AGE ESTIMATION FROM BRAIN MRI IMAGES USING DEEP LEARNING

Tzu-Wei Huang[1], *Hwann-Tzong Chen*[1], *Ryuichi Fujimoto*[2], *Koichi Ito*[2], *Kai Wu*[3], *Kazunori Sato*[4], *Yasuyuki Taki*[4], *Hiroshi Fukuda*[5], *and Takafumi Aoki*[2]

[1]Department of Computer Science, National Tsing-Hua University, Taiwan
[2]Graduate School of Information Science, Tohoku University, Japan
[3]South China University of Technology, China
[4]Institute of Development, Aging and Cancer, Tohoku University, Japan
[5]Tohoku Medical and Pharmaceutical University, Japan

ABSTRACT

Estimating human age from brain MR images is useful for early detection of Alzheimer's disease. In this paper we propose a fast and accurate method based on deep learning to predict subject's age. Compared with previous methods, our algorithm achieves comparable accuracy using fewer input images. With our GPU version program, the time needed to make a prediction is 20 ms. We evaluate our methods using mean absolute error (MAE) and our method is able to predict subject's age with MAE of 4.0 years.

Index Terms— MRI, T1-weighted image, deep learning, age estimation, brain-aging

1. INTRODUCTION

As a human gets older, the structure of brain changes. Previous researches show that neurodegenerative diseases such as Alzheimer's disease (AD) or Parkinson's disease are associated with defective autophagy and usually result in brain atrophy [1, 2]. By comparing the actual age and the age estimated from brain MR images, a computer can help to identify if someone is a possible AD patient. In order to identify possible patients as early as possible, an accurate prediction method is needed. The most popular technology to scan the internal structure of human body is MRI. MRI scanner uses strong magnetic fields and radio waves to detect the response strength from different tissue. Several types of MR images can be computed from the response signal using different weighting methods. One of the weighting methods is T1-weighted image, and for human brain, the image can be further segmented into gray matter (GM), white matter (WM), and celebrospinal fluid (CSF) regions. Most of existing age predicting methods [3, 4] take one or more region masks as the input. Here we propose a method that takes raw T1-weighted images as input to reduce possible information loss during GM / WM / CSF segmentation. The method proposed in [4] uses segmented GM image following by relevance vector machine (RVM) to predict subject's age. In [3], they select important brain regions based on brain atlas map for better prediction accuracy. However, the execution time of their method increases if the number of selected regions increases. In this paper we propose a method that is fast and accurate compared to [3, 4].

2. THE DATASET

The dataset is provided by Aoba brain image research center and Sendai Tsurugaya project. All brain MR images are T1-weighted and gathered by 0.5T MR modalities. We select the brain images of healthy subjects among 20 to 80 years old. There are total 1099 subjects after the selection process. We randomly choose 600 subjects for training and the other subjects are used for evaluation. The spatial resolution of the original data for each subject is a grid of $256\times256\times124$ voxels. Following [3], all of the images are aligned and cropped with ICBM152 template by the MATLAB package SPM2. The new size of the data becomes $189\times157\times124$ voxels. In the following, we denote $\mathbf{X}_i$ the aligned images and y_i the age of subject i. Also, we use *slices* to represent transverse planes in the following literature.

3. METHOD

Given input MR images $\mathbf{X}_i$ of subject i, our algorithm predicts the subject's age $\hat{y}_i$, $i \in \{1, 2, \cdots, N\}$, where N is the number of training data. Each $\mathbf{X}_i$ contains S slices of the subject's brain images. We use $\mathbf{X}_i^{(s)}$, $s \in \{1, 2, \cdots, S\}$ to denote a certain image slice.

The neural network function is defined as $\hat{y}_i = f(\mathbf{X}_i; \mathbf{W})$, where $\mathbf{W}$ is the learnable parameter in the convolutional layers and the fully connected layers. We treat the prediction task as a regression problem and use

$$L = \sum_i |y_i - f(\mathbf{X}_i, \mathbf{W})|_2^2 \quad (1)$$

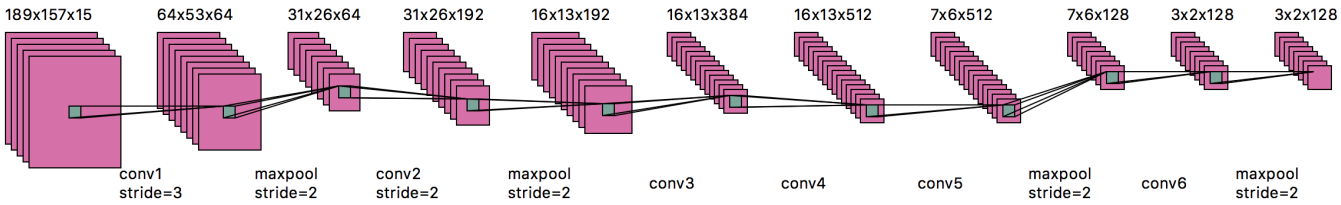


Fig. 1: The first part of our network architecture. Also known as feature extractor. The input slices are represented by the leftmost stacks. After the operations, the information in the slices is represented by a $3 \times 2 \times 128$ tensor. Each operation (conv, maxpool) is associated with a stride parameter, which controls how many units the filter shifts at a time. After training, the filter of the convolution will be learned. Note that due to page layout, the number of slices shown in this figure is not accurate and the last 3 fully connected layers are not drawn.

as our objective function. Training the neural network can be viewed as minimizing the above function with respect to W. We use back-propagation to get the gradient of the objective function and update the weight with stochastic gradient descent.

3.1. Data preprocessing and data augmentation

It is important to preprocess the data so that the neural network converges faster. The standard way is normalization. Specifically, we make the input data zero mean and unit variance. We preprocess both the training and testing data using channel-wise normalization. To do the channel-wise normalization, we first compute

$$M^{(s)} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}_i^{(s)} \tag{2}$$

for each s-th slice of training data $\mathbf{X}_i$. Also, we compute

$$\sigma_s = \frac{1}{N} \sum_{i=1}^{N} std(\mathbf{X}_i^{(s)}). \tag{3}$$

for each channel, where $std(\cdot)$ computes the standard deviation. Finally, for each $\mathbf{X}_i^{(s)}$, we compute

$$\hat{\mathbf{X}}_i^{(s)} = (\mathbf{X}_i^{(s)} - M^{(s)})/\sigma_s \tag{4}$$

as the input of the convolutional neural network.

Since there are only 600 subjects for training, in order to reduce overfitting and reduce the regression error, we augment the training set as follows. (*i*) Randomly sample c from $[0, 10]$, then crop $[c, c, W-c, H-c]$ from $\mathbf{X}_i^{(s)}$, and then scale it back to the original size $W \times H$. (*ii*) Randomly sample dx, dy from $[-5, 5]$, shift the input images with dx, dy, and then pad the image with zero. (*iii*) Flip the images horizontally with a probability of 0.5. Each slice of the brain undergoes the same cropping/translation parameter in one augmentation process.

3.2. Network architecture

Our network architecture is based on the idea of VGG net [5]. VGG net is a well known network architecture for image classification task. In this paper, we use *the power of small kernel* to build the network for regression task. There are 5 convolution layers with 3×3 kernel and several maxpooling layers with stride of 2. Each of the convolutional layer is followed by an ReLU activation layer. The last 3 layers are fully connected layers which blend the parameters to combine the feature vectors. The output of the network is a scalar, which indicates the predicted age. Please see Fig. 1 for details.

During development, we discover that there is no need to use all of the brain images of the subject. The reason is that mutual information of nearby slices is large. We test several sparsity configuration and find that we can reduce the input slices to merely 15 images without hurting performance. We finally choose slices numbered 31, 39, 47, 55, 63, 71, 79, 87, 95, 103, 111, 119, 127, 135, 143 that contain most of brain tissue as our input.

3.3. Implementation

The extremely fast execution speed is the strength of our algorithm. With CPU computation environment, the time for testing is 200 ms for each prediction. If GPU computation environment is available, the time for testing is 20 ms for each prediction. Compared with previous methods, our method has a great improvement over the execution time and achieves better accuracy.

Recent deep learning models require large GPU memory to hold the neural network. Unlike VGG net, with our tailored network architecture, the GPU memory required is 650MB. Hence almost all consumer GPUs are capable of running it. The time needed to train the network is 12 hours (for best accuracy). To be par with other methods (MAE 4.3), the network only need 0.5 hours to train. All experiments are performed on a PC with Intel i5-4460 CPU and Nvidia TITAN X GPU. We use torch [6] to train the network.

Hyperparameter	value
learning rate	0.0001
learning rate decay	0.0001
weight decay	0.001
momentum	0.9
batch size	16

Table 1: Hyper-parameters used in the experiment.

Method	MAE	Time[hour]
Franke et al. [4]	4.61	-
Kondo et al. [3]	4.32	0.0032
Proposed (GPU)	4.0	5×10^{-6}
Proposed (CPU)	4.0	5×10^{-5}

Table 2: Testing result. The computation time of the proposed method is 64 times faster than [3]. By using GPU, the speedup is 640 times faster.

3.4. Training

In this section, we describe the details for training the neural network. Here we explain *learning rate decay* and *weight decay*. *Learning rate decay* defines the speed to slowdown the step for the stochastic gradient descent algorithm. If this value is large, the training procedure converges to a local minimum fast. On the other hand, if this value is small, the training procedure converges slower, but will usually achieve better local minimum. *Weight decay* is used for regularization, if this value is large, the network will be less overfitted to the training data. The hyper-parameter used to train the neural network is listed in Table 1.

4. EXPERIMENT RESULTS

We follow the criterion used in [3] for comparison. Specifically, the criterion mean absolute error, MAE, is defined as

$$E = \frac{1}{N} \sum_{i=1}^{N} |y_i - f(\mathbf{X}_i, \mathbf{W})|. \quad (5)$$

We run our experiment for 10 times. Each time, we randomly sample 600 subjects as training data and use the other 499 subjects as testing data. We show the decrease of error in one of the trials in Fig. 3. The reported MAE in Table 2 is averaged among the ten trials.

4.1. Importance for different part of the brain

To show the influence of different brain locations with respect to prediction accuracy, we occlude different parts of the brain before feeding the MR images into the trained network. The induced prediction error is proportional to the importance of the occluded region. Two types of occlusions are tested, one is occluding the image with a 20×20 block and the other is with a 10×10 block. Each occlusion block changes its location in a sliding window fashion. Fig. 2 shows the accuracy map after occluding different parts of brain. For the big occlusion size, the largest MAE it induced is around 4.3 (brightest area, near *pineal* and *choroid plexus*), and for regions near the four corners, since the area has no information and thus has unchanged MAE around 4.0 (darkest area). For the smaller occlusion size, there is almost no performance impact. So we enhance the error map to show the relative importance. According to the above observation, our neural network has learned where to look at and has pretty robust performance dealing with corrupted data. In other words, even if the input image is not clear enough due to machine condition, we can still predict the result pretty accurately.

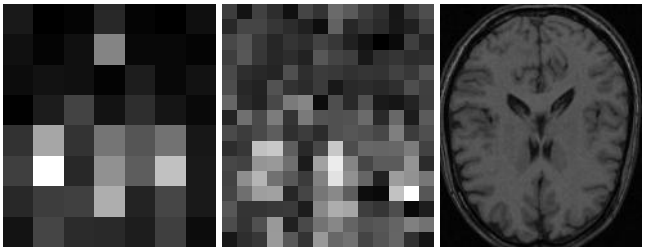


Fig. 2: Visualization of learned critical regions. Left: Error induced by occluding image parts with 20×20 block. The max error induced is MAE 4.3. Center: Relative error induced by occluding image parts with 10×10 block. The max error induced is MAE 4.04. Right: One of the image slice in the test set is shown for reference. The lower part of the image corresponds to face side. (slice number 87)

4.2. Training MAE vs. testing MAE

Fig. 3 shows the network learning process. The prediction error drops to less than 5 years in the first 200 epochs. One epoch means all training data are fed to the neural network once. At test time, the test image slices are resampled using the same method as we used for training data augmentation. We feed each resampled test image slices into the network and use the median of the outputs as our final prediction for certain subject's age.

5. CONCLUSION

In this paper, we propose a novel method to solve the problem of age estimation from brain MRI images. Furthermore, we speed up the computation and achieve good accuracy. In the future we are looking forward to predicting the age of both healthy and unhealthy subjects as well.

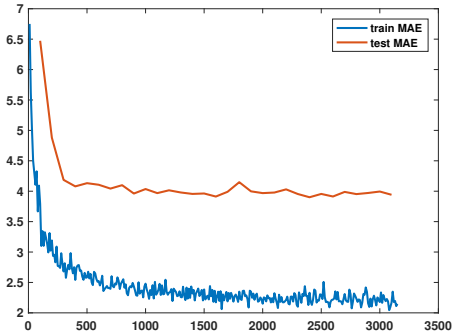


Fig. 3: Training error and testing error of one trial.

6. ACKNOWLEDGEMENT

MRI data acquisition and the use of them for the studies by H. Fukuda were approved by the Institutional Review Board of Medicine, Tohoku University, Japan. Informed consent was obtained from each subject after a full explanation of the purpose and procedures of the study, according to the Declaration of Helsinki (1991), prior to MR image scanning. Tzu-Wei would like to thank Bio A+ and COLABS programs.

7. REFERENCES

[1] Catriona D. Good, Ingrid S. Johnsrude, John Ashburner, Richard N.A. Henson, Karl J. Friston, and Richard S.J. Frackowiak, "A voxel-based morphometric study of ageing in 465 normal adult human brains," *NeuroImage*, vol. 14, no. 1, pp. 21 – 36, 2001.

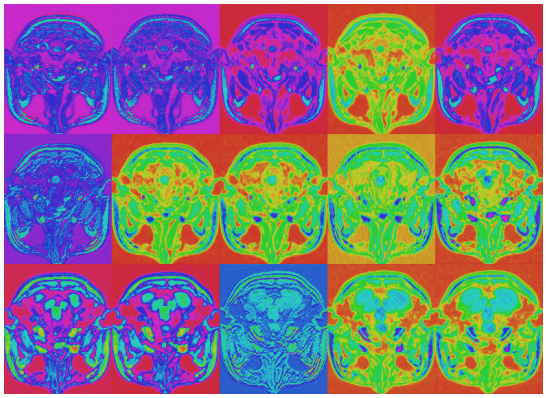
[2] Ralph A Nixon, "The role of autophagy in neurodegenerative disease," *Nat Med*, vol. 19, no. 8, pp. 983–997, aug 2013.

[3] C. Kondo, K. Ito, K. Wu, K. Sato, Y. Taki, H. Fukuda, and T. Aoki, "An age estimation method using brain local features for t1-weighted images," in *EMBC*, Aug 2015, pp. 666–669.

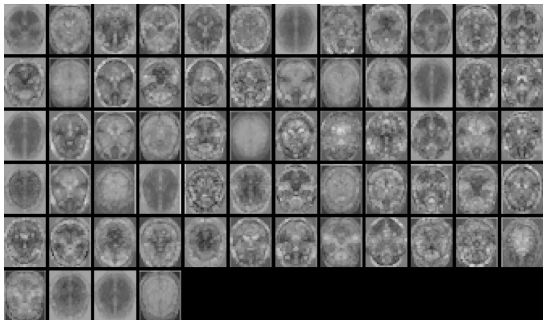
[4] Katja Franke, Gabriel Ziegler, Stefan Klöppel, and Christian Gaser, "Estimating the age of healthy subjects from T1-weighted MRI scans using kernel methods: Exploring the influence of various parameters," *NeuroImage*, vol. 50, no. 3, pp. 883–892, 2010.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

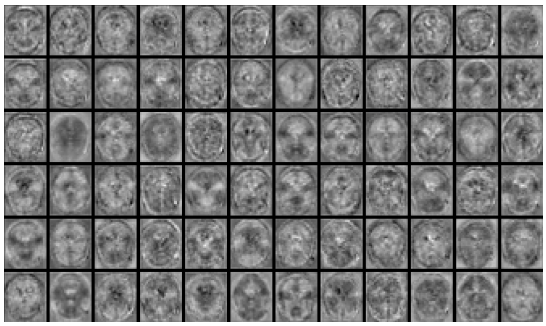
[6] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.



(a) For the first convolutional layer, there are 15 kernels for each output channel (64 in total). We choose one of the kernel set for visualization. The response map of each kernel is single channel, hence the original form is gray image. To enhance the result, we use false color to visualize the response.



(b) Feature response after *conv*1 layer. Each response map is computed by summing up the convolutional response of the 15 kernels.



(c) The response map after *conv*2 can be computed in the same way. There are totally 192 feature maps in this layer, only first 72 feature responses are shown.

Fig. 4: Visualization of feature response.