ISSN: 1813-4890

Quality Optimization of Low Dose CT Based on RedCNN

Tingyu Li

College of Computer Science and Engineering, Northeastern University, Shenyang, 110819, China

20203465@stu.neu.edu.cn

Abstract

Consulting the potential danger of X-ray to patients, low-dose CT has aroused great interest in the field of medical imaging. At present, the mainstream low-dose CT methods include vendor-specific sinusoidal domain filtering and iterative reconstruction algorithms, but they need to access the original data while format is opaque to most users. Because it is different to mode El statistical features in image domain, the existing methods of directed processing reconstructed images can not eliminate image noise well, while preserving structural details. Inspired by the concept of deep learning, I connect automatic encoder, resolution network and shortcut to residual encoder-decoder conversion neural network (RED-CNN) for low-dose CT imaging.AD After patch-based training, RED-CNN has a competitive advantage over previous methods. It mainly invades three dimensions: image denoising level, detail protection degree and what to introduce artifacts. Finally, PSNR and SSIM are used as evaluation criteria.

Keywords

RedCNN, low-dose CT images, deep learning, residual neural network, auto-encoder.

1. Introduction

1.1. Significance of research

Low-dose CT imaging can greatly reduce the radiation dose to patients. By using lower radiation doses, patients' exposure to potential radiation risks can be reduced, especially for patients who need frequent CT examinations, such as cancer patients. However, reducing the radiation dose will lead to an increase in noise in the image. Noise may degrade the quality and visualization of images, thus affecting doctors' accurate interpretation of images and making correct diagnosis. Therefore, an important challenge in low-dose CT image processing is how to reduce radiation dose and image noise. At the same time, another important problem faced by low-dose CT images is how to preserve important structural details. The decrease of radiation dose may lead to the lack of clear contour and edge information in images, which may affect doctors' evaluation of anatomical structure. Therefore, the low-dose CT image processing method needs to reduce the image noise while preserving the structural details of the image. The research significance of this paper is to explore the application of depth learning method in low-dose CT imaging, especially using residual encoder-decoder convolutional neural network (RED-CNN). Traditional low-dose CT methods have some limitations in image noise elimination and structural detail preservation, and deep learning method may provide effective solutions to these problems. By connecting automatic encoders, deconvolution networks and shortcuts to RED-CNN, researchers aim to improve the quality of low-dose CT images and provide better patient safety and diagnostic accuracy for the medical imaging field. It is helpful to improve the safety of patients undergoing low-dose CT examination. In addition, by reducing image noise and preserving structural details, this method can improve doctors' interpretation and

ISSN: 1813-4890

diagnostic accuracy of CT images. This is very important for the detection and evaluation of early lesions, reducing misdiagnosis and improving the accuracy of treatment planning.

1.2. Solutions

The residual encoder-decoder convolutional neural network is introduced as the main image reconstruction model. RedCNN combines the structure of encoder and decoder, and adds shortcut connection in the middle to promote the flow and gradient propagation of information, thus improving the performance and convergence speed of the network.

1.3. Value

I. Deep learning method is introduced: traditional low-dose CT methods usually use filtering and iterative reconstruction algorithms based on mathematical models to reduce image noise. In contrast, this paper adopts a method based on depth learning, which uses automatic encoder and convolution neural network to learn image features and reconstruct images, thus achieving better performance in noise reduction and structure preservation.

II. Innovative network structure: the residual encoder-decoder convolutional neural network is introduced. RedCNN combines the structure of encoder and decoder, which can promote the flow and gradient propagation of information, thus improving the performance and convergence speed of the network.

Ⅲ. Image-domain-based processing: Different from the traditional methods, this paper adopts the depth learning method in image domain. This means that the network directly trains and processes the reconstructed CT images without accessing the original data or processing in other domains, thus improving the practical feasibility and ease of use of the algorithm.

ⅳ. patch-based training strategy: In order to improve the performance and generalization ability of the network, this paper adopts the training strategy based on patch. This means that the network samples and processes the image blocks in the training process, so as to better capture the local image features and their relationships, and improve the processing effect of the network on image noise and details.

2. Related work

2.1. Data set processing

This data set includes 9 groups of conventional dose CT and corresponding low dose CT. IMA file format, each group includes low-dose CT and conventional-dose CT, and the number of each group is not necessarily the same.

2.1. 1 Specific treatment method

In this project, the input CT is first sorted according to Z axis coordinates to reorganize the CT slices of patients. This step is regarded as the correct order of slices in 3D data set. Try to get the slice thickness of the entire dataset by measuring the distance between the first slice and the second slice. This is because CT data is usually a continuous 3D data set, so slice thickness can be approximately estimated by measuring this distance. If an exception occurs, the slice thickness is calculated by measuring the SliceLocation property of the first and second slices. This method can calculate the true CT slice thickness more accurately. Then traverse each slice and re-scale, according to the formula:

$$HU = \text{pixel value} * \text{Rescale Slope} + \text{Rescale Intercept}$$

The pixel value of the image is converted into the corresponding HU value.

Some illegal values are zeroed and truncated at the same time.

ISSN: 1813-4890

2.2. The Network Architecture

Build RedCNN network and train.

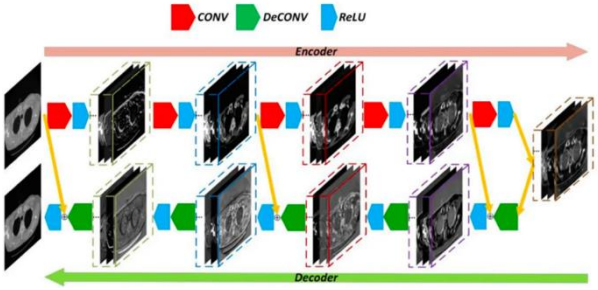

Fig.1. Overall architecture of proposed RED-CNN network.

2.1.1 specific network structure

This model is a convolution neural network model, which is used for image processing tasks. It is composed of encoder and decoder, and realizes cross-layer information transmission through residual connection. The main structure of the model includes convolution layer, deconvolution layer and ReLU activation function. The input of the model is a single channel image, and the output is a reconstructed image.

The following is an introduction to the relevant parts of the model:

I. Encoder section:

The model uses a series of convolution layers to extract features and encode input images. Specifically, it contains five convolution layers, and each convolution layer is followed by a ReLU activation function. The number of output channels of these convolution layers is out_ch, which is specified by setting parameters. In the process of encoder, the model uses residual connection to save the output of some layers as residuals, so as to fuse them with the output of corresponding deconvolution layers in the decoder. Here, residual_1, residual_2, and residual_3 denote the first, second, and third residuals, respectively. The output of the last convolution layer conv5 is the output of the encoder.

II. Decoder section:

The decoder section maps the output of the encoder back to the size of the original image through a series of deconvolution layers. The number of output channels of deconvolution layer is the same as that of convolution layer. In the process of decoding, ReLU activation function and deconvolution operation are used to restore the output of the encoder to the reconstructed image step by step. The output of each deconvolution layer in the decoder is fused with the corresponding residual, and the low-level and high-level features are fused through residual connection. The output of the last deconvolution layer, tconv5, is non-linearly mapped by the ReLU activation function, and redual_1 is added for final reconstruction.

On the whole, the RedCNN model uses residual connection to realize the cross-layer transmission of information, thus better preserving the details and context information of the image. Feature extraction and coding are carried out by the encoder, then feature reconstruction and image reconstruction are carried out by the decoder, and finally the reconstructed image is generated as the output of the model.

2.3. loss function

MSE is a commonly used loss function or evaluation index used to measure the difference between the predicted value and the true value. It calculates the average of the square of the difference between the predicted value and the real value.

2.3.1 MSE specific content

MSE is calculated by the following formula:

ISSN: 1813-4890

$$MSE = \left(\frac{1}{n}\right) * \sum (y - \hat{y})^2$$

Among them,

MSE: Mean square error

n: Sample size

y: True value

$\hat{y}$: Forecast value

Sigma: Sum symbol

The MSE calculation steps are as follows:

I. For each sample, calculate the difference between the true value y and the predicted value: y-$\hat{y}$

II. Square the difference value, that is.$(y - \hat{y})^2$

III. Add the square values of the differences of all samples, namely.$\Sigma(y - \hat{y})^2$

Ⅳ. Divide the sum by the number of samples n to obtain the mean square error MSE.

MSE is a non-negative value, and the closer it is to 0, the smaller the difference between the predicted value and the real value, and the more accurate the prediction result of the model. A larger MSE value indicates that there is a big difference between the predicted value and the true value, and the prediction of the model is not accurate enough.

MSE is often used in machine learning for model evaluation and loss function calculation of regression problems. By minimizing the MSE, the model can be closer to the real value, and the prediction accuracy of the model can be improved. In the optimization process, the parameters of the model can be adjusted to reduce the prediction error by gradient descent of MSE.

2.4. Optimizer

Adam Optimizer is a gradient-based optimization algorithm, which combines momentum method with the idea of adaptive learning rate.

2.4. 1 Adam Optimizer details

The principle of the Adam optimizer is as follows:

I. Initialization parameters: Adam optimizer needs to set learning rate and momentum parameters beta1 and beta2, and initialize the first moment m and the second moment v of momentum as zero vectors.

II. Calculating gradient: For each training batch, the gradient is calculated for the parameters of the model according to the loss function.

III. Updating the First and Second Moment Estimators: The exponentially weighted moving average method is used to calculate the First Moment Estimator m and the Second Moment Estimator v.

Update of first-order moment estimation m:

m = beta1* m + (1-beta1) * gradient

Update of second-order moment estimation v:

v = beta2 * v + (1-beta2) * ()$gradient^2$

Where gradient represents the gradient of the current batch, beta1 and beta2 are the specified hyper-parameters that control the smoothness of the first and second moment estimates.

IV. Correction deviation: Since the initial values of m and v are zero vectors, they tend to favor lower values. In order to solve this deviation problem, carry out deviation correction operation:

Deviation correction of first-order moment estimation;

ISSN: 1813-4890

$$m_{hat} = \frac{m}{beta1^t}$$

Deviation correction of second-order moment estimation;

$$v_{hat} = \frac{v}{(1 - beta2^t)}$$

Where t represents the number of steps of the current iteration.

V. Parameter updating: Update the parameters of the model according to the corrected first-order moment estimation, second-order moment estimation and learning rate: $m_{hat} v_{hat}$

$$\text{parameter} = \text{parameter} - \text{learningrate} * \frac{m_{hat}}{\sqrt{v_{hat}} + \text{epsilon}}$$

Where parameter represents a parameter of the model, learningrate is the learning rate, and epsilon is a tiny constant to avoid zero-dividing errors.

VI. Repeat steps 2 to 5 for subsequent batch iterations until a predetermined number of iterations or convergence conditions are reached.

Key features of the Adam optimizer:

I. Adaptive learning rate: Adaptively adjust the learning rate of each parameter, and scale according to the second moment estimation of gradient, so that different parameters have adaptive learning rate.

II. Momentum optimization: By using the first-order moment estimation to store the information of the previous gradient, it has the function similar to momentum and accelerates the process of gradient descent.

III. Deviation correction: The deviation of first-order moment estimation and second-order moment estimation is corrected by deviation correction operation, especially in the early stage of training.

3. Discussion on main achievements

3.1. data set processing

3.1.1.rezoom processing

```
ef get_pixels_hu(slices):
    image = np.stack([s.pixel_array for s in
slices])
    image = image.astype(np.int16)
   image[image == -2000] = 0
   filename = []
   for slice_number in range(len(slices)):
      intercept=
slices[slice_number].RescaleIntercept
      slope                                    =
slices[slice_number].RescaleSlope
            filename.append(slices[slice_numb
er].filename)
      if slope != 1:
        image[slice_number]=slope*
                image[slice_number].astype(n
                p.float64)
        image[slice_number]=
image[slice_number].astype(np.int16)
```

ISSN: 1813-4890

```
    image[slice_number]+=
np.int16(intercept)
 return np.array(image, dtype=np.int16),
filename
```

3.1.2 normalization

```
def normalize_(image,MIN_B=-1024.0,
MAX_B=3072.0):
  image = (image - MIN_B) / (MAX_B - MIN_B)
  return image
```

```
def normalize_ (image, MIN_B=-1024. 0,
MAX_B=3072. 0):
image = (image-MIN_B)/(MAX_B-MIN_B)
return image
```

3.1.3 calculation of slice thickness

```
def load_scan(path):
   slices = [pydicom.read_file(os.path.join(path, s)) for s in
        os.listdir(path)]
        slices.sort(key=lambda x: float(x.ImagePositionPatient[2]))
     try:
      slice_thickness=np.abs(slices[0].ImagePositionPatient[2] -
           slices[1].ImagePositionPatient[2])
   except:
      slice_thickness=np.abs(slices[0].SliceLocation -
           slices[1].SliceLocation)
   for s in slices:
      s.SliceThickness = slice_thickness
   # filename = []
   # for s in slices:
   #    filename.append(s.filename)
    return slices
```

3.2. RedCNN

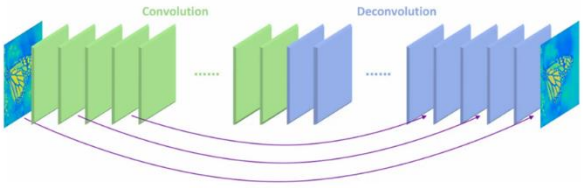

Fig.2. The Process of RedCNN

```
class RED_CNN(nn.Module):
   def __init__(self, out_ch=96):
      super(RED_CNN, self).__init__()
      self.conv1 = nn.Conv2d(1, out_ch,
kernel_size=5, stride=1, padding=0)
      self.conv2 = nn.Conv2d(out_ch, out_ch,
kernel_size=5, stride=1, padding=0)
      self.conv3 = nn.Conv2d(out_ch, out_ch,
kernel_size=5, stride=1, padding=0)
      self.conv4 = nn.Conv2d(out_ch, out_ch,
kernel_size=5, stride=1, padding=0)
```

ISSN: 1813-4890

```
        self.conv5 = nn.Conv2d(out_ch, out_ch, kernel_size=5, stride=1, padding=0)

        self.tconv1 = nn.ConvTranspose2d(out_ch, out_ch, kernel_size=5, stride=1, padding=0)
        self.tconv2 = nn.ConvTranspose2d(out_ch, out_ch, kernel_size=5, stride=1, padding=0)
        self.tconv3 = nn.ConvTranspose2d(out_ch, out_ch, kernel_size=5, stride=1, padding=0)
        self.tconv4 = nn.ConvTranspose2d(out_ch, out_ch, kernel_size=5, stride=1, padding=0)
        self.tconv5 = nn.ConvTranspose2d(out_ch, 1, kernel_size=5, stride=1, padding=0)

        self.relu = nn.ReLU()

    def forward(self, x):
        # encoder
        residual_1 = x
        out = self.relu(self.conv1(x))
        out = self.relu(self.conv2(out))
        residual_2 = out
        out = self.relu(self.conv3(out))
        out = self.relu(self.conv4(out))
        residual_3 = out
        out = self.relu(self.conv5(out))
        # decoder
        out = self.tconv1(out)
        out += residual_3
        out = self.tconv2(self.relu(out))
        out = self.tconv3(self.relu(out))
        out += residual_2
        out = self.tconv4(self.relu(out))
        out = self.tconv5(self.relu(out))
        out += residual_1
        out = self.relu(out)
        return out
```

4. Advantages and disadvantages of RedCNN

The RedCNN network has the following advantages:

I. Self-encoder structure: Using self-encoder structure, we can learn the low-dimensional representation of data, so as to capture important features in data. This has a good effect on image processing tasks such as image denoising and image compression.

II. Convolution operation: The convolution layer in the network can effectively process image data, and has the characteristics of translation invariance and local perception. Convolution operation can extract spatial features from images, and parameter sharing can reduce the number of parameters of the model and improve the computational efficiency.

III. Reconstruction ability: The network reconstructs the image through deconvolution layer, which can restore the details of the original image and make the reconstructed image as close as possible to the original image.

The RedCNN network may also have the following disadvantages:

I. Fixed network structure: The number of layers and convolution kernels in each layer of the network are fixed, which may limit its modeling ability for complex data. For complex image data or tasks, deeper networks or larger number of convolution kernels may be needed to improve the expressive ability of the model.

II. Information loss: The goal of self-encoder is to reconstruct the original data by learning the low-dimensional representation of the data, which may lead to a certain degree of information loss. Especially in the process of compression and reconstruction, some details may be lost or some reconstruction errors may be introduced.

III. Training complexity: For large-scale image data sets, the training of the network may require longer time and more computing resources. Especially in deep-layer networks, the deconvolution operation requires a large amount of computation, which may increase the complexity of training.

5. Experimental results

PSNR avg:33.7290
SSIM avg:0.9180
RMSE avg: 9.4261

Acknowledgements

This experiment was successfully completed under the patient guidance and strict requirements of the teachers in the Medical Imaging Laboratory of Northeastern University. From the subject design to the concrete implementation to the completion, all of them condensed the teachers' hard work. In this experiment, I can feel the careful guidance of the teachers, which made me benefit a lot. I would like to express my deep gratitude and lofty respect to the teacher.

References

[1] Li Zhaofeng. Construction method of efficient web page classifier in topic search engine [J]. Science and Technology Bulletin, 2013, 29 (08): 109-111. DOI: 10.13774/J.cnki.kjtb.2013. 08.047.

[2] M. Balda, J. Hornegger, and B. Heismann, “Ray contribution masks for structure adaptive sinogram filtering,” IEEE Trans. Med. Imaging 30(5), 1116–1128 (2011).

[3] Y. Zhang, Y. Xi, Q. Yang, W. Cong, J. Zhou, and G. Wang, “Spectral CT reconstruction with image sparsity and spectral mean,” IEEE Trans. Comput. Imaging 2(4), 510–523 (2016).

[4] D. Kang, P. Slomka, R. Nakazato, J. Woo, D. S. Berman, C.-C. J. Kuo and D. Dey, “Image denoising of low-radiation dose coronary CT angiography by an adaptive block-matching 3D algorithm,” Proc. SPIE 8669, 86692G (2013).