Tutorial

Healthc Inform Res. 2018 January;24(1):86-92.
https://doi.org/10.4258/hir.2018.24.1.86
pISSN 2093-3681 • eISSN 2093-369X

HIR
Healthcare Informatics Research

Applying Deep Learning in Medical Images: The Case of Bone Age Estimation

Jang Hyung Lee, PhD, Kwang Gi Kim, PhD
Department of Biomedical Engineering, Gachon University School of Medicine, Incheon, Korea

Objectives: A diagnostic need often arises to estimate bone age from X-ray images of the hand of a subject during the growth period. Together with measured physical height, such information may be used as indicators for the height growth prognosis of the subject. We present a way to apply the deep learning technique to medical image analysis using hand bone age estimation as an example. **Methods:** Age estimation was formulated as a regression problem with hand X-ray images as input and estimated age as output. A set of hand X-ray images was used to form a training set with which a regression model was trained. An image preprocessing procedure is described which reduces image variations across data instances that are unrelated to age-wise variation. The use of Caffe, a deep learning tool is demonstrated. A rather simple deep learning network was adopted and trained for tutorial purpose. **Results:** A test set distinct from the training set was formed to assess the validity of the approach. The measured mean absolute difference value was 18.9 months, and the concordance correlation coefficient was 0.78. **Conclusions:** It is shown that the proposed deep learning-based neural network can be used to estimate a subject's age from hand X-ray images, which eliminates the need for tedious atlas look-ups in clinical environments and should improve the time and cost efficiency of the estimation process.

Keywords: Bone Age, Deep Learning, Python, Tensorflow, X-ray Imaging

I. Introduction

Medical imaging provides a modality of data that is important for a range of diagnostic decisions. Machine learning techniques have been applied in medical image analysis to aid diagnostic efforts for many years. Typically, a set of raw input images together with labeled ground truths (e.g., cancer positive and negative, or quantification of severity) is used to train a model. A trained model is supposed to provide a suitable mapping from raw input images to an appropriate label. With traditional machine learning approaches, features to represent input are usually hand-crafted.

Deep learning is a method of finding optimal feature representations and a combination thereof through the use of the representational and learning capacity of neural networks. Since the successful use of a deep learning neural network was demonstrated for an image classification task [1], the deep learning technique has been gaining ground in medical image analysis. Cognitive and inferential tasks involving medical images include classification, regression, segmentation, and detection. In particular, convolutional neural networks proved effective in extracting and quantifying image characteristics.

Submitted: December 13, 2017
Revised: 1st, January 12, 2018; 2nd, January 23, 2018
Accepted: January 23, 2018

Corresponding Author
Kwang Gi Kim, PhD
Department of Biomedical Engineering, Gachon University School of Medicine, 38 Dokjeom-ro 3beon-gil, Namdong-gu, Incheon 21565, Korea. Tel: +82-32-458-2770, E-mail: kimkg@gachon.ac.kr

A deep learning architecture typically consists of a succession of distinct kinds of layers that perform operations, including image patch convolution and maximum response sampling. A large number of deep learning platforms have been proposed and are in use, including Caffe, Tensorflow, Keras, Theano and Torch, to name a few. Caffe was one of the first proposed platforms. A feature afforded by the platform is orthogonal separation of the network architecture and training schedule definitions from executable programs to interpret them and perform necessary operations. In this tutorial, we present an example case in which deep learning is used for medical image analysis, specifically hand bone age estimation.

Hand bone morphology is known to be related to the stage and prognosis of physical growth of young children and adolescents [2]. Greulich and Pyle [2] proposed an atlas to facilitate bone age estimation. Tanner and Whitehouse method [3] specified a reference set of age-discriminative points on hand. Hand x-ray images exhibit quite distinct morphological stages as subjects' age from infancy, early childhood, middle childhood, and adolescence (Figure 1). During the early stage of bone growth, the metaphyses constitute the bulk of bone. Ossification taking place in the epiphyses is a notable characteristic of the aging process.

While bone age in general correlates and is commensurate with chronological age, some people show substantial deviation from the typical phases of bone growth. People well under average height during the growth period may require special treatment depending on the severity and diagnosis outcome. Existing bone age estimation methods provide a reference set of regions where ossification is conspicuous. The user compares images from reference regions in the atlas against corresponding regions from the X-ray image of the subject and determines the bone age based on criteria of similarity. Manual determination using atlases may suffer from subjectivity and lack of quantitativeness.

II. Applying Deep Learning to Estimate Bone Ages

1. Problem Formulation and Data Preparation

A model is sought that takes hand x-ray images as input and outputs the corresponding subject's age. The continuity of age variable naturally suggests a regression-based approach.

$$a = R(I),$$

where a is age (months), R is a regression function, and input image, $I \in N\hat{}(W^{*}H)$, (W, H: width and height of image).

A set of hand X-ray images were collected to form a training data set for the regression model. Four hundred images were randomly selected and pooled to be used as the training set, and 200 images were randomly selected for use as a test set. Any hand X-ray images from other sources could be used as well, as long as they properly contain the hand region and tips of the lower parts of the forearm bones (radius and ulna). In this tutorial, the Radiological Society of North America (RSNA) challenge dataset [5] was used, which includes approximately 12 thousand image files in 8-bit-depth grayscale portable network graphics (png) format. The images exhibited a widely varying range of intensities, and the hands therein showed a substantial variation in postures and directions. Image size ranged from 169 to 3639 kilobytes, and the side dimensions ranged from 594 to 2970 pixels.

Since the conditions under which X-ray images were

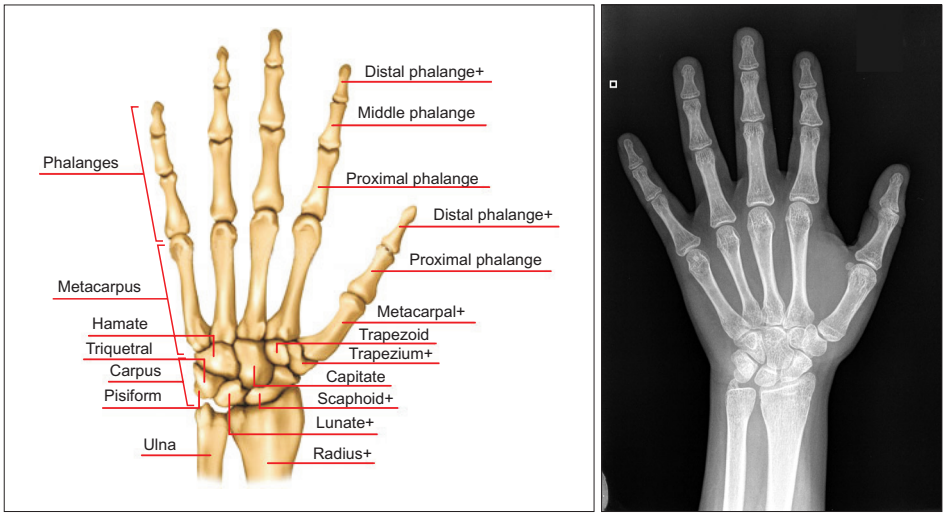

Figure 1. Skeletal anatomy [4] and an X-ray image of a hand [5].

taken varied from one medical institution to another and the filmed subjects' hand postures were highly variable, the raw input images were first normalized. Two points located towards the distal ends of ulna and radius and five points on proximal ends of the phalangeal bones were marked to serve as a set of reference points. A rectangle that is aligned to the borders of each image and assumes the smallest area was uniquely defined that includes all the reference points in its interior. This rectangle included the epiphyses of the carpal and metacarpal bones and the proximal ends of the phalanges to help reduce appearance variability due to stretched out or cupped fingers (Figure 2).

The rectangle region was cropped out to represent the corresponding image. Each image was then histogram-equalized to help reduce intensity components that are unrelated to bone morphology. The MATLAB function *histeq* was used. Caffe requires either *lmdb* or *leveldb* format files as its input. We used lmdb format, which facilitates high-speed memory-mapped data retrieval. The Caffe tool suite features the program *convert_imageset* to support packing of data files into an lmdb format file. The CaffeNet network assumes a fixed width and height of 224 pixels for input images. The command that was used to invoke the program was the following:

./convert_imageset -resize_height 224 -resize_width 224 (input image folder name) (age label list filename) (target lmdb folder name)

The pooled and packed dataset was then fed to the neural network for training. The overall process of data preparation was conducted as shown in Figure 3.

2. Environment Setting

Using a neural network entails a large number of convolution operations, which purpose-built hardware such as a GPU can more efficiently perform than a general purpose CPU. The hardware used for our process was an i7 CPU, an NVIDIA GTX 1060 GPU with 8 GB RAM. A Linux Ubuntu OS was used. The NVIDIA GPU required installation of a set of library and driver software. Fresh Ubuntu distribution comes with a default display driver, which is different from Nvidia's. A suite of NVIDIA driver software has to overwrite it in order to take effect. Installation was conducted in the following order: graphics drivers, CUDA library, and CUDNN library.

Caffe comes pre-packaged with Ubuntu versions 17.04 or later. It can be simply installed by using the command: *sudo apt install caffe-cuda*. For earlier Ubuntu versions, the following installs the required library files:

sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler
sudo apt-get install --no-install-recommends libboost-all-dev
sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev

Then the Caffe package can be downloaded and installed by using:

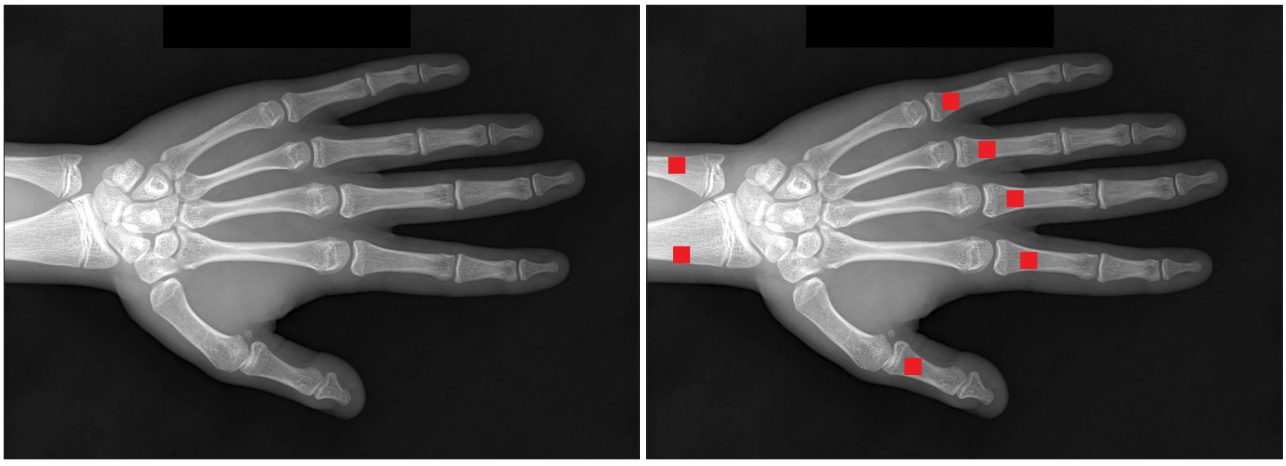

Figure 2. Example input image [5] with marked feature points.

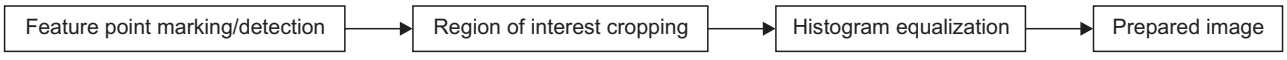

Figure 3. Data preparation process.

 https://doi.org/10.4258/hir.2018.24.1.86

git clone https://github.com/BVLC/caffe
cp Makefile.config.example Makefile.config
make all
make test
make runtest

Then the Caffe package can be downloaded from [6] and installed.

More details for Caffe installation are available in manuals, such as [6].

3. Neural Network: Architecture and Weight Setting

Traditional machine learning methods often entail processes of manually defining features, which may not necessarily be optimal representations for problems at hand. However, deep neural networks for image recognition, including CNN, can assume minimally processed input and find optimal network configuration through a training procedure.

Several distinct layers are employed as its building blocks. The convolutional layer performs convolution between kernel filters and patches of an input image. The max pooling layer then pools out the maximum values from the convolution filter responses from the preceding convolutional layer. This reduces intermediate data representation within the network to a more manageable size. Typically, one or more fully connected layers then follow to perform inner product operations to produce the final output value(s). Various kinds of neural network architectures have been proposed for a diverse range of applications.

In this tutorial, we used CaffeNet [7], which has rather low complexity to nicely serve the purpose of illustration of hand X-ray age estimation using deep learning. CaffeNet has many edges connecting its neurons, values of which should be fixed to be put into use. Frequently, the order of the training dataset size is far smaller than the order of the number of edge weights to be determined. A well curated image database may serve as a seed dataset to help steer the edge weight configuration.

We used a weight set pre-trained on ImageNet [8] which comprises more than one million image data instances. A set of prototext files are included; the file *train_val.prototxt* defines the network architecture to be used during the training phase. The file *solver.prototxt* specifies the training schedule, and *deploy.prototxt* defines the network architecture that is effective during the test and deployment phases. Originally, CaffeNet was designed for classification tasks, and it uses softmax loss as its error measure (Figure 4). Since the function our model has to learn is regression, the final layer was re-defined as shown in Appendix 1.

Due to the stochastic nature of neural network training and the convoluted topology of the learning space, setting appropriate training parameters is important for securing the chances and determining the rate of convergence. The parameter values used are shown in Appendix 2. The following command was used to invoke the Caffe program with input files prepared according to the previous description:

/home/jang/caffe/build/tools/caffe train --weights (weight file name) --solver solver.prototxt --gpu 0

Here, the names of the pretrained weight file and the training schedule file were supplied as arguments. A single GPU device numbered 0 was used. Each training data instance was forward propagated through the network, and the resulting output was compared against the target label to produce difference error value. Error values accumulated over a set number of data instances (batch) were summed and then backward propagated to adjust the weight values [9]. Training was iterated over the training dataset until the error reached a pre-determined level or plateau.

Mean absolute difference (MAD) was used to quantify the

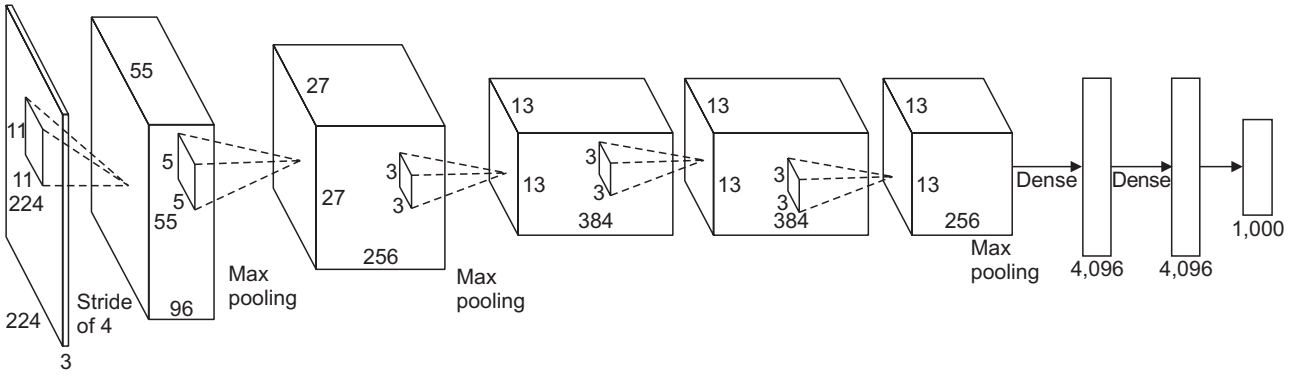

Figure 4. Caffenet architecture [1].

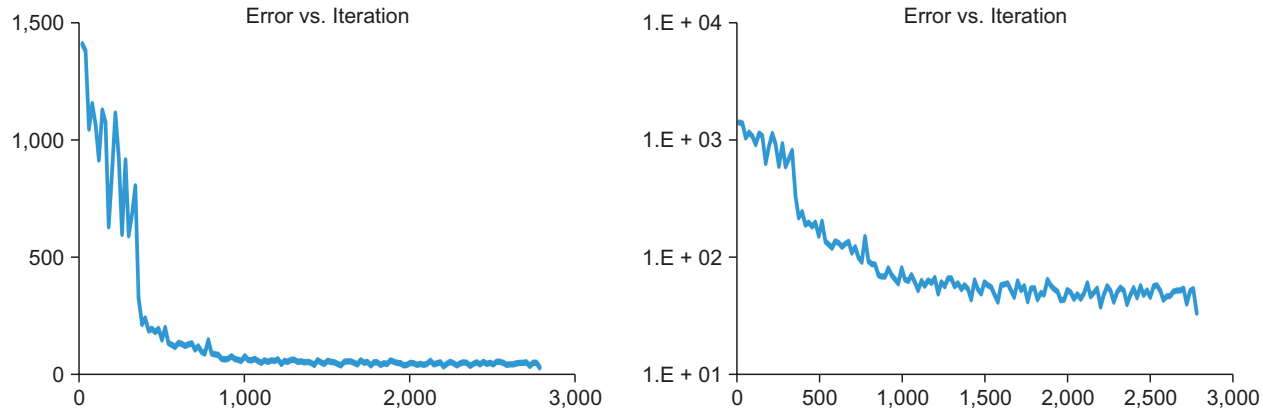

Figure 5. Error versus iteration in linear and log scales.

difference errors between true subjects' ages and the estimated ages:

$$MAD = \frac{\sum_i^n |x_i - avg(x)|}{n}$$

The concordance correlation coefficient (CCC) was used to quantify the collinearity between two sets of corresponding values:

$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$$

The MAD value obtained over the training set was 6.4 months, and the CCC value was 0.98. The maximum error was 24.4 months. For the test set, the MAD value was 18.9 months, and the CCC value was 0.78, exhibiting a slight performance deterioration. The maximum error was 69.2 months implying substantial room for improvement in order for the proposed method to be viable as a practical clinical tool.

4. Training Iteration

The maximum number of iterations was set to 3,000. At the beginning of training, error decreased rather quickly, although with substantial fluctuations. By the 1,000th iteration, error already reached a plateau, implying convergence of the network weights. Snapshots could be taken throughout training to store weight value configuration into files, the period of which is specified by the parameter *snapshot*: (*snapshot period*) in the file *solver.prototxt* (Figure 5).

III. Discussion

In this paper, we presented a way to use a deep learning convolutional neural network to estimate bone age from a subject's hand X-ray images. A set of feature points on the hand were defined to serve as reference points for cropping a certain region that is informative in terms of aging-induced morphological changes. Age estimation was cast as a regression problem, and a relatively simple network was employed and modified for illustrative purpose. The process of marking reference points on the hand was performed manually. To realize a fully automatic estimation process, procedures such as hand image detection and feature point detection may be implemented.

There was a substantial difference between the training and test performance values, which suggests a slight overfit due to the rather small training data set size used and/or to the rather shallow network architecture. Possibly using a larger training set with a larger network could help improve the generalizability of this method of model training.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgments

We would like to acknowledge the financial support from the Gachon University research fund of 2017 (No. GCU-2017-0211).

References

1. Krizhevsky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25:1097-105.
2. Greulich WW, Pyle SI. Radiographic atlas of skeletal de-

 https://doi.org/10.4258/hir.2018.24.1.86

velopment of the hand and wrist. 2nd ed. Stanford (CA): Stanford University Press; 1959.
3. Tanner JM, Whitehouse RH. Clinical longitudinal standards for height, weight, height velocity, weight velocity, and stages of puberty. Arch Dis Child 1976;51(3):170-9.
4. Visual Dictionary Online. Hand [Internet]. [place unknown]: Visual Dictionary Online; c2017 [cited at 2018 Jan 5]. Available from: http://visual.merriam-webster.com/human-being/anatomy/skeleton/hand.php.
5. Radiological Society of North America. Pediatric bone age challenge [Internet]. Oak Brook (IL): Radiological Society of North America; c2017 [cited at 2018 Jan 5]. Available from: http://rsnachallenges.cloudapp.net/competitions/4.
6. Berkeley Artificial Intelligence Research. Caffe Installation [Internet]. Berkeley (CA): Berkeley Artificial Intelligence Research; c2014 [cited at 2018 Jan 5]. Available from: http://caffe.berkeleyvision.org/installation.html.
7. BVLC/caffe [Internet]. [place unknown: place unknown]: c2017 [cited at 2018 Jan 5]. Available from: https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.
8. Deng J, Dong W, Socher R, Li LJ, Li K, Li FF. ImageNet: a large-scale hierarchical image database. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2009 Jun 20-25; Miami, FL. p. 248-55.
9. Wikipedia. Backpropagation [Internet]. [place unknown]: Wikipedia; c2017 [cited at 2018 Jan 5]. Available from: https://en.wikipedia.org/wiki/Backpropagation.

Appendix 1. Final layer for regression network

```
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}

layer {
  name: "loss"
  type: "EuclideanLoss"
  bottom: "fc8"
  bottom: "label"
  top: "loss"
}
```

Appendix 2. Training parameter values

```
net: "train_val.prototxt"
test_iter: 100
test_interval: 100
base_lr: 0.000003
lr_policy: "step"
gamma: 0.1
stepsize: 800
display: 20
max_iter: 3000
momentum: 0.9
weight_decay: 0.0005
snapshot: 1000
snapshot_prefix: "SNAPSHOT"
solver_mode: GPU
```


https://doi.org/10.4258/hir.2018.24.1.86