Identification of COVID-19 X-ray Images using CNN with Optimized Tuning of Transfer Learning

Grega Vrbančič
University of Maribor, FERI
Maribor, Slovenia
grega.vrbancic@um.si

Špela Pečnik
University of Maribor, FERI
Maribor, Slovenia
spela.pecnik@um.si

Vili Podgorelec
University of Maribor, FERI
Maribor, Slovenia
vili.podgorelec@um.si

***Abstract*—At this early stage in the COVID-19 epidemic, 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. We first developed a new adapted classification method that is able to identify COVID-19 patients based on a chest X-ray, and then adopted a local interpretable model-agnostic explanations approach to provide the insights. The classification method uses a grey wolf optimizer algorithm for the purpose of optimizing hyper-parameter values within the transfer learning tuning of a CNN. The trained model is then used to classify a set of X-ray images, upon which the qualitative explanations are performed. The presented approach was tested on a dataset of 842 X-ray images, with the overall accuracy of 94.76%, outperforming both conventional CNN method as well as the compared baseline transfer learning method. The achieved high classification accuracy enabled us to perform a qualitative in-depth analysis, which revealed that there are some regions of greater importance when identifying COVID-19 cases, like aortic arch or carina and right main bronchus. The proposed classification method proved to be very competitive, enabling one to perform an in-depth analysis, necessary to gain qualitative insights into the characteristics of COVID-19 disease.**

***Index Terms*—COVID-19, classification, CNN, transfer learning, optimization**

I. Introduction

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 15th of April 2020, there were 1.9 million confirmed cases and 123,010 deaths globally, spreading across 210 countries and territories [1].

The COVID-19 is the seventh known coronavirus to infect humans. The two also known examples of coronavirus include severe acute respiratory syndrome (SARS) and Middle East respiratory syndrome (MERS). The first began in southern China and resulted in 774 deaths out of 8,098 infected individuals in 29 countries from November 2002 through July 2003, while the second originated in Saudi Arabia and was responsible for 848 deaths among 2,458 individuals in 27 countries through July 2019 [2].

PREPRINT

Since mid February 2020, various researchers [3] started to collect and publish anonymized X-ray chest images of patients diagnosed with COVID-19, which let the researchers to study the collected data and possibly identify useful patterns, which could give us useful insights and enable us to design and develop computer-aided diagnosis systems which could facilitate work for radiologists.

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 [4]. Especially in the last months there is an increase of research focused on applying the machine learning algorithms to identification of COVID-19. One of most common approaches to tackle the mentioned issue is to utilize the transfer learning approach as presented in [5], [6]. Based on our previous experience with the detection of brain hemorrage from head CT images [7] as well as promising results from similar studies, we set our goals to adapting the method to identify a COVID-19 chest X-ray images from a relatively small dataset. Beside providing and evaluating a predictive model, we also conducted an analysis of interpretable representations of our model in order to gain useful insights on how the model perceives the chest X-ray images, evaluating the model's decisions from a qualitative perspective.

II. Methods

Since the first introduction of CNNs in 1980s [8], 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 [9].

Those capabilities make the CNNs de facto standard for solving the image recognition tasks in various domains from medicine [10], [11], information security [12] to seismology [13] or even agriculture [14]. 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 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.

A. Transfer Learning

The first appearances of transfer learning in publications are dating back to the 1995 [15], mostly under different names such as inductive transfer [16], incremental or cumulative learning [17], and multitask learning [18], 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 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 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 [19]. Commonly, we only transfer the weights in the so called convolutional base of CNN architecture, which is composed from 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) [19].

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.

B. Grey Wolf Optimizer for Transfer Learning Tuning

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 Parze Estimator approach or Random search approach [20]. But in recent years, population-based metaheuristic algorithms are becoming more and more popular in successfully solving HPO problems [7], [21], [22].

Based on our previous success with utilization of Grey Wolf Optimizer (GWO) algorithm for the purpose of optimizing hyper-parameter values, we decided to adapt our Grey Wolf Optimizer for Transfer Learning Tuning (GWOTLT) method presented in [7]. The GWOTLT method is based on the GWO algorithm [23], 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 wolfs (Canis lupus). As defined by authors in [23], there are three main phases of grey wolfs 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.

The basic concept of our GWOTLT method can be generally defined in the following steps:

1) GWO 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 GWO.

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

The GWOTLT 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 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 last fully-connected (dense) layer. Formally, the individuals of such GWOTLT produced solutions are presented as a real-valued vectors:

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

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 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 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 GWOTLT solution, and AUC defines the area under the ROC curve metric.

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

C. Local Interpretable Model-agnostic Explanations

The success of adopting machine learning models, especially in more sensitive domains such as medicine, depends on its interpretation or on how well decision makers are able to understand and trust model predictions. The level of confidence in the results is increased when competent decision makers have a clear insight into what influenced the decision most, what are the possible errors and what is the behavior of the model. For this purpose, different interpretative methods that interpret models built over unstructured data have been developed [24].

In our case, we used the Local Interpretable Model-Agnostic Explanations (LIME) method, which was introduced in 2016 by Ribeiro et al. [24]. Beside the SHapley Additive exPlanations (SHAP) method, it is one of the most used, because it also explains models that are built over images. The LIME method explains the prediction of an individual input by sampling its neighboring inputs and builds a sparse linear model based on the predictions of these inputs. Features with larger coefficients calculated in this linear model are then considered more relevant to forecast the given input. Generating a local explanation for an input requires sampling around the input so that an explanation can be generated to its prediction [25]. The algorithm behind LIME can explain the predictions of any black box classifier with two or more output classes, and its goal is to identify an interpretive model over an interpretative representation that is locally faithful to the classifier. In the case of image classification, 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. LIME requires an implementation of a function from a classifier that accepts a set of classes and then returns the probabilities for each class.

The explanation given by LIME 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:

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

So, x denotes 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. Not every model is simple enough to be interpretive, so with $\Omega(g)$ we mark the complexity of model interpretation, that is opposite to its interpretability. 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. In order to achieve high interpretability, this value must be kept to a minimum [24].

III. Experiments

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

- **base**, where the CNN is trained in a conventional manner without pre-training,
- **TL**, where transfer learning methodology is utilized, and
- **GWOTLT**, where our proposed method is used.

All conducted experiments were implemented in Python programming language with the support of following libraries: scikit-learn [26], Pandas [27], Numpy [28], NiaPy [29], Keras [30] and Tensorflow [31].

Experiments were performed using the Intel Core i7-6700K quad-core CPU running at 4 GHz, 64GB of RAM, and three Nvidia GeForce Titan X Pascal GPUs each with dedicated 12GB of GDDR5 memory.

A. Dataset

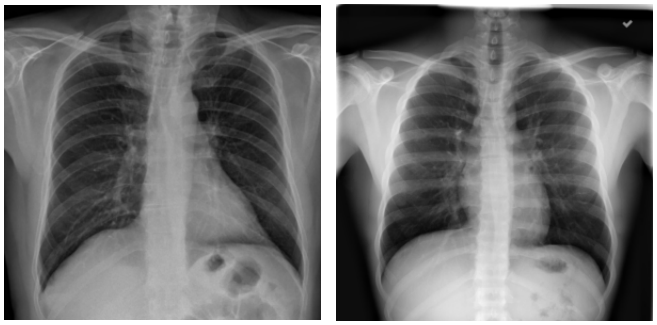
For the task of identifying COVID-19 from the chest X-ray images, we used a publicly available dataset which was recently prepared by Cohen et. al [3]. The dataset is composed

TABLE I
TARGET CLASS DISTRIBUTION OF COVID-19 IMAGE DATA COLLECTION.

Class	COVID-19 image data collection
COVID-19	182
ARDS	4
SARS	11
Pneumocystis	15
Streptococcus	17
No finding	2
Chlamydophila	2
E.Coli	4
COVID-19, ARDS	2
Klebsiella	1
Legionella	2
Total	242

TABLE II
TARGET CLASS DISTRIBUTION OF AN EXTENDED COVID-19 IMAGE DATA COLLECTION.

Class	Extended COVID-19 image data collection
COVID-19	182
Other	660
Total	842



a) b)

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.

TABLE III
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

from the various public sources and already published papers. In the Table I is presented the total number of collected images as well as the distribution between the target classes. As the dataset is being continuously improved in terms of total number of collected images, we are reporting those statistics as of April 4th 2020.

Observing the distribution between the target classes, we can see that the majority of images belongs to "COVID-19" target class, while the rest of the target classes are quite under-represented. Since our goal is to perform the task of identification of COVID-19 from the chest X-ray images, we decided to transform the original COVID-19 image data collection from multi-classification dataset to binary classification dataset putting all of the "COVID-19" labeled images into one group and the remaining images into the other group. This way, we obtained the dataset with two different classes, namely "COVID-19" and "other", each with total of 182 and 60 images respectively. Since the target classes are still quite unbalanced, we extended the "other" labeled group of images with the X-ray images collected from the RSNA Pneumonia Detection Challenge [32]. The RSNA Pneumonia Detection Challenge addresses the task of locating lung opacities on chest radiographs. In total the dataset is composed of 26,684 chest images annotated with three different classes: No Lung Opacity, Lung Opacity and Normal. In order to mimic the approximate distribution in our extended COVID-19 dataset, we randomly selected 600 "Normal" labeled chest images from RSNA Pneumonia Detection Challenge dataset, which resulted in a extended dataset with properties presented in Table II.

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.

B. Data Pre-processing

As are the images in the extended 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 III. 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.

C. CNN Setup

For the deep CNN architecture, we adapted a well known VGG19 architecture presented by Simonyan et. al in 2014

[33]. 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 experiment utilizing the GWOTLT method 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 GWOTLT experiment is also being optimized by the method itself.

For the TL and GWOTLT experiment, 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).

D. GWOTLT Settings

Since the utilized GWOTLT method works in an iterative manner, where 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 GWOTLT produced solutions and evaluating them – calculating the fitness value against the remaining smaller subset of the initial training set.

For each fold, the GWOTLT 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 GWOTLT 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.

E. Training Parameter Settings

Presented in Table IV 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 GWOTLT which in worst case scenario, consumes a total of 2500 epochs (50 epoch 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 GWOTLT is set (optimized) by the method itself and therefore is not explicitly defined since it is not chosen deterministically.

TABLE IV
TRAINING PARAMETER SETTINGS FOR CONDUCTED EXPERIMENTS.

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

TABLE V
COMPARISON OF AVERAGE ACCURACIES, AUCS, PRECISIONS, RECALLS, F-1 SCORES, KAPPA COEFFICIENTS, AND TIME (IN SECONDS) WITH STANDARD DEVIATIONS OVER 10-FOLD CROSS-VALIDATION.

metric	base	TL	GWOTLT
Accuracy	76,73 $\pm$ 4.88	91.56 $\pm$ 2.44	**94.76 $\pm$ 3.19**
AUC	50.00 $\pm$ 0.00	85.64 $\pm$ 4.36	**93.47 $\pm$ 4.22**
Precision	76.73 $\pm$ 0.41	92.71 $\pm$ 2.25	**97.42 $\pm$ 2.70**
Recall	**100.00 $\pm$ 0.00**	96.76 $\pm$ 2.48	95.83 $\pm$ 4.66
F-1	86.83 $\pm$ 0.26	94.62 $\pm$ 1.55	**96.52 $\pm$ 2.24**
Kappa	0.00 $\pm$ 0.00	0.75 $\pm$ 0.08	**0.86 $\pm$ 0.08**
Time	425.40 $\pm$ 4.88	**131.00 $\pm$ 2.11**	2207.30 $\pm$ 626.47

IV. RESULTS

A. Classification Performance

In order to evaluate the COVID-19 X-ray image classification results objectively, we compared our GWOTLT method with two baseline methods – 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 [33] CNN architecture, pre-trained on the ImageNet [34] 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 V. As can be observed from the table, our proposed GWOTLT method showed the best performance among the three compared methods regardless of the selected compared metric, with the exception of training time. In general, the second best results were obtained by the TL method, while the worst results were obtained by the base method.

Figure 3 shows a comparison of test AUC results, one of the most important metric when evaluating classification models in medicine. As we can see, the GWOTLT method achieved the highest accuracy in all 10 folds, followed by the TL method with second best results, while the results of the base method lag quite far behind. The domination of the GWOTLT method with regard to AUC can be easily observed also on a violin plot (Figure 4). Not only that the mean AUC is the highest, the GWOTLT produced results also with a bit smaller standard deviation than the TL method. Similar results

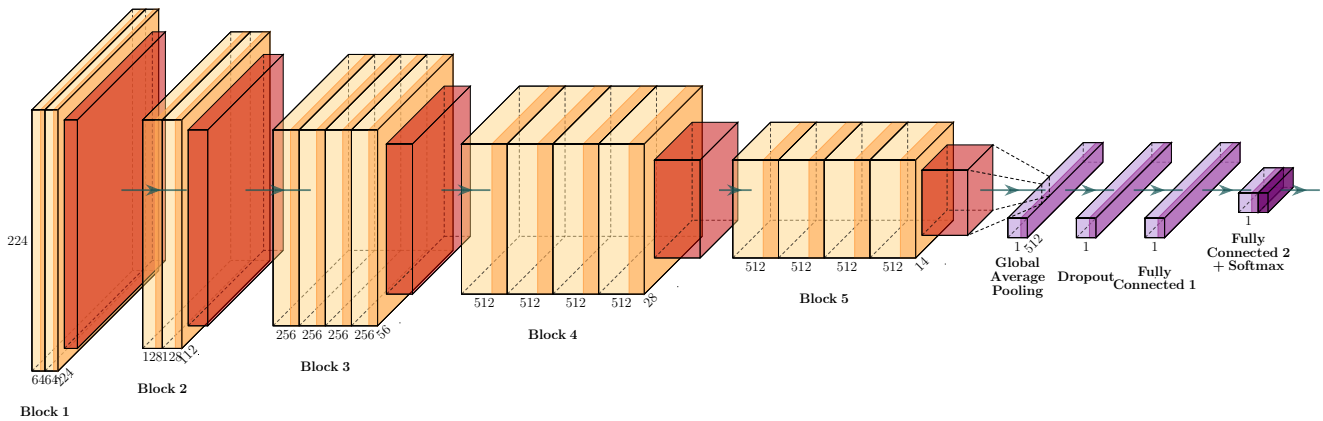


Fig. 2. The adapted VGG19 convolutional neural network architecture.

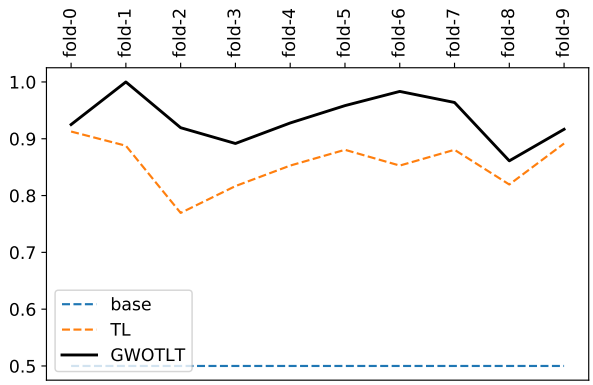


Fig. 3. AUC results of compared methods on 10 folds.

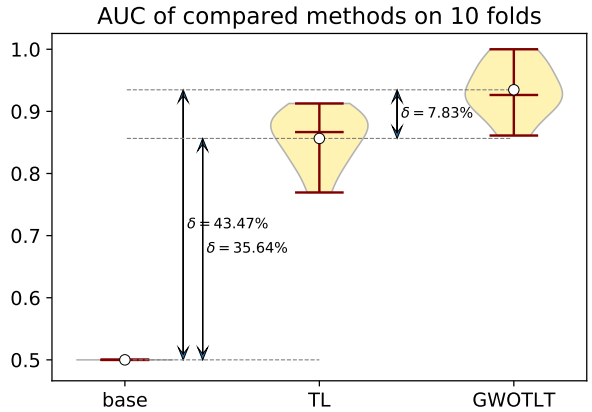


Fig. 4. Comparison of AUC for the three methods.

have been obtained also for other metrics, but are not shown here in order not to unnecessarily extend the article.

To evaluate the statistical significance of classification performance results, 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 [35]. The statistical results are summarized in Table VI. We can see that there is a significant difference among the three methods for all seven measures. The GWOTLT is significantly better than the compared two methods with regard to acc, AUC, precision, F-1 and kappa, while it is significantly worse than the other two methods with regard to the required training time.

TABLE VI
STATISTICAL COMPARISON (p-VALUES) OF THE FRIEDMAN TEST AMONG THE THREE METHODS AND WILCOXON SIGNED RANK TEST FOR GWOTLT VS. OTHER TWO METHODS FOR ALL 7 METRICS.

	Friedman test	**Wilcoxon signed rank test**	
metric	**all three**	**GWOTLT vs. base**	**GWOTLT vs. TL**
Accuracy	<0.001*	0.005*	0.005*
AUC	<0.001*	0.005*	0.005*
Precision	<0.001*	0.005*	0.005*
Recall	0.002*	0.018*	0.767
F-1	<0.001*	0.005*	0.075
Kappa	<0.001*	0.005*	0.028*
Time	<0.001*	0.005*	0.005*

B. GWOTLT Parameter Selection Analysis

Presented in Table VII are the best performing selected values for optimized parameters for each fold. Interestingly, we can see that in the majority of the folds, the number of selected units in last fully-connected (dense) layer was set to 256, which is in line with the value which we handpicked for the base and TL experiments. The selected dropout probabilities are roughly ranging from 0.6 to 0.8 which is a bit higher than what we manually selected for the remaining experiments. Focusing on the selected optimizer function, we can observe that the selection is evenly distributed between the RMSprop and Adam optimizer, while the SGD is not a part of the best found solution in any fold. Regarding the selection learning rates, in the majority of folds (7 occurrences) the learning rate is set to $5 * 10^{-4}$ which is a bit higher than what we manually set for other two experiments.

TABLE VII
BEST ACHIEVED SOLUTIONS FOR THE SOUGHT PARAMETERS PER FOLD.

Fold	Neurons in last dense layer	Dropout probability	Optimizer	Learning rate
0	64	0.637667	rmsprop	0.00050
1	256	0.598971	adam	0.00050
2	256	0.698623	adam	0.00050
3	128	0.681882	adam	0.00050
4	1024	0.709137	rmsprop	0.00005
5	256	0.684796	rmsprop	0.00005
6	256	0.637454	adam	0.00050
7	256	0.729944	rmsprop	0.00100
8	128	0.728626	rmsprop	0.00050
9	256	0.794064	adam	0.00050

C. Interpretable Representation of Model

One of the most important problems when utilizing any kind of predictive models for the task of decision making is determining trust in individual predictions, especially when such models are being used as a mission critical components in the decision systems or are being used in the fields like medicine, where predictions cannot be acted upon blind faith, as the consequences may be catastrophic [24].

Commonly, the models are evaluated only using various metrics on an available test dataset, but the metrics may not be necessarily indicative of the models goal. Therefore, inspecting individual instances and their interpretable representations is a good complementary solution to gain useful insights on how our model perceives it and also help us increase the understanding and trust in our predictive model.

In Figure 5 some test instances of used dataset with their corresponding interpretable representations of our best performing predictive model are presented. The interpretable representations are being generated using LIME method, where green groups of pixels (super-pixel) are denoting identified sections of image which have a positive impact towards particular target class, while the red super-pixels are denoting the sections of image which have a negative impact towards the particular target class. In other words, the green super-pixels in images above the label a) have positive impact for classifying COVID-19 images, while the green super-pixels in images above the label b) have positive impact for classifying as other on no pathology identified.

Inspecting the group of samples diagnosed with COVID-19, we can observe that the marked super-pixels in general are a bit more focused on the thorax body region in contrast to the marked super-pixels in the samples without any diagnosed pathology, where neck, shoulders and hands are being marked with large patches. Additionally, if we compare the images pair-wise COVID-19 diagnosed versus the normal images, we can observe that locations of positive (green) super-pixels in COVID-19 images are roughly covering the same positions as the negative (red) super-pixels in other classified images. Focusing on each image pair from top to bottom, looking at the first pair, in the chest region of COVID-19 (left) image, we can see the that green super-pixels cover region of aortic arch and a part of heart while the heart region in the image classified as other (right) covers roughly the same position with negative impact on predicted class. Similar can be observed also in the second image pair where big green super-pixel on COVID-19 image is positioned in the second zone of chest covering carina and right main bronchus, where on the right image a similar position is covered with red super-pixel. The third pair is interesting from a perspective that the inverse of the green and red patches when comparing the COVID-19 classified and other classified images is located at the top of the third chest zone on the right side of the lung. Also in the image classified as other, it is interesting that the red super-pixel is surrounded by green super-pixels, which is not the case in other image pairs. Additionally, in the COVID-19 image a region of green super-pixel covering the aortic arch and carina is clearly visible which was also the case in previous pair but the right image classified as other lacks the negative super-pixel in this case.

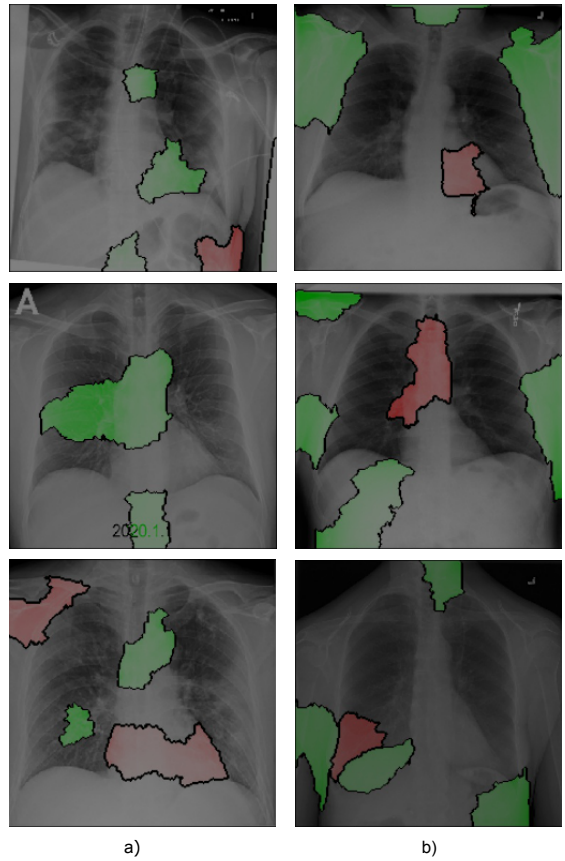


Fig. 5. Examples of explained predictive model decisions using LIME method. The images above the a) represents the images with diagnosed COVID-19, while the images above the b) represents the images with other or no pathology identified.

V. CONCLUSIONS

In this work, we proposed a new adapted image classification method GWOTLT that trains a CNN using transfer learning with fine-tuning, in which hyper-parameter values are optimized with GWO approach. The proposed method has

been applied on a dataset of COVID-19 chest X-ray images. The obtained results showed an impressive performance of the method, achieving on average 94.76% accuracy and AUC of 93.47%, with a very small standard deviation.

Encouraged by the model's predictive performance, we adopted a local interpretable model-agnostic explanations approach to provide insights of the COVID-19 disease, based on classification of chest X-rays. Thus, the LIME explanation approach was able to provide some interesting insights into the characteristics of COVID-19 disease, 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 compare it against VGG19 as well as use extended COVID-19 datasets.

ACKNOWLEDGEMENT

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

REFERENCES

[1] W. H. Organization *et al.*, "Coronavirus disease 2019 (covid-19): situation report, 86," 2020.

[2] A. Bernheim, X. Mei, M. Huang, Y. Yang, Z. A. Fayad, N. Zhang, K. Diao, B. Lin, X. Zhu, K. Li *et al.*, "Chest ct findings in coronavirus disease-19 (covid-19): relationship to duration of infection," *Radiology*, p. 200463, 2020.

[3] J. P. Cohen, P. Morrison, and L. Dao, "Covid-19 image data collection," *arXiv 2003.11597*, 2020. [Online]. Available: https://github.com/ieee8023/covid-chestxray-dataset

[4] A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on mri," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019.

[5] I. D. Apostolopoulos and T. A. Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, p. 1, 2020.

[6] T. Majeed, R. Rashid, D. Ali, and A. Asaad, "Covid-19 detection using cnn transfer learning from x-ray images," *medRxiv*, 2020.

[7] G. Vrbančič, M. Zorman, and V. Podgorelec, "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*, 2019, pp. 61–66.

[8] K. Fukushima, "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*, vol. 130, 1980.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[10] G. Vrbancic and V. Podgorelec, "Automatic Classification of Motor Impairment Neural Disorders from EEG Signals Using Deep Convolutional Neural Networks," *Elektronika ir Elektrotechnika*, vol. 24, no. 4, pp. 3–7, aug 2018. [Online]. Available: http://eejournal.ktu.lt/index.php/elt/article/view/21469

[11] G. Vrbancic, I. J. Fister, and V. Podgorelec, "Automatic Detection of Heartbeats in Heart Sound Signals Using Deep Convolutional Neural Networks," *Elektronika ir Elektrotechnika*, vol. 25, no. 3, pp. 71–76, jun 2019. [Online]. Available: http://eejournal.ktu.lt/index.php/elt/article/view/23680

[12] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "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)*. ICST (Institute for Computer Sciences, Social-Informatics and . . . , 2016, pp. 21–26.

[13] Q. Kong, D. T. Trugman, Z. E. Ross, M. J. Bianco, B. J. Meade, and P. Gerstoft, "Machine learning in seismology: Turning data into insights," *Seismological Research Letters*, vol. 90, no. 1, pp. 3–14, 2018.

[14] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[15] J. Y. Ching, A. K. C. Wong, and K. C. C. Chan, "Class-dependent discretization for inductive learning from continuous and mixed-mode data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 641–651, 1995.

[16] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *7th International Conference on Information and Knowledge Management*, January 1998, pp. 148–152. [Online]. Available: https://www.microsoft.com/en-us/research/publication/inductive-learning-algorithms-and-representations-for-text-categorization/

[17] X. Zhu and X. Wu, "Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering," *IEEE Transactions on Knowledge & Data Engineering*, vol. 18, no. 10, pp. 1435–1440, 2006.

[18] Q. Yang, C. Ling, X. Chai, and R. Pan, "Test-cost sensitive classification on data with missing values," *IEEE Transactions on Knowledge & Data Engineering*, vol. 18, no. 5, pp. 626–638, 2006.

[19] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[20] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[21] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 481–488.

[22] G. Vrbancic, I. J. Fister, and V. Podgorelec, "Parameter Setting for Deep Neural Networks Using Swarm Intelligence on Phishing Websites Classification," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 6, p. 28, oct 2019.

[23] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[24] M. T. Ribeiro, S. Singh, and C. Guestrin, "" 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*, 2016, pp. 1135–1144.

[25] Y. Zhang, K. Song, Y. Sun, S. Tan, and M. Udell, ""why should you trust my explanation?" understanding uncertainty in lime explanations," *arXiv preprint arXiv:1904.12991*, aug 2014.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

[28] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.

[29] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," *Journal of Open Source Software*, vol. 3, 2018. [Online]. Available: https://doi.org/10.21105/joss.00613

[30] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[31] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[32] "RSNA Pneumonia Detection Challenge — Kaggle." [Online]. Available: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[35] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research 7*, 2006.