Transfer Learning With Adaptive Fine-Tuning

Article *in* IEEE Access · October 2020

DOI: 10.1109/ACCESS.2020.3034343

2 authors:

Grega Vrbančič
University of Maribor
29 PUBLICATIONS **492** CITATIONS
SEE PROFILE

Vili Podgorelec
University of Maribor
192 PUBLICATIONS **2,869** CITATIONS
SEE PROFILE

Received September 2, 2020, accepted October 8, 2020, date of publication October 28, 2020, date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034343

Transfer Learning With Adaptive Fine-Tuning

GREGA VRBANČIČ, (Graduate Student Member, IEEE), AND VILI PODGORELEC, (Member, IEEE)
Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia

Corresponding author: Grega Vrbančič (grega.vrbancic@um.si)

This work was supported by the Slovenian Research Agency (Research Core Funding) under Grant P2-0057.

ABSTRACT With the utilization of deep learning approaches, the key factors for a successful application are sufficient datasets with reliable ground truth, which are generally not easy to obtain, especially in the field of medicine. In recent years, this issue has been commonly addressed with the exploitation of transfer learning via fine-tuning, which enables us to start with a model, pre-trained for a specific task, and then fine-tune (train) only certain layers of the neural network for a related but different target task. However, the selection of fine-tunable layers is one of the major problems of such an approach. Since there is no general rule on how to select layers in order to achieve the highest possible performance, we developed the Differential Evolution based Fine-Tuning (*DEFT*) method for the selection of fine-tunable layers for a target dataset under the given constraints. The method was evaluated against the problem of identifying the osteosarcoma from the medical imaging dataset. The performance was compared against a conventionally trained convolutional neural network, a pre-trained model, and the model trained using a fine-tuning approach with manually handpicked fine-tunable layers. In terms of classification accuracy, our proposed method outperformed the compared methods by a margin of 4.45% to 32.75%.

INDEX TERMS Deep learning, fine-tuning, medical imaging, optimization, transfer learning.

I. INTRODUCTION

In recent years the expansion of deep learning has been tremendous, with a significant impact on almost every field. The outstanding classification performance of modern deep learning approaches has attracted many researchers from various fields such as agriculture [1], seismology [2], information security [3], software engineering [4], computational biology [5], healthcare [6], and medicine [7] to employ modern deep learning techniques to solve different domain problems. One such field, where the increase of deep learning applications has led to a number of domain problems, is medical image processing [8]–[10]. However, especially in the medical image processing field, there is a problem with regard to the availability of large datasets, with reliable ground truth, which are needed in order to build a good predictive model with high predictive performance, utilizing deep learning methods.

Nowadays, the lack of adequate datasets is most commonly addressed by the utilization of transfer learning approaches [10]–[13]. Transfer learning enables us to transfer the knowledge of a model, previously trained for a specific task, to a new model, trying to tackle a similar but not equivalent task. Recent studies have shown that such approaches, especially when utilizing deep convolutional neural networks (CNN), are quite beneficial in terms of not needing a large dataset. In addition to that, the time complexity of training is also reduced since models are already somewhat pre-trained. For example, in [11] the authors achieved great performance results when classifying mammographic tumors with the use of transfer learning trained only on 607 full-field digital mammographic images. In [14] the authors utilize transfer learning to effectively classify images for macular degeneration and diabetic retinopathy.

However, with the application of transfer learning, there are also downsides. In order to appropriately adapt a pre-trained model to a new task, its specific parts (layers) need to be retrained, while the others need to remain unchanged. This adaptation is usually made with fine-tuning approaches when facing a problem in determining which of the layers should be enabled for training (fine-tuning) and which ones to leave frozen [15]. Besides, there is a common problem when setting the hyper-parameter values, the same as in the conventional training of deep neural networks. All these issues have a direct impact on training capabilities and also on the classification performance. If the hyper-parameter

The associate editor coordinating the review of this manuscript and approving it for publication was Liang Ding.

values are not set correctly, the outcome of a model is most likely to be unsatisfying [16]. Currently, there is no general rule or recipe to follow in order to determine which layers to fine-tune or which hyper-parameter settings to use. Most of the decisions are based on previous experiences of dealing with such problems. Solving the above-mentioned issues is most commonly a recurring, time-consuming process in which various settings are tried and tested out to find the ones that will result in a high predictive performance of the model.

The motivation behind the selection of fine-tunable layers is based on the empirical evidence that the initial (bottom) layers of CNNs preserve more abstract, generic features, applicable to a broader range of tasks as presented in [15], [17], [18]. In contrast, layers toward the end (top) of a CNN tend to provide more specific, task-related features. Therefore, it should generally be more reasonable to fine-tune more top layers. However, recent studies [19] show that it is not really clear if restricting the fine-tuning to the last layers is the best option. Azizpour *et al.* [17] suggested that the success of knowledge transfer depends on the dissimilarity between the primary task for which the CNN was trained, and new target task for which we would like to transfer knowledge. When utilizing a fine-tuning strategy, the CNNs are most often pre-trained on an ImageNet dataset [20] or datasets similar to ImageNet. The distance between natural images in an ImageNet dataset and medical imaging datasets is by no means negligible, so at this point, the question regarding which layers of CNN to fine-tune remains.

The mentioned empirical findings motivated us to take a look at the problem from a different angle. We found an analogy for this problem in the feature selection techniques and mechanisms utilized in machine learning, which are well explored. The problem of feature selection could be easily translated to the problem of finding the most optimal selection of layers for a given neural network architecture, enabled for fine-tuning, which would produce a predictive model achieving the best classification performance. Based on those grounds, we set our goals to develop a new straightforward, automatic and CNN architecture agnostic Differential Evolution based Fine-Tuning (*DEFT*) method. The *DEFT* method features an adaptive layer selection mechanism for finding the most optimal combination of fine-tunable layers for achieving high classification performance in transfer learning tasks.

The proposed method is evaluated against the task of identifying the osteosarcoma from Hematoxylin and eosin (H & E) stained images. The performance of the *DEFT* method is compared against a conventional trained CNN model, a pre-trained CNN model with all layers being fine-tuned, a CNN model with handpicked layers fine-tuned, and also specific state-of-the-art methods evaluated on the same dataset.

We summarize our contributions as follows:

- We propose a novel adaptive fine-tuning mechanism for transfer learning, which automatically determines how many and which layers of a CNN to fine-tune for a given set of images.
- We conducted an empirical evaluation of the proposed method tackling the problem of identifying osteosarcoma from medical images.
- We performed an extensive performance analysis and comparison of the results obtained from conducted experiments.
- We analyzed the impact of different selections of fine-tunable layers on the model's performance.

II. RELATED WORK

The problem of layer selection, when employing transfer learning with fine-tuning, has been receiving a lot of attention in recent years. With the general popularization of deep learning techniques, transfer learning with fine-tuning has become the most common strategy for transferring knowledge in the context of deep learning, which enables researchers and practitioners to apply such deep learning methods to various domain problems more quickly.

Various methods, following different strategies and approaches, were presented in recent studies to improve the standard fine-tuning. In general, there are two approaches addressing the aforementioned problem. The first approach focuses on selecting the input samples relevant to the target task, as presented in [21]–[23]. The other approach – that we also used in this paper – focuses on the selection of the portions, or layers, of the network in order to optimize information extraction from the pre-trained network. Standard techniques adopting this approach either fine-tune all network layers as presented in [24], or only fine-tune the last few layers or blocks of the network as presented in [10], [25]. Another interesting technique proposed in [26], [27] is to use a pre-trained network as a feature extractor with a classifier such as SVM on top of it. While those fine-tuning methods have proven that they are capable of delivering a promising performance when compared to conventionally trained networks, the biggest drawback of such techniques is their inability to adapt automatically. Therefore, to apply such a method, one needs to adjust various parameters manually or perform layer selection by hand, which is commonly seen as a challenging, burdensome task.

In 2018, Guo *et al.* [19] presented a method more related to ours, which also works in an adaptive manner. The method features the automatic layer selection per target instance. Introducing the policy network, the authors achieved the adaptiveness of the method. The policy network is used to make routing decisions on whether to pass the image through the fine-tuned layers or through the pre-trained layers. Combined with ResNet CNN architecture's ability to be resilient to residual block swapping and dropping [28], the method delivers encouraging classification performance. Although the authors stated that the method could be applied to different neural network architectures, such an application would most certainly not be a straight-forward task because of the method's heavy dependence on the mentioned ResNet architecture abilities.

In contrast to the mentioned methods, our proposed DEFT method features an automatic adaptive layer selection mechanism, which works per target dataset, while also being CNN architecture agnostic, since it does not rely whatsoever on the capabilities of a particular architecture.

III. METHODS

A. TRANSFER LEARNING

Conventional machine learning approaches make future predictions based on the statistical models, trained on previously collected, labeled, or unlabeled data. An approach that utilizes the labeled data in the process of model training is most commonly referred to as supervised learning, while the utilization of unlabeled data in the process of model training is commonly known as unsupervised or self-supervised training. When dealing with small, insufficient sets of labeled data, building a good classifier is a hard and burdensome task. Many studies [29]–[31] have been conducted to tackle this issue, utilizing semi-supervised training or some variation of such an approach where the usage of a large unlabeled set of data and a small set of labeled data is combined. The most common issue with such approaches is the assumption that the labeled and unlabeled data distributions are the same [32]. In contrast to semi-supervised approaches, transfer learning enables the domains, tasks and data distributions to be different.

The first studies to focus on transfer learning date back to 1995 [33]. However, it can be found under different names such as inductive transfer [34], multitask learning [35], incremental/cumulative learning [36], with one of the most closely related learning techniques for the transfer learning approach being the multitask learning framework [35]. In general, the transfer learning technique can be defined as the improvement of learning a new task through the transfer of knowledge from a related task that has already been learned. In machine learning terms, as presented in Fig. 1, transfer learning roughly translates to transferring the weights of the already trained model, specialized for a specific task, to the model solving a different, but related task [37]. With the expansion of the deep learning field, transfer learning also gained momentum, due to the requirements common to all deep neural networks – the need for large datasets. Additionally, the process of training deep neural network architectures requires a lot of computational power and thus is a time-consuming task. In such cases, with the utilization of the transfer learning techniques, one could benefit significantly in terms of time complexity as well as in terms of the large, required dataset.

In general, transfer learning techniques are used in two ways – one being the approach where the weights of the pre-trained model are preserved (frozen) on some of the layers and fine-tuned (trained) in the remaining layers, and the other being the approach where the pre-trained deep neural network is utilized as a feature extractor, while the extracted features are fed to the classifier of choice [38]. Our research focuses on the first-mentioned transfer learning approach or strategy known as fine-tuning – the mechanism which is presented in-depth in the following section.

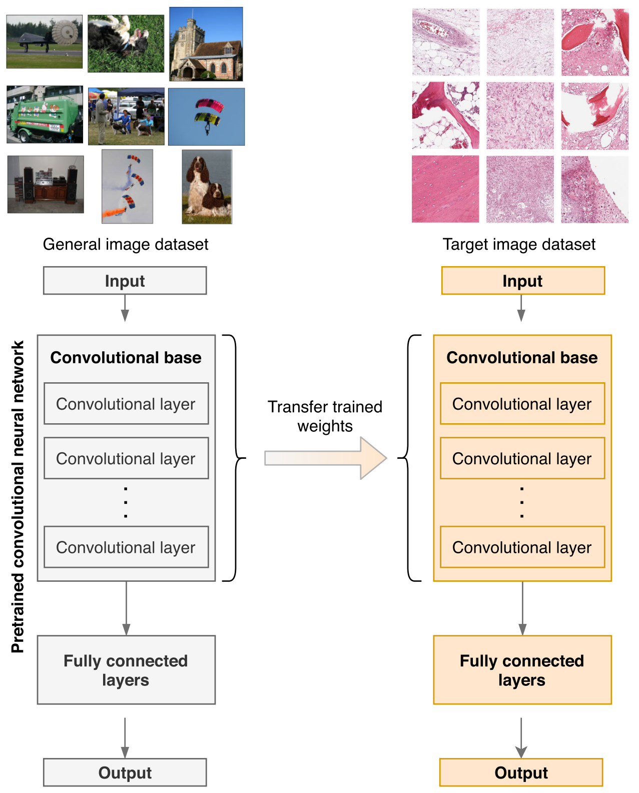


FIGURE 1. **The conceptual diagram of transfer learning technique.**

B. FINE-TUNING

A general transfer learning approach is to train a base network and then copy its first n layers to the first n layers of the target network. The new target network's remaining layers are most commonly randomly initialized and trained for a specific target task. We can also choose to backpropagate the errors from the new task into the base features to fine-tune them for the new task, or we can freeze some of the feature layers, which are not going to be trained (fine-tuned) against the new task [15].

The fine-tuning approach is one of the most popular transfer learning strategies among applications in neural networks. Its use was pioneered in [39] by transferring knowledge from a generative to a discriminative model, thereby achieving high generalization. The initial pipeline was composed out of a pre-trained network where the last classifier layer was replaced with a randomly initialized one.

Nowadays, the concept of fine-tuning a strategy is quite similar. While we do not, in general, replace the last classifier layer with randomly initialized one, we most commonly enable fine-tuning of some of the layers in a pre-trained neural network instead of replacing them. However, at this point, the question of how to select fine-tunable layers for a pre-trained network model arises.

The last few layers of a deep neural network are usually fine-tuned, while the remaining initial layers are kept frozen with their initial pre-trained values. The motivation behind such a strategy is driven by a combination of size-limited datasets and some empirical evidence that the initial layers (bottom layers) of a deep neural network preserve more abstract, generic features. Such features are commonly applicable to a broader range of tasks, while the layers closer to the top provide more specific task-related features [19]. However, while the selection of fine-tunable layers is still a more or less manual process, which most commonly requires a tremendous amount of experimenting, and while there are significant number of empirical studies [10], [18], [40] which show great success with regard to limiting the fine-tuning of the last few layers, there are also some recent studies [19], [28] which diminish the assumption that the early or middle layer features should be shared. Based on the mentioned empirical findings and given that there is no general rule or recipe to follow when selecting fine-tunable layers, we decided to tackle the problem from the perspective of representing it as an optimization problem and tried to solve it utilizing a well-known optimization meta-heuristic algorithm.

C. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is one of the most popular population-based meta-heuristic algorithms, introduced by Storn and Price in 1997 [41]. Thanks to many wins at international competitions, DE is considered one of the most appropriate algorithms for continuous optimization. Besides the general popularity of the DE algorithm for solving optimization tasks, it was also successfully applied to various machine learning problems, such as the hyper-parameter optimization problem [42], the problem of designing neural network architecture [43] or the feature selection problem [44].

The DE algorithm is composed of *Np* real-coded vectors and three operators: mutations, crossovers and selections. The *Np* real-coded vectors are representing the candidate solutions (individuals) which can be formally defined as presented in Eq. 1, where each element of the solution is in the interval $x_{i,1}^{(t)} \in [x_i^{(L)}, x_i^{(U)}]$, while $x_i^{(L)}$ and $x_i^{(U)}$ denote the lower and upper bounds of the i-th variable, respectively.

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

The DE's basic strategy consists of mutation, crossover, and selection operations. The mutation operation can be formally expressed as follows:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r1}^{(t)} + F \cdot (\mathbf{x}_{r2}^{(t)} - \mathbf{x}_{r3}^{(t)}), \quad \text{for } i = 1, \ldots, Np, \tag{2}$$

where F represents the scaling factor as a positive real number that scales the rate of modification while $r1$, $r2$ and $r3$ are randomly selected values in the interval $1 \ldots Np$.

In order to increase the diversity of the parameter vectors, the crossover operator is introduced as presented in Eq. 3, where $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution.

$$w_{i,j}^{(t+1)} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0,1) \leq CR \vee j = j_{rand}, \\ x_{i,j}^{(t)} & \text{otherwise}, \end{cases} \tag{3}$$

Finally, the selection operator is utilized to decide whether a produced vector should become a generation member utilizing the greedy criterion. The selection could be formally expressed as follows:

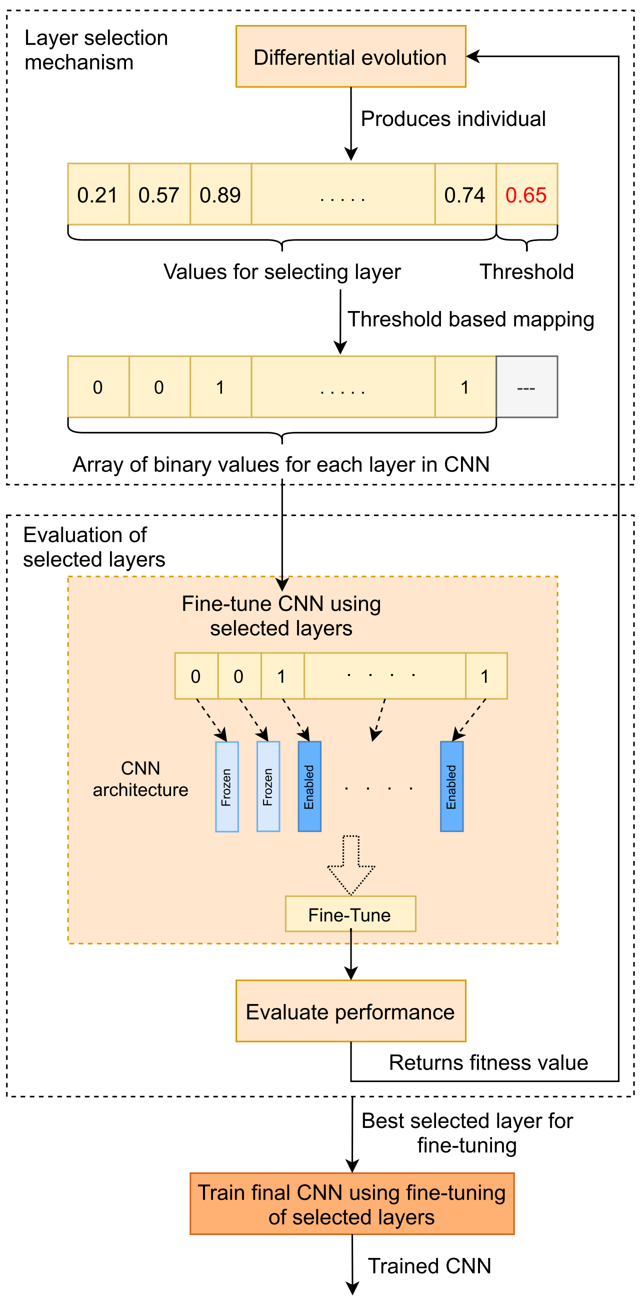
$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise}. \end{cases} \tag{4}$$

where $f(\mathbf{x}_i^{(t)})$ denotes the fitness function defined for solving a specific optimization problem. The DE algorithm works in an iterative manner, where each produced solution is evaluated using the given fitness function. Based on the selection operator, presented in Eq. 4, the DE algorithm's search mechanism seeks solutions for ever-better fitness scores. Naturally, the DE algorithm is optimized for solving the minimization optimization problems, and therefore the fitness function should be tailored to give the lower score to the preferred solutions.

The choice of DE parameters, namely *Np*, *F*, and *CR* can have a enourmous impact on optimization performance. The *Np* parameter, as presented, denotes the population size of real-coded vectors (individuals) on top of which the mutation, crossover, and selection operators are applied. Based on research by Piotrowski [45], a too small population size limits the number of available moves, which may lead to stagnation (the population stops proceeding towards the optimum, although population diversity remains high) or premature convergence. On the contrary, a vast population slows down individuals' clustering and frequently wastes many function calls on almost random explorative moves. The *F* parameter is a scale factor which controls the length of the exploitation vector and thus determines how far from point x_i the offspring should be generated. Based on research from Das *et al.* [46], a good initial choice of *F* is 0.5, while the effective range of *F* is usually between 0.4 and 1. The DE parameter *CR* controls how many parameters are expected to change in a population member. When *CR* is set to a low value, a small number of parameters are changed in each generation, and the step-wise movement tends to be orthogonal to the current coordinate axes. On the other hand, high values of *CR* cause most of the mutant vector directions to be inherited, prohibiting the generation of axis orthogonal steps [46].

IV. ADAPTIVE FINE-TUNING

To tackle the problem of finding and selecting which layers of given a CNN architecture to fine-tune, we have developed the adaptive *DEFT* (Differential Evolution based Fine-Tuning) method. As the problem of finding and selecting the layers to fine-tune can be easily translated into an optimization problem, we adopted the DE algorithm for the purposes of finding the most optimal solution to the problem. The solution represents an optimal combination of layers, which should be


Layer selection mechanism
Differential evolution
Produces individual
0.21
0.57
0.89
.....
0.74
0.65
Values for selecting layer
Threshold
Threshold based mapping
0
0
1
.....
1

Array of binary values for each layer in CNN
Evaluation of selected layers
Fine-tune CNN using selected layers
0
0
1
....
1
CNN architecture
Frozen
Frozen
Enabled
Enabled
Fine-Tune
Evaluate performance
Returns fitness value
Best selected layer for fine-tuning
Train final CNN using fine-tuning of selected layers
Trained CNN

FIGURE 2. The conceptual diagram of the Differential Evolution based Fine-Tuning (*DEFT*) method.

selected for fine-tuning in order to train the best performing CNN.

In Fig. 2 the conceptual diagram of the *DEFT* method is presented. The *DEFT* method is composed of two components: the layers selection mechanism and the evaluation of selected layers. The layers selection mechanism is responsible for providing the individual, i.e., array of binary values, where each value in the array reflects whether the corresponding layer of CNN architecture is selected for fine-tuning or not. Based on the provided individual, the evaluation component fine-tunes the CNN model and evaluates its predictive performance. The evaluation of CNN is performed using a fitness function, for which we utilized the well-known categorical cross-entropy (*CCE*) loss. The performance evaluation score (fitness value) is passed back to the layers selection mechanism, based on which the new individual is produced. The process reiterates, trying to minimize the fitness value, until the maximum number of model evaluations is reached. The maximum number of model evaluations is a parameter which can be set manually and represents a total number of produced individuals within the DE loop. The best individual (the selection of layers that produced the best performing CNN – with the lowest categorical cross-entropy loss) is deemed to be optimal under the given circumstances. The optimal found selection of layers is then used for fine-tuning the final CNN.

Each of the *DEFT* components is explained in detail in the following subsections.

A. LAYERS SELECTION MECHANISM

The layers selection mechanism is based on the DE algorithm, modified for the task of selecting which layers of CNN architecture will be enabled for fine-tuning and which will remain frozen. In order to be able to perform the layer selection using the DE algorithm, it is mandatory to introduce the modifications to the representation of the individuals for the DE optimization process.

The individuals in the proposed *DEFT* method are presented as an array (a vector) containing real values, which can be formally expressed as presented in Eq. 5, for $i = 0, \ldots, Np$, where each layer $x_{i,0}^{(t)}$ for $i = 0, \ldots, N$ is selected from the interval [0, 1] and N represents the total number of layers of the selected CNN architecture.

$$\mathbf{x}_i^{(t)} = (x_{i,0}^{(t)}, \ldots, x_{i,N}^{(t)}, Th_i^{(t)}), \tag{5}$$

For the purpose of obtaining more diverse individuals, we exploited the moving threshold method initially presented in [47]. The mechanism behind the moving threshold method is that the threshold value is part of an individual solution vector and is therefore set dynamically for each solution candidate, which removes the need for setting it manually. In addition to that, it provides us with potentially more diverse individuals. The *Th* denotes the threshold value, based on which the mapping function presented in Eq. 6 determines whether a corresponding layer is enabled for fine-tuning or not.

$$s_{i,j}^{(t)} = \begin{cases} 1, \text{if } x_{i,j}^{(t)} > Th^{(t)} \\ 0, \text{otherwise} \end{cases} \tag{6}$$

The individual vector s_i presents the mapped array of binary values (0 and 1) for i-th individual, where the value 0 for j-th layer reflects in a layer not being selected for fine-tuning, while the value 1 reflects in j-th layer being selected for fine-tuning.

B. EVALUATION OF SELECTED LAYERS

For each produced individual s – described in the previous subsection – the fine-tuning of CNN is conducted. To determine how good or bad the produced individual is, we define a fitness function L. The fitness function is calculated after CNN model fine-tuning based on the given individual is finished and returned to the DE algorithm in order to enable the DE to find better individuals. The fine-tuning is conducted for a maximum number of epochs utilizing the early stopping technique to stop the fine-tuning of not-so-promising layer selections. The maximum number of epochs is a parameter that can be set manually; in general, it should be set with regard to a target dataset. To evaluate the performance of a model, trained by fine-tuning the layers selected by the produced individual, we adopted a well-known categorical cross-entropy (CCE) loss function [48], which can be formally expressed as follows:

$$L = CCE = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{C} y_{ij} log(p_{ij}) \tag{7}$$

where i indexes samples from the total of M samples, j indexes classes from total of C classes, y denotes the sample label, and $p_{ij} \in (0, 1) : \sum_j p_{ij} = 1 \forall i, j$ represents the prediction for a sample.

C. TRAINING THE FINAL CNN MODEL

To obtain the best performing selection of layers, a subset of 80% of the training set was used within the DE loop for training CNN models based on the DE algorithm's individual solution. The remaining 20% of the training set is then used to evaluate the performance of trained CNN models using the fitness function in Eq. 7.

After all candidate solutions are evaluated, the best performing one is selected, based on which the combination of layers selected for fine-tuning is used to fine-tune the final CNN. In contrast to the models trained on candidate solutions, this final CNN model is trained from the start upon the whole training set. This final CNN model is the result of the *DEFT* method.

V. EXPERIMENTAL SETUP

To evaluate the performance of the proposed adaptive *DEFT* method, the experimental approach was utilized. Experiments were conducted with four methods – the three compared methods and our proposed *DEFT* method, tackling the task of identifying the osteosarcoma from Hematoxylin and eosin (H & E) stained osteosarcoma images:

- *conventional* where the conventional approach for training a CNN was utilized,
- *pretrained* where the CNN was trained using the conventional approach with pre-trained weights,
- *baseline* where the transfer learning approach with the fine-tuning of handpicked layers of CNN architecture was used, and
- *DEFT* where the transfer learning approach with our proposed method was utilized to select the most optimal fine-tunable layers.

To objectively evaluate the performance of compared methods, the experiments were conducted in two different scenarios. In the first scenario, the *conventional*, *pretrained*, and *baseline* methods are utilizing the same early stopping technique as the *DEFT*. In contrast, in the second scenario, no early stopping criteria is used for the three compared methods.

All of the conducted experiments were implemented in the Python programming language with the use of the following libraries: Keras [49] with Tensorflow [50] backend for developing and training CNNs, NiaPy [51] for providing a DE algorithm implementation, and PyCM [52] for classification performance metrics calculation.

The experiments were executed on a single Intel Core i7-6700K based PC, with 4 cores (8 threads) CPU running at 4 GHz, with 64 GB of RAM, and three Nvidia GeForce Titan X Pascal GPUs each with 12 GB of dedicated GDDR5 memory, running the Linux Mint 19 operating system.

TABLE 1. The basic information about *Osteosarcoma data from UT Southwestern/UT Dallas for Viable and Necrotic Tumor Assessment* dataset class distribution.

Label	Number of images
Non-tumor	536 (47%)
Necrotic tumor	263 (23%)
Viable tumor	345 (30%)
Total	1144

A. DATASET

The proposed *DEFT* method is evaluated on the problem of identifying the osteosarcoma from hematoxylin and eosin (H & E) stained osteosarcoma images. We used a publicly available dataset *Osteosarcoma data from UT Southwestern/UT Dallas for Viable and Necrotic Tumor Assessment* [53], the properties of which are presented in Table 1. The data were collected by clinical scientists at the University of Texas Southwestern Medical Center, Dallas. The archival samples of 50 patients treated at the Children's Medical Center, Dallas, between 1995 and 2015, were used to create this dataset from which 942 histology glass slides were digitized into whole slide images (WSI). From those, two pathologists manually selected 40 WSIs representing the heterogeneity of a tumor and response characteristics under study. Thirty 1024 x 1024 pixel image tiles at 10X magnification factor, as seggested by the pathologists, were randomly selected from each WSI. From the resulting 1,200 images tiles, 66 irrelevant image tiles such as image tiles falling in non-tissue, ink-mark regions, and blurry images were removed. The performed randomization of tile-generation was conducted to remove any bias in the dataset, prepared for feature-generation and subsequent machine/deep-learning steps. Two medical experts performed

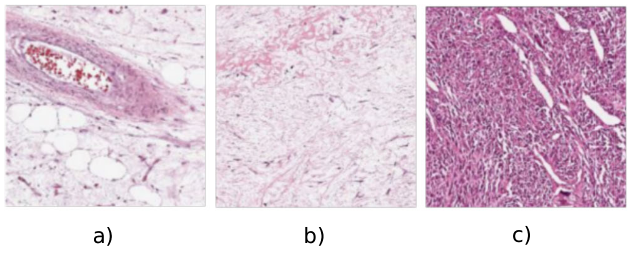


FIGURE 3. Sample images from the dataset. The a) sample represents the non-tumor image, b) represents the necrotic tumor image while c) represents the viable tumor image.

the labeling and annotation, each labeling half of the image tiles [54].

In the pre-processing data phase, images were scaled down to 224 × 224 pixels, which is the default input size of the utilized VGG19 CNN architecture. Additionally, numerous studies have shown that such scaling when utilizing VGG19 architectures results in encouraging classification performance [10], [12], [40]. Fig. 3 shows the samples of the individual target class, namely non-tumor, necrotic tumor, and viable tumor. The viable tumor can be identified based on the nuclei (cells) densely aggregated together while the necrotic tumor reflects on images with disintegrated nuclei but with less color density than the viable tumor. In medical terms, the necrotic tumor denotes the dying parts of the tumor cells while the viable tumor denotes the H&E stained tissue images where the tumor cells are capable of normal growth.

B. CONVOLUTIONAL NEURAL NETWORK

As a CNN architecture, we adapted the original VGG19 CNN architecture presented in Fig. 4, initially presented in [55] (denoted as configuration E in the original paper). At the bottom of the VGG19 CNN convolutional base is an input layer consuming the 224 × 224 pixel RGB images. The input layer is followed by five convolutional blocks presented in Table 2. Each convolutional block comprises several convolutional layers chained one after another, followed by a maximization pooling layer.

The choice of selecting the VGG19 CNN architecture is based on the encouraging results presented in various studies where the VGG19 CNN with transfer learning was utilized for different target tasks. For example, the VGG19 is adopted for automatic glaucoma classification in [12], for breast cancer histology images classification in [13], and for hologram classification for molecular diagnostics in [56].

In the case of the *pretrained*, *baseline* and *DEFT* experiment, the VGG19 convolutional base was pre-trained on an ImageNet dataset, while in the case of a *conventional* experiment, the CNN is trained from scratch.

C. PARAMETER SETTINGS

1) CNN SETTINGS

For the *conventional* method, the VGG19 convolutional base is used with the random weights initialization; for the *pretrained* method all layers were fine-tuned, while for the *baseline* method the selection of layers enabled for fine-tuning was done manually. Based on the encouraging results from various studies [10], [40], we followed the strategy of fine-tuning only the last convolutional block (in our case *block5* of VGG19 architecture) in the convolutional base, while the layers towards the beginning of the convolutional base were kept frozen. For the *DEFT* method, the fine-tunable layers were dynamically chosen by utilizing the proposed adaptive approach.

On top of the convolutional base from *VGG19* CNN architecture, after the last convolutional block, in each experiment, the following randomly initialized feed-forward layers are added and trained in order to perform the classification task (see Fig. 4):

- flatten layer,
- dropout layer with dropout probability set to 0.8,
- dense layer with 512 neurons and *ReLU* activation function,
- dropout layer with dropout probability set to 0.5, and
- dense layer with 3 neurons (number of target classes) and *Softmax* activation function.

TABLE 2. VGG19 convolutional base. The convolutional layers are denoted as "convolution (kernel size), number of kernels".

Block	Layers
conv_block1	convolution (3x3), 64 convolution (3x3), 64 maxpooling
conv_block2	convolution (3x3), 128 convolution (3x3), 128 maxpooling
conv_block3	convolution (3x3), 256 convolution (3x3), 256 convolution (3x3), 256 convolution (3x3), 256 maxpooling
conv_block4	convolution (3x3), 512 convolution (3x3), 512 convolution (3x3), 512 convolution (3x3), 512 maxpooling
conv_block5	convolution (3x3), 512 convolution (3x3), 512 convolution (3x3), 512 convolution (3x3), 512 maxpooling

2) TRAINING PARAMETERS

For each of the conducted experiments, the training parameters were set as presented in Table 3. The training parameters' values were picked based on previous experiences utilizing CNNs for various image recognition tasks [10], [57], [58]. Each experiment was trained using the adam optimizer function with the initial learning rate set to $1 * 10^{-4}$ and batch size set to 128. For the *conventional*, *pretrained* and *baseline*

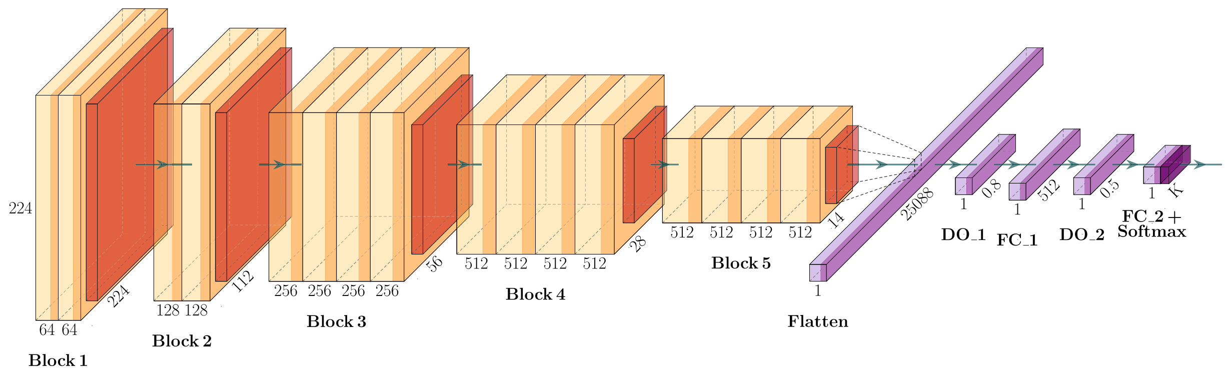


FIGURE 4. The presentation of the utilized CNN architecture. The convolutional base (Blocks 1 - 5) is adopted from VGG19 CNN architecture. The dropout layers and fully connected layers are denoted with *DO* and *FC*, respectively.

TABLE 3. Used training parameters for the conducted experiments.

Parameter	**conventional, pretrained, baseline**	**DEFT**
Optimizer	Adam	Adam
Initial learning rate	$1 * 10^{-4}$	$1 * 10^{-4}$
Batch size	128	128
Epochs	500	20

TABLE 4. Used parameter values for the DEFT method.

Parameter	**Value**
Dimension of problem D	22 (N +1)
Population size Np	10
Scale factor F	0.5
Crossover probability CR	0.8
Number of function evaluations	50

experiments, the training was performed for 500 epochs. For the *DEFT* experiment, the maximum number of epochs was set to 20, which means that the model was fine-tuned for up to 20 epochs for each produced individual. Additionally, the early stopping technique was utilized, which was configured to stop the training if there were no improvement in lowering the loss value for any three consecutive epochs. The maximum number of epochs was selected experimentally, as in the vast majority of cases the early stopping criteria was met before reaching 20 epochs; in general, it should be set with regard to a target dataset.

3) *DEFT* PARAMETERS

For the proposed *DEFT* method, we set the parameters as presented in Table 4. The total number of function (model) evaluations was set to 50, while each of the individuals had a maximum number of epochs set to 20 to achieve the lowest (best) possible loss. The selected combination of layers for fine-tuning, which trained the CNN with the lowest (best) loss value, was then used to fine-tune the final CNN from the beginning for the whole 20 epochs; all the other fine-tuned models were discarded. Due to the non-deterministic nature of the underlying DE, the experiment conducted with the *DEFT* method was executed in 10 independent runs. Therefore, all the reported results are the averages across all those runs.

D. EVALUATION METHOD AND METRICS

To objectively evaluate the performance of the proposed *DEFT* method against the compared methods, we conducted a gold standard 10-fold cross-validation procedure, where the given dataset was divided into training and testing sets at a ratio of 90:10. In this way, the process was repeated for a total of ten times, each time leaving a different 10% of the initial dataset for testing. The 10-fold cross-validation provides a more reliable evaluation of the classification performance since, for each experiment, we train CNN on ten different training sets and, more importantly, evaluate the trained model on 10 test sets. It enables us to observe how the method behaves when trained and evaluated on different subsets of the target dataset. Additionally, such an approach can also highlight the potential layer selection bias of the used method.

In the process of conducting the 10-fold cross-validation, we calculated well-known classification metrics such as accuracy, the *F-1* macro and micro score, and AUC. As we are dealing with the a multi-class classification problem, we utilized the AUC metric for multi-classification problems, namely *AUNP*, and the Cohen's Kappa coefficient, which successfully handles multi-classification as well as imbalanced classification problems. Since we have both multi-classification and a bit of an imbalanced dataset, it seemed like the reasonable choice to use the Cohen's Kappa coefficient as one of the performance metrics.

The *AUNP* metric is calculated as the *AUC* of each class against the rest, using the a priori class distribution. The *AUC* (Area under the ROC Curve) of a binary classifier, formally defined in Eq. 8, is equivalent to the probability that the classifier's rank of randomly chosen positive instance is higher than a randomly chosen negative instance. The $f(i,j)$ represents the actual probability of the example i to be class j, where we assume that $f(i,j)$ always takes a value of 0 or 1 and is strictly an indicator function. With $m_j = \sum_{i=1}^{m} f(i,j)$, we denote the number of class examples j, and with $p(i,j)$, we denote

TABLE 5. Averages of classification performance metrics over 10 folds when classifiers were trained for up to 500 epochs with early stopping criteria.

	conventional	pretrained	baseline	DEFT
Accuracy	0.5485	0.5382	0.5985	**0.8657**
AUNP	0.5860	0.6009	0.6479	**0.8939**
F1-macro	0.3652	0.3612	0.4159	**0.8483**
Kappa	0.1726	0.1961	0.2873	**0.7866**
Epochs	**7.1**	7.3	10.4	461.7
Time	79.6	41.5	**40.1**	2447.5

TABLE 6. Averages of classification performance metrics over 10 folds when classifiers were trained for full 500 epochs.

	conventional	pretrained	baseline	DEFT
Accuracy	0.8209	0.6476	0.6945	**0.8657**
AUNP	0.8592	0.7206	0.7198	**0.8939**
F1-macro	0.8053	0.5491	0.5641	**0.8483**
Kappa	0.7172	0.4423	0.4383	**0.7866**
Epochs	500	500	500	**461.7**
Time	2162.7	1457.5	**1077.1**	2447.5

* The DEFT method uses the early stopping mechanism in both experiments.

the prior probability of class j. In other words, the $p(i, j)$ represents the estimated probability of example i to be of class j taking the values in [0, 1]. The $l(\cdot)$ is a comparison function satisfying $l(a, b) = 1$ if $a > b$, $l(a, b) = 0$ if $a < b$ and $l(a, b) = 0.5$ if $a = b$ [59].

$$AUC(j, k) = \frac{\sum_{i=1}^{m} f(i, j) l(p(i, j), p(t, j))}{m_j \cdot m_k} \tag{8}$$

The *AUNP* metric introduced in 2001 by Fawcett [60] computes the *AUC* treating a c-dimensional classifier as c two-dimensional classifiers, taking into account the prior probability of each class ($p(j)$). Formally, the *AUNP* can be expressed as follows:

$$AUNP = \sum_{j=1}^{c} p(j) AUC(j, rest_j), \tag{9}$$

where $rest_j$ represents all classes different from class j.

VI. RESULTS

Using the presented experiment setup, evaluation method and classification metrics, we obtained the results, which are presented and discussed in detail in the following subsections.

A. CLASSIFICATION PERFORMANCE COMPARISON

The classification results obtained from the conducted experiments identifying the osteosarcoma from H&E stained osteosarcoma images, using the dataset *Osteosarcoma data from the UT Southwestern/UT Dallas for Viable and Necrotic Tumor Assessment*, are presented in Table 5 and Table 6. Table 5 shows the averages of various classification performance metrics over ten folds when using the three compared methods (*conventional*, *pretrained* and *baseline*), which were trained up to 500 epochs with the aid of early stopping criteria. On the other hand, the results presented in Table 6 reveal the same metrics for the case where all three compared methods had been trained for a full 500 epochs. Our *DEFT* method uses the same approach (up to 50 iterations of 20 epochs with early stopping) in both cases.

In both cases, for all the classification metrics (accuracy, *AUNP*, *F1*-macro and Cohen's Kappa coefficient), the results of the *DEFT* method stand out, outperforming all the compared methods by a great margin, although the difference is smaller in cases where compared methods are trained for a full 500 epochs. For all three compared methods, the results are better in cases when they were trained for 500 epochs. In terms of accuracy, the *DEFT* method outperforms the second-best *conventional* method by a margin of 4.5% and the other two methods by a margin of 17.1-21.8%. Focusing on *AUNP*, *F1*-macro and Cohen's kappa coefficient the performance improvements are 3.5-17.4%, 4.3-29.9% and 6.9-34.8% respectively.

It is interesting to observe that in cases where early stopping was used, the results of all three compared methods are quite similar, with the best of them being the *baseline*. However, in the case of a full 500 epochs of training, the *conventional* method outperformed the other two methods by a significant margin, while the other two performed very similarly. This is an intriguing behavior since, due to the small dataset, we would expect that the *pretrained* and especially the *baseline* method would, overall, perform better than the conventionally trained one, due to the high risk of extreme over-fitting of such deep CNN architectures trained against a small dataset. Also, the handpicked selection of fine-tunable layers of a *baseline* method was made based on the previous empirical results in which such a selection of layers proved to be successful.

1) CLASSIFICATION PERFORMANCE METRICS: ACCURACY, F1-SCORE, AUNP AND KAPPA

In order to perform a more in-depth classification performance analysis of our proposed *DEFT* method, we present the performance comparison among the compared classifier methods on 10 folds using the box-dot-plot visualizations for accuracy, F1-score, AUNP and Cohen's kappa coefficient. For each metric, we performed two experiments: a) first, we compared the results of training the classifiers for up to 500 epochs using the early stopping criteria, and b) second, we compared the results of training the classifiers for a full 500 epochs.

Fig. 5 presents the accuracy performance comparison among the compared methods in the case of using early stopping. We can easily observe that the *DEFT* method outperforms the three compared methods, both in terms of mean accuracy over 10-folds as well as in terms of the standard deviation of the accuracy. The small standard deviation of accuracy also shows the capability of our method to generalize well. The second best method in terms of overall accuracy seems to be the *baseline*, followed by the *pretrained*

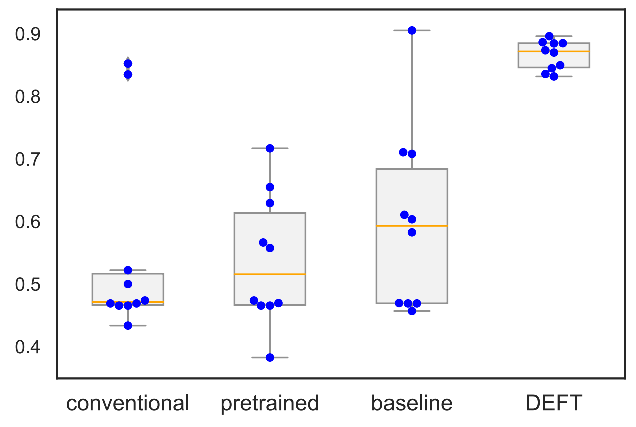


FIGURE 5. Accuracy of compared methods trained for up to 500 epochs with early stopping; each dot represents one fold.

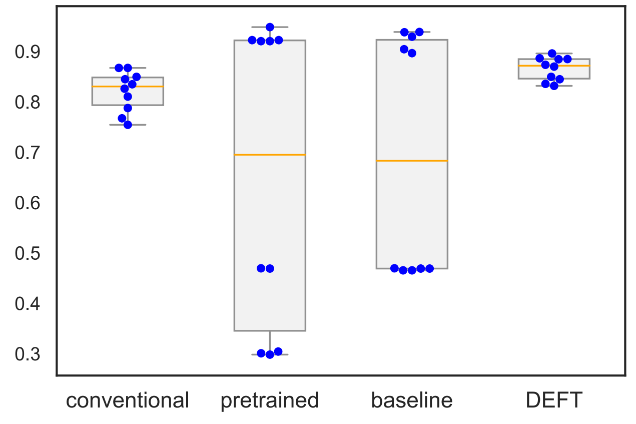


FIGURE 6. Accuracy of compared methods trained for full 500 epochs; each dot represents one fold.

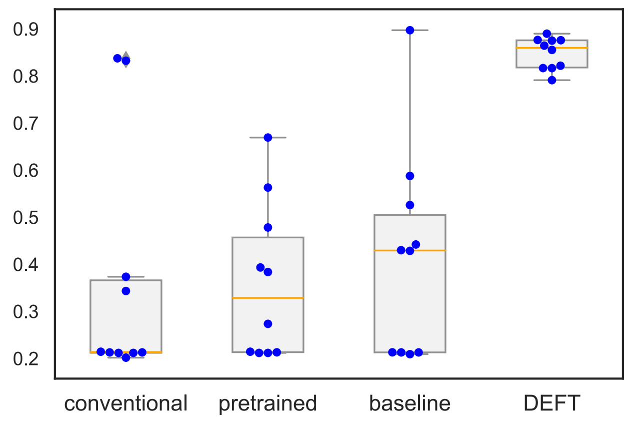


FIGURE 7. F1-macro of compared methods trained for up to 500 epochs with early stopping; each dot represents one fold.

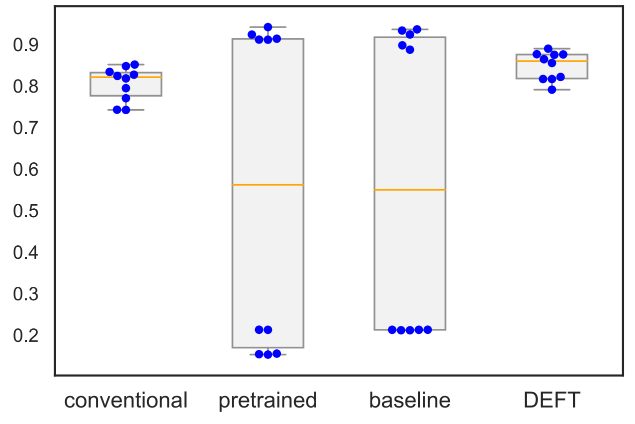


FIGURE 8. F1-macro of compared methods trained for a full 500 epochs; each dot represents one fold.

one, while the *conventional* method performed the worst. The accuracy results of the three compared methods are in line with expectations. While the *conventional* method, trained from scratch, was not able to achieve good performance within a few epochs, the *pretrained* method benefited from the pre-trained weights. The *baseline* method benefited even more as the CNN with the pre-trained weights was fine-tuned with regard to reasonably hand-picked layers only rather than all layers as the *pretrained* method.

Although all four methods used the same early stopping criteria and could have been theoretically trained for the same amount of epochs, it turned out that the three compared methods used much fewer epochs to train (see Table 5). In order to make a more fair comparison, in the second experiment, we fixed the number of epochs at 500 for all three compared methods by removing the early stopping mechanism (Fig. 6). We can see that the results of all three compared methods improved, but were still not able to outperform our *DEFT* method, both with regard to the overall accuracy and standard deviation. The method which benefited the most from prolonged training was the *conventional*, while the other two methods clearly finished in local minima in several folds. This benefit could be attributed to the quite drastic regularization applied in the top layers with the utilization of two dropout layers. The side effect of the applied regularization in the case of the *conventional* method resulted in a slow but steady convergence throughout training. In contrast, for the methods that utilized fine-tuning, some sort of regularization was necessary to mitigate the over-fitting, which is generally a quite common effect when dealing with transfer learning approaches.

Very similar results can also be observed for the other three performance metrics: F1-score, AUNP, and kappa.

The F1-score can be, in general, interpreted as a harmonic mean of the precision and recall score. When dealing with a multi-classification problem, we obtain a per class F1-score, which we would like to represent in the form of one value representing the classifiers' performance. One possible way to achieve that is to use a macro-averaged F1-score, which is computed as the simple arithmetic mean of per-class F1-scores. The F1-score results are presented in Fig. 7 and Fig. 8.

The results of AUNP metric performance for all compared methods are presented in Fig. 9 and Fig. 10. AUNP is a metric that combines the AUC measure of each class against the rest, using the a priori class distribution. It is one of the most common AUC variations when dealing with multi-class classifiers, as in our case.

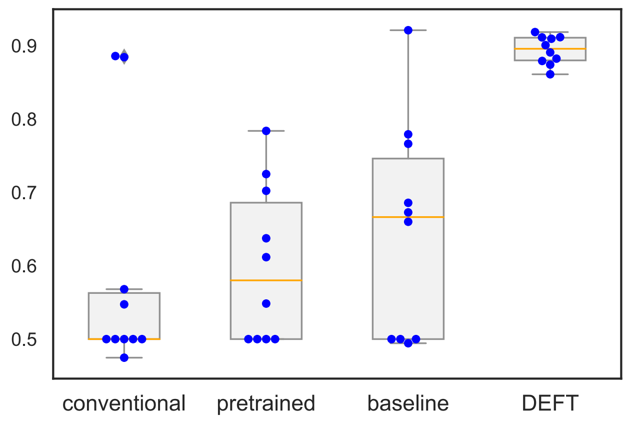


FIGURE 9. AUNP of compared methods trained for up to 500 epochs with early stopping; each dot represents one fold.

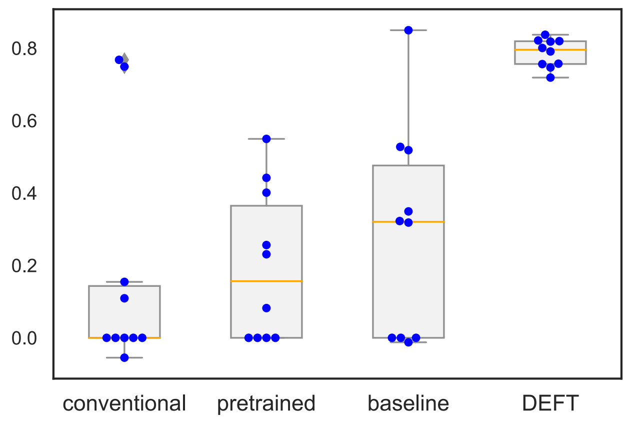


FIGURE 11. Kappa of compared methods trained for up to 500 epochs with early stopping; each dot represents one fold.

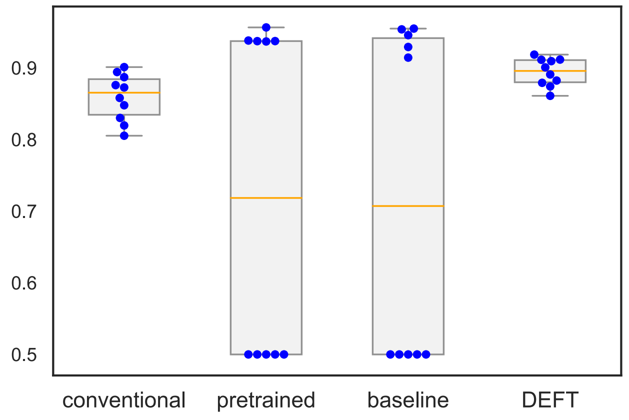


FIGURE 10. AUNP of compared methods trained for a full 500 epochs; each dot represents one fold.

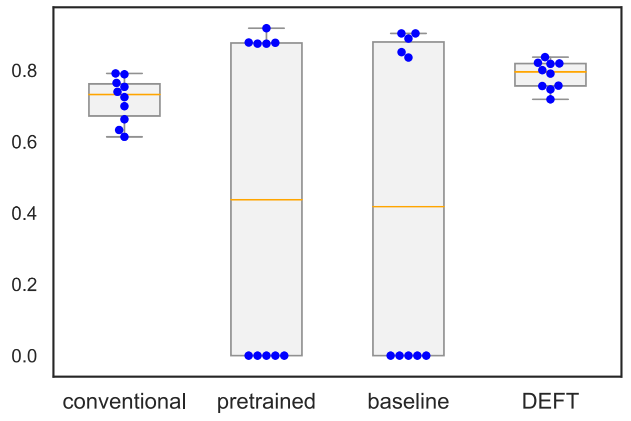


FIGURE 12. Kappa of compared methods trained for a full 500 epochs; each dot represents one fold.

Fig. 11 and Fig. 12 shows a comparison of Cohen's Kappa coefficient values for all of the compared methods. Essentially, the Cohen's Kappa metric compares an observed accuracy with an expected accuracy (random chance), which is generally less misleading than the accuracy metric itself, due to the Cohen's Kappa taking random chance into account. Fleiss's characterization [61] of the Kappa coefficient values are translated into a poor agreement when the value is < 0.40, fair to a good agreement when the value is $0.40 - 0.75$ and an excellent agreement when the value is > 0.75. Following the above-mentioned characterization of Cohen's Kappa coefficient values, our proposed *DEFT* method, on average, achieves an excellent agreement with an average of 0.7866. On the other hand, the remaining compared methods achieved, on average, a poor agreement in the case of using early stopping (see Table 5) and fair to good agreement in the case of not using early stopping (see Table 6).

2) COMPUTATIONAL AND TIME COMPLEXITY: NUMBER OF EPOCHS AND TOTAL TRAINING TIME

When focusing on the number of epochs, in cases when early stopping criteria was applied, the proposed *DEFT* method consumed a lot more epochs due to its iterative nature. For the *DEFT* method, such behavior was expected. On the other hand, we expected other methods to use more epochs before stopping, but it turned out that the training stopped after no more than 10 epochs on average (see Table 5). When the full 500 epochs were used for training, the three compared methods' classification performance improved but were still unable to outperform our proposed *DEFT* method.

The overall time spent on training was highly correlated with the number of used epochs, although it turned out that there were some differences. Interestingly, although trained for the same amount of epochs, the *baseline* method seems to consume the least amount of time, being followed by the *pretrained* and *conventional* methods, while our proposed *DEFT* method used the highest amount of time. This could be attributed to the fact that the DE optimization algorithm's inner workings consume a significant amount of time.

The analysis of the total spent training time is presented in Fig. 13. Observing it, we can easily see that the time consumption does not vary greatly on a per fold basis, which is encouraging and proves that the layer selection mechanism of our *DEFT* method is capable of finding the most optimal solution based on the given constraints. Additionally, it also

TABLE 7. Average ranks (the best result is shown in bold) and statistical comparison (all differences are significant) of the four methods trained with early stopping.

	average rank				Friedman	Wilcoxon test		
	conventional	pretrained	baseline	DEFT	test	v. conventional	v. pretrained	v. baseline
acc	1.7	2.0	2.3	**3.9**	0.0006	0.0051	0.0051	0.0069
AUNP	1.7	2.1	2.2	**3.9**	0.0007	0.0051	0.0051	0.0069
F1	1.7	2.1	2.2	**3.9**	0.0007	0.0051	0.0051	0.0069
kappa	1.7	2.1	2.2	**3.9**	0.0007	0.0051	0.0051	0.0069
epochs	1.7	1.7	**1.5**	2.8	0.0074	0.0069	0.0166	0.0166
time	2.9	1.6	**1.5**	4.0	0.0000	0.0051	0.0051	0.0051

TABLE 8. Average ranks (the best result is shown in bold) and statistical comparison (only time and epochs have significant differences) of the four methods trained for full 500 epochs.

	average rank				Friedman	Wilcoxon test		
	conventional	pretrained	baseline	DEFT	test	v. conventional	v. pretrained	v. baseline
acc	2.2	2.5	2.5	**2.8**	0.7766	–	–	–
AUNP	2.2	2.5	2.5	**2.8**	0.7766	–	–	–
F1	2.2	2.5	2.5	**2.8**	0.7766	–	–	–
kappa	2.2	2.5	2.5	**2.8**	0.7766	–	–	–
epochs	2.9	2.9	2.9	**1.3**	0.0002	0.0069	0.0069	0.0069
time	3.0	2.0	**1.0**	4.0	0.0000	0.0051	0.0051	0.0051

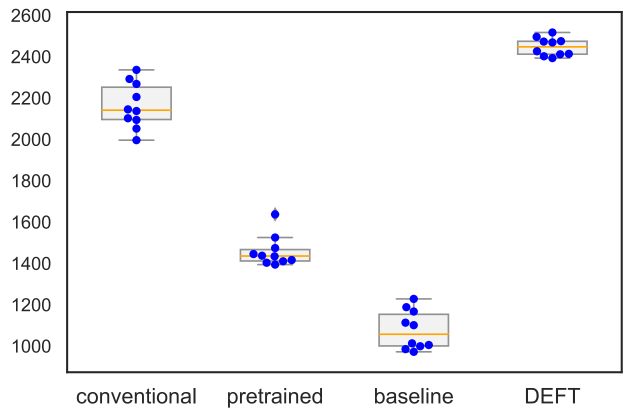


FIGURE 13. Time spent training a full 500 epochs; each dot represents one fold.

gives us a solid basis to further investigate how to make the selection mechanism even more effective in terms of consumed epochs and time.

3) STATISTICAL COMPARISON

To evaluate the statistical significance of these results, we first applied the Friedman test as suggested by Demšar [62] by calculating the asymptotic significance for the four compared methods on all 10 folds (using α=0.05). As the results are not normally distributed, the Friedman test was applied, which is a non-parametric statistical test used to detect differences in the results of various methods across multiple test attempts.

When early stopping was used, the results of the performed Friedman test, with regard to all six metrics, show that differences between the four methods are statistically significant (Table 7). In the case of using the full 500 epochs for training, the Friedman test results show significant differences for time and number of epochs (Table 8).

TABLE 9. Accuracy performance comparison with similar studies.

	CNN	Accuracy
Mishra [63] [a]	LeNet based	0.84
Mishra extended [64] [a]	Mishra [63] based	0.92
Arunachalam [54] [a, b]	LeNet based	0.93
DEFT	VGG19	0.8657

[a] Using smaller patches instead of tiles.
[b] Using expert-guided generation of features.

To further compare the results of our *DEFT* method with the remaining three methods, the Wilcoxon signed-rank test was applied next (using α=0.05), as suggested by Demšar [62]. If the Wilcoxon test resulted in a statistically significant difference between the two methods, the method with a better average rank could be regarded as the better method. Our proposed *DEFT* method achieved the highest average rank (which is the best) among the four methods for accuracy, AUNP, F1, and kappa, both when using early stopping or not (see also Figs. 5–12). In the case of using early stopping, our *DEFT* method performed the worst in both the time and number of epochs spent on training. When the classifiers were trained for the full 500 epochs, however, *DEFT* achieved the lowest amount of epochs, but still performed a bit slower than the remaining three methods (see also Fig. 13).

4) A COMPARISON WITH OTHER SIMILAR STUDIES

The problem of identifying osteosarcoma has already been addressed in previous studies using the same presented dataset. However, the authors of those studies split the original images into smaller patches and performed a classification over those patches taking into account additional contextual information about the tumor. Even though the reported results are quite similar to ours, as presented in Table 9. In [63], the authors proposed a CNN architecture with

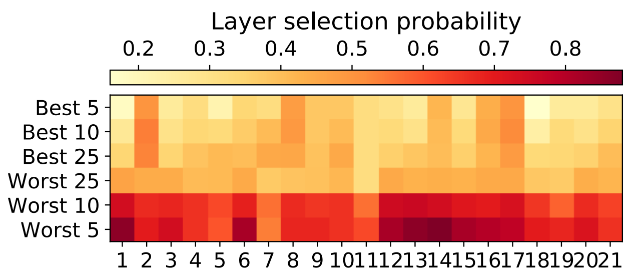


FIGURE 14. Layer selection analysis, based on a *DEFT* method results. The y-axis denotes the average selections from the best 5 performing individual candidates to the worst 5 performing ones. On the x-axis there are numbers denoting the layers of the VGG19 architecture.

7 layers, 3 of them being convolutional layers, 3 of them being maximization pooling layers, and two being fully connected layers. In terms of accuracy, our proposed method outperforms the mentioned one by a margin of 2.57%. In [64] and [54], the reported accuracy was somewhat higher than it is in our case (by 5.83% and 4.63%, respectively). However, we must also consider that the authors in [54] are using mechanisms for the expert-guided generation of features, while on the other side, our proposed *DEFT* method does not utilize any domain expert knowledge in the process of building the predictive model.

B. LAYER SELECTION ANALYSIS

In this section, we analyze the *DEFT* method's selection of fine-tunable layers and the performance of such various solutions produced by the method. As presented in previous sections, there is no general rule or recipe to follow when selecting which layers of CNN to fine-tune and which ones to leave frozen. So we dove deeper into the performance of various layer combinations, conducted by our method in the process of finding the most optimal solution under the given constraints.

Fig. 14 shows a heat-map of layer selection probabilities averaged from the best five found solutions to the worst five found solutions by our *DEFT* method. The rationale behind grouping the individual based candidate models is that, interestingly, the best performing individuals are quite different in terms of selected layers but still deliver similar classification performance. As can be seen from the figure, the most obvious thing is that the better performing individuals in general have a lower selection probability rate than the worse performing individual-based candidate models. In other words, the CNN architectures with fewer layers enabled for fine-tuning are achieving better performance results than the ones with more layers. This effect is also consistent with the low classification performance achieved by the *pretrained* method, where all convolutional layers are being trained. An interesting observation is also the occurrence of higher selection probabilities for layers 2, 8, 14, 16, and 17 when looking at the best performing individuals, which could be explained by the fact that the features extracted on those layers have a more significant impact on final prediction than the others. We can also generally observe relatively low selection probabilities for layers towards the end, which diminishes the assumption that early or middle layer features should be shared, as has already been noted in previous sections.

C. THREATS TO VALIDITY

Commonly, in the machine learning field, the validity threats often relate to the diversity, quality and quantity of the data. Since all supervised machine learning methods, techniques, and approaches rely on how the given data is labeled or pre-classified, for our research we picked the dataset that was collected by clinical scientists and labeled by two medical experts to minimize potential threats to validity. Nevertheless, our obtained results and findings may not be generalized to all specific situations.

Splitting the data into a training and test set could also be a potential a threat to validity. To reduce the possibility of such a threat, we adopted a well-known 10-fold cross-validation procedure.

Due to the stochastic nature of our proposed *DEFT* method, in order to reduce the internal threat to validity, the experiment conducted with the *DEFT* method was executed in 10 runs, and the reported performance metrics were the averages of those runs.

VII. CONCLUSION

In this work, we presented a novel *DEFT* adaptive method for transfer learning with fine-tuning, featuring the layer selection mechanism based on the DE algorithm. The method addresses the problem of selecting which layers of given CNN architecture to fine-tune and which ones to leave frozen, in order to achieve the best possible classification performance. The exploited DE algorithm enables the *DEFT* method to find the most optimal layer selection solution given the constraints and available dataset in an automatic, straightforward, adaptive manner.

The presented method was evaluated against three other methods, one where the CNN is trained conventionally (from scratch), one where the CNN was trained using the conventional approach with convolutional layers being pre-trained on the ImageNet dataset, and one where the transfer learning was utilized and the fine-tunable layers were handpicked based on general recommendations and our previous experiences. For our experiments' target task, we selected the image classification task of identifying the osteosarcoma from H&E stained osteosarcoma images. The results obtained from the conducted experiments show that our proposed *DEFT* method outperformed the compared methods in all predictive performance metrics. On the other hand, the proposed method is significantly more time-consuming, utilizing an iterative optimization mechanism.

The *DEFT* method could be easily utilized with any kind of task, where a standard fine-tuning methodology is applicable. Furthermore, the proposed *DEFT* method could also be utilized with any other CNN architecture regardless of the number of convolutional layers, with respect to adjusting method parameters such as the dimension of the problem and the number of function evaluations. Generally, the latter one should be increased when utilizing deeper CNN architectures since such architectures feature a larger number of layers, which translates to a larger search space and increased time complexity in order to find the most suitable combination of layers selected for fine-tuning. Similar to the utilization of more conventional methods, when applying our proposed *DEFT* method against various datasets with a different number of samples, one should also revise and appropriately adapt the classifier layers as well as initial training parameters to achieve the best possible outcome.

In the future, we would like to extend our research to the utilization of various optimization algorithms such as the Firefly algorithm or the Particle Swarm Optimization. We would also like to apply the proposed *DEFT* method on different medical imaging datasets, and possibly also to other image classification tasks. To analyse how the method performs regardless of the classification domain, it would have been useful to test it on several datasets on multiple tasks, including those from other domains. Additionally, we would like to explore the possibilities of how to make the layers selection mechanism more efficient, which would enable the *DEFT* method to deliver better classification performance with lower time complexity, closer to the one where conventional training is utilized. Finally, we would also like to investigate the possibilities of combining our *DEFT* method with active learning approaches.

REFERENCES

[1] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agricult.*, vol. 147, pp. 70–90, Apr. 2018.

[2] 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," *Seismol. Res. Lett.*, vol. 90, no. 1, pp. 3–14, Jan. 2019.

[3] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (formerly BIONETICS)*, 2016, pp. 21–26.

[4] J. Flisar and V. Podgorelec, "Identification of self-admitted technical debt using enhanced feature selection based on word embedding," *IEEE Access*, vol. 7, pp. 106475–106494, 2019.

[5] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Mol. Syst. Biol.*, vol. 12, no. 7, p. 878, Jul. 2016.

[6] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges," *Briefings Bioinf.*, vol. 19, no. 6, pp. 1236–1246, Nov. 2018.

[7] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, and M. M. Hoffman, "Opportunities and obstacles for deep learning in biology and medicine," *J. Roy. Soc. Interface*, vol. 15, no. 141, 2018, Art. no. 20170387.

[8] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.

[9] M. I. Razzak, S. Naz, and A. Zaib, "Deep learning for medical image processing: Overview, challenges and the future," in *Classification in BioApps*. Cham, Switzerland: Springer, 2018, pp. 323–350.

[10] G. Vrbančič, M. Zorman, and V. Podgorelec, "Transfer learning tuning utilizing grey wolf optimizer for identification of brain hemorrhage from head ct images," in *Proc. StuCoSReC: 6th Student Comput. Sci. Res. Conf.*, 2019, pp. 61–66.

[11] B. Q. Huynh, H. Li, and M. L. Giger, "Digital mammographic tumor classification using transfer learning from deep convolutional neural networks," *J. Med. Imag.*, vol. 3, no. 3, Aug. 2016, Art. no. 034501.

[12] J. J. Gómez-Valverde, A. Antón, G. Fatti, B. Liefers, A. Herranz, A. Santos, C. I. Sánchez, and M. J. Ledesma-Carbayo, "Automatic glaucoma classification using color fundus images based on convolutional neural networks and transfer learning," *Biomed. Opt. Express*, vol. 10, no. 2, pp. 892–913, 2019.

[13] R. Mehra, "Breast cancer histology images classification: Training from scratch or transfer learning?" *ICT Express*, vol. 4, no. 4, pp. 247–254, 2018.

[14] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, and F. Yan, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[15] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[16] G. Vrbancic, I. J. Fister, and V. Podgorelec, "Parameter setting for deep neural networks using swarm intelligence on phishing Websites classification," *Int. J. Artif. Intell. Tools*, vol. 28, no. 6, Oct. 2019, Art. no. 1960008.

[17] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic ConvNet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1790–1802, Sep. 2016.

[18] 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 Trans. Med. Imag.*, vol. 35, no. 5, pp. 1299–1312, May 2016.

[19] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "SpotTune: Transfer learning through adaptive fine-tuning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4805–4814.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[21] L. Zhu, S. O. Arik, Y. Yang, and T. Pfister, "Learning to transfer learn: Reinforcement learning-based selection for adaptive transfer learning," arXiv, New York, NY, USA, Tech. Rep., 2020.

[22] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4109–4118.

[23] W. Ge and Y. Yu, "Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1086–1095.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[25] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 97–105.

[26] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features Off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 806–813.

[27] Z. Shi, H. Hao, M. Zhao, Y. Feng, L. He, Y. Wang, and K. Suzuki, "A deep CNN based transfer learning method for false positive reduction," *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 1017–1033, Jan. 2019.

[28] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 550–558.

[29] X. J. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2005.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[31] L. I. Kuncheva and J. J. Rodriguez, "Classifier ensembles with a random linear oracle," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 4, pp. 500–508, Apr. 2007.

[32] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[33] 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 Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 7, pp. 641–651, Jul. 1995.

[34] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, ''Inductive learning algorithms and representations for text categorization,'' in *Proc. 7th Int. Conf. Inf. Knowl. Manage. (CIKM)*, 1998, pp. 148–152.

[35] Q. Yang, C. Ling, X. Chai, and R. Pan, ''Test-cost sensitive classification on data with missing values,'' *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 5, pp. 626–638, May 2006.

[36] X. Zhu and X. Wu, ''Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering,'' *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1435–1440, Oct. 2006.

[37] M. Hussain, J. J. Bird, and D. R. Faria, ''A study on CNN transfer learning for image classification,'' in *Proc. UK Workshop Comput. Intell.* Cham, Switzerland: Springer, 2018, pp. 191–202.

[38] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, ''Towards better exploiting convolutional neural networks for remote sensing scene classification,'' *Pattern Recognit.*, vol. 61, pp. 539–556, Jan. 2017.

[39] G. E. Hinton, ''Reducing the dimensionality of data with neural networks,'' *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[40] H. Chougrad, H. Zouaki, and O. Alheyane, ''Deep convolutional neural networks for breast cancer screening,'' *Comput. Methods Programs Biomed.*, vol. 157, pp. 19–30, Apr. 2018.

[41] R. Storn and K. Price, ''Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces,'' *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[42] B. Nakisa, M. N. Rastgoo, A. Rakotonirainy, F. Maire, and V. Chandran, ''Long short term memory hyperparameter optimization for a neural network based emotion recognition framework,'' *IEEE Access*, vol. 6, pp. 49325–49338, 2018.

[43] N. Xue, I. Triguero, G. P. Figueredo, and D. Landa-Silva, ''Evolving deep CNN-LSTMs for inventory time series prediction,'' in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1517–1524.

[44] L. Brezočnik, I. Fister, and G. Vrbančič, ''Applying differential evolution with threshold mechanism for feature selection on a phishing Websites classification,'' in *New Trends in Databases and Information Systems*, T. Welzer, J. Eder, V. Podgorelec, R. Wrembel, M. Ivanović, J. Gamper, M. Morzy, T. Tzouramanis, J. Darmont, and A. K. Latifić, Eds. Cham, Switzerland: Springer, 2019, pp. 11–18.

[45] A. P. Piotrowski, ''Review of differential evolution population size,'' *Swarm Evol. Comput.*, vol. 32, pp. 1–24, Feb. 2017.

[46] S. Das and P. N. Suganthan, ''Differential evolution: A survey of the State-of-the-Art,'' *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[47] D. Fister, I. Fister, T. Jagric, I. Fister, and J. Brest, ''A novel self-adaptive differential evolution for feature selection using threshold mechanism,'' in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 17–24.

[48] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[49] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[50] (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

[51] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister, Jr., ''NiaPy: Python microframework for building nature-inspired algorithms,'' *J. Open Source Softw.*, vol. 3, no. 23, p. 613, Mar. 2018, doi: 10.21105/joss.00613.

[52] S. Haghighi, M. Jasemi, S. Hessabi, and A. Zolanvari, ''PyCM: Multiclass confusion matrix library in Python,'' *J. Open Source Softw.*, vol. 3, no. 25, p. 729, May 2018, doi: 10.21105/joss.00729.

[53] P. Leavey, A. Sengupta, D. Rakheja, O. Daescu, H. B. Arunachalam, and R. Mishra, ''Osteosarcoma data from ut southwestern/UT Dallas for viable and necrotic tumor assessment [data set],'' Cancer Imag. Arch., Fayetteville, AR, USA, Tech. Rep., 2019.

[54] H. B. Arunachalam, R. Mishra, O. Daescu, K. Cederberg, D. Rakheja, A. Sengupta, D. Leonard, R. Hallac, and P. Leavey, ''Viable and necrotic tumor assessment from whole slide images of osteosarcoma using machine-learning and deep-learning models,'' *PLoS ONE*, vol. 14, no. 4, Apr. 2019, Art. no. e0210706.

[55] K. Simonyan and A. Zisserman, ''Very deep convolutional networks for large-scale image recognition,'' 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[56] S.-J. Kim, C. Wang, B. Zhao, H. Im, J. Min, H. J. Choi, J. Tadros, N. R. Choi, C. M. Castro, R. Weissleder, H. Lee, and K. Lee, ''Deep transfer learning-based hologram classification for molecular diagnostics,'' *Sci. Rep.*, vol. 8, no. 1, pp. 1–12, Dec. 2018.

[57] 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

[58] 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

[59] C. Ferri, J. Hernández-Orallo, and R. Modroiu, ''An experimental comparison of performance measures for classification,'' *Pattern Recognit. Lett.*, vol. 30, no. 1, pp. 27–38, 2009.

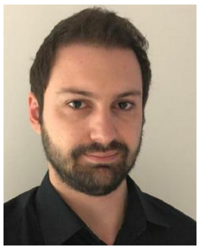
[60] T. Fawcett, ''Using rule sets to maximize ROC performance,'' in *Proc. IEEE Int. Conf. Data Mining*, Nov./Dec. 2001, pp. 131–138.

[61] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical Methods for Rates and Proportions*, 3rd ed. Hoboken, NJ, USA: Wiley, 2003.

[62] J. Demsar, ''Statistical comparisons of classifiers over multiple data sets,'' *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[63] R. Mishra, O. Daescu, P. Leavey, D. Rakheja, and A. Sengupta, ''Histopathological diagnosis for viable and non-viable tumor prediction for osteosarcoma using convolutional neural network,'' in *Proc. Int. Symp. Bioinf. Res. Appl.* Springer, 2017, pp. 12–23.

[64] R. Mishra, O. Daescu, P. Leavey, and D. Rakheja, ''Convolutional neural network for histopathological analysis of osteosarcoma,'' *J. Comput. Biol.*, vol. 25, no. 3, pp. 313–325, 2018.



GREGA VRBANČIČ (Graduate Student Member, IEEE) received the B.Sc. and M.Eng. degrees in informatics and communication technologies from the University of Maribor, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in computer science. He is also a Young Researcher with the Faculty of Electrical Engineering and Computer Science, University of Maribor. He is the author of three peer-reviewed scientific journal articles and eight 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.

VILI PODGORELEC (Member, IEEE) received the Ph.D. degree from the University of Maribor, Slovenia, in 2001.

He worked as a Visiting Professor and/or a Researcher at several universities around the world, including the University of Osaka, Japan; the Federal University of Sao Paulo, Brazil; the University of Nantes, France; the University of La Laguna, Spain; the University of Madeira, Portugal; the University of Applied Sciences Seinäjoki, Finland; and the University of Applied Sciences Valencia, Spain. He is currently a Professor of computer science with the University of Maribor. He has been involved in AI and intelligent systems for 20 years, where he gained professional experience in the implementation of many scientific and industrial research and development projects related to the analysis, design, implementation, integration, and evaluation of intelligent information systems. He has authored more than 50 peer-reviewed scientific journal articles, more than 100 conference papers, three books, and several book chapters on machine learning, computational intelligence, data science, medical informatics, and software engineering. He has top expertise in AI and machine learning methods and algorithms in the field of transparent data-driven decision making (especially in medicine), in-depth knowledge of system integration technologies and methods applied to intelligent data analysis, classification and prediction of human-centered data, as well as large experience in designing and implementing information retrieval, natural language processing and text mining solutions for academia, industrial partners and international companies using state-of-the-art approaches, and methods and tools. He received several international awards and grants for his research activities.