

Metadata of the chapter that will be visualized in SpringerLink

Book Title	6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6)	
Series Title		
Chapter Title	Human Organ Classifications from Computed Tomography Images Using Deep-Convolutional Neural Network	
Copyright Year	2018	
Copyright HolderName	Springer Nature Singapore Pte Ltd.	
Author	Family Name	Khanh
	Particle	
	Given Name	Ho Thi Kieu
	Prefix	
	Suffix	
	Division	Biomedical Engineering Department
	Organization	International University, Vietnam National University-HCM
	Address	Ho Chi Minh, Vietnam
	Email	
Author	Family Name	Hung
	Particle	
	Given Name	Tran Cong
	Prefix	
	Suffix	
	Division	
	Organization	Posts and Telecommunications Institute of Technology-HCM
	Address	Ho Chi Minh, Vietnam
	Email	
Author	Family Name	Hung
	Particle	
	Given Name	Dang Viet
	Prefix	
	Suffix	
	Division	
	Organization	Duy Tan University
	Address	Da Nang, Vietnam
	Email	
Corresponding Author	Family Name	Thang
	Particle	
	Given Name	Nguyen Duc
	Prefix	
	Suffix	
	Division	Biomedical Engineering Department
	Organization	International University, Vietnam National University-HCM
	Address	Ho Chi Minh, Vietnam
	Email	

Abstract

Deep neural networks (DNNs) have recently indicated outstanding performance on image learning features tasks while Convolutional neural networks (CNNs) have been applied for classification tasks by reducing spectral variations and the spectral correlations of the model which existed in images. In this paper, we independently approached our work as a sequence of steps. We first implemented sparse autoencoders as unsupervised algorithm to obtain learned features in two hidden layers for the DNN model by evaluating the appropriate input features and the optimal number of hidden units, which allow us to validate basic capabilities of the dataset. Secondly, we trained a deep CNN which consisted of five main convolutional layers, followed by Rectified Linear Units (ReLU)s layers, max-pooling layers, three fully-connected layers and a final softmax probability layer, to classify the high-resolution medical images of Computed Tomography (CT) into five anatomical classes, corresponding to five organs in abdominal regions. As a result, we considerably achieved the classification accuracy of $83.74 \pm 3.34\%$ in testing. We also visualized the layer representations on CT datasets, where they indicated the state-of-the-art performance, and could hold much promise to initialize further research on computer-aided diagnosis.

Keywords
(separated by '-')

DNN - Sparse autoencoders - Optimization - CNN - Convolution - ReLU)s - Pooling - Fully-connected - Softmax - CT images



Human Organ Classifications from Computed Tomography Images Using Deep-Convolutional Neural Network

Ho Thi Kieu Khanh, Tran Cong Hung, Dang Viet Hung,
and Nguyen Duc Thang

Abstract

Deep neural networks (DNNs) have recently indicated outstanding performance on image learning features tasks while Convolutional neural networks (CNNs) have been applied for classification tasks by reducing spectral variations and the spectral correlations of the model which existed in images. In this paper, we independently approached our work as a sequence of steps. We first implemented sparse autoencoders as unsupervised algorithm to obtain learned features in two hidden layers for the DNN model by evaluating the appropriate input features and the optimal number of hidden units, which allow us to validate basic capabilities of the dataset. Secondly, we trained a deep CNN which consisted of five main convolutional layers, followed by Rectified Linear Units (ReLUs) layers, max-pooling layers, three fully-connected layers and a final softmax probability layer, to classify the high-resolution medical images of Computed Tomography (CT) into five anatomical classes, corresponding to five organs in abdominal regions. As a result, we considerably achieved the classification accuracy of $83.74 \pm 3.34\%$ in testing. We also visualized the layer representations on CT datasets, where they indicated the state-of-the-art performance, and could hold much promise to initialize further research on computer-aided diagnosis.

Keywords

DNN • Sparse autoencoders • Optimization • CNN • Convolution • ReLUs • Pooling • Fully-connected • Softmax • CT images

1 Introduction

Recently, different research efforts on 2D appearance for deep learning have developed, such as Principle Component Analysis, Support Vector Machine, Clustering and K-Nearest Neighbor classifiers, amongst supervised and

unsupervised learning methods have achieved familiarity. However, learned features are not a part for such features extractions, classification schemes and needs to compute features separately such as edges, gradients and pixels intensities. Therefore, our primary interest was to gain experience in implementing other learning methods.

The bloom of the techniques for representation learning have been witnessed in the past few years, which allow a machine to be fed with raw data and automatically represent the needs for detection and classification in computer science. Partially inspired by neuroscience, DNN and CNN models are closely related to the findings of complex cells in the primary visual cortex [1]. As a large and generally unknown of biological inputs in the brain, anatomical evidences have shown ubiquitous inter-connections between

H.T.K. Khanh • N.D. Thang (✉)
Biomedical Engineering Department, International University,
Vietnam National University-HCM, Ho Chi Minh, Vietnam
e-mail: ndthang@hcmiu.edu.vn

T.C. Hung
Posts and Telecommunications Institute of Technology-HCM,
Ho Chi Minh, Vietnam

D.V. Hung
Duy Tan University, Da Nang, Vietnam

neurons. After being weighted and transformed to other neurons repeatedly by an activation function, the output neuron is activated. This process motivates the neural nets to be capable for learning.

DNNs refers, closely, to computational models that are consisted of multiple processing layers and have the capacity to learn complex features from high-dimensional data than shallow ones [2]. By using back-propagation algorithm, the change of internal parameters is used to compute representation in the future layer from the representation in the previous layer. DNNs have emerged as a powerful learning technique, including object classification [3], speech recognition [4] and empirically demonstrated on the challenging across thousands of classes [5, 6].

Meanwhile, variants CNNs [7, 8] have recently emerged as robust supervised technique, successfully applied in medical image classifications [9, 10] in combination of convolution [5] and pooling algorithm [11]. By organizing units of convolutional layers in feature maps, within each unit is connected to local patches through the set of the weights, this local weights sum is passed through non-linearity. After being detected local con-junctions of features, pooling layers merge semantically similar features into one, thereby reducing dimension of representation and creating small shifts and distortions from invariance.

From those advances, our approaches proposed explored the features learning through unsupervised layers in DNN using sparse autoencoders, and then followed by training deep CNN model. We presented a method for organ-specific classification of medical images acquired using computed tomography (CT) images to learn five anatomical classes.

2 Model Discriptions

2.1 Deep Neural Network (DNN)

Our method was built upon sparse autoencoders to extract features in two hidden layers. Each feature in autoencoders

could connect to a small region of the lower image based on biologically inspired ideas to scale the autoencoders for such a large image dataset.

Sparse Autoencoders

By automatically learn features from unlabeled data, sparse autoencoders are applied to reconstruct the input

$X = \{(x_1, y_1), \dots, (x_j, y_j)\}$ as non-linear approximation—the output $\hat{X} = \{(\hat{x}_1, \hat{y}_1), \dots, (\hat{x}_m, \hat{y}_m)\}$ via a possibly lower dimensional hidden layers (Fig. 1a) that the sum of weight $w^{(1)} = \{w_{11}^{(1)}, w_{12}^{(1)}, \dots, w_{ij}^{(1)}\}$, and bias $b^{(1)} = \{b_1^{(1)}, b_2^{(1)}, \dots, b_i^{(1)}\}$ are weighted to denote the activation units $a_i^{(l)} = f(W_{ij}^{(l)} + b_i^{(l)})$ and the hidden layer node $Z = \{z_1, z_2, \dots, z_i\}$:

$$Z_i = b_i^{(l)} + \sum_{j=1}^m w_{ij}^{(l)} X_j \quad (1)$$

The activation function for three layers in autoencoders:

$$h_{(W,b)}(x) = f(W^T x) = f\left(\sum_{i=1}^3 W_i X_i\right)$$

The initial cost function:

$$J(W, b) = \frac{1}{m} \sum_{i=1}^m \|h_{(W,b)} - y_i\|^2 + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^l)^2 \quad (2)$$

In order to minimize the difference between the output $\hat{X}$ and the corresponding target values which must be equal to the input, we used back-propagation with gradient descent to adjust the weights and biases of the network's connections. After passing the feed-forward algorithm, we computed the output layer by:

$$\delta = \frac{\partial}{\partial Z_i^n} \frac{1}{2} \|y_i - h_{(W,b)}\|^2 = -(y_i - a_i^n) f'(Z_i^n) \quad (3)$$

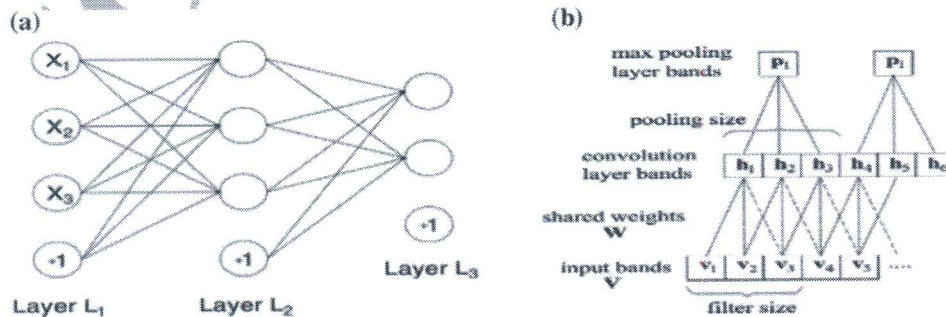


Fig. 1 a The sparse autoencoders during training. b A typical CNN architecture composing of a convolutional and pooling layer

Each node of each layer:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{S_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \quad (4)$$

The desired partial derivatives:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \quad (5a)$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)} \quad (5b)$$

Hence, the overall cost function after applying the gradient descent:

$$J_{sparse}(W, b) = J(W, b) + \beta \sum_{i=1}^{S_l} KL(\rho || \hat{\rho}_i) \quad (6)$$

where ρ is a sparsity parameter and $\hat{\rho}_i = \frac{1}{m} \sum_{i=1}^m [a_i X_i]$. We then enforced the constraint $\hat{\rho}_i = \rho$ to get the Leiber (KL divergence term):

$$KL(\rho || \hat{\rho}_i) = \rho \log \frac{\rho}{\hat{\rho}_i} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_i}$$

The sparse autoencoders have implemented the optimal compressing, the imposing constraints on the number of hidden layer nodes and the sparsity of the hidden layer representations to reduce the input's dimensionality.

Optimization

Since back-propagation is notoriously difficult to debug that cause the problem of buggy implementation, a method to numerically check and carry out the derivatives computed will significantly increase the correctness of our algorithm by minimizing $J(\theta)$ [e.g. each iteration $\Theta := \Theta - \alpha \frac{d}{d\Theta} J(\Theta)$] in which

$$\frac{d}{d\Theta} J(\Theta) \approx \frac{J(\Theta + \epsilon) - J(\Theta - \epsilon)}{2\epsilon} \text{ where } \epsilon = 10^{-4}$$

Then,

$$\nabla_{W^{(l)}} J(W, b) = \left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \quad (7a)$$

$$\nabla_{b^{(l)}} J(W, b) = \frac{1}{m} \Delta b^{(l)} \quad (7b)$$

2.2 Convolutional Neural Network (CNN)

Compared to fully-connected network like DNNs which might restrict the connections between hidden units and input units, the convolutional layer computes the hidden activation which allow us to use features at all location since the property in CT images are stationary. Figure 1b illustrated a basic CNN model. The weights W_i are shared across all convolutional layer band. Then, after getting convolved feature, we aggregated statistics of a million unwieldy and prone over-fitting features at various locations by using max pooling, which was given by the activation over non-overlapping region to prominently select invariant features, lead to faster convergence rate and imply next for better classification tasks. The max pooling algorithm was applied by the function:

$$a_j = \max_{N \times N} a_i^{n \times n}(u(n, n)) \quad (8)$$

In our CNN model, beside the traditional convolution and pooling, essential layers were added. Firstly, Rectified Linear Units (ReLUs) [8] acted as a rectifier which was defined an activation function of $f(x) = \max(0; x)$ to set the value of neuron input x to zero if it was negative and did nothing otherwise and hence left the dimensions unchanged for faster learning, compared with conventional functions of $f(x) = \tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$. These layers were added, following convolutional layers.

Furthermore, fully connected layer, known as a special case of convolutional layer, connected all activations of previous layers to the output for not losing information from all neurons learned. These layers were used in the latter part of our network to store high level abstraction information

In order to reduce overfitting, we used dropout process [12] to regularize the network. With a certain probability during training, we dropped connections between different layers that made the network become less reliant and more diverse with shared weights in parallel and sampling them at test time.

As a final layer, a softmax classifier was used to normalize and interpret the output which was defined as a function

$$l(x, c) = -\log \frac{e^{x_c}}{\sum_{k=1}^C e^{x_k}} \quad (9)$$

where x is the output vector with probabilities, c is categorical class and C is the total of number classes.

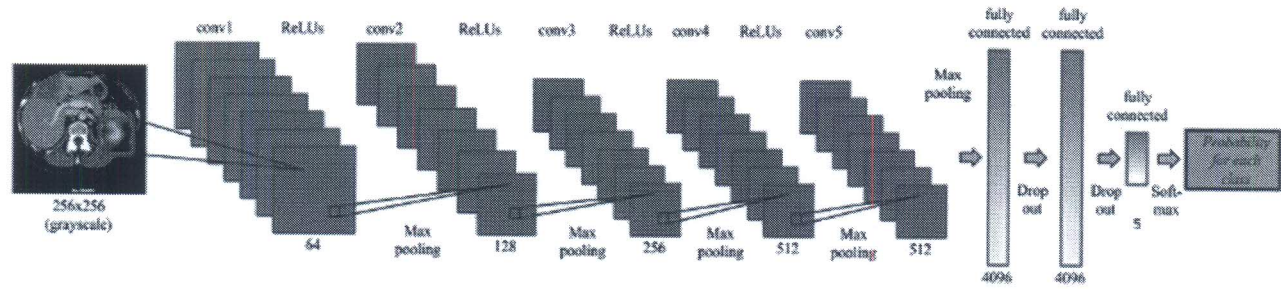


Fig. 2 CNN architecture applied to CT images. The number of convolutional filters was show

In summary, we investigated the efficient effect of our CNN, illustrated by Fig. 2, by building five convolutional layers with a small convolutional filter of 3×3 in which every kernel element was trained, following ReLUs layers, max-pooling layers of 2×2 and a 1×1 convolutional filter in the fully connected layers during convolutional periods, and a final softmax layer for organ classifications.

3 Results

3.1 Deep Neural Network (CNN)

As training data for DNN, we used the set of CT images over ten patients, associated with 128 slices of each. We cropped the renders to remove unnecessary parts and resized to a common size of 450×450 pixels, cut the window of 50×50 pixels for creating randomly of 10,000 of grab-samples, a few of these patches was shown on Fig. 3.

The initial network parameters were trained randomly, using gradient descent to optimize the weight with a fixed learning rate of 0.05.

Figure 4a, b, c displayed the learned features in layer L_2 by a regular under-complete autoencoders that used 50 hidden units, feature learnt by a complete autoencoders of

100 hidden units, as well as those learnt by over-complete autoencoders using 200 hidden units.

We evaluated how the proposed training strategy of our network behaves as we increased the model's capacity both in breadth represented for the number of neurons per layer and in depth associated with the number of hidden layers. Obviously, the advantageous point appeared to ascend the number of hidden units over two hidden layers. The features obtained from layer L_1 of under-complete were likely not apparent compared to the features learnt in the complete and over-complete cases and seem not formulating Gabor filters [13]. The complex weights corresponding to the number of hidden units showed on Fig. 5a, b, c stated that the more hidden units, the more features have been extracted from weights in layer L_2 and represented in layer L_3 .

3.2 Convolutional Neural Network (CNN)

We compiled a database of CT images by preprocessing to convert the original DICOM images into grayscale images, remove unnecessary parts, annotate and label these images into five classes: 3. Liver, 4. Bone, 2. Left kidney, 1. Venous system, and 5. Spleen (Fig. 6). Then, we divided a total dataset into 90% (8719 examples) for training and 10% (1281 samples) for validating and testing purposes.

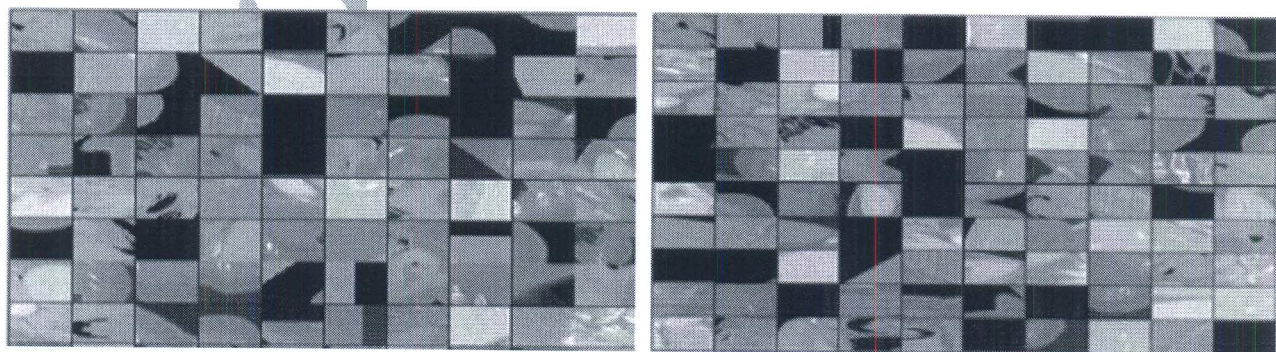


Fig. 3 Random grab samples cut from the data set of CT images

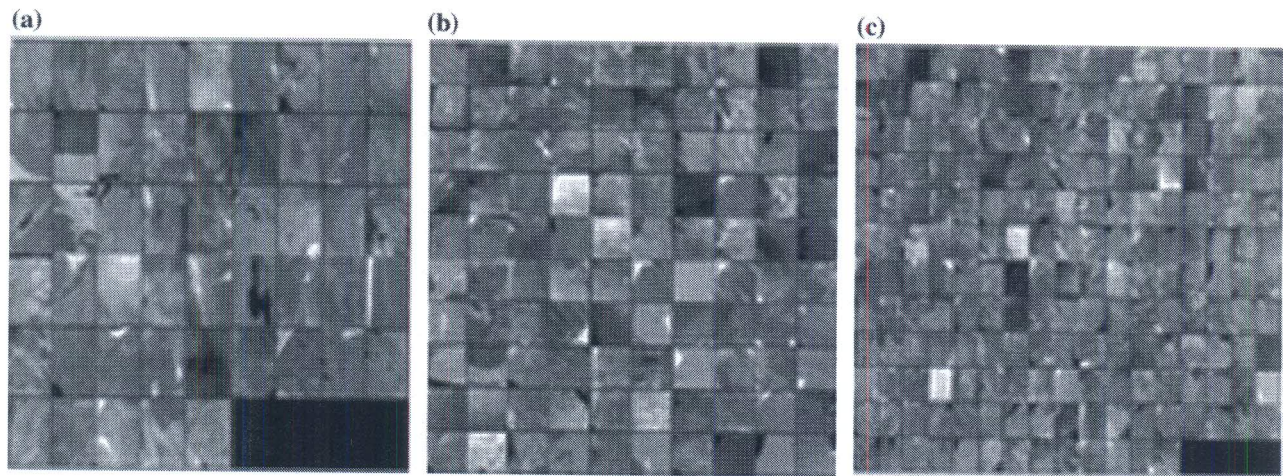


Fig. 4 Learned features in hidden layer L_2 for different levels of sparsities **a** 50, **b** 100 and **c** 200 hidden units

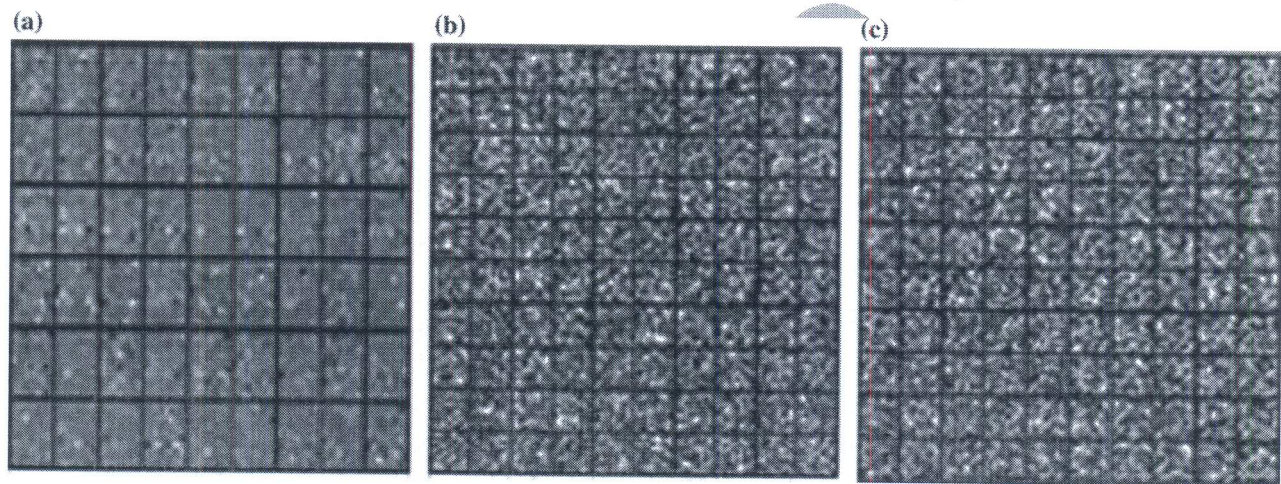
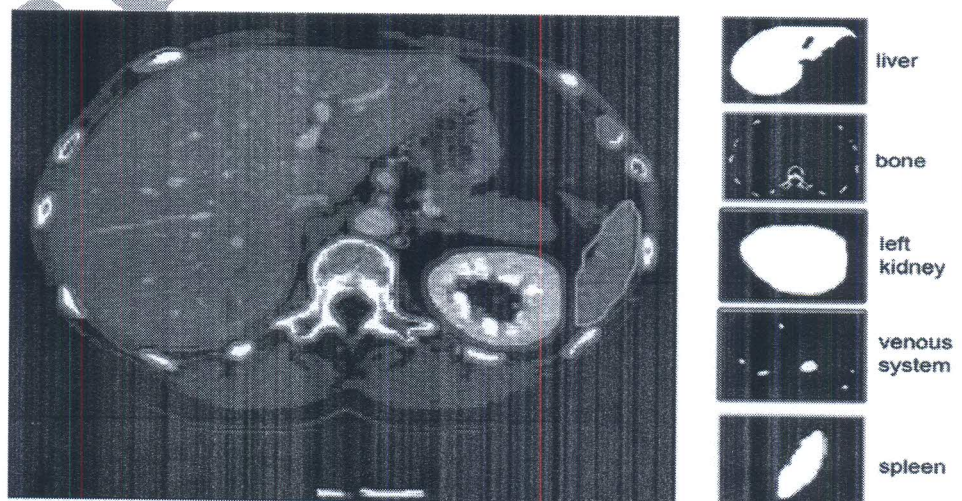


Fig. 5 Learned features in hidden layer L_3 for different levels of sparsities **a** 50, **b** 100 and **c** 200 hidden units

Fig. 6 A CT image example of five labeled anatomical organs



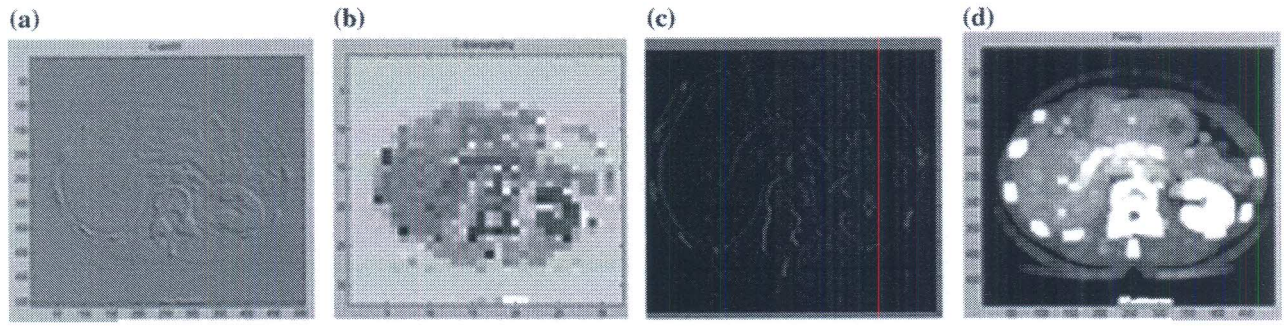


Fig. 7 The first check on CT images with CNN architecture: **a** convolution; **b** down sampling; **c** ReLUs; **d** pooling

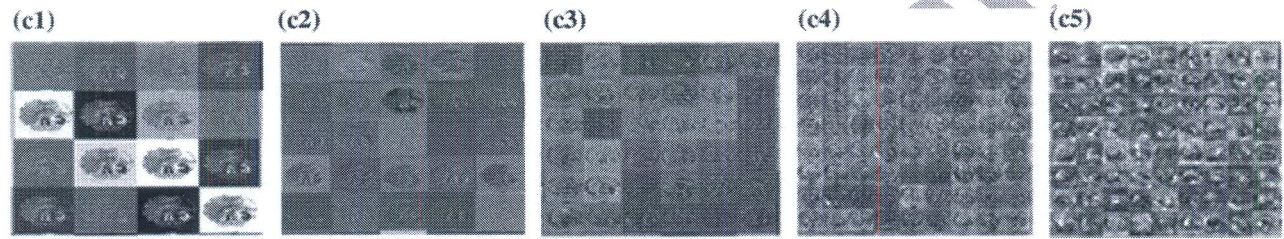


Fig. 8 Feature learning of CT images from convolutional layer No. 1 to convolutional layer No. 5

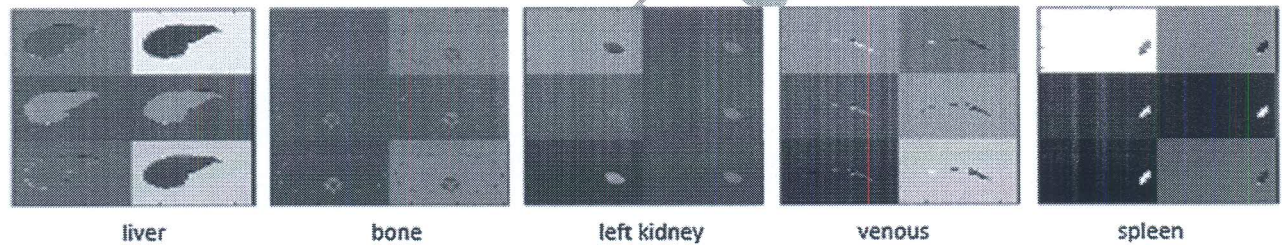


Fig. 9 Some examples of five organs segmentation from dominated features learned in the CNN model

To precisely evaluate our algorithms, we checked and visualized the first few layer representations in the whole CT images, shown on Fig. 7. Furthermore, necessary features were learnt and extracted throughout layer-by-layer, represented convolutional layers by Fig. 8.

After passing through the deep CNN network, we were able to automatically specify in what region object could be found by semantic segmentation [14]. For each pixel in every object, we checked in where if the objects were lied in and features were dominated (illustrated on Fig. 9), for instance left, middle or right parts on the x-coordinates and top, center, or bottom parts on the y-coordinates. Then, we extracted the median value of the x- and y-coordinates and specified the object's position. Once the object was segmented as the output by its localization, we considerably evaluated the probability of Jaccard index accuracy with the output M_1 and the target label M_2 , given by:

$$(M_1, M_2) = \frac{|M_1 \cap M_2|}{|M_1 \cup M_2|} \quad (10)$$

Figure 10 indicated that while the training error (blue curve), which showed the error running the CNN model with

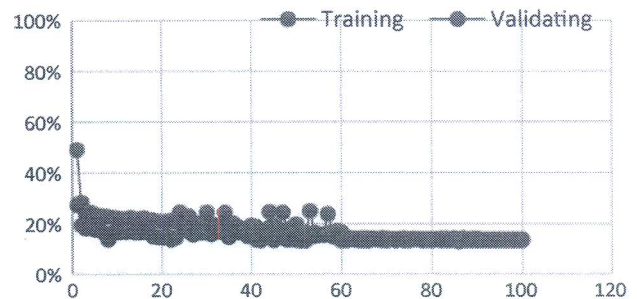


Fig. 10 Error of misclassification during 100 training and validating epochs

Table 1 The accuracy result of five anatomical organ classifications

Organs	Testing accuracy (%)
Venous system	78.09
Left kidney	83.61
Liver	84.72
Bone	85.97
Spleen	86.33
Average	83.74 ± 3.34

the same images, was gradually decreasing, the validating error (red curve) on the other hand tested the CNN with weights not been effected, also was reduced. This meant the trained classes had high probability of being correct, compared with the target labels, which could be possible to distinguish different classes.

As shown in Table 1, it should be noted that despite of the spatial complexity and variability of medical images, the deep CNN classifier was acceptably able to identify the distinguished organs of the abdominal regions in CT images, based on feature recognitions.

4 Conclusions

In this work, we have manifested the expressivity of primary DNN model for learning features by using sparse autoencoders. The multiple layers as deep auoencoders might improve the presentation on the dataset of CT images by creating a hierarchy of both the neuron units' numbers per layer and the number of hidden layers for feature extractions. Meanwhile, deep stages represented for CNN model composed of five convolutional layers, combining with ReLUs layers, max pooling layers, fully connected layers and a soft classifier have been explored to reduce the features' dimensionality, improve generalized performances and classification tasks of five anatomy-specific organs in CT images. As a future work, we aim to explore the use of stacked autoencoders for multiple-layers in DNN model in which each layer captures high-order correlations of the activation units and present progressively more complex nonlinear structure of the input. Additionally, the exploration of deep CNN model used for automatically detection abnormal organs as aided diagnosis is an indispensable topic to address.

Acknowledgements This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2016.18.

References

- Hubel DH (1962) Wiesel TN Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol* 160(1):106
- Bengio Yoshua (2009) Learning deep architectures for AI. *Found Trends R Mach Learn* 2(1):1–127
- Girshick R, Donahue J, Darrell T, Malik J (2013) Rich feature hierarchies for accurate object detection and semantic segmentation (arXiv preprint arXiv: 1311.2524)
- Dahl GE, Yu D, Deng L, Acero A (2012) Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans Audio Speech Lang Process* 20(1):30–42
- Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: *Computer vision and pattern recognition*
- Le QV, Marc'Aurelio Ranzato RM, Matthieu Devin KC, Greg SC, Dean J, Ng AY (2012) Building high-level features using large scale unsupervised learning. In: *International Conference on Machine Learning*
- LeCun Y, Kavukcuoglu K, Farabet C (2010) Convolutional networks and applications in vision. In: *Proc. of the IEEE Intl. Symp. on Circuits and Systems*, pp 253–226. Jun. 2010
- Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. In: *NIPS*. pp 1097–1105
- Roth H, Lu L, Seff A, Cherry K, JHoffman J, Wang S, Liu J, Turkbey E, Summers R (2014) A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations. In: *MICCAI*. Springer, pp 520–527
- Roth H, Yao J, Lu L, Stieger J, Burns J, Summers R (2014) Detection of sclerotic spine metastases via random aggregation of deep convolutional neural network classifications. In: *MICCAI spine imaging workshop* (arXiv preprint arXiv:1407.5976)
- Scherer D, Muller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: *Proceeding of the international conference on artificial neural networks*. pp 92–101
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
- Fidler S, Leonardis A (2007) Towards scalable representations of object categories: learning a hierarchy of parts. In: *Computer vision and pattern recognition*
- Noh H, Hong S, Han B (2015) Learning deconvolution network for semantic segmentation. In: *Proceedings of the IEEE international conference on computer vision*

Author Query Form

Book ID : 418946_1_En

Chapter No : 158



Springer

the language of science

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your chapter, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query Refs.	Details Required	Author's Response
AQ1	In order to maintain the sequential order the figures and their respective citations are renumbered in this chapter. Kindly check and amend if necessary.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

<i>Instruction to printer</i>	<i>Textual mark</i>	<i>Marginal mark</i>
Leave unchanged	... under matter to remain	Ⓟ
Insert in text the matter indicated in the margin	Ⓟ	New matter followed by Ⓟ or Ⓟ ²
Delete	/ through single character, rule or underline or through all characters to be deleted	Ⓟ or Ⓟ ²
Substitute character or substitute part of one or more word(s)	/ through letter or through characters	new character / or new characters /
Change to italics	— under matter to be changed	Ⓟ
Change to capitals	≡ under matter to be changed	≡
Change to small capitals	≡ under matter to be changed	≡
Change to bold type	~ under matter to be changed	~
Change to bold italic	~ under matter to be changed	~
Change to lower case	Encircle matter to be changed	Ⓟ
Change italic to upright type	(As above)	Ⓟ
Change bold to non-bold type	(As above)	Ⓟ
Insert 'superior' character	/ through character or Ⓟ where required	Y or Y under character e.g. Y or Y
Insert 'inferior' character	(As above)	Ⓟ over character e.g. Ⓟ
Insert full stop	(As above)	Ⓟ
Insert comma	(As above)	,
Insert single quotation marks	(As above)	Y or Y and/or Y or Y
Insert double quotation marks	(As above)	Y or Y and/or Y or Y
Insert hyphen	(As above)	Ⓟ
Start new paragraph	Ⓟ	Ⓟ
No new paragraph	Ⓟ	Ⓟ
Transpose	Ⓟ	Ⓟ
Close up	linking Ⓟ characters	Ⓟ
Insert or substitute space between characters or words	/ through character or Ⓟ where required	Y
Reduce space between characters or words	 between characters or words affected	Ⓟ