



HHS Public Access

Author manuscript

Proceedings (IEEE Int Conf Bioinformatics Biomed). Author manuscript; available in PMC 2019 July 24.

Published in final edited form as:

Proceedings (IEEE Int Conf Bioinformatics Biomed). 2018 December ; 2018: 1209–1214. doi:10.1109/BIBM.2018.8621107.

Deep Generative Classifiers for Thoracic Disease Diagnosis with Chest X-ray Images

Chengsheng Mao,

Dept. of Preventive Medicine, Northwestern University, Chicago, IL, USA

Yiheng Pan,

Dept. of EECS, Northwestern University , Chicago, IL, USA

Zexian Zeng,

Dept. of Preventive Medicine, Northwestern University, Chicago, IL, USA

Liang Yao,

Dept. of Preventive Medicine, Northwestern University, Chicago, IL, USA

Yuan Luo

Dept. of Preventive Medicine, Northwestern University, Chicago, IL, USA

Abstract

Thoracic diseases are very serious health problems that plague a large number of people. Chest X-ray is currently one of the most popular methods to diagnose thoracic diseases, playing an important role in the healthcare workflow. However, reading the chest X-ray images and giving an accurate diagnosis remain challenging tasks for expert radiologists. With the success of deep learning in computer vision, a growing number of deep neural network architectures were applied to chest X-ray image classification. However, most of the previous deep neural network classifiers were based on deterministic architectures which are usually very noise-sensitive and are likely to aggravate the overfitting issue. In this paper, to make a deep architecture more robust to noise and to reduce overfitting, we propose using deep generative classifiers to automatically diagnose thorax diseases from the chest X-ray images. Unlike the traditional deterministic classifier, a deep generative classifier has a distribution middle layer in the deep neural network. A sampling layer then draws a random sample from the distribution layer and input it to the following layer for classification. The classifier is generative because the class label is generated from samples of a related distribution. Through training the model with a certain amount of randomness, the deep generative classifiers are expected to be robust to noise and can reduce overfitting and then achieve good performances. We implemented our deep generative classifiers based on a number of well-known deterministic neural network architectures, and tested our models on the chest X-ray14 dataset. The results demonstrated the superiority of deep generative classifiers compared with the corresponding deep deterministic classifiers.

Index Terms

chest X-ray; computer-aided diagnosis; deep learning; generative model; classification

I. Introduction

Thoracic diseases encompass a variety of serious illnesses and morbidities with high prevalence, e.g. pneumonia affect millions of people worldwide each year and about 50,000 people die from pneumonia each year in the United States only [1]. Detecting the thoracic diseases early and correctly can help clinicians to improve patient treatment effectively. Chest X-ray (CXR), also known as chest radiograph, is a projection radiograph of the chest and is used to diagnose conditions affecting the chest, its contents, and nearby structures. Due to its affordable price and quick turnaround, CXR is currently one of the most popular radiological examinations to diagnose thoracic diseases. Currently, reading CXR and giving an accurate diagnosis rely on expert knowledge and medical experience of radiologists. With the increasing amount of CXR images, to handle the heavy and tedious workload of reading the CXR images with subtle texture changes, even the most experienced expert may be prone to make mistakes. Therefore, it is important to develop a Computer-Aided Diagnosis (CAD) system to automatically detect different types of thoracic diseases by reading patients' CXR images.

Developing a CAD system to understand medical images and to diagnose diseases has attracted wide research interests for decades [2]–[4]. However, traditional statistical learning methods, such as Bayesian classifier [5]–[7], SVM [8]–[10] and KNN [11]–[14] etc., are not expert in directly handling the medical images in the high-dimensional pixel-level features. They usually require onerous highly customized feature engineering before classification, thus, they cannot generalize well and is expert labor intensive. With the success of deep learning in computer vision, it is natural to apply deep learning models to assist in disease diagnosis based on medical images. Recently, deep learning based CAD has benefited many biomedical applications [15], e.g., diabetic eye disease diagnosis [16], cancer metastases detection and localization [17], lung nodule detection [18], survival analysis [19] and clinical notes classification [20], [21], etc. In this work, we develop a generative deep neural network architecture and apply it to diagnosing thoracic diseases based on CXR images.

Deep neural networks usually require large-scale datasets for training. Recently, Wang et al. [22] released the datasets ChestX-ray8 and later ChestX-ray14 which is considered one of the largest public chest X-ray dataset (details in Section IV-A). There are 14 thoracic diseases included in ChestX-ray14, i.e., Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening, and Hernia. We focus on this dataset to train a deep generative classifier to diagnose the 14 diseases.

In this paper, we propose using deep generative classifiers to automatically diagnose thorax diseases from CXR images. A Deep Generative Classifier (DGC) contains an encoder network and a classifier network. The encoder network encodes each input CXR image to a low-dimensional distribution of latent features. The classifier network classifies a sample

using features drawn from the latent low-dimensional distribution and outputs probabilities of class label assignment. Our main idea is to use the random sampling connection between the encoder network and the classifier network rather than a deterministic connection which is adopted in most previous frameworks, e.g. AlexNet [23], ResNet [24], VGGNet [25] and DenseNet [26]. Intuitively, by using the random sampling connection, the model would be more robust to noise and can reduce overfitting. An overview of our generative classifier is shown in Fig. 1. As shown in Fig. 1, given a CXR image, our model will output a list of probabilities corresponding to a list of thorax diseases, and a low-dimensional distributional representation of this image as a by-product which can be used for other classification or clustering tasks.

II. Related work

A. Deep Learning for CXR Image Analysis

Many research efforts have been directed towards automatic detection of thorax diseases based on diverse data generated by chest X-ray scanning. Before ChestX-ray14 dataset was released, there were also some works about thoracic disease classification on some relatively small dataset. Bar et al. [27] applied a pre-trained Decaf Convolutional Neural Network (CNN) model [28] to classify 8 thoracic diseases on a dataset of 433 images. Lajhani et al. [29] ensembled both AlexNet [23] and GoogleNet [30] for tuberculosis classification on a dataset of 1007 posteroanterior CXR images.

Since Wang et al. [22] released the datasets ChestX-ray14, there has been an increasing amount of research on CXR analysis using deep neural networks. Wang et al. [22] also evaluated the performance of four classic deep learning architectures (i.e., AlexNet [23], VGGNet [25], GoogLeNet [30] and ResNet [24]) to diagnose 14 thoracic diseases from CXR images. To explore the correlation among the 14 diseases, Yao et al. [31] used a Long-short Term Memory (LSTM) [32] to repeatedly decode the feature vector from a DenseNet [26] and produced one disease prediction at each step. Kumar et al. [33] explored suitable loss functions to train a convolutional neural network (CNN) from scratch and presented a boosted cascaded CNN for multi-label CXR classification. Rajpurkar et al. [34] achieved good multi-label classification results by fine-tuning a pre-trained DenseNet121 [26]. Li et al. [35] used a pre-trained ResNet to extract features and divided them into patches which are passed through a fully convolutional network (FCN) [36] to obtain a disease probability map.

Previous deep learning architectures all had a deterministic mapping between encoded features and CXR classification. In this paper, we propose using deep generative classifiers for classifying thoracic disease with CXR images. By introducing a generative process, the learned DGC model should be more robust to noise and reduce the overfitting issue.

B. Variational Autoencoder

The deep generative classifiers have similar traits with Variational Auto-Encoder (VAE) [37]. In VAE, a high-dimensional sample is encoded to a low-dimensional feature distribution, a sample from this distribution is decoded to the original high-dimensional

sample. To generate new samples better, VAE needs to constrain the low-dimensional distribution to a certain known distribution. VAE was usually used to reduce the data dimension or to generate new samples, but was rarely used for supervised classification directly. In this paper, we leverage VAE to design a generative classification model where the class label was generated from the latent low-dimensional distribution.

III. Models

Our purpose is to judge whether one or more thoracic diseases are presented in a CXR image, it can be modeled as a multi-label classification problem. We integrate the losses of the multiple objectives for multiple labels, and tackle this problem using deep generative classifiers. In this section, we explicitly describe the technical details of the proposed deep generative classifiers. First, we present the detailed architecture of the deep generative classifier. Then, we explain the training strategy of the model. Finally, we give a probabilistic interpretation of our model.

A. Architecture

A DGC receives a batch of CXR image as input and computes a list of probabilities for each disease. In the framework, each input image is encoded to a latent low-dimensional distribution by the encoder network, the classifier network classifies the sample based on features drawn from the latent low-dimensional distribution to generate a probabilistic output. Fig. 2 illustrates the detailed framework of DGC. Given a CXR image input X , the computation flows through a series of sub-modules, including the encoder, the transition layer, the sampling layer, and the classifier. Next, we explain the submodules in detail

1) Encoders: As shown in Fig. 2, the encoder in our network is leveraged from a part of a pre-trained model on ImageNet [38], [39], e.g. AlexNet [23], VGGNet [25], ResNet [24] and DenseNet [26]. For the pre-trained models, we discard the fully-connected layers and classification layers, and keep the feature layers to extract feature maps for CXR images. Through the encoder, an original image X ($224 \times 224 \times 3$) is encoded to C feature maps with size $S \times S$ represented by $En(X; \Phi_e)$ where Φ_e is the set of trainable parameters of the encoder.

2) Transition Layer: The transition layer is to transform the output feature maps of the encoder to a flat feature vector that has a uniform dimension for different images. Due to the variety of the output feature maps of the encoder we adopt, we use a convolution layer with a certain number (D) of filters to get D feature maps, and then adopt batch normalization [40] right after the convolution layer to ease the training process. Finally, a max pooling layer [41] with kernel size equal to the feature map size is applied to reduce each feature map to $1 \times 1 \times D$. The $1 \times 1 \times D$ feature map is then squeezed to a D -dimensional feature vector, represented by

$$\mu = Trans(En(X; \Phi_e); \Phi_t) \quad (1)$$

where Φ_t is the set of trainable parameters of the transition layer.

3) Sampling Layer: Instead of using the output of the transition layer directly as features, we treat it as the mean of a latent distribution. We let this sampling layer samples from this latent distribution to produce a feature vector. In our design, the features of a CXR image adopt a latent distribution whose parameters are computed from the upstream networks or jointly learned with the network parameters by Stochastic Gradient Descent (SGD) [42]. In our architecture, we treat the latent distribution as Gaussian distributions where the mean is computed from the upstream network, i.e., the output of the transition layer, and the covariance matrix is diagonal with the diagonal elements learned with the network parameters. Thus, for a D -dimensional latent Gaussian distribution, the sampling layer has a total of D parameters each of which corresponds to the standard deviation of each dimension. The output of sampling layer is a feature vector sampled from the latent distribution. The sampling layer is crucial to make the model generative rather than deterministic. The sampling layer can be represented by

$$z \sim N(\mu, \sigma) \quad (2)$$

where μ is the output of the transition layer and σ is the trainable parameter of the sampling layer.

4) Classifier: Since we use a low-dimensional representation as the feature vector of an input CXR image, we need a classifier to discriminate if a disease is presented. For the 14 thoracic diseases, there are 14 probabilist outputs considered as a 14-dimensional vector. In our architecture, we use a fully-connected layer with 14 score outputs that are then transformed by a sigmoid function to probabilistic outputs. The classifier is represented by

$$p = f(z; \Phi_c) \quad (3)$$

where z is the output of the sampling layer and Φ_c is the parameters of the classifier.

B. Training Strategy

We next describe two crucial strategies related to training that concern the loss function for multi-label classification and the reparameterization for distribution parameter learning.

1) Loss Function: To train our network, we must define a loss function for the multiple outputs corresponding to the 14 diseases. The true label of each image is considered a 14-dimensional vector $y = [y_1, \dots, y_i, \dots, y_C]$, $y_i \in \{0, 1\}$, $C = 14$ where y_i represents whether the corresponding disease is presented, i.e., 1 for presence and 0 for absence. An all-zero vector represents “No Findings” in the 14 diseases. We compute the cross entropy loss l_i for disease i as (4), where p_i is the output probability of disease i .

$$l_i = -y_i * \ln p_i - (1 - y_i) * \ln(1 - p_i) = \begin{cases} -\ln p_i, & y_i = 1; \\ -\ln(1 - p_i), & y_i = 0 \end{cases} \quad (4)$$

For a mini-batch with n samples, the corresponding targets are n C -dimensional (0,1)-vectors which can be considered as a (0,1)-matrix with shape $n \times C$ which we call target matrix. Since there usually are only a few pathologies present in a CRX image, the target matrix should be a sparse matrix where there are many more '0's than '1's. To balance the influence of '0's and '1's on the loss, we weight the losses for different classes. In the mini-batch, we design the weights as (5), where $|P|$ and $|N|$ are, respectively, the number of '1's and '0's in the target matrix of the mini-batch. Thus, combining (4) and (5), we define the loss function for a CRX image as (6)

$$w_i = \begin{cases} \frac{|P| + |N|}{|P|}, & y_i = 1; \\ \frac{|P| + |N|}{|N|}, & y_i = 0 \end{cases} \quad (5)$$

$$L(p, y) = \sum_{i=1}^{14} w_i l_i = -\frac{|P| + |N|}{|P|} \sum_{y_i=1} \ln p_i - \frac{|P| + |N|}{|N|} \sum_{y_i=0} \ln(1 - p_i) \quad (6)$$

2) Reparameterization Trick: Since the sampling operation is not differentiable, we use the reparameterization trick [37]. In our architecture, the latent distribution is Gaussian (assumed $\mathcal{N}(\mu, \sigma^2)$). Due to random sampling, the derivative of a sample z from this distribution with respect to μ and σ cannot be directly obtained. To learn the parameters with SGD, we construct a deterministic relation between z and μ, σ by introducing an auxiliary variable ϵ , $\epsilon \sim \mathcal{N}(0, 1)$. Thus, sampling from $\mathcal{N}(\mu, \sigma^2)$ is equal to sampling ϵ from $\mathcal{N}(0, 1)$ and then computing the sample by

$$z = \mu + \epsilon\sigma, \quad \epsilon \sim \mathcal{N}(0, 1) \quad (7)$$

The derivative of z can then be easily computed as

$$\begin{aligned} \partial z / \partial \mu &= 1 \\ \partial z / \partial \sigma &= \epsilon \end{aligned} \quad (8)$$

Algorithm 1

Deep Generative Classifier Training. N is the number of samples in the training set. The parameter list $[\Phi = \Phi_e, \Phi_t, \sigma, \Phi_c]$.

Input:

Training set $Tr = \{(X_i, Y_i)\}, i \in \{1, \dots, N\}$;
 Batch size $|B|$;
 The terminating condition (e.g., a number of epochs or early stop).

Output: The updated parameter list of DGC Φ .

```

1: Initialize the parameter list of DGC  $\Phi$ 
2: repeat
3:   Randomly split the training set  $Tr$  into  $n$  mini-batches  $\{B_1, \dots, B_n\}$  with each batch has  $|B|$  samples, the last batch can have less samples
4:   for  $B$  in  $\{B_1, \dots, B_n\}$  do
5:     Count the number of '0's  $|N|$  and number of '1's  $|P|$  in the target matrix  $Y$ 
6:      $\mathcal{L} = 0$ 
7:     for  $(X, y)$  in  $B$  do
8:       Random sampling  $\epsilon$  with the same shape as  $\sigma$  from  $\mathcal{N}(0,1)$ 
9:       Compute the output  $p$  of the model by (9)
10:      Compute the loss  $L$  by (6)
11:       $\mathcal{L} = \mathcal{L} + L$ 
12:    end for
13:    Compute the gradient  $g = \nabla_{\Phi} \mathcal{L}$ 
14:    Update parameters  $\Phi$  using gradients  $g, \Phi = \text{update}(\Phi, g)$ . e.g. SGD [42] or Adam [43]
15:  end for
16: until the terminating condition is met
17: return  $\Phi$ 

```

Combining (1), (2), (7), (3), we can compute the output of the model by (9). Thus, through the reparameterization trick, the network can be trained by SGD.

$$p = f(\text{Trans}(\text{En}(X; \Phi_e); \Phi_t) + \epsilon\sigma; \Phi_c) \quad (9)$$

3) Training Algorithm: The training algorithm of DGC is briefly described in Algorithm 1. It starts by initializing the model parameter Φ (Line 1), then repeats the training procedure until the terminating condition (e.g., a threshold number of epochs or early stop) is met (Lines 2–16). In an epoch in the training procedure, the training set is split into batches (Line 3). For each batch in the training set (Line 4–15), we compute the loss of the batch (Lines 5–12), and the gradients with respect to the parameters Φ for error backpropagation (Line 13). We then update the parameters Φ using the gradients and a certain optimization algorithm (Line 14). Finally, if the terminating condition is met, the model is updated with parameters Φ .

C. Test Strategy

After the model is trained, in the test process, a CXR image can be encoded to a Gaussian distribution $N(\mu, \sigma)$. To classify the image, we need sampling the feature vector of a sample from the distribution, and then input the sample to the classifier to get the probability output. However, the sampling process is random, that is, the output may change with the different samplings. Thus, to achieve a consistent classification output in the test process, we use the expectation of the distribution as the sample input to the classifier, i.e.,

$$z = \mathbb{E}N(\mu, \sigma) = \mu \quad (10)$$

IV. Experiments

In this section, we evaluate the performance of the proposed DGC model and compared it with the corresponding deep deterministic classifiers.

A. ChestX-ray14 Dataset

We evaluate and validate the DGC model using the ChestX-ray14 dataset¹ [22]. The ChestX-ray14 dataset consists of 112,120 frontal-view chest X-ray images of 30,805 unique patients. There are 14 thoracic disease labels included in these images (i.e., Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening and Hernia). The labeled ground truth is obtained through Natural Language Processing (NLP) on the patients' diagnostic reports, the labeling accuracy is estimated to be > 90% [22]. Out of the 112,120 CXR images, 51,708 contains one or more pathologies. The remaining 60,412 images are considered normal. In our experiments, we split the dataset into training-validation set and test set on the patient level using the publicly available data split list¹. All studies from the same patient will only appear in either training-validation set or testing set. The detailed information about the number of images and patients in training-validation set and test set is shown in Table I.

The training-validation set is further randomly split into a training set and a validation set, 7/8 as the training set and 1/8 as the validation set. The training set is used to train the model and the validation set is used to select a model according to the performance.

B. Preprocessing

Since the ImageNet pre-trained models only accept 3-channel images with size 224×224 , while the images in ChestX-ray14 dataset are 1024×1024 grayscale, we convert the grayscale images to 3-channel RGB images, down-scale the original resolution to 256×256 and then crop the image to 224×224 at the center. We normalized the images by mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]) according to the images from ImageNet. We do not apply any data augmentation techniques.

¹<https://nihcc.app.box.com/v/ChestXray-NIHCC>

C. Experimental setting

Our experimental setting includes the following aspects.

1) Encoder: In our experiments, we tried 6 pretrained models as the encoder in our architecture, including AlexNet[23], VGGNet16 [25], ResNet50 [24] and DenseNet121 [26], DenseNet161 [26], DenseNet201 [26]. As described in Section III-A1, we discarded the high-level fully-connected layers and classification layers of the pretrained models and only used the feature layers as the encoder. As shown in Fig. 2, different encoders have different inner structure and have different parameters. We respectively denote the DGC based on these encoder as DGC-AlexNet (G-AN), DGC-VGGNet16 (G-VN16), DGC-ResNet50 (G-RN50), DGC-DenseNet121 (G-DN121), DGC-DenseNet161 (G-DN161) and DGC-DenseNet201 (GDN201).

2) Baselines: The specific part of our architecture is the sampling layer which improves the model robustness to noise. Thus, we remove the sampling layer in the architecture and consider the remaining parts as the baselines, i.e., the output of the transition layer is directly input to the classifier in baselines. Correspondingly, we have 6 baselines, respectively denoted as AlexNet (AN), VGGNet16 (VN16), ResNet50 (RN50), DenseNet121 (DN121), DenseNet161 (DN161) and DenseNet201 (DN201).

3) Hyperparameters:

- Initialization: the encoder was initialized with a pretrained model. In the transition layer, the convolution layer was initialized with kaiming uniform initialization [44], the batch normalization layer was initialized with weights drawn from the uniform distribution $U(0, 1)$ and bias as zero. To ensure the positivity of σ in the sampling layer, we computed σ from $\sigma^2 = e^v$ where v was initialized with values drawn from the uniform distribution $U(0, 1)$. In the classifier, the parameters of the fully-connected layer were initialized with kaiming uniform initialization [44]
- Latent dimension: we set the latent dimension to 1024, i.e., each image was encoded to 1024-dimensional feature vector adopting Gaussian distribution.
- Batch size: the batch size was 16, i.e., the model updated parameters per 16 images.
- Optimizer: we trained the model by Adam optimizer [43] with parameters $lr = 10^{-5}$, $\beta = (0.9, 0.999)$, $eps = 10^{-8}$, $weight_decay = 0$
- Terminating condition: we terminated the training procedure when it repeated 10 epochs. In each epoch, we tested the model on the validation set and save the model with the best performance.

4) Evaluation: Since our model outputs the probability for each disease, it is natural to plot a Receiver Operating Characteristic curve (ROC) for each disease. In our experiments, we calculated the Area Under ROC (AUC) for each disease, and evaluated the classification performance by the average AUC of the 14 diseases.

D. Implementation

Our models and all the baselines were implemented using Python 3.6 with PyTorch framework on a CentOS Linux server. The models were trained and tested on 4 Tesla K40 GPUs. We made our source code publicly available at <https://github.com/mocherson/deep-generative-classifiers>

E. Classification Results and Analysis

In each of the 10 epochs of training, we evaluated the model on the validation set, and selected the model that achieved the highest classification performance to test on the test set. We repeated the training and classification procedure 5 times and reported the average results. The classification results for each model and the corresponding baselines are given in Table II. From Table II, overall, DGCs have higher classification performance than the corresponding baselines. The results, together with the fact that the only difference between DGC and its baselines is the sampling layer as described in Section IV-C2, suggest that adding a sampling layer can improve the classification performance of a deterministic classifier.

From Table II, as for the average AUC of the 14 diseases, AlexNet encoder has the worst performance and VGGNet16 encoder have the best performance, DGC can improve the most when the encoder is AlexNet (AUC from 0.7619 to 0.7654), and improve the least when the encoder is VGG16 (AUC from 0.7875 to 0.7877). Additionally, DGC-AlexNet can consistently outperform AlexNet for all the 14 diseases. For “Mass”, “Pneumothorax” and “Consolidation”, the DGC classifiers can consistently outperform the deterministic classifiers for all the encoders.

Horizontal comparison shows that different classification models achieve different classification performances even for the same disease. In most cases, classifiers based on VGGNet16 can outperform other types of classifiers. Out of the 14 diseases, 8 diseases achieve the best performance on VGGNet16 encoder, 5 are based on DenseNet161 and only 1 (“Hernia”) achieves the best performance on DenseNet201 encoder. From Table II, we can also see that only 2 diseases (“Cardiomegaly” and “Infiltration”) can achieve their best classification results on deterministic classifier (VGGNet16), all of the other diseases achieve their best performance with DGC models.

Vertical comparison shows that the same classification model can achieve different classification performance for the 14 diseases. The most easily identified disease is “Hernia” (AUC=0.9113) and the least easily identified disease is “Infiltration” (AUC=0.6894). From Table II, the diseases that are difficult to identify include “Infiltration”, “Pneumonia”, “Nodule”, “Consolidation” and “Atelectasis” (AUC<0.75); the easily identified diseases include “Hernia”, “Emphysema” and “Cardiomegaly” (AUC>0.85).

We also plot the ROC curves of different DGC models and their baselines over the 14 diseases for one of the 5 runs, shown in Fig. 3. From Table II and Fig. 3, for most of the diseases, a DGC model can usually outperform its corresponding deterministic model. Moreover, we can see that the ROC curves of diseases “Infiltration”, “Pneumonia”, “Nodule” and “Consolidation” are relatively flat, which means that the classifications on

these diseases are not as good as other diseases like “Hernia”, “Emphysema” and “Cardiomegaly”. This is consistent with our previous analysis.

V. Conclusion

In this paper, we proposed using deep generative classifiers to diagnose thoracic diseases with chest X-ray images. The deep generative classifiers contained an encoder and a classifier. The encoder encoded the input CXR image to a low-dimensional distribution, the classifier classified using features drawn from this distribution. Different from the deterministic classifiers, in the training process, generative classifiers introduce Gaussian noise and learn the variance in the training process. Through training the model with a certain amount of noise, the learned model was expected to be more robust to noise and to reduce overfitting. In this paper, we implemented the DGC architecture by adding a sampling layer between the encoder and the classifier. We used the reparameterization trick to train the DGC model through SGD. Our experimental results on ChestX-ray14 dataset demonstrated the effectiveness of the DGC models.

The proposed DGC has similar traits with variational autoencoders (VAE). VAE is considered unsupervised, because it is trained without labels, its target is to reconstruct the original input. However, if there is a label corresponding to an image and the target is to predict the label for new images, we can solve the supervised classification problem using an adapted VAE and reconstructing the labels. This is the key idea of DGC. The difference between DGC and a deep deterministic classifier (e.g., AlexNet, VGGNet) is similar to the difference between VAE and a general autoencoder [45].

In our architecture, we used a complex model to identify the 14 diseases, and learned the model by a sparsity-weighted cross entropy loss, while the weights for different diseases are the same, i.e., we equally regarded all the 14 diseases and learned a generic latent low-dimensional distribution for different diseases. This may somewhat influence the classification results of a certain disease. If only a certain disease is considered, one should train a specific model based on the loss corresponding to this disease. We will experiment with this setting in the future work. In addition, we did not explore the pathology localization problem using DGC, which will also be a part of our future work.

Acknowledgments

This work was supported in part by NIH Grant 1R21LM012618-01.

REFERENCES

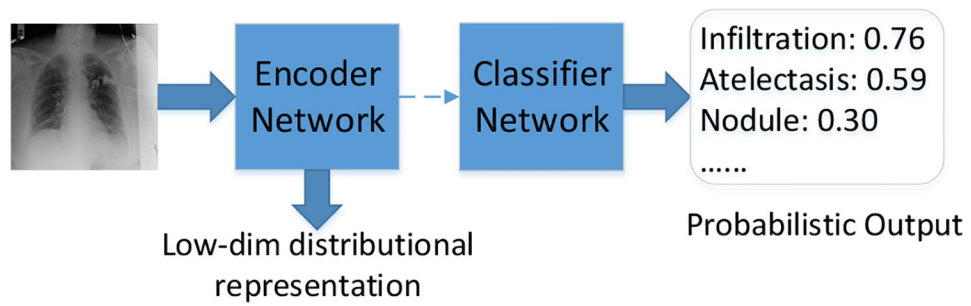
- [1]. (2017) Pneumonia can be prevented vaccines can help. [Online]. Available: <https://www.cdc.gov/features/pneumonia/index.html>
- [2]. Murphy K, van Ginneken B, Schilham AM, De Hoop B, Gietema H, and Prokop M, “A large-scale evaluation of automatic pulmonary nodule detection in chest ct using local image features and k-nearest-neighbour classification,” *Medical image analysis*, vol. 13, no. 5, pp. 757–770, 2009. [PubMed: 19646913]
- [3]. Chen S, Suzuki K, and MacMahon H, “Development and evaluation of a computer-aided diagnostic scheme for lung nodule detection in chest radiographs by means of two-stage nodule

enhancement with support vector classification,” *Medical physics*, vol. 38, no. 4, pp. 1844–1858, 2011. [PubMed: 21626918]

- [4]. Papadopoulos A, Fotiadis DI, and Likas A, “Characterization of clustered microcalcifications in digitized mammograms using neural networks and support vector machines,” *Artificial Intelligence in Medicine*, vol. 34, no. 2, pp. 141–150, 2005. [PubMed: 15894178]
- [5]. Domingos P and Pazzani M, “Beyond independence: Conditions for the optimality of the simple bayesian classifier,” in *Proc. 13th Intl. Conf. Machine Learning*, 1996, pp. 105–112.
- [6]. Hu B, Mao C, Zhang X, and Dai Y, “Bayesian classification with local probabilistic model assumption in aiding medical diagnosis,” in *Bioinformatics and Biomedicine (BIBM)*, 2015 IEEE International Conference on. IEEE, 2015, pp. 691–694.
- [7]. Wang X-Z, He Y-L, and Wang DD, “Non-naive bayesian classifiers for classification problems with continuous attributes,” *Cybernetics, IEEE Transactions on*, vol. 44, no. 1, pp. 21–39, 2014.
- [8]. Cortes C and Vapnik V, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [9]. Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, and Lin C-J, “Liblinear: A library for large linear classification,” *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.
- [10]. Chang C-C and Lin C-J, “Libsvm: a library for support vector machines,” *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.
- [11]. Larose DT and Larose CD, “k-nearest neighbor algorithm,” *Discovering Knowledge in Data: An Introduction to Data Mining*, Second Edition, pp. 149–164, 2006.
- [12]. Cover T and Hart P, “Nearest neighbor pattern classification,” *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [13]. Mao C, Hu B, Moore P, Su Y, and Wang M, “Nearest neighbor method based on local distribution for classification,” in *Advances in Knowledge Discovery and Data Mining*. Springer, 2015, pp. 239–250.
- [14]. Mao C, Hu B, Wang M, and Moore P, “Learning from neighborhood for classification with local distribution characteristics,” in *Neural Networks (IJCNN)*, 2015 International Joint Conference on. IEEE, 2015, pp. 1–8.
- [15]. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, van der Laak JA, Van Ginneken B, and Sánchez CI, “A survey on deep learning in medical image analysis,” *Medical image analysis*, vol. 42, pp. 60–88, 2017. [PubMed: 28778026]
- [16]. Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, Venugopalan S, Widner K, Madams T, Cuadros J et al., “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs,” *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016. [PubMed: 27898976]
- [17]. Liu Y, Gadepalli K, Norouzi M, Dahl GE, Kohlberger T, Boyko A, Venugopalan S, Timofeev A, Nelson PQ, Corrado GS et al., “Detecting cancer metastases on gigapixel pathology images,” *arXiv preprint arXiv:170302442*, 2017.
- [18]. Setio AAA, Traverso A, De Bel T, Berens MS, van den Bogaard C, Cerello P, Chen H, Dou Q, Fantacci ME, Geurts B et al., “Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: the luna16 challenge,” *Medical image analysis*, vol. 42, pp. 1–13, 2017. [PubMed: 28732268]
- [19]. Zhu X, Yao J, Zhu F, and Huang J, “Wsis: Making survival prediction from whole slide histopathological images,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7234–7242.
- [20]. Luo Y, Cheng Y, Uzuner O, Szolovits P, and Starren J, “Segment convolutional neural networks (seg-cnns) for classifying relations in clinical notes,” *Journal of the American Medical Informatics Association*, vol. 25, no. 1, pp. 93–98, 2017.
- [21]. Luo Y, “Recurrent neural networks for classifying relations in clinical notes,” *Journal of biomedical informatics*, vol. 72, pp. 85–95, 2017. [PubMed: 28694119]
- [22]. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, and Summers R, “Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3462–3471.

- [23]. Krizhevsky A, Sutskever I, and Hinton GE, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.
- [24]. He K, Zhang X, Ren S, and Sun J, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- [25]. Simonyan K and Zisserman A, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:14091556, 2014.
- [26]. Huang G, Liu Z, Van Der Maaten L, and Weinberger KQ, "Densely connected convolutional networks," in CVPR, vol. 1, no. 2, 2017, p. 3.
- [27]. Bar Y, Diamant I, Wolf L, Lieberman S, Konen E, and Greenspan H, "Chest pathology detection using deep learning with non-medical training," in ISBI. Citeseer, 2015, pp. 294–297.
- [28]. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, and Darrell T, "Decaf: A deep convolutional activation feature for generic visual recognition," in International conference on machine learning, 2014, pp. 647–655.
- [29]. Lakhani P and Sundaram B, "Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks," Radiology, vol. 284, no. 2, pp. 574–582, 2017. [PubMed: 28436741]
- [30]. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, and Rabinovich A, "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
- [31]. Yao L, Poblens E, Dagunts D, Covington B, Bernard D, and Lyman K, "Learning to diagnose from scratch by exploiting dependencies among labels," arXiv preprint arXiv:171010501, 2017.
- [32]. Hochreiter S and Schmidhuber J, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997. [PubMed: 9377276]
- [33]. Kumar P, Grewal M, and Srivastava MM, "Boosted cascaded convnets for multilabel classification of thoracic diseases in chest radiographs," in International Conference Image Analysis and Recognition. Springer, 2018, pp. 546–552.
- [34]. Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T, Ding D, Bagul A, Langlotz C, and Shpanskaya K, "CheXnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," arXiv preprint arXiv:171105225, 2017.
- [35]. Li Z, Wang C, Han M, Xue Y, Wei W, Li L-J, and Li F, "Thoracic disease identification and localization with limited supervision," arXiv preprint arXiv:171106373, 2017.
- [36]. Long J, Shelhamer E, and Darrell T, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.
- [37]. Kingma DP and Welling M, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114, 2013.
- [38]. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al., "Imagenet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.
- [39]. Deng J, Dong W, Socher R, Li L-J, Li K, and Fei-Fei L, "Imagenet: A large-scale hierarchical image database," in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. Ieee, 2009, pp. 248–255.
- [40]. Ioffe S and Szegedy C, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37. JMLR. org, 2015, pp. 448–456.
- [41]. Boureau Y-L, Ponce J, and LeCun Y, "A theoretical analysis of feature pooling in visual recognition," in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 111–118.
- [42]. Sutskever I, Martens J, Dahl G, and Hinton G, "On the importance of initialization and momentum in deep learning," in International conference on machine learning, 2013, pp. 1139–1147.
- [43]. Kingma DP and Ba JL, "Adam: A method for stochastic optimization," in Proc. 3rd Int. Conf. Learn. Representations, 2014.

- [44]. He K, Zhang X, Ren S, and Sun J, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
- [45]. Hinton GE and Zemel RS, “Autoencoders, minimum description length and helmholtz free energy,” in Advances in neural information processing systems, 1994, pp. 3–10.

**Fig. 1:**

An overview of the deep generative classifier for thorax disease diagnosis. The network reads chest X-ray images and produces a probability score for each thorax disease. The dashed arrow between the encoder network and the classifier network denotes the two networks are connected by random sampling.

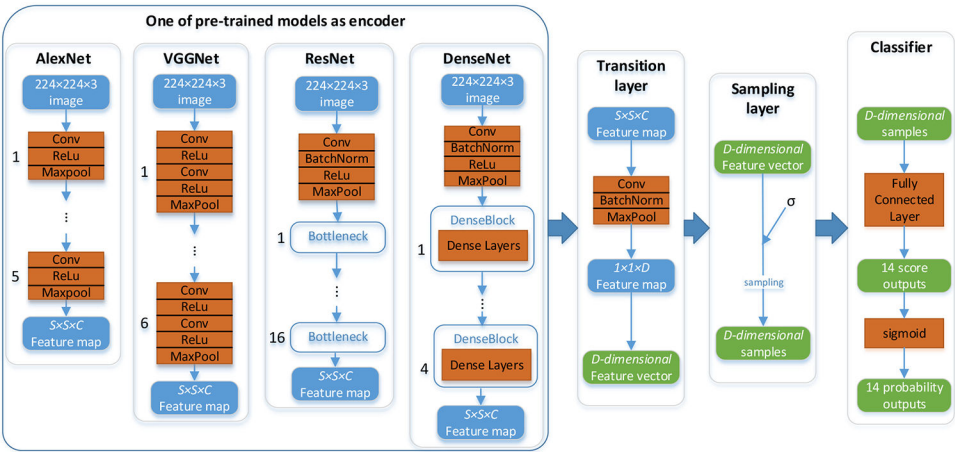


Fig. 2:
The framework of deep generative classifiers.

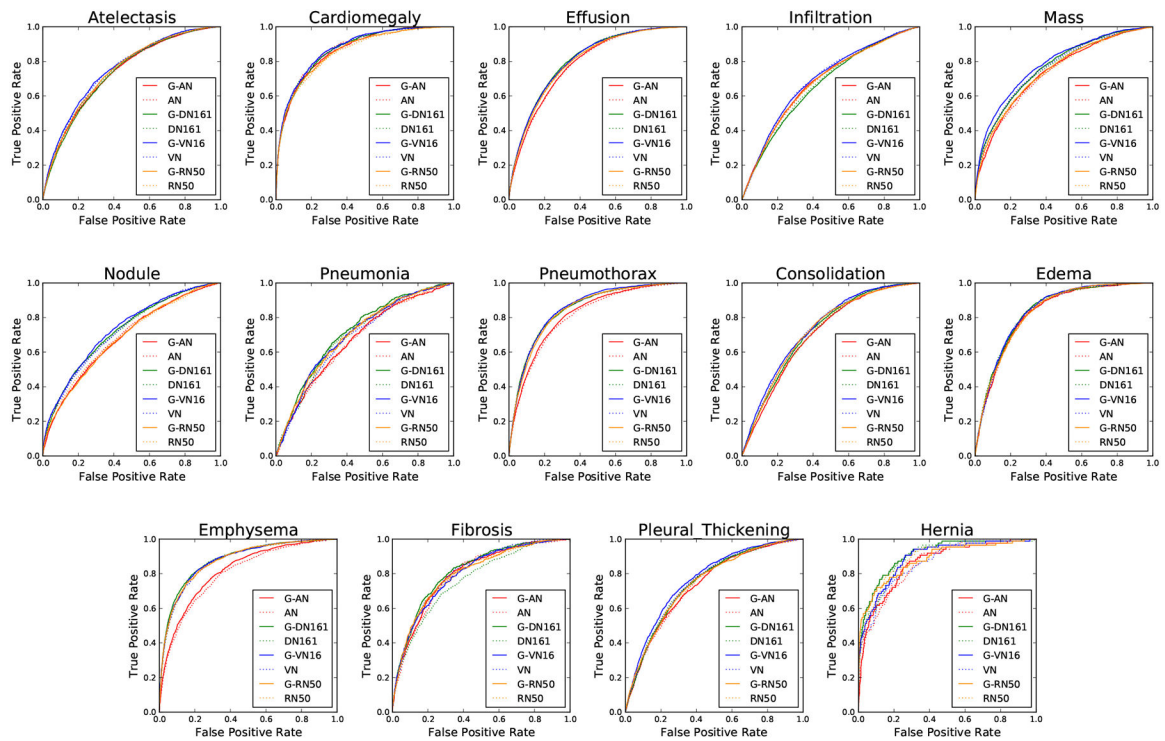


Fig. 3: ROC curves of different DGC models (i.e., DGC-AlexNet, DGC-DenseNet161, DGC-VGGNet16 and DGC-ResNet50) and their baselines over the 14 diseases.

Table I:

Number of images and patients in the ChestX-ray14 dataset

Diseases	Entire Set		Train- val Set ²		Test Set	
	#imgs ³	#pts ⁴	#imgs	#pts	#imgs	#pts
Atelectasis	11535	4974	8280	4182	3255	792
Cardiomegaly	2772	1565	1707	1228	1065	337
Consolidation	4667	2150	2852	1617	1815	533
Edema	2303	1073	1378	747	925	326
Effusion	13307	4273	8659	3502	4648	771
Emphysema	2516	1046	1423	762	1093	284
Fibrosis	1686	1260	1251	1003	435	257
Hernia	227	134	141	102	86	32
Infiltration	19870	8031	13782	7111	6088	920
Mass	5746	2550	4034	2115	1712	435
Nodule	6323	3390	4708	2855	1615	535
PT ¹	3385	2006	2242	1559	1143	447
Pneumonia	1353	955	876	697	All	258
Pneumothorax	5298	1484	2637	1080	2661	404
Normal	60412	16405	50500	14892	9912	1513
Summary	112120	30805	86524	28008	25596	2797

¹PT — Pleural Thickening;²Train-val set — Training-validation set;³#imgs — the number of CXR images;⁴#pts — the number of patients.

Table II:

comparison of AUC scores between DGCs and the corresponding baselines on ChestX-ray14 dataset. The better results based on the same encoder are **bolded**. The best results among all the models are colored red.

Diseases	AN	G-AN	RN50	G-RN50	DN201	G-DN201	DN121	G-DN121	DN161	G-DN161	VN16	G-VN16
Atelectasis	0.7327	0.7359	0.7407	0.7418	0.7426	0.7415	0.7391	0.7378	0.7433	0.7421	0.7488	0.7495
Cardiomegaly	0.8681	0.8699	0.8553	0.8623	0.8541	0.8606	0.8483	0.8517	0.8662	0.8653	0.8700	0.8687
Effusion	0.7874	0.7896	0.7984	0.7991	0.7987	0.7997	0.7982	0.7976	0.8026	0.8052	0.8075	0.8096
Infiltration	0.6725	0.6771	0.6798	0.6773	0.6756	0.6790	0.6775	0.6792	0.6751	0.6773	0.6894	0.6869
Mass	0.7340	0.7374	0.7480	0.7502	0.7567	0.7614	0.7495	0.7538	0.7628	0.7652	0.7756	0.7820
Nodule	0.6778	0.6795	0.6962	0.6934	0.7107	0.7118	0.7032	0.7029	0.7142	0.7188	0.7248	0.7255
Pneumonia	0.6632	0.6744	0.6877	0.6868	0.6861	0.6917	0.6878	0.6858	0.6939	0.7020	0.6943	0.6954
Pneumothorax	0.8149	0.8195	0.8364	0.8405	0.8371	0.8396	0.8356	0.8405	0.8437	0.8497	0.8441	0.8451
Consolidation	0.7088	0.7105	0.7112	0.7136	0.7129	0.7177	0.7124	0.7167	0.7174	0.7190	0.7259	0.7283
Edema	0.8280	0.8311	0.8273	0.8311	0.8294	0.8302	0.8287	0.8280	0.8355	0.8368	0.8363	0.8340
Emphysema	0.7991	0.8059	0.8659	0.8724	0.8704	0.8668	0.8579	0.8590	0.8793	0.8823	0.8707	0.8699
Fibrosis	0.7956	0.7977	0.7939	0.7885	0.7982	0.8036	0.7873	0.7905	0.7972	0.8103	0.7979	0.7978
PT *	0.7356	0.7373	0.7379	0.7424	0.7424	0.7432	0.7465	0.7457	0.7451	0.7510	0.7554	0.7581
Hernia	0.8485	0.8498	0.8879	0.8816	0.8974	0.9113	0.8951	0.8898	0.9097	0.9015	0.8837	0.8776
Average	0.7619	0.7654	0.7762	0.7772	0.7794	0.7827	0.7762	0.7771	0.7847	0.7876	0.7875	0.7877

* PT — Pleural Thickening; AN — AlexNet; VN16 — VGGNet16; RN50 — ResNet50; DN121 — DenseNet121; DN161 — DenseNet161; DN201 — DenseNet201; G-AN — DGC-AlexNet; G-VN16 — DGC-VGGNet16; G-RN50 — DGC-ResNet50; G-DN121 — DGC-DenseNet121; G-DN161 — DGC-DenseNet161; G-DN201 — DGC-DenseNet201.