Transfer learning based classification of biomedical images using VGG16 architecture

Ch. Pavan Sathish[1], D. Lalitha Bhaskari[2]
[1]Research Scholar, [2]Professor
[12]Department of CS&SE, Andhra University College of Engineering (Autonomous), Andhra University, Visakhapatnam, Andhra Pradesh, India.

Abstract- In recent years, the technical advances and improvements in the field of biomedical imaging techniques has provoked increasing interest to resolve the previously intractable problems. From the standpoint of medical arena, the task of classifying various biomedical MRI scans under their appropriate categories is quiet tedious owing to the distinct characteristics of these images. That too, with the large-scale influx of medical reports, there is a wide necessity for medical data to be automated so that it can be easily organized and accessed in the databases of hospitals and clinics. This mode of execution is conducive and speedy for the primary diagnosis of the patients while saving plentiful of time in categorizing them. Lately, with the evolution of deeper networks, a derivative from the notion of artificial neural networks known as deep learning has emerged. VGG16 model which adopts the perception of deep learning has performed fairly well to fulfill the objective of this paper.

In this paper, 3 different datasets of MRI modality of 3 various interior body structures namely knee joint, lumbar spine and cardiac are considered. The motive is to classify these images into 3 separate classes using VGG16 model. This model is applied on the medical dataset comprising of 328 MRI scans and subsequently followed by data augmentation to increase the size of the data. The model is implemented in keras as front-end library with TensorFlow framework. The training on this dataset to generate a custom model file on GTX 1070 video card consumed 2 hours yielding 99.62%, 98.48%, 96.6% training, validation and testing accuracies respectively.

Key words- Deep learning, Image classification, artificial neural networks, biomedical image classification, keras, tensorflow and VGG16

I. INTRODUCTION

A. Motivation

Healthcare sector is of higher priority when compared to other industries and is entirely different from them. Irrespective of price, people seek for highest standard of services and care. In most of the cases, medical experts interpret the medical images and there exists wide variations across various interpreters owing to the complexity and subjectivity of those images.

In the field of computer vision, extracting and classifying the features of the images is a significant and essential step. This is accomplished by traditional machine learning (ML) techniques and other shallower neural network models until few years ago. In recent times,

Deep learning (DL) [1], a subfield of ML makes use of convolutional neural networks (CNNs) [2] which are firmly established as a powerful method for image recognition. The fundamental building block of DL is CNNs which discovers and learns the latent features of the images itself.The baseline architecture followed by any deep neural networks (DNNs) including VGG16 [3] is CNNs. CNNs make use of local receptive field, sharing of the weights and max pooling techniques. It will insure to drastically reduce the parameters needed for training when compared to the shallower neural network models. The victory of DL in other real world data sets has encouraged for automatic classification of medical images which is otherwise not an easy task.

B. Problem Definition

Many recent research works on medical images have concentrated on AlexNet[4] and Inception v3, a version of GoogleNet [5]. In medical data classification, when there is a scarcity of labeled data, VGG [3] has become popular. So opting for pre-trained networks is always a best choice to train a DNN [6, 7, and 8].

The conventional toilsome process experienced by manual labor to segregate the scan reports can be overcome by enforcing these DNN models. The work in this paper is focused on classifying medical images of 3 categories using the pre-trained VGG16 model on TensorFlow [9] and has resulted in exceptional accuracy. TensorFlow is the Google's second generation artificial learning system that has earned lots of popularity world-wide and has stood top so far in all the ML and DL programs.

C. Terminology

a. Transfer Learning

In the absence of ample labeled data for training a deeper network, one can draw attention towards employing the weights of a pre-trained network. This ML technique known as 'transfer learning' (TL) [10] is based on reusability concept that plays a vital role in the area of medical imaging. Despite of training the network from scratch on the new dataset, the weights that are already obtained by training on the globally

admissible datasets can be utilized. Customarily, these datasets are massive in size that would constitute millions of non-medical images. Contrarily, the medical image datasets are sparse, with only few thousands of images in number. In comparison to the traditional artificial neural networks, TL serves well even with limited training data and benefits the model with higher accuracy. This is usually followed by fine-tuning step for still better accuracy.
Apart from TL, other techniques like multi-stream and multi-view architectures, augmentation of sparse datasets to dense are also prevalent in medical images when compared to the non-medical images.
The rest of the paper is organized as follows. Section 2 contains the related works. Section 3 describes the classification architecture and the steps for classification. Section 4 depicts the experimentation results. Section 5 elucidates the conclusions and discussions.

II. RELATED WORKS

The proficiency of TL [10, 11] persists in complying with the prominent image classification model to a custom task at hand. Segregation of the biomedical images produced at medical labs into their appropriate categories assists the radiologists, doctors, and other clinical and non-clinical staff. Although TL may not be as competent enough as training the network completely from scratch, but is amazinglyartful for lots of applications. It permits for the creation of the model with considerably less training time and data by adapting itself to the subsisting rich DL neural network models.
Alex Krizhevsky [1] has built a CNN model which has produced prominent results on larger datasets. NitishSrivastava, Hinton [12] [13] has turned up with a method called as 'dropout' to avoid overfitting which is caused by smaller and insufficient samples.Oquab et al. [14] has used the transfer learning concept on cross-domain dataset by first pre-training the ImageNet dataset [15] on the CNN training. Jason Yosinski et al. [16] in his experiments has enhanced the performance of DNNs by transferring the weights between cross- domains and later fine –tuned on the target dataset.
In this paper, TL approach is imposed on well trained VGG16 model. It is pre-trained on nearly 1000 object classes comprising of 1.28 million images from ILSVRC [17], 2014.

III. VGG CLASSIFICATION ARCHITECTURE

Oxford's VGG model [3] is characterized by conceptually simple and flexible deep convolutional network. It has been approved that network depth plays a vital role as the better results can be obtained by deeper networks. VGGNet has rendered admirably in the ILSVRC [17] organized in the year 2014 by securing first position on the task of image localization and securing second position on the task of image classification on ImageNet dataset [15]. The model accepts the input image of size 224 * 224 * 3 with 3*3 size convolutional layers that are stacked upon one another in the increasing order of depth. It also uses 2*2 size max pooling layers for minimizing the size of the volume. Two fully connected (FC) layers each consisting of 4,096 nodes followed by a softmax classifier [18] is appended at the end.
In this work, the weight vector of Imagenet dataset is used for the classification of biomedical images by means of TL [10, 11] technique. So, the last dense layer FC-1000 is reset from the basic VGG16 model. Rather, an FC-1024 dense layer with ReLU activation function [19] followed by one dropout layer with rate 0.5 is replaced. A smaller learning rate is enforced on the pretrained layers of the model wherein the features can be adapted to the target dataset. By doing so, the goal is to adapt the features to the new dataset. This is finally followed by FC-3 dense layer with softmax activation at the end. Figure 1 illustrates the architecture of VGG16 where the cardiac image is given as an input and as a resultant the final fc 3 layer categorizes it as cardiac image. The binary representation of cardiac, knee joint and lumbar spine is [0,1,0], [0,0,1] and [1,0,0] respectively.

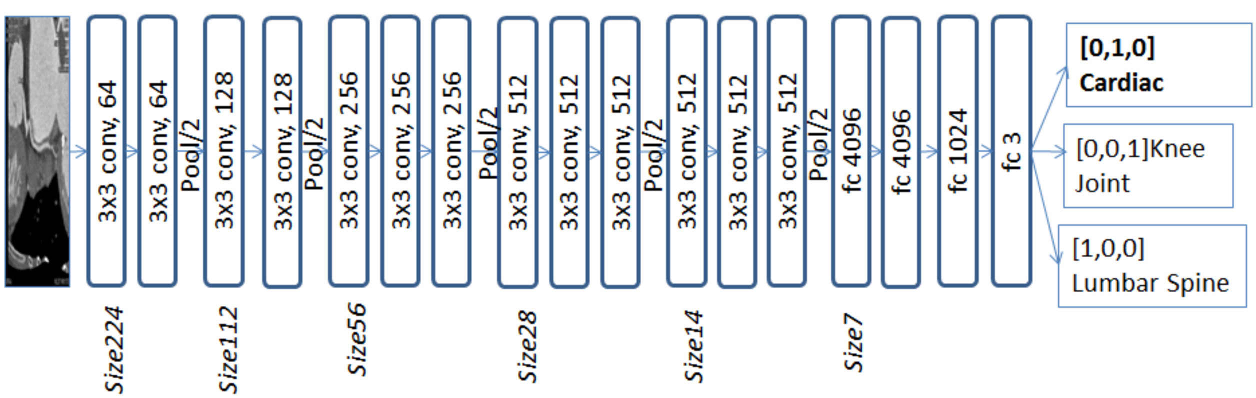


Fig.1: VGG16 architecture for classification of biomedical images of 3 categories

IV. EXPERIMENTATION AND RESULTS

For the purpose of experimentation, the medical image database of 262 samples altogether from MRI cardiac, MRI knee joint, MRI Lumbar datasets was collected from various online repositories. The biomedical image classification is instigated on the VGG16 model [3] with tensorflow [14] platform. The training on this biomedical imagery dataset to create a custom model file on GTX 1070 video card has expended 2 hours.

The following table 1 illustrates the list of parameters employed in this training. These values are static for all the 3 trials. Table 2 depicts the training, validation and testing accuracies where it is perceived that trial 2 has yielded the best results. Here, the network is trained with a batch size of 16 for 15 epochs which has consumed around two hours of time on the specified hardware.

Table 1: Illustration of various parameters which is fixed for training the network

Parameter	**Value**
Transfer model	VGG16
Transfer weights	ImageNet
Optimizer	stochastic gradient descent
Loss	categorical cross entropy
Momentum	0.9
Batch size	16
Number of dense layers	2 (1024, 3)
Number of dropout layers	1 (rate=0.5)

Table 2: Depicts the parameters vs. trials for the medical image dataset

Parameter/Trial	**Trial 1**	**Trial 2**	**Trial 3**
Epochs	15	15	20
Validation split	0.2	0.3	0.2
Training accuracy	94.03%	**99.62%**	overfits
Validation accuracy	89.16%	**98.48%**	overfits
Testing accuracy	92.8%	**96.4%**	overfits

The below screenshot figures 2, 3 and 4 make it evident on the analysis of results. Figure 3 depicts the training and validation accuracies of trial 2 for epoch sizes of 20 and 15. Figure 2 shows the appearance of cardiac, knee joint and lumbar spine images in the test dataset. The notation used to signify cardiac images starts with '0', knee joint images with '1' and lumbar spine images with '2'. Figure 4 signifies the task of image classification which is realized by prediction of the classes aptly based on the sequence of input images of random notation order supplied from the test dataset.

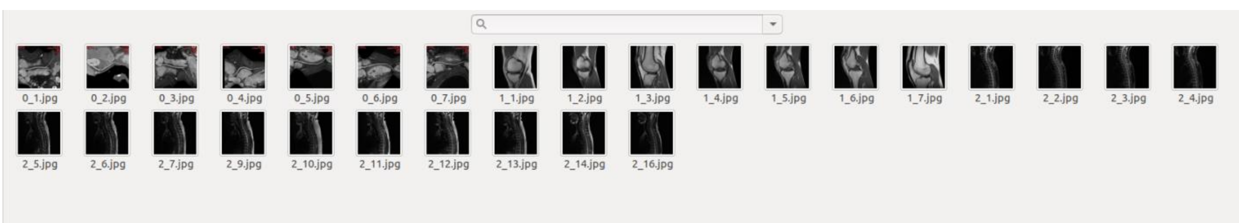
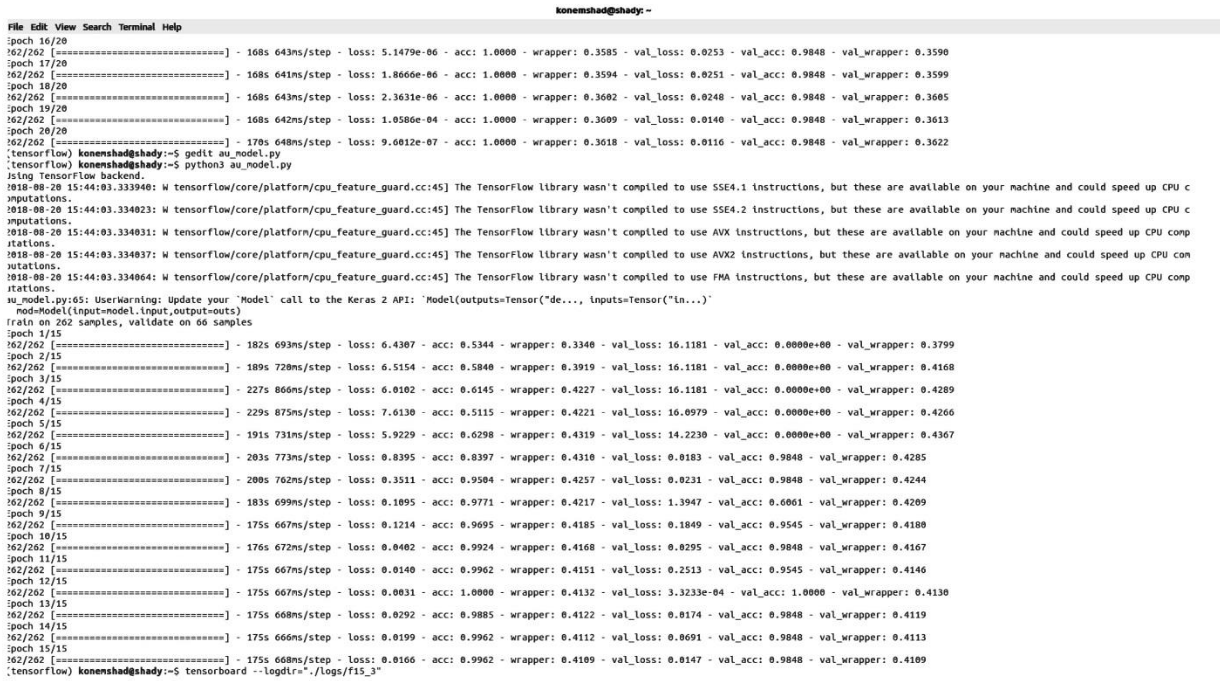


Fig.2: The test dataset comprising of cardiac, knee joint and lumbar spine images



```
konemshad@shady: ~
File  Edit  View  Search  Terminal  Help
:poch 16/20
262/262 [==============================] - 168s 643ms/step - loss: 5.1479e-06 - acc: 1.0000 - wrapper: 0.3585 - val_loss: 0.0253 - val_acc: 0.9848 - val_wrapper: 0.3590
:poch 17/20
262/262 [==============================] - 168s 641ms/step - loss: 1.8666e-06 - acc: 1.0000 - wrapper: 0.3594 - val_loss: 0.0251 - val_acc: 0.9848 - val_wrapper: 0.3599
:poch 18/20
262/262 [==============================] - 168s 643ms/step - loss: 2.3631e-06 - acc: 1.0000 - wrapper: 0.3602 - val_loss: 0.0248 - val_acc: 0.9848 - val_wrapper: 0.3605
:poch 19/20
262/262 [==============================] - 168s 642ms/step - loss: 1.0586e-04 - acc: 1.0000 - wrapper: 0.3609 - val_loss: 0.0140 - val_acc: 0.9848 - val_wrapper: 0.3613
:poch 20/20
262/262 [==============================] - 170s 648ms/step - loss: 9.6012e-07 - acc: 1.0000 - wrapper: 0.3618 - val_loss: 0.0116 - val_acc: 0.9848 - val_wrapper: 0.3622
(tensorflow) konemshad@shady:~$ gedit au_model.py
(tensorflow) konemshad@shady:~$ python3 au_model.py
Jsing TensorFlow backend.
2018-08-20 15:44:03.333940: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU c
omputations.
2018-08-20 15:44:03.334023: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine and could speed up CPU c
omputations.
2018-08-20 15:44:03.334031: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine and could speed up CPU comp
utations.
2018-08-20 15:44:03.334037: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but these are available on your machine and could speed up CPU com
putations.
2018-08-20 15:44:03.334064: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but these are available on your machine and could speed up CPU comp
utations.
au_model.py:65: UserWarning: Update your `Model` call to the Keras 2 API: `Model(outputs=Tensor("de..., inputs=Tensor("in...)`
  mod=Model(input=model.input,output=outs)
Train on 262 samples, validate on 66 samples
:poch 1/15
262/262 [==============================] - 182s 693ms/step - loss: 6.4307 - acc: 0.5344 - wrapper: 0.3340 - val_loss: 16.1181 - val_acc: 0.0000e+00 - val_wrapper: 0.3799
:poch 2/15
262/262 [==============================] - 189s 720ms/step - loss: 6.5154 - acc: 0.5840 - wrapper: 0.3919 - val_loss: 16.1181 - val_acc: 0.0000e+00 - val_wrapper: 0.4168
:poch 3/15
262/262 [==============================] - 227s 866ms/step - loss: 6.0102 - acc: 0.6145 - wrapper: 0.4227 - val_loss: 16.1181 - val_acc: 0.0000e+00 - val_wrapper: 0.4289
:poch 4/15
262/262 [==============================] - 229s 875ms/step - loss: 7.6130 - acc: 0.5115 - wrapper: 0.4221 - val_loss: 16.0979 - val_acc: 0.0000e+00 - val_wrapper: 0.4266
:poch 5/15
262/262 [==============================] - 191s 731ms/step - loss: 5.9229 - acc: 0.6298 - wrapper: 0.4319 - val_loss: 14.2230 - val_acc: 0.0000e+00 - val_wrapper: 0.4367
:poch 6/15
262/262 [==============================] - 203s 773ms/step - loss: 0.8395 - acc: 0.8397 - wrapper: 0.4310 - val_loss: 0.0183 - val_acc: 0.9848 - val_wrapper: 0.4285
:poch 7/15
262/262 [==============================] - 200s 762ms/step - loss: 0.3511 - acc: 0.9504 - wrapper: 0.4257 - val_loss: 0.0231 - val_acc: 0.9848 - val_wrapper: 0.4244
:poch 8/15
262/262 [==============================] - 183s 699ms/step - loss: 0.1095 - acc: 0.9771 - wrapper: 0.4217 - val_loss: 1.3947 - val_acc: 0.6061 - val_wrapper: 0.4209
:poch 9/15
262/262 [==============================] - 175s 667ms/step - loss: 0.1214 - acc: 0.9695 - wrapper: 0.4185 - val_loss: 0.1849 - val_acc: 0.9545 - val_wrapper: 0.4180
:poch 10/15
262/262 [==============================] - 176s 672ms/step - loss: 0.0402 - acc: 0.9924 - wrapper: 0.4168 - val_loss: 0.0295 - val_acc: 0.9848 - val_wrapper: 0.4167
:poch 11/15
262/262 [==============================] - 175s 667ms/step - loss: 0.0140 - acc: 0.9962 - wrapper: 0.4151 - val_loss: 0.2513 - val_acc: 0.9545 - val_wrapper: 0.4146
:poch 12/15
262/262 [==============================] - 175s 667ms/step - loss: 0.0031 - acc: 1.0000 - wrapper: 0.4132 - val_loss: 3.3233e-04 - val_acc: 1.0000 - val_wrapper: 0.4130
:poch 13/15
262/262 [==============================] - 175s 668ms/step - loss: 0.0292 - acc: 0.9885 - wrapper: 0.4122 - val_loss: 0.0174 - val_acc: 0.9848 - val_wrapper: 0.4119
:poch 14/15
262/262 [==============================] - 175s 666ms/step - loss: 0.0199 - acc: 0.9962 - wrapper: 0.4112 - val_loss: 0.0691 - val_acc: 0.9848 - val_wrapper: 0.4113
:poch 15/15
262/262 [==============================] - 175s 668ms/step - loss: 0.0166 - acc: 0.9962 - wrapper: 0.4109 - val_loss: 0.0147 - val_acc: 0.9848 - val_wrapper: 0.4109
(tensorflow) konemshad@shady:~$ tensorboard --logdir="./logs/f15_3"
```

Fig.3: Screenshot visualization of accuracies for epochs 20 and 15 respectively

```
File  Edit  View  Search  Terminal  Help

        test acc:0.964286
Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at 0x7f59e72b7e48>>
Traceback (most recent call last):
  File "/home/konemshad/.local/lib/python3.5/site-packages/tensorflow/python/client/session.py", line 595, in __del__
TypeError: 'NoneType' object is not callable
(tensorflow) konemshad@shady:~/Downloads/au_data/final_models/final_dir$ clear

(tensorflow) konemshad@shady:~/Downloads/au_data/final_models/final_dir$ python3 inference.py
Using TensorFlow backend.
2018-08-28 02:13:59.813620: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU c
omputations.
2018-08-28 02:13:59.813662: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine and could speed up CPU c
omputations.
2018-08-28 02:13:59.813669: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine and could speed up CPU comp
utations.
2018-08-28 02:13:59.813675: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but these are available on your machine and could speed up CPU com
putations.
2018-08-28 02:13:59.813681: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but these are available on your machine and could speed up CPU comp
utations.
['0_1.jpg', '1_3.jpg', '2_3.jpg', '2_16.jpg', '1_7.jpg', '2_10.jpg', '2_6.jpg', '2_5.jpg', '0_6.jpg', '1_2.jpg', '2_14.jpg', '2_4.jpg', '2_12.jpg', '2_13.jpg', '0_2.jpg', '0_5.jpg', '2_11.jpg', '0_3.jpg', '2_9.j
pg', '0_4.jpg', '2_2.jpg', '2_1.jpg', '1_4.jpg', '2_7.jpg', '1_5.jpg', '1_6.jpg', '1_1.jpg', '0_7.jpg']
cardiac
knee
spine
spine
spine
spine
spine
spine
cardiac
knee
spine
spine
spine
spine
cardiac
cardiac
spine
cardiac
spine
cardiac
spine
spine
knee
spine
knee
knee
knee
cardiac


        test acc:0.964286
(tensorflow) konemshad@shady:~/Downloads/au_data/final_models/final_dir$ |
```

Fig.4: Screenshot visualization for classification of the biomedical images from test set for trail

V. CONCLUSIONS AND DISCUSSIONS

In the arena of clinical applications, VGG16 model which is based on CNNs has made a breakthrough in performing classification provided decent training, sufficient dataset and adequate training time. Although this approach initially anticipates little assistance from the side of radiologists/ clinicians for training, it offers a practical design to deal with the clinical reports to organize and access them as and when desired. This can disentangle the factor of time pressure on medical laboratories besides classifying the images appropriately.

In this work, VGG16 architecture is trained on the biomedical imaging community using TL. It has delivered venerable results of 99.62% training, 98.48% validation and 96.4 % testing accuracies respectively.

Currently there is a drastic increase in the number of complexities in the modalities of medical images such as CT, PET, MRI, fusion imaging and ultrasound etc. So, it is quite difficult for the radiologists to perform imaging tests to fetch adequate time in reading them and end up with precise reports. Yet, with the development of DL technology, it is becoming easy to segregate the scans into their respective classes automatically. In this manner, classifying the images can be refrained from manual intervention which will otherwise take lot of time. The radiologists can thereby analyze and diagnose the lesions that are suspicious in a particular labeled class of images.

VI. REFERENCES

[1]. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., . . . Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60-88. doi:10.1016/j.media.2017.07.005

[2]. Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014). Medical image classification with convolutional neural network. 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV).doi:10.1109/icarcv.2014.7064414

[3]. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition, EB/OL. 2015-11-04.

[4]. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolution neural networks, C. Proceedings of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2012: 1106-1114.

[5]. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., &Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).doi:10.1109/cvpr.2016.308.

[6]. Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. C., Mega, J. L., Webster, D. R., Dec. 2016. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs.JAMA 316, 2402–2410.

[7]. Liu, Y., Gadepalli, K., Norouzi, M., Dahl, G. E., Kohlberger, T., Boyko, A., Venugopalan, S., Timofeev, A., Nelson, P. Q., Corrado, G. S., Hipp, J. D., Peng, L., Stumpe, M. C., 2017. Detecting cancer metastases on gigapixel pathology images.arXiv:1703.02442.

[8]. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118.

[9]. Martin Abadi, AshishAgarwal, etaI, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. CoRR abs/1603.04467 , 2016.

[10]. Zhao, W. (2017). Research on the deep learning of the small sample data based on transfer learning.doi:10.1063/1.4992835

[11]. Wu, Y., &Ji, Q. (2016). Constrained Deep Transfer Feature Learning and Its Applications. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).doi:10.1109/cvpr.2016.551

[12]. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, RuslanSalakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, J. Machine Learning Reasearch. 2014(15): 1929-1958.

[13]. Geoffrey E Hinton, NitishSrivastava, Alex Krizhevsky, IlyaSutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, C. arXiv preprint arXiv:1207.0580, 2012.

[14]. Oquab M, Bottou L, Laptev I, et al. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks, C. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). USA: IEEE Computer Society, 2014: 1717-1724

[15]. Deng, J., Dong, W., Socher, R., Li, L., Li, K., &Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition.doi:10.1109/cvprw.2009.5206848

[16]. Jason Yosinski, Jeff Clune, YoshuaBengio, Hod Lipson. How transferable are features in deep neural networks J. Neural Information Processing Systems 27 (NIPS'14), NIPS Foundation, 2014.

[17]. Russakovsky, O. et al. ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vision 115, 211-252, doi:10.1007/s11263-015-0816-y (2015).

[18]. Rajaguru, H., &Prabhakar, S. K. (2017). An approach to classification of oral cancer using Softmax Discriminant Classifier. 2[19] Ide, H., & Kurita, T. (2017).Improvement of learning for CNN with ReLU activation by sparse regularization.2017 International Joint Conference on Neural Networks (IJCNN).doi:10.1109/ijcnn.2017.79661850[19] Ide, H., & Kurita, T. (2017).Improvement of learning for CNN with ReLU activation by sparse regularization. 2017 International Joint Conference on Neural Networks (IJCNN).doi:10.1109/ijcnn.2017.7966185
17 2nd International Conference on Communication and Electronics Systems (ICCES). doi:10.1109/cesys.2017.8321313

[19]. Ide, H., & Kurita, T. (2017). Improvement of learning for CNN with ReLU activation by sparse regularization. 2017 International Joint Conference on Neural Networks (IJCNN).doi:10.1109/ijcnn.2017.7966185