A New Convolution Neural Layer Based on Weights Constraints

Abstract. With the rapid development of artificial intelligence (AI) techniques, deep learning is one of the most effective ways to solve a variety of tasks. Convolution neural networks (CNNs) are considered one of the most popular AI techniques used to extract and analyze meaningful features for image datasets, especially in the medical diagnosis field. In this paper, a new convolution layer for the CNN model is proposed. The new layer uses a constrained number of weights in each kernel trained in the phase of learning and excludes the others weights with zero values. The proposed method is introduced to extract a special type of feature considering for local shape of a sub-image (window) and topological relations between group pixels. The features extract according to the distribution of weights in kernels that are determined considering a particular desired percentage. Furthermore, this paper proposed CNN model architecture uses the new convolution layer rather than the traditional CNN layers. The efficiency of the method is evaluated using three types of medical image datasets compared with the traditional convolution layer, pre-trained deep neural networks (DNNs), and state-of-art methods. The proposed method outperforms other methods in terms of accuracy and F1 score metrics and exceeds more than 98%, 89%, and 93% for three datasets used in the evaluation.
Keywords: Convolution layer, 2D CNN, COVID-19, Accuracy, F1 score.

1. Introduction

In recent years, deep neural networks (DNNs) are considered one of the significant and developed artificial intelligence (AI) techniques used in many applications [1]. Convolution neural networks (CNNs) are a special type of DNN widely used in several methods like pattern recognition, classification, image retrieval, and many computer vision techniques [2]. The power of using the convolution layer in CNN networks is extracting the data features from the local features as texture features in earlier layers to global features of data as the object as a whole [3]. Essentially, the traditional convolution layer works in the multilayer perceptron neural network (MLP) principle and receptive field theory and views the features of images as multilevel hierarchies [4]. Typically, CNN consists of three types of layers in terms of levels; input layer, hidden layers, and output layer. The convolution process, performed in the hidden layer, includes the dot product of kernels with the feature maps in that layer considering the receptive field [5]. Hidden layers contain other types of layers as max pooling and average pooling layers. Each convolution layer may be followed by max or average pooling layers [6].

As a principle, max-pooling selects the maximum value of each local sub-image in the feature map and average pooling takes its average value [7]. Dense layers as batch normalization are of the most popular types of layers used in CNN networks. The dense layer or (full-connected layer) has the same work as backpropagation neural network layers and is used in the later CNN layers. A flattening process is needed for the resulted dense layer feature maps to predict the classes of images [8].

To increase the efficiency and accuracy of the 2D CNN model used in this work, there are some preprocessing steps are implemented on the medical images datasets; the image enhancement method and data augmentation are used. Contrast Limited Adaptive Histogram Equalization (CLAHE) is one of the most efficient indirect contrast image enhancement methods used for medical image preprocessing [9]. As the name indicates, the CLAHE method is based on an image histogram and some statistical steps. The image is divided into several non-overlap sub-regions and clips the sub regions histogram to a specific range then equalized the resulted histograms [10]. The enhancement of the output image is appearing equally on image details rather than the image background [11].

Data augmentation is one of preprocessing methods that is used to increase the training part of the dataset and raise the efficiency of the CNN model performance [12]. The augmentation process includes flipping horizontally or vertically, scaling (zoom in/ zoom out), rotation at a small angle, clipping, and others [13].

In this paper, a new convolution layer is proposed to extract more efficient features with less trainable parameters considering the local shape and topological pixel relations. The idea of the new layer is motivated by the shape of the image processing filters used for a special task. The filters are specified with values in specific locations in it according to the task applied for it.

In the new convolution layer, each kernel is initialized with random weights (trainable weights) in specific locations selected randomly with the desired percentage specified by the user. The rest of the weights (non-trainable weights) are initialized with zero and not trained or changed during the CNN model training phase. The distribution of weights in each kernel is diverse from other kernels; therefore, it extracts a special local shape localized in that receptive field of feature maps. Furthermore, the distribution of the weights in a specific structure in each kernel helps to extract topological relations between pixels. Topological relations give an impression about image pixels organized and their regularity structure, as well as if there are a group of pixels that share relations or features and have the same properties.

The new method considers a less number of trainable parameters considering for range determined by the user. That means, a faster CNN model and get generalization with a smaller number of trainable weights.

There are many contributions are presented in this paper and can summarize as follow:

- Propose a new 2D CNN layer with a new distribution for weights in each convolution filter. The proposed method includes a new suggestion for the filters' form in the convolution layer. The weights in these filters concentrate the local shape and topological relations of the sub-image (window). The proposed convolution layer uses introduces a new work for the convolution process and uses a fewer number of trainable parameters compared with the traditional convolution layer.

- Proposing a new 2D CNN model architecture with preprocessing steps (enhancement and data augmentation) to increase the meaningful and efficiency of medical images feature extraction process and therefore the classification process.
- The work uses three various datasets type that considers more recent challenges COVID19 medical CT and X-Ray images and proves the efficiency of the method by achieving outperformance results compared with traditional CNN layer model, pre-trained DNNs as well as state-of-art methods in terms of accuracy and F1 score metrics.

In the rest of the paper, sections are organized as; section 2 includes the works and research that introduced an improving trend in CNN models and layers. Section 3 contains the proposed convolution layer and the CNN model architecture that includes the new layer rather than the traditional CNN layers for classification purposes. Section 4 introduces the experiment results and discussions that analyze the efficiency of the new CNN model using three different medical image datasets. Finally, conclusions are explained in section 5.

2. Related work

There are many works and researches introduced in context improving a CNN layer or suggesting CNN model architecture. Ephy R. Love et al. introduced a topology CNN layer that used manifold relationships metrics to parameterize the kernels' weights values. They used two types of manifold metrics; first localized the weights concerning the location of the circle in addition to using the traditional CNN locality metric. Second, they localized layer kernels weights for the Klein bottle manifold metric. They set kernel weights to zero if the slices' layer weights are greater than the threshold to the two manifolds metrics and initialize the rest layers' kernels weights randomly.

The method is proven to reduce the number of weights, but in only specific two forms of topological relations and that is constrained by the produced features in a limited form direction [14].

Belhassen Bayar et al. introduced a new type of CNN called constrained convolutional layer for image manipulation detection for forensic tasks. They adaptively learn the kernel weights as image manipulation traces. The constraint used in their approach included set -1 to the weight located in the center of the kernel and the sum of the rest weights is equal to 1. This constraint is applied during training and the error computing is constrained with the constrained kernels in this layer. This layer made the CNN model tend to do a specific task, but the number of training weights is stilled not changed [15].

Juan P. et al. introduced a new CNN layer for regularization purposes. The layer applied a random rotation process with a small rate on some feature maps after applying the convolution process. They rotated the feature map at an oriented rate selected randomly and this was done during the training phase without increasing the size and number of rotated feature maps [16].

Seunmin Han and Jongpil Jeong adopted weighted arithmetic mean CNN. They divided the training dataset into a set of traditional CNN layers with a percentage of 1:3. Then they found the optimal weights by increasing 2 each time. Finally, they found the weighted mean function on the resulted CNN layers feature maps [17].

Wahyudi Setiawan and Fitri Damayanti modified a CNN model with 35 layers for chest pneumonia disease detection. They built the model with 3×3 kernel size and convolution layers with 8, 16, 32, 64, 64, 128, 256, and 512 dimensions. The model performance was compared with pre-trained CNN models such as VGG16 and VGG19 networks [18].

Hajer Fradi et al. introduced multi-layer CNNs fusion by aggregating the activations of CNN produced by different CNN layers and fusing it into a single vector of features after passing into pooling layers. The method included a feature selection process by reducing the irrelevant features as well as the final dimension. To build the CNN model architecture, they used different inception architectures that contained from few convolution layers followed by max-pooling layers. The outputs of inception parts are aggregated using global average-pooling, and finally, passed to the full connected layer. To reduce the dimension of the feature, they used Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to find texture patterns projected to the same samples [19].

This work introduces a new CNN layer with a new distribution for weights in kernels considering the desired percentage to extract the local shape and topological relations in each sub-image (window). The proposed method reduces the number of CNN model parameters with the desired percentage, furthermore, improves the efficiency and performance of the CNN model.

3. Proposed method and framework

This paper introduces an improvement in CNN work. Furthermore, this paper proposed a new CNN architecture including the new proposed convolution layer, rather than, the traditional CNNs layers. The CNN model is built for classification tasks preceded by preprocessing steps to increase the model performance. This section introduces a description and explanation of the new CNN layer as well as the new CNN model architecture applied to classification task.

3.1 The new proposed CNN layer

The convolution layer is the base construction block in the deep neural network architecture. The convolution process includes a dot product between a sub-image matrix (window) also called the receptive field and another matrix containing values that represent the trainable parameters called kernel or filter. To explain the work of proposed CNN, there is need to brief explanation how the traditional CNN parameters is computed. For each convolution layer, the input of the layer is the image channels (in the first convolution layer) or feature maps (the output of the previous layer). The size of input feature maps is X×X×A, where X×X and A are the size and number of feature maps, respectively. In the traditional convolution layer, each kernel is a square matrix have a size with k×k dimensions (e.g. k= 3,5,7,…) with depth equal to A. That means each kernel has k×k×A trainable parameters. Also, number of kernels specified to the layer that is referred as F is also represents number of feature maps produced from this layer (the layer output). Thus, the output of the layer will be X×X×F and number of trainable parameters (weights) is k×k×A×F [32].

The proposed convolution layer suggests diverse distribution for weights of kernels in each convolution layer rather than the traditional square distribution. Figure (3.13) illustrates the difference between the traditional and proposed convolution layer work for one feature map convolved with N kernels each has 3×3 size (to simplify the

example comprehend we use 2D kernels and one 2D feature map). The main idea of proposed layer is determining the percentage of trainable parameters (weights) distribution within each k×k kernel for depth A in the convolution layer and the rest non-trainable parameters set to zeros values. The trainable parameters form specific shapes and topological relations to extract the local features in these shapes. Furthermore, the distribution of the trainable parameters in specific (random) locations in each kernel with a particular percentage will determine only the useful and meaningful local features related to those shapes and topological relations. In other words, the method specifies the weights (trainable parameters) in each kernel with a particular percentage in some locations and allocated zero values to weights (non-trainable parameters) in rest locations. Thus, this method reduces the number of trainable parameters in kernels.
Fig. 2 illustrates examples of kernel shapes and their appropriate parts on some sub-images that it is useful to extract features from it. As explained in Fig. 2, each kernel in the proposed layer contains a particular shape represented by weights considering a specific percentage. To more explain the understanding of the new convolution layer, the initialization, forward and backward training phases are explained for the proposed layer in the following:

- **Kernels initialization:** in traditional convolution layer, the weights in kernels are initialized randomly using uniform distribution using the Eq. 1 [33]:

$$W_m^l = Uni[-\frac{1}{\sqrt{k}}\ , \frac{1}{\sqrt{k}}] \quad (1)$$

Where, W_m^l is kernel *m* in layer *l*, Uni[.] is the uniform distribution function to generate random values in range [-a, a], and k is the size of kernel.
In the proposed convolution layer, the initial values of kernel weights are defined as Eq. 2:

$$W_{f(i,j,a)}^l = \begin{Bmatrix} Uni\left[-\frac{1}{\sqrt{k}}\ , \frac{1}{\sqrt{k}}\right] \ and\ set\ trainable\ property = 1\ ,\quad if\ loc_{f(i,j,a)}^l = 1 \\ 0 \quad and\ set\ trainable\ property = 0\ ,\quad if\ loc_{f(i,j,a)}^l = 0 \end{Bmatrix} \quad (2)$$

Where, $W_{f(i,j,a)}^l$ is weight with (i, j, a) index in kernel number f ($f \in F$) with size k×k×A in layer *l*, *trainable property* is property for each weight in kernel give trainable ability to the weights, $loc_{f(i,j,a)}^l$ is index of one location in loc matrix that contains 0 or 1 random values are generated using a random function with particular percentage for appearing 1 in it. *loc* matrix has the same dimension as W matrix. Eq. (3) explains generation of each k×k loc matrix in depth A for F kernels.

$$loc = Genrate\_0\_1\_random\ (k \times k\ , \text{per}) \quad (3)$$

Where *Genrate_0_1_random (.)* is random generation function to generate random values [0 or 1]. *per* is the percentage of appearance of 1 in the random generation function. *loc* matrix with 0/1 values forms a specific shape of weights in kernel instead of square shape where 1's values represent the trainable weights and 0 values represent non-trainable weights in kernels.
In this form of kernels, the convolution operation will take the effect of sub-image positions that is convolved with actual weights and get a zero effect on the zeros weights values in each kernel.

- Forward phase: the convolution process for traditional convolution layer is defined as Eq. 4 [34]:

$$o_m^l = \sigma((W_m^l * a^l) + b_m^l) \quad (4)$$

Where o_m^l is the output of convolution operation (*) between kernel W_m^l number m ($m \in F$) and input feature maps a^l of the layer l, b_m^l is the bias parameter, σ is a non-linear activation function.

For proposed convolution layer, the convolution process for convolution layer is stay the same, where the data in locations multiplying with the zero weights is neglected (not added in convolution process).

- **Backward phase:** In traditional convolution layer, the weights in kernel are affected by the loss function of computing the error values and updating of weights in the backward phase. Eq. 5 represents the weights kernel updating function using gradient descent error function that compute the error [35]:

$$W_m^{*\,l} = W_m^l - Lr \times \partial W_m^l \qquad (5)$$

Where, $W_m^{*\,l}$ is the updated weights kernel number m for layer l, W_m^l is old weights kernels, ∂W_m^l is the gradient of error function computed of each weights kernel, Lr is the learning rate.

In proposed convolution layer, since the zeros weights within the kernel have zero error, for that, it is not updated in the backward phase. Thus, the weights in kernel are updated using gradient descent error function as in Eq. 6:

$$W_{f(i,j,a)}^{*l} = \begin{cases} W_{f(i,j,a)}^l - Lr * \partial W_{f(i,j,a)}^l & if\ W_{f(i,j,a)}^l\ \ trainable\ property = 1 \\ W_{f(i,j,a)}^l & if\ W_{f(i,j,a)}^l\ \ trainable\ property = 0 \end{cases} \qquad (6)$$

Where, *(i, j, a)* denoted the index of weight in kernel $f\,(f \in F)$ with size k×k×A of layer *l*.

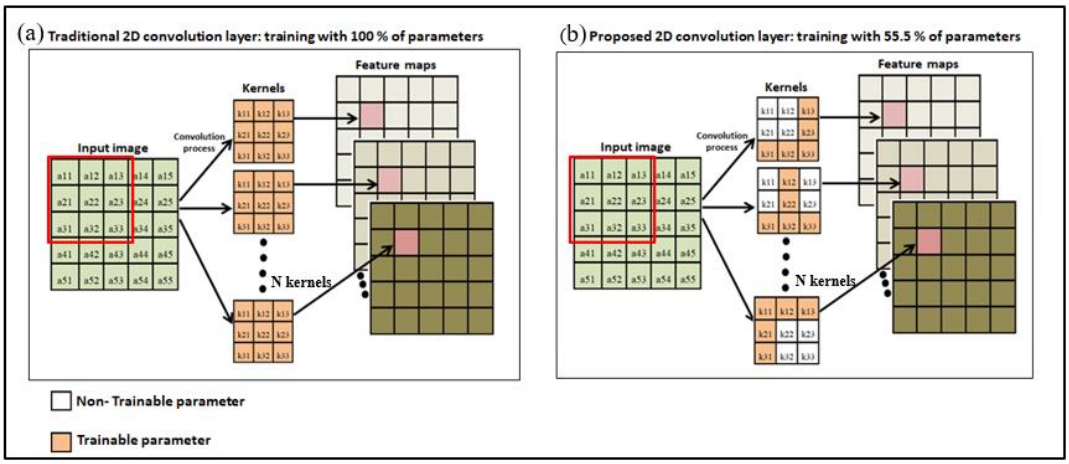

Fig. 1 The convolution layer work; a: traditional and b: proposed convolution layer

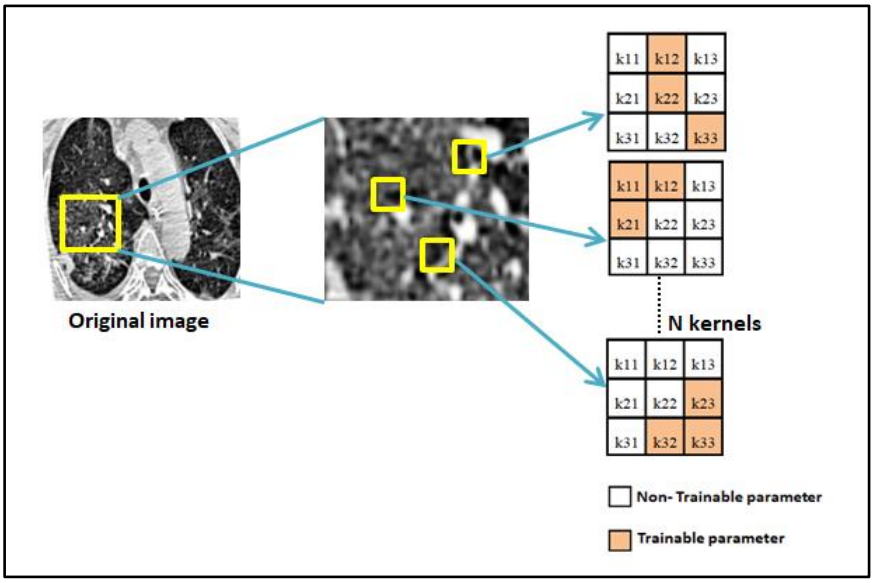

Fig. 2 Examples of kernels shapes and its appropriate parts on some sub images

The pseudocode of the proposed convolution layer is illustrated in the algorithm (1).

Algorithm (1): Initialization weights for proposed convolution layer
Inputs: input feature maps (*A*), output feature maps (*F*), kernels size (*k*) and percentage of trainable weights (*per*)
Output: weights matrix with initial values (*Weight*)
Begin

1. **Set** the dimension of *Weight* and *loc* matrices with [*k, k, A, F*]
2. **Initialize** *Weight* with random uniform values ranged [0-1] using Eq. 1
3. **Initialize** *loc* with random (0 or 1) values with percentage per using Eq. 3
4. **For** f =1 to *F*
   For a =1 to *A*
   For i=1 to *k*
   For j =1 to *k*
   If (*loc* [i, j, a, f] =0) **then**
   Weight [i, j, a, f] =0
   Set *Weight* [i, j, a, f] *trainable property* with 0
   Else
   Set *Weight* [i, j, a, f] *trainable property* with 1
   End if
   End for
   End for
   End for
   End for
5. **Set** *Weight* matrix as new convolution layer weights

End

As explained in algorithm (1), the weights of the layer are selected randomly and in specific indices in each kernel. The weights with the trainable property are updated in the backward phase but the weights with the non-trainable property are not changed and are still with zeros values. As mentioned above, the proposed convolution layer reduces trainable parameters by the desired percentage. To explain the difference between the proposed and traditional convolution layer, we compute the trainable parameters for these two types of layers. In any convolution layer, the trainable parameters are computed as Eq. 7 below:

$$P = k \times k \times A \times F + F \qquad (7)$$

Where P is the number of trainable parameters, and we add F for the bias parameter that used F times. In contrast, the trainable parameters in the proposed convolution layer are computed as Eq. 8 below:

$$P = (k \times k \times per) \times A \times F + F \qquad (8)$$

Where, *per* is the desired percentage of trainable parameter in each $k \times k$ kernel in depth A.

3.2 proposed CNN model architecture

In this section, feature extraction and classification are applied by proposing a new CNN model architecture. The objective of building a CNN model is to prove the efficiency and the power of the new 2D CNN layer that is one of the earlier layers in the proposed model. Fig. 3 illustrates the 2D CNN model architecture with a number of kernels and feature maps dimensions inputs for each layer in the model. Any convolution layer, including the proposed convolution layer, have number of input feature maps A with size X×X and produced output number of feature maps F with size X×X depending on the number of kernels in that layer. In this model, all convolution layers kernels have a size of 3×3, strides are 1 and *same* padding that is zero-padding on left, right, up, and down returning the same size of the feature maps dimension. Max pooling and average pooling layers also are used after convolution layers in the CNN model that is, as well known, considered as a down-sampling for feature maps by reducing its dimensions. The model contains a successive number of convolution and pooling layers followed by flattening and dense fully connected layers. For classification, the model used the Softmax activation function at the end of the architecture.

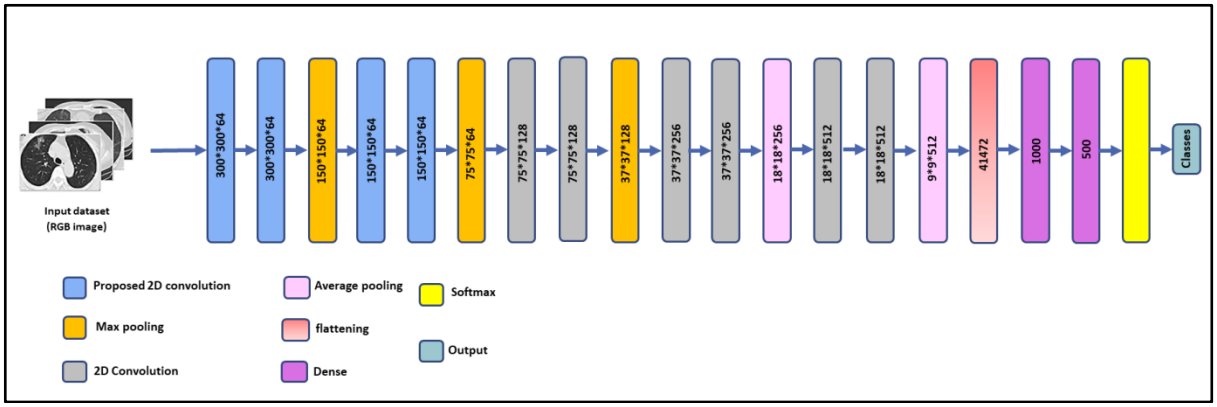

Fig. 3 The proposed 2D CNN model architecture

As explained in Fig. 3, each RGB color image in the medical dataset is passed through two of the proposed convolution layers with 32 kernels and then the max-pooling layer to reduce the feature maps size. Feature maps are passed through two proposed convolution layers with 64 kernels and then to the max-pooling layer too. After that, feature maps passed through a sequence of traditional convolution, max, and average pooling, and dense layers to reach the activation function (Softmax) to the decision making step.

4. Results and discussions

To illustrate the work of the proposed layer and evaluate its efficiency compared with the traditional convolution layer, we build CNN architecture to classify three types of datasets. The three datasets contain a CT scan and X-Ray images for COVID19 disease. The results of the classification process are compared to traditional convolution layer work using the same architecture, as well as, other state-of-art methods and pre-trained DNNs.

4.1 Description of datasets and Data preprocessing

The new layer proposed in this work is evaluated using three types of datasets for COVID19 medical images. Before entering the datasets into the CNN model, there are some preprocessing steps are implemented on the datasets. In this section, we describe applied datasets and preprocessing steps as well as the evaluation metric used in this paper.

4.1.1 Datasets description

The first dataset used in this work is called the SARS-CoV-2 CT scan dataset [20] which consists of 2482 CT scan lung images. The dataset has two classes; 1252 CT scan lung images with COVID19 infection and 1230 CT lung images with other infections. The dataset is available at https://www.kaggle.com/datasets/plamenedu-ardo/sarscov2-ctscan-dataset. The second dataset used in this work is called the COVID-CT dataset [21] which consists of 746 lung CT scan images; 349 infected CT images with COVID19 and 397 with no infection. The dataset is available at http://medicalsegmentation.com/COVID-19/. The third dataset is called DLAI3Hackathon COVID19 Chest X-Ray [22]. This dataset is available at https://www.kaggle.com/c/dlai3/data as a DLAI3 Hackathon COVID19 challenge on the Kaggle website. The dataset consists of 1135 X-Ray Chest images divided into train and test sets. There are 269 X-Ray Chest images with COVID19 infection; 151 for train and 118 for the test. Also, there are 866 X-Ray images for the normal Chest; 576 for the train, and 290 for the test.

4.1.2 Evaluation metrics

There are two evaluation metrics used in this work to measure its efficiency of it; Accuracy and F1 score metrics [23]. Accuracy is defined as a ratio of true positive and negative classification results to the total one that considered as Eq. 9:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (9)$$

Where the TP represents a true positive, TN is a true negative, FP is a false positive, and FN is a false negative of the test validation results. Also, the precision and recall harmonic mean is denoted by the F1 score defined as Eq. 10:

$$F1\ score = \frac{2\times Prec\times Rec}{Prec+Rec} \qquad (10)$$

Where *Prec* represents the precision for the classification result that defined as Eq. 11:

$$Prec = \frac{TP}{TP+FN} \quad (11)$$

And *Rec* represents the recall value for the classification result that defined as Eq. 12:

$$Rec = \frac{TP}{TP+FP} \quad (12)$$

4.1.3 Preprocessing steps

To enhance the CNN model performance and increase the classification accuracy results, image enhancement method is applied on the implemented datasets. As mentioned in section 2, Contrast Limited Adaptive Histogram Equalization (CLAHE) is one of efficient indirect contrast image enhancement methods and is used in this work. Also, we use data augmentation o increase the number of datasets images, furthermore, increasing the efficiency of the classification model. The steps of augmentation are used in this work contain horizontally flipping, rotation in a small angle with 10 range, scaling with parameter 0.01. This step doubles the dataset images number.

4.2 Experiment results and Discussions

The proposed convolution layer is used as one of the CNN model layers within a new classification of CNN architecture as explained in section (3). The CNN model is implemented in PyCharm 2022.1 with Python 3.8 Tensorflow 2 using an GPU 8G RTX 2070 personal PC device. The 2D CNN model is trained using a Stochastic gradient descent (SGD) optimizer, 0.0001 learning rate, 0.9 momentum, and decay rate with learning (rate/epochs) rate. The model trains for 100 epochs with 16 batch sizes.

As we mentioned in section (3.1) and according to Eq.7 and Eq.8, the proposed 2D CNN convolution layer reduces number of trainable parameters (weights) of the model, thus, the training process of the model will be faster. To explain difference in the number of trainable parameters for proposed and traditional convolution layer, we use two models; first is the proposed CNN model that used proposed convolution layers and second is the same model with the same architecture used traditional convolution layers. Table 1 illustrates comparison between number of trainable parameters in earlier convolution layers used in the two models (the rest model layers have the same number of parameters). As shown in table 1, the trainable parameters are reduced considering the desired percentage of the kernel size in the proposed convolution layer compared with the traditional one. Fewer trainable parameters faster learning processes and fewer saved model weights are required.

As mentioned in section 4.1, to evaluate the model efficiency three different types of datasets are used and they also difference in evaluation conditions. For that, and to introduce a fair comparison, we introduce the results and discussions for each dataset and compared it with the state-of-art and pre-trained method as well as the traditional CNN model.

Table 1. Number of trainable parameters for earlier traditional and proposed convolution layers with a specific number of layers parameters using Eq.7 and Eq.8

Convolution layer type	k=3, A=3, F=30, per=0.3	k=3, A=30, F=60, per=0.3
Traditional convolution layer	3×3×3×30+30=840	3×3×30×60+60=16260
Proposed convolution layer	(3×3×0.3)×3×30+30=273	(3×3×0.3)×30×60+60=4920

4.2.1 SARS-CoV-2 CT-scan dataset

The first dataset we used to evaluate our method efficiency is the SARS-CoV-2 CT-scan dataset. First, we compare our proposed 2D CNN model using the proposed convolution layer with the same structure of the model used traditional convolution layer. Also, the model is compared with other pre-trained deep learning methods such as GoogleNet, VGG-16, ResNet, and AlexNet methods. Furthermore, the proposed model is compared with DNN method proposed by Soares et al. [20]. To divide the dataset, the protocol presented in [20] is used to divide the dataset randomly to 80% for train and 20% for test. Images size used in this comparison is 300×300 pixels. Table 2 explains the comparison results in terms of accuracy and F1 score, metrics. As explained in the table 2, the model uses the proposed method and produces results very near to the CNN model using the traditional CNN layers despite the difference in trainable parameters.

Also, the model uses the proposed method outperforms the other pre-trained and state-of-art methods by increasing the accuracy and F1 score metrics up to (1.1%-6.7%) and (1.6%-6.7%), respectively.

Table 2. Proposed method results compared with traditional convolution layer, other pre-trained DNNs and state-of-art approaches applied on SARS-CoV-2 CT-scan Dataset

methods	Accuracy	F1 score
ResNet	94.87%	94.89%
GoogleNet	91.68%	91.79%
VGG-16	94.9%	94.96%
AlexNet	93.69%	93.58%
xDNN [20]	97.38%	97.31%
Our method	98.387%	98.387%
Traditional CNN	98.374%	98.374%

4.2.2 COVID-CT dataset

The second dataset used in this work is the COVID-CT dataset described in section (4.1.1). The results of the model using the proposed method are compared with the same model used traditional convolution layer. Also, we illustrate the results of applying the same dataset using pre-trained DNNs trained on a large dataset as the ImageNet dataset. Pre-trained DNNs are included VGG16 [24], ResNet18 [25], ResNet50 [25], DenseNet-121 [26], DenceNet-169 [26], EffecientNet-b0 [27], and EffiecientNet-b1 [27]. Furthermore, the comparison illustrates the implementation of state-of-art methods applied to the same dataset by Mobiny et al. [28], Polsinelli et al. [29], He et al. [30], and Pedro et al. [31]. The protocol of the division dataset is the same as that used in [28] where the dataset is divided into 85% for train and 15% for test purposes. Table 3 illustrates the experiment results of applying the above methods to the COVID-CT dataset in terms of Accuracy and F1 score metrics.

From observation for the results in a table 3, the results of the proposed CNN layer compared with the traditional CNN layer are so closed. despite the difference in trainable parameters.

Also, whereas the pre-trained DNNs are learned using a big dataset like the ImageNet dataset and these models are more complex and deeper, our model outperforms these pre-trained models. The proposed model has an improvement by increasing in the range of (6.26%-15.26%) and (8.18%-16.18%), in terms of Accuracy and F1 score, respectively, compared with pre-trained DNNs. Table 3 illustrates the performance of our

proposed method compared with state-of-art methods introduced in [24,26,28,30] and get improvement in range of (1.66%-4.7%) and (2.08%-5.2%), in terms of Accuracy and F1 score, respectively.

Table 3 The our proposed method results compared with other pre-trained DNNs and state-of-art approaches applied on COVID-CT dataset

Methods	Accuracy	F1 score
VGG-16 [24]	76%	76%
ResNet-18 [25]	74%	73%
ResNet-50 [25]	80%	81%
DenseNet-121 [26]	79%	79%
DenseNet-169 [26]	83%	81%
EfficientNet-b0 [27]	77%	78%
EfficientNet-b1[27]	79%	79%
Mobiny et al. [28]	87.6%	87.1%
Polsinelli et al. [29]	84.56%	83.98%
He et al. [30]	86%	85%
Pedro et al. [31]	87.6%	86.19%
Our method	89.26%	89.18%
Traditional CNN	88.24%	88.19%

4.2.3 DLAI3 Hackathon Dataset

As described in section (4.1.1), the DLAI3 Hackathon dataset is introduced on the Kaggle website to achieve a challenge called Hackathon COVID-19 Chest X-Ray challenge [22]. The baseline method introduced in this challenge gets an accuracy of 92%. Table 4 illustrates the proposed CNN layer method model results compared with the traditional CNN layer model, baseline method, and pre-trained DNNs including VGG16, AlexNet, VGG19, and Dense-169 networks.

Table 4 Our proposed method results compared with other pre-trained DNNs and state-of-art approaches applied on DLAI3 Hackathon dataset

Methods	Accuracy	F1 score
VGG-16	86.28%	82.96%
VGG-19	78.91%	76.87%
DenseNet-121	90.18%	88.11%
DenseNet-169	90.12%	87.38%
Baseline[22]	92%	-
Our method	93.38%	93.16%
Traditional CNN	93.27%	93.04%

The results are shown in table 4 explains the efficiency of our proposed method that exceeded the pre-trained methods above and the baseline method [22] with a range of increasing (3.21%-14.46%) and (4.98%-16.21%) for accuracy and F1 score, respectively. Also, our proposed method results compared with the traditional CNN layer method gets very closed results in terms of accuracy and F1 score despite the difference in trainable parameters.

Conclusions

This paper introduces a proposed convolution layer for the CNN model considering the number of trainable parameters focusing on extraction of local shape and topological relation features. This method aims to reduce the trainable parameters in the CNN model and thus achieving generalization and faster and more efficient model. The

reduction of learning weights is constricted with the desired percentage determined by the user therefore the number of parameters is reduced by that percentage. Therefore, the comparison results between proposed method and traditional CNN show how close the efficiency of the two methods is, despite the large difference in the number of trainable parameters. Also, this paper proposes a new CNN model architecture that uses the new convolution layer in earlier cascaded layers, in addition to use traditional CNN layers. The proposed model experiment results using three types of medical image datasets showed outperformance of the proposed model compared to pre-trained DNN and state-of-art methods.

References

1. Y. LeCun, Y. Bengio, G. Hinton: Deep Learning. Nature 521(7553), 436-444 (2015).
2. Y. LeCun, K. Kavukcuoglu, C. Farabet: Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, 253–256, IEEE, (2010)
3. Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, Ronen Basri: Actions as space-time shapes. Transactions on Pattern Analysis and Machine Intelligence, 29(12), 2247-2253 (2007)
4. Teow, Matthew YW.: Understanding convolutional neural networks using a minimal model for handwritten digit recognition. In 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS), 167-172. IEEE (2017)
5. L. Deng and D. Yu.: Deep learning: Methods and applications. Foundations and Trends in Signal Processing, 7(3–4). 197–387 (2014)
6. Ciregan, Dan, Ueli Meier, Jürgen Schmidhuber.: Multi-column deep neural networks for image classification. 2012 IEEE conference on computer vision and pattern recognition. 3642-3649, IEEE (2012)
7. Sharma S, Mehra R.: Implications of pooling strategies in convolutional neural networks: A deep insight. Foundations of Computing and Decision Sciences. 44(3), 303-330 (2019)
8. Venkatesan R, Li B.: Convolutional neural networks in visual computing: a concise guide. CRC Press, (2017)
9. Akila, K., L. S. Jayashree, A. Vasuki.: Mammographic image enhancement using indirect contrast enhancement techniques–a comparative study. Procedia Computer Science, 47, 255-261 (2015)
10. Pisano, Etta D., Shuquan Zong, Bradley M. Hemminger, Marla DeLuca, R. Eugene Johnston, Keith Muller, M. Patricia Braeuning, Stephen M. Pizer.: Contrast limited adaptive histogram equalization image processing to improve the detection of simulated spiculations in dense mammograms. Journal of Digital Imaging 11(4), 193-200 (1998)
11. Sivaramakrishna, Radhika, Nancy A. Obuchowski, William A. Chilcote, Gilda Cardenosa, Kimerly A. Powell.: Comparing the performance of mammographic enhancement algorithms: a preference study. American Journal of Roentgenology 175(1), 45-51 (2000)
12. Gu, Shanqing, Manisha Pednekar, Robert Slater.: Improve image classification using data augmentation and neural networks. SMU Data Science Review, 2(2), 1(2019)
13. Wang, Jason, Luis Perez.: The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11, 1-8 (2017)
14. Love, Ephy R., Benjamin Filippenko, Vasileios Maroulas, Gunnar Carlsson.: Topological deep learning. arXiv preprint arXiv:2101.05778 (2021)
15. Bayar, Belhassen, Matthew C. Stamm.: Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. IEEE Transactions on Information Forensics and Security, 13(11), 2691-2706 (2018)
16. Vigueras-Guillén, Juan P., Joan Lasenby, Frank Seeliger.: Rotaflip: A New CNN Layer for Regularization and Rotational Invariance in Medical Images. arXiv preprint arXiv:2108.02704 (2021)

17. Han, Seungmin, Jongpil Jeong.: An weighted cnn ensemble model with small amount of data for bearing fault diagnosis. Procedia Computer Science, 175, 88-95 (2020)
18. Setiawan, Wahyudi, Fitri Damayanti.: Layers modification of convolutional neural network for pneumonia detection. In Journal of Physics: Conference Series, 1477(5), 052055. IOP Publishing, (2020)
19. Fradi, Hajer, Anis Fradi, Jean-Luc Dugelay.: Multi-layer Feature Fusion and Selection from Convolutional Neural Networks for Texture Classification. In VISIGRAPP (4: VISAPP), 574-581 (2021)
20. E. Soares, P. Angelov, S. Biaso, M.H. Froes, D.K. Abe, SARSCoV- 2 CT-scan dataset: a large dataset of real patients CT scans for SARS-CoV-2 identification, medRxiv, 143767 1e8, eprint (2020)
21. X. He, X. Yang, S. Zhang, J. Zhao, Y. Zhang, E. Xing, P. Xie.: Sample-efficient deep learning for covid-19 diagnosis based on ct scans, medRxiv, 1, 1e10 (2020)
22. Jonathan H. Chan, Puttipong Thammachart, Vasin Virasak, Phubeth Rodklang, DLAI3 Hackathon: COVID-19 chest Xray challenge, version 1.0, (2020). https://www.kaggle.com/c/ dlai3/data.
23. Abbood, Elaf Ali, Tawfiq A. Al-Assadi.: GLCMs Based multi-inputs 1D CNN Deep Learning Neural Network for COVID-19 Texture Feature Extraction and Classification. Karbala International Journal of Modern Science, 8(1), 28-39 (2022)
24. K. Simonyan, A. Zisserman: Very deep convolutional networks for large-scale image recognition. In: 3rd int. Conf Learn. Represent. ICLR 2015 - conf. Track proc., 1e14 (2015)
25. K. He, X. Zhang, S. Ren, J. Sun: Deep residual learning for image recognition, in: Proc. IEEE Comput. Soc. Conf. Comput Vis. Pattern Recognit., 770e778 (2016)
26. G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger: Densely connected convolutional networks, in: Proc. - 30th IEEE conf. Comput. Vis. Pattern recognition, CVPR 2017, 4700e4708 (2017)
27. M. Tan, Q.V. Le: EfficientNet: rethinking model scaling for convolutional neural networks, in: 36th int. Conf. Mach. Learn. ICML 2019, 6105e6114 (2019)
28. Aryan Mobiny, Pietro Antonio Cicalese, Samira Zare,Pengyu Yuan, Mohammad sajad Abavisani, Carol C. Wu, Jitesh Ahuja, Patricia M. de Groot, Hien Van Nguyen: Radiologist-level covid-19 detection using ct scans with detail-oriented capsule networks, arXiv. preprint 07407,. 1e11 (2020)
29. M. Polsinelli, L. Cinque, G. Placidi: A light CNN for detecting COVID-19 from CT scans of the chest, Pattern Recogn. Lett. 140, 95e100 (2020).
30. X. He, X. Yang, S. Zhang, J. Zhao, Y. Zhang, E. Xing, P. Xie: Sample-efficient deep learning for covid-19 diagnosis based on ct scans, medRxiv 1, 1e10 (2020).
31. P. Silva, E. Luz, G. Silva, G. Moreira, R. Silva, D. Lucio, D. Menotti: COVID-19 detection in CT images with deep learning: a voting-based scheme and cross-datasets analysis, Inform. Med. Unlock, 20, 1e9 (2020).
32. Wu, Jianxin: Introduction to convolutional neural networks., National Key Lab for Novel Software Technology. Nanjing University. China 5, 23, 495 (2017)
33. Glorot, Xavier, and Yoshua Bengio.: Understanding the difficulty of training deep feedforward neural networks, In Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 249-256 (2010)
34. Zan, Tao, Hui Wang, Min Wang, Zhihao Liu, and Xiangsheng Gao: Application of multi-dimension input convolutional neural network in fault diagnosis of rolling bearings, Applied Sciences 9,13, 2690 (2019)
35. Indolia, Sakshi, Anil Kumar Goswami, Surya Prakesh Mishra, and Pooja Asopa: Conceptual understanding of convolutional neural network-a deep learning approach, Procedia computer science, 132, 679-688 (2018)