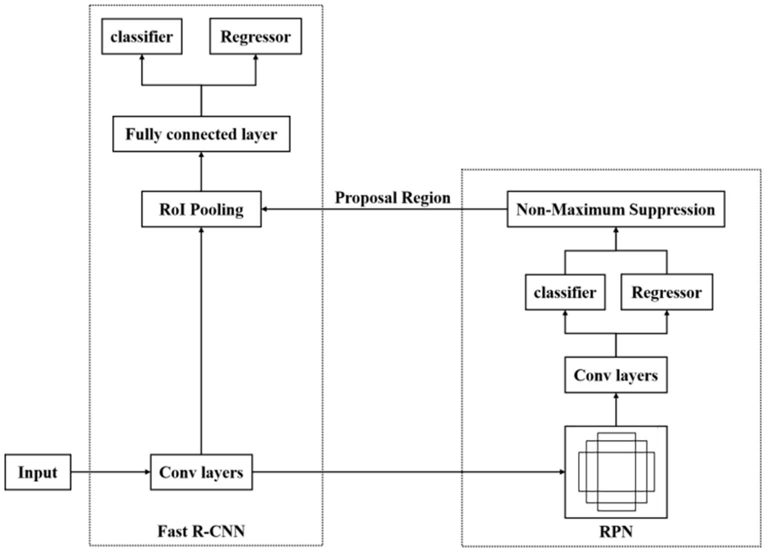
Supplementary

Appendix 1

Faster R-CNN model

Following figure shows the specific structure of the Faster R-CNN model.



The specific scheme consisted of the following steps.

1. Train the dataset using faster R-CNN and select the optimal model based on the recall and average accuracy of the evaluation metrics of the validation set.
2. Input the test set into the already trained faster R-CNN model and output the recall and average precision of the evaluation metrics to test the generalization ability of the model.
3. Input the training set, validation set, and test set into the trained faster R-CNN model respectively, save the vertical coordinates of the 3 vertebrae detection frames in the target image, and compare them with the labelled positions to observe the difference between them.

R-CNN, region-based convolutional neural network

3D AnatomyNet model

The specific scheme of 3D AnatomyNet model consisted of the following steps.

1. Divide the data into a training set, a validation set, and a test set according to a ratio of 6:2:2.
2. Train the training and validation sets using the AnatomyNet model and use the model with the highest average Dice coefficient of the evaluation metrics on the validation set as the final model.
3. Input the test set data into the trained segmentation model and check the average Dice coefficient to test the generalization ability of the model.
4. Input the training set, validation set, and test set into the trained model and output the Dice coefficients and segmentation results for each data to check whether there are multiple bones or mis-segmented vertebrae.

3D DenseNet model

The initial convolutional layer comprises convolution of a kernel size of 7×7×7, stride of 2, and padding of 3, followed by a BN layer and a ReLU activation function. The use of the ReLU activation function adds a non-linear factor to the neural network, which can improve the expressive power of the network and, to a certain extent, effectively alleviate the problem of gradient explosion and gradient disappearance during the training process of the network. It is defined by the following equation:

$$\mathrm{ReLU}(x)=\begin{cases}0, x\le 0\\ x, x>0\end{cases} \qquad [1]$$

The Max-pooling layer comprises pooling of a kernel size of 3×3×3, stride of 2, and padding of 1.

The bottleneck layer is used in dense block 1 to dense block 4, with the number of bottleneck layers in dense block 1 to dense block 4 being 6, 12, 24, and 16, respectively. Each bottleneck layer consists of 2 parts, each of which includes a BN layer, a ReLU layer, and a convolutional layer; the convolutional layer in the first part consists of a 1×1×1 convolutional kernel with a stride of 1; the convolutional layer in the second part consists of a 3×3×3 convolutional kernel with a stride of 1 and a padding of 1. Introducing 1×1×1 convolution before 3×3×3 convolution in each layer reduces the number of input feature-maps to improve computational efficiency.

Each transition layer connects 2 contiguous dense blocks, and each transition layer contains in turn a BN layer, a ReLU layer, a convolution layer, and an average pooling layer. The convolution layer comprises convolution of a kernel size of 1×1×1 and stride of 1. The pooling layer comprises pooling of a kernel size of 2×2×2 and stride of 2. To further improve the compactness of the model, a compression factor θ was added to the transition layer to reduce the number of feature-maps output by the transition layer. If a dense block contains m feature-maps, adding a compression factor will generate θm output feature-maps ($0<\theta\le 1$) for the subsequent transition layers. We set θ=0.5, growth rate=32 in our experiments.

Global average pooling was performed after dense block 4, specifying the output feature map size as 1×1×1.

The softmax layer was a softmax function applied to the output of the fully connected layer of the network to predict the class probability. Suppose y_1,y_2 are the output results of the fully connected layer of the network, the softmax function is defined as follows:

$$\mathrm{softmax}(y)_i = y_i' = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \qquad [2]$$

We trained the model for 50 epochs. The α-balanced variant of the focal loss was chosen as the loss function, which plays the role of balancing hard and easy samples, with the following expressions:

$$\mathrm{FL}(p_t) = -\alpha_t (1-p_t)^{\gamma} \log(p_t) \qquad [3]$$

$p_t\in[0,1]$ is the model's estimated probability for the class labeled 1. Meanwhile, $\alpha_t\in[0,1]$ is used to adjust the weights of positive and negative samples, and the tunable focusing parameter γ smoothly adjusts the rate at which tunable focusing, allowing the model to focus more on learning the features of hard samples, $\gamma\ge 0$.

Our experiment set the loss function parameters α_t=0.6, γ=2, initial learning rate of 3e-5, used the MultiStepLR decay strategy, milestones=[16,33], gamma=0.1, and used Adam as the default optimizer. After 50 epochs of iterative training, the model parameters were saved for each epoch and the best result was selected as the final model parameters.

ReLU, rectified linear unit; BN, batch normalization