ENPM809K Final Project Report

Aditya Varadaraj
UID: 117054859

Saurabh Palande
UID: 118133959

Param Dave
UID: 117476323

Abstract

Computed Tomography (CT) scanners that are commonly-used in hospitals and medical centers nowadays produce low-resolution images, e.g. one voxel in the image corresponds to at most one-cubic millimeter of tissue. In order to accurately segment tumors and make treatment plans, radiologists and oncologists need CT scans of higher resolution. The same problem appears in Magnetic Resonance Imaging (MRI). In the paper, an CNN-based approach for single image super-resolution for 3D CT and MRI data is proposed. The model proposed in the main paper [1], has 2 CNNs with 10 convolutional layers and an intermediate upscaling layer that is placed after the first 6 convolutional layers. The First CNN increases the resolution on two axes (width and height), which is followed by a second CNN, which increases the resolution on the third axis (depth). Different from other methods, they compute the loss with respect to the ground-truth high-resolution image right after the upscaling layer, in addition to computing the loss after the last convolutional layer. The intermediate loss forces their network to produce a better output, closer to the ground-truth.

1. Introduction

The main motivation behind our work is to allow radiologists and oncologists to accurately segment tumors and make better treatment plans. In order to achieve the desired goal, we propose a machine learning method that takes as input a 3D image and increases the resolution of the input image by a factor of 2x or 4x, providing as output a high-resolution 3D image. Most popular Super-Resolution methods usually involve the use of either CNNs ([9], [7], [8], [11], [10]), GANs [6] or Attention Networks [5]. CNNs and Attention Networks are more widely used in super-resolution.

In the main paper [1], different from related methods [11] and [10], they compute the loss with respect to the ground-truth high-resolution image right after the upscaling layer, in addition to computing the loss after the last convolutional layer. The intermediate loss forces our network to produce a better output, closer to the ground-truth.

We note that the model in [1] and our modified model belongs to a class of deep neural networks known as fully convolutional neural networks. The main advantage of using such models, which do not include dense (fully-connected) layers, is that the input samples do not have to be of the same size. This flexibilty enables a broad range of applications such as image segmentation, object tracking, crowd detection, time series classification and single-image super-resolution.

2. Related Work

2.1. State of the art (S.O.T.A.)

According to papers with code benchmarks the current state of art paper is Multimodal Multi-Head Convolutional Attention with Various Kernel Sizes for Medical Image Super-Resolution (MHCA). The benchmark is trained in IXI Dataset which we are trying to use to train our network. The comparison is done on the basis of SSIM (Structural similarity index measure) loss and PSNR (Peak Signal to Noise Ratio) losses.

2.1.1 MMHCA [5]

The paper proposes a novel multi-modal multi-head convolutional attention module to super resolve CT and MRI scans. The network uses transform-type architecture. Their attention module in the transformer network uses convolution operation to perform spatial-channel attention. Here the shape of the kernel decides/controls the reduction rate of the spatial attention and also the channel attention. The paper introduces a unique multi-attention head where each head of the multi-attention head corresponds to a particular reduction rate for spatial attention. The multimodal attention is put in two deep learning networks. Generally Medical image super-resolution works can be grouped into two categories, where one category is focused on increasing

the resolution of individual CT or MRI slices (2D images), while the other is focused on increasing the resolution of entire 3D scans (volumes). The paper focuses on increasing the resolution of CT and MRI slices.

The general method is as follows. The network uses multi-contrast input formed of n low-resolution input images of pxp pixels denoted by LR1, LR2,..., LRn, and the goal is to obtain an HR image of rxr pixels, where r ¿ p generally r = 2p or 4p.

As shown in the Fig. 1 the low-resolution input is fed into the CNN encoder and converted into a tensor and all those tensors are concatenated. These concatenated tensors are then fed into different attention heads. Each attention head applies convolution and deconvolution operations, using a kernel size that is unique to the respective head. Then these tensors are summed up and passed through a sigmoid layer. The three main contributions of the paper are as follows

- First paper to present network that performs multi-modal low-resolution medical image super-resolution.
- Paper presents a novel multimodel multi-head convolution attention mechanism for multi-contrast medical image SR.
- Paper shows that compared to vanilla convolution networks, attention networks bring significant performance gains on three multi-contrast data sets.

The losses in the paper are the peak signal-to noise ratio (PSNR) and the structural similarity index measure (SSIM). Peak signal-to-noise ratio as the name suggests takes the ratio of the maximum possible signal and the power of the noise.

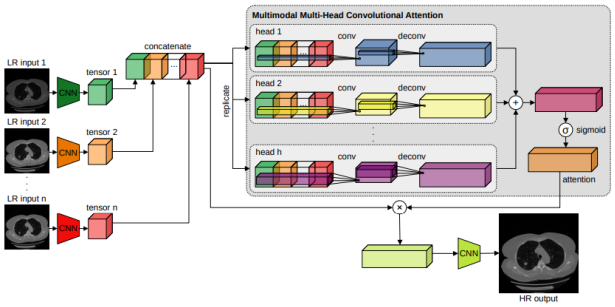

Figure 1. MMHCA Network Architecture

2.1.2 CNN-based methods [8] [9]

In [9], an enhanced super-resolution group CNN (ESRGCNN) with a shallow architecture (Fig. 2)is presented by fully fusing deep and wide channel features to extract more accurate low-frequency information in terms of correlations of different channels in single image super-resolution (SISR).

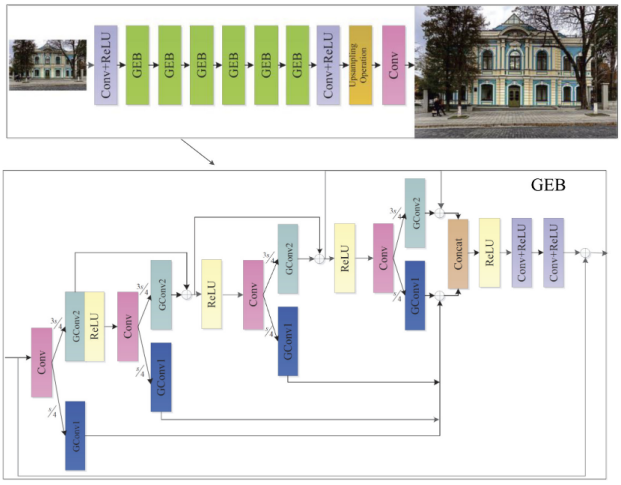

Figure 2. ESRGCNN Network Architecture

In [8], CNN is based on the form of a Unet autoencoder (Fig. 3) and we get to successfully recover the resolution from pediatric LR MRI. The CNN architecture includes skip connections, residual operations, and feature maps with distinct dimensions depending on the layer, allowing the network to learn features with many levels of complexity. Using consecutive convolutions, they extract critical features that determine the SR process during the encoding stage. Then, the objective is to reconstruct the output from deep features maps as dimension and domain from input and output MRI of CNN need to be the same in order to compare both at the evaluation stage.

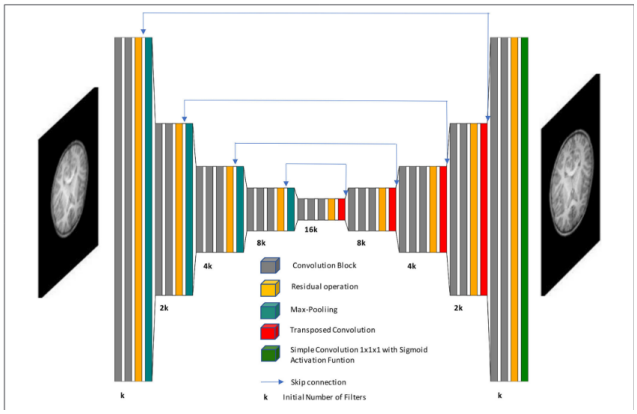

Figure 3. Pediatric MRI SR U-Net type CNN Architecture

3. Dataset

The Datasets used in the original paper are a custom dataset provided by the Medical School at Coltea Hospital and NAMIC (National Alliance for Medical Image Com-

puting) Brain Multi-modality MRI dataset [4] containing data of 20 subjects, which is no longer available online. They have used both T1 and T2-weighted images independently for their experiments.

The dataset we used is the IXI dataset [3] which is available online in NIFTI format. We performed some preprocessing to convert these images to PNG format for ease of visualization and performing mathematical operations. Also, we downsampled the 3D images so that the original images can be treated as ground-truth. We used the T1 and T2 - weighted images independently and treated each 3D MRI scan data as a collection of N (N=256 for T1) Axial plane 2D images (256 x 150 each in T1) stacked together. All the 3 views are shown in Figure 4, 5 and 6. We used 3D MRI data of 5 Subjects from IXI Dataset to train the model and data of 1 Subject to test the model.

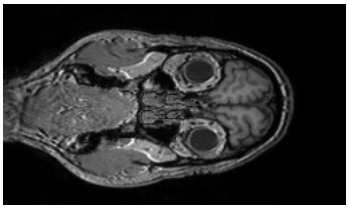

Figure 4. Axial MRI scan

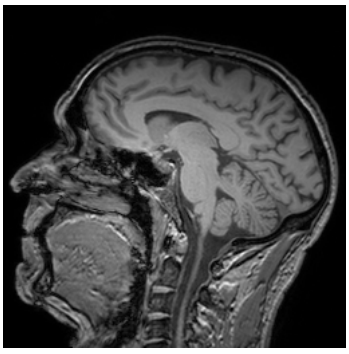

Figure 5. Sagittal MRI scan

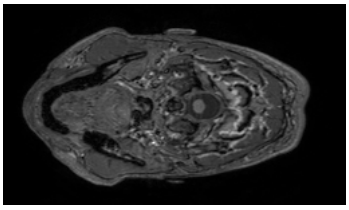

Figure 6. Coronal MRI scan

4. Methodology

4.1. Model in the paper (CNN-IL)

4.1.1 Input/Output of Model

Since ground truth high-resolution images are not readily available, we treat the original MRI scan images (256 x 256 x 150) as the Ground-truth as is done in the paper. Then, we downsample these images by a factor of 2x to (128 x 128 x 75) and then use these as **input** to the proposed CNN-IL model to get super-resoluted 2x HR (High-resolution) **output** images of 256 x 256 x 150.

4.1.2 Model Architecture

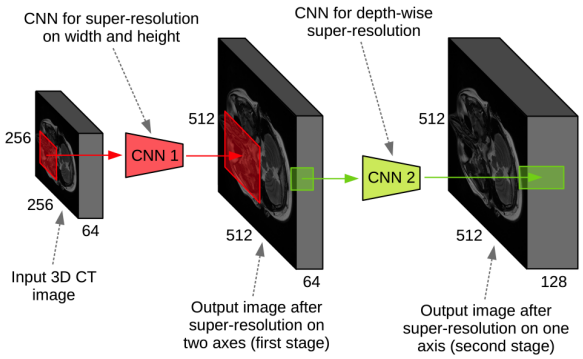

Figure 7. Overall high-level model

The paper follows a 2 step approach as shown in Fig. 7. We first apply a CNN to perform super-resolution in the width and height dimensions. Then, we apply another CNN to perform super-resolution in depth dimension.

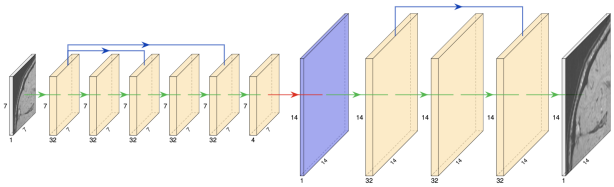

Figure 8. Detailed Model architecture for each CNN in Fig. 7

Our 10 conv layers in each CNN are divided into two blocks (Fig. 8). The first block, formed of the first 6 conv layers, starts with the input of the neural network and ends just before the upscaling layer. Each of the first 5 convolutional layers are formed of 32 filters. For the CNN used in the first stage, the number of filters in the sixth convolutional layer is equal to the square of the scale factor, e.g. for a scale factor of 4× the number of filters is 16. For CNN used in second stage, number of filters in the sixth convolutional layer is equal to the scale factor, e.g. for a scale

factor of 4× the number of filters is 4. The first convolutional block contains a short-skip connection, from the first conv layer to the third conv layer, and a long-skip connection, from the first conv layer to the fifth conv layer.

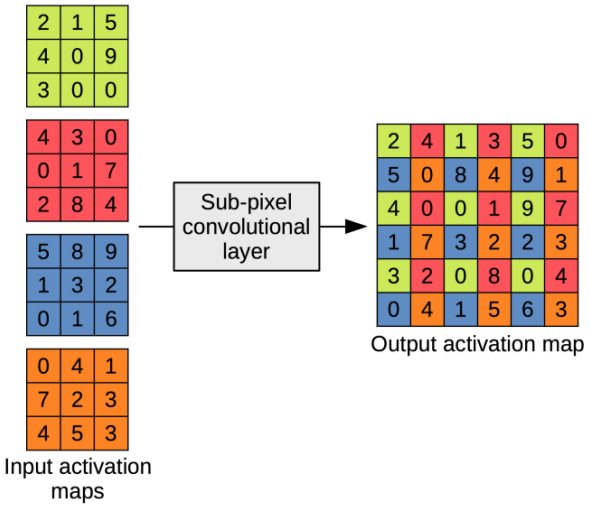

Figure 9. Sub-pixel Convolutional Upsampling layer for 2 axis (width and height)

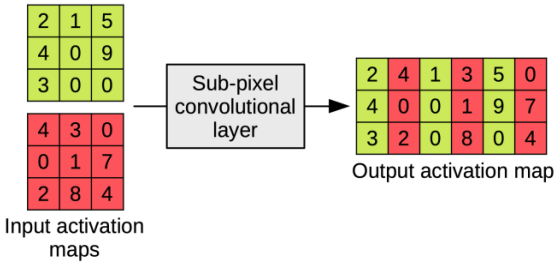

Figure 10. Sub-pixel Convolutional Upsampling layer for 1 axis (depth)

These 6 convolutional layers are followed by an upsampling sub-pixel convolutional layer (Fig. 9 for CNN-1, i.e., height and width and Fig. 10 for CNN-2, i.e., depth) which is followed by 3 more convolutional layers. In these 3 convolutional layers, the 1st layer is connected to 3rd layer by short-skip. The detailed architecture is shown in Fig.

4.2. Our Model (Modifications)

We implemented a simple CNN from scratch in PyTorch for the project. While the original architecture is written in TensorFlow [2], we use PyTorch to write our custom network. For height and width, we use a CNN-based architecture. As we want to preserve depth and height, we set the height and width of the network similar to that of the input patch. We use padding = 'same' for each layer to preserve the original dimensions of the image. We use skip connections between some layers to pass on information. This helps in transferring information from the initial layer to layer after that. We use PyTorch's implementation of sub-pixel convolution to change the channels from 4 to 1 and increase dimensions in both axes. We create a much larger network for better learning. For the activation function, we use an Adam optimizer and a lambda decay rate of 0.9 after every epoch. We also use ELU with default values instead of RELU. For Depth, we feed the network a square 32x32 image patch instead of a rectangular patch of 32x64. We use a smaller network based on the same network as that for height and weight. The only difference is that the sub-pixel convolution is at the final layer of the network. We use ELU and adam optimizer with lambda learning rate decay. We explain the Height and Width Network in figure 10 and the depth net in figure 11.

For the height and width network, the skip connection is between layer 1 and layer 3 , layer 1 and layer 4, layer 4 and layer 7, layer 7 and layer 8, layer 11 and layer 13, and layer 11 and layer 14. For depth network, the skip connections are between layer 1 and layer 3 , layer 1 and layer 4 , layer 4 and layer 7 , layer 7 and layer 8.

```
Net(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU1): ELU(alpha=1.0)
  (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU2): ELU(alpha=1.0)
  (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU3): ELU(alpha=1.0)
  (conv4): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU4): ELU(alpha=1.0)
  (conv5): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU5): ELU(alpha=1.0)
  (conv6): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU6): ELU(alpha=1.0)
  (conv7): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU7): ELU(alpha=1.0)
  (conv8): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU8): ELU(alpha=1.0)
  (conv9): Conv2d(32, 4, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU9): ELU(alpha=1.0)
  (depth2space): PixelShuffle(upscale_factor=2)
  (conv10): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU10): ELU(alpha=1.0)
  (conv11): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU11): ELU(alpha=1.0)
  (conv12): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU12): ELU(alpha=1.0)
  (conv13): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU13): ELU(alpha=1.0)
  (conv14): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU14): ELU(alpha=1.0)
  (conv15): Conv2d(32, 1, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU15): ELU(alpha=1.0)
)
```

Figure 11. Custom Height and Width Network

```
Net(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU1): ELU(alpha=1.0)
  (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU2): ELU(alpha=1.0)
  (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU3): ELU(alpha=1.0)
  (conv4): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU4): ELU(alpha=1.0)
  (conv5): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU5): ELU(alpha=1.0)
  (conv6): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU6): ELU(alpha=1.0)
  (conv7): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU7): ELU(alpha=1.0)
  (conv8): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU8): ELU(alpha=1.0)
  (conv9): Conv2d(32, 4, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (ELU9): ELU(alpha=1.0)
  (depth2space): PixelShuffle(upscale_factor=2)
)
```

Figure 12. Custom Depth Network

4.2.1 Input/Output of CNNs

The input of the height and width network is a 32x32 path from one of the 2D slices of 3D dataset. We enhance the height and width of the patch to 64x64 as our output of the height and width network. We combine these patches to get a 3D slice. We slice these 3D slices in a different way to get depth and other channel 2D patches of size 32x64. We use these patches and reshape them in the shape of 32x32 to feed into our depth network. The final output is 64x64 patches which are joined together to create a 3D enhances superpixel MRI image.

4.2.2 Model Architecture

5. Experiments

We tried to change layer dimensions and the position of the sub-pixel convolution layer and found the optimal location for both locations. We found L1 loss more useful than L2 loss when comparing SSIM and PSNR matrices. We played with different learning rates and found 0.0001 with learning rate decay to give better performance than the learning rates greater than 0.001. Also, higher decay leads to deterioration of performance.

5.1. Evaluation Metrics

The paper uses the following evaluation metrics and we will be using the same.

5.1.1 PSNR

Peak signal-to-noise ratio (PSNR) is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Although the PSNR is one of the most used metrics for image reconstruction, some researchers argued that it is not highly indicative of the perceived similarity.

5.1.2 SSIM

SSIM aims to address this shortcoming by taking contrast, luminance and texture into account. The result of the SSIM is a number between -1 and 1, where a value of 1 means the ground-truth image and the reconstructed image are identical.

5.1.3 IFC

Since PSNR and SSIM values cannot guarantee a visually favorable result, we employ an additional metric for the final results, namely the information fidelity criterion (IFC). As for PSNR and SSIM, higher IFC values indicate better results.

	Height and width model in Paper	Our Height and Width Model
L1 Loss	7.4505	2.9542
SSIM	0.9325	0.9610
PSNR	36.8375	37.5569

Table 1. Comparison of SSIM,PSNR and Losses of our custom height and width network with original [1]height and width network for Training

	Depth model in Paper	Our Depth Model
L1 Loss	5.9	3.2820
SSIM	0.95	0.9614
PSNR	38.31	38.6807

Table 2. Comparison of SSIM,PSNR and Losses of our custom depth network with original [1] depth network for Training

6. Results

6.1. Training Results

We can see from Table 1 that our model gives better SSIM, PSNR and L1 loss values while training on height and width.

We can see from Table 2 that our model gives better SSIM, PSNR and L1 loss values while training on height and width.

We can see that the network outputs match closely with ground truth in both original paper's network [1] and our modified network. (Fig. 13 and Fig. 14)

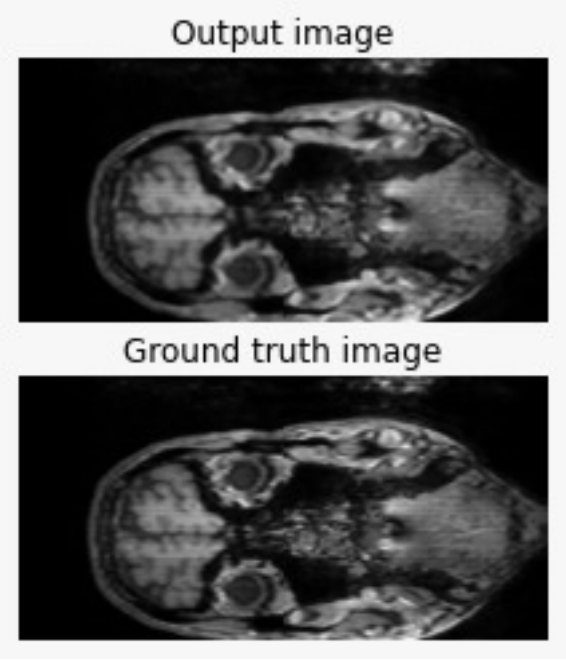

Figure 13. Original Paper's Output for Depth

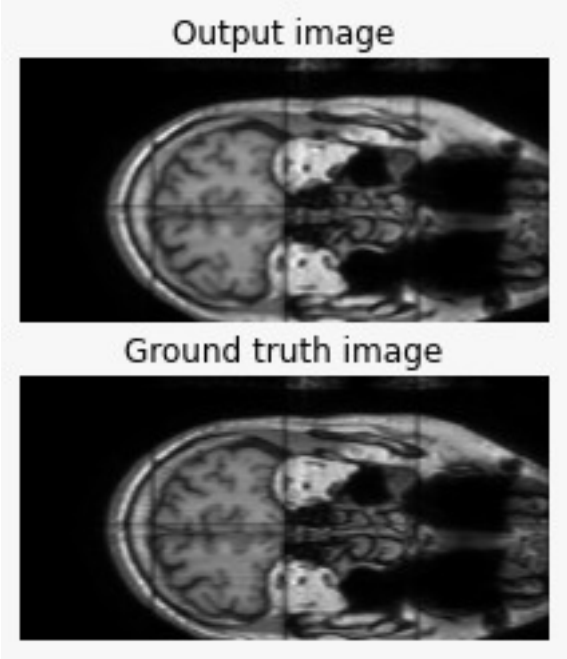

Figure 14. Our Custom Network's Output for Depth

6.2. Testing Results

We observed L1 loss of 3.77, SSIM 0.95, and PSNR 35.98 while testing our custom height and width network.

7. Conclusion

The paper's method [1] and our method is able to reliably upscale 3D CT/MRI image up to a scale factor of 2x. In the paper, they have compared their approach with several baseline interpolation (Lanczos interpolation) and state-of-the-art methods ([7], [12], [13] and [14]). We can conclude that our model performs better than the one given in paper (CNN-IL) based on the results observed. And since their model is better than the various state-of-the-art methods and Lanczos interpolation method, we can conclude that our model also performs better than those methods.

8. Limitation

The model may not work well for higher upscaling factors (greater than 4x). Also, the training has been done for single-channel inputs and may not work for multi-channel inputs.

References

[1] M. I. Georgescu, R. T. Ionescu and N. Verga, "Convolutional Neural Networks With Intermediate Loss for 3D Super-Resolution of CT and MRI Scans," in IEEE Access, vol. 8, pp. 49112-49124, 2020, doi: 10.1109/ACCESS.2020.2980266.

[2] GitHub Link for CNN-IL

[3] http://brain-development.org/ixi-dataset/

[4] https://insight-journal.org/midas/collection/view/190

[5] M. I. Georgescu, R. T. Ionescu, A. I. Miron, O. Savencu, N.C. Ristea, N. Verga and F.S. Khan, "Multimodal Multi-Head Convolutional Attention with Various Kernel Sizes for Medical Image Super-Resolution", arXiv, 2022, doi: 10.48550/arXiv.2204.04218.

[6] Zhang K, Hu H, Philbrick K, Conte GM, Sobek JD, Rouzrokh P, Erickson BJ. SOUP-GAN: Super-Resolution MRI Using Generative Adversarial Networks. Tomography. 2022 Mar 24;8(2):905-919. doi: 10.3390/tomography8020073. PMID: 35448707; PMCID: PMC9027099.

[7] C.H. Pham, C. Tor-Díez, H. Meunier, N. Bednarek, R. Fablet, N. Passat, and F. Rousseau, "Multiscale brain MRI super-resolution using deep 3D convolutional networks," Computerized Medical Imaging and Graphics, vol. 77, no. 101647, 2019.

[8] Molina-Maza JM, Galiana-Bordera A, Jimenez M, Malpica N, Torrado-Carvajal A. Development of a Super-Resolution Scheme for Pediatric Magnetic Resonance Brain Imaging Through Convolutional Neural Networks. Front Neurosci. 2022 Oct 25;16:830143. doi: 10.3389/fnins.2022.830143. PMID: 36389232; PMCID: PMC9641213

[9] Chunwei Tian, Yixuan Yuan, Shichao Zhang, Chia-Wen Lin, Wangmeng Zuo, David Zhang, Image super-resolution with an enhanced group convolutional neural network, Neural Networks, Volume 153, 2022, Pages 373-385, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2022.06.009.

[10] X. Zhao, Y. Zhang, T. Zhang, and X. Zou, "Channel splitting network for single MR image super-resolution," IEEE Transactions on Image Processing, vol. 28, no. 11, pp. 5649–5662, 2019.

[11] H. Yu, D. Liu, H. Shi, H. Yu, Z. Wang, X. Wang, B. Cross, M. Bramler, and T. S. Huang, "Computed tomography super-resolution using convolutional neural networks," in Proceedings of ICIP, 2017, pp. 3944–3948.

[12] C. You, G. Li, Y. Zhang, X. Zhang, H. Shan, M. Li, S. Ju, Z. Zhao, Z. Zhang, W. Cong et al., "CT super-resolution GAN constrained by the identical, residual, and cycle learning ensemble (GAN-CIRCLE)," IEEE Transactions on Medical Imaging, 2019.

[13] K. Zeng, H. Zheng, C. Cai, Y. Yang, K. Zhang, and Z. Chen, "Simultaneous single- and multi-contrast super-resolution for brain MRI images based on a convolutional neural network," Computers in Biology and Medicine, vol. 99, pp. 133–141, 2018.

[14] X. Du and Y. He, "Gradient Guided Convolutional Neural Network for MRI Image Super Resolution", Applied Sciences, vol. 9, no. 22, p. 4874, 2019.