SUPPLEMENTAL METHODS

Introduction

Our system generates end-to-end bone mineral density (BMD) estimates from abdominal computed tomography (CT) scans via three successive steps:

1. Localization: Given a subject's abdominal CT scan, the system detects the center transverse CT slice from the L1, L2, L3, and L4 vertebrae.
2. Segmentation: For each lumbar vertebra, four CT slices—the center slice, the slice above the center, and two slices below the center—were selected based on the output from the localization step and segmented to retain only the vertebral bone.
3. Estimation: Four cropped regions corresponding to each vertebra are concatenated counterclockwise to generate a BMD using convolutional neural network (DXA_{CNN}) prediction for each vertebra independently.

Localization of lumbar spines

Inspired by the work of Belharbi et al. [1] and Kanavati et al. [2], we proposed a method for localizing the center L1, L2, L3, and L4 axial CT slices using maximum intensity projections (MIPs). The MIPs were images generated by taking the highest intensity value along a specific axis for each pixel of the projection. As bones have higher Hounsfield unit (HU) values than soft tissues, MIPs from CT stacks can generate two-dimensional (2D) representations of the bone structure of a subject. In this study, we clipped the intensity values between –100 and 1,000 HU to project both coronal and sagittal MIPs.

Our system takes a two-stage approach for this subtask: the first stage finds the slice corresponding to the top of the L1 vertebra, whereas the second stage outputs the central slice locations of the top four lumbar vertebrae based on the results of the first stage. Both stages were trained separately and combined during inference. Channel attention (CA) was employed and pre-trained networks were not used for slice localization.

Training details

1) Network architecture

Supplemental Fig. S1A illustrates the network architectures of the two models used in the subtask. Our networks are based on the Visual Geometry Group 19 (VGG19) architecture [3], which consists of 3×3 convolution layers, batch normalization [4], rectified linear unit (ReLU) activations, max-pooling layers for feature extraction, and a fully connected layer to generate outputs. In contrast to standard VGG19 models, CA blocks were applied before each pooling layer to improve model performance. Supplemental Fig. S1B shows the architecture of a CA block. First, the global average pooling was applied to the input features. Subsequently, the weight of each channel was calculated as a value between 0 and 1 after the 1×1 convolution and sigmoid layers. The calculated weights were then multiplied by the input feature.

2) Implementation

Both the sagittal and frontal MIPs were normalized to a range of 0 to 1. Horizontal flipping and shifting data augmentations were applied. An Adam optimizer with a momentum $\beta_1=0.9$, $\beta_2=0.999$, mean absolute error (MAE) loss, and a batch size of four were used. The initial learning rate (LR) is set to 1×10^{-4} and reduced by half for every 200 epochs. In addition, the neural networks in the first and second stages were trained for 1,000 epochs. Both networks were implemented using PyTorch (https://pytorch.org).

Ablation study for the channel attention

Supplemental Table S1 presents the results of the models trained with and without CA on our test set. The model trained with CA had lower MAE means and standard deviations, as well as fewer outliers, than the models trained without attention.

Lumbar spine segmentation

To reduce the cost associated with annotating a vast number of 2D CT slices and improve the segmentation performance, as shown in Supplemental Fig. S2A, a semi-supervised segmentation method was applied. Since the Mumford-Shah loss [5] does not require ground-truth masks, the network uses many unlabeled CT slices.

Specifically, given an input image x and predicted segmentation map y, the network is trained by minimizing the following loss function:

$$\text{Loss}=\alpha L_{CE}+\beta L_{MS} \quad (1)$$

where α and β are hyper-parameters, L_{CE} is the cross-entropy loss, and L_{MS} is the Mumford-Shah loss. The hyperparameter α is 1 if the input contains labeled masks and $\alpha=0$ otherwise. Accordingly, as illustrated in Supplemental Fig. S2A, the L_{CE} that requires ground-truth labels and predicted segmentation maps is computed when the networks use annotated data, whereas the Mumford-Shah loss, which requires inputs and network outputs, is applied to all training data. The L_{CE} is computed as

$$L_{CE}(y,g)=-\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} g_m(n)\log y_m(n) \quad (2)$$

where g is the ground truth label, M is the number of classes,

and N is the number of pixels. In addition, under the assumption that the segmentation map y is the output of the softmax layer of the network, the Mumford-Shah loss can be expressed as:

$$L_{MS}(x,y) = \sum_{m=1}^{M} \int_{\Omega} |x(n) - c_m|^2 y_m(n) dn + \lambda \sum_{m=1}^{M} \int_{\Omega} |\nabla y_m(n)| dn \quad (3)$$

where

$$c_m = \frac{\int_{\Omega} x(n) y_m(n) d(n)}{\int_{\Omega} y_m(n) dn} \quad (4)$$

This is the average pixel value of the m-th class of the input images. Therefore, the network was trained in a semi-supervised manner using the proposed loss Function (1).

Training details

1) Network architecture

Supplemental Fig. S2B illustrates the U-Net-like architecture used in the segmentation network. U-Net consists of encoder and decoder paths with skip connections between them and is widely used in image segmentation. The encoder was composed of four repeated convolution blocks for down-sampling. The convolution block has two-unit blocks formed by a series of 3×3 convolutions: batch normalization [4] and ReLU activation. The decoder was configured similarly to the encoder, with the convolution module containing a convolution transpose layer with a stride of two for up-sampling, which was repeated four times. For each up-sample, the number of channels was halved. The skip connections link the features from the encoder to the up-sampled features by concatenation. In the last layer of the network, a convolution layer with 1×1 kernels and a softmax layer was applied to generate probability maps for segmentation.

2) Implementation

The proposed semi-supervised vertebral segmentation method was implemented in Python using the PyTorch library. We down-sampled the 2D CT images from 512×512 to 256×256 by sub-sampling and augmented the data using horizontal or vertical flipping and 90° rotations. The network was trained using an Adam optimizer [6] with an initial LR of 1×10^{-5}. The LR was decreased by half after every 20 epochs. The batch size was set to four and the hyperparameter β to 0.01. The parameters of the network were initialized by the weights of a pre-trained network using only labeled data, and the network was trained for 100 epochs using a single NVIDIA Quadro RTX 6000 GPU (NVIDIA, Santa Clara, CA, USA).

Additional experimental results

Supplemental Table S2 presents the cross-validation results. When comparing the proposed semi-supervised method to the baseline supervised method, the Dice scores of the proposed method across all validation sets were higher than those of the comparative method. In addition, the average global Dice score of the validation sets using the proposed method showed a 0.12% gain over the baseline.

1) Qualitative segmentation results

Supplemental Fig. S3 illustrates the qualitative evaluation results for vertebral segmentation. For both male and female subjects, all lumbar vertebral regions were segmented more accurately using the proposed method than with the supervised method. Comparing the Dice score results of the test dataset for the two models, the proposed semi-supervised method outperformed the baseline.

Bone mineral density estimation

The inputs for the estimation task are based on the results of the two preceding steps. Four central axial CT slices were selected from each of the top four lumbar vertebrae and binary vertebral segmentation masks were acquired for the 16 images from the segmentation subtask. The Hadamard product of the mask and CT slice were taken, the pixel intensity values were clipped to (0,1000) HU, and a 196×196-pixel area in the image centered on the center of mass of the segmentation mask was cropped. These four images were then concatenated clockwise to generate a 392×392-pixel image that was resized to 98×98-pixels and min-max normalized by the sample to (0,1).

Network architecture

The regression network used for BMD estimation was based on the DenseNet169 architecture [7], with the output layer replaced by four blocks of successive fully connected layers with ReLU activation and batch normalization [4]. The fully connected layers (from bottom to top) had 1,024, 512, 128, and 8 nodes, respectively. The final output layer is a fully connected layer with a single node and linear activation. All layers in the model were initialized using the Glorot uniform initialization [8].

Implementation

The regressor was trained using CT-dual-energy X-ray absorptiometry pairs from 158 subjects and validated using data from 15 subjects. In line with other studies on deep learning in medical imaging [9,10], 30 regressors with a fixed random seed, archi-

tecture, dataset splits, and training procedure were trained and the five models with the lowest validation loss were assembled. The sliding window method generated 4,727 training and 417 validation samples. During the training, rotation (between –10° and 10°), horizontal and vertical shifts (by a factor of 0.05), horizontal flipping, and zooming (between 0.95 and 1.05) augmentations were randomly applied. The regressor models were trained using MAE loss with a batch size of 32 and an Adam optimizer [6] with an initial LR of 0.01, a β_1 of 0.9, a β_2 of 0.999, and an ϵ of 10^{-7} for 200 epochs. Both learning-rate scheduling and early stopping were applied based on the validation loss. LR was reduced by a factor of 0.33 if the validation loss did not improve for five epochs, when the LR was greater than 0.0001, and training was stopped if the validation loss did not improve for 15 epochs. The model was trained using Python 3.6, TensorFlow 1.13 (https://www.tensorflow.org/?hl=ko), and CUDA 10.0 (https://developer.nvidia.com/cuda-10.0-download-archive) on an Ubuntu 18.04 (https://releases.ubuntu.com/18.04/) workstation with an Intel i9-10940X processor (https://www.intel.co.kr/), 128 GB of memory, and an NVIDIA RTX Titan GPU.

SUPPLEMENTAL REFERENCES

1. Belharbi S, Chatelain C, Herault R, Adam S, Thureau S, Chastan M, et al. Spotting L3 slice in CT scans using deep convolutional network and transfer learning. Comput Biol Med 2017;87:95-103.
2. Kanavati F, Islam S, Aboagye EO, Rockall A. Automatic L3 slice detection in 3D CT images using fully-convolutional networks. arXiv [Preprint] 2018 Nov 22. https://doi.org/10.48550/arXiv.1811.09244.
3. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv [Preprint] 2015 Apr 10. https://doi.org/10.48550/arXiv.1409.1556.
4. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. Proc Mach Learn Res 2015;37:448-56.
5. Kim B, Ye JC. Mumford-Shah loss functional for image segmentation with deep learning. IEEE Trans Image Process 2019;29:1856-66.
6. Kingma DP, Ba JL. Adam: a method for stochastic optimization. arXiv [Preprint] 2014 Dec 22. https://doi.org/10.48550/arXiv.1412.6980.
7. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR); 2017 Jul 21-26; Honolulu, HI. Washington DC: IEEE Computer Society; 2017. p. 2261-9. Available from: https://doi.org/10.1109/CVPR.2017.243.
8. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proc Mach Learn Res 2010;9:249-56.
9. Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA 2016;316:2402-10.
10. Pierson E, Cutler DM, Leskovec J, Mullainathan S, Obermeyer Z. An algorithmic approach to reducing unexplained pain disparities in underserved populations. Nat Med 2021;27:136-40.