EDL-COVID: Ensemble Deep Learning for COVID-19 Cases Detection from Chest X-Ray Images

Shanjiang Tang, Chunjiang Wang, Jiangtian Nie, Neeraj Kumar, Yang Zhang, Zehui Xiong, Ahmed Barnawi

***Abstract*—Effective screening of COVID-19 cases has been becoming extremely important to mitigate and stop the quick spread of the disease during the current period of COVID-19 pandemic worldwide. In this work, we consider radiology examination of using chest x-ray images, which is among the effective screening approaches for COVID-19 cases detection. Given deep learning is an effective tool and framework for image analysis, there have been lots of studies for COVID-19 cases detection by training deep learning models with x-ray images. Although some of them report good prediction results, their proposed deep learning models might suffer from overfitting, high variance and generalization errors caused by noise and limited number of datasets. Considering ensemble learning can overcome the shortcomings of deep learning by making prediction with multiple models instead of a single model, we propose *EDL-COVID*, an ensemble deep learning model employing deep learning and ensemble learning. The EDL-COVID model is generated by combining multiple snapshot models of COVID-Net, which has pioneered in an open-sourced COVID-19 cases detection method with deep neural network processed chest x-ray images, by employing a proposed weighted averaging ensembling method that is aware of different sensitivities of deep learning models on different classes types. Experimental results show that EDL-COVID offers promising results for COVID-19 cases detection with an accuracy of 95%, better than COVID-Net of 93.3%.**

***Index Terms*—COVID-19, Ensemble Learning, Deep Learning, EDL-COVID, Chest X-Ray Images.**

I. Introduction

THE novel Coronavirus Disease 2019 (COVID-19), as an unprecedented infectious and dangerous disease around the world, is caused by SARS-CoV-2, a severe acute respiratory syndrome coronavirus 2, which has not been ever found in humans before Dec 2019 [1]. There is a rapid person-to-person corona virus transmission between two people in a close contact via aerosols or small droplets created by talking, coughing and sneezing. Once infected, people tend to have the following common symptoms after several days, including fever, cough, taste/small loss, and shortness of breath. As of 10^{th} Jan 2020, there are still more than 90 million active infected cases, and 1.94 million people have died worldwide.

To stop the fast spread of COVID-19, an important task is to find out infected people via effective screening such that they can be isolated and received immediate treatment. So far, the most commonly used screening approach for COVID-19 cases detection is to take a reverse transcription polymerase chain reaction (RT-PCR) test over a sample of nasopharyngeal exudate collected from suspectable people for the qualitative detection of nucleic acid from SARS-CoV-2 attributes to its merits as a simple but specific qualitative assay [6], [30]. Although RT-PCR testing has been recognized as a "gold standard" for infected case discovery of the disease, there are still several issues about it. First, the sensitivity of its detection results is highly variable, which can generate false-negative and false-positive results according to a recent study by Tahamtan et al. [34]. Second, due to the quick spread of COVID-19, there are not sufficient PCR reagent kits to satisfy the overwhelming screening demand especially in poor and heavily-affected areas [35].

In addition to RT-PCR, radiography examination is an alternative effective screening method for fast detection of COVID-19, where the chest X-ray (CXR) and CT images are performed and analyzed by radiologists to judge whether a suspectable person has been infected or not by SARS-Cov-2 [36], [3]. Recent studies have observed the *abnormal* features in radiography images of COVID-19 cases and it has been widely used in China at the earlier stage of the global outbreak [15], [16], [38]. Although CT scan has a higher sensitivity to pulmonary diseases, there are several limitations for its practical uses in COVID-19 cases detection at the larger scale , including non-portability, long-time scanning, and the risk of exposing the hospital staff. In comparison to CT scans, CXR imaging is portable, faster, more readily available, and can be performed within an isolated room while offering an acceptable accuracy in COVID-19 cases detection [29]. Due to the these benefits, many recent studies [18], [7], [29], [36] have now focused on CXR images analysis for COVID-19 cases detection. Particularly, there are some studies [25] suggesting to take portable CXR imaging as a reliance method for COVID-19 cases detection with the quick spread of the pandemic.

While CXR imaging is very fast, it needs expert radiologists to make judgement for COVID-19 cases detection *manually*, which requires professional knowledge and is a time-consuming process. Meanwhile, the number of radiologists is

S.J. Tang, C.J. Wang are with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China.
E-mail: {tashj, 2019216020}@tju.edu.cn.
J.T. Nie is with the School of Computer Science&Engineering, Nanyang Technological University, Singapore.
E-mail: jnie001@e.ntu.edu.sg. (Corresponding authors: Jiangtian Nie.)
N. Kumar is with Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India.
E-mail: neeraj.kumar@thapar.edu.
Y. Zhang is with the School of Computer Science& Technology, Technology University of Wuhan, Wuhan, China.
E-mail: yangzhang@whut.edu.cn
Z.H. Xiong is with Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore.
E-mail: zehui_xiong@sutd.edu.sg.
A. Barnawi is a faculty of Computing and IT, King Abdulaziz University, Jeddah, Saudi Arabia.
E-mail: ambarnawi@kau.edu.sa. .

much fewer than that of people under detection. An AI-aided diagnostic system is thus needed to assist radiologists to make screening of COVID-19 cases in a more rapid and accurate way, otherwise it is prone to occur that infected people cannot be detected and quarantined as soon as possible and in turn cannot receive treatment timely [24], [36].

Essentially, COVID-19 cases detection with CXR images is a classification problem in machine learning domain, for which Convolutional neural network (CNNs) enabled data-driven deep learning methods have demonstrated promising performance [21]. As such, many recent studies [6], [36], [24], [26] are available for attempting to train new deep learning models for infected cases detection with CXR images by either reusing or modifying existing deep neural networks atop of collected CXR images datasets. However, due to the noise and limited training data size in practice, a deep learning model might suffer from overfitting, high variance and generalization errors although some studies in their articles report much high prediction accuracy for their proposed deep learning models with their own datasets.

To alleviate it, instead of using a single model, ensemble learning that combines multiple models together with a proper strategy (e.g., random forest [10], boosting [32], stacking [12]) is assumed to be an effective technique. It is able to not only reduce the variance and generalization errors of predictions but also can yield a better result than any single model [28].

In this study, to take advantage of both deep learning and ensemble learning, we propose *EDL-COVID*, an ensemble deep learning model for detecting COVID-19 cases with CXR images. There are several challenging issues for ensemble deep learning model development. First, it needs to have multiple deep learning models for combination, whereas training a deep model generally takes a significant amount of computational cost and a long time to converge. Moreover, the development of a good ensemble deep learning model depends on both the accuracy of each individual model and the diversity among these deep learning models [13].

To reduce the computation cost as well as training time, instead of training multiple different deep neural networks, we consider an alternative approach of generating multiple deep learning models using a single deep neural networks. Specifically, we execute a single training run with multiple model snapshots first, and then ensemble the prediction results for a final prediction. However, there is a big problem for this approach that the produced multiple models can be quite *similar* to each other since they have the same network architecture as well as training initialization, violating the model diversity requirement of ensemble learning. Using these similar models tends to result in similar predictions and predictions errors, indicating that combination of these models cannot offer much benefit. To resolve it, one effective approach is to make an aggressive learning rate change during a single neural network training process, which can force the exploration of different model weights and produce a diversity of multiple snapshot models [22].

Specifically, our proposed EDL-COVID is based on COVID-Net [36], which is state-of-the-art open-sourced deep convolutional neural network for COVID-19 cases detection from CXR images. By using COVID-Net network architecture as well as its datasets of COVIDx [2], we first train multiple snapshot deep learning models with a cosine annealing learning rate schedule, for which the learning rate fluctuates significantly in that it starts high and drops to a minimum value close to zero rapidly before going up to the maximum value again [22]. Next, the ensemble deep learning model of EDL-COVID is developed by combining these models with a proposed model ensembling approach called weighted averaging ensembling (WAE), which is based on two observations that 1) there are different sensitivities for different classes types of an individual deep learning model, and 2) different deep learning models have different sensitivities for each class type. WAE is based on the assumption that for a class type, a model with a higher sensitivity should contribute more to the final ensembling result by estimating its weight proportional to its sensitivity. Finally, the experimental results show that EDL-COVID achieves promising results for cases detection from CXR images with 95.0% accuracy, 96.0% sensitivity and 94.1% PPV for COVID-19 class type.

In summary, the following contributions are made in this work:

- We propose a Weighted Average Ensembling (WAE) ensembling strategy with the awareness of varied class-level accuracies for different machine learning models.
- We propose a snapshot ensemble deep learning model called EDL-COVID based on the state-of-the-art deep learning architecture of COVID-Net.
- We evaluate EDL-COVID experimentally, showing the promising results of EDL-COVID.

The rest of this work is organized as follows. Section II provides a background of ensemble deep learning and COVID-Net. In Section III, related work are reviewed. The proposed EDL-COVID model is introduced in Section IV. Experimental evaluation of the proposed model is provided in Section V. Lastly, Section VI concludes the paper with possible future directions.

II. Background

For the sake of better understanding EDL-COVID model, we give a description of ensemble deep learning and COVID-Net network that EDL-COVID is built on for COVID-19 cases detection.

A. Ensemble Deep Learning

As a powerful machine learning technique, deep learning is widely applied to many studies, e.g., computer vision, speech recognition, medical image analysis, drug design, etc [23], [8]. It contains a multi-layer neural network that can progressively extract higher level features from raw data such as images and produce prediction outputs based on those features. A deep learning model computation consists of two stages, namely *training* and *inference*. Training is an iterative and stochastic computation for generating a model based on a training data. Several arguments need to be initialized before training computation, including learning rate, epoch number,

and batch size, where different configurations tend to result in models with different accuracies. Inference is a prediction process with the trained deep learning model. Some popular deep learning architectures are available, e.g., convolutional neural networks (CNN) and recurrent neural networks (RNN). In comparison, CNN is the most popular one, which is good for applications such as image detection/recognition, image classification and medical image analysis [5].

With the existance of the noise in the training data and the randomness in the deep learning algorithm, it however tends to suffer from high variance problem and generalization error [19]. Although there are some commonly used techniques such as data augmentation and regularization [9], the problems are still not well addressed for deep learning models.

To overcome these problems for a single machine learning model, ensemble learning is assumed to be an effective approach [28]. It is a hybrid learning paradigm that is able to produce more accurate and robust prediction results than a single model by combining multiple machine learning models intelligently. Various ensemble strategies are available, including averaging, random forest [10], boosting [32] and stacking [12]. To make ensemble be effective, we must make sure that multiple models for combination is *diverse*. In the proposed work, we study ensemble deep learning for COVID-19 cases detection by combining multiple deep learning models via an ensemble strategy.

B. COVID-Net

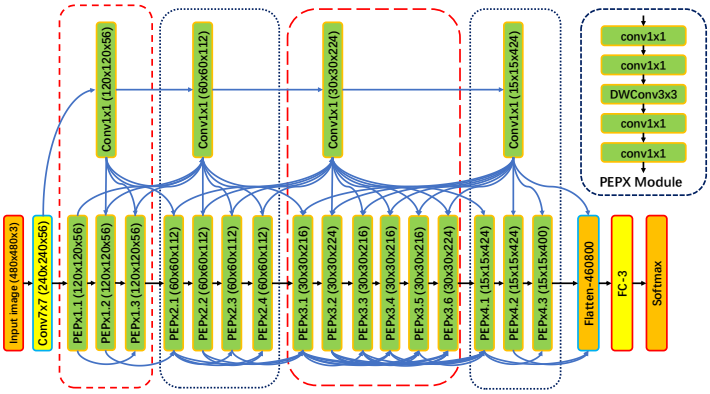


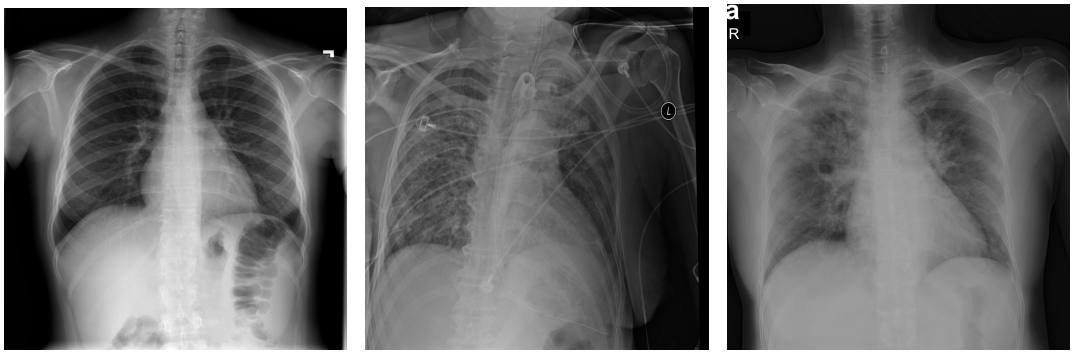
Fig. 1: The convolutional neural network architecture of COVID-Net [36].

COVID-Net [36] is by far the state-of-the-art deep convolution neural network for CXR image processing based COVID-19 cases detection, available at https://github.com/lindawangg/COVID-Net. Figure 1 presents its network architecture, which is carefully designed model for COVID-19 cases detection in a human-machine collaborative manner with the following characteristics. First, it exploits a design pattern of lightweight residual projection-expansion-projection-extension (PEPX) heavily, which makes up of first-stage projection, expansion, depth-wise representation, second-stage projection and extension. Specifically, the *First-Stage Projection* is a 1×1 convolution to project input features from a high dimension to a lower one. *Expansion* is a reverse procedure that expands features to a higher dimension with a 1×1 convolution. *Dep-wise Representation* is responsible for learning spatial features to minimize computational complexity while keeping representational capacity via a 3×3 depth-wise convolutions. *Second-stage Projection* projects features back to a lower dimension with a 1×1 convolutions. Finally, *Extension* extends channel dimensionality to a higher one to produce the final features with 1×1 convolutions. Second, it contains long-range connectivities at different places of the network architecture. Third, the COVID-Net network is considerably diverse in order to realize a strong representational capacity for COVID-19 cases detection. All of these features enable COVID-Net to achieve a strong representational capacity while keeping computational complexity reduced. It offers a COVIDx dataset for training, which classifies people into the following cases, i.e., Normal, Pneumonia, as well as COVID-19 as Figure 2 shows. By using COVIDx dataset [2], it is reported for COVID-Net that it can achieve an accuracy of 93.3%, and positive predictive value of $90.5\%, 91.3\%, 98.9\%$ respectively for Normal, Pneumonia, and COVID-19.

III. Related Work

To reduce the burden of detection from radiography images (e.g., CT images and CXR images) for radiologists, there have been several artificial intelligence (AI)-aided detection systems proposed, which are based on deep learning [20], [37], [31]. Due to several benefits of CXR imaging including portability, availability, accessibility and fast checking compared to CT imaging, CXR images are becoming a popular and commonly utilized data source for COVID-19 cases detection.

Chowdhury et al. [11] made a model training comparison of different deep learning networks including AlexNet, ResNet18, DenseNet201, and SqueezeNet to classify two classes (i.e., COVID-19 and Normal) for CXR images, concluding that SqueezeNet outperforms other neural networks. Instead, Narin et al. [26] made a comparison of different CNN models (e.g., ResNet-50, Inception V3, and Inception-ResNetV2) trained on CXR images for COVID-19 cases detection, showing that ResNet-50 outperforms other two models with 98% accuracy. Farooq et al. [14] provided a COVID-ResNet by fine tuning a pre-trained ResNet-50 architecture with a reported accuracy of 96.23%. Wang et al. [36] made a tailored and first open-sourced deep neural network called COVID-Net for COVID-19 cases detection with CXR images. Tulin et al. [27] trained a deep learning scheme named DarkCovidNet, which detects COVID-19 cases based on a number of only 125 CXR images. Maghdid et al. [24] built deep learning models with pre-trained AlexNet based on its own established dataset of X-ray images and CT images for COVID-19 cases detection. Alom et al. [6] proposed a multi-task deep learning system called COVID_MTNet for case detection by considering both CXR and CT images together. HSMA_WOA [4] hybrids the novel Slime mould algorithm together with whale optimization algorithm to address the CXR based image segmentation issue for detecting COVID-19 cases. In contrast, we focus on the COVID-19 cases detection and HSMA_WOA is complementary to our work.



(a) Normal. (b) Pneumonia. (c) COVID-19.

Fig. 2: CXR images for normal and illness people from COVIDx datasets [2], which categorizes CXR images into classes of: (a) Normal case, (b) pneumonia case, and (c) COVID-19 case.

While these studies seem to offer pretty good results, the reliability of their proposed deep learning models can be questioned due to a serious bias problem for COVID-19 dataset collected from a small sized group of COVID-19 cases [33]. Moreover, Moreover, since the training processes of deep learning models rely on stochastic algorithms, they are sensitive to the specifies of training data and can produce different weights each time they are trained. Hence, the prediction result of a single deep learning model is prone to suffer from high variance and generalization errors due to the noise and a limited amount of COVID-19 datasets collected by far. Fortunately, these issues can be alleviated with the adoption of ensemble learning technique. An ensemble deep learning model called EDL-COVID is proposed in this work by combining multiple snapshot deep learning models trained from state-of-the-art neural network of COVID-Net with a proposed weighted averaging ensembling approach (WAE) (See section IV-B) that is aware of different sensitivities for different models on each class type.

IV. EDL-COVID: Ensemble Deep Learning Model for COVID-19 Cases Detection

In this section, we introduce EDL-COVID, a snapshot ensemble deep learning model based on COVID-Net. As illustrated in Figure 3, the overall training flow for EDL-COVID consists of two phases, namely, *snapshot model training* and *model ensembling*. The snapshot model training phase is responsible for producing multiple model snapshots (Section IV-A), which are then combined together for a final prediction in the model ensembling phase (Section IV-B). The detailed implementation of EDL-COVID can be found in Appendix A.

A. Snapshot Model Training

To enable deep learning ensembling, there is a need to have multiple pre-trained deep learning models. However, the model training process generally takes hours and a large amount of computing resources, making deep learning ensembling become a time-consuming and heavy computation process. To alleviate it, we can instead train multiple model snapshots of a deep learning network for ensembling during a single training run, rather than training multiple models from different deep learning networks separately. In this paper, we choose COVID-Net as the candidate to generate multiple model snapshots in terms of its promising performance for its CXR image based COVID-19 case detection, as well as public accessibility for its source code.

To make model ensembling effective in practice, there is a *diversity* requirement for multiple deep learning models that have different distributions of prediction errors. However, a limitation of such a model snapshot approach of a single deep learning network training is that the generated model snapshots tend to be similar, which can produce similar predictions and prediction errors. To address it, a commonly used approach is to take aggressive learning rate schedule during a single training run that makes large changes of model weights and in turn the nature of model snapshots [22].

We take cosine annealing learning rate schedule proposed by Loshchilov et al. [22] to change the learning rate aggressively but systematically to generate different model weights over training epochs, by allowing the learning rate to start high and decrease to a minimum value near zero at a relatively rapid speed before being increased again to the maximum based on the following formula,

$$\alpha(t) = \frac{\alpha_0}{2}(\cos(\frac{\pi mod(t-1, \lceil T/M \rceil)}{\lceil T/M \rceil}) + 1), \qquad (1)$$

where α_0 is the maximum learning rate, $\alpha(t)$ is the learning rate at epoch t and, M denotes the number of cycles, and T is the overall number of epochs. Then each time a new model snapshot is produced after training a network for M cycles. Specifically, we train multiple model snapshots by initializing $\alpha_0 = 0.002, T = 50$ and $M = 6$ for COVID-Net network with COVIDx dataset. Then 6 model snapshots would be produced from COVID-Net network. Figure 4 shows the aggressive variations of learning rates and corresponding training losses with cosine annealing learning rate schedule during the model training process. We can see that such a significant variation of learning rate can produce different model weights by observing the big changes of intermediate training losses.

B. Model Ensembling

After generating multiple model snapshots in the first phase, we now come to the *model ensembling* phase for building

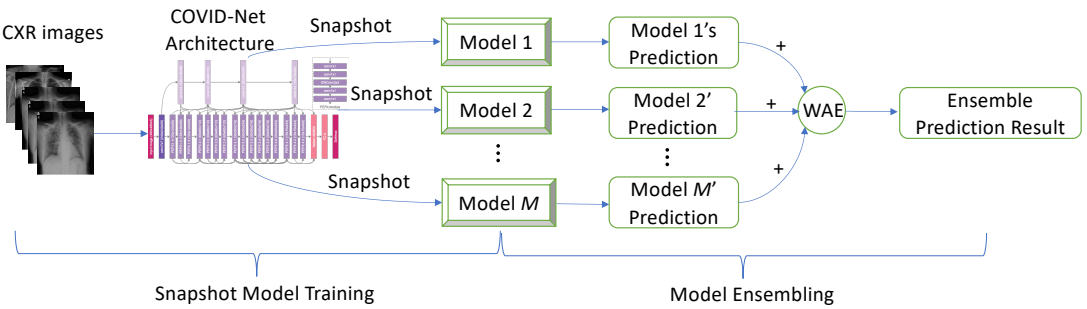


Fig. 3: The overall flow for EDL-COVID ensemble model training. It consists of two phases, namely, *snapshot model training*, and *model ensembling*. We propose WAE approach for model ensembling as described in Algorithm 1.

EDL-COVID by combining these models as illustrated in Figure 3. As we discussed in Section II-A, there are many ensembling strategies for model ensemble. For snapshot ensemble learning, *averaging* is a popular ensembling strategy [17]. For an input sample, it simply averages the class probabilities for each class from all models, respectively. Let M be the number of classifiers (i.e., deep learning models). Let $p_{m,k}(d_i)$ be the class probability of the k^{th} class output by the m^{th} classifier with respect to the input sample d_i. Then the average class probability is $\frac{\sum_{m=1}^{M} p_{m,k}(d_i)}{M}$, for $k \in [1, K]$ of the input sample d_i, with K denoting the number of classes.

The traditional averaging ensembling strategy is implicitly based on the assumption that all models have the same weights. However, there are two key observations as follows,

- For a deep learning model, the testing accuracies for different classes are generally different.
- Different deep learning models tend to have different accuracies for each class. It indicates that we cannot simply treat each model equally during the model ensembling.

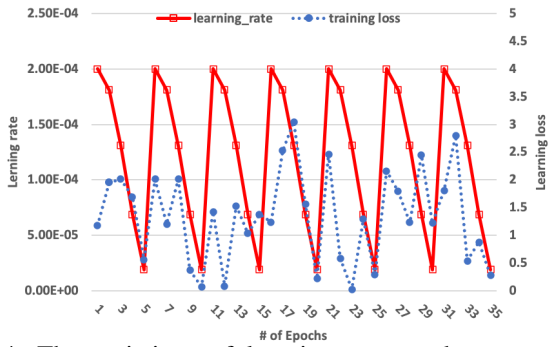


Fig. 4: The variations of learning rates and corresponding learning losses under cosine annealing learning rate schedule over training epochs. The aggressive change of learning rate produces different model weights with significantly different training loss, making a set of diverse models.

Based on the two observations above, we instead propose a *Weighted Average Ensembling (WAE)* approach for snapshot model ensemble as depicted in Algorithm 1. Let $a_{i,j}$ be the test accuracy of the i^{th} model for the j^{th} class over the test data of CXR images dataset. Let $w_{i,j}$ denote the normalized weight of the i^{th} model for the j^{th} class. Then we have $w_{i,j} = \frac{a_{i,j}}{\sum_{m=1}^{M} a_{m,j}}$ (*Line 7*). For each input sample d_i, we first get the output of every class probability $p_{m,k}(d_i)$ from the m^{th} model for $\forall m \in M$ (*Line 10-12*). Then we can estimate the class probability $p_k(d_i)$ of EDL-COVID by summing up the weighted class probabilities of all models (*Line 13-14*) for $\forall k \in K$. Finally, we can get the predicted class by returning the class index with the maximum class probability for each input sample (*Line 15*).

Algorithm 1 Pseudo code of the proposed Weighted Average Ensembling (WAE) approach.

1: **Input**:
2: M : the number of classifiers (i.e., deep learning models).
3: N : the size of CXR images dataset.
4: K : the number of classes for CXR images dataset.
5: $a_{i,j}$: the test accuracy of the i^{th} model for the j^{th} class.
6: d_i : the i^{th} input sample of CXR images dataset.
7: $w_{i,j}$: the normalized weight of the i^{th} model for WAE over the j^{th} class prediction, where $w_{i,j} = \frac{a_{i,j}}{\sum_{m=1}^{M} a_{m,j}}$.
8: **Output**:
9: $c(d_i)$: the predicted class index for the i^{th} input sample with EDL-COVID.
10: **for** $i = 1$ **to** N **do**
11: **for** $m = 1$ **to** M **do**
12: Get the output (i.e., class probability vector $\mathbf{p}_m(d_i) = \{p_{m,k}(d_i)\}$ for all K classes where $1 \leq k \leq K$) of the m^{th} model w.r.t. the input sample d_i.
13: **for** $k = 1$ **to** K **do**
14: $p_k(d_i) = \sum_{m=1}^{M} p_{m,k}(d_i) \cdot w_{i,k}$.
15: $c(d_i) = \arg\max_{1 \leq k \leq K} \{p_k(d_i)\}$. ▷ Get the class index with the maximum class probability for the i^{th} input sample.

V. Experimental Evaluation

We have implemented EDL-COVID atop of TensorFlow based on COVID-Net, and evaluated EDL-COVID using CXR images of COVIDx dataset in a GPU machine.

A. Experimental Setup

GPU Machine. We train and evaluate our proposed model on a GPU machine consisting of two Intel Xeon E5-2640 CPUs, 256GB of RAM, and two NVIDIA Tesla P100 GPUs. It runs on CentOS 7.7 OS with TensorFlow 2.0.0 installed for

deep learning model training and inference, where cuDNN is enabled to speedup the training computation on GPU device.

Dataset. We take the latest *COVIDx* dataset proposed by Wang et al. [2] for model training and evaluation in our experiment. It totally contains $15,477$ CXR images from $13,870$ cases by 20^{th} June 2020, comprising of $6,053$ Pneumonia, $8,851$ Normal, and 573 COVID-19 cases images. The COVIDx dataset is collected from five different data sources, namely, ActualMed COVID-19 dataset[1], COVID-19 Image Data Collection[2], COVID-19 radiography database[3], COVID-19 CXR Dataset Initiative[4], as well as RSNA Pneumonia Detection Challenge[5]. In our experiment below, we take $13,898$ CXR images from COVIDx dataset for model training and the remaining $1,579$ CXR images for testing. Specifically, for training data, it consists of 7966 Normal, 5459 Pneumonia and 473 COVID-19 CXR images. In contrast, there are 885 Normal, 594 Pneumonia and 100 COVID-19 CXR images for testing data.

B. Performance Evaluation for Individual Model

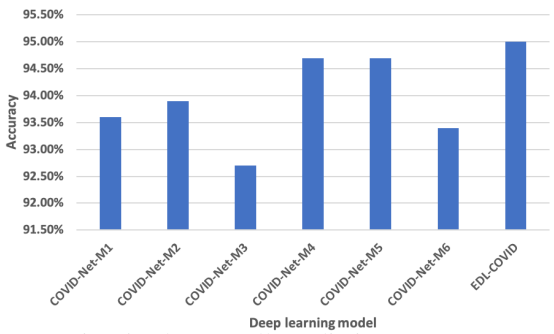


Fig. 5: The overall prediction accuracy.

Performance metrics of each individual model are evaluated in the following aspects.

Accuracy Evaluation. We have trained 6 deep learning models (denoted as *COVID-Net-M1*, *COVID-Net-M2*, $\cdots$, *COVID-Net-M6*) with COVID-Net network architecture on top of COVIDx dataset. Figure 5 presents the prediction accuracy for all models. It shows that the ensembling approach of our proposed EDL-COVID model can make it effectively outperform other six deep learning models by about 0.3%.

However, in practice, we cannot simply make a judgement that a model with a higher accuracy must work better than a model with a lower accuracy for a multi-class problem. We need to further take a look at other two important class-level metrics, namely, *Sensitivity* and *Positive Predictive Value (PPV)* as well.

Sensitivity Evaluation. In medical analysis, the *sensitivity* of a disease can be interpreted as the proportion of people with a certain disease that have been successfully identified. Taking COVID-19 for example, achieving a high sensitivity is quite important since we do not want to omit any affected people during our COVID-19 testing, otherwise the affected people that have been omitted cannot receive immediate treatment and meanwhile they can affect other people. Table I gives a sensitivity analysis for each model with respect to each class type. We have the following observations. It is seldom to have a model that works best for all three classes. For example, COVID-Net-M5 has the highest sensitivity in *Normal* class but not for two other class types. In comparison, EDL-COVID has the highest sensitivities for both *Pneumonia* and *COVID-19* classes although its sensitivity of *Normal* class is not the best across all models. From practical point of view, there is no doubt to consider EDL-COVID since a *high* sensitive screening for *infectious* diseases such as COVID-19 is very important.

Sensitivity (%)			
Model	**Normal**	**Pneumonia**	**COVID-19**
COVID-Net-M1	93.2	94.8	91.0
COVID-Net-M2	93.4	95.1	91.0
COVID-Net-M3	94.1	90.2	95.0
COVID-Net-M4	94.8	94.8	94.0
COVID-Net-M5	**95.8**	93.6	92.0
COVID-Net-M6	95.3	90.2	96.0
EDL-COVID	95.0	**94.8**	**96.0**

TABLE I: The sensitivities of different models on each class (e.g., Normal, Pneumonia, and COVID-19), where the best results are highlighted in **bold**.

PPV Evaluatiion. Positive Predictive Value (PPV) denotes the probability of positive results that are *true* positive results in diagnostic tests according to Formula (4). If this value is small, it means that there are many false positives and a follow-up testing is needed for any positive result with a more reliable result. For COVID-19 screening, if a model's PPV is small, we cannot make a judgement that a person with positive testing result is the *true* COVID-19 case and a more accurate testing is needed for positive testing results. Table II gives a positive predictive value (PPV) analysis for each model on each class type. Still, no model performs the best for all classes types. COVID-Net-M6 has a highest PPV on Pneumonia class, and our EDL-COVID achieves the highest PPVs for Normal and COVID-19 classes.

In summary, we can conclude that, although no model outperforms others on all metrics for all classes types, for COVID-19 case detection, our proposed EDL-COVID is the best choice since it outperforms other models on the accuracy, sensitivity and PPV for COVID-19 class type.

Positive Predictive Value (%)			
Model	**Normal**	**Pneumonia**	**COVID-19**
COVID-Net-M1	96.3	89.9	94.0
COVID-Net-M2	96.2	90.5	92.9
COVID-Net-M3	95.2	92.7	75.4
COVID-Net-M4	96.3	92.8	93.1
COVID-Net-M5	95.7	93.6	92.9
COVID-Net-M6	94.8	**94.4**	78.7
EDL-COVID	**96.4**	93.1	**94.1**

TABLE II: The Positive Predictive Value (PPV) of different models on each class (e.g., Normal, Pneumonia, and COVID-19), where the best results are highlighted in **bold**.

[1]https://github.com/agchung/Actualmed-COVID-chestxray-dataset.
[2]https://github.com/ieee8023/covid-chestxray-dataset.
[3]https://kaggle.com/tawsifurrahman/covid19-radiography-database.
[4]https://github.com/agchung/Figure1-COVID-chestxray-dataset.
[5]https://kaggle.com/c/rsna-pneumonia-detection-challenge.

C. EDL-COVID Evaluation Results

In this section, we evaluate EDL-COVID model from the following perspectives.

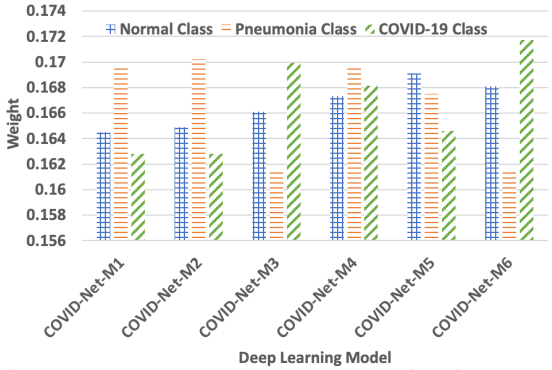


Fig. 6: The estimated class-level weights for 6 deep learning models needed by WAE in model ensembling.

Weights Estimation for WAE. Recall in Section IV-B that we proposed a model ensembling strategy called WAE to combine multiple deep learning models with the awareness of different accuracies on different classes types for different models. We first need to obtain weights for all deep learning models on each class type for WAE. Figure 6 illustrates the estimated class-level weights for all deep learning models, which is based on the estimated sensitivity result on each class type as shown in Table I. We can see that different classes tend to have different weights for different deep learning models. For example, for COVID-Net-M6, it has the largest weight for COVID-19 class but relatively low weight for Pneumonia class.

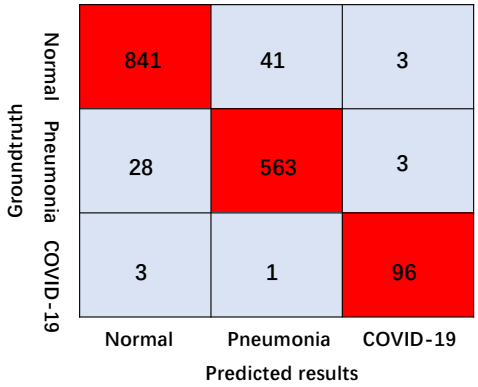


Fig. 7: Confusion matrix for EDL-COVID on COVIDx dataset that consists of 100 COVID-19, 594 Pneumonia, and 885 Normal images. The **red** box is the true prediction, and the **light blue** box is the false prediction.

Predicted Results. The confusion matrix for the proposed EDL-COVID is presented as in Figure 7, which analyzes the COVIDx dataset, including CXR images of 100 COVID-19 cases, 594 Pneumonia cases, and 885 Normal cases. For COVID-19 testing, only 4 out of 100 CXR images of COVID-19 are not screened out, and 6 out of 1579 CXR images are mistakenly considered as COVID-19, indicating that the error ratio is minor compared to the total number of CXR images for EDL-COVID.

To show the detection capability of EDL-COVID, we draw ROC curves for EDL-COVID prediction on COVIDx test dataset with respect to each class type in Figure 8. Larger area of ROC area is an indication of better prediction ability. We can see that the ROC area for each class under EDL-COVID is much closer to the maximum value of one, indicating that our proposed EDL-COVID has a good prediction capability for each class type in practice.

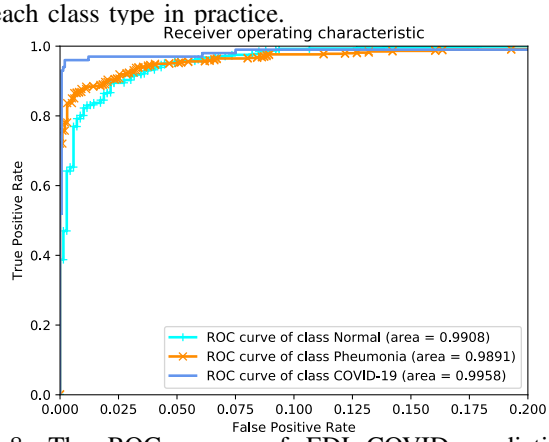


Fig. 8: The ROC curves of EDL-COVID prediction on COVIDx test dataset with respect to each class type.

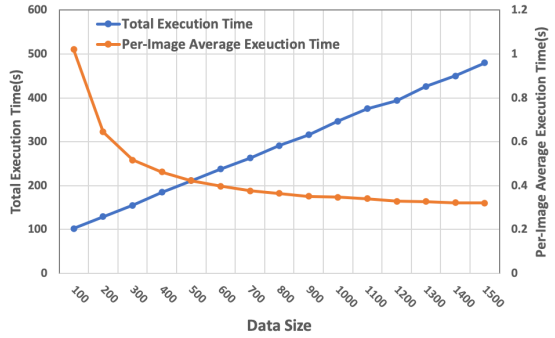


Fig. 9: The execution time for EDL-COVID under different numbers of testing CXR images.

Model Execution Time Evaluation. Figure 9 illustrates the execution time for EDL-COVID to handle different numbers of testing CXR images. We can see that a linear relationship is demonstrated between the data size and the total execution time, indicating that our proposed EDL-COVID is scalable. Moreover, we also present the average execution time for processing a CXR image. Interestingly, the average execution time drops significantly we increase the number of CXR images at the beginning and later becomes smoothly when the total number of images is larger than 1000. This is because EDL-COVID takes time to load the six models during its computing process (i.e., overhead), which cannot be ignored when workload size is small. However, it becomes relatively small for such an overhead when the workload size becomes large, making the average execution time per image becomes smaller first and later keep stable as observed.

VI. Conclusion and Future Work

In this paper, to overcome several problems of overfitting, high variance and generalization errors of an individual

deep learning model for improved performance on COVID-19 cases detection, we propose EDL-COVID, an ensemble deep learning model called for COVID-19 cases detection from CXR images on the basis of the open-sourced network architecture called COVID-Net. We first generate multiple model snapshots by training COVID-Net network on top of COVIDx CXR datasets, followed by ensembling these models with a proposed WAE ensembling approach that is aware of different sensitivities for different deep learning models on different classes types. Experiments on a COVIDx test data of 1579 CXR images show that EDL-COVID can detect COVID-19 cases with good promising results of 96% sensitivity and 94.1% PPVs, outperforming each individual deep learning model. We hope our AI-based screening approach can aid radiologists to accelerate the screening of COVID-19 cases while guaranteeing a high accuracy in the current COVID-19 pandemic around the world.

Yet, current work is heavily dependent on Wang et al.'s work [36] of COVID-Net network architecture and COVIDx dataset. There are several future work that can be done to enhance EDL-COVID for practical use. First, the number of COVID-19 CXR images is still relatively small compared to CXR images of other classes for COVIDx dataset, indicating that we cannot simply make a judgement that the currently trained model snapshots from COVID-Net architecture continue to work well with a high accuracy for unseen COVID-19 CXR images. To improve EDL-COVID, we need to retrain model snapshots of COVID-Net whenever a new CXR images are available. Besides snapshot models, we can also incorporate some other publicly available good deep learning models into EDL-COVID for better performance. Second, we would plan to extend EDL-COVID for other COVID-19 applications such as risk stratification for COVID-19 cases survival analysis, risk status analysis of COVID-19 cases, which are important for patient hospitalization and care planning.

APPENDIX A
THE IMPLEMENTATION OF EDL-COVID

As discussed in Section IV, EDL-COVID consists of two parts, i.e., snapshot model training and model ensembling. In this section, we present its key implementation codes for the sake of better understanding of EDL-COVID.

Figure 10 shows the snapshot model training codes for EDL-COVID. To output a set of *diverse* snapshot models, we leverage cosine annealing learning rate schedule to dynamically estimate and adjust learning rate aggressively and systematically (*Line 26*), which is implemented in *CosineAnnealingLearningRateSchedule()*(*Line 2-22*). Next, we load the training and testing data (*Line 27 - 46*). After that, we start to train models with provided COVID-Net network architecture (*Line 47-101*). Typically, at the beginning of each epoch, we adjust the learning rate dynamically (*Line 78*). We spill the intermediate snapshot model into disks at every *epochs_per_cycle* configured by users (*Line 99-101*).

Figure 11 presents the model ensemble implementation for EDL-COVID. We first use test data to estimate the *class-level* accuracy for each snapshot model trained at the previous stage with *prediction_accuracy* function (*Line 2-14*), whose implementation is given in *Line 21-52*. With the class-level accuracies of snapshot models on each class, we next move to estimate the class-level weights for each model (*Line 16-18*) with *class_weight_estimate* function, whose implementation is given in *Line 53-70*. Finally, we perform model ensemble for all snapshot models with WAE strategy (*Line 19-22*) via *wae_estimate* function, which are detailed in *Line 71-103*.

ACKNOWLEDGEMENT

This work is sponsored by the National Natural Science Foundation of China (61972277) and Tianjin Natural Science Foundation (18JCZDJC30800).

REFERENCES

[1] Coronavirus disease 2019. In *https://en.wikipedia.org/wiki/Coronavirus_disease_2019*, 2020.

[2] Covidx dataset. In *https://github.com/lindawangg/COVID-Net/blob/master/docs/COVIDx.md*, 2020.

[3] M. Abdel-Basset, V. Chang, H. Hawash, R. K. Chakrabortty, and M. Ryan. Fss-2019-ncov: A deep learning architecture for semi-supervised few-shot segmentation of covid-19 infection. *Knowledge-Based Systems*, 212:106647, 2021.

[4] M. Abdel-Basset, V. Chang, and R. Mohamed. Hsma_woa: A hybrid novel slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest x-ray images. *Applied Soft Computing*, page 106642, 2020.

[5] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[6] M. Z. Alom, M. Rahman, M. S. Nasrin, T. M. Taha, and V. K. Asari. Covid_mtnet: Covid-19 detection with multi-task deep learning approaches. *arXiv preprint arXiv:2004.03747*, 2020.

[7] A. Borghesi and R. Maroldi. Covid-19 outbreak in italy: experimental chest x-ray scoring system for quantifying and monitoring disease progression. *La Radiologia Medica*, page 1, 2020.

[8] Z. Cao, T. Liao, W. Song, Z. Chen, and C. Li. Detecting the shuttlecock for a badminton robot: A yolo based approach. *Expert Systems with Applications*, 164:113833, 2020.

[9] S. Chen, E. Dobriban, and J. H. Lee. Invariance reduces variance: Understanding data augmentation in deep learning and beyond. *arXiv preprint arXiv:1907.10905*, 2019.

[10] Y.-Y. Cheng, P. P. Chan, and Z.-W. Qiu. Random forest based ensemble system for short term load forecasting. In *2012 International Conference on Machine Learning and Cybernetics*, volume 1, pages 52–56. IEEE, 2012.

[11] M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al-Emadi, et al. Can ai help in screening viral and covid-19 pneumonia? *arXiv preprint arXiv:2003.13145*, 2020.

[12] F. Divina, A. Gilson, F. Goméz-Vela, M. García Torres, and J. F. Torres. Stacking ensemble learning for short-term electricity consumption forecasting. *Energies*, 11(4):949, 2018.

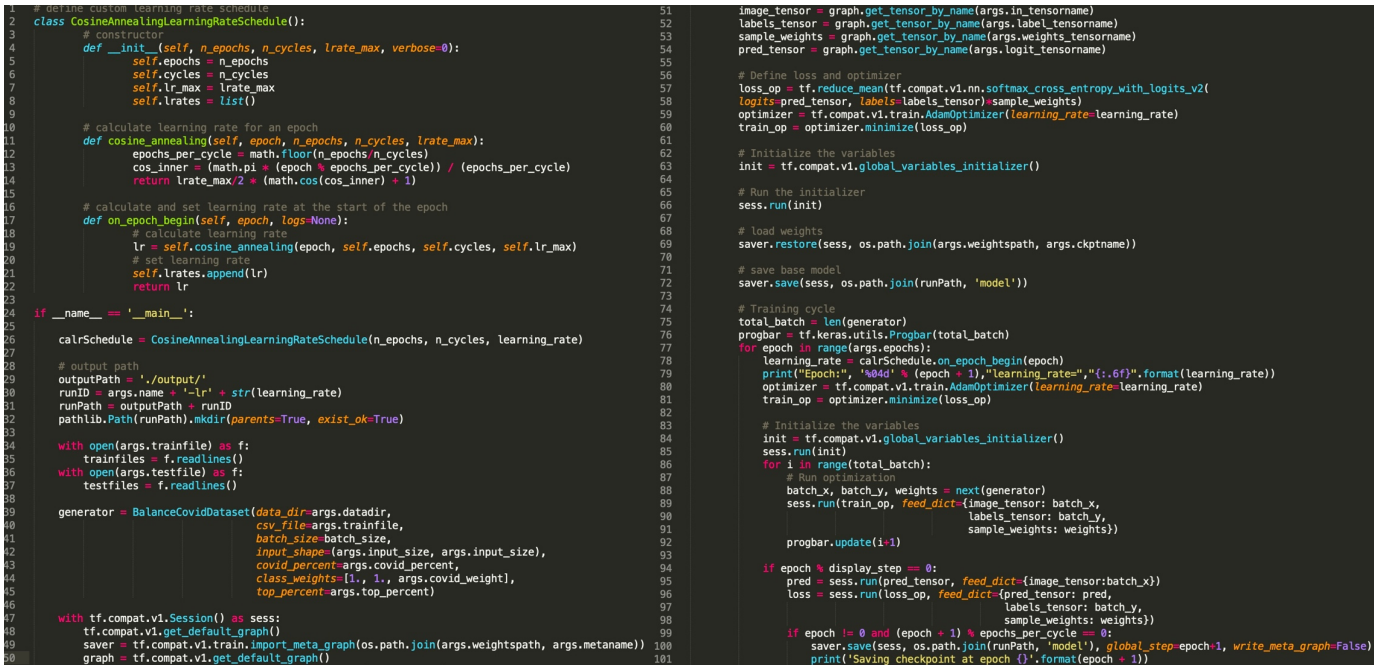
[13] F. A. Faria, J. A. Dos Santos, A. Rocha, and R. d. S. Torres. A framework for selection and fusion of pattern classifiers in multimedia recognition. *Pattern Recognition Letters*, 39:52–64, 2014.

[14] M. Farooq and A. Hafeez. Covid-resnet: A deep learning framework for screening of covid19 from radiographs. *arXiv preprint arXiv:2003.14395*, 2020.

[15] W.-j. Guan, Z.-y. Ni, Y. Hu, W.-h. Liang, C.-q. Ou, J.-x. He, L. Liu, H. Shan, C.-l. Lei, D. S. Hui, et al. Clinical characteristics of coronavirus disease 2019 in china. *New England journal of medicine*, 382(18):1708–1720, 2020.

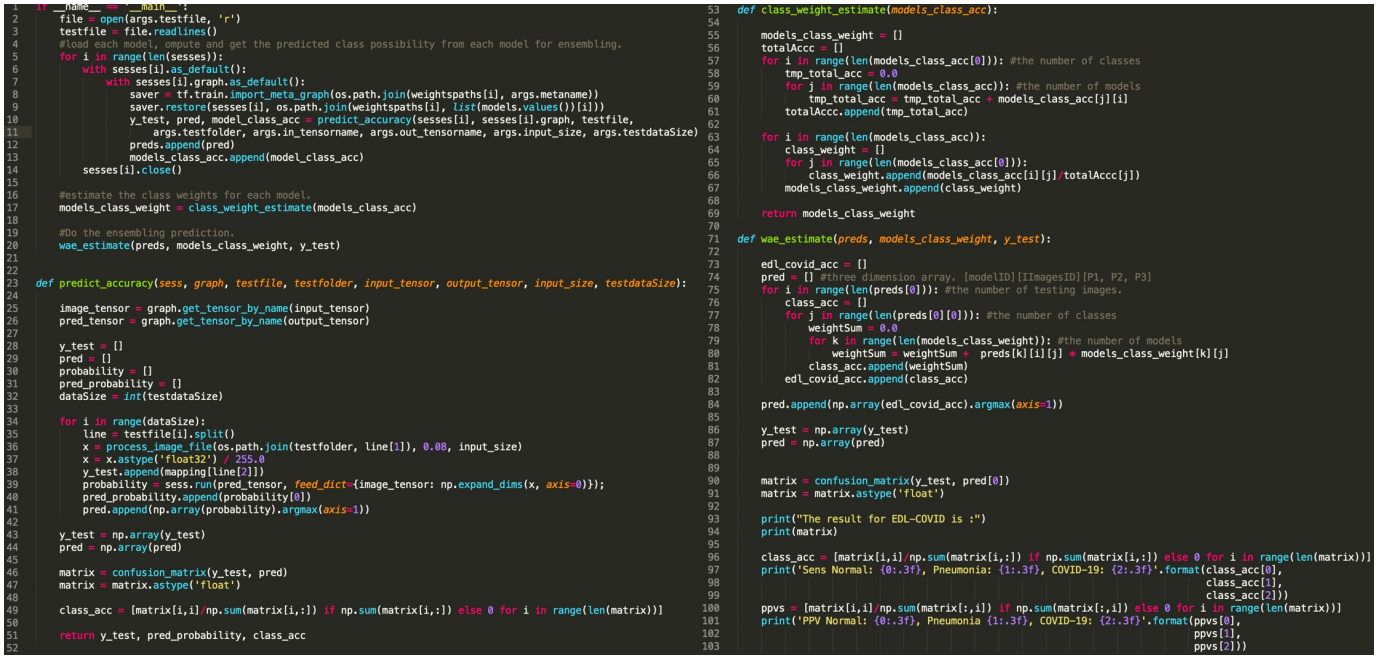
[16] C. Huang, Y. Wang, X. Li, L. Ren, J. Zhao, Y. Hu, L. Zhang, G. Fan, J. Xu, X. Gu, et al. Clinical features of patients infected with 2019 novel coronavirus in wuhan, china. *The lancet*, 395(10223):497–506, 2020.

[17] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.



```
# define custom learning rate schedule
class CosineAnnealingLearningRateSchedule():
        # constructor
        def __init__(self, n_epochs, n_cycles, lrate_max, verbose=0):
                self.epochs = n_epochs
                self.cycles = n_cycles
                self.lr_max = lrate_max
                self.lrates = list()

        # calculate learning rate for an epoch
        def cosine_annealing(self, epoch, n_epochs, n_cycles, lrate_max):
                epochs_per_cycle = math.floor(n_epochs/n_cycles)
                cos_inner = (math.pi * (epoch % epochs_per_cycle)) / (epochs_per_cycle)
                return lrate_max/2 * (math.cos(cos_inner) + 1)

        # calculate and set learning rate at the start of the epoch
        def on_epoch_begin(self, epoch, logs=None):
                # calculate learning rate
                lr = self.cosine_annealing(epoch, self.epochs, self.cycles, self.lr_max)
                # set learning rate
                self.lrates.append(lr)
                return lr

if __name__ == '__main__':

    calrSchedule = CosineAnnealingLearningRateSchedule(n_epochs, n_cycles, learning_rate)

    # output path
    outputPath = './output/'
    runID = args.name + '-lr' + str(learning_rate)
    runPath = outputPath + runID
    pathlib.Path(runPath).mkdir(parents=True, exist_ok=True)

    with open(args.trainfile) as f:
        trainfiles = f.readlines()
    with open(args.testfile) as f:
        testfiles = f.readlines()

    generator = BalanceCovidDataset(data_dir=args.datadir,
                                    csv_file=args.trainfile,
                                    batch_size=batch_size,
                                    input_shape=(args.input_size, args.input_size),
                                    covid_percent=args.covid_percent,
                                    class_weights=[1., 1., args.covid_weight],
                                    top_percent=args.top_percent)

    with tf.compat.v1.Session() as sess:
        tf.compat.v1.get_default_graph()
        saver = tf.compat.v1.train.import_meta_graph(os.path.join(args.weightspath, args.metaname))
        graph = tf.compat.v1.get_default_graph()
    image_tensor = graph.get_tensor_by_name(args.in_tensorname)
    labels_tensor = graph.get_tensor_by_name(args.label_tensorname)
    sample_weights = graph.get_tensor_by_name(args.weights_tensorname)
    pred_tensor = graph.get_tensor_by_name(args.logit_tensorname)

    # Define loss and optimizer
    loss_op = tf.reduce_mean(tf.compat.v1.nn.softmax_cross_entropy_with_logits_v2(
    logits=pred_tensor, labels=labels_tensor)*sample_weights)
    optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=learning_rate)
    train_op = optimizer.minimize(loss_op)

    # Initialize the variables
    init = tf.compat.v1.global_variables_initializer()

    # Run the initializer
    sess.run(init)

    # load weights
    saver.restore(sess, os.path.join(args.weightspath, args.ckptname))

    # save base model
    saver.save(sess, os.path.join(runPath, 'model'))

    # Training cycle
    total_batch = len(generator)
    progbar = tf.keras.utils.Progbar(total_batch)
    for epoch in range(args.epochs):
        learning_rate = calrSchedule.on_epoch_begin(epoch)
        print("Epoch:", '%04d' % (epoch + 1),"learning_rate=","{:.6f}".format(learning_rate))
        optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=learning_rate)
        train_op = optimizer.minimize(loss_op)

        # Initialize the variables
        init = tf.compat.v1.global_variables_initializer()
        sess.run(init)
        for i in range(total_batch):
            # Run optimization
            batch_x, batch_y, weights = next(generator)
            sess.run(train_op, feed_dict={image_tensor: batch_x,
                                          labels_tensor: batch_y,
                                          sample_weights: weights})
            progbar.update(i+1)

        if epoch % display_step == 0:
            pred = sess.run(pred_tensor, feed_dict={image_tensor:batch_x})
            loss = sess.run(loss_op, feed_dict={pred_tensor: pred,
                                                labels_tensor: batch_y,
                                                sample_weights: weights})
            if epoch != 0 and (epoch + 1) % epochs_per_cycle == 0:
                saver.save(sess, os.path.join(runPath, 'model'), global_step=epoch+1, write_meta_graph=False)
                print('Saving checkpoint at epoch {}'.format(epoch + 1))
```

Fig. 10: The core source codes of snapshot models training for EDL-COVID.



```
if __name__ == '__main__':
    file = open(args.testfile, 'r')
    testfile = file.readlines()
    #load each model, ompute and get the predicted class possibility from each model for ensembling.
    for i in range(len(sesses)):
        with sesses[i].as_default():
            with sesses[i].graph.as_default():
                saver = tf.train.import_meta_graph(os.path.join(weightspaths[i], args.metaname))
                saver.restore(sesses[i], os.path.join(weightspaths[i], list(models.values())[i]))
                y_test, pred, model_class_acc = predict_accuracy(sesses[i], sesses[i].graph, testfile,
                    args.testfolder, args.in_tensorname, args.out_tensorname, args.input_size, args.testdataSize)
                preds.append(pred)
                models_class_acc.append(model_class_acc)
        sesses[i].close()

    #estimate the class weights for each model.
    models_class_weight = class_weight_estimate(models_class_acc)

    #Do the ensembling prediction.
    wae_estimate(preds, models_class_weight, y_test)


def predict_accuracy(sess, graph, testfile, testfolder, input_tensor, output_tensor, input_size, testdataSize):

    image_tensor = graph.get_tensor_by_name(input_tensor)
    pred_tensor = graph.get_tensor_by_name(output_tensor)

    y_test = []
    pred = []
    probability = []
    pred_probability = []
    dataSize = int(testdataSize)

    for i in range(dataSize):
        line = testfile[i].split()
        x = process_image_file(os.path.join(testfolder, line[1]), 0.08, input_size)
        x = x.astype('float32') / 255.0
        y_test.append(mapping[line[2]])
        probability = sess.run(pred_tensor, feed_dict={image_tensor: np.expand_dims(x, axis=0)});
        pred_probability.append(probability[0])
        pred.append(np.array(probability).argmax(axis=1))

    y_test = np.array(y_test)
    pred = np.array(pred)

    matrix = confusion_matrix(y_test, pred)
    matrix = matrix.astype('float')

    class_acc = [matrix[i,i]/np.sum(matrix[i,:]) if np.sum(matrix[i,:]) else 0 for i in range(len(matrix))]

    return y_test, pred_probability, class_acc
def class_weight_estimate(models_class_acc):

    models_class_weight = []
    totalAccc = []
    for i in range(len(models_class_acc[0])): #the number of classes
        tmp_total_acc = 0.0
        for j in range(len(models_class_acc)): #the number of models
            tmp_total_acc = tmp_total_acc + models_class_acc[j][i]
        totalAccc.append(tmp_total_acc)

    for i in range(len(models_class_acc)):
        class_weight = []
        for j in range(len(models_class_acc[0])):
            class_weight.append(models_class_acc[i][j]/totalAccc[j])
        models_class_weight.append(class_weight)

    return models_class_weight

def wae_estimate(preds, models_class_weight, y_test):

    edl_covid_acc = []
    pred = [] #three dimension array. [modelID][IImagesID][P1, P2, P3]
    for i in range(len(preds[0])): #the number of testing images.
        class_acc = []
        for j in range(len(preds[0][0])): #the number of classes
            weightSum = 0.0
            for k in range(len(models_class_weight)): #the number of models
                weightSum = weightSum +  preds[k][i][j] * models_class_weight[k][j]
            class_acc.append(weightSum)
        edl_covid_acc.append(class_acc)

    pred.append(np.array(edl_covid_acc).argmax(axis=1))

    y_test = np.array(y_test)
    pred = np.array(pred)


    matrix = confusion_matrix(y_test, pred[0])
    matrix = matrix.astype('float')

    print("The result for EDL-COVID is :")
    print(matrix)

    class_acc = [matrix[i,i]/np.sum(matrix[i,:]) if np.sum(matrix[i,:]) else 0 for i in range(len(matrix))]
    print('Sens Normal: {0:.3f}, Pneumonia: {1:.3f}, COVID-19: {2:.3f}'.format(class_acc[0],
                                                                               class_acc[1],
                                                                               class_acc[2]))
    ppvs = [matrix[i,i]/np.sum(matrix[:,i]) if np.sum(matrix[:,i]) else 0 for i in range(len(matrix))]
    print('PPV Normal: {0:.3f}, Pneumonia {1:.3f}, COVID-19: {2:.3f}'.format(ppvs[0],
                                                                             ppvs[1],
                                                                             ppvs[2]))
```

Fig. 11: The core source codes of snapshot models ensembling for EDL-COVID.

[18] A. Jacobi, M. Chung, A. Bernheim, and C. Eber. Portable chest x-ray in coronavirus disease-19 (covid-19): A pictorial review. *Clinical Imaging*, 2020.

[19] D. Jakubovitz, R. Giryes, and M. R. Rodrigues. Generalization error in deep learning. In *Compressed Sensing and Its Applications*, pages 153–193. Springer, 2019.

[20] A. Kumar, P. K. Gupta, and A. Srivastava. A review of modern technologies for tackling covid-19 pandemic. *Diabetes and Metabolic Syndrome: Clinical Research and Reviews*, 14(4):569 – 573, 2020.

[21] P. Lakhani and B. Sundaram. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2):574–582, 2017.

[22] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] C. Ma, Z. Liu, Z. Cao, W. Song, J. Zhang, and W. Zeng. Cost-sensitive deep forest for price prediction. *Pattern Recognition*, 107:107499, 2020.

[24] H. S. Maghdid, A. T. Asaad, K. Z. Ghafoor, A. S. Sadiq, and M. K. Khan. Diagnosing covid-19 pneumonia from x-ray and ct images using deep learning and transfer learning algorithms. *arXiv preprint arXiv:2004.00038*, 2020.

[25] A. Nair, J. Rodrigues, S. Hare, A. Edey, A. Devaraj, J. Jacob, A. Johnstone, R. McStay, E. Denton, and G. Robinson. A british society of thoracic imaging statement: considerations in designing local imaging diagnostic algorithms for the covid-19 pandemic. *Clinical Radiology*, 75(5):329–334, 2020.

[26] A. Narin, C. Kaya, and Z. Pamuk. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *arXiv preprint arXiv:2003.10849*, 2020.

[27] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. R. Acharya. Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in Biology and Medicine*, page 103792, 2020.

[28] R. Polikar. Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer, 2012.

[29] S. Rajaraman, J. Siegelman, P. O. Alderson, L. S. Folio, L. R. Folio, and S. K. Antani. Iteratively pruned deep learning ensembles for covid-19 detection in chest x-rays. *arXiv preprint arXiv:2004.08379*, 2020.

[30] M. Shen, Y. Zhou, J. Ye, A. A. A. AL-maskri], Y. Kang, S. Zeng, and

S. Cai. Recent advances and perspectives of nucleic acid detection for coronavirus. *Journal of Pharmaceutical Analysis*, 10(2):97 – 101, 2020.
[31] F. Shi, J. Wang, J. Shi, Z. Wu, Q. Wang, Z. Tang, K. He, Y. Shi, and D. Shen. Review of artificial intelligence techniques in imaging data acquisition, segmentation and diagnosis for covid-19. *IEEE Reviews in Biomedical Engineering*, pages 1–1, 2020.
[32] V. Svetnik, T. Wang, C. Tong, A. Liaw, R. P. Sheridan, and Q. Song. Boosting: An ensemble learning tool for compound classification and qsar modeling. *Journal of chemical information and modeling*, 45(3):786–799, 2005.
[33] S. Tabik, A. Gómez-Ríos, J. Martín-Rodríguez, I. Sevillano-García, M. Rey-Area, D. Charte, E. Guirado, J. Suárez, J. Luengo, M. Valero-González, et al. Covidgr dataset and covid-sdnet methodology for predicting covid-19 based on chest x-ray images. *arXiv preprint arXiv:2006.01409*, 2020.
[34] A. Tahamtan and A. Ardebili. Real-time rt-pcr in covid-19 detection: issues affecting the results. *Expert Rev Mol Diagn*, 10(5):453 – 454, 2020.
[35] B. Udugama, P. Kadhiresan, H. N. Kozlowski, A. Malekjahani, M. Osborne, V. Y. Li, H. Chen, S. Mubareka, J. B. Gubbay, and W. C. Chan. Diagnosing covid-19: the disease and tools for detection. *ACS nano*, 14(4):3822–3835, 2020.
[36] L. Wang and A. Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *arXiv preprint arXiv:2003.09871*, 2020.
[37] L. Wynants, B. Van Calster, M. M. J. Bonten, G. S. Collins, T. P. A. Debray, M. De Vos, M. C. Haller, G. Heinze, K. G. M. Moons, R. D. Riley, E. Schuit, L. J. M. Smits, K. I. E. Snell, E. W. Steyerberg, C. Wallisch, and M. van Smeden. Prediction models for diagnosis and prognosis of covid-19 infection: systematic review and critical appraisal. *BMJ-BRITISH MEDICAL JOURNAL*, 369, APR 7 2020.
[38] S. Zhou, Y. Wang, T. Zhu, and L. Xia. Ct features of coronavirus disease 2019 (covid-19) pneumonia in 62 patients in wuhan, china. *American Journal of Roentgenology*, 214(6):1287–1294, 2020.

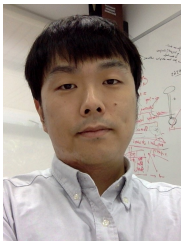
Shanjiang Tang received the PhD degree from School of Computer Engineering, Nanyang Technological University, Singapore in 2015, and the MS and BS degrees from Tianjin University (TJU), China, in Jan 2011 and July 2008, respectively. He is currently an associate professor in College of Intelligence and Computing, Tianjin University, China. His research interests include parallel computing, cloud computing, big data analysis, and machine learning. He has published many papers in TCC, TPDS, TKDE, TSC, SC, ICS, HPDC, etc.

Chunjiang Wang received the B.E. degree from the School of Computer Science and Technology, Tiangong University, Tianjin, China, in 2019. He is currently pursuing an M.S. degree in the College of Intelligence and Computing, Tianjin University (TJU), China. His research interests include deep learning, video analysis, cloud computing, parallel computing.

Jiangtian Nie received her B.Eng degree with honors in Electronics and Information Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2016. She is currently working towards the Ph.D. degree with ERI@N in the Interdisciplinary Graduate School, Nanyang Technological University, Singapore. Her research interests include incentive mechanism design in crowdsensing and game theory.

Neeraj Kumar is a Professor in the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala (Pb.), India. He has published more than 400 technical research papers in top-cited journals and conferences which are cited more than 15000 times from well-known researchers across the globe with current h-index of 67 in Google scholar. His research areas are Green computing and Network management, IoT, Big Data Analytics, Deep learning and cyber-security.

He is serving as editors of ACM Computing Survey, IEEE Transactions on Sustainable Computing, IEEE Systems Journal, IEEE Network Magazine, IEEE Communication Magazine, Elsevier Journal of Networks and Computer Applications, Elsevier Computer Communication, Wiley International Journal of Communication Systems. He is also TPC Chair and member for various International conferences such as IEEE MASS 2020, IEEE MSN2020. He has won the best papers award from IEEE Systems Journal and IEEE ICC 2018, Kansas-city in 2018. He won the best researcher award from parent organization every year from last eight consecutive years.



Yang Zhang is currently an associate professor at Wuhan University of Technology, China. He received B. Eng., B. Eng. (Minor), and M. Eng. from Beijing University of Aeronautics and Astronautics (Beihang Univ.) in 2008, 2010 and 2011, respectively. He obtained a Ph.D. degree in Computer Engineering from Nanyang Technological University (NTU), Singapore, in 2015. He is an associate editor of EURASIP Journal on Wireless Communications and Networking, and technical committee member of Computer Communications.

Zehui Xiong is currently an Assistant Professor in the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design. He received the PhD degree in Nanyang Technological University, Singapore. His research interests include wireless communications, network games and economics, blockchain, and edge intelligence. He has published more than 90 research papers in leading journals and flagship conferences and 4 of them are ESI Highly Cited Papers. He has won 5 Best Paper Awards in international conferences and technical committee. He is now serving as the editor or guest editor for many leading journals including IEEE Transactions. He is the recipient of the Chinese Government Award for Outstanding Students Abroad in 2019, and NTU SCSE Best PhD Thesis Runner-Up Award in 2020.

Ahmed Barnawi is currently a professor at the Faculty of Computing and IT in King Abdulaziz University. He is the managing director of the KAU Cloud computing and Big Data Research group. He acquired his Phd from the University of Bradford, UK, in 2005 and his MSC from UMIST, UK, in 2001. His research interest includes Big data, cloud computing, and advanced mobile robotic applications. He published more than 100 papers in peer reviewed journals.