Computer Science and Information Systems 19(1):327–352 https://doi.org/10.2298/CSIS210209056V

Hyper-parameter Optimization of Convolutional Neural Networks for Classifying COVID-19 X-ray Images[⋆]

Grega Vrbančič[⋆⋆], Špela Pečnik, and Vili Podgorelec

University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, SI-2000 Maribor, Slovenia
{grega.vrbancic, spela.pecnik, vili.podgorelec}@um.si

Abstract. For more than a year the COVID-19 epidemic is threatening people all over the world. Numerous researchers are looking for all possible insights into the new corona virus SARS-CoV-2. One of the possibilities is an in-depth analysis of X-ray images from COVID-19 patients, commonly conducted by a radiologist, which are due to high demand facing with overload. With the latest achievements in the field of deep learning, the approaches using transfer learning proved to be successful when tackling such problem. However, when utilizing deep learning methods, we are commonly facing the problem of hyper-parameter settings. In this research, we adapted and generalized transfer learning based classification method for detecting COVID-19 from X-ray images and employed different optimization algorithms for solving the task of hyper-parameter settings. Utilizing different optimization algorithms our method was evaluated on a dataset of 1446 X-ray images, with the overall accuracy of 84.44%, outperforming both conventional CNN method as well as the compared baseline transfer learning method. Besides quantitative analysis, we also conducted a qualitative in-depth analysis using the local interpretable model-agnostic explanations method and gain some in-depth view of COVID-19 characteristics and the predictive model perception.

Keywords: COVID-19, classification, CNN, transfer learning, optimization.

1. Introduction

Not much more than a year since December 2019, when in Wuhan city, the capital of Hubei province in China, the cases of ”unknown viral pneumonia” started to gather, the world is witnessing a huge spread of coronavirus disease 2019 (COVID-19) caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Based on the World Health Organization report published on the 2nd of Februar 2021, there were more than 102 million confirmed cases and more than 2.2 million deaths globally, spreading across 220 countries and territories [34].

Currently, one of the mostly used method globally for detecting a COVID-19 disease is using the real-time transcription-polymerase chain reaction (RT-PCR) test [35]. However, the sensitivity of such method ranges around 70%, while the alternative methods using CT or X-ray imaging can achieve significantly better performance, up to 98% [14]. While such methods can provide us with better sensitivity performance, the main bottleneck is that analysing such imaging requires an experienced radiologist, who manually,

[⋆] This is an extended version of a INISTA 2020 conference paper.
[⋆⋆] Corresponding author

visually scans such images trying to detect some pathology. This bottleneck especially in current situation comes to the fore, when a large number of such imaging should be analyzed in very short time, and thus increasing the probability of miss-classification and putting large amount of stress on medical staff. In those terms the use of advanced machine learning approaches for classification of images radiography imaging can be justified.

With the advancements of deep learning methods and techniques in recent years, especially the ones utilizing convolutional neural networks (CNNs), various research works proved that the application of such methods against the medical domain problems is resulting in encouraging results [25]. In the last year, there were large amount of researches published, focusing on applying the machine learning algorithms to identification of COVID-19. One of the most common approaches to tackle the mentioned issue is to utilize the transfer learning approach as presented in [2, 26].

While such approaches enable us to successfully train a predictive model, we are still faced with a major problem common to all training approaches of deep neural networks – setting the values of hyper-parameters [52] also known as hyper-parameter optimization (HPO). Setting the appropriate values of hyper-parameters for the process of training has a direct impact on the final predictive performance of such models, therefore the values should be carefully chosen. While commonly this is still a manual process, a great amount of research was put into developing automatic methods [38, 43, 49], which would take care of this problem. Since many studies have addressed the problem of identifying a COVID-19 from X-ray images and since the chosen hyper-parameter values have a direct impact on the final classification performance, it is crucial to set hyper-parameter values appropriately especially when addressing such sensitive problem.

Based on our previous experience with the identification of COVID-19 [43], promising results from similar studies [3, 36] and our previous work on solving HPO problem [38,43], we set our goal to generalize our GWOTLT [43] from our previous research, in which we utilized the grey wolf optimizer (GWO) algorithm to find the most suitable values of hyper-parameters, to make it agnostic to the usage of different optimization algorithms. Such a generalized HPO method for transfer learning (HPO-TL) enables us to employ various optimization algorithms in order to find the most suitable values of hyper-parameters in order to achieve the best possible predictive model utilizing transfer learning. Beside providing predictive model using the HPO-TL method and evaluating the performance of such models from a quantitative standpoint, we also conducted an analysis of interpretable representations of our model using local interpretable model-agnostic explanations (LIME) method. To gain useful insights on how the model perceives the chest X-ray images, evaluating the model's decisions from a qualitative perspective, we took a different approach where multiple interpretable representations obtained by LIME were aggregated into one single representation, which could enable us to gain different insights into perception of predictive model.

We can sum up our main contributions presented in this research as follows:

– We generalize GWOTLT method in order to make it optimization algorithm agnostic, which enables us to use various optimization algorithms for the task of hyper-parameter optimization.

- We conducted an empirical evaluation of the generalized HPO-TL method with three optimization algorithms (GWO, DE, GA), tackling the problem of detecting COVID-19 from X-ray images.
- We conducted an extensive performance analysis and comparison against the conventional approaches of training the predictive CNN model.
- We performed a qualitative analysis of the predictive model using the LIME method.

The remaining of the paper is structured as follows. In section 2, a brief review of related work is presented. Utilized methods and generalized HPO-TL method are presented in section 3, while in section 4 the experimental framework is described. In section 5 the results of conducted experiments are presented and interpreted, while section 6 presents the conclusions.

2. Related work

So far, many analyses have been performed using convolutional neural networks over chest X-ray images of patients, which try to help better identify COVID-19 cases. For example, Apostolopoulos and Mpesiana [2] were among the first to evaluate the performance of CNNs using transfer learning over a collection of images showing COVID-19 condition, pneumonia, or a normal condition. They found that in this way we could extract significant biomarkers related to the COVID-19 disease with great accuracy (above 96%). The use of eight different pre-trained CNNs over a dataset of normal and COVID-19 cases was also used in [32], where the authors report that the best model achieved an accuracy of up to 98%. Marques et al. [28] proposed a medical decision support system based on CNN with EfficientNet architecture. The built model was used for both binary classification and multiclass classification. In the case of binary classification, X-ray images of COVID-19 positive patients and healthy patients were used. For multiclass classification, images of patients with pneumonia were added to the dataset. The results showed that better values of different metrics are achieved in binary classification. S. Govindarajan and R. Swaminathan [18] acquired critical image features using CNN with several different hyper-parameter settings and cross-validation methods. They visualized them using occlusion sensitivity maps. The resulting images showed some localized abnormal regions, which indicate COVID-19. In [21], the authors conducted a study on images obtained from portable chest X-ray (pCXR), which included two types of pneumonia, the normal condition and the COVID-19 condition. CNN with transfer learning was used over whole pCXR and over segmented lungs. Better results were obtained over segmented lungs (accuracy 88%) than over whole pCXR (accuracy 79%). Majeed et al. [27] used 12 CNN architectures with transfer learning and a shallow CNN architecture which they trained from scratch. The X-ray images were also not preprocessed before the use. The parts of the images that were supposed to influence the decision of the model were visualized by class activation maps (CAMs), which, according to their findings, are not reliable, as they indicate parts that are not characteristic for COVID-19 disease.

As we can see, the use of CNNs with transfer learning is very common in this problem area. Differences can be found in the optimization of the algorithms, the parameter settings and used datasets. In the original, our research differs from the existing ones in that we used a dataset to predict COVID-19 status, which contains X-ray images of the chest of COVID-19 patients and images showing a normal condition or any other respiratory disease. So our main purpose was to predict whether it is a COVID-19 case or some other condition.

3. Methods

Since the first introduction of CNNs in the 1980s [16], the remarkable progress has been made in the image recognition field especially due to the availability of large annotated datasets, development of various deep CNN architectures and increased computational capabilities. The CNNs or more precisely the convolutional layers leverage three important ideas that can help improve a machine learning system: sparse interaction, parameter sharing and equivariant representations. In contrast to the traditional neural network layers which use matrix multiplication by a matrix of parameters with a separate parameter describing the interaction between each input unit and each output unit, the CNNs, however, typically have sparse interactions, also known as sparse connectivity or sparse weights. The sparse interactions are achieved by making a kernel smaller than the input, which on the one side enables us to detect small, meaningful features with kernels that occupy only tens of pixels, while on the other side reduces the memory consumption of the models and improves its statistical efficiency, since we need to store fewer parameters. Additionally, the use of parameter sharing in CNNs also increases the memory and statistical efficiency in comparison to the traditional neural network, where each element of the weight matrix is used exactly once when computing the output of a layer. Furthermore, in the case of convolution, the particular form of parameter sharing causes the layer to have a property called equivariance. Basically, equivariance enables convolution to create a 2-dimensional map of where certain features appear in the input. If the object in the input is moved, its representation will also move for the same amount [17].

Those capabilities make the CNNs de facto standard for solving the image recognition tasks in various domains from medicine [45, 48], information security [19] to seismology [22] or even agriculture [20]. However, training such CNN models requires a large amount of labeled data, which can be in certain fields, especially in medicine, a challenging task. To overcome the lack of sufficient labeled dataset, one of the commonly used methods is transfer learning with fine-tuning, which enables us to adapt a pre-trained model to our domain problem, without requiring a large dataset.

3.1. Transfer Learning

The first appearances of transfer learning in publications are dating back to the 1995 [6], mostly under different names such as inductive transfer [12], incremental or cumulative learning [55], and multitask learning [51], the latter one being the most closely related to the transfer learning as we know it today. In the most broader terms, the transfer learning technique can be defined as the improvement of learning a new task through the transfer of knowledge from a related task which has been already learned. However, in the machine learning terms, the transfer learning can be defined as transferring the weights of an already trained predictive model, specialized for a specific task, to the new model addressing similar but not the same task.

There are many different techniques on how to utilize the transfer learning, one of the most commonly used being the fine-tuning. When utilizing the fine-tuning approach to transfer learning, we are transferring the weights from a pre-trained CNN to the new one [41]. Commonly, we only transfer the weights in the so-called convolutional base of CNN architecture, which is composed of a sequence of convolutional layers and pooling layers, since those layers' weights contain general feature extraction capabilities. In general, the bottom layers (more towards the input) of the CNN tend to extract more abstract, generally applicable features than the top layers (more toward the output), which tend to extract more task-specific features. Therefore, when utilizing a fine-tuning technique, most commonly we only fine-tune (train) the layers more towards the top of the CNN architecture and leave the bottom ones frozen (disabled for training) [41].

Regardless of the benefits of the transfer learning with fine-tuning, such approach still has some challenges common to the traditional approach of training CNN. One of such problem is the selection of training parameters also known as hyper-parameters. Setting appropriate value for hyper-parameters such as learning rate, batch size, optimization function, etc. directly reflects on how well the model is capable to train and consequently impacts the model classification performance.

3.2. Hyper-parameter Optimization for Transfer Learning

The problem of setting the right values for the hyper-parameters is also known as hyper-parameter optimization (HPO) task. Most commonly are such tasks addressed with the Gaussian Process approach, Tree-structured Parzen Estimator approach or Random search approach [4]. But in recent years, population-based metaheuristic algorithms are becoming more and more popular in successfully solving HPO problems [23, 46, 49].

Based on our previous success with utilization of various optimization algorithms for the purpose of optimizing hyper-parameter values [38, 47], we decided to generalize our Grey Wolf Optimizer for Transfer Learning Tuning (GWOTLT) method presented in [49] to make it work and evaluate it with other popular metaheuristics, such as genetic algorithm (GA) and differential evolution (DE). The basic concept of our generalized Hyper-parameter optimization for transfer learning (HPO-TL) method can be generally defined in the following steps:

1. Optimization algorithm produces the solution.
2. Solution is mapped to the values of sought hyper-parameters.
3. CNN with mapped hyper-parameter values is trained.
4. The solution is evaluated, calculating fitness value.
5. Fitness value is being passed back to the optimization algorithm.

Those steps are then being executed in an iterative manner, for the given number of function evaluations.

The HPO-TL is producing a solution with the same number of elements as is the number of sought hyper-parameter values. In our case the dimension of the produced solution is 4, since we are searching for the most optimal value of four different hyper-parameters, namely: learning rate, optimizer function, dropout probability of dropout layer, and the number of neurons in the last fully-connected (dense) layer. Formally, the individuals of such HPO-TL produced solutions are presented as a real-valued vectors:

$$\mathbf{x}_i^{(t)} = (x_{i,0}^{(t)}, \ldots, x_{i,n}^{(t)}), \quad \text{for } i = 0, \ldots, Np - 1, \tag{1}$$

where each element of the solution is in the interval $x_{i,1}^{(t)} \in [0, 1]$. These real-valued vectors (solutions) are then mapped to the used hyper-parameter values as defined in equations 2 to 5, where y_1 denotes the number of neurons in the last fully connected layer, y_2 denotes dropout probability, y_3 denotes optimization function and y_4 denotes learning rate. Each y_1 value is mapped to the particular member of the population $N = \{64, 128, 256, 512, 1024\}$ according to the members position in the population, which represents a group of available numbers of neurons in the last fully connected layer. All of the y_3 values are mapped to the specific member of population $O = \{adam, rmsprop, sgd\}$, which represents a group of available optimizer functions, while each y_4 value is mapped to the member of population $L = \{0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$, which represents a group of learning rate choices.

$$y_1 = \begin{cases} \lfloor x[i] * 5 + 1 \rfloor; y_1 \in [1, 5] & x[i] < 1 \\ 5 & \text{otherwise}, \end{cases} \tag{2}$$

$$y_2 = x[i] * (0.9 - 0.5) + 0.5; y_2 \in [0.5, 0.9] \tag{3}$$

$$y_3 = \begin{cases} \lfloor x[i] * 3 + 1 \rfloor; y_3 \in [1, 3] & x[i] < 1 \\ 3 & \text{otherwise}, \end{cases} \tag{4}$$

$$y_4 = \begin{cases} \lfloor x[i] * 5 + 1 \rfloor; y_4 \in [1, 5] & x[i] < 1 \\ 5 & \text{otherwise}, \end{cases} \tag{5}$$

The training utilizing the fine-tuning with mapped hyper-parameter values is then being conducted in a straight-forward manner where the last block of used CNN architecture is being fine-tuned while the other (more bottom) layers remain frozen. After the training is finished, the fitness values are being calculated. We defined the fitness value as:

$$f(sol) = 1 - AUC(sol) \tag{6}$$

where sol denotes the model trained with hyper-parameters set based on the obtained HPO-TL solution, and AUC defines the area under the ROC curve metric.

The fitness value is then being passed back to the chosen optimization algorithm, based on which the new solution will be produced.

3.3. HPO-TL variations

Since our presented HPO-TL method is designed to work with various optimization algorithms, we selected three of the most popular algorithms to showcase the advantages of such an approach where the method is not conceptually bonded to a particular algorithm. For this purpose, we utilized a grey wolf optimizer algorithm, which is in recent works [15, 43, 49] showing a great performance solving various tasks, differential evolution which is still dominating in various solutions [5, 9, 50], and one of most conventional nature-inspired evolutionary algorithms – genetic algorithm.

Grey Wolf Optimizer variation (GWO-TL) is based on the GWO algorithm [31], which is one of the most popular representatives of nature-inspired population-based metaheuristic algorithms. The GWO is inspired from a strict leadership hierarchy and hunting mechanisms of grey wolves (Canis lupus). As defined by authors in [31], there are three main phases of grey wolves hunting. First one is tracking, chasing and approaching the prey, the second one is pursuing, encircling and harassing the prey until it stops moving, and final third phase is the attack toward the prey. In GWO, we consider the fittest solution as the alpha, therefore consequently, the second and third-best solutions are named beta and delta. Other candidate solutions are assumed to be omega. In general, the search process starts by creating a random population of grey wolves in the GWO algorithm. In each iteration alpha, beta, and delta candidate solutions estimate the probable position of the prey. The parameter a is decreased from 2 to 0, to emphasize the exploration and exploitation, respectively. In each iteration the candidate solutions tend to converge to the prey when vector A, which is mathematically modeling divergence, is decreasing below 1 and diverge from the prey when A is increasing above 1 [31].

Differential Evolution variation (DE-TL) is based on arguably one of the most powerful and versatile evolutionary optimizers in recent times. Standard DE algorithm consists of four basic steps: initialization, mutation, recombination or crossover, and selection. From those steps, only the last three are repeated into the subsequent DE iterations [9]. In the initialization phase, Np real-value coded vectors are randomly initialized. Each such vector is also known as genome or chromosome and forms a candidate solution. After initialization, DE creates a donor or mutant vector corresponding to each population member in the current iteration with utilization of mutation. In order to increase the diversity of the parameter vectors of DE, the crossover step is conducted, where CR parameter controls the fraction of parameters that are copied to the candidate solution. Finally, the selection step is executed in which the decision whether a produced candidate solution should become a generation member is made, using the greedy criterion [9]. Those three steps are being repeated until stopping condition is not reached.

Genetic Algorithm variation (GA-TL) is based on the one of the first population-based stochastic algorithm. Similar to the other evolutionary algorithms, the main steps of the GA are selection, crossover, and mutation [30]. In the same manner as the DE, GA starts with a random population, which represents chromosomes of individual candidate solutions. Nature is the main inspiration for the selection step in GA algorithm, which is trying to mimic the phenomena where the fittest individuals have a higher chance of getting food and mating. For this purpose GA is employing a roulette wheel to assign probabilities to individuals and select them for creating the next-generation proportional to their objective. In the crossover step the selected individuals are being combined producing new solutions in GA algorithm. In the last step, mutation is conducted, in which one or multiple genes of created new solutions are altered. This step in GA maintains the diversity of population by introducing another level of randomness [30]. The algorithm iterates those three steps in the same manner until the stopping condition is not reached.

3.4. Local Interpretable Model-agnostic Explanations

Many times, in the world of machine learning, it is not enough just to build a good decision model, its success is also influenced by how decision makers understand and trust its predictions. This is especially important in more sensitive domains, such as medicine. Decision-makers' confidence in model results usually increases when they have a clear insight into what influenced the model's decision, what is its behavior and what are the possible errors. For this purpose, various interpretive methods have been developed. Some of them are also able to give an explanation for a model built over unstructured data (in our case images). [39].

The interpretive method we used in our case is the Local Interpretable Model-Agnostic Explanations (LIME) method, which was first introduced in 2016 by Ribeiro et al. [39]. An interpretive method that also allows the interpretation of models built above images is the SHapley Additive exPlanations (SHAP) method, which is, in addition to LIME, considered as one of the most widely used methods of this kind. LIME creates an explanation for an individual input prediction by sampling its neighboring inputs and builds a sparse linear model based on the predictions of these inputs. The most relevant features for a specific prediction are then those that have the highest coefficient calculated in this linear model [54]. One of the main advantages of the algorithm behind the LIME method is that it can explain the predictions of any black box classifier with two or more output classes. The condition for its operation is that the classifier implements a function that accepts a set of classes and then returns the probabilities for each class. The main goal of the algorithm is to identify an interpretive model over an interpretative representation that is locally faithful to the classifier. In our case or in general when working with images, the interpretative representation is a binary vector that indicates the presence or absence of neighboring sets of similar pixels, while the classifier can display the image as a tensor with three color channels per pixel.

As mentioned earlier, LIME explanation is based on the sampling of neighboring inputs of the selected input x and their outputs, while returning as a result a model g from the class of potential interpretive models G according to the following formula:

$$\arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g). \tag{7}$$

If we explain the formula in more detail, then we can say that x represents the input for which we want to know on the basis of which value was determined to belong to the selected class, f denotes the built model that we want to explain and π_x denotes the probability distribution around x. With $\Omega(g)$ we mark the complexity of model interpretation, that is opposite to its interpretability. Not every model is simple enough to be interpretive. The part of the equation $\mathcal{L}(f, g, \pi_x)$ tells us how the values of g approach the values of f at the location defined by π_x. If we want to achieve high interpretability, this value must be as low as possible [39].

4. Experiments

To objectively evaluate the COVID-19 image classification results, we conducted the following experiments:

– **base**, where the CNN is trained in a conventional manner without pre-training,
– **TL**, where transfer learning methodology is utilized, and
– three **HPO-TL** methods: TL-GWO, TL-GA, and TL-DE experiments where our proposed method is used.

All conducted experiments were implemented in Python programming language with the support of following libraries: scikit-learn [37], Pandas [29], Numpy [42], NiaPy [44], Keras [7] and Tensorflow [1].

Experiments were performed using the octa-core Intel CPU, 64 GB of RAM, and two Nvidia Tesla V100 GPUs each with dedicated 32 GB of memory.

4.1. Datasets

Almost a year after we published our previous work on COVID-19 [43], the COVID-19 dataset initially prepared by Cohen et al. [8] was greatly enlarged by various researchers from all over the world. Different contributors provided additional COVID-19 and other chest x-ray images, performed double-checking for potential labeling errors, and improved the dataset both in terms of quality and quantity. Therefore, for the purpose of evaluating the proposed methods, we obtained an updated version of the COVID-19 dataset, which in current state, on January 20th 2021, consists in total of 929 chest x-ray images.

Since the chest X-ray images are collected from various sources, the image size and format are varying. In Figure 1 are presented two samples from each of the target classes.

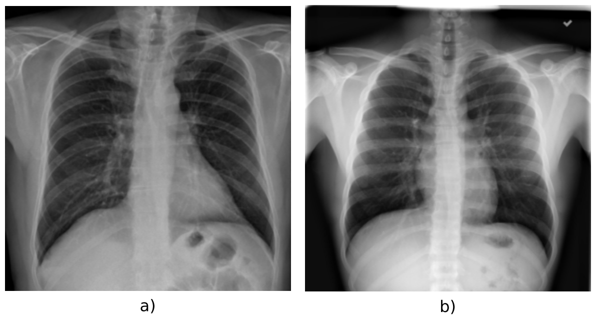

Fig. 1. Examples of X-ray images, where a) represents a COVID-19 case image, while b) represents an image with other or no pathology identified.

Inspecting the obtained dataset more in-depth, we can see that the majority of the collected chest x-ray images are labeled as an COVID-19 instances, as presented in Table 1. Comparing the number of classes in the updated version of COVID-19 dataset in comparison to the older version, we can see a significant increase. Additionally, in the updated version of the dataset we can see that one of the classes in labeled as ”todo”, which means that the instances with such label are not yet classified. Therefore, we removed instances with such label in order to avoid having some instances miss-classified and consequently training the predictive model with wrong labeled chest x-ray images. This way we ended up with the total of 846 instances, 563 of them being labeled as ”COVID-19” and the remaining 283 labeled as ”other”.

Table 1. Target class distribution of updated COVID-19 image data collection.

Class	**COVID-19 image data collection**
COVID-19	563
Pneumonia	81
SARS	16
Pneumocystis	30
Streptococcus	22
No finding	22
Chlamydophila	3
E.Coli	4
Klebsiella	10
Legionella	10
Unknown	1
Lipoid	13
Varicella	6
Bacterial	4
Mycoplasma	11
Influenza	5
todo	83
Tuberculosis	18
H1N1	2
Aspergilliosis	2
Herpes	3
Aspiration	1
Nocardia	8
MERS-CoV	10
MRSA	1
Total	929

Similar as we did in our previous research [43], with the older version of COVID-19 dataset, we have extended the updated version. Additional 600 randomly selected ”Normal” labeled chest images from RSNA Pneumonia Detection Challenge [33] were added to the existing ”other” labeled chest x-ray images, which resulted in a final updated and extended version of COVID-19 dataset with properties presented in Table 2.

Table 2. Target class distribution of an updated and extended COVID-19 image data collection.

Class	**Extended COVID-19 image data collection**
COVID-19	563
Other	883
Total	1446

4.2. Data Pre-processing

As are the images in the COVID-19 image data collection in various sizes, we applied the image resizing to uniform target size of 224 x 224 pixels, which is in line with default input size of the selected VGG19 CNN architecture. Additionally, in the train time, we applied an image augmentation technique, to prevent the over-fitting which commonly occurs when dealing with pre-trained complex CNN architecture and relatively small datasets.

The image augmentation in train time is conducted in a manner where each training instance is randomly manipulated e.g. rotated, zoomed, shifted, flipped, etc. within the given value range. The complete list of utilized augmentation parameters and its values can be observed in Table 3. The value for rotation range specifies the degree range for random rotation, while the values for width shift and height shift range specifies the fraction of a total image size for corresponding dimension. Shear range value defines a shear intensity – the shear angle (in radians) in counter-clockwise direction and zoom range value specifies the randomly selected zoom between the lower and upper bounds defined as $1 - zoom\_range$ and $1 + zoom\_range$ respectively. Lastly, the horizontal flip value defines whether each image instance can be randomly flipped horizontally or not.

Table 3. Utilized image augmentation parameter settings.

Parameter	**Value**
Rotation range	5
Width shift range	0.1
Height shift range	0.1
Shear range	0.1
Zoom range	0.1
Horizontal flip	True

4.3. CNN Setup

For the deep CNN architecture, we adapted a well known VGG19 architecture presented by Simonyan et al. in 2014 [40]. As presented in Figure 2, we left the convolutional base (blocks from 1 to 5) of VGG19 as it was presented originally, while the classifier part of the architecture was customized. Instead of a flatten layer, we utilized a 2-dimensional global average pooling layer, followed by a dropout layer, fully connected layer with ReLU activation function and fully connected output layer with sigmoid activation function.

The dropout probability values for the base and TL experiments were set to 0.5, while the dropout value for the experiments utilizing the HPO-TL methods is being optimized (set) by the method itself. The number of units in fully connected layer, followed by the dropout layer, was for the base and TL experiments set to 256, while the number of units for HPO-TL based experiments are also being optimized by the method itself.

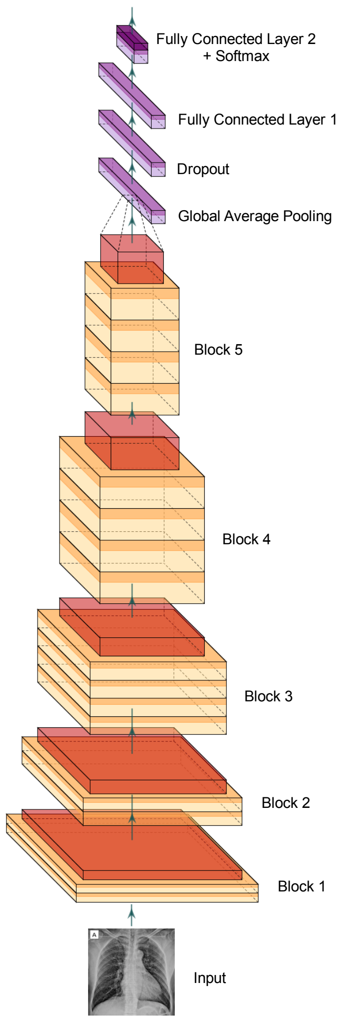

Fig. 2. The adapted VGG19 convolutional neural network architecture.

For the TL and HPO-TL based experiments, the transfer learning was utilized. The VGG19 convolutional base was pre-trained on the ImageNet dataset, while for the fine-tuning we enabled only the last convolutional block (block 5). The rest of the layers in convolutional base remained frozen (disabled for fine-tuning).

4.4. Settings of HPO-TL methods

Since the utilized HPO-TL based methods work in an iterative manner, where the next produced solution is based on fitness of the previous one, we tailored the dataset split methodology in order to retain the fairness between the compared approaches. While the base and TL experiments consume the whole training split of the dataset for the training purpose, we additionally divided the given training set in ratio 80:20, where the larger subset was used for training different solutions produced by a HPO-TL based method and evaluating them – calculating the fitness value against the remaining smaller subset of the initial training set.

For each fold, the method generates and evaluates 50 different solutions, from which the best – the one with the best (lowest) fitness value is selected and finally evaluated against the test split of the dataset. While this approach makes the such method computationally complex, we also introduced the early stopping approach to the evaluation of each solution, where the solutions which training is not improving for 5 consecutive epochs is prematurely stopped.

Table 4 presents parameter settings of three utilized optimization algorithms, namely grey wolf optimizer, differential evolution, and genetic algorithm, which were used together with the HPO-TL method. Other than population number NP parameter, all parameter values are set to default values as are defined in the NiaPy framework, from which we utilized the implementations of the selected optimization algorithms.

Table 4. Parameter settings for used optimization algorithms.

Parameter	**Value**		
	Grey Wolf Optimizer	**Differential Evolution**	**Genetic Algorithm**
Population NP	10	10	10
Scaling factor F	-	1	-
Crossover rate CR	-	0.8	0.25
Mutation rate MR	-	-	0.2

4.5. Training Parameter Settings

Presented in Table 5 are utilized training parameter settings for each of the conducted experiments. For each fold every method is provided with the total of 50 epochs, except the HPO-TL methods which, in worst-case scenario, consume a total of 2500 epochs (50 epochs for each solution evaluation). The batch size remains the same for all three experiments and it is set to 32. For the base and TL experiments, we set the learning rate to $1 * 10^{-5}$ and optimizer to RMSprop, while the learning rate and optimizer for HPO-TL methods is set (optimized) by the method itself and therefore is not explicitly defined since it is not chosen deterministically.

Table 5. Training parameter settings for conducted experiments.

Parameter	**Value**		
	base	**TL**	**HPO-TL**
Nr. of epochs	50	50	2500
Batch size	32	32	32
Learning rate	$1 * 10^{-5}$	$1 * 10^{-5}$	-
Optimizer	RMSprop	RMSprop	-

5. Results

5.1. Classification Performance

In order to evaluate the COVID-19 X-ray image classification results, we first compared our three HPO-TL methods: TL-GWO, TL-GA, and TL-DE. For this purpose, we applied them upon the same CNN architecture using the same 10-fold cross-validation train-test folds, in order to objectively identify which of the three performed the best.

Results, obtained from the conducted experiments, are summarized in Table 6. As can be observed from the table, the difference among the three methods are quite small. Anyhow, the TL-DE method performed the best on average in most of the performance measures, while also achieving the lowest time for training. Interestingly, the results of the TL-DE method on all 10 folds were also the most stable, achieving the lowest standard deviation among the three methods regardless of the selected metric. In general, the second-best results were obtained by the TL-GWO method, while the results of the TL-GA lag a bit behind.

Table 6. Comparison of classification performance results on selected metrics over 10-fold cross-validation (averages and standard deviations are reported) for the three HPO-TL methods.

metric	**TL-GWO**	**TL-GA**	**TL-DE**
Accuracy	84.10 ± 3.2	82.45 ± 4.54	**84.44 ± 2.91**
AUC	83.61 ± 3.93	80.89 ± 6.26	**83.89 ± 3.36**
Precision	**88.52 ± 5.59**	85.09 ± 7.53	88.16 ± 3.84
Recall	85.82 ± 7.16	**87.75 ± 6.03**	86.38 ± 3.98
F**-1**	86.79 ± 2.93	86.01 ± 3.08	**87.16 ± 2.43**
Kappa	66.70 ± 6.92	62.27 ± 10.86	**67.40 ± 6.19**
Time	6096.40 ± 427.33	5383.10 ± 501.70	**5020.30 ± 380.97**

Fig. 3 shows a comparison of test accuracy results obtained by the three methods for all 10 folds on the Covid-19 X-ray image dataset. As we can see, in two folds the TLGA method performed a bit worse than the other two methods, while other differences are rather insignificant. If we look in detail, there is one situation where TL-GWO performed noticeably better than TL-DE (fold-6), while TL-DE performed noticeably better than TL-GWO in two situations (in fold-0 and fold-9). Very similar to accuracy were also the results of the rest of the metrics. Fig. 4 shows the box-plot comparison of the three methods with regard to AUC. It can be seen that the TL-DE achieved the best average AUC result, while also being the most stable among the methods.

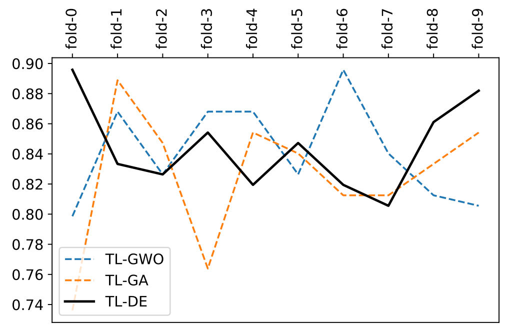

Fig. 3. Accuracy of the three HPO-TL methods on 10 folds.

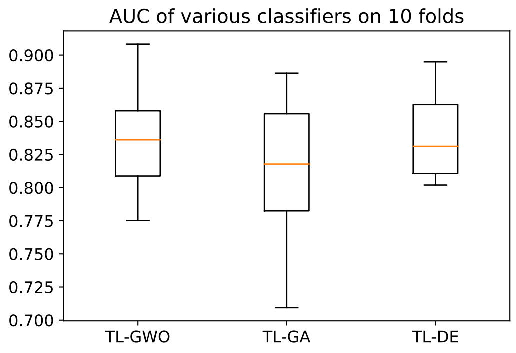

Fig. 4. Comparison of AUC for the three HPO-TL methods.

While the predictive performance results of the TL-DE and the TL-GWO methods were barely distinguishable, there was a bigger difference with regard to the time elapsed for training the CNN model (Fig. 5). As it can be seen, the TL-GWO method consumed the most time for training, while the TL-DE was the fastest of the three methods.

In general, with regard to the presented classification performance results, the TL-DE can be considered as the best method overall, although the differences between the three methods turned out to be very small.

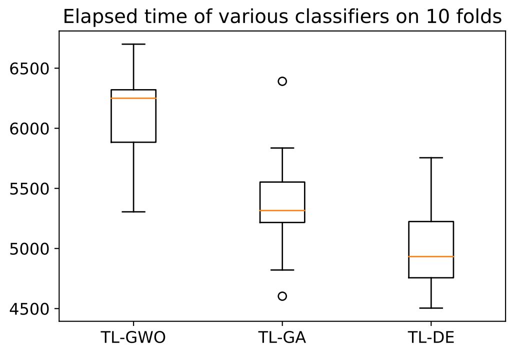

Fig. 5. Comparison of consumed time for training for the three HPO-TL methods.

5.2. Comparison with Other Methods

As the TL-DE turned out to be the best of the three methods, we wanted to compare it with the two most common existing approaches – base method for training the CNN model and TL method that performs transfer learning upon the same CNN architecture. For the base method, we utilized the VGG19 [40] CNN architecture, pre-trained on the ImageNet [11] dataset. For the TL method, we utilized the transfer learning approach and applied it on the same VGG19 CNN convolutional base. In this manner, the differences among the obtained predictive performance results can be contributed solely to the consequence of different learning method used. For the sake of comparison, we performed a series of experiments on the COVID-19 X-ray image dataset using the 10-fold cross-validation approach.

Results, obtained from the conducted experiments, are summarized in Table 7. As can be observed from the table, our proposed TL-DE method showed the best performance among the three compared methods regardless of the selected performance metric, with the exception of elapsed training time. In general, the second-best results were obtained by the TL method, while significantly the worst results were obtained by the base method.

Table 7. Comparison of classification performance results on selected metrics over 10-fold cross-validation (averages and standard deviations are reported) for the three compared methods.

metric	base	TL	TL-DE
Accuracy	54.51 ± 10.78	80.97 ± 4.37	$\mathbf{84.44 \pm 2.91}$
AUC	50.00 ± 0.00	80.85 ± 4.25	$\mathbf{83.89 \pm 3.36}$
Precision	42.85 ± 29.57	86.94 ± 4.18	$\mathbf{88.16 \pm 3.84}$
Recall	70.00 ± 48.30	81.38 ± 7.53	$\mathbf{86.38 \pm 3.98}$
F**-1**	53.16 ± 36.68	83.84 ± 4.20	$\mathbf{87.16 \pm 2.43}$
Kappa	0.00 ± 0.00	60.69 ± 8.56	$\mathbf{67.40 \pm 6.19}$
Time	377.50 ± 9.23	$\mathbf{340.40 \pm 6.45}$	5020.30 ± 380.97

Fig. 6 shows a comparison of test accuracy results obtained by the three compared methods for all 10 folds on the Covid-19 X-ray image dataset. As we can see, the TL-DE method achieved the highest accuracy in 7 out of 10 folds, followed by the TL method with 3 remaining wins, while the results of the base method lag quite distinctively behind. In all three folds, where the TL method outperformed the TL-DE, the differences were hardly noticeable, while the advantage of the TL-DE method were substantial in 6 out of 7 folds. Very similar results were obtained for all other predictive performance metrics. Fig. 7 shows the box-plot comparison of the three methods with regard to AUC, one of the most important metric when evaluating classification models in medicine, where the advantage of the TL-DE method can be easily observed. Not only that the mean AUC is the highest, the TL-DE produced results also with smaller standard deviation than the TL method.

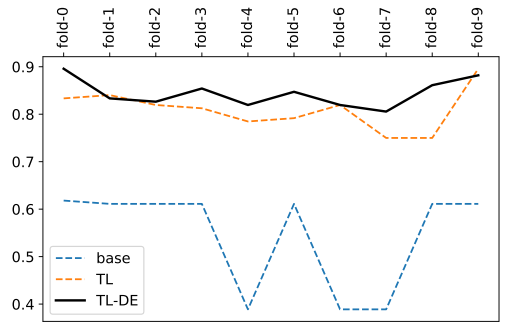

Fig. 6. Accuracy of the three compared methods on 10 folds.

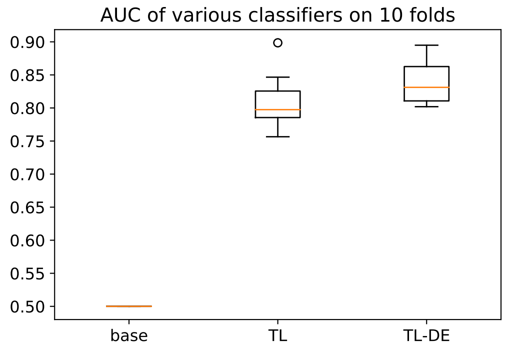

Fig. 7. Comparison of AUC for the three compared methods.

To achieve such excellent classification results, however, the TL-DE method pays its price with a much longer training time. While the base method spent on average 377.5

seconds to fully train the CNN mode, and the TL method 340.4 seconds, the HPO-based TL-DE method spent on average 5020.3 seconds, which is of course the consequence of the used optimization method.

5.3. Statistical Comparison

To evaluate the statistical significance of classification performance results of the three compared methods (base, TL, and TL-DE), we first applied the Friedman test by calculating the average Friedman ranks, Friedman asymptotic significance and p-values for all the three methods and for all 7 measures (acc, auc, prec, rec, F-1, kappa, and time), as suggested by Demšar [10]. The statistical results are summarized in Table 8. We can see that there is a significant difference among the three methods for all measures but the recall. The TL-DE is significantly better than the base method with regard to accuracy, AUC, precision, F-1, and kappa. It is also significantly better than the TL method with regard to accuracy, F-1, and kappa, while the difference is nearly significant with regard to AUC. On the other hand, the TL-DE method is significantly worse than the other two methods with regard to the required training time, as expected.

Table 8. Statistical comparison (p-values) of the Friedman test and Wilcoxon signed rank test for TL-DE vs. other two methods for all 7 metrics; significant differences are marked with *.

	Friedman test	**Wilcoxon signed rank test**	
metric	**all three**	**TL-DE vs. base**	**TL-DE vs. TL**
Accuracy	<0.001*	0.002*	0.033*
AUC	<0.001*	0.005*	0.084
Precision	<0.001*	0.002*	0.625
Recall	0.154	—	—
F**-1**	<0.001*	0.002*	0.014*
Kappa	<0.001*	0.002*	0.049*
Time	<0.001*	0.002*	0.002*

5.4. HPO-TL Methods Parameter Selection Analysis

Presented in Table 9 are the best performing selected values for optimized parameters for each fold. Inspecting the presented selected values, we can see that in the 4 folds, the number of selected units in the last fully-connected (dense) layer was set to 128, while also in 4 folds the number of selected units was set to 256, which is in line with the value which we handpicked. Those two values together were selected in 80% of all folds, while the remaining 2 selections were the lowest (64) and highest (1024) of possible values. The selected dropout probabilities are roughly ranging from 0.5 to 0.76, 7 of being in a range between 0.5 and 0.57 which is somewhat similar to what we manually selected for the remaining experiments (0.5). Focusing on the selected optimizer function, we can observe that the selection is almost evenly distributed between the RMSprop (4 out of 10 folds) and Adam optimizer (6 out of 10 folds), while the SGD is not a part of the best found solution in any fold. Regarding the selection of learning rates, 4 times each were selected learning rates $5*10^{-4}$ and $5*10^{-5}$. The latter is also the same as our handpicked value for the learning rate in the TL experiment.

Table 9. Best achieved solutions for the sought parameters per fold using TL-DE.

Fold	Neurons in last dense layer	Dropout probability	Optimizer	Learning rate
0	128	0.660450	adam	0.00050
1	64	0.500000	adam	0.00050
2	128	0.754095	adam	0.00005
3	256	0.500000	rmsprop	0.00005
4	256	0.512172	rmsprop	0.00050
5	128	0.537716	rmsprop	0.00001
6	256	0.571608	adam	0.00005
7	128	0.570084	adam	0.00050
8	1024	0.763849	rmsprop	0.00010
9	256	0.536784	adam	0.00005

We have also analyzed and compared parameter selections of remaining two HPO-TL methods, TL-GWO and TL-GA. Comparing the best parameter selections from best performing variation (TL-DE) against parameters selections of TL-GWO (Table 10), we can see that in general, the values for number of last hidden layer are lower, but on the other side, the dropout probability values of the best parameter selections are more similar to the best performing TL-DE variation. Also, the selection of optimizer is somewhat similar to the TL-DE variation, with a bit more tendency to selection of adam optimizer function.

Table 10. Best achieved solutions for the sought parameters per fold using TL-GWO.

Fold	Neurons in last dense layer	Dropout probability	Optimizer	Learning rate
0	128	0.660413	adam	0.00005
1	128	0.703526	adam	0.00010
2	64	0.732706	rmsprop	0.00005
3	512	0.513470	adam	0.00050
4	64	0.532255	adam	0.00010
5	64	0.642986	adam	0.00010
6	256	0.516314	rmsprop	0.00005
7	128	0.734866	rmsprop	0.00010
8	256	0.777165	adam	0.00010
9	64	0.555470	adam	0.00010

Table 11. Best achieved solutions for the sought parameters per fold using TL-GA.

Fold	Neurons in last dense layer	Dropout probability	Optimizer	Learning rate
0	1024	0.731663	rmsprop	0.00001
1	512	0.778245	adam	0.00050
2	128	0.557576	rmsprop	0.00050
3	256	0.872333	rmsprop	0.00050
4	256	0.567304	adam	0.00010
5	512	0.512127	adam	0.00010
6	256	0.532396	adam	0.00050
7	128	0.511880	adam	0.00010
8	256	0.821332	adam	0.00010
9	256	0.613837	adam	0.00050

If we compare the TL-DE further, with the worst performing of three variations TL-GA (Table 11), we can observe that selection of values for number of hidden units are quite similar to the best performing TL-DE variation. The selection of the optimizer function is proportionally the same as in TL-GWO. Interestingly, none of the three variations chose the SGD optimizer function as the best performing optimizer in any combination of parameter selections. The biggest difference between the parameter selection values can be seen in the range of dropout probabilities, which is in the case of TL-GA varying from 0.51 to 0.87.

5.5. Interpretable Representation of Model

When employing predictive models in various mission-critical decision-making systems, one of the biggest problem is determining trust in individual prediction of such models. Especially if such systems are being used in the fields like medicine, where predictions cannot be acted upon blind faith, consequences may be catastrophic [39].

In general, it is a common practice to evaluate predictive models using different metrics against the available test dataset. However, such common metrics may not be necessarily indicative of the model's goal. Therefore, inspecting individual instances and their representations which can be interpreted is a good complementary solution, especially when dealing with so called ”black-box” methods, to gain useful insights on how our model perceives it. Additionally, such evaluation can also help us increase the understanding and trust in our predictive model.

In Figure 8, we are showcasing LIME interpretable representations of our best performing predictive model, obtained by HPO-TL variation named TL-DE, which utilizes a DE algorithm for finding most suitable set of hyper-parameter values. In our previous research on COVID-10 identification [43], we have also used LIME method for evaluating the models' performance from a qualitative standpoint. The conducted analysis in the mentioned research was performed in such way, that all corresponding interpretable representations obtained from LIME were plotted on each corresponding sample (chest x-ray image), and each sample was then evaluated individually. In contrast to our previous research, here we are taking a different approach where we are obtaining the interpretable representations in the same manner for each sample as in our previous research, but in this case we are aggregating them into one. This allows us to get an insight into predictive model behavior over all test samples in one aggregated interpretable representation, instead of analyzing each sample individually.

Labeled as *a)* in Figure 8, we are showing green groups of pixels (super-pixel) denoting sections of image which have a positive impact towards classifying our input chest x-ray image as COVID-19, while labeled as *b)*, we are showing red super-pixels denoting sections of image which have a negative impact. The scale on the right side of each analyzed image is representing the intensity of each marked region, where a darker color is denoting a higher intensity and vice versa. Inspecting the image labeled *a)*, we can see similar patterns as we already identified in our previous research [43]. In the image showing green super-pixels, those are a bit more focused on the central thorax body region in contrast to the marked red super-pixels which are spread more across the whole upper body including shoulders and neck. Also, the intensity of the red super-pixels is a bit higher in those regions.

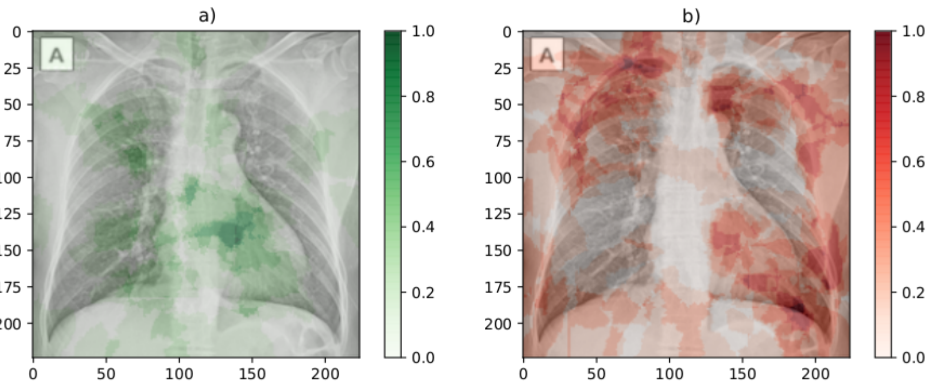

Fig. 8. Explaining predictive model decisions using LIME method. The image below the a) label represents the super-pixels which positively impact towards the COVID-19 class, while the image below the b) label is showcasing the super-pixels which negatively impact towards the COVID-19 class.

Comparing those two aggregated positive and negative interpretable representations could also be challenging, trying to compare specific regions of each sample. Therefore, we decided to aggregate those two into only one interpretable representation, which would possibly give us more clear insight into the regions which the predictive model is identifying as the ones having positive or negative impact. Such aggregated interpretable representation is presented on Figure 9 labeled as *a)*. As we can observe from the image, we can see that the most green super-pixels are still positioned in more central part of thorax body region, while the red super-pixels are also still positioned more on the outer parts of upper body. Interestingly, we can see the that green super-pixels also cover a region of the aortic arch and a part of the heart, which is similar to findings in our previous research.

Also, if we compare interpretable representations of our predictive model with similar researches [13, 53] and their interpretations, we can observe that our green super-pixels are in similar positions as in the most intense regions of the mentioned researches.

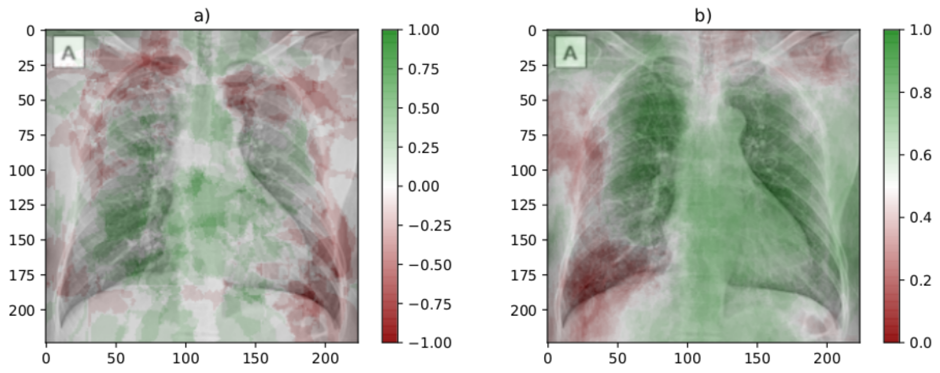

Fig. 9. Comparison of aggregated explanations between TL-DE and TL-GWO. The image below the a) label represents the TL-DE aggregated LIME explanations, while the image below the b) label is showcasing TL-GWO aggregated LIME explanations.

Interpretable representation labeled as *b)* presented on Figure 9 is also showing aggregated representation obtained from LIME but for the TL-GWO variation of the HPO-TL method. Since by the accuracy classification metric the TL-GWO lagged behind only by 0.34%, we were curious how does the representations from best performing model obtained by TL-GWO look like in comparison to the TL-DE. Comparing those two images, we can see that regardless of lagging behind for a small amount, the visual representation reveals that the difference is quite noticeable. In the case of TL-DE, the bounds of each super-pixel are more sharp and the border between them is more noticeable, while in the case of TL-GWO the borders of super-pixels are more blurred and the border between them is not so distinct. Overall, the TL-DE super-pixels are more exact in contrast to TL-GWO. Also, the green super-pixels of TL-GWO cover the majority of the upper body, which is not necessarily useful when trying to detect particular affected regions.

6. Conclusions

In this work, we proposed a generalized image classification method, based on GWOTLT [43], that trains a CNN using transfer learning with fine-tuning approach, in which hyper-parameter values are optimized with an optimization algorithm. Such generalized version, named HPO-TL, enables us to use different optimization algorithms which can be useful when dealing with various domain problems where different optimization algorithms can result in better final predictive model. The generalized method has been applied on a dataset of COVID-19 chest X-ray images using three different optimization algorithms DE, GWO, and GA. The obtained results showed that the best performing variation of

HPO-TL method is TL-DE which featured DE as an optimization algorithm. The best performing TL-DE also showed an impressive performance in all classification metrics when comparing to the conventional approaches of training a CNN.

We have also adopted a local interpretable model-agnostic explanations approach to provide insights of the COVID-19 disease, based on classification of chest X-rays. In contrast to straight-forward usage of such explanations, we have aggregated them into one, trying to get an insight on overall perception of a predictive model over all test samples instead of anayzing one by one. Thus, such approach was able to provide some interesting insights into the characteristics of COVID-19 disease and predictive model behaviour, by performing qualitative explanations upon the results of the trained model classification of a set of X-ray images.

In the future, we would like to expand our research to utilize different CNN architectures and conducted qualitative evaluations using additional methods such as SHAP [24].

Acknowledgments. The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

References

1. et al., M.A.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Apostolopoulos, I.D., Mpesiana, T.A.: Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in Medicine 43(2), 635–640 (jun 2020)
3. Apostolopoulos, I.D., Mpesiana, T.A.: Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in Medicine p. 1 (2020)
4. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems. pp. 2546–2554 (2011)
5. Brezočnik, L., Fister, I., Vrbančič, G.: Applying differential evolution with threshold mechanism for feature selection on a phishing websites classification. In: Welzer, T., Eder, J., Podgorelec, V., Wrembel, R., Ivanović, M., Gamper, J., Morzy, M., Tzouramanis, T., Darmont, J., Kamišalić Latifić, A. (eds.) New Trends in Databases and Information Systems. pp. 11–18. Springer International Publishing, Cham (2019)
6. Ching, J.Y., Wong, A.K.C., Chan, K.C.C.: Class-dependent discretization for inductive learning from continuous and mixed-mode data. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(7), 641–651 (1995)
7. Chollet, F., et al.: Keras (2015), https://keras.io
8. Cohen, J.P., Morrison, P., Dao, L.: Covid-19 image data collection. arXiv 2003.11597 (2020), https://github.com/ieee8023/covid-chestxray-dataset
9. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution–an updated survey. Swarm and Evolutionary Computation 27, 1–30 (2016)
10. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7 (2006)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)

12. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: 7th International Conference on Information and Knowledge Management. pp. 148–152 (January 1998), https://www.microsoft.com/en-us/research/publication/inductive-learning-algorithms-and-representations-for-text-categorization/
13. Duran-Lopez, L., Dominguez-Morales, J.P., Corral-Jaime, J., Vicente-Diaz, S., Linares-Barranco, A.: Covid-xnet: a custom deep learning system to diagnose and locate covid-19 in chest x-ray images. Applied Sciences 10(16), 5683 (2020)
14. Fang, Y., Zhang, H., Xie, J., Lin, M., Ying, L., Pang, P., Ji, W.: Sensitivity of chest ct for covid-19: comparison to rt-pcr. Radiology 296(2), E115–E117 (2020)
15. Faris, H., Aljarah, I., Al-Betar, M.A., Mirjalili, S.: Grey wolf optimizer: a review of recent variants and applications. Neural computing and applications 30(2), 413–435 (2018)
16. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, BioL Cybem. 36 (1980) 193-202. S. Shiotani et al./Neurocomputing 9 (1995) Ill-130 130 (1980)
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
18. Govindarajan, S., Swaminathan, R.: Differentiation of COVID-19 conditions in planar chest radiographs using optimized convolutional neural networks. Applied Intelligence (2020)
19. Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). pp. 21–26. ICST (Institute for Computer Sciences, Social-Informatics and ... (2016)
20. Kamilaris, A., Prenafeta-Boldú, F.X.: Deep learning in agriculture: A survey. Computers and electronics in agriculture 147, 70–90 (2018)
21. Kikkisetti, S., Zhu, J., Shen, B., Li, H., Duong, T.Q.: Deep-learning convolutional neural networks with transfer learning accurately classify COVID-19 lung infection on portable chest radiographs. PeerJ 8 (nov 2020)
22. Kong, Q., Trugman, D.T., Ross, Z.E., Bianco, M.J., Meade, B.J., Gerstoft, P.: Machine learning in seismology: Turning data into insights. Seismological Research Letters 90(1), 3–14 (2018)
23. Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., Pastor, J.R.: Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proceedings of the genetic and evolutionary computation conference. pp. 481–488 (2017)
24. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions. arXiv preprint arXiv:1705.07874 (2017)
25. Lundervold, A.S., Lundervold, A.: An overview of deep learning in medical imaging focusing on mri. Zeitschrift für Medizinische Physik 29(2), 102–127 (2019)
26. Majeed, T., Rashid, R., Ali, D., Asaad, A.: Covid-19 detection using cnn transfer learning from x-ray images. medRxiv (2020)
27. Majeed, T., Rashid, R., Ali, D., Asaad, A.: Issues associated with deploying CNN transfer learning to detect COVID-19 from chest X-rays. Physical and Engineering Sciences in Medicine 43(4), 1289–1303 (dec 2020)
28. Marques, G., Agarwal, D., de la Torre Díez, I.: Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. Applied Soft Computing Journal 96 (nov 2020)
29. McKinney, W.: Data structures for statistical computing in python. In: van der Walt, S., Millman, J. (eds.) Proceedings of the 9th Python in Science Conference. pp. 51 – 56 (2010)
30. Mirjalili, S.: Genetic algorithm. In: Evolutionary algorithms and neural networks, pp. 43–55. Springer (2019)
31. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in engineering software 69, 46–61 (2014)

32. Nayak, S.R., Nayak, D.R., Sinha, U., Arora, V., Pachori, R.B.: Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study. Biomedical Signal Processing and Control 64, 102365 (feb 2021), https://doi.org/10.1016/j.bspc.2020.102365
33. of North America, R.S.: RSNA Pneumonia Detection Challenge — Kaggle, https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview
34. Organization, W.H., et al.: Covid-19 weekly epidemiological update - 2 february 2021. In: COVID-19 Weekly Epidemiological update - 2 February 2021. World Health Organization (2021)
35. Ozturk, T., Talo, M., Yildirim, E.A., Baloglu, U.B., Yildirim, O., Acharya, U.R.: Automated detection of covid-19 cases using deep neural networks with x-ray images. Computers in biology and medicine 121, 103792 (2020)
36. Pathak, Y., Shukla, P.K., Tiwari, A., Stalin, S., Singh, S.: Deep transfer learning based classification model for covid-19 disease. Irbm (2020)
37. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
38. Podgorelec, V., Pečnik, Š., Vrbančič, G.: Classification of similar sports images using convolutional neural network with hyper-parameter optimization. Applied Sciences 10(23), 8494 (2020)
39. Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
41. Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J.: Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE transactions on medical imaging 35(5), 1299–1312 (2016)
42. Van Der Walt, S., Colbert, S.C., Varoquaux, G.: The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering 13(2), 22 (2011)
43. Vrbančič, G., ˇŠ. Pečnik, Podgorelec, V.: Identification of covid-19 x-ray images using cnn with optimized tuning of transfer learning. In: 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA). pp. 1–8 (2020)
44. Vrbančič, G., Brezočnik, L., Mlakar, U., Fister, D., Fister Jr., I.: NiaPy: Python microframework for building nature-inspired algorithms. Journal of Open Source Software 3 (2018), https://doi.org/10.21105/joss.00613
45. Vrbancic, G., Fister, I.J., Podgorelec, V.: Automatic Detection of Heartbeats in Heart Sound Signals Using Deep Convolutional Neural Networks. Elektronika ir Elektrotechnika 25(3), 71–76 (jun 2019), http://eejournal.ktu.lt/index.php/elt/article/view/23680
46. Vrbancic, G., Fister, I.J., Podgorelec, V.: Parameter Setting for Deep Neural Networks Using Swarm Intelligence on Phishing Websites Classification. International Journal on Artificial Intelligence Tools 28(6), 28 (oct 2019)
47. Vrbancic, G., Fister, I.J., Podgorelec, V.: Parameter Setting for Deep Neural Networks Using Swarm Intelligence on Phishing Websites Classification. International Journal on Artificial Intelligence Tools 28(6), 28 (oct 2019)
48. Vrbancic, G., Podgorelec, V.: Automatic Classification of Motor Impairment Neural Disorders from EEG Signals Using Deep Convolutional Neural Networks. Elektronika ir Elektrotechnika 24(4), 3–7 (aug 2018), http://eejournal.ktu.lt/index.php/elt/article/view/21469
49. Vrbančič, G., Zorman, M., Podgorelec, V.: Transfer learning tuning utilizing grey wolf optimizer for identification of brain hemorrhage from head ct images. In: StuCoSReC: proceedings of the 2019 6th Student Computer Science Research Conference. pp. 61–66 (2019)

50. Vrbančič, G., Podgorelec, V.: Transfer learning with adaptive fine-tuning. IEEE Access 8, 196197–196211 (2020)
51. Yang, Q., Ling, C., Chai, X., Pan, R.: Test-cost sensitive classification on data with missing values. IEEE Transactions on Knowledge & Data Engineering 18(5), 626–638 (2006)
52. Yu, T., Zhu, H.: Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689 (2020)
53. Zebin, T., Rezvy, S.: Covid-19 detection and disease progression visualization: Deep learning on chest x-rays for classification and coarse localization. Applied Intelligence pp. 1–12 (2020)
54. Zhang, Y., Song, K., Sun, Y., Tan, S., Udell, M.: "why should you trust my explanation?" understanding uncertainty in lime explanations. arXiv preprint arXiv:1904.12991 (aug 2014)
55. Zhu, X., Wu, X.: Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering. IEEE Transactions on Knowledge & Data Engineering 18(10), 1435–1440 (2006)

Grega Vrbančič received the B.Sc. and M.Sc. degrees in informatics and communication technologies from the University of Maribor in 2015 and 2017, respectively. In 2021, he received a Ph.D. degree from the University of Maribor. Currently, he is a teaching assistant and a researcher with the Faculty of Electrical Engineering and Computer Science, University of Maribor. He is the author of six peer-reviewed scientific journal articles and several conference papers. He has been involved in several industrial research and development projects. His research interests include deep learning, especially convolutional neural networks, focusing on training strategies and transfer learning.

Špela Pečnik received her bachelor's degree in Information Technology in 2017, and a master's degree in Information Technology in 2019 from the University of Maribor, Faculty of Electrical Engineering and Computer Science. Since 2019, she has been employed as a teaching assistant and researcher at the Faculty of Electrical Engineering and Computer Science, University of Maribor. Her research areas include artificial intelligence, machine learning, data mining, and information systems.

Vili Podgorelec is a professor of computer science at the University of Maribor, Slovenia, where he received the Ph.D. degree in 2001. He has been involved in AI and ML for 20 years, where he gained professional experience in implementation of many scientific and industrial R&D projects related to analysis, design, implementation, integration, and evaluation of intelligent information systems using AI and ML. He has authored more than 50 peer-reviewed scientific journal papers, more than 100 conference papers, three books and several book chapters on machine learning, computational intelligence, data science, medical informatics, and software engineering. Dr. Podgorelec has worked as a visiting professor and/or researcher at several universities around the world, including University of Osaka, Japan; Federal University of Sao Paulo, Brazil; University of Nantes, France; University of La Laguna, Spain; University of Madeira, Portugal; University of Applied Sciences Seinäjoki, Finland; University of Applied Sciences Valencia, Spain. He received several international awards and grants for his research activities.

Received: February 09, 2021; Accepted: October 08, 2021.