Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/5/20230591

Lung CT recognition based on convolutional neural network

Ruilin Tian

Tiangong University, school of Economics and Management, Tianjin, China, 300387

ruilinstu@163.com

Abstract. Lung medical image recognition plays an integral role in today's clinical diagnosis. This process involves finding valid recognition parts in lung CT images and classifying them as normal and abnormal. although many breakthroughs have been made in image classification and lung CT image recognition, current models are mostly for 2D lung images. This paper uses convolutional neural networks (CNN) to extract features for image recognition of infected and uninfected pneumonia virus. Based on the characteristics of the dataset, this paper introduces dynamic learning rate and defines an early stopping strategy to compensate for the traditional manual parameter adjustment methods which are time-consuming and consume a lot of computing power. By comparing with other methods, the model proposed in this paper is able to produce very competitive results in a shorter time, which is exciting.

Keywords: CT identification, machine learning, CNN, dynamic learning rate, early stopping strategy

1. Introduction

Diagnostic imaging is an indispensable tool for today's medical community, helping physicians to visualize organ or tissue lesions. A typical example is computer tomography (CT) [1], which can be used to image a patient's tissues in a tomographic fashion to aid in clinical diagnosis. In the period of neocoronary pneumonia, a large number of patients emerge and doctors need to handle a large number of patients' CT every day, which cannot be satisfied by manual judgment alone. Deep learning [2] is widely used because it can learn the features of images autonomously, especially the convolution and pooling operations in convolutional neural networks [3] can extract and map the features of original images with translation invariance.

In this paper, we construct a 3D convolutional neural network (3DCNN) [4] to predict the presence of viral pneumonia on computed tomography (CT) scans and to provide a preliminary diagnosis of pneumonia to doctors [5]. The model construction process includes data pre-processing, data integration, data augmentation and the construction of a 3DCNN model. Once the model is built, its performance is also visualized and the final application is able to give a judgement on whether the input CT images are normal or not. The algorithm involves the deep learning databases Keras and Tensorflow, and its core data structure is a sequential architecture that builds the model through a series of sequentially stacked network layers.

DOI: 10.54254/2755-2721/5/20230591

2. Related works

2.1. Convolutional neural networks

The concept of convolutional neural networks dates back to 1962, when Hubel and Wiesel [6] introduced the concept of receptive fields in their study of the visual system in the cat brain and used it for information processing in the visual system. This discovery laid the foundation for the development of convolutional neural networks. With further research, in 1980, Japanese scientist Kunihiko Fukushima [7] proposed a neural network structure containing convolutional and pooling layers in his paper. Based on this, Yann Lecun [8] added the BP algorithm to train the neural network structure and proposed LeNet-5, forming the prototype of contemporary convolutional neural networks. It was not until 2012 that the Imagenet image recognition competition turned the field of image recognition upside down with the introduction of a new deep structure and dropout method by Alexnet, mentioned in a paper by Hinton et al. [9], which instantly raised the error rate from over 25% to 15%. In recent years, CNNs have made great breakthroughs in the fields of image recognition, video analysis, sound recognition, natural language processing, and have even been applied to the game of Go. Among the many applications, the impact of CNNs on the medical field cannot be ignored, as it has contributed to cancer recognition and tumour identification.

2.2. Main work

Unlike the other datasets, the images in this dataset are stored in Nifti format, so the nibabel package to read such files prior to performing image recognition is needed to introduce. In addition, the images in this dataset are all grey-scale images of size 128 x 128 x 64, and the dataset is divided into two subsets, comprising normal lung images and non-normal lung images respectively.

This paper first uses the nibabel package to read the images contained in the dataset, which contains a total of two class directories: normal and abnormal. After data pre-processing of all images, the dataset is split into a training set and a validation set in a 7:3 ratio. Before building the model, a data enhancement operation is performed: during training, the data is enhanced by having the CT images rotated at random angles. As the data are stored in Rank-3 shapes (sample, height, width, depth), we add a size of 1 at axis 4 to be able to perform a 3D convolution of the data, resulting in a new shape of (sample, height, width, depth, 1). A preliminary model was then obtained by learning the features of each lung CT image and its corresponding label. During the model training process, the dynamic learning rate [10] is set and an early stopping strategy [11] is defined to enable the model to achieve better convergence in less time.

3. Model construction

3.1. Keras &tensorflow

Keas is a high-level Python based application programming interface (API) for neural networks written with Tensorflow as the backend.

Layers are the basic building blocks of neural networks in Keras. A layer is made up of a tensor - an input tensor - an output computation function and some state stored in Tensorflow variables. The model applied in this paper is the Keras sequence model, which is a linear stacking of multiple network layers in an architecture formed by the crossover of Keras and Tensorflow. The model is characterized by the need to know the size of its expected inputs, which means that the first layer of the model needs to receive information about the size of its inputs. This paper meets this requirement by passing parameters to the first layer using a tuple "keras.Input" representing the size.

3.2. API

API, known as Application Programming Interface [12], is a number of predefined functions. Its purpose is to give applications and developers the ability to access a set of routines based on a piece of software or hardware. In this process, the application or developer does not need to access the source

DOI: 10.54254/2755-2721/5/20230591

code or understand the details of the internal working mechanism. The application of APIs can help to couple different programs and greatly improve the efficiency. This paper uses the layer API directly as a convolution layer. The convolution layer is shown in the Figure 1.

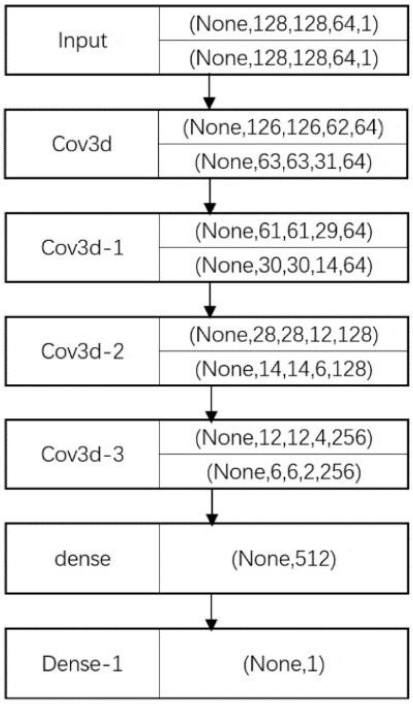

Figure 1. Layer Structure Diagram.

3.3. Dropout

Dropout [13] works by making half of the hidden node values in the training batch zero in machine learning, which is more effective in reducing the occurrence of overfitting and, to a certain extent, regularization. In addition to this, Dropout can improve the generalization and robustness of the model. This is because when forward propagation is performed, Dropout stops the activation value of a neuron with probability p, which also ensures that the model does not rely too much on local features. The implementation of Dropout in Keras in this paper is to block out certain neurons so that after their activation values are 0, the vector of activation values is multiplied by 1/(1 - p).

3.4. Batch normalization

The role of BN is to pull the layer eigenvalue distribution back to the standard normal distribution, that is, to make the mean of the output data close to 0 and the standard deviation close to 1. It can avoid gradient disappearance and also accelerate convergence and enhance the non-linear learning ability of the model.

3.5. Common mistakes

In the Kera model, the error in the training set is much higher than the error in the validation set, and some of the regularization mechanisms will not be activated when running with the validation set.

For mismatch of development environments, Keras & Tensorflow are very environment-dependent deep learning databases, so the frequent errors reported during dataset calls are likely to be related to the current python environment. The authors recommend building models and debugging algorithms using Anaconda.

As for the problem of mismatching the size of the input vector and the hidden vector, a similar problem occurs when the size of the dataset is not specified and no pre-processing is performed.

DOI: 10.54254/2755-2721/5/20230591

4. Model runs and optimization

4.1. Data pre-processing

Compared to ordinary image datasets, medical image CT differs in two ways: firstly, the data files are in Nifti format with a .nii extension, and this paper uses the nibabel package to read such files; secondly, the pixels of ordinary images consist of the three primary colors of RGB, while the pixels of medical image CT are 16bit grey and have a much higher precision, usually expressed as HU values. To address these two characteristics, we first rotate the volume of the image by 90 degrees to ensure that the orientation is fixed, and then perform a normalization operation to scale the HU value to between 0 and 1. Afterwards, the width, height and depth of the image are adjusted so that all images have a uniform format. Finally, the label is digitized, with CT images of the "normal" type being given the label "0" and those of the "abnormal" type being given the label "1".

4.2. Construction of training and validation sets

The lung CT images were read from the class catalogue to assign labels, and the images were downsampled to have a format of 128 × 128 × 64. The original HU values were rescaled to 0~1 and split into training and validation sets in a 7:3 ratio.

4.3. Set dynamic learning rates

Deep network models are trained on a complex function with unknown and large parameter magnitude. The aim of the training is to find an optimal set of parameters so that the network corresponds to the minimum value of the function. Currently the adjustment of parameters in deep learning is basically based on the principle of gradient descent, i.e. input a Batch, calculate the difference between the model output and the real distribution for the gradient of the parameters, and update the parameters based on the current gradient. The general learning rate can be expressed by the formula (1):

$$LearnRate = \frac{LearnRate}{1+decay*epoch} \tag{1}$$

When the learning rate is fixed, there is a risk that the gradient may fall too slowly or oscillate around the lowest point, and the dynamic learning rate can be used to solve this problem and improve the efficiency of model training. The dynamic learning rate function used in this paper is schedules. ExponentialDecay, which is given by:

$$\alpha = \alpha_0 e^{-kt} \tag{2}$$

In this paper, the value of initial_learning_rate is set to $1e^{-4}$, decay_steps is set to 30, decay_rate is set to 0.96 and staircase type is True.

4.4. Compilation model

The loss function used to compile the model is binary_crossentropy, which uses accuracy to reduce the loss. Sigmoid is used as the activation function, and Adam is used as the optimizer. After compiling, the model is saved.

4.5. Defining an early stop strategy

During the training of the model, an optimal model will be obtained at some point, but if the preset number of iterations has not been completed at this point, we will end up missing this optimal model. The purpose of setting the early stop strategy is to improve the tuning efficiency, and after using it, the model will directly use the optimal model obtained during the training process, otherwise there is a risk of getting an over-fitted model.This study sets the program to run for 100 epoch and set the patience value to 15, i.e. if the model does not improve within 15 rounds we exit the program. According to the experimental results, the program stopped running at the 46th epoch of the run, which means that the model we want has been obtained.

DOI: 10.54254/2755-2721/5/20230591

4.6. Save model

By using the “callbacks.ModelCheckpoint” function and setting save_best_only to "True", the best model automatically obtained from the early stop strategy is saved and this is the best model we end up with.

4.7. Model performance visualization

According to Figure 2, when the program is run to the 46th epoch to get the optimal model, the model achieves the highest accuracy on the training set and less than 70% accuracy on the validation set, but the model gets the minimum value of loss both on the training and validation sets. According to the principle of the early stop strategy, this model is the optimal model we want.

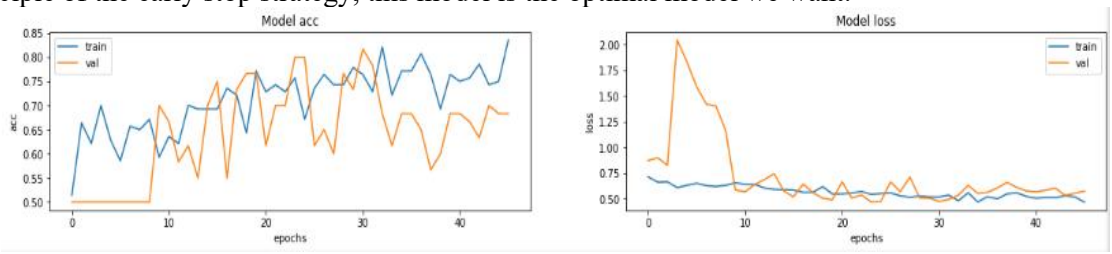

Figure 2. Model performance visualization diagram.

5. Conclusion

Substituting a CT image of a lung from the dataset into the model gives the following output "This model is 27.88% confident that the CT scan is normal. This model is 72.12% confident that CT scan is abnormal". From this output, it can be initially determined that the patient is more likely to have contracted the virus and have pneumonia.

The dataset used in this paper contains only 200 3D lung CT images, a relatively small amount of data, and no random seeds were specified, which may differ somewhat from the expected results. Subsequent studies can look at these two aspects and aim to obtain a better 3DCNN model.

Acknowledgement

I would like to extend my heartfelt thanks to a host of people, without whose assistance the accomplishment of this thesis would have been impossible.

First and foremost, I am deeply indebted to my supervisor Professor Lin, for his patient guidance and continuous encouragement during the writing of this thesis as well as in the course of my undergraduate studies.

I am also grateful to Professor O. Y., whose valuable instruction has benefited me a great deal.

I owe a lot to my parents as well for their continuous support and encouragement.

References

[1] Xie H., Ma S., Wang X., et al. Noncontrast computer tomography–based radiomics model for predicting intracerebral hemorrhage expansion: preliminary findings and comparison with conventional radiological model [J]. European radiology, 2020, 30(1): 87-98.

[2] LeCun Y., Bengio Y., Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436-444.

[3] Teuwen J, Moriakov N. Convolutional neural networks [M] Handbook of medical image computing and computer assisted intervention. Academic Press, 2020: 481-501.

[4] Kopuklu O., Kose N., Gunduz A., et al. Resource efficient 3d convolutional neural networks [C] Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019: 0-0.

[5] Ji S., Xu W., Yang M, et al. 3D convolutional neural networks for human action recognition [J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 35(1): 221-231.

[6] Hubel D. H., Wiesel T. N. Receptive fields, binocular interaction and functional architecture in

DOI: 10.54254/2755-2721/5/20230591

the cat's visual cortex [J]. The Journal of physiology, 1962, 160(1): 106.
[7] Fukushima K., Miyake S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition [M] Competition and cooperation in neural nets. Springer, Berlin, Heidelberg, 1982, pp. 267-285.
[8] LeCun Y., Bottou L., Bengio Y., et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
[9] Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks [J]. Communications of the ACM, 2017, 60(6): 84-90.
[10] Vasudevan S. Dynamic learning rate using Mutual Information [J]. arXiv preprint arXiv:1805.07249, 2018.
[11] Loughrey J., Cunningham P. Using early-stopping to avoid overfitting in wrapper-based feature selection employing stochastic search [R]. Trinity College Dublin, Department of Computer Science, 2005.
[12] Ofoeda J., Boateng R., Effah J. Application programming interface (API) research: A review of the past to inform the future [J]. International Journal of Enterprise Information Systems (IJEIS), 2019, 15(3): 76-95.
[13] Garbin C., Zhu X., Marques O. Dropout vs. batch normalization: an empirical study of their impact to deep learning [J]. Multimedia Tools and Applications, 2020, 79(19): 12777-12815.