Information Fusion 67 (2021) 208–229

Contents lists available at ScienceDirect

Information Fusion

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

Covid-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network

Shui-Hua Wang [a,b,c,1], Vishnu Varthanan Govindaraj [d,1], Juan Manuel Górriz [e,f,*], Xin Zhang [g,*], Yu-Dong Zhang [b,h,*]

[a] Department of Cardiovascular Sciences, University of Leicester, LE1 7RH, UK
[b] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[c] School of Architecture Building and Civil engineering, Loughborough University, Loughborough LE11 3TU, UK
[d] Department of Biomedical Engineering, Kalasalingam Academy of Research and Education, Srivilliputhur post, Krishnankoil 626 126, Tamil Nadu, India
[e] Department of Signal Theory, Networking and Communications, University of Granada, Granada, Spain
[f] Department of Psychiatry, University of Cambridge, Cambridge CB21TN, UK
[g] Department of Medical Imaging, The Fourth People's Hospital of Huai'an, Huai'an, Jiangsu Province, 223002, China
[h] School of Informatics, University of Leicester, Leicester LE1 7RH, UK

ARTICLE INFO

Keywords:
Deep feature fusion
Convolutional neural network
Graph convolutional network
Multiple-way data augmentation
Batch normalization
Dropout
Rank-based average pooling

ABSTRACT

(**Aim**) COVID-19 is an infectious disease spreading to the world this year. In this study, we plan to develop an artificial intelligence based tool to diagnose on chest CT images.

(**Method**) On one hand, we extract features from a self-created convolutional neural network (CNN) to learn individual image-level representations. The proposed CNN employed several new techniques such as rank-based average pooling and multiple-way data augmentation. On the other hand, relation-aware representations were learnt from graph convolutional network (GCN). Deep feature fusion (DFF) was developed in this work to fuse individual image-level features and relation-aware features from both GCN and CNN, respectively. The best model was named as FGCNet.

(**Results**) The experiment first chose the best model from eight proposed network models, and then compared it with 15 state-of-the-art approaches.

(**Conclusion**) The proposed FGCNet model is effective and gives better performance than all 15 state-of-the-art methods. Thus, our proposed FGCNet model can assist radiologists to rapidly detect COVID-19 from chest CT images.

1. Introduction

COVID-19 (also known as coronavirus) was declared as a Public Health Emergency of International Concern on 30/01/2020, and declared as a worldwide pandemic on 11/03/2020 [1]. Till 16/Sep, this COVID-19 pandemic caused 29.6 million confirmed cases and 936.9 thousand death tolls (US 199.1k deaths, Brazil 133.2k deaths, India 82.0k deaths, Mexico 71.6k deaths, UK 41.6k deaths, etc.)

Two prevail diagnosis are available. One is viral testing via a naso-pharyngeal swab to test the presence of viral RNA fragments. The samples are then tested by the method of real-time reverse transcription polymerase chain reaction (rRT-PCR) [2]. In some situation, a nasal swab or sputum sample may also be used. Results of rRT-PCR are generally available within some hours to two days. Another is imaging methods, among which the chest computed tomography (CCT) is one of the imaging devices that can provide the highest sensitivity. The main biomarkers in CCT differentiating COVID-19 from healthy people are the asymmetric peripheral ground-glass opacities (GGOs) without pleural effusions. The advantages of imaging methods are that they could aid in screening or accelerate the speed of diagnosis, especially with shortages of RT-PCR [3].

However, manual interpretation by radiologists is tedious and easy to be influenced by inter-expert and intra-expert factors (such as fatigue, emotion, etc.). Besides, the diagnosis throughput by human experts are

* Corresponding authors.
E-mail addresses: shuihuawang@ieee.org (S.-H. Wang), g.vishnuvarthanan@klu.ac.in (V.V. Govindaraj), gorriz@ugr.es (J.M. Górriz), 973306782@qq.com (X. Zhang), yudongzhang@ieee.org (Y.-D. Zhang).
[1] Those authors contributed equally to this paper, and should be regarded as co-first authors.

https://doi.org/10.1016/j.inffus.2020.10.004
Received 24 August 2020; Received in revised form 19 September 2020; Accepted 4 October 2020
Available online 9 October 2020

not comparable with machines. Particularly, early symptoms are smaller, which may be neglected by human experts. Smart diagnosis systems via computer vision and artificial intelligence can benefit patients, radiologists, experts and hospitals.

Traditional artificial intelligence (AI) and modern deep learning (DL) methods have achieved excellent results in analyzing medical images, e.g., Lu [4] proposed a radial-basis-function neural network (RBFNN) to detect pathological brains. Yang [5] presented a kernel-based extreme learning classifier (KELM) to create a novel pathological brain detection system. Their methods were robust and effective. Lu [6] proposed a novel extreme learning machine trained by the bat algorithm (ELM-BA) approach. Li and Liu [7] employed a real-coded biogeography-based optimization (RCBBO) for pathological brain detection, and this RCBBO method can be transferred to COVID-19 detection. Jiang [8] proposed a six-layer convolutional neural network with leaky rectified linear unit for fingerspelling recognition. Their method was shorted as 6L-CLF. Szegedy, et al. [9] presented the GoogleNet. Guo and Du [10] suggested the use of ResNet-18 for thyroid ultrasound classification. Fulton, et al. [11] employed ResNet-50 for classification of Alzheimer's disease. Their method is called RN-50-AD. Togacar, et al. [12] used SqueezeNet and MobileNetV2 to extract features, and employed social mimic optimization (SMO) for feature selection and combination. Cohen, et al. [13] used a large non-COVID-19 chest X-ray set to construct features for COVID-19 images. They authors predicted geographic extent score and lung opacity score to gauge severity of COVID-19 infection. Their method was abbreviated as Covid severity score net (CSSNet) in short. Loey, et al. [14] used generative adversarial network (GAN) to generate more images, and they found GoogleNet with GAN works better for two-class classification. They method was called GGNet. Li, et al. [15] used ResNet50 as the backbone. The CNN features were combined by a max-pooling operation. The resulting feature map was fed to a fully-connected layer to generate the probability score of COVID-19, community acquired pneumonia (CAP), and non-pneumonia. Their method was called COVNet. Ni, et al. [16] used 3D U-Net and MVP-Net on 96 COVID-19 patients in CCT images, for pulmonary lobe segmentation, COVID-19 lesion detection and COVID-19 lesion segmentation. Their method is called NiNet in this study. Ko, et al. [17] proposed a simple 2D deep learning framework for single CCT image. The authors compared four pretrained models: VGG16, ResNet-50, Inception-V3, and Xception. They found ResNet-50 showed the best performance. The authors used two augmentation method: image rotation and zoom. Their proposed additional layers consist of a flatten layer, a fully connected layer with 32 neurons, and a fully connected layer with 3 neurons. Their model can classify three classes: non-pneumonia, other pneumonia, and COVID-19. Their method was named as fast-track COVID-19 classification network (FCONet). Wang, et al. [18] developed a weakly-supervised deep learning framework using 3D CT volumes for COVID-19 classification and lesion localization. In their method, the lung region was segmented via a pre-trained UNet. The segmented 3D lung region was fed into a 3D deep neural network to predict the COVID-19 infection probability. Their method was named 3D deep convolutional neural network to detect COVID-19 (DeCovNet). Tabik, et al. [19] proposed a COVID-SDNet for predicting COVID-19 based on chest X-ray images.

However, most of the existing COVID-19 algorithms above employed single feature representation (SFR), and ignore fusing multiple feature representations (MFRs). Commonly, MFRs can yield better results than using SFR, because MFR is more informative and accurate than any SFR, and MFR consists of all the necessary information. The disadvantages of MFR are (i) they are of high-dimensionality and needs fusion technology, but fuse sometimes will introduce the distortion into the fused features; (ii) Fusion is not a static process in nature; (iii) Fusion of trivial features may affect the results. For example, Hasan, et al. [20] fused MFRs in robust hemolytic peptide prediction. Their cross-validation results showed their method outperformed state-of-the-art approaches. Li, et al. [21] used kernel sparse MFRs for hyperspectral (HS) image classification. Their method gave better results than state-of-the-art HS image classification methods. There are more success cases of using MFRs yielding better performances than SFR.

Table 1
Demographics of subjects used in this study.

	Subject nos	Image nos.	Age range
COVID-19	142	320	22–91
HC	142	320	21–76

The main **contribution** of this paper is deep feature fusion (DFF), viz., the fuse of multiple deep feature representations from both convolutional neural network (CNN) and graph convolutional network (GCN). CNN yields individual image-level representation (IIR), while GCN yields relation-aware representation (RAR). Hence, IIR and RAR are fused together at feature-level, and experiments proved the DFF proposed is efficient, and the fused features give better performances than using IIR features alone. Particularly, our method addresses related subproblems, e.g., feature selection, feature fusion, classifier selection. The resulting system is an efficient pipeline for COVID-19 diagnosis. We propose a fully automatic method which aims to ease the burden of the radiologist. Other **contributions** of this study are: (i) we propose to use batch normalization and dropout to our deep neural network model; (ii) we use rank-based average pooling to replace traditional max pooling; (iii) we propose multiple-way data augmentation. Finally, our model was demonstrated to give better performance than state-of-the-art approaches.

2. Dataset and preprocessing

Tables 12 and 13 itemized the abbreviation and mathematical symbols used in this study for easy reading. See Appendix A and Appendix B.

2.1. Dataset

Image acquisition CT configuration and method: Philips Ingenuity 64 row spiral CT machine, KV: 120, MAS: 240, layer thickness 3 mm, layer spacing 3 mm, screw pitch 1.5: lung window (W: 1500 HU, L: −500 HU), Mediastinum window (W: 350 HU, L: 60 HU), thin layer reconstruction according to the lesion display, layer thickness and layer distance are 1 mm lung window image. The patients were placed in a supine position, breathing deeply after holding in, and conventionally scanned from the lung tip to the costal diaphragm angle.

For each subject, 1–4 slices were chosen by radiologists using slice level selection method, because usually 4 slices are sufficient to cover the lesion. For COVID-19 pneumonia patients, the slice showing the largest size and number of lesions was selected. For normal subjects, any level of the image can be selected.

The resolutions of all selected images are $1,024 \times 1,024 \times 3$. Table 1 lists the demographics of subjects, where we have two categories: (i) COVID-19 patient, and (ii) healthy control (HC) subjects.

When there are differences between the analyses of two junior radiologist (A_1, A_2), a superior doctor (A_3) was consulted to reach a consensus. Suppose X means a CCT scan, L means the labeling of each individual expert, and the final labeling $\widehat{L}$is obtained by

$$\widehat{L}(X) = \begin{cases} L(A_1) & L(A_1) = L(A_2) \\ \mathrm{MV}(L_{all}) & \text{otherwise} \end{cases} \quad (1)$$

where L_{all} represents the labeling of all three experts, i.e., $L_{all} = [L(A_1), L(A_2), L(A_3)]$ MV denotes majority voting.

2.2. Preprocessing

The original dataset containing 320 COVID-19 images and 320 HC

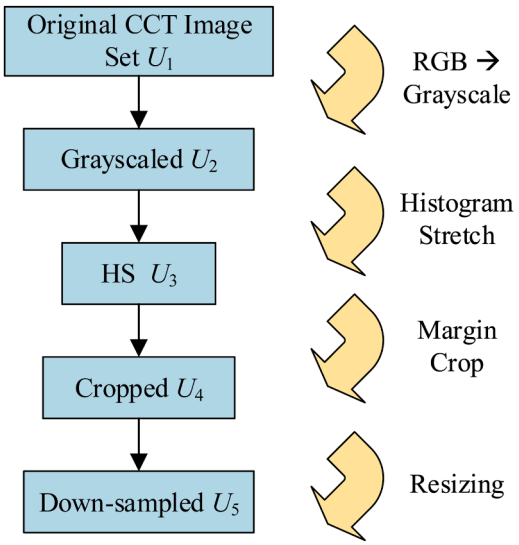


Fig. 1. Illustration of preprocessing.

images. The dataset is symbolized as U_1, each image is symbolized as $u_1(i) \in U_1, i = 1,2,\cdots,|U| = 640$. We have $U_1 = [u_1(1), u_1(2), \ldots, u_1(i), \ldots, u_1(|U|)]$. The size of each image is $size[u_1(i)] = W_1 \times H_1 \times C_1$. Here $W_1 = H_1 = 1024$, $C_1 = 3$.

The raw images are not suitable to train the deep neural networks, because (i) they have redundant information in three color channels; (ii) their contrast are incoherent; (iii) they contained background, checkup bed, and text information; and (iv) their sizes are too large. Fig. 1 shows the pipeline of preprocessing of our COVID-19 dataset.

First, we converted the color images to grayscale by only reserving the luminance information, and thus got the grayscale image set U_2 as

$$\begin{aligned} U_2 &= G(U_1) \\ &= \{u_2(1), u_2(2), \ldots, u_2(i), \ldots u_2(|U|)\} \end{aligned} \tag{2}$$

where means the grayscale operation. Now $size[u_2(i)] = W_2 \times H_2 \times C_2$. Here $W_2 = H_2 = 1024$, $C_2 = 1$.

Second, histogram stretching (HS) method was used to increase every image's contrast. For ith image $u_2(i)$, $i = 1, 2, \cdots, |U|$, we first calculate their minimum grayscale value $\mu_{min}(i)$ and maximum grayscale value $\mu_{max}(i)$ respectively by

$$\mu_{min}(i) = \min_{x=1}^{W_1} \min_{y=1}^{H_1} u_2(i|x, y) \tag{3.a}$$

$$\mu_{max}(i) = \max_{x=1}^{W_1} \max_{y=1}^{H_1} u_2(i|x, y) \tag{3.b}$$

here (x, y) means coordinates of pixel of the image $u_2(i)$. The new histogram stretched image $u_3(i)$ is obtained by

$$u_3(i) = \frac{u_2(i) - \mu_{min}(i)}{\mu_{max}(i) - \mu_{min}(i)} \tag{4}$$

In all, we get the histogram stretched image set $U_3 = HS(V_2) = \{u_3(1), u_3(2), \cdots, u_3(i), \ldots u_3(|U|)\}$.

Third, we crop the images to remove the texts at the margin areas, and the checkup bed at the bottom area. Thus, we get the cropped dataset U_4 as

$$\begin{aligned} U_4 &= C(U_3, [v_t, v_b, v_l, v_r]) \\ &= \{u_4(1), u_4(2), \cdots, u_4(i), \cdots, u_4(|U|)\} \end{aligned} \tag{5}$$

where C represents crop operation. Parameter (v_t, v_b, v_l, v_r) means the crop values in unit of pixel from top, bottom, left, and right. We set $v_t = v_b = v_l = v_r = 150$. Now the size of each image $size[u_4(i)] = W_4 \times H_4 \times C_4$. We can have $W_4 = H_4 = 724$, and $C_4 = C_2 = 1$.

Fourth, we downsampled each image to size of $[W_5, H_5]$, and we now get the resized image set U_5 as

$$\begin{aligned} U_5 &= \Downarrow(U_4, [W_5, H_5]) \\ &= \{u_5(1), u_5(2), \ldots, u_5(i), \ldots u_5(|U|)\} \end{aligned} \tag{6}$$

where $\Downarrow$: $x \mapsto y$ means the downsampling (DS) function, where y is a downsampled image of original image x. In this study, $W_5 = H_5 = 256$, $C_5 = 1$. The advantage of DS are two parts: (i) It can save storage, as shown in Table 2. (ii) Smaller-size dataset can help the following classification system from overfitting. The reason why we set $W_5 = H_5 = 256$is based on trial-and-error method. We found that larger size will bring in overfitting which impairs the performance, and meanwhile, smaller size will make the images blurry which also decreases the classifier's performances.

Table 2
Image size and storage per image at each preprocessing step.

Preprocess	Symbol	W	H	C	Size (per image)	Storage (per image)
Original	$u_1(i)$	1024	1024	3	3, 145, 728	12,582,912
Grayscale	$u_2(i)$	1024	1024	1	1, 048, 576	4194,304
HS	$u_3(i)$	1024	1024	1	1, 048, 576	4194,304
Crop	$u_4(i)$	724	724	1	524, 176	2096,704
DS	$u_5(i)$	256	256	1	65, 536	262,144

Table 2 compares the size and storage of each image $u_s(i), s = 1, \cdots, 5$, $i = 1, \cdots, |U|$at every preprocessing step. We can see here after preprocessing procedure, each image will only cost about 2.08% of its original storage or size. The compression ratio (CR) rates of ith image of final state U_5 to original stage U_1 were calculated as. $CR_{Storage}(i) = byte[u_5(i)]/byte[u_1(i)] = 262, 144/12, 582, 912$, and $CR_{size}(i) = size[u_5(i)]/size[u_1(i)] = 65, 536/3, 145, 728$. Hence, we can get $CR_{size}(i) = CR_{size}(i) = 2.083\%, \forall i = 1, 2, \ldots, |U|$. Fig. 2(a and b) shows two samples (COVID of a and HC of b) from the preprocessed dataset U_5. Fig. 2(c) delineates the lesions of (a) within red circles.

3. Methodology

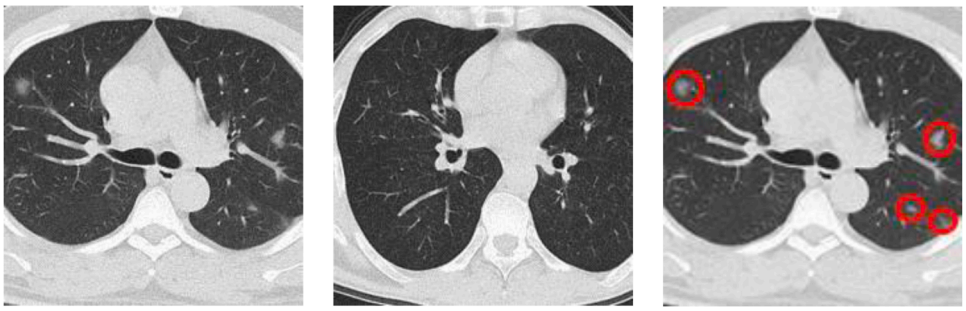
The motivation of this study is two-fold. First, we plan to create a customary state-of-the-art comparable convolutional neural network with several improvements, including batch normalization, dropout, rank-based average pooling, and multiple-way data augmentation. This motivation of using CNN is to extract individual image-level representation (IIR).

Nevertheless, CNN cancels out the relation of a particular image among a group of images. In contrast, this relation-aware representation (RAR) can be captured by graph convolutional network (GCN). Hence, the second main motivation is (i) to use GCN to establish connectivity analysis and extract RAR features; and (ii) to fuse CNN features and GCN features together to enhance the classifier's performance.

Using GCN with CNN can obtain better performance than using CNN merely. Shi, et al. [22] used GCN for cervical cell classification. Their results were significantly better than ResNet-101 and DenseNet-121. Bin, et al. [23] used GCN to extract structure-aware human pose estimation. Their experiments on single- and multi-person estimation benchmark datasets showed that GCN consistently outperforms competing state-of-the-art methods. Tian, et al. [24] proposed a novel GCN-based interactive prostate segmentation in MR images. Their method yielded mean Dice similarity coefficients of 93.8 ± 1.2% and 94.4 ± 1.0% on their in-house and PROMISE12 datasets, respectively. All those three literatures show the powerfulness of GCN.

3.1. Basics of CNN

Traditional machine learning achieved excellent results on disease detections [25,26]. Convolutional neural network (CNN) is a new artificial neural network. Generally, CNN is composed of conv layers (CLs),



(a) COVID-19 (b) HC (c) Lesions of (a)

Fig. 2. Two samples of preprocessed dataset $\boldsymbol{U}_5$.

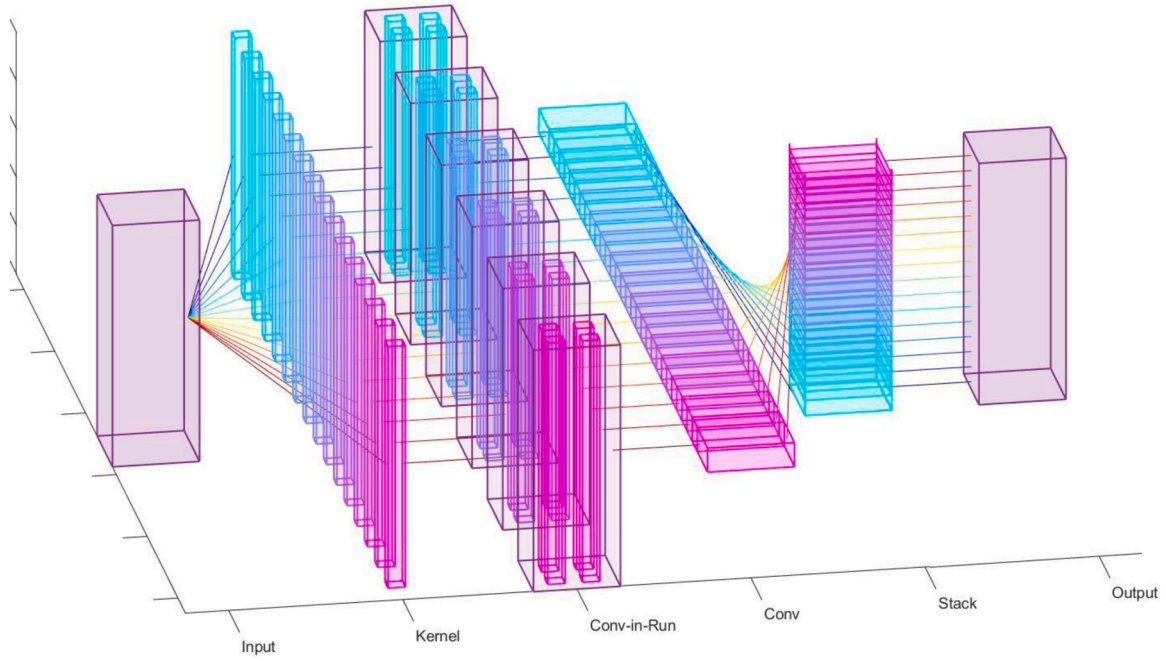


Fig. 3. Illustration of a complete conv layer.

pooling layers (PLs), non-linear activation functions (NLAFs) and fully connected layers (FCLs) [27,28].

The essential operation in CNN is convolution. A complete CL performs 2D convolution along the width and height directions [29]. Note that the weights in CNN are initialized randomly, and then weights are learnt from the data itself by network training. Fig. 3 illustrates the pipeline of input feature maps passing across a complete CL.

Fig. 3 shows there are three steps during a complete conv layer: (i) Kernel-based convolution; (ii) Stack; (iii) NLAF. Assume there is an input matrix Γ, kernels Q_j, $\forall j \in [1, \cdots, J]$, and an output T, (here output T means output of the whole three-step complete conv layer, not the output of merely convolution operation). Note a conv layer means the layer runs convolution, and the "complete conv layer" means the combination of the conv layer; the stack, and NLAF. In Fig. 3, we used the same color to denote the input and output, because the output is the input of next conv layer.

For each kernel Q_j, the convolution output is

$$\text{Step1}: f(j) = \Gamma \otimes Q_j, \forall j \in [1, \cdots, J] \tag{7}$$

where $\otimes$ means convolution operation. Then, all $f(j)$ matrixes are stacked into a three-dimensional matrix F.

$$\text{Step2}: F = [f(1), \cdots, f(J)] \tag{8}$$

where means the stack operation. Finally, the matrix F is passed into the NLAF and output the final matrix [30]

$$\text{Step3}: T = \text{NLAF}(F) \tag{9}$$

We can calculate theirs sizes S of three main components (input, kernel, and output) as

$$S(x) = \begin{cases} W_\Gamma \times H_\Gamma \times C_\Gamma & x = \Gamma \\ W_Q \times H_Q \times C_Q & x = Q_j, \forall j \in [1, \cdots, J] \\ W_T \times H_T \times C_T & x = T \end{cases} \tag{10}$$

Where the triple elements (W, H, C) represent the size of height, width, and channels of the matrix, respectively [31]. The subscript Γ, Q, and T represent input, kernel, and output, respectively. J denotes total number of filters. Note that $C_\Gamma = C_Q$, which means the channel of input $C_\Gamma$should equal the channel of kernel C_Q.

Assume those filters move with padding of v_p and stride of v_s, we can get the sizes ($W_T \times H_T \times C_T$) of output matrix T by simple math as [32]:

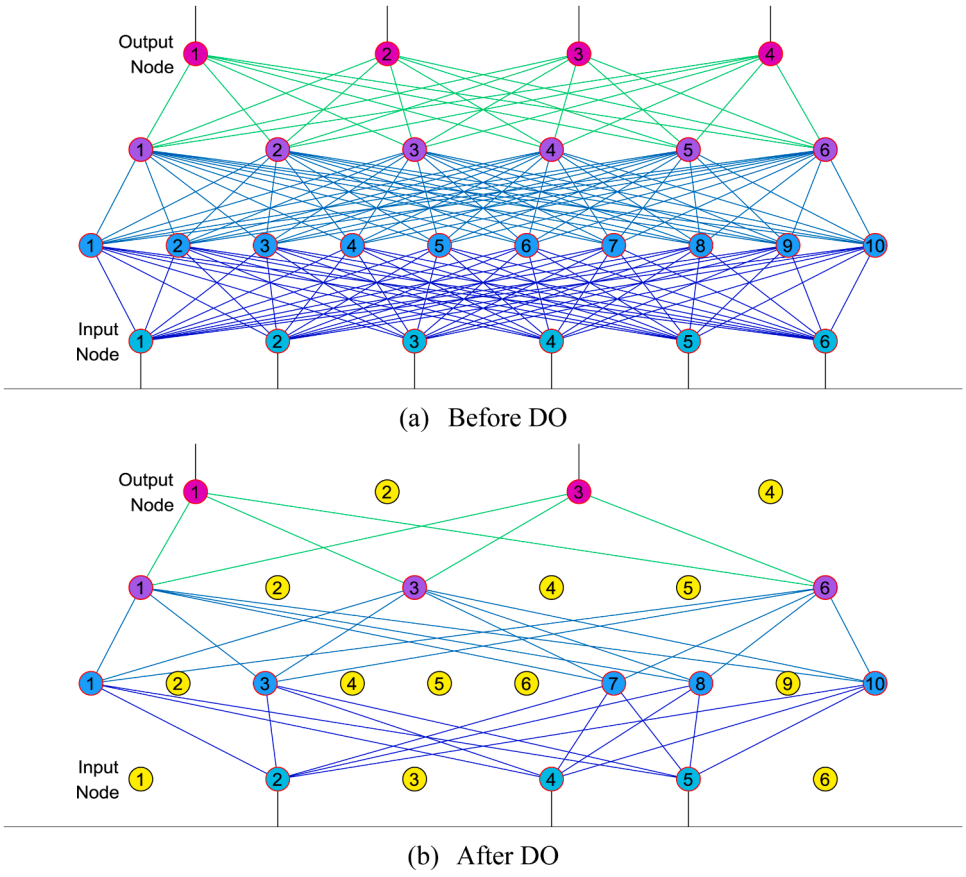


Fig. 4. A simplistic example of a 4-layer DON.

$$\begin{cases} W_T = 1 + \dfrac{(2 \times v_p + W_\Gamma - W_Q)}{v_s} \\ H_T = 1 + \dfrac{(2 \times v_p + H_\Gamma - H_Q)}{v_s} \\ C_T = J \end{cases} \tag{11}$$

where $\lfloor \cdot \rfloor$ represents the floor function. The channel of output C_T should equal the number of filters J.

For the last step, viz., the NLAF β, it usually selects the rectified linear unit (ReLU) function [33]. Suppose f_{ij} is the entry of the matrix F, we have

$$\begin{aligned} \beta_{\mathrm{ReLU}}(f_{ij}) &= ReLU(f_{ij}) \\ &= max(0, f_{ij}) \end{aligned} \tag{12}$$

ReLU is preferred to traditional NLAFs such as sigmoid (SM) function and hyperbolic tangent (HT) as below:

$$\beta_{\mathrm{SM}}(f_{ij}) = \left(1 + e^{-f_{ij}}\right)^{-1} \tag{13}$$

$$\begin{aligned} \beta_{\mathrm{HT}}(f_{ij}) &= tanh(f_{ij}) \\ &= \frac{\left(e^{f_{ij}} - e^{-f_{ij}}\right)}{\left(e^{f_{ij}} + e^{-f_{ij}}\right)} \end{aligned} \tag{14}$$

3.2. Improvement 1: batch normalization and dropout

The motivation of batch normalization (BAN) is to solve the "internal covariant shift (ICS)", which means the effect of randomness of the distribution of inputs to internal CNN layers during training. The existence of ICS will worsen the CNN's performance [34,35].

This study introduced BAN to normalize those internal layer's inputs $\Gamma = \{\gamma_i\}$over every mini-batch (suppose its size is $|\Gamma|$), in order to guarantee the batch normalized output $T = \{t_i\}$have a uniform distribution. Mathematically, BAN is to learn a function from

$$\underbrace{\{\gamma_i, i = 1, 2.\cdots, |\Gamma|\}}_{\Gamma} \mapsto \underbrace{\{t_i, i = 1, 2, \cdots, |\Gamma|\}}_{T} \tag{15}$$

During training, the empirical mean μ_e and empirical variance ϕ_e can be calculated as

$$\begin{cases} \mu_e = \dfrac{1}{|\Gamma|}\left(\sum_{i=1}^{|\Gamma|} \gamma_i\right) \\ \phi_e = \dfrac{1}{|\Gamma|}\sum_{i=1}^{|\Gamma|} (\gamma_i - \mu_e)^2 \end{cases} \tag{16}$$

The input $\gamma_i \in \Gamma$ was first normalized to $\hat{\gamma}_i$

$$\hat{\gamma}_i = \frac{\gamma_i - \mu_e}{\sqrt{(\phi_e + \alpha_s)}} \tag{17}$$

where α_s in denominator in Eq. (17) is stability factor, used to enhance the numerical stability. Now the $\hat{\gamma}_i$have zero-mean and unit-variance characteristics. In order to have a more expressive deep neural network [36] (here expressive means the network's expressive power, i.e., the ability to express functions), a transformation is usually carried out as

$$t_i = A_1 \times \hat{\gamma}_i + A_2, i = 1, 2, \cdots, |\Gamma| \tag{18}$$

where the parameters A_1 and A_2 are two learnable parameters during training. The transformed output $t_i \in T$ is then passed to the next layer and the normalized $\hat{\gamma}_i$remains internal to the current layer.

In the inference stage, we no longer have minibatch. So instead of

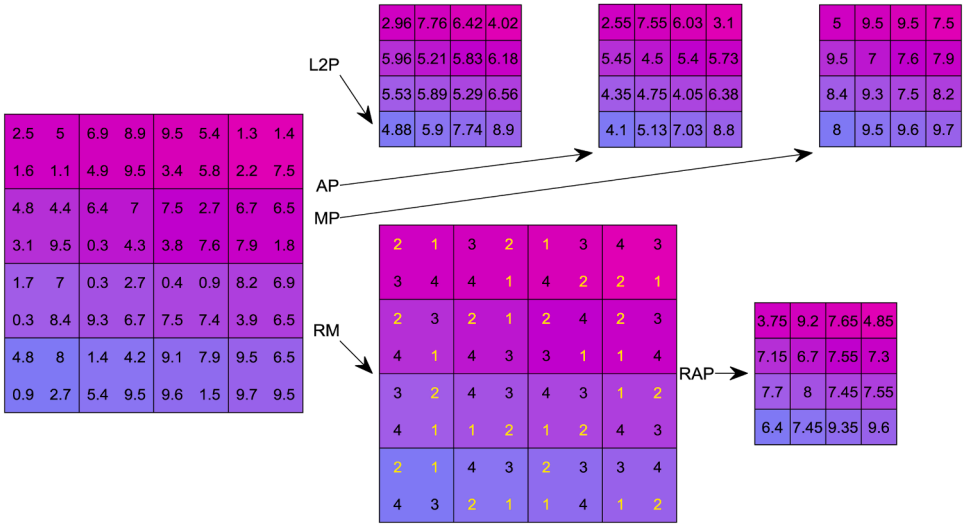


Fig. 5. A simplistic example of four pooling technologies (L2P, AP, MP, and RAP).

calculating μ_e and ϕ_e, we will calculate population mean μ_p and population variance ϕ_p, and we have the output $\widehat{t_i}$at the inference stage as

$$\widehat{t_i} = A_1 \times \frac{\gamma_i - \mu_p}{\sqrt{(\phi_p + \Delta)}} + A_2 \tag{19}$$

On the other hand, a dropout layer will be introduced before the fully-connected layer. It is a regularization technique which means randomly dropping out neurons during the training. Dropout can help avoid overfitting of deep neural networks. Srivastava, et al. [37] proposed the concept of dropout neurons (DN) by randomly drop neurons and set to zero their neighboring weights from the CNN during training. Suppose collection of all fully-connected neurons is {}, the collection of dropped neurons is {}, the collection of reserved neuron is $\{-\}$. The selections of DN are random with a retention probability (α_{rp}) defined as:

$$\alpha_{rp} = \frac{|\mathrm{D} - \mathrm{N}|}{\mathrm{N}} \tag{20}$$

Suppose we have a neuron $N(i, j)$ and its corresponding original weights are $w(i, j)$. During training, the neuron's weights $w^T(i, j)$ will update as:

$$w^T(i,j) = \begin{cases} w(i,j) & N(i,j) \in \mathrm{D} \\ 0 & N(i,j) \notin \mathrm{D} \end{cases} \tag{21}$$

During inference, we run the entire CNN without dropout, but the weights of FCLs $w^I(i, j)$ using DNs are downscaled (viz., multiplied) by α_{rp}.

$$w^I(i,j) = \alpha_{rp} \times w(i,j) \tag{22}$$

The compression ratio of learnable weights (CRLW), is the squared value of retention probability α_{rp}.

$$\mathrm{CRLW} = C^D \Big/ C = \alpha_{rp}^2 \tag{23}$$

Where C^D is the total number of learnable weights after dropout, and C is the total number of learnable weights before dropout. Fig. 4 shows a simplistic example, and the detailed analysis is in Appendix C.

3.3. Improvement 2: rank-based average pooling

The pooling function fundamentally replaces the output of a layer (particularly conv layers) with a summary statistic of the adjacent outputs at a particular position. The pooling method is capable of making less-sensitive activations in the pooled map than the original feature map to the accurate spots of structures within the image.

For a region to be pooled Ψ with size of $n \times n$, here n means pooling size. Suppose the pixels within the region $\Psi = \{\psi_{i,j}\}, (1 \le i.j \le n)$are

$$\Psi = \begin{bmatrix} \psi_{1,1} & \cdots & \psi_{1,n} \\ \cdots & \cdots & \cdots \\ \psi_{n,1} & \cdots & \psi_{n,n} \end{bmatrix} \tag{24}$$

The l_2 norm pooling (L2P) calculates the l_2 norm of the given region Ψ. Assume the output pooling matrix is P, the L2P output $P^{L2P}(\Psi)$ is defined as $P^{L2P}(\Psi) = sqrt(\sum_{i,j=1}^{n} \psi_{ij}^2)$. In this study, we add a constant $1/|\Psi|$, where $|\Psi|$ means the number of elements of region Ψ. We have $|\Psi| = n^2$. We added the new constant $1/|\Psi|$ under the square root, and the constant $1/|\Psi|$ does not influence training and inference.

$$P^{L2P}(\Psi) = \sqrt{\frac{\sum_{i,j=1}^{n} \psi_{ij}^2}{|\Psi|}} \tag{25}$$

The average pooling (AP) calculates the mean value in the region Ψ as

$$\begin{aligned} P^{AP}(\Psi) &= average(\Psi) \\ &= \frac{\sum_{i,j=1}^{n} \psi_{i,j}}{|\Psi|} \end{aligned} \tag{26}$$

The max pooling (MP) operates on the region Ψ and selects the max value.

$$\begin{aligned} P^{MP}(\Psi) &= max(\Psi) \\ &= \max_{i,j=1}^{n} \psi_{i,j} \end{aligned} \tag{27}$$

Shi, et al. [38] proposed three different rank-based pooling approaches. Their advantages compared to ordinary pooling methods are: (i) Ranking list is invariant under slight changes of activation values; (ii) Important activation values can be easily distinguished by their cognate ranks; and (iii) Usage of rank can avoid scale problems arise from value-based pooling methods. Among rank-based pooling approaches, the rank-based average pooling (RAP) gives better performances than state-of-the-art tactics, and has been applied in many fields. For example, Jiang [39] added RAP to convolutional neural networks for the susceptibility-weighted imaging based cerebral microbleed detection. They got a high accuracy of 97.18%. Sun, et al. [40] added RAP between each subspace mapping layer for facial expression recognition. Their method is better than PCANet and LDANet approaches. Akhtar and Ragavendran [41] compared rank-based pooling with traditional pooling methods, and stated the advantages of rank-based pooling are they

can assign ranks and weights to activations simultaneously.

RAP first calculate the rank matrix (RM) based on the values of each element $\psi_l \in \Psi$, usually lower ranks $r_l \in [1, 2, \cdots, n^2]$ are assigned to higher values (ψ_l) as

$$\psi_{l1}\langle\psi_{l2}\Rightarrow r_{l1}\rangle r_{l2} \tag{28}$$

Providing tied values ($\psi_{l1} = \psi_{l2}$), a constraint is added to Eq. (28).

$$(\psi_{l1} = \psi_{l2}) \wedge (l1 > l2) \Rightarrow r_{l1} > r_{l2} \tag{29}$$

RAP output of input Ψ is $P^{RAP}(\Psi)$, which used the a_g greatest activations

$$P^{RAP}(\Psi) = \frac{1}{a_g}\sum_{l}(\psi_l | 1 \le r_l \le a_g) \tag{30}$$

a_g is the rank threshold. If $a_g = 1$, then RAP will degrade to MP. On the other side, if $a_g = n^2$, then RAP will degrade to AP. Therefore, RAP is regarded as a trade-off between average pooling and max pooling. Note that L2P, AP, MP, and RAP work on every slice separately. Fig. 5shows a simplistic example of four pooling techniques and the explanation is shown in Appendix D.

3.4. Improvement 3: multiple-way data augmentation

To circumvent the small-size dataset (SSD) and lack of generation (LG) problems, there are four possible types of solutions, e.g., data augmentation (DA), data generation (DG), ensemble approaches (EA), and regularization.

DA will generate fake images by perturbing existing data, such as cropping, rotation. DG create data from a sampled data source. Synthetic minority over-sampling technique (SMOTE) [42] is a typical algorithm of DG. EA methods use multiple models to obtain better predictive performance than any model alone [43]. Regularization is mainly for the weights of models. Large weights will make the models unstable, because minor variation on the inputs will yield large differences in the output for large weights. Smaller weights are regarded to be more regular (i.e., less specialized). Hence, this type of technique is called weight regularization. DA is used due to its simple and ease to realize.

We proposed a η_{DA}-way multiple-way data augmentation (MDA) technology. The difference of our MDA to traditional DA is we use multiple ($\eta_{DA} > 10$) DA techniques. Assume the preprocessed dataset $U_5 = [u_5(1), \cdots, u_5(|U|)]$. The dataset $U_5$is divided into three sets:

$$U_5 \underset{\text{split}}{\Rightarrow} \{X^t, X^v, Y\} \tag{31}$$

where training set $X^t = [x^t(1), \cdots x^t(i), \cdots, x^t(|X^t|)]$, validation set $X^v = [x^v(1), \cdots, x^v(|X^v|)]$, and test set $Y = [y(1), \cdots, y(|Y|)]$. Meanwhile, the sum of sizes of training set, validation set, and test set equals to the size of preprocessed dataset, $|X^t| + |X^v| + |Y| = |U_5|$.

From the whole training image set X^t, we first performed the following seven DA techniques with different MDA factors χ. Note that each MDA technique will generate η_n new images. Suppose the output MDA training set is symbolized as $X^{tD} = \{x^{tD}(i)\}$.

(i) Rotation. Rotation angle vector χ^R skip the value of 0.

$$\begin{aligned}\overrightarrow{x^{t1}(i)} &= R[x^t(i)] \\ &= \left[x_1^{t1}\left(i, \chi_1^R\right), \ldots b_{\eta_n}^R\left(i, \chi_{\eta_n}^R\right)\right]\end{aligned} \tag{32}$$

where R means rotation operation.

(ii) Noise injection. The χ_m^N-mean χ_v^N-variance Gaussian noises

$$p(z) = \frac{1}{\sqrt{2\pi \times \chi_v^N}}\exp\left[-\frac{\left(z - \chi_m^N\right)^2}{2 \times \chi_v^N}\right] \tag{33}$$

were added to all training images to produce η_n new noised images, where z is the gray levels, and p is the probability density function. We have

$$\begin{aligned}\overrightarrow{x^{t2}(i)} &= N[x^t(i)] \\ &= \left[x_1^{t2}(i), \ldots x_{\eta_n}^{t2}(i)\right]\end{aligned} \tag{34}$$

where N means the noise injection operation. We used Gaussian noise in this study because it is the most common type found in images compared to impulse noise, speckle noise, and salt and pepper noise.

(iii) Horizontal Shear (HS) transform. New V images were generated by HT transform

$$\begin{aligned}\overrightarrow{x^{t3}(i)} &= H[x^t(i)] \\ &= \left[x_1^{t3}\left(i, \chi_1^H\right), \ldots x_{\eta_n}^{t3}\left(i, \chi_{\eta_n}^H\right)\right]\end{aligned} \tag{35}$$

where H means HS transform. HS factors χ^H skip the value of $\chi^H = 0$. Mathematically, if original coordinates are (u, v), and HS transformed coordinates are (u_1, v_1), then we have

$$\begin{cases} u_1 = u + \chi^H \times v \\ v_1 = v \end{cases} \tag{36}$$

Clearly, the HS transform is a special affine transform, which can be written as

$$[u_1, v_1, 1] = [u, v, 1] * \begin{bmatrix} 1 & 0 & 0 \\ \chi^H & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{37}$$

(iv) Vertical Shear (VS) transform.

$$\begin{aligned}\overrightarrow{x^{t4}(i)} &= V[x^t(i)] \\ &= \left[x_1^{t4}\left(i, \chi_1^V\right), \ldots x_{\eta_n}^{t4}\left(i, \chi_{\eta_n}^V\right)\right]\end{aligned} \tag{38}$$

where V means VS transform, which ran similarly as ST transform. Particularly, the VS factor is the same as HS factor $\chi_j^V = \chi_j^H, \forall j \in 1, 2, \cdots, \eta_n$.

(v) Random translation (RT). All training images $x^t(i)$ were translated η_n times with random horizontal shift ε^x and random vertical shift ε^y, both values of which are in the range of $[-a_Z, a_Z]$, and obey uniform distribution.

$$\varepsilon_m^\theta \sim [-a_Z, a_Z], \forall m \in [1, \eta_n] \wedge \forall \theta \in \{x, y\} \tag{39}$$

where a_Z is the maximum shift factor. So, we have

$$\begin{aligned}\overrightarrow{x^{t5}(i)} &= RT[x^t(i)] \\ &= \left[x_1^{t5}\left(i, \varepsilon_1^x, \varepsilon_1^y\right), \ldots x_{\eta_n}^{t5}\left(i, \varepsilon_{\eta_n}^x, \varepsilon_{\eta_n}^y\right)\right]\end{aligned} \tag{40}$$

(vi) Gamma correction (GC). The factor of GC χ^G will skip the value of 1.

$$\begin{aligned}\overrightarrow{x^{t6}(i)} &= G[x^t(i)] \\ &= \left[x_1^{t6}(i,\chi_1^G),\ldots x_{\eta_n}^{t6}(i,\chi_{\eta_n}^G)\right]\end{aligned} \tag{41}$$

where G means GC operation.

(vii) Scaling. All training images $\{x^t(i)\}$ were scaled with scaling factor χ^S, skipping $\chi^S = 1$.

$$\begin{aligned}\overrightarrow{x^{t7}(i)} &= S[x^t(i)] \\ &= \left[x_1^{t7}(i,\chi_1^S),\ldots x_{\eta_n}^{t7}(i,\chi_{\eta_n}^S)\right]\end{aligned} \tag{42}$$

(viii) Mirror and concatenation. All the above results are mirrored, we have

$$\overrightarrow{x^{n+\frac{\eta_{DA}}{2}}(i)} = S\left[\overrightarrow{x^n(i)}\right], \forall n \in \left\{1,2,\cdots,\frac{\eta_{DA}}{2}\right\} \tag{43}$$

where represents the mirror function. All the η_{DA}-way results are finally concatenated as

$$\underbrace{\overrightarrow{x^{tD}(i)}}_{\eta_{EF}} = \left\{\underbrace{x(i)}_{1}, \underbrace{\overrightarrow{x^1(i)}}_{\eta_n}, \cdots, \underbrace{\overrightarrow{x^{\eta_{DA}}(i)}}_{\eta_n}\right\} \tag{44}$$

where means concatenation, η_{EF} the enhance factor, meaning the ratio of enhanced training set to the original training set. η_{EF} is defined as

$$\eta_{EF} = \frac{\left|\overrightarrow{x^{tD}(i)}\right|}{|x^t(i)|} \tag{45}$$

We can calculate $\eta_{EF} = \eta_n \times \eta_{DA} + 1$. Thus, the MDA can be regarded as a function, making the enhanced training set η_{EF}times as large as raw training set X^t.

$$\{x^t(i) \in X^t\} \overset{MDA}{\longmapsto} \left\{\overrightarrow{x^{tD}(i)} \in X^{tD}\right\} \tag{46}$$

3.5. Improvement 4: deep feature fusion by graph convolutional network

To further improve the performance, we introduced a deep feature fusion (DFF) method based on graph convolutional network (GCN). GCN helps find the relation-aware representation (RAR) [44], and thus fuse RAR from GCN with IIR from CNN.

For a given graph $G = (V,E)$, where we have $|V|$ nodes $v_i \in V, i = 1, \cdots, |V|$and corresponding links $(v_i, v_j) \in E$. We can define an adjacency matrix $A \in {}^{|V| \times |V|}$ which embeds the relationship of all nodes. The purpose of GCN is to encode the graph G via a neural network model $f(X, A)$ where $X \in {}^{|V| \times D}$, where D means the feature dimension of each node [45]. Note that AX means the sum of all neighboring node features. So GCN can capture the RAR information [46].

A multi-layer GCN will update the node features based on following layer-wise rule:

$$H^{l+1} = f_{ReLU}\left(\widehat{A}H^lW^l\right) \tag{47}$$

where $\widehat{A} \in {}^{|V|\times|V|}$represents the normalized version of adjacency matrix A. f_{ReLU} is ReLU function. The variable $H^{(l)} \in {}^{|V|\times d_l}$is the feature representation at Lth layer [47].

To carry out the normalization $A \mapsto \widehat{A}$, we first calculate the degree

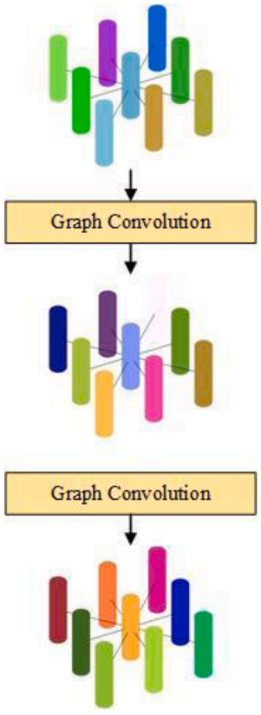


Fig. 6. Illustration of a two-layer GCN. (Different color cylinders mean different cluster centroids).

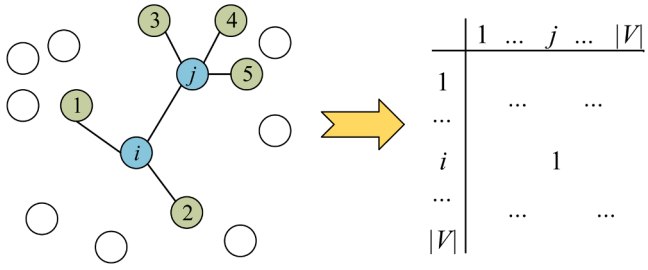


Fig. 7. Illustration of KNN-based adjacency matrix.

matrix $\delta \in {}^{|V| \times |V|}$which is a diagonal matrix

$$\delta_{ij}\underline{\text{def}}\begin{cases}\deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise}\end{cases} \tag{48}$$

The normalized $\widehat{A}$is obtained based on original adjacency matrix A and degree matrix δ [48].

Note the input $X = H^{(0)}$, so for a two-layer GCN as shown in Fig. 6, we have

$$H^{(1)} = f_{ReLU}\left(\widehat{A}XW^{(0)}\right) \tag{49.a}$$

$$H^{(2)} = f_{ReLU}\left(\widehat{A}H^1W^{(1)}\right) \tag{49.b}$$

where $W^{(0)} \in {}^{d_0\times d_1}$, and $W^{(1)} \in {}^{d_1\times d_2}$are two trainable weight matrixes. (d_0, d_1, d_2) are hyperparameters will be set in the experiment.

In our COVID-19 classification task, the GCN will be fused with previous CNN models $N(1) - N(4)$. The last FCL in previous CNN models is used as the individual image-level representation (IIR) $I \in {}^{d_0}$Afterwards, k-means clustering (KMC) is performed on those image-level representation features, and we can get $|V|$ cluster centroids $X \in {}^{|V|\times d_0}$. The clustering correlation shows the potential relationships of images. The adjacency matrix $A \in {}^{|V| \times |V|}$is defined as

Algorithm 1
Proposed deep feature fusion.

Input: IIR Feature I from CNN models
Algorithm of DFF
Step 1: Create RAR features $H^{(2)}$ from pre-constructed two-layer GCN model
Step 2: Dot product fusion combining IIR features and RAR features, $y = H^{(2)}I$
Step 3: Linear Projection, $z = yW^{(2)} + b$
Step 4: Softmax and cross-entropy (CE) loss

$$A_{mn} = \begin{cases} 1 & \text{if } X_m \in \Delta(X_n) \vee X_n \in \Delta(X_m) \\ 0 & \text{otherwise} \end{cases} \quad (50)$$

where $\vee$ means operation "or", Δ means the k-nearest neighbors (kNN) based on cosine similarity. The number of neighbors in kNN is symbolized as .

Fig. 7 shows an example, the three nearest neighbors of node i and node j are $\Delta(X_i) = (1, 2, j)$, $\Delta(X_j) = (3, 4, 5)$. So, we have $X_j \in \Delta(X_i)$, $X_i \notin \Delta(X_j)$. Using the 'or' operation we can conclude $A_{ij} = 1$. The node features $X$and adjacency matrix A are sent into two-layer GCN, and we can get $H^{(2)} \in {}^{|V|\times d_2}$. The fusion between $H^{(2)}$and I is accomplished by dot product fusion. Note we need to set $d_0 = d_2$,

$$y = H^{(2)}I \quad (51)$$

A linear projection (LP) with learnable weight $W^{(2)} \in {}^{|V| \times C}$, where C means the number of categories, we have

$$z = yW^{(2)} + b \quad (52)$$

where $z \in {}^{C}$, and b represents the bias. $C = 2$in this study because our task is a binary classification problem, i.e., COVID-19 and healthy people. Finally, softmax operation were performed on z, and cross entropy (CE) loss were calculated. Algorithm 1 shows the proposed deep feature fusion algorithm. During inference stage, the CNN's IIR features are gained and its corresponding GCN's RAR features are obtained by pre-constructed graph and trained two-layer GCN. Using both CNN and GCN, each image is represented by the fusion of its individual image-level representations and its relation-aware representations [22]. Fig. 8 shows the fusion flowchart.

3.6. Summary of proposed eight networks

In total, we proposed eight new networks [$N(1)$, ⋯, $N(8)$]: (i) We first designed a base network $N(1)$, $N(1)$ is called the base network (BN). (ii) we added batch normalization (BAN) and dropout (DO) techniques, and obtained the improved network, $N(2)$ named as "BDBN". (iii) Next, we developed $N(3)$ termed BDRBN, by introducing rank-based average pooling (RAP) to replace traditional max pooling (MP) in $N(2)$. (iv) Multiple data augmentation (MDA) was proposed and added to $N(3)$, so we get new network $N(4)$, which is short named as BDRMBN.

For the rest four networks, we add a new deep feature fusion (DFF) approach that combines the features from above networks [$N(1)$, $N(2)$, $N(3)$, $N(4)$] to RAR features from 2-layer GCN. The short names of [$N(5)$, $N(6)$, $N(7)$, $N(8)$] are defined as DBN, DBDBN, DBDRBN, DBDRMBN. Table 3 gives all the eight proposed network relationships.

As our expectation, the best model should be within $N(5-8)$since they fuse the features from CNNs $N(1-4)$with GCN. The best model will have a formal name as FGCNet, which means Fusion of GCN and CNN networks.

The configuration of proposed networks was designed by trial-and-error method. We set the number of conv layers as u_c. Similarly, the number of FCL layers is symbolized as u_f. The details of the base network can be found in Table 6.

3.7. Measures

The algorithm will run W runs, which help to reduce the randomness. At each run $w = 1, 2, \cdots, W$, the ideal $E^i$and real E^r confusion matrix over validation set are

Table 3
Eight proposed networks.

Index	Inheritance	Short Name	Description
$N(1)$		BN	Base Network
$N(2)$	←$N(1)$+BAN+DO	BDBN	Add BAN and DO to $N(1)$
$N(3)$	←$N(2)$-MP+RAP	BDRBN	Use RAP to replace MP in $N(2)$
$N(4)$	←$N(3)$+MDA	BDRMBN	Add MDA to $N(3)$
$N(5)$	←$N(1)$+DFF	DBN	Add DFF to $N(1)$
$N(6)$	←$N(2)$+DFF	DBDBN	Add DFF to $N(2)$
$N(7)$	←$N(3)$+DFF	DBDRBN	Add DFF to $N(3)$
$N(8)$	←$N(4)$+DFF	DBDRMBN (FGCNet*)	Add DFF to $N(4)$

(* Experiment below shows $N(8)$ gives the best performance, and we name it as FGCNet).

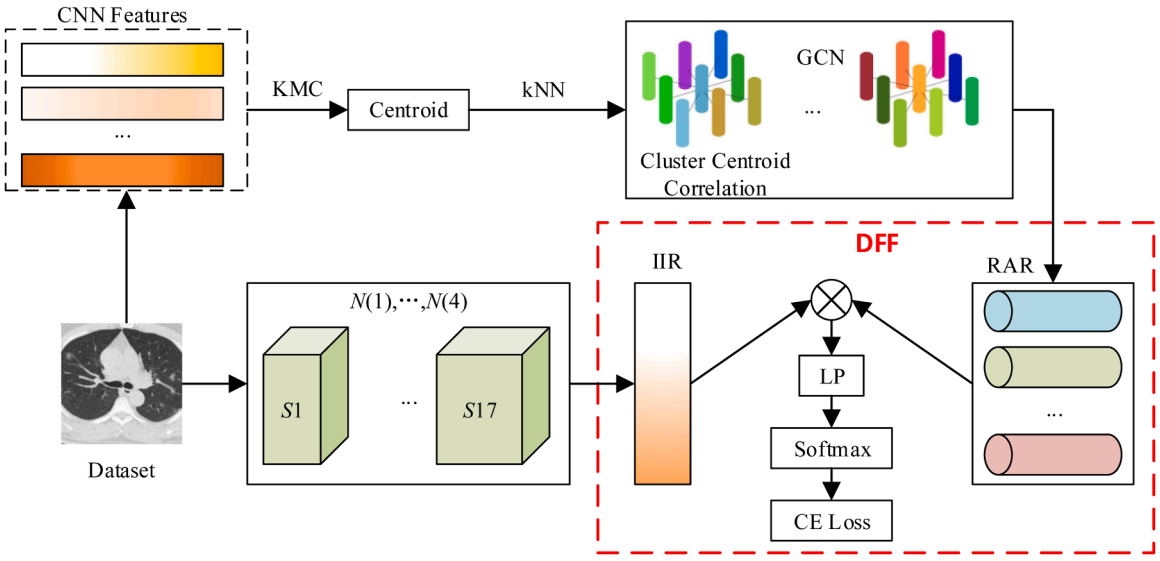


Fig. 8. Flowchart of deep feature fusion strategy. (LP = Linear Projection; CE = Cross Entropy).

Algorithm 2
Pseudocode of our algorithm.

Input: Original Image Set U_1 and its ground truth label $\hat{L}$.

Phase I: Preprocessing $U_1 \to U_5$

Grayscaling: $U_1 \to U_2$. See Eq. (2).

HS: $U_2 \to U_3$, See Eq. (4).

Crop: $U_3 \to U_4$. See Eq. (5).

Downsampling: $U_4 \to U_5$. See Eq. (6).

Phase II: Model Initialization

***N*(1)**: Create a base network. See Figure 9(a).

***N*(2)**: See Figure 9(b).

Add BAN to conv layers of ***N*(1)**. See Eqs. (18)(19).

Add DO to fully connected layers of ***N*(1)**. See Eqs. (21)(22).

***N*(3)**: Replace MP with RAP. See Eq. (30) and Figure 9(c).

***N*(4)**: Replace training set with MDA training set. See Eq. (46).

***N*(5)**: Add DFF to ***N*(1)**.

***N*(6)**: Add DFF to ***N*(2)**.

***N*(7)**: Add DFF to ***N*(3)**.

***N*(8)**: Add DFF to ***N*(4)**.

Phase III: Model Validation

Split preprocessed dataset U_5 into test set Y and non-test set $U_5 - Y$.

for $w = 1{:}W$ % w is run index

Step III.A Random Split

(*continued on next page*)

Randomly split non-test set into training set $X^t(w)$, validation set $X^v(w)$.

Step III.B MDA on training set

for $i = 1{:}|X^t(w)|$

Training image: $x^t(i,w)$ and its ground truth labels $\hat{L}[x^t(i,w)]$.

$x^t(i,w)$: i-th training image in w-th run

$x^t(i,w) \to \overrightarrow{x^{tD}(i,w)}$. See Eq. (46).

end

Enhanced training set: $X^{tD}(w) = \{\overrightarrow{x^{tD}(i,w)} | i = 1, \cdots, |X^t(w)|\}$.

Enhanced training set labels: $\widehat{L^t}(w) = \{\hat{L}[x^t(i,w)] | i = 1, \cdots, |X^t(w)|\}$.

Step III.C Training on eight proposed models

for $n = 1{:}8$ % n is the model index

Create an initial model $N(n,w)$;

Train $N(n,w)$ with training set

Record the trained model $N(n,w)$

if $n == 4 \vee n == 8$

$N(n,w) = \text{trainnetwork}\{N(n), X^{tD}(w), \widehat{L^t}(w)\}$.

else

$N(n,w) = \text{trainnetwork}\{N(n), X^t(w), \widehat{L^t}(w)\}$.

end

end

Algorithm 2 (*continued*)

Step III.D Validation confusion matrix

Validation Set: $X^v(w)$ and its labels $\widehat{L^v}(w)$.

Validation prediction $Pred(n,w)$

$Pred(n,w) = \text{predict}[N(n,w), X^v(w)]$;

Validation performance $E^r(n,w)$

$E^r(n,w) = \text{compare}\left[pred(n,w), \widehat{L^v}(w)\right]$. See Eq. (54).

Step III.E Indicator Evaluation

for $m = 1{:}7$ % m is the indicator index

Extract $\{a_1(n,w), a_2(n,w), a_3(n,w), a_4(n,w)\}$ from confusion matrix $E^r(n,w)$.

Calculate indicators $v^m(n,w)$. See Eq. (55)-(62).

end

end

Phase IV: Compare and select the best model FGCNet on Validation Set

for $n = 1{:}8$

Calculate MSD of $N(n)$. See Eqs. (63)(64).

end

Select the best model n^* in terms of mean accuracy

$n^* = \text{argmax}\{\text{M}[v^4(n)]\}$.

$\text{FGCNet} = N(n^*)$.

Phase V: Report the test performance of the best model

Training set is $U_5 - Y$, and the ground truth is $\widehat{L^{nt}}$.

Test set is Y, and the ground truth is $\widehat{L^Y}$.

for $w = 1{:}W$ % w is run index

We initialized a random seed $S(w)$ at each run.

if $n^* == 4 \vee n^* == 8$

$N(n^*,w) = \text{trainnetwork}\left\{N(n^*), MDA[U_5 - Y], \widehat{L^{nt}}, S(w)\right\}$.

else

$N(n^*,w) = \text{trainnetwork}\left\{N(n^*), U_5 - Y, \widehat{L^{nt}}, S(w)\right\}$.

end

$Pred(n^*,w) = \text{predict}[N(n^*,w), Y]$;

$E^r(n^*,w) = \text{compare}\left[pred(n^*,w), \widehat{L^Y}\right]$.

End

Calculate MSD of $N(n^*)$ based on W runs of $E^r(n^*,w)$.

Output: The best model FGCNet $N(n^*)$ and its test performance.

$$E^i(w) = \begin{bmatrix} \frac{|X^v|}{2} & 0 \\ 0 & \frac{|X^v|}{2} \end{bmatrix}, \forall w \in 1, \cdots, W \tag{53}$$

where $\frac{|X^v|}{2}$ is because we have a balanced dataset. If the confusion matrix was run on the test set, then the diagonal elements turn to $\frac{|Y|}{2}$. In realistic situation, suppose we have a confusion matrix as

$$E^r(w) = \begin{bmatrix} a_1(w) & a_2(w) \\ a_3(w) & a_4(w) \end{bmatrix}, \forall w \in 1, \cdots, W \tag{54}$$

where the four variables $\{a_1(w), a_2(w), a_3(w), a_4(w)\}$ represent TP, FN, FP, and TN at w-th run, respectively. Here P means COVID-19 and N means healthy lung. TP means a COVID-19 image is classified correctly as COVID-19, FN means a COVID-19 image is wrongly classified as healthy, FP means a healthy lung is wrongly classified as COVID-19, and TN means a healthy lung is classified correctly as healthy. It is obvious that $0 \le a_k(w) \le \frac{|X^v|}{2}, \forall w \in 1, \cdots, W \wedge \forall k \in [1,2,3,4]$if confusion matrix

run on validation set.

Four simple measures $\{\nu^1(w), \nu^2(w), \nu^3(w), \nu^4(w)\}$ can be below, here ν^1 means sensitivity, ν^2 means specificity, ν^3 precision, and ν^4 accuracy.

$$\nu^1(w) = \frac{a_1(w)}{a_1(w) + a_2(w)} \tag{55}$$

$$\nu^2(w) = \frac{a_4(w)}{a_3(w) + a_4(w)} \tag{56}$$

$$\nu^3(w) = \frac{a_1(w)}{a_1(w) + a_3(w)} \tag{57}$$

$$\nu^4(w) = \frac{a_1(w) + a_4(w)}{a_1(w) + a_2(w) + a_3(w) + a_4(w)} \tag{58}$$

F1 score at w-th run is defined as $\nu^5(w)$

$$\nu^5(w) = \frac{2 \times a_1(w)}{2 \times a_1(w) + a_2(w) + a_3(w)} \tag{59}$$

Besides, F1 score can be expressed in the format of sensitivity and precision as $\nu^5(w) = 2 \times [\nu^3(w) \times \nu^1(w)] \div [\nu^3(w) + \nu^1(w)]$. The F1 score $\nu^5$is the harmonic mean of the precision ν^3 and sensitivity ν^1. The range of F1 score is [0, 1]. The highest possible value 1 indicates perfect precision ν^3 and sensitivity ν^1, and the lowest possible value 0 means either the precision ν^3 or the sensitivity is zero ν^1.

$$\nu^5 = \begin{cases} 1 & \nu^3 = 1 \wedge \nu^1 = 1 \\ 0 & \nu^3 = 0 \vee \nu^1 = 0 \end{cases} \tag{60}$$

Matthews correlation coefficient at w-th run (MCC) $\nu^6(w)$ is defined as

$$\nu^6(w) = \frac{a_4(w) \times a_1(w) - a_3(w) \times a_2(w)}{\sqrt{\varepsilon(w)}} \tag{61}$$

where $\varepsilon(w)$ is a temporary variable defined as $\varepsilon(w) = [a_3(w) + a_1(w)] \times [a_1(w) + a_2(w)] \times [a_4(w) + a_3(w)] \times [a_4(w) + a_2(w)]$.

Finally, Fowlkes–Mallows index (FMI) at w-th run $\nu^7(w)$ can be defined as:

$$\nu^7(w) = \sqrt{\frac{a_1(w)}{a_1(w) + a_3(w)} \times \frac{a_1(w)}{a_1(w) + a_2(w)}} \tag{62}$$

FMI can be expressed in the terms of sensitivity and precision as $\nu^7(w) = sqrt[\nu^3(w) \times \nu^1(w)]$.

After recording the seven indicators of all W runs, we can calculate the mean and standard deviation (MSD) of all m-th ($\forall m \in [1, 7]$) measures as

$$\mathrm{M}(\nu^m) = \frac{1}{W} \times \sum_{w=1}^{W} \nu^m(w) \tag{63}$$

$$\mathrm{SD}(\nu^m) = \sqrt{\frac{1}{W-1} \times \sum_{w=1}^{W} [\nu^m(w) - \mathrm{M}(\nu^m)]^2} \tag{64}$$

The final result over W runs were reported in the form of Mean ± SD format.

3.8. Proposed algorithm

Algorithm 2 presents the pseudocode of our algorithm, composed of one input, one output, and five phases. Phase I presents the preprocessing. Phase II presents how to construct the eight network models. Phase III shows the detailed procedures of W runs over validation set. Phase IV presents how to select the best network model based on validation performance. Phase V shows to calculate the test performance using the best network model.

Table 4
Hyperparameter setting.

Parameter	Value
α_s	10^{-5}
α_{rp}	0.5
n	2
a_g	2
η_{DA}	14
η_n	30
a_Z	25
χ^R	$\chi_1^R = -30^\circ, \chi_2^R = -28^\circ, \ldots, \chi_{15}^R = -2^\circ, \chi_{16}^R = +2^\circ, \ldots, \chi_{\eta_n}^R = +30^\circ$.
χ_m^N	0
χ_v^N	0.01
χ^H	$\chi_1^H = -0.15, \chi_2^H = -0.14, \ldots, \chi_{15}^H = -0.01, \chi_{16}^H = +0.01, \ldots, \chi_{\eta_n}^H = +0.15$.
χ^G	$\chi_1^G = 0.4, \chi_2^G = 0.44, \ldots, \chi_{15}^G = 0.96, \chi_{16}^G = 1.04, \ldots, \chi_{\eta_n}^G = 1.6$
χ^S	$\chi_1^S = 0.7, \chi_2^S = 0.72, \ldots, \chi_{15}^S = 0.98, \chi_{16}^S = 1.02, \cdots, \chi_{\eta_n}^S = 1.3$.
η_{EF}	421
u_c	7
u_f	2
$\lvert V \rvert$	256
d_0	120
d_1	60
d_2	120
	7
W	10

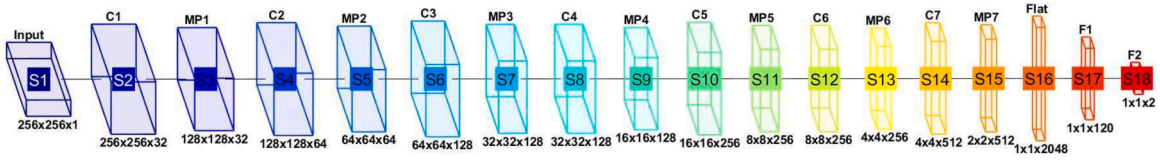
Table 5
Training, validation, and test set.

Set	Symbol	COVID C	Healthy H
Training	X^t	160	160
MDA Training	X^{tD}	67,360	67,360
Validation	X^v	64	64
Test	Y	96	96
Total	$U_5 = X^t \cup X^v \cup Y$	113	209

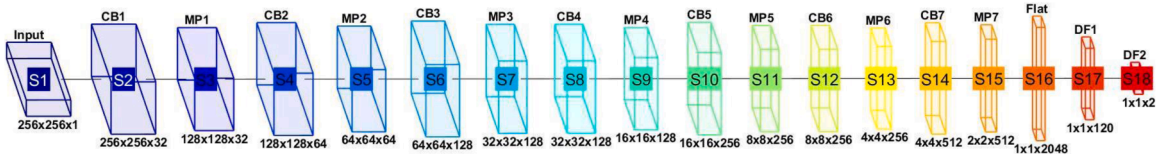
4. Experiments and results

4.1. Hyperparameter setting

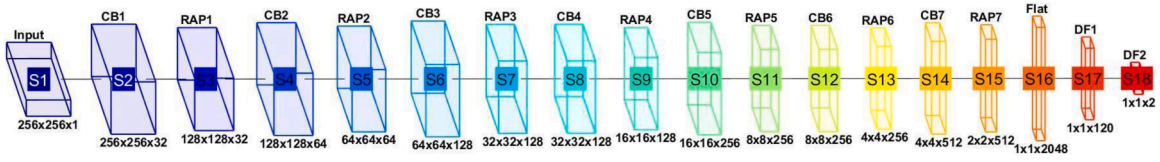
Table 4 shows the hyperparameter setting in this study. Most of the values are set by trial-and-error method. The stability factor is set as 10^{-5}. The retention probability is set as 0.5. The pooling size is set to 2. The rank threshold is set to 2. The number of DA techniques and new images per DA are set to 14 and 30, respectively. The maximum shift factor is 25, the mean and variance of noise injection is set to 0 and 0.01, respectively. The rotation parameter vector χ^R, horizontal shift parameter vector χ^H, Gamma correction parameter vector χ^G, scaling parameter vector $\chi^S$are all listed here. Enhanced factor is calculated as 421. The number of conv layers and fully connected layers are set as 7 and 2, respectively. Number of cluster centroids is set to 256. Feature dimension in GCN is set as $d_0 = d_2 = 120$, and $d_1 = 60$. The number of neighbors in KNN is set to 7. The number of runs $W$is 10 since it is a



(a) $N(1)$



(b) $N(2)$



(c) $N(3)$

Fig. 9. Block chart of first three proposed networks.

Table 6
Hyperparameters of $N(1)$.

Index	Layer	Hyperparameter	Size of activation map
1	Input		256 × 256 × 1
2	Conv-1	32 3 × 3	256 × 256 × 32
3	P-1	/2	128 × 128 × 32
4	Conv-2	64 3 × 3	128 × 128 × 64
5	P-2	/2	64 × 64 × 64
6	Conv-3	128 3 × 3	64 × 64 × 128
7	P-3	/2	32 × 32 × 128
8	Conv-4	128 3 × 3	32 × 32 × 128
9	P-4	/2	16 × 16 × 128
10	Conv-5	256 3 × 3	16 × 16 × 256
11	P-5	/2	8 × 8 × 256
12	Conv-6	256 3 × 3	8 × 8 × 256
13	P-6	/2	4 × 4 × 256
14	Conv-7	512 3 × 3	4 × 4 × 512
15	P-7	/2	2 × 2 × 512
16	Flatten		1 × 1 × 2048
17	FCL-1	120 × 2048 120 × 1	1 × 1 × 120
18	FCL-2	2 × 120 2 × 1	1 × 1 × 2

default value used in many other publications.

Table 5 shows the training, MDA training, validation and test set. Where we can see the total size of training set is $|X^t| = 320$. The total size of enhanced MDA training set is $|X^{tD}| = 134,720$. The validation set's and test set's sizes are $|X^v| = 128$, and $|Y| = 192$, respectively. In total, the size of the whole dataset is $|U_5| = |X^t| + |X^v| + |Y| = 640$.

4.2. Base network configuration

The top row of Fig. 9(a) shows the activation maps of the proposed base network $N(1)$. Here, the size of input is $S1 = 256 \times 256 \times 1$, the output of the first conv layer (C1) is $S2 = 256 \times 256 \times 32$. Then after the first pooling (P1), the output is $S3 = 128 \times 128 \times 32$, We repeat the conv layers in total seven times, and the output is $S15 = 2 \times 2 \times 512$, $S15$ was flattened into one column vector $S16 = 1 \times 1 \times 2048$, and passed into two fully-connected blocks (first two block contains FCL and ReLU, last block contains FCL and softmax), and the outputs are $S17 = 1 \times 1 \times 120$, and $S18 = 1 \times 1 \times 2$. All the 18 matrices $S(k)$, $k \in [1, 18]$ correspond to the cuboids in Fig. 9(b). Note $S17$ will be used as IIR features. The hyperparameters of $N(1)$ are presented in Table 6. Based on $N(1)$, we can create the rest seven networks.

4.3. Illustration of MDA

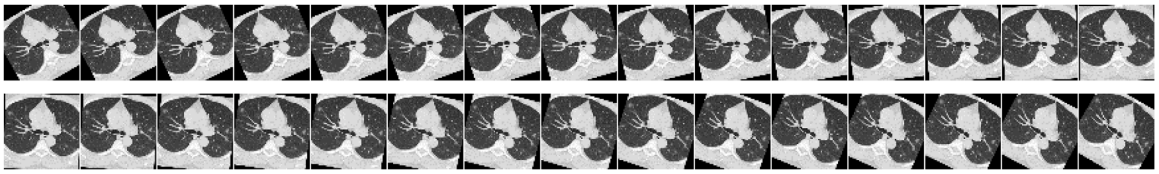
Fig. 10 shows the MDA results. The original image is Fig. 2(a). We can observe that one image will generate 421 new images. This is why we called our algorithm multiple-way data augmentation (MDA).

4.4. Comparison among proposed networks

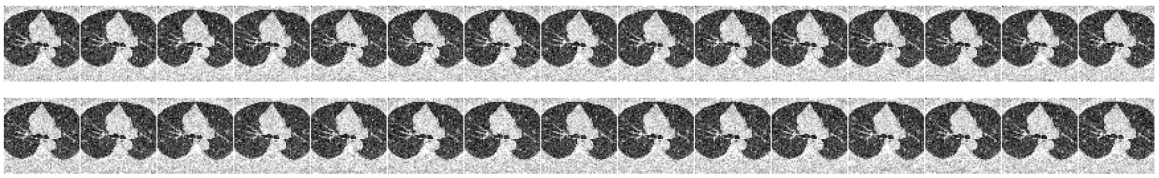
Table 7 gives the 10 runs of results using $N(1)$ to $N(4)$. $N(1)$ is BN, $N(2)$ is BDBN, $N(3)$ is BDRBN, and $N(4)$ is BDRMBN. Table 7 clearly shows that $N(1)$ model yielded the following seven performances as: $\nu^1_{N1} = 89.22 \pm 2.38$, $\nu^2_{N1} = 92.50 \pm 2.31$, $\nu^3_{N1} = 92.31 \pm 2.10$, $\nu^4_{N1} = 90.86 \pm 1.28$, $\nu^5_{N1} = 90.70 \pm 1.31$, $\nu^6_{N1} = 81.82 \pm 2.53$, $\nu^7_{N1} = 90.73 \pm 1.30$. Definition of νcan be found in Section 3.7.

For $N(2)$, the performances improved as $\nu^1_{N2} = 94.22 \pm 1.05$, $\nu^2_{N2} = 94.69 \pm 2.47$, $\nu^3_{N2} = 94.71 \pm 2.29$, $\nu^4_{N2} = 94.45 \pm 1.40$, $\nu^5_{N2} = 94.45 \pm 1.34$, $\nu^6_{N2} = 88.93 \pm 2.78$, $\nu^7_{N2} = 94.46 \pm 1.33$. Comparing the results of BN as $N(1)$, we can observe the effectiveness of using batch normalization and dropout.

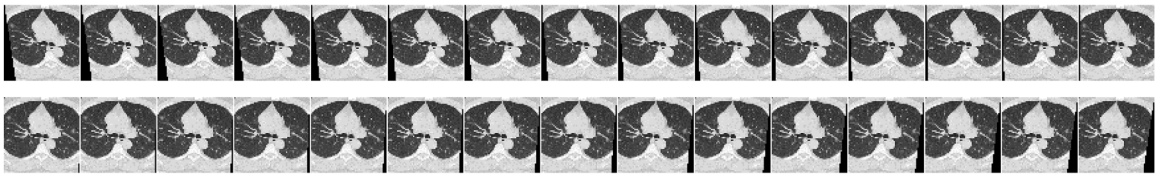
Furthermore, N(3) yields the performances as $\nu^1_{N3} = 94.53 \pm 1.69$, $\nu^2_{N3} = 95.47 \pm 2.14$, $\nu^3_{N3} = 95.47 \pm 2.05$, $\nu^4_{N3} = 95.00 \pm 1.23$, $\nu^5_{N3} = 94.98 \pm 1.23$, $\nu^6_{N3} = 90.04 \pm 2.47$, $\nu^7_{N3} = 94.99 \pm 1.23$. Comparing the seven indicator performances between BDBN of $N(2)$ and BDRBN of N



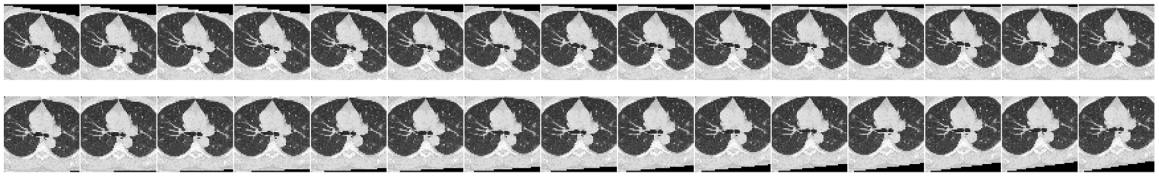
(a) Rotation



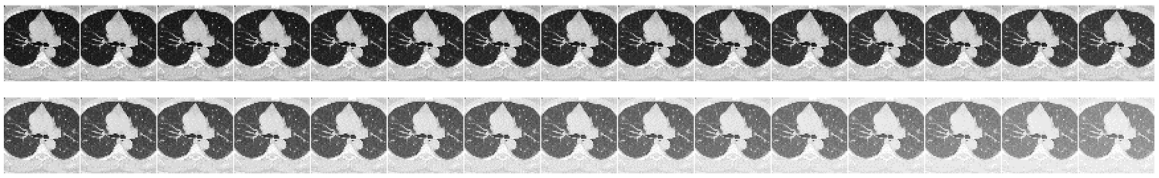
(b) Noise Injection



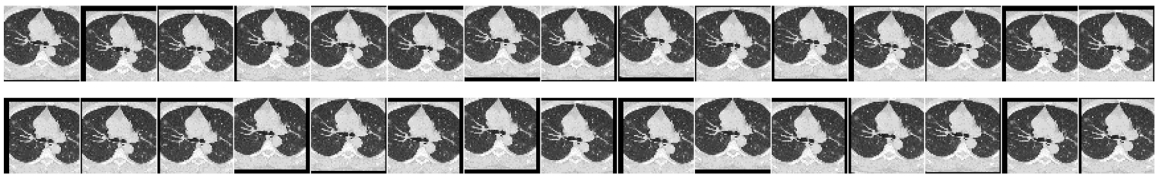
(c) HS Transform



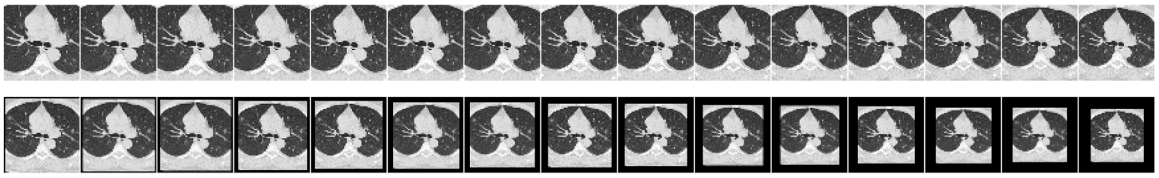
(d) VS Transform



(e) GC



(f) Random Translation

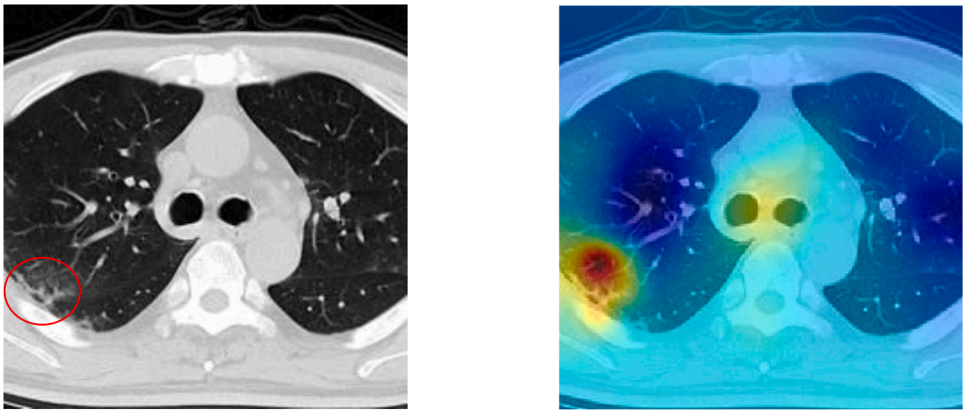


(g) Scaling

Fig. 10. Results of proposed MDA.

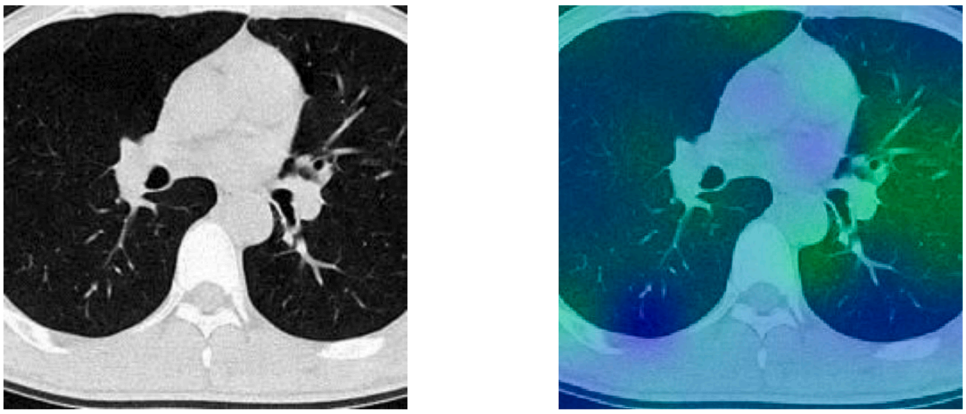
Table 7
Comparison among N(1–4) over validation set.

N(1)	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	89.06	93.75	93.44	91.41	91.20	82.90	91.23
2	93.75	92.19	92.31	92.97	93.02	85.95	93.03
3	90.63	87.50	87.88	89.06	89.23	78.16	89.24
4	89.06	92.19	91.94	90.63	90.48	81.29	90.49
5	89.06	92.19	91.94	90.63	90.48	81.29	90.49
6	90.63	92.19	92.06	91.41	91.34	82.82	91.34
7	89.06	93.75	93.44	91.41	91.20	82.90	91.23
8	89.06	95.31	95.00	92.19	91.94	84.54	91.98
9	87.50	90.63	90.32	89.06	88.89	78.16	88.90
10	84.38	95.31	94.74	89.84	89.26	80.17	89.41
MSD	89.22±2.38	92.50±2.31	92.31±2.10	90.86±1.28	90.70±1.31	81.82±2.53	90.73±1.30
N(2)	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	93.75	95.31	95.24	94.53	94.49	89.07	94.49
2	95.31	96.88	96.83	96.09	96.06	92.20	96.07
3	93.75	95.31	95.24	94.53	94.49	89.07	94.49
4	95.31	93.75	93.85	94.53	94.57	89.07	94.58
5	95.31	96.88	96.83	96.09	96.06	92.20	96.07
6	95.31	95.31	95.31	95.31	95.31	90.63	95.31
7	93.75	92.19	92.31	92.97	93.02	85.95	93.03
8	92.19	96.88	96.72	94.53	94.40	89.16	94.43
9	93.75	89.06	89.55	91.41	91.60	82.90	91.63
10	93.75	95.31	95.24	94.53	94.49	89.07	94.49
MSD	94.22±1.05	94.69±2.47	94.71±2.29	94.45±1.40	94.45±1.34	88.93±2.78	94.46±1.33
N(3)	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	96.88	92.19	92.54	94.53	94.66	89.16	94.68
2	95.31	93.75	93.85	94.53	94.57	89.07	94.58
3	93.75	93.75	93.75	93.75	93.75	87.50	93.75
4	93.75	95.31	95.24	94.53	94.49	89.07	94.49
5	93.75	98.44	98.36	96.09	96.00	92.29	96.03
6	95.31	98.44	98.39	96.88	96.83	93.80	96.84
7	96.88	95.31	95.38	96.09	96.12	92.20	96.13
8	92.19	96.88	96.72	94.53	94.40	89.16	94.43
9	95.31	96.88	96.83	96.09	96.06	92.20	96.07
10	92.19	93.75	93.65	92.97	92.91	85.95	92.92
MSD	94.53±1.69	95.47±2.14	95.47±2.05	95.00±1.23	94.98±1.23	90.04±2.47	94.99±1.23
N(4)	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	96.88	98.44	98.41	97.66	97.64	95.32	97.64
2	96.88	96.88	96.88	96.88	96.88	93.75	96.88
3	96.88	96.88	96.88	96.88	96.88	93.75	96.88
4	98.44	98.44	98.44	98.44	98.44	96.88	98.44
5	93.75	98.44	98.36	96.09	96.00	92.29	96.03
6	96.88	96.88	96.88	96.88	96.88	93.75	96.88
7	95.31	98.44	98.39	96.88	96.83	93.80	96.84
8	95.31	95.31	95.31	95.31	95.31	90.63	95.31
9	95.31	93.75	93.85	94.53	94.57	89.07	94.58
10	96.88	96.88	96.88	96.88	96.88	93.75	96.88
MSD	96.25±1.32	97.03±1.55	97.03±1.52	96.64±1.11	96.63±1.10	93.30±2.21	96.63±1.10



(a) Original image (b) Heat map

Fig. 11. Grad-CAM result on a covid-19 case.



(a) Original image (b) heat map

Fig. 12. Grad-CAM result on a normal case.

Table 8
Comparison among $N(5–8)$ over validation set.

$N(5)$	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	96.88	90.63	91.18	93.75	93.94	87.67	93.98
2	90.63	93.75	93.55	92.19	92.06	84.42	92.08
3	93.75	92.19	92.31	92.97	93.02	85.95	93.03
4	95.31	89.06	89.71	92.19	92.42	84.54	92.47
5	95.31	90.63	91.04	92.97	93.13	86.03	93.15
6	90.63	90.63	90.63	90.63	90.63	81.25	90.63
7	92.19	92.19	92.19	92.19	92.19	84.38	92.19
8	95.31	89.06	89.71	92.19	92.42	84.54	92.47
9	89.06	93.75	93.44	91.41	91.20	82.90	91.23
10	95.31	90.63	91.04	92.97	93.13	86.03	93.15
MSD	93.44±2.64	91.25±1.68	91.48±1.37	92.34±0.89	92.41±0.98	84.77±1.80	92.44±0.98
$N(6)$	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	96.88	96.88	96.88	96.88	96.88	93.75	96.88
2	92.19	96.88	96.72	94.53	94.40	89.16	94.43
3	95.31	95.31	95.31	95.31	95.31	90.63	95.31
4	98.44	93.75	94.03	96.09	96.18	92.29	96.21
5	98.44	93.75	94.03	96.09	96.18	92.29	96.21
6	95.31	95.31	95.31	95.31	95.31	90.63	95.31
7	95.31	95.31	95.31	95.31	95.31	90.63	95.31
8	93.75	95.31	95.24	94.53	94.49	89.07	94.49
9	93.75	95.31	95.24	94.53	94.49	89.07	94.49
10	95.31	93.75	93.85	94.53	94.57	89.07	94.58
MSD	95.47±2.01	95.16±1.15	95.19±1.04	95.31±0.82	95.31±0.86	90.66±1.66	95.32±0.86
$N(7)$	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	95.31	95.31	95.31	95.31	95.31	90.63	95.31
2	93.75	96.88	96.77	95.31	95.24	90.67	95.25
3	96.88	93.75	93.94	95.31	95.38	90.67	95.40
4	96.88	95.31	95.38	96.09	96.12	92.20	96.13
5	95.31	95.31	95.31	95.31	95.31	90.63	95.31
6	95.31	95.31	95.31	95.31	95.31	90.63	95.31
7	95.31	95.31	95.31	95.31	95.31	90.63	95.31
8	98.44	95.31	95.45	96.88	96.92	93.80	96.93
9	96.88	95.31	95.38	96.09	96.12	92.20	96.13
10	96.88	95.31	95.38	96.09	96.12	92.20	96.13
MSD	96.09±1.33	95.31±0.74	95.36±0.67	95.70±0.55	95.72±0.57	91.42±1.11	95.72±0.57
$N(8)$	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	98.44	98.44	98.44	98.44	98.44	96.88	98.44
2	98.44	96.88	96.92	97.66	97.67	95.32	97.68
3	96.88	98.44	98.41	97.66	97.64	95.32	97.64
4	96.88	95.31	95.38	96.09	96.12	92.20	96.13
5	96.88	96.88	96.88	96.88	96.88	93.75	96.88
6	100.00	98.44	98.46	99.22	99.22	98.45	99.23
7	96.88	96.88	96.88	96.88	96.88	93.75	96.88
8	100.00	90.63	91.43	95.31	95.52	91.03	95.62
9	95.31	98.44	98.39	96.88	96.83	93.80	96.84
10	96.88	98.44	98.41	97.66	97.64	95.32	97.64
MSD	97.66±1.52	96.88±2.44	96.96±2.21	97.27±1.12	97.28±1.08	94.58±2.17	97.30±1.06

Table 9
Comparison of eight network models.

Model	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
N(1)	89.22±2.38	92.50±2.31	92.31±2.10	90.86±1.28	90.70±1.31	81.82±2.53	90.73±1.30
N(2)	94.22±1.05	94.69±2.47	94.71±2.29	94.45±1.40	94.45±1.34	88.93±2.78	94.46±1.33
N(3)	94.53±1.69	95.47±2.14	95.47±2.05	95.00±1.23	94.98±1.23	90.04±2.47	94.99±1.23
N(4)	96.25±1.32	97.03±1.55	97.03±1.52	96.64±1.11	96.63±1.10	93.30±2.21	96.63±1.10
N(5)	93.44±2.64	91.25±1.68	91.48±1.37	92.34±0.89	92.41±0.98	84.77±1.80	92.44±0.98
N(6)	95.47±2.01	95.16±1.15	95.19±1.04	95.31±0.82	95.31±0.86	90.66±1.66	95.32±0.86
N(7)	96.09±1.33	95.31±0.74	95.36±0.67	95.70±0.55	95.72±0.57	91.42±1.11	95.72±0.57
N(8)	97.66±1.52	96.88±2.44	96.96±2.21	97.27±1.12	97.28±1.08	94.58±2.17	97.30±1.06

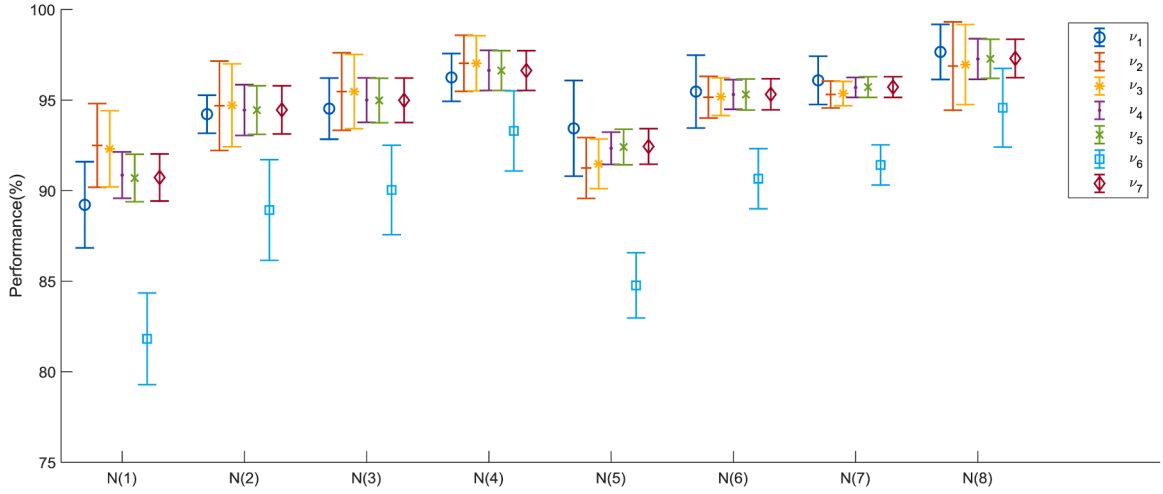


Fig. 13. Error bar of proposed eight models.

(3), we can conclude that rank-based average pooling gives significant better performance than using max pooling in *N*(2).

Finally, checking *N*(4) in Table 7 obtained the performance as $\nu^1_{N4} = 96.25 \pm 1.32$, $\nu^2_{N4} = 97.03 \pm 1.55$, $\nu^3_{N4} = 97.03 \pm 1.52$, $\nu^4_{N4} = 96.64 \pm 1.11$, $\nu^5_{N4} = 96.63 \pm 1.10$, $\nu^6_{N4} = 93.30 \pm 2.21$, $\nu^7_{N4} = 96.63 \pm 1.10$. The increase of performances of BDRMBN of *N*(4) compared to BDRBN of *N*(3) indicate that multiple-way data augmentation can help improving the performance of the AI classifier.

4.5. Visual explanation

We used Gradient-weighted Class Activation Mapping (Grad-CAM) [49] to visually show why our model, BDRMBN of *N*(4), can make the decision. Grad-CAM uses the gradient of the classification score with respect to the convolutional features determined by the network in order to understand which parts of the image are most important for classification. The "jet" pseudo-color was used in the heat map. Hence, red colors mean important areas for AI diagnosis, and blue colors mean unimportant areas for AI diagnosis.

Fig. 11 shows the Grad-CAM heat map results of a COVID-19 CCT

Table 10
Performance of proposed *N*(8) FGCNet on test set.

Run	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
1	98.96	96.88	96.94	97.92	97.94	95.85	97.94
2	96.88	96.88	96.88	96.88	96.88	93.75	96.88
3	96.88	98.96	98.94	97.92	97.89	95.85	97.90
4	96.88	95.83	95.88	96.35	96.37	92.71	96.37
5	96.88	95.83	95.88	96.35	96.37	92.71	96.37
6	97.92	95.83	95.92	96.88	96.91	93.77	96.91
7	95.83	95.83	95.83	95.83	95.83	91.67	95.83
8	100.00	97.92	97.96	98.96	98.97	97.94	98.97
9	96.88	93.75	93.94	95.31	95.38	90.67	95.40
10	100.00	97.92	97.96	98.96	98.97	97.94	98.97
MSD	97.71±1.46	96.56±1.48	96.61±1.43	97.14±1.26	97.15±1.25	94.29±2.52	97.16±1.25

Table 11
Comparison with state-of-the-art approaches.

Approach	ν^1	ν^2	ν^3	ν^4	ν^5	ν^6	ν^7
RBFNN [4]	67.08	74.48	72.52	70.78	69.64	41.74	69.64
KELM [5]	57.29	61.46	59.83	59.38	58.46	18.81	58.46
ELM-BA [6]	57.08 ±3.86	72.40 ±3.03	67.48 ±1.65	64.74 ±1.26	61.75 ±2.24	29.90 ±2.45	61.76 ±2.24
RCBBO [7]	69.48 ±4.47	81.15 ±3.16	78.79 ±1.80	75.31 ±0.82	73.72 ±1.86	51.10 ±1.28	73.93 ±1.66
6L-CLF [8]	81.04 ± 2.90	79.27 ± 2.21	79.70 ± 1.27	80.16 ± 0.85	80.31 ± 1.13	60.42 ± 1.73	80.35 ± 1.15
GoogLeNet [9]	76.88 ± 3.92	83.96 ± 2.29	82.84 ± 1.58	80.42 ± 1.40	79.65 ± 1.92	61.10 ± 2.62	79.65 ± 1.91
ResNet-18 [10]	78.96 ± 2.90	89.48 ± 1.64	88.30 ± 1.50	84.22 ± 1.23	83.31 ± 1.53	68.89 ± 2.33	83.32 ± 1.53
RN-50-AD [11]	83.96 ±3.19	90.31 ±2.14	89.73 ±1.78	87.14 ±1.07	86.69 ±1.34	74.50 ±2.00	86.77 ±1.27
SMO [12]	93.23 ±1.72	95.52 ±1.30	95.44 ±1.22	94.38 ±0.64	94.31 ±0.68	88.80 ±1.27	93.23 ±1.72
CSSNet [13]	92.08 ±1.01	93.33 ±2.61	93.32 ±2.40	92.71 ±0.95	92.67 ±0.85	85.47 ±1.93	92.69 ±0.86
GGNet [14]	94.38 ±2.09	90.63 ±2.02	91.00 ±1.77	92.50 ±1.16	92.64 ±1.14	85.11± 2.34	92.66 ±1.15
COVNet [15]	90.83 ±2.45	96.67 ±0.82	96.47 ±0.82	93.75 ±1.13	93.55 ±1.24	87.68 ±2.14	93.60 ±1.21
NiNet [16]	95.31 ±1.13	77.19 ±2.62	80.73 ±1.70	86.25 ±1.02	87.40 ±0.79	73.76 ±1.73	87.71 ±0.73
FCONet [17]	93.54 ±1.69	96.04 ±1.46	95.96 ±1.38	94.79 ±0.92	94.72 ±0.94	89.64 ±1.83	94.74 ±0.94
DeCovNet [18]	90.00 ±1.49	90.52 ±1.99	90.52 ±1.72	90.26 ±0.65	90.24 ±0.61	80.56 ±1.32	90.25 ±0.61
FGCNet (Ours)	97.71 ±1.46	96.56 ±1.48	96.61 ±1.43	97.14 ±1.26	97.15 ±1.25	94.29 ±2.52	97.16 ±1.25

slice. On the left part of Fig. 11(a), the red circle delineated the lesions, where we can see the GGO occurs. Fig. 11(b) show the corresponding heat map. We can see here AI pay the most attention on the GGO lesion (See the red circle on the Fig. 11a), indicating AI successfully capture the GGO lesions. Secondly, AI pay somewhat attention on the tracheae (in the middle of the Fig. 11b). The reason may be COVID-19 influences the grayscale values of tracheae tissues, where we can see yellow blots on the middle areas in Fig. 11(b).

Fig. 12 shows the Grad-CAM heat map of a normal CCT slice. Our AI model scans through the whole image and does not find any strong activations (suspicious areas). Hence, the AI model judges this image "healthy".

4.6. Effect of deep feature fusion

We compared the performance of using deep feature fusion against not using deep feature fusion. The comparison was done on the validation set. The results using $N(5–8)$ are presented in Table 8. Comparing Tables 7 and 8, we can find that using DFF can increase the classification performance.

For clear view, the results using all the proposed eight models are presented in Table 9. $N(1–4)$ did not use DFF while $N(5–8)$ added DFF to the corresponding networks (See Table 3). Fig. 13 shows the mean and standard deviation of the eight neural network models.

Comparing BDRMBN of $N(4)$ against DBDRMBN of $N(8)$, we can see that adding DFF can improve all the five indicators (ν^1, ν^4, ν^5, ν^6, ν^7) except (ν^2, ν^3). This indicates the DFF is effective in increasing the classifier's performance.

The same scenario can be observed by comparing BN of $N(1)$ against DBN of $N(5)$, comparing BDBN of $N(2)$ DBDBN of $N(6)$, and comparing BDRBN of $N(3)$ and DBDRBN of $N(7)$. The reason why DFF can improve the performance is because DFF fuses features from GCN, which learns the relation-awareness relationships (RARs) among the validation images. Hence, classifiers with DFFs were more accurate than those without DFFs.

Besides, from Table 9 and Fig. 13, we can find the optimal $n^* = 8$. That means, $N(8)$ achieved the best performance among all our proposed eight network models. This falls within our expectation, because DBDRMBN of $N(8)$ fuses RAR features from GCN with features from BDRMBN of $N(4)$. This feature fusion helps our classifier obtained the best performance. As designed in Section 3.6, the best model among all eight proposed networks is dubbed as FGCNet, indicating the fusion of GCN and CNN networks.

4.7. Comparison to state-of-the-art approaches

Above performances ran on the validation set. Now we run the best model FGCNet, i.e., DBDRMBN model on the test set, and report its performance. The results are shown in Table 10. As is shown, $\nu^1_{N8} = 97.71 \pm 1.46$, $\nu^2_{N8} = 96.56 \pm 1.48$, $\nu^3_{N8} = 96.61 \pm 1.43$, $\nu^4_{N8} = 97.14 \pm 1.26$, $\nu^5_{N8} = 97.15 \pm 1.25$, $\nu^6_{N8} = 94.29 \pm 2.52$, $\nu^7_{N8} = 97.16 \pm 1.25$. Comparing Tables 9and 10, we observe that the performances on validation set and test set are quite similar, only the test performance is slightly less than the validation performance.

We compared our DBDRMBN method, i.e., $N(8)$ model, with 15 state-of-the-art approaches: RBFNN [4], KELM [5], ELM-BA [6], RCBBO [7], 6L-CLF [8], GoogLeNet [9], ResNet-18 [10], RN-50-AD [11], SMO [12], CSSNet [13], GGNet [14], COVNet [15], NiNet [16], FCONet [17], and

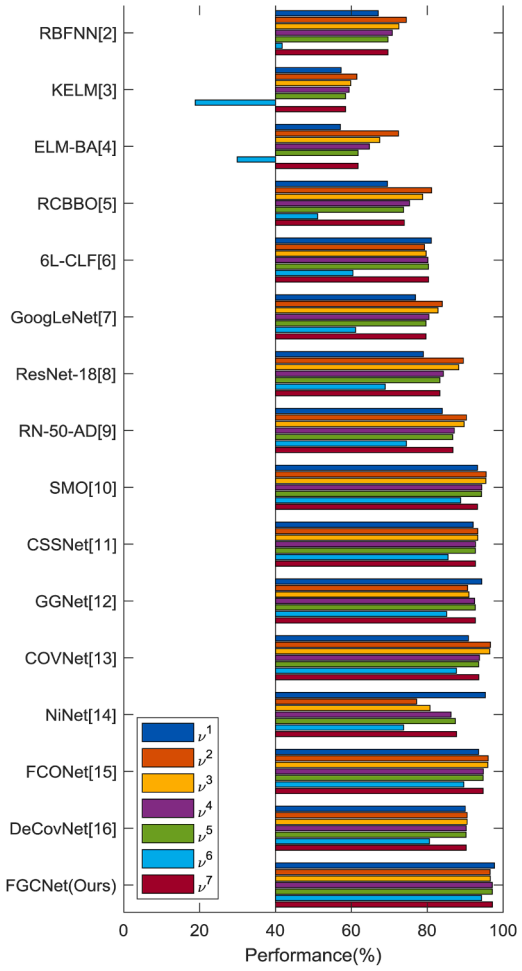


Fig. 14. Comparison plot.

DeCovNet [18]. All the methods were compared on the test set of our 640-image dataset. Some methods were not proposed for detecting COVID-19, some methods work on Chest X-ray images, and some are for multi-class classification, nevertheless, we modified and transferred their methods to our CCT images. The comparison and its plot are presented in Table 11 and Fig. 14. The results in Table 11 and Fig. 14 show that the proposed FGCNet (DBDRMBN) achieved the best results among all algorithms.

5. Conclusions

This paper proposed a total eight network models for COVID-19 detection in CCTs. Our experiments showed our DBDRMBN of $N(8)$ can achieve the best performance among all the eight proposed models. This model is also named as FGCNet for short, and it obtains superior results to other 15 state-of-the-art approaches.

The reason why our FGCNet has the best performance is (i) because the FGCNet (DBDRMBN) is a deep feature fusion combination of an improved CNN model $N(4)$ of BDRMBN and a GCN model. Here, $N(4)$ (BDRMBN) helps to extract learnt individual image-level representations, while GCN helps to extract learnt relation-aware representation. Finally, the DFF strategy help fuse those two types of features. (ii) The proposed $N(4)$ (BDRMBN) is a novel neural network trained with its structure developed and weights trained from scratch. Besides, BDRMBN used several advanced techniques, such as batch normalization, dropout, rank-based average pooling, and multiple-way data augmentation.

The shortcomings of this proposed FGCNet can only handles CCT images. For chest X-ray images and other sources of data, the FGCNet may not work correctly. So, it is necessary to define a network that can fuse different sources/modalities of data and give an overall decision. Another shortcoming is this proposed FGCNet is not verified by a strict clinical test.

The future work directions are: (i) Expand the dataset and test our algorithm on different sources of COVID-19, such as the combination of X-ray and CT. (ii) Test other fuse strategies of CNN and GCN. (iii) Try to employ deeper GCN, and test whether deeper GCN will help improve the performance.

CRediT authorship contribution statement

Shui-Hua Wang: Conceptualization, Methodology, Software, Validation, Data curation, Writing - original draft, Investigation, Data curation. **Vishnu Varthanan Govindaraj:** Formal analysis, Writing - original draft, Writing - review & editing. **Juan Manuel Górriz:** Writing - original draft, Writing - review & editing, Supervision, Funding acquisition. **Xin Zhang:** Writing - original draft, Writing - review & editing. **Yu-Dong Zhang:** Resources, Formal analysis, Investigation, Data curation, Writing - review & editing, Supervision, Project administration, Funding acquisition.

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.

Acknowledgement

This paper is partially supported by Royal Society International Exchanges Cost Share Award, UK (RP202G0230); Hope Foundation for Cancer Research, UK (RM60G0680); Medical Research Council Confidence in Concept Award, UK (MC_PC_17171); British Heart Foundation Accelerator Award, UK; and MINECO/FEDER under the RTI2018-098913-B100 and A-TIC-080-UGR18 projects.

Appendix A

Table 12.

Table 12
Abbreviation list.

Abbreviation	Full name
CCT	chest computed tomography
GGO	ground-glass opacity
CAP	community acquired pneumonia
HS	hyperspectral
CNN	convolutional neural network
GCN	graph convolutional network
IIR	individual image-level representation
RAR	relation-aware representation
MV	majority voting
NLAF	non-linear activation function
ICS	internal covariant shift
BAN	Batch Normalization
DO	Dropout
L2P	l_2 pooling
MDA	Multiple-way data augmentation
SSD	small-size dataset
LG	lack of generation
KMC	k-means clustering
kNN	k-nearest neighbors
BN	Base Network
FGCNet	Fusion of GCN and CNN Network

Appendix B

Table 13

Table 13
Symbol list.

Symbol	Meaning
L	Labeling from each individual expert
$\widehat{L}$	Final labeling
U	Dataset
U_1	Raw dataset
U_5	Preprocessed dataset
$\|U\|$	Number of samples in the dataset
X^t	Training set
X^v	Validation Set
Y	Test set
μ_{min}	Min grayscale value
μ_{max}	Max grayscale value
W	Width
H	Height
C	Channel
α_s	Stability factor
α_{rp}	Retention probability
μ_e	Empirical mean
ϕ_e	Empirical variance
μ_p	Population mean
ϕ_p	Population variance
Ψ	Region to be pooled
n	Pooling size.
P	Pooling output
a_g	Rank threshold
χ	Various data augmentation factor
η_{DA}	Number of MDA techniques
η_n	Number of new images generated for each DA
a_Z	Maximum shift factor
	Mirror function
η_{EF}	Enhance factor
	Concatenation
u_c	Number of conv layers
u_f	Number of FCL layers
$\|V\|$	Cluster centroids
	Number of neighbors in kNN
W	Number of runs
w	Run index

Appendix C

Fig. 4 shows a simplistic CNN example with four FCL layers. Suppose we have $C(k), k = 1,2,3,4$neurons at k-th layer, and assume $C(1) = 6$, $C(2) = 10$, $C(3) = 6$, $C(4) = 4$. Thus, we have in total $\sum_{k=1}^{4} N(k) = 26$nodes. Suppose we do not consider incoming and outgoing weights, and do not consider the number of biases, the size of learnable weights $C(i, j)$, where $(i,j) = \{(1,2),(2,3),(3,4)\}$as number of weights between layer i and layer j before dropout, roughly calculating, can be written as $C(1,2) = 6 \times 10 = 60$, $C(2,3) = 10 \times 6 = 60$, $C(3,4) = 6 \times 4 = 24$. In total, we have the total number of learnable weights before dropout as $C = \sum_{i,j} C(i,j) = 144$.

Using $\alpha_{rp} = 0.5$, the size of learnable weights after dropout between layer $i$and layer j is symbolized as $C^D(i, j)$, and we can calculate the total number of learnable weights as $C^D = \sum_{i,j} C^{'}(i,j) = 15 + 15 + 6 = 36$. The compression ratio of learnable weights (CRLW), roughly, can be calculated by $C^D/C = 36/44 = 0.25$,

Appendix D

Using Fig. 5 as a simplistic example, and assuming the region $\Psi(1, 1)$ at 1st row 1st column of the input is chosen. For the sake of format, we use its row-vector $\Psi(1,1) \leftarrow \overrightarrow{\Psi(1,1)}$to represent in this paragraph, so we have $\Phi(1,1) = (2.5,5,1.6,1.1)$. We can calculate the results of L2P as $P^{L2P}(\Psi(1,1)) = \text{sqrt}((2.5^2 + 5^2 + 1.6^2 + 1.1^2)/4) = \sqrt{35.02/4} = \sqrt{8.755} = 2.96$. The pooling result of AP is $P^{AP}(\Psi(1,1)) = \text{average}(\Psi(1,1)) = (2.5 + 5 + 1.6 + 1.1) \div 4 = 2.55$. The MP result is $P^{MP}(\Psi(1,1)) = \max(\Psi(1,1)) = \max(2.5,5,1.6,1.1) = 5$. The rank matrix of $\Psi(1, 1)$ also expressed in row-vector format $R_{\Psi(1,\,1)} \leftarrow vec(R_{\Psi(1,\,1)})$, we have $R_{\Psi(1,1)} = (2,1,3,4)$. The RAP result is $P^{RAP}(\Psi(1,1)) = (2.5 + 5) \div 2 = 3.75$.

References

[1] O.A. Ataguba, J.E. Ataguba, Social determinants of health: the role of effective communication in the COVID-19 pandemic in developing countries, Glob. Health Action 13 (2020) 5. Article ID: 1788263Dec.

[2] B.B. Ketencioglu, F. Yigit, M. Almadqa, N. Tutar, I. Yilmaz, Non-infectious diseases compatible With COVID-19 Pneumonia, Cureus 12 (2020) 5. Article ID: e9989, Aug.

[3] M.D. Hope, C.A. Raptis, A. Shah, M.M. Hammer, T.S. Henry, S. Six, A role for CT in COVID-19? What data really tell us so far, Lancet 395 (2020) 1189–1190. Apr.

[4] Z. Lu, A pathological brain detection system based on radial basis function neural network, J. Med. Imaging Health Inform. 6 (2016) 1218–1222.

[5] J. Yang, A pathological brain detection system based on kernel based ELM, Multimed. Tools Appl.. 77 (2018) 3715–3728.

[6] S. Lu, A pathological brain detection system based on extreme learning machine optimized by bat algorithm, CNS Neurol. Disord.-Drug Targets 16 (2017) 23–29.

[7] P. Li, G. Liu, Pathological brain detection via wavelet packet Tsallis entropy and real-coded biogeography-based optimization, Fundam. Inform. 151 (2017) 275–291.

[8] X. Jiang, Chinese sign language fingerspelling recognition via six-layer convolutional neural network with leaky rectified linear units for therapy and rehabilitation, J. Med. Imaging Health Inform.. 9 (2019) 2031–2038.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Boston, MA, USA, 2015, pp. 1–9.

[10] M.H. Guo, Y.Z. Du, Classification of thyroid ultrasound standard plane images using ResNet-18 networks, in: IEEE 13th International Conference on Anti-Counterfeiting, Security, and Identification, Xiamen, China, 2019, pp. 324–328.

[11] L.V. Fulton, D. Dolezel, J. Harrop, Y. Yan, C.P. Fulton, Classification of Alzheimer's disease with and without imagery using gradient boosted machines and ResNet-50, Brain Sci. 9 (2019) 16. Article ID: 212, Sep.

[12] M. Togacar, B. Ergen, Z. Comert, COVID-19 detection using deep learning models to exploit social mimic optimization and structured chest X-ray images using fuzzy color and stacking approaches, Comput. Biol. Med. 121 (2020) 12. Article ID: 103805, Jun.

[13] J.P. Cohen, L. Dao, P. Morrison, K. Roth, Y. Bengio, B.Y. Shen, et al., Predicting COVID-19 pneumonia severity on chest X-ray with deep learning, Cureus 12 (2020) 10. Article ID: e9448, Jul.

[14] M. Loey, F. Smarandache, N.E.M. Khalifa, Within the lack of chest COVID-19 X-ray dataset: a novel detection model based on GAN and deep transfer learning, Symmetry-Basel 12 (2020) 19. Article ID: 651, Apr.

[15] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang, B. Kong, et al., Using artificial intelligence to detect COVID-19 and community-acquired pneumonia based on pulmonary CT: evaluation of the diagnostic accuracy, Radiology 296 (2020) E65–E71.

[16] Q.Q. Ni, Z.Y. Sun, L. Qi, W. Chen, Y. Yang, L. Wang, et al., A deep learning approach to characterize 2019 coronavirus disease (COVID-19) pneumonia in chest CT images, Eur. Radiol. (2020) 11. https://link.springer.com/article/10.1007/s00330-020-07044-9.

[17] H. Ko, H. Chung, W.S. Kang, K.W. Kim, Y. Shin, S.J. Kang, et al., COVID-19 pneumonia diagnosis using a simple 2D deep learning framework with a single chest CT image: model development and validation, J. Med. Internet Res. 22 (2020) 13. Article ID: e19569, Jun.

[18] X.G. Wang, X.B. Deng, Q. Fu, Q. Zhou, J.P. Feng, H. Ma, et al., A weakly-supervised framework for COVID-19 classification and lesion localization from chest CT, IEEE Trans. Med. Imaging 39 (2020) 2615–2625. Aug.

[19] S. Tabik, A. Gómez-Ríos, J. Martín-Rodríguez, I. Sevillano-García, M. Rey-Area, D. Charte, et al., "COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on Chest X-Ray images," arXiv Preprint 2020.

[20] M.M. Hasan, N. Schaduangrat, S. Basith, G. Lee, W. Shoombuatong, B. Manavalan, HLPpred-Fuse: improved and robust prediction of hemolytic peptide and its activity by fusing multiple feature representation, Bioinformatics 36 (2020) 3350–3356. Jun.

[21] D. Li, Q. Wang, F.Q. Kong, Adaptive kernel sparse representation based on multiple feature learning for hyperspectral image classification, Neurocomputing 400 (2020) 97–112. Aug.

[22] J. Shi, R. Wang, Y. Zheng, Z. Jiang, L. Yu, Graph convolutional networks for cervical cell classification, in: Second MICCAI Workshop on Computational Pathology (COMPAT), Shenzhen, China, 2019.

[23] Y.R. Bin, Z.M. Chen, X.S. Wei, X.Y. Chen, C.X. Gao, N. Sang, Structure-aware human pose estimation with graph convolutional networks, Pattern Recognit. 106 (2020) 13. Article ID: 107410, Oct.

[24] Z.Q. Tian, X.J. Li, Y.Y. Zheng, Z. Chen, Z. Shi, L.Z. Liu, et al., Graph-convolutional-network-based interactive prostate segmentation in MR images, Med. Phys. (2020) 13, https://doi.org/10.1002/mp.14327.

[25] D. Castillo-Barnes, L. Su, J. Ramirez, D. Salas-Gonzalez, F.J. Martinez-Murcia, I. A. Illan, et al., Autosomal dominantly inherited Alzheimer disease: analysis of genetic subgroups by machine learning, Inf. Fusion 58 (2020) 153–167. Jun.

[26] J. Rodriguez-Rivero, J. Ramirez, F.J. Martinez-Murcia, F. Segovia, A. Ortiz, D. Salas, et al., Granger causality-based information fusion applied to electrical measurements from power transformers, Inf. Fusion 57 (2020) 59–70. May.

[27] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, N.I. Chervyakov, Application of the residue number system to reduce hardware costs of the convolutional neural network implementation, Math. Comput. Simul. 177 (2020) 232–243. Nov.

[28] S. Tabik, R.F. Alvear-Sandoval, M.M. Ruiz, J.L. Sancho-Gomez, A.R. Figueiras-Vidal, F. Herrera, MNIST-NET10: a heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. Ensembles overview and proposal, Inf. Fusion 62 (2020) 73–80. Oct.

[29] G. Mittal, P. Korus, N. Memon, FiFTy: large-scale file fragment type identification using convolutional neural networks, IEEE Trans. Inf. Forensics Secur. 16 (2021) 28–41.

[30] M. Azamfar, J. Singh, I. Bravo-Imaz, J. Lee, Multisensor data fusion for gearbox fault diagnosis using 2-D convolutional neural network and motor current signature analysis, Mech. Syst. Signal Process. 144 (2020) 18. Article ID: 106861, Oct.

[31] E. Kim, Interpretable and accurate convolutional neural networks for human activity recognition, IEEE Trans. Ind. Inf. 16 (2020) 7190–7198. Nov.

[32] S. Jeon, J. Moon, Malware-detection method with a convolutional recurrent neural network using opcode sequences, Inf. Sci. 535 (2020) 1–15. Oct.

[33] D.R. Nayak, D. Das, R. Dash, S. Majhi, B. Majhi, Deep extreme learning machine with leaky rectified linear unit for multiclass classification of pathological brain images, Multimed. Tools Appl. 79 (2020) 15381–15396. Jun.

[34] J.M. Górriz, Artificial intelligence within the interplay between natural and artificial computation: advances in data science, trends and applications, Neurocomputing 410 (2020) 237–270.

[35] Y.-D. Zhang, Advances in multimodal data fusion in neuroimaging: overview, challenges, and novel orientation, Inf. Fusion 64 (2020) 149–187, 2020/12/01/.

[36] J. Kileel, M. Trager, J. Bruna, On the expressive power of deep polynomial neural networks, in: Advances in Neural Information Processing Systems 32, 2019, pp. 1–10, 32H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc, E. Fox, and R. Garnett, Eds., ed La Jolla: Neural Information Processing Systems (NIPS) https

://papers.nips.cc/paper/9219-on-the-expressive-power-of-deep-polynomial-neural-networks.pdf.
[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958. Jun.
[38] Z.L. Shi, Y.D. Ye, Y.P. Wu, Rank-based pooling for deep convolutional neural networks, Neural Netw. 83 (2016) 21–31. Nov.
[39] Y.Y. Jiang, Cerebral micro-bleed detection based on the convolution neural network with rank based average pooling, IEEE Access 5 (2017) 16576–16583.
[40] Z. Sun, R. Chiong, Z.P. Hu, An extended dictionary representation approach with deep subspace learning for facial expression recognition, Neurocomputing 316 (2018) 1–9. Nov.
[41] N. Akhtar, U. Ragavendran, Interpretation of intelligence in CNN-pooling processes: a methodological survey, Neural. Comput. Appl. 32 (2020) 879–898. Feb.
[42] A.S. Tarawneh, A.B.A. Hassanat, K. Almohammadi, D. Chetverikov, C. Bellinger, SMOTEFUNA: synthetic minority over-sampling technique based on furthest neighbour algorithm, IEEE Access 8 (2020) 59069–59082.
[43] Z. Jan, B. Verma, Multiple strong and balanced cluster-based ensemble of deep learners, Pattern Recognit. 107 (2020) 11. Article ID: 107420, Nov.
[44] I. Spinelli, S. Scardapane, A. Uncini, Missing data imputation with adversarially-trained graph convolutional networks, Neural Netw. 129 (2020) 249–260. Sep.
[45] T. Mallick, P. Balaprakash, E. Rask, J. Macfarlane, Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting, Transp Res Rec 2674 (2020) 473–488.
[46] T. Derr, Y. Ma, W.Q. Fan, X.R. Liu, C. Aggarwal, J.L. Tang, et al., Epidemic graph convolutional network, in: 13th International Conference on Web Search and Data Mining, Houston, TX, 2020, pp. 160–168.
[47] C. Jeong, S. Jang, E. Park, S. Choi, A context-aware citation recommendation model with BERT and graph convolutional networks, Scientometrics 124 (2020) 1907–1922. Sep,.
[48] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, presented at the, in: International Conference on Learning Representations (ICLR), ICLR, Palais des Congrès Neptune, 2017, pp. 1–14.
[49] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2020) 336–359, 2020/02/01.