Applied Soft Computing 132 (2023) 109872

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

CJT-DEO: Condorcet's Jury Theorem and Differential Evolution Optimization based ensemble of deep neural networks for pulmonary and Colorectal cancer classification

Gaurav Srivastava, Aninditaa Chauhan, Nitesh Pradhan *

Department of Computer Science and Engineering, Manipal University Jaipur, 303007, Rajasthan, India

ARTICLE INFO

Article history:
Received 6 July 2022
Received in revised form 26 October 2022
Accepted 22 November 2022
Available online 26 November 2022

Keywords:
Lung cancer
Colorectal cancer
Histopathological images
Bio-medical imaging
Deep feature extraction
Deep transfer learning
Ensemble learning
Condorcet's Jury Theorem
Metaheuristic optimization

ABSTRACT

Cancer is one of the most dangerous diseases globally, causing adverse effects on human life, with early detection and treatment planning being crucial for patients. Amongst different malignancies, Lung and Colorectal cancer cause the first and second most cancer deaths in the world, respectively. In this study, the authors aim to analyze LC25000 histopathological image dataset for lung and colon cancer detection. The fundamental goal of the proposed research is to leverage the ensemble learning approach to improve the classification performance of deep learning models. Many previous studies have proposed several ensemble methods and weighting schemes. However, none of them optimized the assigned weights using a meta-heuristic-based approach as per our best knowledge. The authors have applied Differential Evolution optimization to optimize and find the optimal assigned weights to the classifiers while training the ensemble model. In addition, a novel approach to ensemble base learners with majority voting based on Condorcet's Jury Theorem has also been proposed. This proposed method has been shown to save a lot of computational efforts by eliminating the training procedure of meta-learners. Besides this, the authors also demonstrated that Condorcet's Jury Theorem holds while ensembling the N number of classifiers in Neural Networks. Our proposed method and experimental results outperformed compared to the state-of-the-art with the optimized ensemble model showing an accuracy of 99.78% and Condorcet's Jury Theorem-based ensemble model 99.88% on 5-class classification.

1. Introduction

Cancer is a life-threatening disease affecting millions of people yearly and is the second leading cause of death worldwide [1,2]. In 2020 there were 1.3 million new cases reported in India alone, causing around 0.8 million deaths [3]. The most common causes of cancer death in 2020 were Lung cancer(1.8 million) and colorectal cancer(0.9 million) [2]. Lung and colon cancer combined account for over 20% of the total cancer cases, with studies indicating that they may develop synchronously [4,5]. According to Kurishima et al. 17 out of 3102 lung cancer patients were diagnosed with colon cancer within just one month [6]. While metastases from lung cancer to the colon are rare, it is much more common for colon cancer to spread to the lungs, with the lungs being the second most common site for metastases of colorectal cancer [6,7].

These mind-numbing statistics show the immense necessity for an efficient diagnostic method to detect these cancer cells, which will aid in early detection, proper treatment planning for the patient's health, and assist in studying preventive measures for this disease. The use of artificial intelligence for analyzing biomedical images to detect various diseases has shown great potential, with performance comparable to and, in some cases, better than that of medical professionals [8,9]. This has increased the need to utilize machine learning (ML) and deep learning (DL) techniques to detect cancerous cells using histopathology images. Computer Aided Diagnosis of the disease also eases the testing burden on doctors and hospitals and assists patients with efficient and reliable detection [10,11].

Recent studies on cancer detection using artificial intelligence use deep learning techniques such as transfer learning, that is, pre-trained models or previous state-of-the-art models for feature extraction on their particular problem dataset [12]. This, along with popular image processing techniques, has achieved substantially better results than traditional machine learning algorithms [12,13]. However, applying these deep learning methods requires enormous amounts of reliable and labeled data that is hard to get in the medical field [14]. To overcome this issue, the dataset used for this work [15] is a histopathological image dataset that consists of 5 classes of lung and colon cancer: 25,000

* Corresponding author.
E-mail addresses: mailto.gaurav2001@gmail.com (G. Srivastava), pradhan.nitesh943@gmail.com (N. Pradhan).

https://doi.org/10.1016/j.asoc.2022.109872
1568-4946/

images equally balanced into each class with 5000 images in each. The availability of such an abundant and evenly balanced multiclass dataset for biomedical imaging is scarce, and this fueled our motivation to implement deep learning techniques on this dataset.

Image Processing techniques are utilized to enhance the input images before extracting the features using Convolutional Neural Network (CNN) models. These techniques are used to denoise and improve the overall quality of input image data, making it easier for the neural network to analyze the images and drastically improving classifier accuracy. Several Image processing techniques can be found in the literature, such as the Fast Non-Local Means (NLM) Denoising algorithm, Contrast Limited Adaptive Histogram Equalization (CLAHE), Balance Contrast Enhancement Technique (BCET), and OTSU's Thresholding for Image Binarization. Applying these techniques highlights the features in an image, making it easier for the model to extract those.

Ensembling learning approaches are commonly used in deep learning problems to combine the powers of different classifiers to train an ensemble model that should perform better than all of them. Unfortunately, ensembling the base classifiers gives equal weightage to all of them. There are times when we want extremely competent models to contribute more to an ensemble prediction, and other times when we want less competent models to contribute less to an ensemble prediction. So, penalizing a less accurate model and rewarding a more precise model can lead to even better predictions.

Several methods are used to assign weights to the base classifier according to their predicted performance reliability. Searching for optimum weights is an extensive procedure if we perform a random search or use a Grid Search technique. Instead, we can use a heuristic-based approach to search for optimum weights. Researchers used many meta-heuristic-based optimization algorithms in the past for different purposes. The authors have used a metaheuristic search to find the best weights when ensembling as a result of their analysis. Several meta-heuristic-based algorithms are accessible, including coral reefs [16], particle swarm optimization [17], and evolutionary algorithms [18]. Since evolutionary algorithms are global optimization techniques that are scalable to higher dimensional problems, the authors have employed evolutionary algorithm-based optimization [19]. They can handle evaluation functions that do not produce a good result in a specific time and are resilient to noisy evaluation processes [20]. Moreover, the algorithms are easily adaptable to new problems. The algorithm can almost always be modified and tailored. Nevertheless, many studies have been done to determine which evolutionary method is appropriate for a specific task.

Many algorithms may combine different classifiers to create an ensemble model, with the majority vote being the most straightforward. Despite its simplicity, it has been suggested that the majority vote is the optimal technique if the mistakes among the classifiers are not connected. The mathematical concept of Condorcet's Jury Theorem implies that the majority vote taken by independent members in a group is bound to produce better results than just one individual in the group and this result improves as the number of voters increases. This theorem states that if a majority of independent members in a group, individually, can make the correct decision rather than making a random choice, they are better at decision-making than just one group member. This theorem in applications with Neural Networks helps ensemble the output of multiple trained deep learning models with good outcomes to give results better than any individual models. To the best of the author's knowledge, the jury theorem is being validated and implemented on neural networks for the first time in this research. The proposed method uses the theorem to ensemble the outputs of multiple Deep CNN classifiers to produce results better than any individual model. Furthermore, the Jury theorem-based proposed method is computationally much more efficient than traditional ensembling approaches as it is not required to be trained again at the end to initialize the models' weights.

The primary contributions of our work can be summarized as follows:

1. Various Image preprocessing techniques such as NLM Denoising, CLAHE, BCET, and Otsu's thresholding are implemented on the LC25000 Lung and Colon histopathological dataset to enhance the input image quality. On this substantially improved input data, features were extracted using Deep CNN pre-trained models, with the model performance analyzed after implementation.
2. Ensemble learning methods are implemented on six DCNN classifiers from different architectural families as base learners. In addition, the deep stacking Ensemble approach is proposed to combine the features of the different classifiers to produce higher accuracy than any individual models.
3. The authors proposed a Metaheuristic optimization based approach — Differential Evolution for optimizing the weights assigned to the meta learner on ensembling the top DCNN models. This method shall drastically enhance performance as weights are optimized instead of assigned randomly.
4. Finally, the authors proposed and validated the mathematical concept of Condorcet's Jury theorem in the field of study of neural networks. The proposed Jury based approach uses the 'N' individual classifiers' score to determine the final score. Compared to ensembling methods, this method is computationally efficient and has produced the best classifier accuracy.

The remaining contents of the study can be summarized as follows. Section 2 is dedicated to analyzing the previous work of various researchers in detecting Lung and Colorectal Cancer and the Ensembling approaches they have employed. Section 3 deals with the various Materials and Methods used and entails proposed algorithms and methods in this study. Finally, Section 4 discloses and covers experimentation and the results of the aforementioned experiments.

2. Related works

Various studies have been conducted to assist with the rapid detection and diagnosis of cancer in patients. In this section, the authors reviewed and analyzed previous work on detecting lung and colon cancer using deep learning techniques. In particular, the authors aim to understand and compare the work done on the dataset chosen for this research. The research studies through [21,22] are conducted on the LC25000 lung and colon dataset for histopathological images. Although the dataset is relatively very recent, researchers have done some substantial work. These findings after review work are presented below.

Masud et al. [21] employed a classification method for the 5 classes of lung and colon cancer in the histopathological image dataset, achieving a maximum accuracy of 96.33%. For this, 4 sets of features were extracted from two image processing algorithms. These were combined to form a new set of features for image classification. Whereas in Mangal et al. [23], a shallow neural network architecture was proposed for the classification of histopathological images of the lung (3 classes) and colon (2 classes) with an accuracy of more than 97% and 96%, respectively. The basic CNN architecture was used, which consisted of the Input, Convolutional, Pooling, Flatten, Dense, and Dropout layers. A split of 80-10-10 was taken into account with future work in utilizing different architectures with hyperparameter

optimization considered. In Kumar et al. [24], the authors aim to compare and analyze two feature extraction approaches as extraction from handcrafted features and extraction from deep neural networks. Popular classifiers such as Gradient boosting, Multilayer Perceptron, Support Vector Machine (SVM) with RBF Kernel, and Random forest (RF) were used to classify lung and colon cancer in both the approaches. A significant improvement of approximately 5% was observed in accuracy and other performance metrics in the case of utilizing deep CNN models compared to the handcrafted features. DenseNet-121 with RF classifier was observed to show the best performance of the experimented deep learning models, with an accuracy of 98.6%. The substantial work done by Mesut Togacar [25] used the model DarkNet-19 for training the image classes of the lung cancer and colon cancer dataset. The feature set extracted was used to select inefficient features from the set by using meta-heuristic Manta-Ray Foraging and Equilibrium optimization algorithms. The efficient feature sets were obtained by separating the inefficient sets from the feature set, using the Complementary rule. SVM was used to combine and classify these features with an overall accuracy of 99.69% obtained on a dataset split of 70–30 for training and testing respectively. The proposed method can be used for building specific approaches on different datasets in the future. The limited yet beneficial work done in 5-class classification using histopathological images was described briefly.

Talukder et al. [26] proposed a hybrid ensemble deep feature extraction model to detect lung and colon cancer from histopathological images efficiently. Their proposed methods achieved a 99.05% accuracy in classifying lung images, 100% accuracy in colon images, and overall 99.30% accuracy in 5-class classification of both lung and colon images. Li et al. [27] proposed a novel method named embedded fusion mutual learning (EFML) for pathological image classification. The authors jointly supervised model training by combining the logits output and the fused feature maps. The evaluation has been performed on three datasets: BreaKHis, BACH, and LC25000. On LC25000, the proposed method achieved an overall accuracy of 98.50% on 5-class classification. Lin et al. [28] developed an extremely light plug-and-play module called Pyramidal Deep-Broad Learning (PDBL) to boost classification performance without the hassle of retraining for any well-trained classification backbone. The proposed module achieved a 96.49% accuracy with the ResNet50 backbone on the LC25000 dataset. Fan et al. [29] compared the ability to categorize histopathological cancer images in binary breast cancer datasets and multiclass lung and colon cancer datasets using the traditional softmax classifier and the SVM classifier-based transfer learning technique. An approach that ties the SVM classifier to the fully connected (FC) layer of the softmax-based transfer learning model is proposed to improve classification accuracy. The proposed method calls for training the newly added FC layer using the softmax-based model on the target dataset in the first stage and then training the SVM classifier using the newly trained FC layer in the second step. The overall accuracy achieved was 99.44% on the LC25000 dataset. Mehmood et al. [30] developed an alternative to the present cancer detection techniques, a computationally efficient model for the rapid and accurate diagnosis of lung and colon cancers. The application of the proposed approach increased overall accuracy from 89% to 98.4% and demonstrated computing efficiency.

Furthermore, the researchers have also worked on 2 and 3-class classification using different networks to test the classifier performance. In [31], the authors proposed a CNN model named MA_ColonNET for detecting colon cancer from image data. The 45-layer model consisted of the usual Convolutional, Batch normalization, and Relu layers, with a softmax classifier and SGD optimization method. The classification accuracy achieved was 99.75% on 10,000 images with an 80–20 training and testing split. In Nishio et al. [32] the authors aim to build a CAD system to classify different types of lung cancers using histopathological images, and for this, two datasets were considered. The private dataset consists of 94 images in 5 classes, and the public dataset consists of 15,000 images in 3 classes. Machine learning algorithms were used to classify the image features obtained from 2 feature extraction techniques: conventional texture analysis (TA) and homology-based image processing (HI). In both the datasets, it was observed that the models with HI were more beneficial for the CAD systems than TA. Adu et al. [22] proposed Dual horizontal squash CapsNet. This method uses a new squash function, HSquash, which effectively squashes all vectors, long and small. Encoder feature fusion (EFF) is used to obtain richer features from complex information to improve the classification. Sigmoid activation function was used to get better normalization. The results were analyzed on the LC25000 dataset for lung and colon images showing an accuracy of 99.23%. This was compared with results obtained on traditional CapsNet, showing an improvement of over 14%. The transfer learning application on the proposed DHS-CapsNet will be analyzed for future work. In [33], 8 pre-trained CNN models and image augmentation were used for training on the LC25000 dataset. Model performance was assessed on lung and colon subtypes (binary classification) with 97%–100% accuracy. GradCAM and SmoothGrad were utilized to picture class activation and saliency maps to improve the classification.

The authors also include two other works from different datasets for metastasis detection using deep learning techniques for studying the different approaches used. In Paik et al. [34], the authors developed a CAD algorithm called Surface normal overlap, which was applied to lung nodule and colonic polyps detection in helical CT images. The method's theoretical aspects were illustrated using a statistical shape model. Its performance was optimized using CT simulations and assessed using per-lesion cross-validation on 8 CT chest and 8 CT colonography datasets. The results show that the algorithm proposed achieved 100% sensitivity for colonic polyps at 7.0 false positives and 90% sensitivity for solid nodules at 5.6 false positives/dataset. Khan et al. [35] explore clustering analysis using an extension of fuzzy k-means for feature subspace generation, which was used to form the input clusterings for ensembling. The proposed algorithm extended the k-means fuzzy algorithm by two steps. The first step was to introduce a penalty term for making the algorithm insensitive to initializing cluster centroids. The second step was to automate the clustering process for updating feature weights, which addresses the noise values in the dataset. Experiments were conducted to show that the proposed algorithm outperformed popular clustering algorithms. The authors aim to test their method for cluster analysis on real-world applications and other types of data in the future.

The comparative analysis of existing techniques can be seen in Table 1.

Even though various researchers have worked on this problem domain, and the work done so far is beneficial, with different deep learning methods being used and proposed, it still has many scopes left to work. Also, in the medical domain, the higher the accuracy, the more real-world reliability of the proposed solution. For example, suppose if a model gives an accuracy of 99%, it will only misclassify a single image out of 100 tested images. But if tested on a large scale of 1M million images, it will misclassify 10 000 images. So, the authors tried to improve the accuracy further in this research. This was our primary motivation to conduct this research. The work by Togacar [25], in particular, is taken into account by the authors to compare the classification performance as it had the highest 5-class accuracy observed so far.

Table 1
Comparative analysis of existing techniques.

Year	Author(s)	Techniques used	Class accuracy	Advantages	Findings
2021	Masud et al. [21]	Image classification	5-class - 96.33%	Techniques used for feature extraction led to good results.	Experimentation on various deep learning models would improve the performance.
2021	Adu et al. [22]	Capsule neural network	5-class - 99.23%	Horizontal squash function used on CapsNet along with encoder feature fusion, a novel approach.	Transfer learning can be applied to increase the usability of the model proposed.
2021	Mangal et al. [23]	Shallow neural networks	3-class- >97% 2-class- >96%	Use of simple CNN architecture for achieving reasonable performance	More approaches can be analyzed to further enhance the model.
2022	Kumar et al. [24]	Feature extraction techniques, transfer learning	5-classes - 98.6%	Handcrafted feature extraction techniques have been analyzed and their performances compared.	Feature ranking techniques and stain normalization is to be used in future.
2021	Mesut Togacar [25]	Metaheuristic Optimization techniques	**5-class - 99.69%**	Great jump in accuracy observed after training on DarkNet-19 with optimization techniques used.	The techniques used have to be validated using other datasets, or specifically devised for them.
2021	Yildirim et al. [31]	CNN based model	2 class - 99.75%	Proposed a model for accurate detection of Colon cancer	The model architecture could be implemented using lung image classes as well.
2021	Nishio et al. [32]	Image feature extraction techniques	**3-class - 99.43%**	ImplementIon was done on 2 datasets of lung cancer for analyzing two feature extraction techniques.	The comparison and conclusion deduced on the techniques requires more sufficient proof.
2020	Garg et al. [33]	Pre Trained CNN models	5-class - 96%	One of the earlier works. Smooth Cam and GradCam were used to enhance images after training.	The techniques used are from pretrained models only.

3. Materials and methods

In this section, the authors illustrate the methods implemented and techniques used for the experimentation. The section starts with data-preprocessing techniques used in this study, followed by the background and theory of theorems and optimization methods used in this research, chosen training parameters, existing ensemble methods, and finally, the proposed ensemble methods.

3.1. Data preprocessing

Adjustments are made to the raw data before inputting it to the machine learning, or deep learning algorithm is called Image preprocessing. It is used to format images before the model uses them for training and inference purposes. This includes resizing, orienting, color corrections, etc. Sometimes Image preprocessing plays a vital role in boosting the classifier accuracy in deep learning. Training a convolutional neural network on raw images is likely to result in poor classification results. With the help of Image preprocessing, we can reduce unwanted distortions in an image and enhance some important features for our model to learn. Besides accuracy, Image preprocessing is also necessary to reduce model complexity and the computational cost of the entire training procedure. Four different image preprocessing techniques have been used in this research: NLM, CLAHE, BCET, and Otsu's thresholding.

3.1.1. Fast non-local means denoising algorithm

Image denoising is a fundamental challenge in image processing and computer vision to estimate the original image by suppressing noise from a noise-contaminated version of the image [36]. Image noise may be caused by various intrinsic and extrinsic factors that are difficult to prevent in real-world scenarios.

The non-local means (NLM) algorithm, widely used in image preprocessing, is one of the best image denoising algorithms because of its superior ability to retain image texture details [37]. But due to its nonlocality when searching for similar pixels, the algorithm's time complexity is exceptionally high. Due to this, the authors used a Fast Non-Local Means algorithm [38]. It substitutes window similarity with a modified multi-resolution-based technique that utilizes fewer comparisons rather than comparing all pixels [39]. As shown in Eq. (1), it weighs neighbor pixels with similar neighborhoods.

$$g(\mathbf{p}) = \frac{1}{Z}\sum_{\mathbf{q}\in I} B_W(\mathbf{p}-\mathbf{q}) G_\sigma\left(\mathcal{N}_\mathbf{p} - \mathcal{N}_\mathbf{q}\right) f(\mathbf{q}) \tag{1}$$

where, $B_W(\mathbf{p}-\mathbf{q})$ signifies to choose $\mathbf{q}$ only within W-box radius around $\mathbf{p}$ and $G_\sigma\left(\mathcal{N}_\mathbf{p} - \mathcal{N}_\mathbf{q}\right)$ represent similarity weight based on intensities of neighborhood around $\mathbf{p}$.

Eq. (2) represents the Gaussian weighting based on difference in image intensities.

$$G_\sigma(\mathbf{x}) = e^{-\left(\frac{\sum_i x_i^2}{2\sigma^2}\right)} \tag{2}$$

If W covers the whole image, each output pixel depends on input pixels at any location "non-local", as shown in Eq. (3).

$$B_W(\mathbf{x}) = \begin{cases} 1 & \text{if } \max\left(|x_1|, |x_2|\right) \le W \\ 0 & \text{else} \end{cases} \tag{3}$$

3.1.2. Contrast limited adaptive histogram equalization

After removing noise from the image, the next step is to balance the contrast. The raw dataset we get often contains washed-out images with unbalanced contrast. So, to equalize the histogram, we may stretch it to encompass the whole range. CLAHE is a contrast-over-amplitude equalization version of Adaptive Histogram Equalization (AHE). Instead of a whole image, it works using tiles or slabs, which are tiny parts of an image [40]. Each slab's contrast transform function was computed separately. For the histogram, the clipping rule is shown in Eq. (4).

$$Y = a(X-b)^2 + c \tag{4}$$

where X is the input and Y is output image. The coefficients a, b, c and s are derived from Eqs. (5)–(8) respectively.

$$a = \frac{(H-L)}{(h-l)(h+l-2b)} \tag{5}$$

$$b = \frac{h^2(E-L) - s(H-L) + l^2(H-E)}{2[h(E-L) - e(H-L) + l(H-E)]} \tag{6}$$

$$c = L - a(l - b)^2 \tag{7}$$

$$s = \frac{1}{N}\sum_{i=1}^{n} x_i^2 \tag{8}$$

where, H' is the maximum value of Y, L is the minimum value of Y, E is the mean value of Y, s is the mean square sum of X, l is the minimum value of X, e is the mean value of X, and h is the maximum value of X.

3.1.3. Balance contrast enhancement technique

Following CLAHE, we used BCET to further improve the image. BCET solves the problem of discriminating color (RGB) composition without affecting the histogram pattern of the input image, and the contrast of the image may be stretched or compressed (x).

One of the most common causes of bad color composite images is color bias. To avoid this, the value range and mean of the three bands used for color composition must be identical. For this situation, the BCET offers a straightforward approach. BCET may stretch (or compress) images perfectly to a value range and mean specified by the user without affecting the main forms of the image histograms using a parabolic or cubic function defined by three coefficients a, b, c [41]. The answer is calculated using the parabolic function produced from the input image. Eq. (9) describes the parabolic function utilized in BCET.

$$\begin{aligned} &\text{if } H_{\text{slab}}(i) > N_{CL} \text{ then } H_{\text{slab-clip}}(i) = N_{CL} \\ &\text{else if } H_{\text{slab}}(i) + N_{\text{avggray}} \text{ then } H_{\text{slab-clip}}(i) = N_{CL} \\ &\text{else } H_{\text{slab-clip}}(i) = H_{\text{slab}}(i) + N_{CL} \end{aligned} \tag{9}$$

where, $H_{\text{slab}}(i)$ is original histogram, $H_{\text{slab-clip}}$ is clipped histogram of each slab at ith region, N_{CL} is the actual clip-limit and N_{avggray} is the average of the remaining pixels. The contrast of each slab is enhanced, where the histogram of output closely matches the histogram defined by the 'Distribution' parameter. The step of the redistribution of pixels is given by Eq. (10).

$$\text{Step} = N_{\text{gray}} / N_{\text{remain}}, \tag{10}$$

where N_{remain} is the remaining clipped pixels, and N_{gray} is the gray level of the slabs.

3.1.4. Image binarization - OTSU's thresholding

The image was thresholded to emphasize the features as the final step. Binarizing an image divides pixels into foreground and background, allowing key features to be distinguished from the background. Simple thresholding has the drawback of requiring you to manually select the threshold value [42]. We can manually test how good a threshold is by experimenting with different settings, but this is time-consuming and may fail in a real-world scenario. As a result, we will need a mechanism to calculate the threshold automatically.

In image processing, Otsu's approach is an adaptive thresholding method for binarization [43]. It can determine the best threshold value for the input image by examining all threshold values. The algorithm seeks a threshold that reduces intra-class variance, defined as the weighted sum of the variances of the two classes.

The basic concept is to split the image histogram into two clusters using a threshold established by minimizing the weighted variance of these classes denoted by $\sigma_w^2(t)$ [44]. The entire computation shown in Eq. (11).

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) \tag{11}$$

where, $w_1(t), w_2(t)$ are the probabilities of the two classes split by a threshold t that is between 0 and 255 inclusively.

There are two methods for determining the threshold [45]. The first is to reduce the within-class variance, which is stated above $\sigma_w^2(t)$, and the second is to maximize the between-class variance, which is defined in Eq. (12)

$$\sigma_b^2(t) = w_1(t)w_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2 \tag{12}$$

where μ_i is the class i mean.

The probability P for each pixel value in two independent clusters C_1, C_2 is derived using the cluster probability functions defined in Eqs. (13) and (14) respectively.

$$w_1(t) = \sum_{i=1}^{t} P(i) \tag{13}$$

$$w_2(t) = \sum_{i=t+1}^{I} P(i) \tag{14}$$

It is worth noting that the image may be represented as an intensity function $f(x, y)$, with gray-level values. The number of pixels with a given gray level i is denoted by i. The image's total number of pixels is n. As a result, the probability of gray-level i occurrence is represented by Eq. (15).

$$P(i) = \frac{n_i}{n} \tag{15}$$

The C_1 pixel intensity values are in $[1, t]$, while the C_2 pixel intensity values are in $[t + 1, I]$, where I is the maximum pixel value (255).

The next step is to calculate the means for C_1, C_2, which are indicated by $\mu_1(t), \mu_2(t)$ appropriately shown in Eqs. (16) and (17) respectively.

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{w_1(t)'} \tag{16}$$

$$\mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{w_2(t)} \tag{17}$$

Now recall the within-classes weighted variance equation from earlier. Next, we will determine the remaining components $\left(\sigma_1^2, \sigma_2^2\right)$ by combining all of the ingredients mentioned in Eqs. (18) and (19) respectively.

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{w_1(t)'} \tag{18}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{w_2(t)} \tag{19}$$

It is worth noting that if the threshold is set wrong, the variance of some classes will be relatively high. Therefore, we need to add together the within-class and between-class variances to obtain the overall variance:

$$\sigma_T^2 = \sigma_w^2(t) + \sigma_b^2(t) \tag{20}$$

where $\sigma_b^2(t) = w_1(t)w_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2$.

Overall, the authors have implemented four types of Image preprocessing techniques to improve image quality, as shown above. The results of all mentioned preprocessing techniques can be seen in Fig. 1.

3.2. Ensemble learning

Ensemble learning is an umbrella term for all the techniques formed by combining various inducers or base learners [46]. It is believed that the intuition for ensemble learning techniques

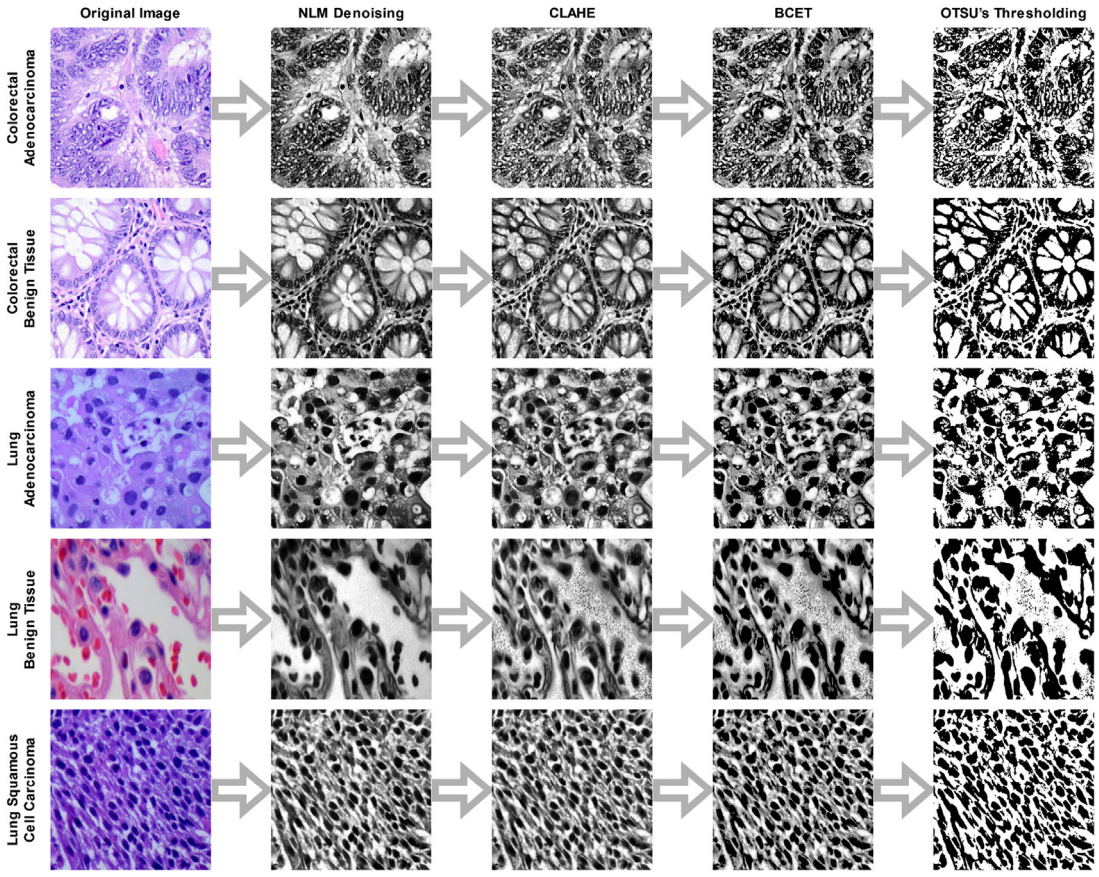

Fig. 1. Results of NLM, CLAHE, BCET, Otsu's thresholding techniques.

for solving a machine learning task stems from the concept of Condorcet's Jury Theorem (1785), which stated that the majority vote of a group of independent voters would be more competent than that of a single voter [47].

Ensembling techniques uses different models and combinedly training them to form the ensemble model or meta learner, improving the performance by reducing the errors or drawbacks of any particular method or base learner [48]. The ensemble approach usually outperforms any of the individual models used. There are various methods of performing ensemble learning on a set of models, with the main ones being — Bagging, Boosting, and Stacking algorithms [49]. Bootstrap aggregation or Bagging is used to form different base learners by varying the sample dataset. Boosting is an ensemble method that uses the errors made by previous classifiers to improve performance. Stacking is a method that uses different types of models on the training data and then a model to combine them. In this research, the authors used the Stacking algorithm to train the meta learner.

3.3. Condorcet's Jury Theorem

Condorcet's Jury Theorem is a mathematical theorem for calculating a group's accumulative decision-making relative probability. It states that if a majority of independent members in a group, individually, can make the correct decision rather than making a random choice, they are better at decision-making than just one member of that group [50]. This theorem in applications with Neural Networks helps ensemble the output of multiple trained deep learning models with good outcomes to give results better than any individual models.

Condorcet's Jury theorem applies to the following hypothetical situation: assume we have to choose between options $+$ or $-$. Assume one of the two choices is 'right', but we do not know which one [51,52]. Furthermore, imagine there are n models in a set, and the entire set must make a decision. A majority vote is one feasible way. So, each model has a vote X_i, which has a value of either $+1$ or -1 based on its calculated weights, and the group choice is either $+$ or $-$ depending on whether $S_n = \sum_{i=1}^{n} X_i$ is positive or negative.

3.3.1. Theorem

If individual votes $X_i, i = 1, \ldots, n$ are independent of one another, and each voter makes the correct decision with probability $p > \frac{1}{2}$, then as $n \to \infty$, the group's chance to reach a correct decision by majority vote approaches 1 as n increases [53]. Fig. 2 shows that as the number of voters increases (value of n), the likelihood of reaching a right choice by majority vote increases.

3.3.2. Proof

This is a consequence of the law of large numbers. Let $a = p - 1/2 > 0$. Since the problem is fair in $+$ and $-$, we may without loss of generality assume the correct answer is $+$ [54].

Then $EX_1 = -\left(\frac{1}{2} - a\right) + \left(\frac{1}{2} + a\right) = 2a > 0$, and the weak law of large numbers states that $\frac{S_n}{n}$ converges in probability [55] to $EX_1 = 2a$, where by converging in probability we mean that for any $\epsilon_1, \epsilon_2 > 0$ there is N large enough such that for every

$$n \geq N, P\left(\left|\frac{S_n}{n} - EX_1\right| < \epsilon_1\right) > 1 - \epsilon_2.$$

Taking $\epsilon_1 = 2a$, we see that the probability of a correct decision is

$$P\left(S_n > 0\right) = P\left(\frac{S_n}{n} > 0\right) \geq P\left(\left|\frac{S_n}{n} - 2a\right| < 2a\right) \to 1$$

which is what we needed to show.

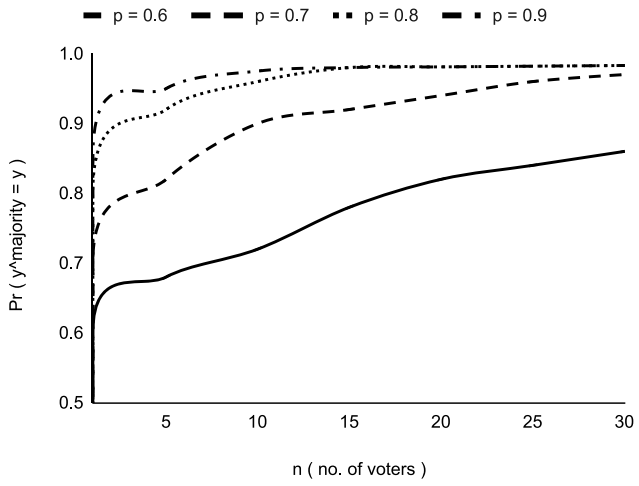

Fig. 2. Probability vs. no. of voters: Condorcet's Jury Theorem.

The probability, P_N, that a model will deliver the correct answer, we calculated using Condorcet's jury theorem [56] as shown in equation.

$$P_N = \sum_{i=m}^{N} \left(\frac{N!}{(N-i)\cdot i!} \right) (p)^i (1-p)^{N-i}$$

where, N = the number of models, p = the probability of an individual model being right m = the number of models required for a majority

3.3.3. Proof based on optimal Bayes classifier

The Bayes classifier which minimizes the probability of misclassification $\Pr(\hat{y} \neq y)$ follows the maximum a posterior (MAP) criteria $\hat{y}_{\text{Baycs}} = \text{argmax}_y \Pr\left(y \mid \{y_i\}_{i=1}^n\right)$. We now show that the majority rule is equivalent to MAP. According to Bayes rule,

$$\Pr\left(y \mid \{y_i\}_{i=1}^n\right) = \frac{\Pr\left(\{y_i\}_{i=1}^n \mid y\right) \Pr(y)}{\Pr\left(\{y_i\}_{i=1}^n\right)}$$

Therefore, for equally probable classes $\Pr(y=0) = \Pr(y=1)$, maximizing the MAP is equivalent to maximizing the likelihood: $\text{argmax}_y \Pr\left(y \mid \{y_i\}_{i=1}^n\right) = \text{argmax}_y \Pr\left(\{y_i\}_{i=1}^n \mid y\right)$ The likelihood for $y=1$ follows the Binomial distribution:

$$\Pr\left(\{y_i\}_{i=1}^n \mid y=1\right) = \prod_{i=1}^{n} p^{(y_i+1)/2} (1-p)^{(1-y_i)/2}.$$

Applying log on both sides, one get the log likelihood as:

$$\log\left(\Pr\left(\{y_i\}_{i=1}^n \mid y=1\right)\right) = \frac{1}{2}\sum_{i=1}^{n} \log(p)\,(y_i+1) + \log(1-p)\,(1-y_i)$$

Similarly, for $y=0$, $\log\left(\Pr\left(\{y_i\}_{i=1}^n \mid y=0\right)\right) = \frac{1}{2}\sum_{i=1}^{n} \log(1-p)\,(y_i+1) + \log(p)\,(1-y_i)$. Therefore, the log likelihood ratio $LLR = \log\left(\frac{\Pr\left(\{y_i\}_{i=1}^n \mid y=1\right)}{\Pr\left(\{y_i\}_{i=1}^n \mid y=0\right)}\right)$. equals to:

$$LLR = \log\left(\frac{p}{1-p}\right) \sum_{i=1}^{n} y_i$$

The latter concludes the proof as it shows that $\hat{y}_{\text{Bayes}} = \text{argmax}_y \Pr\left(y \mid \{y_i\}_{i=1}^n\right) = \text{sign}\left(\sum_{i=1}^n y_i\right)$ so that the MAP criteria coincides with the majority rule, which minimize the error probability (for equiprobable prior).

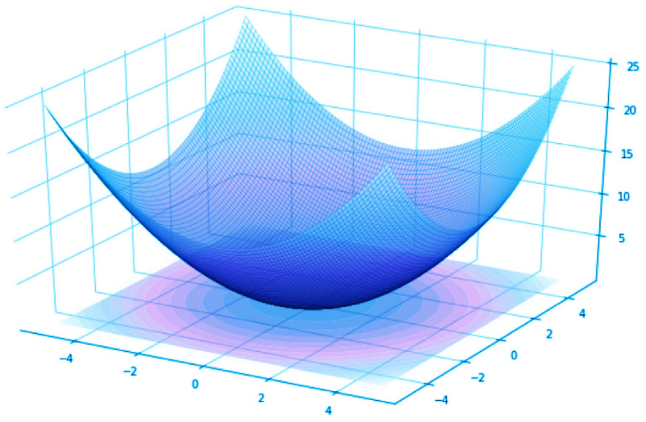

Fig. 3. Representation of $f(x) = \sum_i^n x_i^2/n$.

3.3.4. Condorcet's Jury Theorem in neural networks

Given a set of models who must choose between a right conclusion with probability $0 \leq p \leq 1$ and a wrong one with probability $1-p$, Condorcet's jury theorem [57] states :

1. If $p > 1/2$ (i.e., each model is more likely to classify correctly than incorrectly), increasing the number of models improves the likelihood that the majority selects correctly, and the probability of a correct decision approaches one as the number of models increases as shown in Fig. 2.
2. if $p < 1/2$ (such that each model is less likely to vote erroneously than correctly), adding additional models reduces the likelihood that the majority selects properly, and the probability of a right judgment is maximized for a model of size one.

3.4. Differential evolution — MetaHeuristic search based optimization

Differential evolution (DE) is a population-based metaheuristic search technique that improves a candidate solution through an evolutionary process [58]. These algorithms make minimal assumptions about the underlying optimization issue and may rapidly explore enormous design spaces. With multi-modal problem resilience, DE is one of the most adaptable and stable population-based search algorithms available.

Let $f(x) = \sum_i^n x_i^2/n$, for n = 32 dimensions. $f(x) = \sum_i^n x_i^2/n$ 2D view is shown in Fig. 3.

The goal is to find a solution $\mathbf{m}$ for which $f(\mathbf{m}) \leq f(\mathbf{p})$ for all $\mathbf{p}$ in the search-space, which means that $\mathbf{m}$ is the global minimum. If X_a is a candidate solution and X_{aT} is a target vector for X_a from X_b, X_c, X_d as per Eq. (21).

$$X_{aT} = X_b + F\,(X_c - X_d) \tag{21}$$

where $F \in [0, 2]$ and it is called as scaling factor which controls the amplification of differential variation. X_a and X_{aT} are subject to crossover operator as per Eq. (22).

$$X_{aU,i} = \begin{cases} X_{aT,i} & \text{if} \quad \text{rand} \leq CR \\ X_{a,i} & \text{otherwise} \end{cases} \tag{22}$$

where CR is crossover rate and $CR \in [0, 1]$. After that selection operator is applied as per Eq. (23).

$$X_{\text{anext}} = \begin{cases} X_{aU} & \text{if } f\,(X_{aU}) \leq f\,(X_a) \\ X_a & \text{otherwise.} \end{cases} \tag{23}$$

Differential Evolution differs from standard genetic algorithms because it utilizes directional information within the population

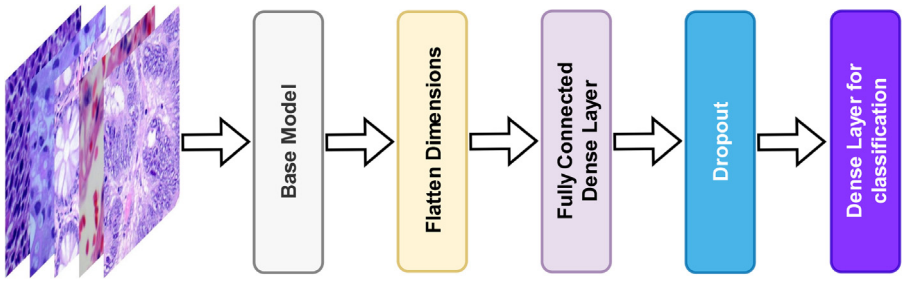

Fig. 4. Modified DCNN model architecture.

through a target and unit vector. These capabilities allow differential evolution to converge faster to solutions at the cost of poor exploration. The detailed procedure of Differential Evolution Optimization is demonstrated in Algorithm 1. Also, a comparison between different metaheuristic computing techniques is shown in Table 2.

Algorithm 1: Differential Evolution Optimization Algorithm

BEGIN
(1) Generate a random population of people at the beginning of the search area.
(2) **while** *iter* <= *max num of generations* **do**
  (3) iterate over every individual in the population
    (A) perform mutation
    (B) perform recombination ("crossover" in GA lingo)
    (C) perform selection
  (4) **if** stopping criterion has been met:
    exit and return best individual
  else
    iter = iter + 1
    go back to step 3
end
END

3.5. Deep feature extraction and model training

Deep feature extraction's primary goal is to extract prominent, discriminating information from the original raw images and express it in a lower-dimensional space. Various pre-trained networks were employed for deep feature extraction in this study. From Fig. 4, we flatten the dimensions after freezing all layers except the final layer of a pre-trained DCNN model. Then we added a fully connected dense layer consisting of 1024 neurons. We added a dropout layer to remove 50% of neurons in each iteration to avoid overfitting. Finally, we added a dense layer consisting of 5 neurons for classification. The deep features were then trained using the softmax classifier for classification purposes. The modified EfficientNet architecture is also shown is Fig. 5.

For feature extraction and classification, there is often a need for optimizing and improving deep learning models. The use of many techniques can do this. The methods and parameters used for the training of our models have been explained in detail in this section. These include the Cross-entropy loss function, Softmax activation function, learning rate scheduler, and optimizers such as Adam.

3.5.1. Loss function

Loss functions are used to analyze how well the algorithm models the dataset. It computes the difference between the current and desired output to optimize the model performance. The lesser the loss value, the better the model will do, as the weights will be optimized to minimize the loss function value. The Cross entropy loss function is a popular function used for classification problems.

Categorical cross-entropy, one of the subtypes of cross-entropy, is a loss function used for multi-class classification and has been used in this work. The mathematical Eq. (24) describes the computation of the cross-entropy loss function.

$$L_{CE} = -\sum_{i=1}^{n} t_i \log\left(p_i\right), \text{ for n classes,} \tag{24}$$

where t_i is the truth label and p_i is the Softmax probability for the ith class.

3.5.2. Optimizer

Adam optimization is a stochastic gradient descent method that has its name derived from Adaptive Moment Estimation [59]. This is because it uses estimations of the first and second moments of gradient descent to adapt and compute the individual learning rates for each network weight. It is inherited from RMSProp and AdaGrad and has their combined advantages. Adam's optimization method is computationally very efficient as it requires less memory even when working with problems involving a large number of data and parameters. Adam is intuitively a combination of RMSProp and Stochastic Gradient Descent with Momentum. It uses the moving average of the gradient in place of the gradient itself, like in SGD with momentum, and it uses squared gradients to scale the learning rate like in RMSProp. Adam optimizer uses an exponentially decaying average of past gradients (mt) and past squared gradients (vt) as defined in Eqs. (25) and (26) respectively. The term β_1 and β_2 are the forgetting factors for the mean and non-centered variance of the gradient respectively.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] \tag{25}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2 \tag{26}$$

where,

1. ϵ = a small +ve constant to avoid 'division by 0' error when $(v_t -> 0) \cdot \left(10^{-8}\right)$
2. $\boldsymbol{\beta_1}\&\boldsymbol{\beta_2}$ = decay rates of the average of gradients in the above two methods. ($\beta_1 = 0.9 \& \beta_2 = 0.999$)
3. $\boldsymbol{\alpha}$ - Step size parameter/learning rate (0.001)

3.5.3. Swish activation function

The study uses the Swish function by Google, proposed by Ramachandran et al. [60]. Swish is an activation function discovered after rigorous experimentation using automatic search techniques to find the best possible activation function. As of now, the most successful and popular activation function is ReLU.

Table 2
Comparison of different metaheuristic computing techniques.

Genetic Algorithm (GA)	Differential Evolution Optimization (DEO)	Particle Swarm Optimization (PSO)
Performance is faster than DEO and PSO Less parameters than DEO Difficult to be implemented	Performance is faster than PSO Less parameters than PSO Easier to be implemented compared to GA and PSO	Performance is faster than GA More parameters than DEO and GA Easier to be implemented than GA
Repeatedly modifies a population of individual solutions by mainly concentrating on global search.	Utilizes directional information within the population through the usage of a target and unit vector	Carries out global search and local searches simultaneously
Converge slower	Converge faster than GA and PSO	Converge faster than GA

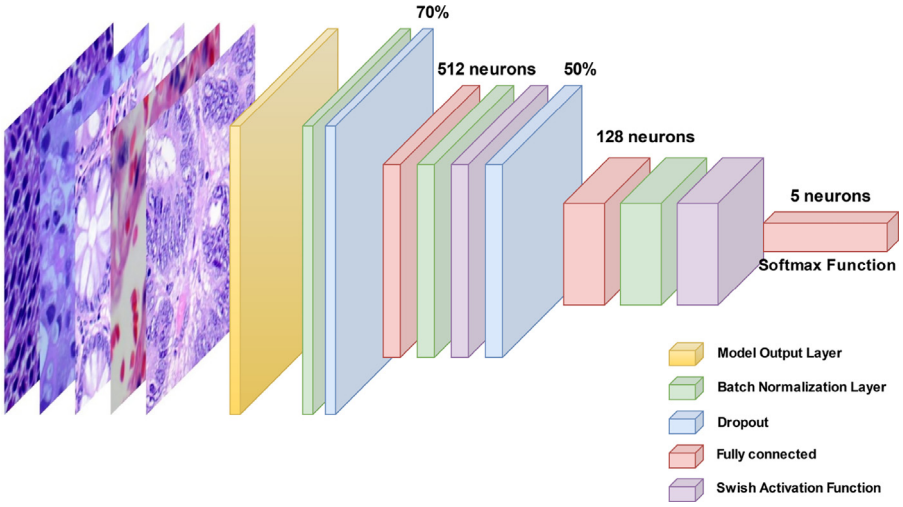

Fig. 5. Modified EfficientNet architecture.

The authors have claimed that Swish has consistently compared to and outperformed the ReLU activation function on deeper networks after implementing the ImageNet dataset on different pre-trained models. Furthermore, Swish was said to perform better than the other functions with a little tuning. The Swish activation function is mathematically represented as shown in Eq. (27).

$f(x) = x \cdot \text{sigmoid}(\beta x)$ where β is a constant or trainable parameter

$$S(x) = \frac{1}{1 + e^{-x}} \tag{27}$$

$S(x) =$ sigmoid function e $=$ Euler's number

3.5.4. Classifier

The softmax classifier generalizes the binary Logistic Regression classifier for multiple classes. It provides normalized class probabilities as output for each class. It takes a vector of real values scores and squashes it to values between 0 and 1 that sum to 1 for all classes. The softmax classifier minimizes the cross-entropy loss between the predicted class probabilities and the truth labels. Softmax function is defined in Eq. (28).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{28}$$

where, $\sigma =$ softmax, $\vec{z} =$ input vector, $e^{z_i} =$ standard exponential function for input, $K =$ number of classes in the multi-class, e^{z_j} $=$ standard exponential function for output.

3.5.5. Learning rate schedule : ReduceLROnPlateau

A learning rate schedule alters the learning rate throughout the learning process and is most commonly altered between epochs/iterations [61]. When learning becomes stagnant, models frequently benefit from slowing the learning rate by a factor of 2-10 [62]. This callback watches a quantity and reduces the learning rate if no progress is noticed after a 'patience' number of epochs [63]. This is primarily accomplished by using two parameters namely decay and momentum. Time-based learning schedules alter the learning rate based on the learning rate of the previous time iteration. When the decay is taken into consideration, the learning rate may be calculated as shown in Eq. (29).

$$\eta_{n+1} = \frac{\eta_n}{1 + dn} \tag{29}$$

where, η is the learning rate, d is a decay parameter and n is the iteration step.

When a measure stops improving, ReduceLROnPlateau is a callback that reduces the learning rate. This callback tracks a quantity and reduces the learning rate by a "factor" value if no progress is noticed after a "patience" number of epochs as shown in Eq. (30).

$$\text{new } lr = lr * \text{ factor} \tag{30}$$

3.6. Existing ensemble methods

Ensemble learning approaches combine the benefits of both deep learning and ensemble learning, resulting in a model with improved generalization performance. Ensemble learning was used in several research studies to improve performance, reduce errors, and avoid overfitting. Researchers have developed several strategies for improving ensemble methods by discovering various weighting schemes. This section discusses some of the previously utilized ensemble and weighting scheme approaches, including their advantages and disadvantages.

3.6.1. Weighted average ensemble

In an ensemble model, each base learner is given the same weight for a prediction. There are times when we want extremely competent models to contribute more to an ensemble prediction, and other times when we want less competent models to

contribute less to an ensemble prediction. So, penalizing a less accurate model and rewarding a more accurate model can lead to even better predictions. A weighted average ensemble allows different models to contribute to a prediction in proportion to their degree of assurance or anticipated performance.

3.6.2. Averaging technique for assigning weights

Instead of generating just one model, ensemble averaging involves creating numerous models and merging them to achieve the desired outcome. Because the multiple flaws of the models "average out", an ensemble of models frequently outperforms a single model. The most basic form of ensemble learning is model averaging.

A more advanced variant of ensemble average considers the final result a weighted sum rather than a simple average of all the experts. If each expert is y_i, then the overall result $\tilde{y}$ can be defined as Eq. (31).

$$\tilde{y}(\mathbf{x};\alpha) = \sum_{j=1}^{p} \alpha_j y_j(\mathbf{x}) \tag{31}$$

where α is a set of weights.

3.6.3. Grid search technique for assigning weights

Grid search values are an exhaustive yet straightforward way of determining ensemble member weights. We can create a course grid with weight values ranging from 0.0 to 1.0 in 0.1 increments and then build all base learner's element vectors. A Cartesian product is made by generating all conceivable combinations as shown in Eq. (32)

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\} \tag{32}$$

We can now count each weight vector produced by the Cartesian product, normalize it, and assess it by generating a prediction and retaining the best for inclusion in our final weight averaging ensemble. We may report the performance of our weight average ensemble on the test dataset after it has been identified, which we anticipate being better than the best single model and ideally better than the model averaging ensemble.

One drawback of this method is that the weight vectors do not total to one (the unit norm), as is necessary. However, by computing the total absolute weight values (known as the L1 norm) and dividing each weight by that value, we may compel the resulting weight vector to have a unit norm.

$$\text{L1 norm: } \|\mathbf{w}\|_1 = \sum_{i}^{n} |w_i|$$

$$\text{Squared L2 norm: } \|\mathbf{w}\|_2^2 = \sum_{i}^{n} w_i^2$$

This procedure is also very time taking and computationally costly. Each time a new best-performing set of weights is identified, its performance on the test dataset is reported. A directed optimization technique is an alternative to exploring weight values which is demonstrated in Section 3.7.2

3.6.4. Rank based Fuzzy ensemble

In this approach let us say the base learner i's confidence scores for C number of classes are $\left(P_1^i, P_2^i, P_3^i, \ldots, P_C^i\right)$, here i = 1, 2, 3. We start by adding up all of the confidence scores from each of the base learners [64]. As $\left(P_1^i, P_2^i, P_3^i, \ldots, P_C^i\right)$ will largely follow equation to express probability as per Eq. (33).

$$\sum_{k=1}^{C} P_k^i = 1, \quad \forall i = 1, 2, 3 \tag{33}$$

Let $\left(R_1^{i_1}, R_2^{i_1}, R_3^{i_1}, \ldots, R_C^{i_1}\right)$ and $\left(R_1^{i_2}, R_2^{i_2}, R_3^{i_2}, \ldots, R_C^{i_2}\right)$ are fuzzy ranks obtained using 2 non-linear functions. Eqs. (34) and (35) are used to compute the fuzzy rankings.

$$R_k^{i_1} = 1 - \tanh\left(\frac{\left(P_k^i - 1\right)^2}{2}\right) \tag{34}$$

$$R_k^{i_2} = 1 - \exp\left(-\frac{\left(P_k^i - 1\right)^2}{2}\right) \tag{35}$$

A classification is rewarded in Eq. (34). If x approaches 1, the value of Eq. (34) rises, indicating that the quantity of reward rises. When we compute deviation from Eq. (34) in Eq. (35), i.e., if x approaches 0, the divergence will be greater.

Let $\left(RS_1^i, RS_2^i, RS_3^i, \ldots, RS_C^i\right)$ be the fused rank scores, where RS_k^i is given by Eq. (36).

$$RS_k^i = R_1^{i_1} \times R_1^{i_2} \tag{36}$$

$$FS_k = \sum_{i=1}^{L} RS_k^i, \forall k = 1, 2, \ldots, C \tag{37}$$

This combined score may be calculated as the final score for each class. Using Eq. (37), we identify the class with the lowest fused score and declare it the winner.

3.7. Proposed ensemble methods

3.7.1. Deep stacking ensemble based approach

In this approach, first, we trained the N number of the best-performing models. We can discover the value of N by experimenting with various models on the provided dataset. Because specific models may perform well on a dataset while others do not, experimenting is the best technique for choosing models. We now trained and save the weights of all models one by one. After training all the models, we freezed all the layers except the top one and concatenate the output layers of all models. This procedure is called stacking. After that, we added a multilayer perceptron layer, a dense layer of 32 neurons, and finally, a three-neuron output layer for classification.

3.7.2. Differential evolution - Heuristics to optimize weights

Optimization is a search process, but instead of randomly or exhaustively sampling the space of potential solutions, it employs any available information to choose the next step in the search, such as a set of weights with reduced error. Differential Evolution is one of the few stochastic global search algorithms which successfully optimizes functions with continuous inputs. It is necessary to specify a function that will assess a collection of weights and return a score to be minimized. We can reduce the classification error (1 – accuracy) as much as possible. The loss function will be utilized during the optimization phase as the evaluation function. The optimization process' boundaries must also be specified. The boundaries can be defined as a five-dimensional hypercube with values ranging from 0.0 to 1.0 (for example, 5 weights for the 5 ensemble members). This proposed approach is demonstrated in Fig. 6.

The detailed training procedure of the Differential Evolution Optimization-based ensemble model can be seen in algorithm 2. After training the initial classifiers, we randomly assigned the weights to each classifier and then optimize them using the algorithm. The Differential Evolution algorithm runs for 1000 iterations, and for each iteration, it computes the loss value of every classifier and then optimizes the weights. After optimizing the weights, we normalized the weights $w_j = \frac{a_j}{\sum_{n=1}^{N} a_j}$. After

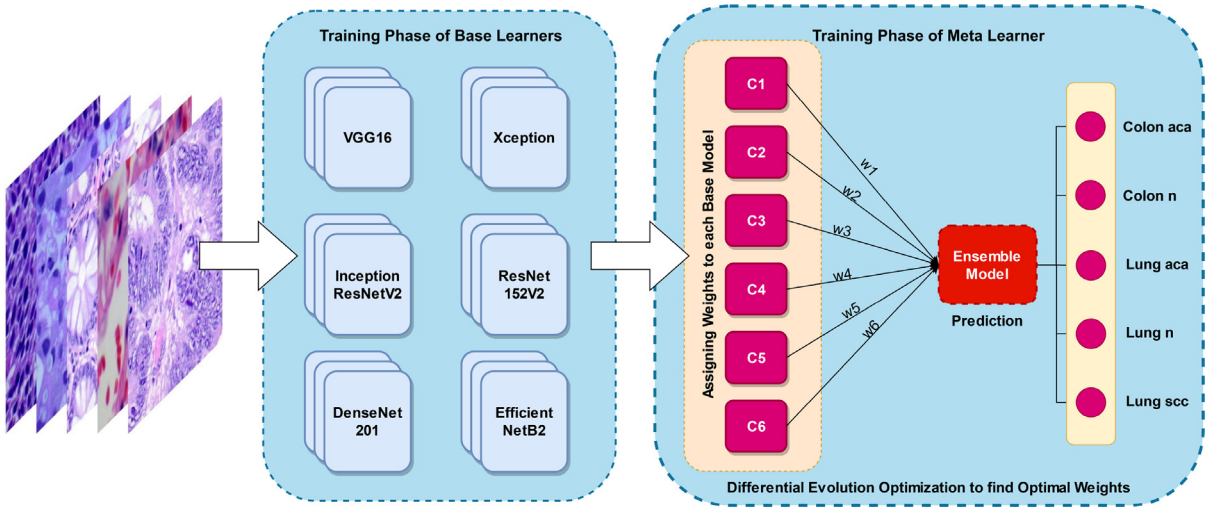

Fig. 6. Flowchart for weights optimization of ensemble model based on Differential Evolution optimization algorithm.

getting the optimized weights, we calculated our ensemble model score. For that, firstly we get the probabilistic output vector as shown in Eq. (38).

$$\mathbf{p}_m(d_i) = \left\{p_{m,k}(d_i)\right\} \tag{38}$$

for all K classes where $1 \leq k \leq K$

Then we computed the weighted sum as per Eq. (39).

$$p_k(d_i) = \sum_{m=1}^{M} p_{m,k}(d_i) \cdot w_{i,k} \tag{39}$$

After that we get the output class label with the maximum class probability as shown in Eq. (40).

$$c(d_i) = \arg\max_{1\leq k\leq K} \{p_k(d_i)\} \tag{40}$$

3.7.3. Majority voting based on Condorcet's Jury Theorem

In this proposed Algorithm, firstly, we trained the N number of the best-performing model. Then, we saved the predicted output labels once each model had been trained. Now, we generated K number of arrays for K number of labels after recording the expected labels of each model for each image. Then, we calculated the score of each label for each model by iterating through each model. Finally, we used the majority vote as the ultimate choice of N models. This proposed approach is demonstrated in Fig. 7. The majority voting based on Condorcet's Jury Theorem is used to calculate the final score as per algorithm 3.

4. Experimental results and discussion

4.1. Dataset description : LC25000

The dataset used in this research is "Lung and Colon Cancer Histopathological Image Dataset (LC25000)" by Borkowski et al. [15] to assist with the requirement for a large variety of freely available image data used by researchers for training ML models. The data is HIPAA compliant and validated and consists of five equally balanced classes as 250 benign lung tissue, 250 lung adenocarcinomas, 250 lung squamous cell carcinomas, 250 benign colon tissue, and 250 colon adenocarcinomas. The images for the 3 lung cancer classes and 2 colon cancer classes were obtained from pathology glass slides. The original images were resized to 768 × 768 pixels, and augmentation was performed using the python package Augmentor. The dataset was expanded 20 times to a total of 25 000 images with the following augmentations performed:

Algorithm 2: Algorithm for weights optimization using Differential Evolution Algorithm

Input:
$\delta 1$: total number of images in training dataset.
$\delta 2$: total number of images in validation dataset.
$\delta 3$: total number of images in testing dataset.
d_i : the ith input sample of testing dataset.
N : total number of classifiers.
K : total number of classes.
ω^* : initial weights of trained classifier.
Output :
optimized weights vector
predicted class probability vector
begin:
while $i = 1$ *to* $\delta 1$ **do**
  while $i = j$ *to* N **do**
    1. Train all N_j and compute the ω^*
  end
end
2. Define bounds of each weights
while $i = 1$ *to* 1000 **do**
  while $j = 1$ *to* N **do**
    3. Compute loss value (error rate) of j on $\delta 2$.
    4. Optimize weights using Differential Evolution Search minimizing loss value.
    5. Normalize obtained weights $w_j = \frac{a_j}{\sum_{n=1}^{N} a_j}$.
  end
end
while $i = 1$ *to* $\delta 3$ **do**
  while $n = 1$ *to* N **do**
    6. Get the probabilistic output vector $(\mathbf{p}_m(d_i) = \left\{p_{m,k}(d_i)\right\}$ for all K classes where $1 \leq k \leq K$) of the n th classifier.
  end
  while $n = 1$ *to* N **do**
    7. $p_k(d_i) = \sum_{m=1}^{M} p_{m,k}(d_i) \cdot w_{i,k}$
  end
  $c(d_i) = \arg\max_{1\leq k\leq K} \{p_k(d_i)\}$
  Get the output class label with the maximum class probability for the i th input sample.
end

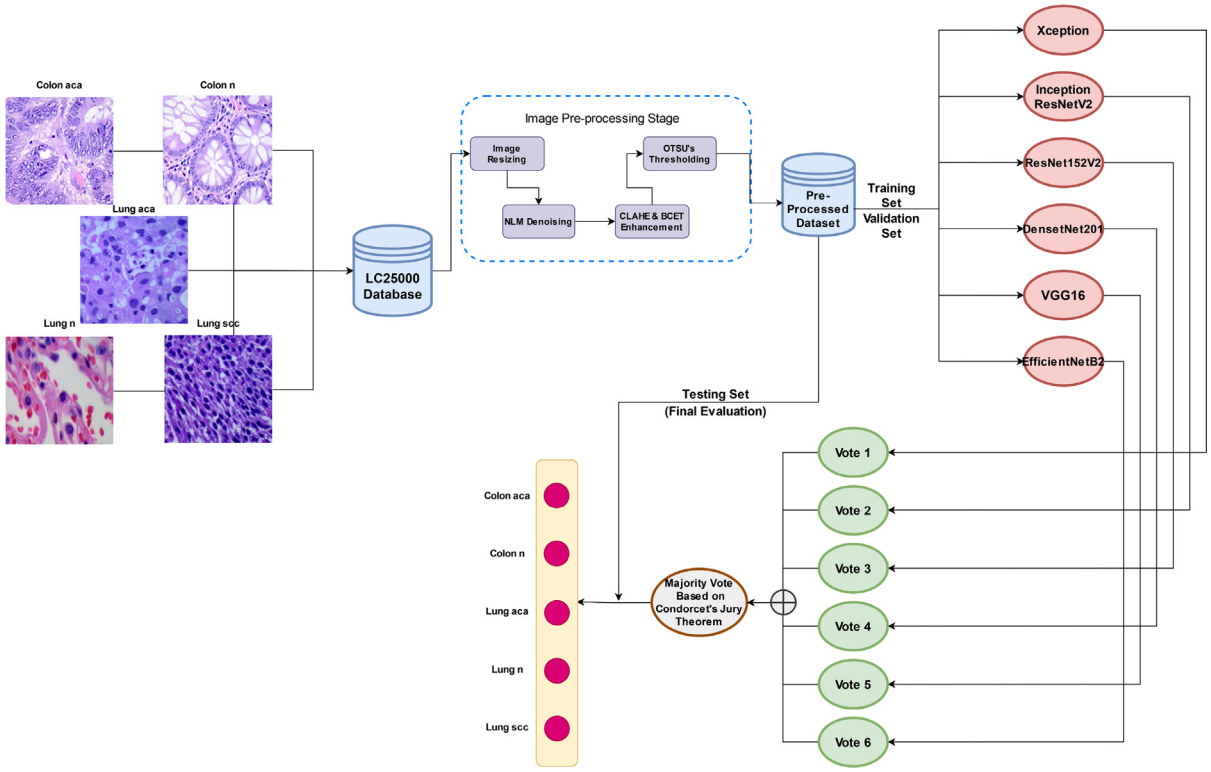

Fig. 7. Flowchart for final score calculation of ensemble model based on Condorcet's Jury Theorem.

1. Left and right rotations (up to 25 degrees, 1.0 probability)
2. Horizontal and vertical flips (0.5 probability)

This publicly available dataset consisting of 25 000 lung and colon histopathological images divided into 5 classes, with 5000 images in each class, was used in this research.

4.2. Dataset division

A deep learning model may achieve an accuracy of 99%, but when tested on real-world images, it fails to classify them accurately. So to avoid overfitting and model selection bias, it is ethical to divide the dataset properly into training, validation, and testing sets. In addition, our parameter estimates have more variance when we have less training data. Likewise, our performance metric will have more variance if we have less testing data. Therefore, we should divide the data in such a way that none of the variances is excessive. As a starting point, generally, everyone chooses the training set as 80% of the whole dataset and the remaining 10%–10% for validation and testing, respectively. But allocating more data in the final testing set ensures the robustness of the proposed method and reduces possible failure in real-world testing [65,66]. Therefore the authors divided the entire dataset into a 70% training, 10% validation, and 20% testing set, respectively as shown in Table 3.

4.3. Experimental setup

All the code implementations in the study were performed using the Tensorflow framework in python. Deep learning models were trained on a workstation equipped with GPU Nvidia RTX 3080 with a compute capability of 8.60, 16 GB of GPU RAM, and 64 GB of RAM. All the models were trained for 150–250 epochs to obtain accurate training, validation, and testing classification accuracy.

4.4. Results and discussion

The first and most important step in constructing a deep learning model is to define the network architecture. The authors prefer to use pre-trained networks to extract deep features as they have been initially trained on a large-scale ImageNet dataset. Therefore, we save a lot of computational power when adjusting weights to match our LC25000 dataset. In this manuscript, the authors implemented 6 DCNN models namely VGG16, Xception, InceptionResNetV2, ResNet152V2, DenseNet201 and EfficientNetB2 to extract deep features.

Before feeding the images to the models, the Image preprocessing step is crucial. It plays a vital role in boosting a classifier's accuracy. With the help of image processing, noise can be removed from the dataset. Furthermore, by basic adjustments, we can highlight the features in an image more clearly so that our classifier can learn it more efficiently, thus boosting accuracy.

In this manuscript, the authors have first removed the noise from the dataset by using the NLM-Denoising Algorithm. Then CLAHE and BCET are used for contrast enhancements equalizing the histograms properly. With great precision and contrast limitation, CLAHE performs histogram equalization in small patches or small tiles, thus reducing the problem of noise amplification too. After that, the automatic image thresholding was done with the OTSU's method.

For optimizing our DCNN models, the Adam optimizer is used with an initial learning rate of 0.001, the exponential decay rate for the 1st moment as 0.9, the exponential decay rate for the 2nd moment as 0.999, and an epsilon value of 1e−7. Experimentation has determined that the learning rate and other hyperparameters are chosen as the most ideal settings as per Table 4. These hyperparameters are selected utilizing the Grid search technique for model tuning and optimization. The batch size is set at 32, and the model training is halted at 150 epochs. As a consequence, the best model weights are preserved. To conserve the finest weights, the

Algorithm 3: Algorithm for Majority Voting Based on Condorcet's Jury Theorem

Input:
$\delta 1$: Training Set
$\delta 2$: Validation Set
$\delta 3$: Testing Set
α : number of classifiers.
β : number of predicted labels for $\delta 3$
ω^* : initial weights of trained classifier.
Output: : Classification as Lung or Colon images.
begin:
while $i \leq \alpha$ **do**
  1. Train DCNN and compute the (ω^*)
  2. Predict the output labels with trained DCNN in x
  3. $z = \text{argmax} f(x)$
end
4. Initialize K arrays
while $i \leq \alpha$ **do**
  while $j \leq \beta$ **do**
    5. If $j == 0$ then $K_0 + +$
    6. If $j == 1$ then $K_1 + +$
    7. If $j == 2$ then $K_2 + +$
    8. If $j == 3$ then $K_3 + +$
    9. If $j == 4$ then $K_4 + +$
  end
end
10. Initialize *final score* array f
while $i = 1 \leq \beta$ **do**
  if $K_0[i] >= K_1[i]$ **and** $K_0[i] >= K_2[i]$ **and** $K_0[i] >= K_3[i]$ **and** $K_0[i] >= K_4[i]$:
    f.append(0)
  elif $K_1[i] >= K_0[i]$ **and** $K_1[i] >= K_2[i]$ **and** $K_1[i] >= K_3[i]$ **and** $K_1[i] >= K_4[i]$:
    f.append(1)
  elif $K_2[i] >= K_0[i]$ **and** $K_2[i] >= K_1[i]$ **and** $K_2[i] >= K_3[i]$ **and** $K_2[i] >= K_4[i]$:
    f.append(2)
  elif $K_3[i] >= K_0[i]$ **and** $K_3[i] >= K_1[i]$ **and** $K_3[i] >= K_2[i]$ **and** $K_3[i] >= K_4[i]$:
    f.append(3)
  elif $K_4[i] >= K_0[i]$ **and** $K_4[i] >= K_1[i]$ **and** $K_4[i] >= K_2[i]$ **and** $K_4[i] >= K_3[i]$:
    f.append(4)
end
11. Calculate the final score between original labels and final predicted labels based on Condorcet's Jury Theorem.

Table 3
Dataset division.

	Training set	Validation set	Testing set	Total
Lung benign tissue	3500	500	1000	**5000**
Lung adenocarcinoma	3500	500	1000	**5000**
Lung squamous cell carcinoma	3500	500	1000	**5000**
Colon adenocarcinoma	3500	500	1000	**5000**
Colon benign tissue	3500	500	1000	**5000**
Total	**17 500**	**2500**	**5000**	**25 000**

authors employed early stopping callbacks. When a monitored parameter stops improving, early stopping stops the training.

After the feature extraction, the feature vectors obtained are trained with a Multi-Layer Perceptron network to perform classification. Here, the authors have done both 2 and 3-class classification on lung and colon images separately. In addition, the authors

Table 4
Hyperparameters.

Hyper-parameters	Values
Optimizer	Adam
Dropout	0.5
Batch size	32
Exponential decay rate for 1st momentum (β_1)	0.9
Exponential decay rate for 2nd momentum (β_2)	0.999
Epsilon (ϵ)	1e−7
Initial learning rate (α)	0.001
Factor	0.1
Patience	10
Total no. of epochs	150

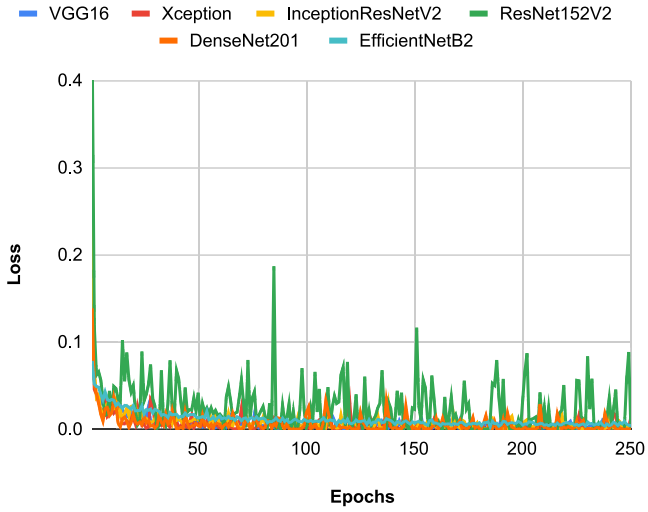

Fig. 8. Loss vs. Epochs curve for 2-class classification during the training procedure.

have also done a 5-class classification on the combined dataset to check the robustness of the proposed methods. Performing classification on all 5 classes is more challenging than 2&3 classes separately since the model has to learn to discriminate the features of the lung from colon cancer. Table 5 reports the 2, 3, and 5-class accuracies of DCNN models.

In this manuscript, the authors proposed three different types of ensembling methods, namely the Deep Stacking ensemble model, Optimized Weights ensemble model, and Jury-based ensemble model. The accuracy of the developed DCNN models is related to the number of epochs. When the number of epochs increases from 1 to 150, the accuracy value rises. The magnitude of loss is also dependent on the number of epochs. When the number of epochs increases from 1 to 150, the loss value decreases. Figs. 8 and 9 depicts the loss and accuracy curves of the DCNN models on 2-class respectively i.e., Colon database only. Similarly Figs. 10 and 11 reports the loss and accuracy curves on 3-class respectively i.e., Lung database only and Figs. 12 and 13 reports the loss and accuracy curve on both the combined dataset of lung and colon respectively.

After training of DCNN models, i.e., our base learners, the authors trained the meta learner, i.e., our ensemble model. Table 5 shows the ensemble model accuracy of the Deep stacking Ensemble Based approach. We picked our base learners from different families since each classifier has its own set of benefits merged when ensembled. The ensemble model is trained on our validation set of only 10% of the whole data. Adjusting the weights of a meta-learner using the same training set will almost certainly lead to overfitting. A more robust strategy is utilizing

Table 5
Validation accuracy (V.A) and Test accuracy (T.A) for the 2-class, 3-class and 5-class classification of 6 base classifiers and the proposed ensemble methods.

Classifier	2-class		3-class		5-class	
	V.A	T.A	V.A	T.A	V.A	T.A
InceptionResNetV2	99.90	99.90	97.73	98.30	98.32	98.05
Xception	99.80	99.90	98.27	98.50	98.28	98.22
VGG16	99.50	99.90	98.13	98.07	98.12	98.37
ResNet152V2	99.90	99.84	97.93	98.43	98.40	98.96
DenseNet201	99.90	99.90	99.40	99.20	99.40	99.44
EfficientNetB2	100	99.55	100	99.78	99.72	99.52
Deep Stacking ensemble model	100	99.90	99.78	99.80	99.60	99.64
Optimized Weights ensemble model	–	99.92	–	99.83	–	99.78
Jury based ensemble model	–	**99.95**	–	**99.90**	–	**99.88**

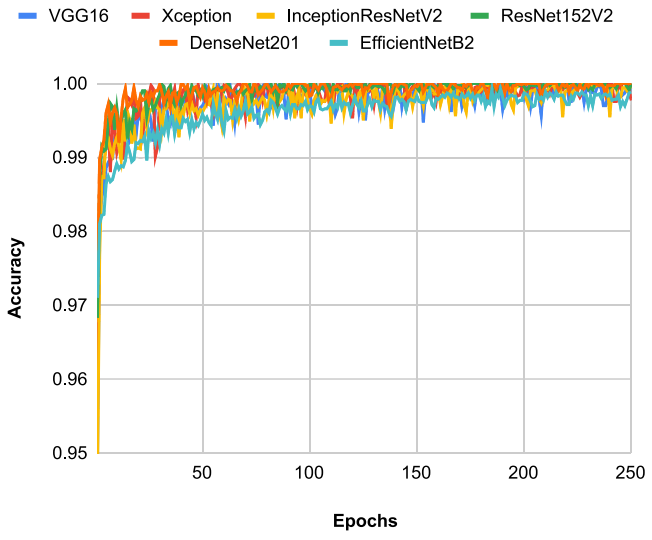

Fig. 9. Accuracy vs. Epochs curve for 2-class classification during the training procedure.

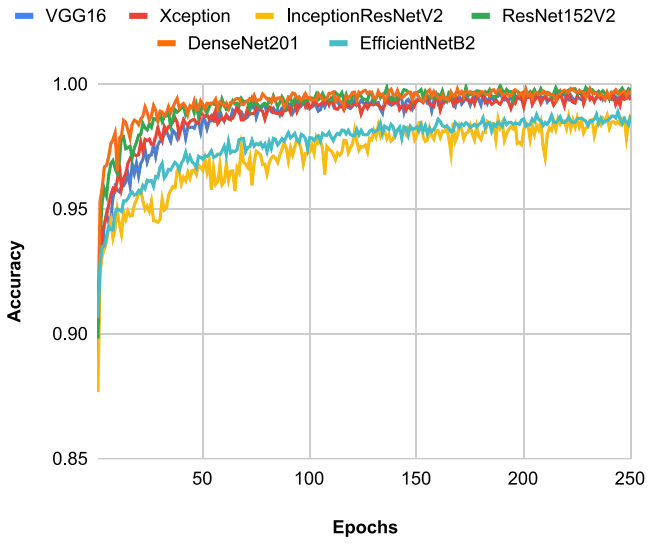

Fig. 11. Accuracy vs. Epochs curve for 3-class classification during the training procedure.

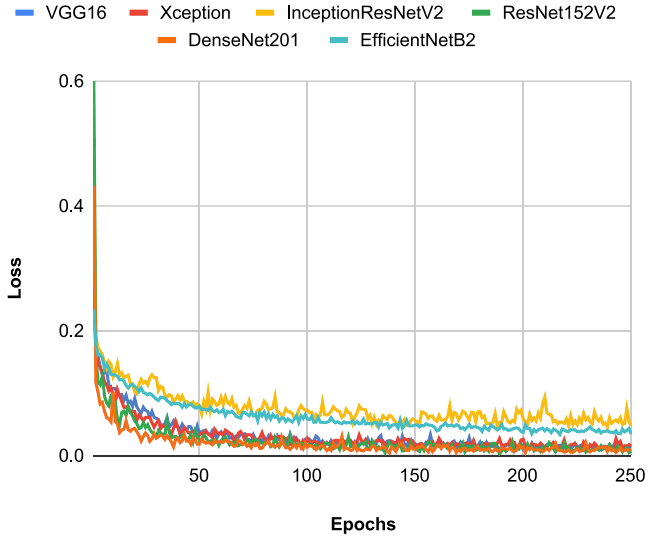

Fig. 10. Loss vs. Epochs curve for 3-class classification during the training procedure.

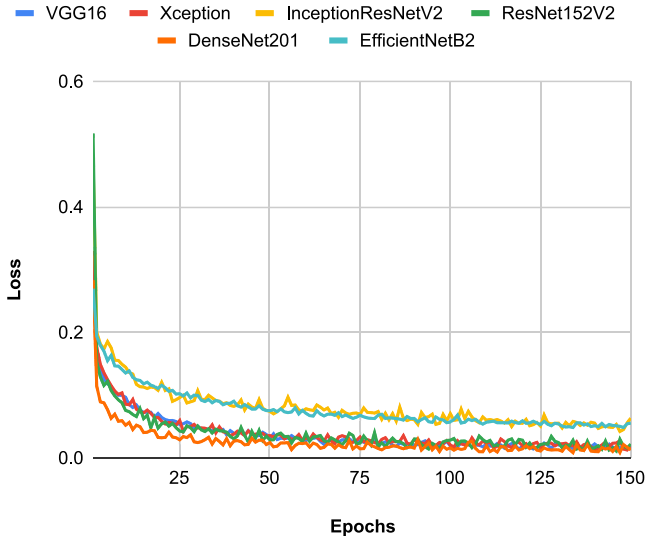

Fig. 12. Loss vs. Epochs curve for 5-class classification during the training procedure.

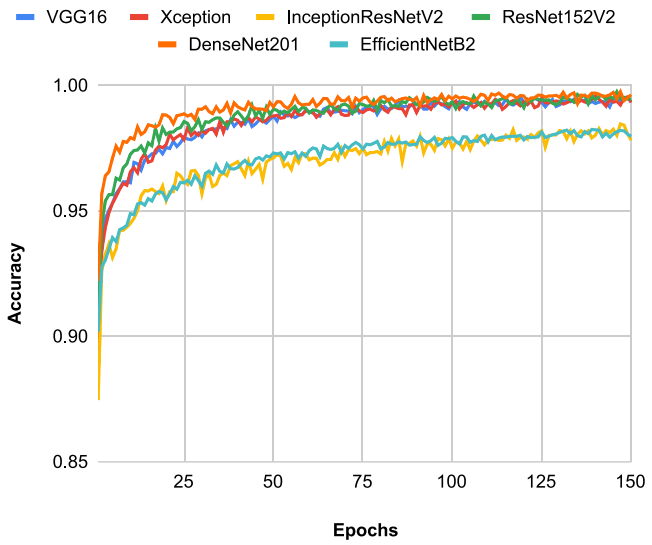

Fig. 13. Accuracy vs. Epochs curve for 5-class classification during the training procedure.

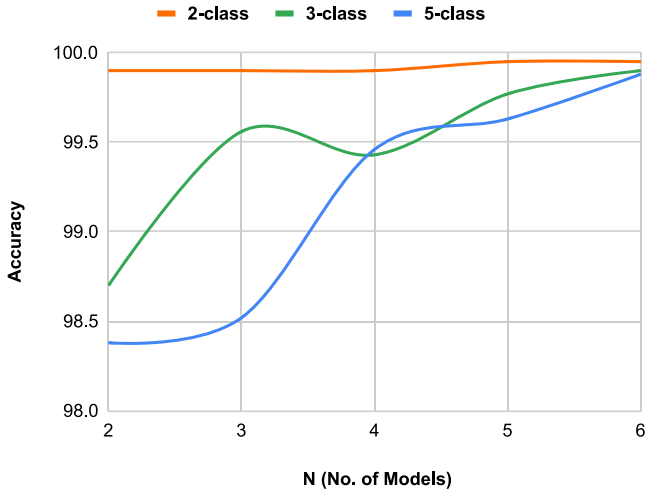

Fig. 14. Result of Condorcet's Jury Theorem.

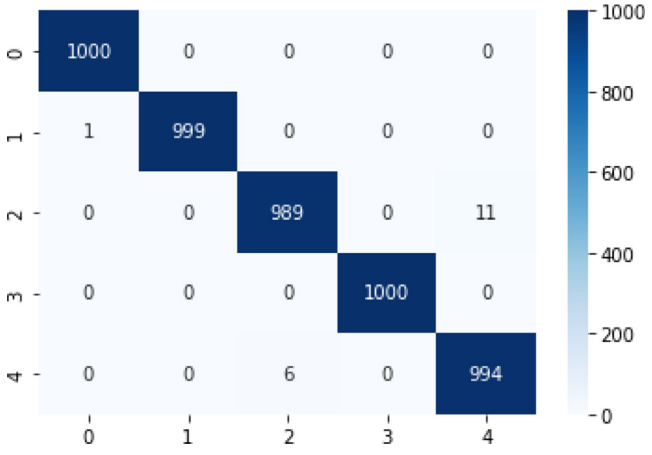

Fig. 15. Confusion matrix of the proposed Deep Stacking based ensemble method.

Table 6
Optimized Weights with differential evolution optimization.

Method	$w1$	$w2$	$w3$	$w4$	$w5$	$w6$
Model averaging	0.166	0.166	0.166	0.166	0.166	0.166
Differential evolution	0.0013	0.009	0.015	0.245	0.348	0.383

a holdout validation dataset that the ensemble members do not see during training. 10% validation data is not a lesser data for training since we only need to adjust the weights of the last layer.

One of the drawbacks of ensembling is that each classifier is given equal weightage. We sometimes want extremely competent models to contribute more to an ensemble prediction and less competent models to contribute less. So to penalize a less accurate model and give importance to a more accurate model, we assigned weights to each base model. Finding the optimal weights is also a challenging task, and a random weight search might be computationally intensive. In this case, a heuristic-based approach can be highly effective. So, the authors have used a differential evolution-based meta-heuristic search to search for optimal weights to maximize the meta learner accuracy. The optimal weights output by the search is shown in Table 6. By assigning these weights to individual classifiers, the accuracy improved by an 0.14% margin as shown in Table 5.

Searching for the optimal weights and training the meta learner again is a computationally heavy task. Instead, the authors have applied Condorcet's Jury theorem to calculate majority votes. In this procedure, only the base learner is trained and validated once. The model accuracy improves significantly by ensembling the top-performing models using Jury's theorem. According to Condorcet's Jury Theorem, if each classifier votes with a probability $p > \frac{1}{2}$ then the final ensemble model's chance to reach a correct decision by majority vote approaches 1.

Table 7 proofs that when the no. of voters was 2, the accuracy was 98.38% for the 5-class, and as we increased the no. of voters, the accuracy also increased. According to the Jury theorem, when a voter with a high probability is added, the chances of getting the decision right by a majority vote increase. So, from Table 7, we can observe that when the ResNet152V2 is added to the group of voters, the accuracy of the 5-class increases by a margin of 0.94%. But when ResNet152V2 is added to the voter's group, the accuracy of the 3-class decreases because a voter with low probability is added. Hence, from Fig. 14 and Table 7, it has been proved that in neural networks application, Ensembling N no. of DCNN classifiers based on Condorcet's Jury Theorem, the accuracy increases as the no. of DCNN classifiers with a high probability of getting a decision correct increases.

From Figs. 15, 17 and 19, it can be observed that the models are mostly making errors on 2nd and 4th classes only i.e., lung_aca and lung_scc. The reason behind it is that the images of lung_aca and lung_scc look very similar, and it can be seen in Fig. 1. In Fig. 15, as we can see, the deep stacking ensemble-based model misclassifies 11 lung_aca images as lung_scc. But when the weights are assigned to the base learners, the meta learner makes fewer mistakes, misclassifying only 4 images. This error is further reduced when Condorcet's Jury theorem-based majority voting is considered. In the jury-based ensemble model, the false alarm rate is very low; only 3 images are miss classified as shown in Fig. 19. Also in Figs. 16, 18 and 20, the ROC plots of Deep Stacking based ensemble model, weighted ensemble model and Jury based ensemble model is shown.

4.5. Comparative analysis

4.5.1. Based on accuracy

Lung and Colon Cancer detection from LC25000 Histopathological Image Dataset has been the subject of a lot of research.

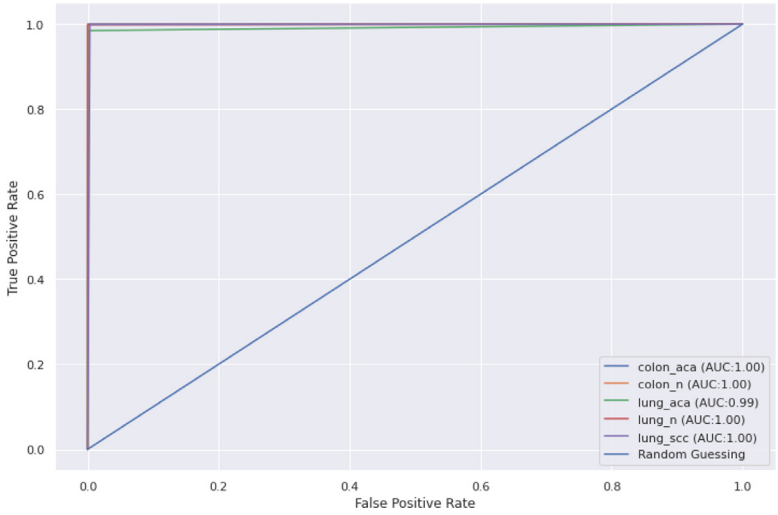

Fig. 16. Multi-labeled ROC of the proposed Deep Stacking based ensemble method.

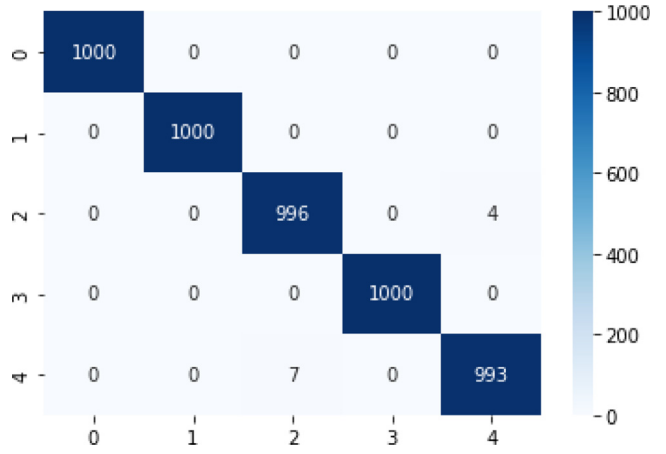

Fig. 17. Confusion matrix of the proposed Differential Evolution based optimization ensemble method.

Several authors have implemented and claimed the performance of their models and approaches on this dataset. Some only used the lung dataset to accomplish the 3-class classification colon dataset to do binary classification, whereas some used the combined dataset of lung and colon to perform 5-class classification as well. As a result, direct comparisons between our model and those studies are difficult. Therefore, the authors ran all 2-class, 3-class, and 5-class classifications on this dataset to compare our model to previous research. As demonstrated in Table 8, our proposed techniques in this research outperform several state-of-the-art models and approaches used in previous studies.

4.5.2. Based on efficiency

Convolutional neural networks have recently shown outstanding results in various computer vision applications. But due to the high computational cost of CNN models, it is crucial to choose the models wisely based on both accuracy and efficiency. The training time is calculated by multiplying the training time per epoch by the number of epochs required to achieve the specified degree of accuracy.

Table 7
Accuracy of Condorcet's Jury theorem with ensembling N no. of classifiers.

No. of classifiers	Classifiers	2-class	3-class	5-class
2	InceptionResNetV2 Xception	99.90	98.70	98.38
3	InceptionResNetV2 Xception VGG16	99.90	99.56	98.52
4	InceptionResNetV2 Xception VGG16 ResNet152V2	99.90	99.43	99.46
5	InceptionResNetV2 Xception VGG16 ResNet152V2 DenseNet201	99.95	99.77	99.63
6	InceptionResNetV2 Xception VGG16 ResNet152V2 DenseNet201 EfficientNetB2	**99.95**	**99.90**	**99.88**

If $X_i, i = 1, \ldots, N$ is the training time taken for N no. of classifiers and Y_i is the training time for ensemble model, then total time taken for training ensemble model based on Deep Stacking Approach is shown in Eq. (41).

$$\sum_{i=0}^{N}(X_i) + Y_i \tag{41}$$

If $X_i, i = 1, \ldots, N$ is the training time taken for N no. of classifiers and β is the time required for optimizing weights through differential evolution algorithm then the total training time taken for final ensemble model is calculated as Eq. (42).

$$\sum_{i=0}^{N}(X_i) + \beta \tag{42}$$

If $X_i, i = 1, \ldots, N$ is the training time taken for N no. of classifiers and α is the training time for calculating the final score

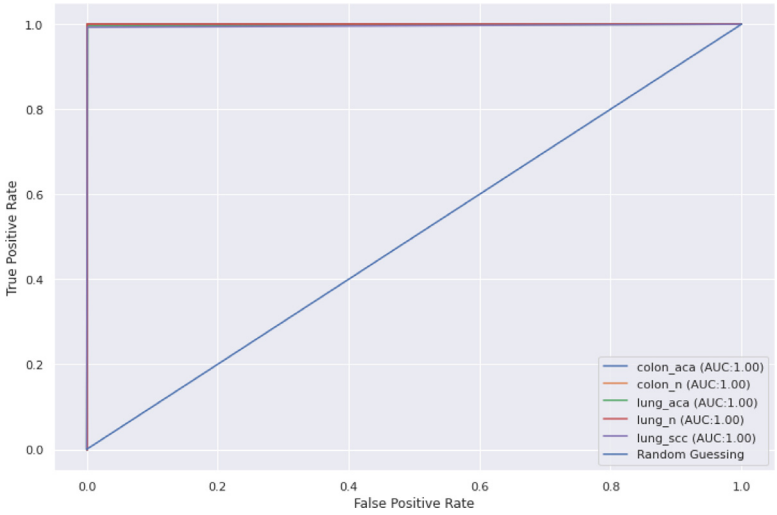

Fig. 18. Multi-labeled ROC of the proposed Differential Evolution based optimization ensemble method.

Table 8
Comparative study between the proposed methods and the existing methods/models.

Papers	2-class accuracy	3-class accuracy	5-class accuracy
Masud et al. [21]	–	–	96.33%
Adu et al. [22]	–	–	99.23%
Mangal et al. [23]	>96%	>97%	–
Kumar et al. [24]	–	–	98.6%
Mesut Togacar [25]	–	–	99.69%
Muhammed Yildirim, Ahmet Cinar [31]	99.75%	–	–
Nishio et al. [32]	–	99.43%	–
Garg et al. [33]	98.50%	97.50%	–
Talukder et al. [26]	99.05%	100%	99.30%
Li et al. [27]	–	–	98.96%
Lin et al. [28]	–	–	96.49%
Fan et al. [29]	–	–	99.44%
Mehmood et al. [30]	–	–	98.40%
Deep Stacking based ensemble model	**99.90%**	**99.80%**	**99.64%**
Optimized Weights ensemble model	**99.92%**	**99.83%**	**99.78%**
Jury based ensemble model	**99.95%**	**99.90%**	**99.88%**

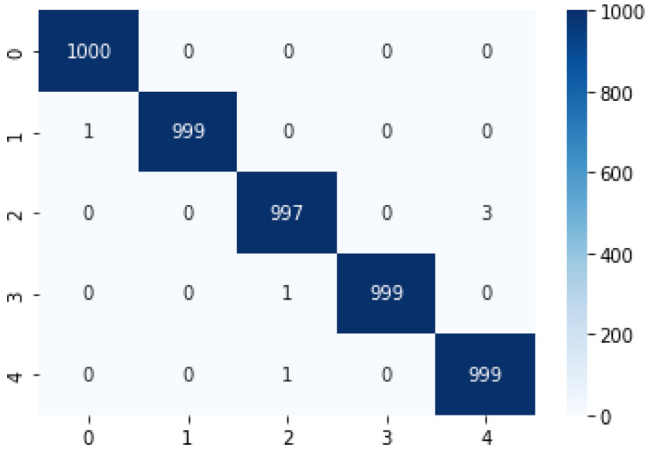

Fig. 19. Confusion matrix of the proposed Condorcet's Jury Theorem based ensemble method.

based on Condorcet's Jury Theorem then the total training time taken for final ensemble model is calculated as Eq. (43).

$$\sum_{i=0}^{N}(X_i) + \alpha \tag{43}$$

where α is negligible so we can ignore it and the total training time can be considered as shown in Eq. (44).

$$\sum_{i=0}^{N}(X_i) + \alpha \approx \sum_{i=0}^{N}(X_i) \tag{44}$$

5. Conclusion

Cancer is a life-threatening disease affecting millions of people worldwide, with Lung and Colorectal cancer being the most deadly, causing the most cancer-related deaths among other types. This calls for the need to study the efficient detection of these cancerous cells using artificial intelligence to assist hospitals and patients with feasible treatment. Image preprocessing techniques were performed on the LC25000 dataset for denoising and enhancing the features before extracting them using Deep CNN models to apply ensemble learning methods. In this research, the authors proposed to use a metaheuristic algorithm — Differential Evolution to optimizing the weights for better performance of the ensemble technique, resulting in a classification accuracy of 99.78% for 5-class. As the computational cost of this method is relatively high, the authors proposed a novel ensemble approach using majority voting based on Condorcet's Jury theorem. Besides, the authors also proved the theorem validity for the 'N' number of classifiers in the case of neural networks. Condorcet's Jury Theorem-based ensemble model shows an accuracy of 99.88% for 5-class classification.

The dataset used in this research is very well balanced and homogeneous, which is unlikely in the real world. In the future, the authors will also aim to test and study the performance of the proposed approaches on other inconsistent datasets.

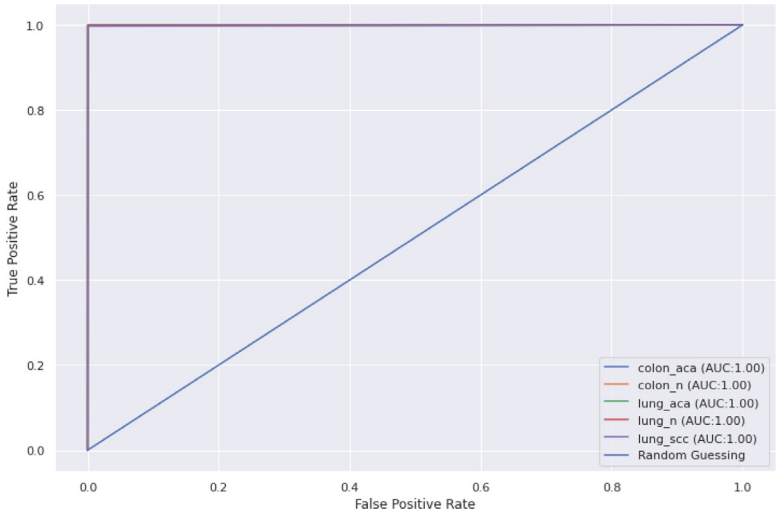

Fig. 20. Multi-labeled ROC of the proposed Condorcet's Jury Theorem based ensemble method.

CRediT authorship contribution statement

Gaurav Srivastava: Conceptualization, Methodology, Software, Code implementations, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Aninditaa Chauhan:** Literature survey, Data curation, Investigation, Writing – original draft, Writing – review & editing. **Nitesh Pradhan:** Validation, Resources, Investigation, Writing – review & editing, Supervision, Research administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] H. Sung, J. Ferlay, R.L. Siegel, M. Laversanne, I. Soerjomataram, A. Jemal, F. Bray, Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries, CA: Cancer J. Clin. 71 (3) (2021) 209–249.
[2] W.H. Organization, Cancer, 0000. https://www.who.int/news-room/fact-sheets/detail/cancer.
[3] W.H. Organization, Cancer Today, https://gco.iarc.fr/today/.
[4] P.S. Malik, V. Raina, Lung cancer: Prevalent trends & emerging concepts, Indian J. Med. Res. 141 (1) (2015) 5.
[5] P. Rawla, T. Sunkara, A. Barsouk, Epidemiology of colorectal cancer: incidence, mortality, survival, and risk factors, Prz. Gastroenterol. 14 (2) (2019) 89.
[6] K. Kurishima, K. Miyazaki, H. Watanabe, T. Shiozawa, H. Ishikawa, H. Satoh, N. Hizawa, Lung cancer patients with synchronous colon cancer, Mol. Clin. Oncol. 8 (1) (2018) 137–140.
[7] J. Li, Y. Yuan, F. Yang, Y. Wang, X. Zhu, Z. Wang, S. Zheng, D. Wan, J. He, J. Wang, et al., Expert consensus on multidisciplinary therapy of colorectal cancer with lung metastases (2019 edition), J. Hematol. Oncol. 12 (1) (2019) 1–11.
[8] M. Kohli, L.M. Prevedello, R.W. Filice, J.R. Geis, Implementing machine learning in radiology practice and research, Am. J. Roentgenol. 208 (4) (2017) 754–760.
[9] W.L. Bi, A. Hosny, M.B. Schabath, M.L. Giger, N.J. Birkbak, A. Mehrtash, T. Allison, O. Arnaout, C. Abbosh, I.F. Dunn, et al., Artificial intelligence in cancer imaging: clinical challenges and applications, CA: Cancer J. Clin. 69 (2) (2019) 127–157.
[10] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G.B. Kim, J.B. Seo, N. Kim, Deep learning in medical imaging: general overview, Korean J. Radiol. 18 (4) (2017) 570–584.
[11] N. Coudray, P.S. Ocampo, T. Sakellaropoulos, N. Narula, M. Snuderl, D. Fenyö, A.L. Moreira, N. Razavian, A. Tsirigos, Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning, Nat. Med. 24 (10) (2018) 1559–1567.
[12] K. Kourou, T.P. Exarchos, K.P. Exarchos, M.V. Karamouzis, D.I. Fotiadis, Machine learning applications in cancer prognosis and prediction, Comput. Struct. Biotechnol. J. 13 (2015) 8–17.
[13] J.F. Mccarthy, K.A. Marx, P.E. Hoffman, A.G. Gee, P. O'neil, M.L. Ujwal, J. Hotchkiss, Applications of machine learning and high-dimensional visualization in cancer detection, diagnosis, and management, Ann. New York Acad. Sci. 1020 (1) (2004) 239–262.
[14] M.I. Razzak, S. Naz, A. Zaib, Deep learning for medical image processing: Overview, challenges and the future, Classif. BioApps (2018) 323–350.
[15] A.A. Borkowski, M.M. Bui, L.B. Thomas, C.P. Wilson, L.A. DeLand, S.M. Mastorides, Lung and colon cancer histopathological image dataset (lc25000), 2019, arXiv preprint arXiv:1912.12142.
[16] S. Salcedo-Sanz, J. Del Ser, I. Landa-Torres, S. Gil-López, J. Portilla-Figueras, The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems, Sci. World J. 2014 (2014).
[17] B. Chopard, M. Tomassini, Particle swarm optimization, in: An Introduction to Metaheuristics for Optimization, Springer, 2018, pp. 97–102.
[18] H.R. Maier, S. Razavi, Z. Kapelan, L.S. Matott, J. Kasprzyk, B.A. Tolson, Introductory overview: Optimization using evolutionary algorithms and other metaheuristics, Environ. Model. Softw. 114 (2019) 195–213.
[19] O. Bozorg-Haddad, M. Solgi, H.A. Loáiciga, Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization, John Wiley & Sons, 2017.
[20] H.R. Maier, Z. Kapelan, J. Kasprzyk, J. Kollat, L.S. Matott, M.C. Cunha, G.C. Dandy, M.S. Gibbs, E. Keedwell, A. Marchi, et al., Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions, Environ. Model. Softw. 62 (2014) 271–299.
[21] A machine learning approach to dia/gnosing lung and colon cancer using a deep learning-based classification framework, 21, (3), (2021) 748.
[22] K. Adu, Y. Yu, J. Cai, K. Owusu-Agyemang, B.A. Twumasi, X. Wang, DHS-CapsNet: Dual horizontal squash capsule networks for lung and colon cancer classification from whole slide histopathological images, Int. J. Imaging Syst. Technol. 31 (4) (2021) 2075–2092.
[23] S. Mangal, A. Chaurasia, A. Khajanchi, Convolution neural networks for diagnosing colon and lung cancer histopathological images, 2020, arXiv preprint arXiv:2009.03878.
[24] N. Kumar, M. Sharma, V.P. Singh, C. Madan, S. Mehandia, An empirical study of handcrafted and dense feature extraction techniques for lung and colon cancer classification from histopathological images, Biomed. Signal Process. Control 75 (2022) 103596.
[25] M. Toğaçar, Disease type detection in lung and colon cancer images using the complement approach of inefficient sets, Comput. Biol. Med. 137 (2021) 104827.
[26] M.A. Talukder, M.M. Islam, M.A. Uddin, A. Akhter, K.F. Hasan, M.A. Moni, Machine learning-based lung and colon cancer detection using deep feature extraction and ensemble learning, Expert Syst. Appl. (2022) 117695.

[27] G. Li, G. Wu, G. Xu, C. Li, Z. Zhu, Y. Ye, H. Zhang, Pathological image classification via embedded fusion mutual learning, Biomed. Signal Process. Control 79 (2023) 104181.
[28] J. Lin, G. Han, X. Pan, Z. Liu, H. Chen, D. Li, X. Jia, Z. Shi, Z. Wang, Y. Cui, et al., PDBL: Improving histopathological tissue classification with plug-and-play pyramidal deep-broad learning, IEEE Trans. Med. Imaging 41 (9) (2022) 2252–2262.
[29] J. Fan, J. Lee, Y. Lee, A transfer learning architecture based on a support vector machine for histopathology image classification, Appl. Sci. 11 (14) (2021) 6380.
[30] S. Mehmood, T.M. Ghazal, M.A. Khan, M. Zubair, M.T. Naseem, T. Faiz, M. Ahmad, Malignancy detection in lung and colon histopathology images using transfer learning with class selective image processing, IEEE Access 10 (2022) 25657–25668.
[31] M. Yildirim, A. Cinar, Classification with respect to colon adenocarcinoma and colon benign tissue of colon histopathological images with a new CNN model: MA_ColonNET, Int. J. Imaging Syst. Technol. 32 (1) (2022) 155–162.
[32] M. Nishio, M. Nishio, N. Jimbo, K. Nakane, Homology-based image processing for automatic classification of histopathological images of lung tissue, Cancers 13 (6) (2021) 1192.
[33] S. Garg, S. Garg, Prediction of lung and colon cancer through analysis of histopathological images by utilizing pre-trained CNN models with visualization of class activation and saliency maps, in: 2020 3rd Artificial Intelligence and Cloud Computing Conference, 2020, pp. 38–45.
[34] D.S. Paik, C.F. Beaulieu, G.D. Rubin, B. Acar, R.B. Jeffrey, J. Yee, J. Dey, S. Napel, Surface normal overlap: a computer-aided detection algorithm with application to colonic polyps and lung nodules in helical CT, IEEE Trans. Med. Imaging 23 (6) (2004) 661–675.
[35] I. Khan, Z. Luo, A.K. Shaikh, R. Hedjam, Ensemble clustering using extended fuzzy k-means for cancer data analysis, Expert Syst. Appl. 172 (2021) 114622.
[36] A. Buades, B. Coll, J.-M. Morel, A review of image denoising algorithms, with a new one, Multiscale Model. Simul. 4 (2) (2005) 490–530.
[37] A. Buades, B. Coll, J.-M. Morel, Non-local means denoising, Image Process. Line 1 (2011) 208–212.
[38] Y.-L. Liu, J. Wang, X. Chen, Y.-W. Guo, Q.-S. Peng, A robust and fast non-local means algorithm for image denoising, J. Comput. Sci. Tech. 23 (2) (2008) 270–279.
[39] A. Dauwe, B. Goossens, H.Q. Luong, W. Philips, A fast non-local image denoising algorithm, in: Image Processing: Algorithms and Systems VI, Vol. 6812, SPIE, 2008, pp. 324–331.
[40] A.M. Reza, Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement, J. VLSI Signal Process. Syst. Signal Image Video Technol. 38 (1) (2004) 35–44.
[41] L.J. Guo, Balance contrast enhancement technique and its application in image colour composition, Remote Sens. 12 (10) (1991) 2133–2151.
[42] L.-K. Huang, M.-J.J. Wang, Image thresholding by minimizing the measures of fuzziness, Pattern Recognit. 28 (1) (1995) 41–51.
[43] J. Yousefi, Image Binarization Using Otsu Thresholding Algorithm, University of Guelph, Ontario, Canada, 2011.
[44] O. Nina, B. Morse, W. Barrett, A recursive Otsu thresholding method for scanned document binarization, in: 2011 IEEE Workshop on Applications of Computer Vision (WACV), IEEE, 2011, pp. 307–314.
[45] X. Xu, S. Xu, L. Jin, E. Song, Characteristic analysis of Otsu threshold and its applications, Pattern Recognit. Lett. 32 (7) (2011) 956–961.
[46] X. Dong, Z. Yu, W. Cao, Y. Shi, Q. Ma, A survey on ensemble learning, Front. Comput. Sci. 14 (2) (2020) 241–258.
[47] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239.
[48] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 8 (4) (2018) e1249.
[49] T.G. Dietterich, Ensemble methods in machine learning, in: International Workshop on Multiple Classifier Systems, Springer, 2000, pp. 1–15.
[50] J. Theorems, Stanford Encyclopedia of Philosophy, https://plato.stanford.edu/entries/jury-theorems/.
[51] E. Then, Lecture: Condorcet's Theorem.
[52] G. Srivastava, N. Pradhan, Y. Saini, Ensemble of deep neural networks based on condorcet's jury theorem for screening Covid-19 and pneumonia from radiograph images, Comput. Biol. Med. 149 (2022) 105979.
[53] D.M. Estlund, Opinion leaders, independence, and Condorcet's jury theorem, Theory and Decision 36 (2) (1994) 131–162.
[54] H. Shteingart, E. Marom, I. Itkin, G. Shabat, M. Kolomenkin, M. Salhov, L. Katzir, Majority voting and the Condorcet's Jury Theorem, 2020, arXiv preprint arXiv:2002.03153.
[55] O. Kallenberg, O. Kallenberg, Foundations of Modern Probability, Vol. 2, Springer, 1997.
[56] C.J. Theorem, Statistical theory, socio-political issues, democracy, 0000. https://www.statisticalconsultants.co.nz/blog/condorcets-jury-theorem.html.
[57] C.J. Theorem, Wolfram mathworld, 0000. https://mathworld.wolfram.com/CondorcetsJuryTheorem.html.
[58] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2010) 4–31.
[59] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[60] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions, 2017, arXiv preprint arXiv:1710.05941.
[61] S. Lau, Learning rate schedules and adaptive learning rate methods for deep learning, 2017, https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1.
[62] Keras, Reducelronplatea, 2020, https://keras.io/api/callbacks/reduce_lr_on_plateau/f.
[63] B. Chen, A practical introduction to keras callbacks in TensorFlow 2, 2020, https://towardsdatascience.com/a-practical-introduction-to-keras-callbacks-in-tensorflow-2-705d0c584966.
[64] A. Manna, R. Kundu, D. Kaplun, A. Sinitca, R. Sarkar, A fuzzy rank-based ensemble of CNN models for classification of cervical cytology, Sci. Rep. 11 (1) (2021) 1–18.
[65] G. Srivastava, A. Chauhan, M. Jangid, S. Chaurasia, CoviXNet: A novel and efficient deep learning model for detection of COVID-19 using chest X-Ray images, Biomed. Signal Process. Control (2022) 103848.
[66] G. Srivastava, A. Chauhan, M. Jangid, A. Jain, An analysis of deep learning models to diagnose COVID-19 using radiography images, in: 2022 International Conference for Advancement in Technology (ICONAT), IEEE, 2022, pp. 1–7.