Yang, Y., Fayad, Z. A., Jacobi, A., Li, K. S., & Shan, H. (2020). CT Imaging features of 2019 novel coronavirus (2019-nCoV). *Radiology, 295*(1). https://doi.org/10.1148/radiol.2020200230

Chung, A. (2020). *Actualmed* covid-*19 chest x-ray data initiative*. https://github.com/agchung/Actualmed-covid-chestxray-dataset

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). *Johannes Kepler University*, Linz, Australia, pp. 1-14. arXiv:1511.07289v5

Cohen, J., Morrison, P., & Dao, L. (2020). covid-*19 image data collection: prospective predictions are the future*. https://github.com/ieee8023/covid-chestxray-dataset

European Society of Radiology (ESR). (2021). [*Eurorad*]. https://www.eurorad.org/advanced-search?search=covid&sort_by=published_at&sort_order=DESC&page=5

Github. (2020). covid-*19 Chest X-Ray Dataset Initiative*. https://github.com/agchung/Figure1-covid-chestxray-dataset

Google. (2023). *Firebase*. https://firebase.google.com/

Gutierrez, A., Ansuategi, A., Susperregi, L., Tubío, C., Rankic, I., & Lenza, L. (2019). A benchmarking of learning strategies for pest dete and identification on tomato plants for autonomous scouting robots using internal databases. *Journal of Sensors, 2019*(1), 1-16. https://doi.org/10.1155/2019/5219471

Ioannis, D. A., & Tzani, B. (2020). covid-*19: automatic detection from X-Ray images utilizing transfer learning with convolutional neural networks*. arXiv:2003.11617

Ionic. (2023). *Cross-platform mobile App development: Ionic framework*. https://ionicframework.com/

Jacobi, A., Chung, M., Bernheim, A., & Eber, C. (2020). Portable chest X-ray in coronavirus disease-19 (covid-19): a pictorial review. *Clinical Imaging, 64*, 35-42. https://doi.org/10.1016/j.clinimag.2020.04.001

Kaggle Inc. (2023). *Chest X-Ray images (Pneumonia)*. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Loey, M., Smarandache, F., & Khalifa, N. E. M. (2020). Within the lack of chest covid-19 X-ray dataset: a novel detection model based on GAN and deep transfer learning. *Symmetry, 12*(4), 1-19. https://doi.org/10.3390/sym12040651

Mahmud, T., Rahman, M. A., & Fattah, S. A. (2020). CovXNet: a multi-dilation convolutional neural network for automatic covid-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization. *Computers in Biology and Medicine, 122*, 103869. https://doi.org/10.1016/j.compbiomed.2020.103869

Narin, A., Kaya, C., & Pamuk, Z. (2020). *Automatic detection of coronavirus disease (*covid-*19) using X-ray images and deep convolutional neural networks*. arXiv:2003.10849

Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., & Yildirim, O. (2020). Automated detection of covid-19 cases using deep neural networks with X-ray images. *Computers in Biology and Medicine, 121*, 1-11. https://doi.org/10.1016/j.compbiomed.2020.103792

Pedamonti, D. (2018). *Comparison of non-linear activation functions for deep neural networks on MNIST classification task*. arXiv:1804.02763v1

Radiopaedia. (2021). [covid-19]. Radiopaedia.org. https://radiopaedia.org/articles/covid-19-4?lang=us

Rahimzadeh, M., & Attar, A. (2020). A modified deep convolutional neural network for detecting covid-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2. *Informatics in Medicine Unlocked, 19*, 1-9. https://doi.org/10.1016/j.imu.2020.100360

Rong, G., Mendez, A., Assi, E. B., Zhao, B., & Sawan, M. (2020). Artificial Intelligence in Healthcare: review and prediction case studies. *Engineering, 6*(3), 291–301. https://doi.org/10.1016/j.eng.2019.08.015

Rosebrock, A. (2017). *Deep learning for computer vision with Python*. PyImageSearch.

SAS Institute Inc. (2023). *Machine learning: What it is and why it matters SAS*. https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-importance

Sethy, P. K., & Behera, S. K. (2020). *Detection of coronavirus disease (*covid-*19) based on deep features*. Preprints.org. https://doi.org/10.20944/preprints202003.0300.v1

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). *Striving for simplicity: the all convolutional net*. arXiv:1412.6806v3

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from over. *Journal of Machine Learning Research, 15*, 1929-1958.

Tensorflow. (2023). *TensorFlow.js | Machine Learning for JavaScript Developers*. TensorFlow.org. https://www.tensorflow.org/js

The Italian Society of Medical and Interventional Radiology (SIRM). (2020). [*covid-19 database*]. https://www.sirm.org/en/category/articles/covid-19-database/

Wang, J., Ma, Y., Zhang, L., Gao, R., & Wu, D. (2018). Deep learning for smart manufacturing: methods and applications. *Journal of Manufacturing Systems, 48*, 1-13. https://doi.org/10.1016/j.jmsy.2018.01.003

Wang, L., Lin, Z. Q., & Wong, A. (2020). covid-*Net: a tailored deep convolutional neural network design for detection of* covid-*19 cases from chest X-Ray images*. arXiv:2003.09871

World Health Organization (WHO). (June 06, 2020). *Coronavirus disease (covid-19) pandemic*. https://www.who.int/emergencies/diseases/novel-coronavirus-2019

World Health Organization (WHO). (2022). *Coronavirus*. https://www.who.int/health-topics/coronavirus#tab=tab_1

Wuest, T., Weimer, D., Irgens, C., & Thoben, K. D. (2016). Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research, 4*(1), 23-45. https://doi.org/10.1080/21693277.2016.1192517

Yu, T., & Zhu, H. (2020). *Hyper-Parameter optimization: a review of algorithms and applications*. arXiv:2003.05689

Zhou, S., Wang, Y., Zhu, T., & Xia, L. (2020). CT features of coronavirus disease 2019 (covid-19). *American Journal of Roentgenology, 214*(6), 1287-1294. https://doi.org/10.2214/AJR.20.22975

Carlos Eduardo Belman López

Design of an application to detect covid-19 using convolutional neural networks and X-ray images
Diseño de una aplicación para detectar covid-19 mediante redes neuronales convolucionales e imágenes de rayos X

Acta universitaria
vol. 34, e3919, 2024
Universidad de Guanajuato, Dirección de Investigación y Posgrado,
ISSN: 0188-6266
ISSN-E: 2007-9621

DOI: https://doi.org/10.15174/au.2024.3919