ENHANCING MRI IMAGES USING FAST SUPER RESOLUTION CONVOLUTIONAL NEURAL NETWORK (FSRCNN)

MINI PROJECT REPORT

Submitted to the Department of Computer Applications, Bharathiar University in partial fulfillment of the requirements for the award of the degree of

MASTER OF SCIENCE IN DATA ANALYTICS

Submitted by

KARTHIKEYAN K (22CSEG15)

Under the guidance of

Dr. J. SATHEESHKUMAR, MCA., Ph.D.,

Professor,

Department of Computer Applications

DEPARTMENT OF COMPUTER APPLICATIONS

BHARATHIAR UNIVERSITY

COIMBATORE – 641 046

DECEMBER – 2023

DECLARATION

DECLARATION

I hereby affirm that the project entitled "**ENHANCING MRI IMAGES USING FAST SUPER RESOLUTION CONVOLUTIONAL NEURAL NETWORK (FSRCNN),"** which is being submitted to the Department of Computer Applications, Bharathiar University, Coimbatore, represents the original work conducted by **KARTHIKEYAN K (22CSEG15).** This work was carried out under the supervision and guidance of **Dr. J SATHEESHKUMAR, MCA., Ph.D.,** from the Department of Computer Applications at Bharathiar University, Coimbatore. Furthermore, I confirm that this project has not been used as the basis for the conferral of a Degree, Diploma, Associate Ship, Fellowship, or any similar title to any candidate from any other university prior to this submission.

Place: Coimbatore　　　　　　　　　　　　　　　　Signature of Candidate

Date:　　　　　　　　　　　　　　　　　　　　　**(KARTHIKEYAN K)**

Countersigned by

Dr. J. SATHEESHKUMAR, MCA., Ph.D.,
Professor,
Department of Computer Applications

CERTIFICATION

CERTIFICATE

This is to certify that the project work titled "**ENHANCING MRI IMAGES USING FAST SUPER-RESOLUTION CONVOLUTIONAL NEURAL NETWORK (FSRCNN)**" submitted to the Department of Computer Applications, Bharathiar University, in partial fulfillment of the requirements for the award of a Degree in Master of Science in Data Analytics, is a record of the original work done by **KARTHIKEYAN K (22CSEG15)** under my supervision and guidance. I confirm that this project work has not formed the basis of the award of any Degree/Diploma/Associate Ship/Fellowship or similar title to any candidate of any university.

Place: Coimbatore

Date:

Project Guide **Head of the Department**

Submitted for the University Viva-Voce Examination held on ………………….

Internal Examiner **External Examiner**

ACKNOWLEGDEMENT

ACKNOWLEDGEMENT

I thank the Almighty God who showered His grace to make this project successful. The success and final outcome of this study required a lot of guidance and assistance from many people and I am extremely fortunate to have got those people all along the work and completion of the project.

I express my sincere gratitude to our Professor & Head of the Department
Dr. M PUNITHAVALLI, MCA., Ph.D., Department of Computer Applications, Bharathiar University, Coimbatore for giving me the opportunity to do this project work.

I really deem it a special privilege to professor **Dr. J. SATHEESHKUMAR, MCA., Ph.D.,**, Department of Computer Application, Bharathiar University for valuable guidance and interest in my project work from the beginning to the end.My heartfelt thanks to my dear parents, family, and friends for their support and encouragement toward the successful completion of this project. I am highly obliged to all those who have helped me directly and indirectly in making this project a success.

TABLE OF CONTENTS

S.No.	CONTENT	PAGE NO
	Abstract	1
	CHAPTER - I	
1	Introduction	2
	1.1 Workflow	3
	CHAPTER – II	
2	Literature Review	4
	CHAPTER - III	
3	Methodology	5
	3.1 FSRCNN	5
	CHAPTER - IV	
4	Model Pipeline	6
	4.1 Data Collection	6
	4.2 Image Preprocessing	6
	4.3 Image Transfomations	7
	4.4 Model Training	8
	4.5 Evaluation	14
	CHAPTER - V	
5	Result and Discussion	19
	CHAPTER -VI	
6	Model Deployment	20
	CHAPTER -VII	
7	Conclusion	23
	Bibliography	24

ABSTRACT

This research looks into the use of the Fast Super-Resolution Convolutional Neural Network (FSRCNN) in the context of MRI brain tumor imaging, with the goal of improving the resolution and quality of reconstructed pictures. Although MRI, which stands for Magnetic Resonance Imaging, is an essential diagnostic technique for brain tumors, precise analysis and diagnosis may be restricted by the intrinsic limitations of picture resolution. To tackle this problem, FSRCNN—a deep learning architecture created especially for image super-resolution is used.

The FSRCNN model is trained on a dataset comprising low-resolution MRI brain tumor images, learning intricate features and patterns inherent to high-resolution counterparts. The trained model demonstrates a remarkable capability to enhance image resolution, providing a more detailed and precise representation of tumor structures. Evaluation metrics, including peak signal-to-noise ratio (PSNR) and structural similarity index (SSI), affirm the superior performance of FSRCNN in comparison to traditional interpolation methods.

The proposed FSRCNN-based method has a lot of potential to progress medical picture reconstruction, especially when it comes to MRI brain tumor imaging. FSRCNN's higher resolution has the ability to provide physicians with more precise and comprehensive data for better diagnosis and treatment planning.

CHAPTER – I

INTRODUCTION

Medical image reconstruction plays a crucial role in enhancing the diagnostic accuracy of imaging techniques. In the realm of MRI brain tumor images, the Fast Super-Resolution Convolutional Neural Network (FSRCNN) emerges as a powerful tool for reconstructing high-resolution images from their low-resolution counterparts. FSRCNN leverages deep learning and convolutional neural network architectures to learn complex mappings from low-resolution to high-resolution images.

We implement an FSRCNN model tailored for MRI brain tumor image reconstruction. The model architecture involves a series of convolutional layers for feature extraction, shrinking, non-linear mapping, expanding, and deconvolution. Additionally, an augmentation strategy is employed using the ImageDataGenerator to enhance the model's ability to generalize to diverse data.

Finally, the application of the trained model to predict high-resolution tumor images from new low-resolution inputs. Evaluation metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are utilized to assess the quality of the reconstructed images, providing quantitative insights into the model's performance.

This FSRCNN implementation serves as a valuable tool for medical professionals, aiding in the enhancement of MRI brain tumor images and contributing to more accurate and reliable diagnoses.

1.1 WorkFlow of a FSRCNN Model

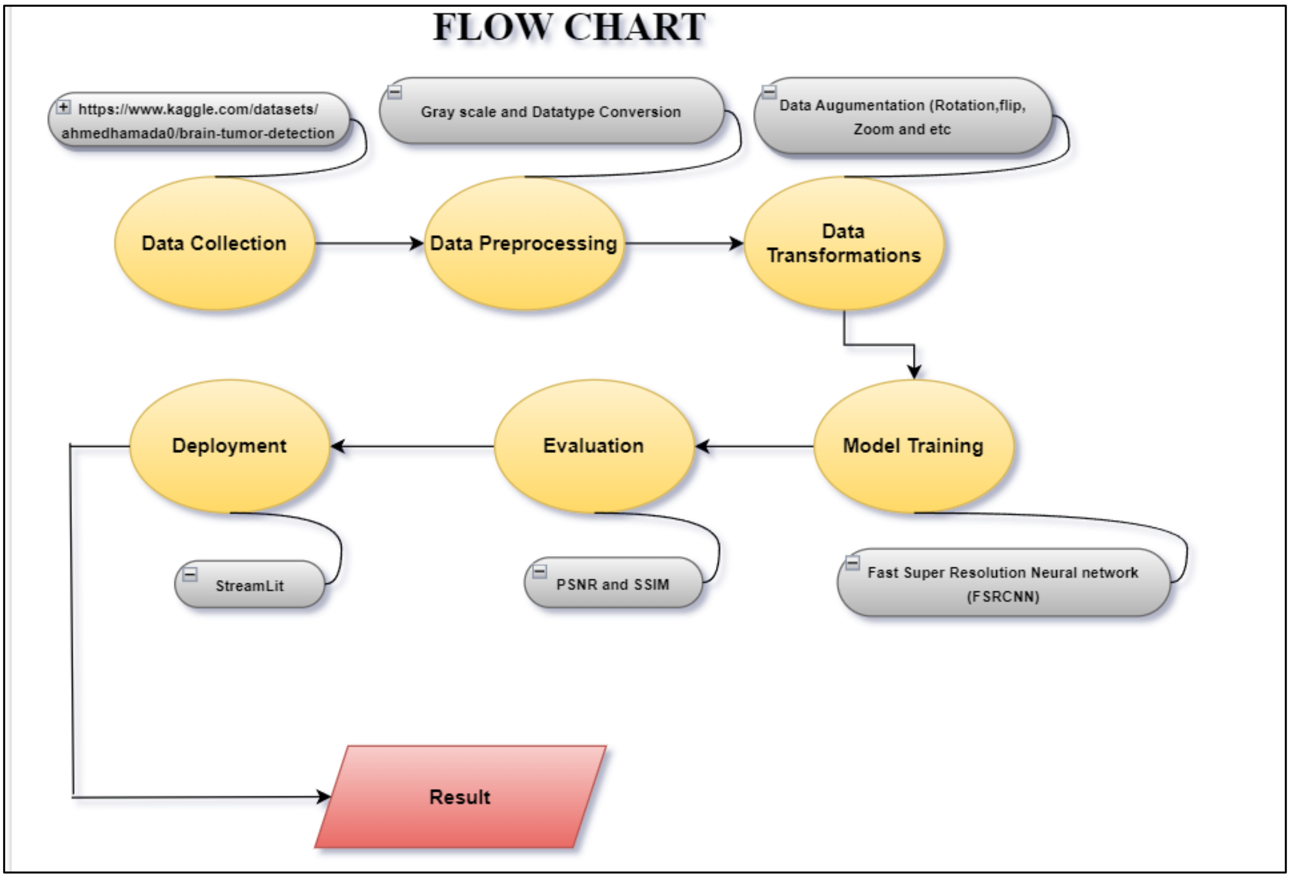


Fig. 1 Work Flow of a FRCNN Model

CHAPTER II

LITERATURE REVIEW

Study introduces SRGAN for image and face super-resolution using a GAN framework. The High and Low-Resolution (HLR) Dataset is employed, and evaluation metrics such as FID and PSNR are used. The Widerface dataset with 182,866 faces is also utilized, achieving FID of 14.89 and PSNR of 19.3.[1]

Focused on medical imaging, this research employs SRGAN for anisotropic super-resolution in prostate MRI. Evaluation is performed on the Prostate-Diagnosis and PROSTATEx datasets, yielding PSNR of 29.51 and SSIM of 0.82 for anisotropic super-resolution, and PSNR of 21.27 and SSIM of 0.66 for SRGAN.[2]

This study applies SRGAN and Dilated Convolutional networks for CT image super-resolution. The DeepLesion dataset, consisting of 10,594 CT scans, is used. The evaluation includes PSNR, SSIM, and MOS, with SRGAN achieving 28.91 PSNR, 0.819 SSIM, and 3.81 MOS.[3]

This research introduces ESRGAN for image and face super-resolution, comparing it with SRCNN. The evaluation is conducted on the Flickr2K dataset, with ESRGAN achieving 20.35 PSNR and 1.98 Perceptual Index, while SRCNN scores 22.73 PSNR and 5.73 Perceptual Index.[4]

Combining wavelet transform with SRGAN for face image super-resolution, this study uses the MUCT Face database. Evaluation metrics include PSNR and SSIM, with SRGAN achieving 27.86 PSNR and 0.81 SSIM, while the wavelet-transform-based method scores 29.91 PSNR and 0.84 SSIM.[5]

Focused on achieving photo-realistic results, this study utilizes SRGAN with various VGG losses. Evaluation on 350,000 images from the ImageNet database yields impressive results, with PSNR of 29.84, SSIM of 0.8468, and MOS of 3.78, indicating high accuracy.[6]

This research applies SRGAN and SRCNN for super-resolution in textile flaw detection. Using 3000 high-quality images, SRGAN achieves PSNR of 21.32 and SSIM of 0.84, while SRCNN scores 24.45 PSNR and 0.842 SSIM.[7]

Focusing on remote sensing, this study uses SRGAN for super-resolution in Terra SAR images. Evaluation metrics include Mean Squared Error (MSE) and SSIM, with SRGAN achieving MSE of 0.0014 and SSIM of 0.9083.[8]

CHAPTER - III

METHODOLOGY

3.1 Fast Super Resolution Convolutional Neural Network (FSRCNN)

Fast Super-Resolution Convolutional Neural Network (FSRCNN) is a deep learning model designed for the task of image super-resolution. Super-resolution is the process of enhancing the resolution of an image, generating a higher-resolution version from a lower-resolution input. This can be particularly useful in various applications, such as upscaling images for better visual quality or improving the resolution of images in medical imaging.

The breakdown of the key components and steps involved in **FSRCNN** are

Architecture

Feature Extraction: The network begins with a series of convolutional layers that extract hierarchical features from the low-resolution input image. These layers capture different levels of abstraction in the input.

Non-linear Mapping: After feature extraction, a non-linear mapping layer is applied to enhance the representation of features, allowing the network to learn more complex relationships in the data.

Deconvolution (Transposed Convolution): The network then utilizes deconvolution layers (transposed convolution or fractionally strided convolution) to increase the spatial resolution of the features.

Reconstruction: Finally, the reconstructed high-resolution image is obtained through additional convolutional layers.

Training

FSRCNN is trained using pairs of high-resolution and low-resolution images. The network learns to map the low-resolution images to their corresponding high-resolution counterparts during the training process. The training involves minimizing a loss function that measures the difference between the predicted high-resolution images and the ground truth high-resolution images.

CHAPTER – IV
MODEL PIPELINE

4.1 Data Collection

To acquire MRI image data of brain tumors from online repositories like UCI, medical imaging repositories, and Kaggle, the initial step is to identify a reliable dataset suitable for the Super Resolution task. This is crucial because deep learning architectures necessitate a substantial number of parameters or features for effective image reconstruction

The selected image data set contains approximately 500 photos illustrating MRI-positive cases of brain tumors. Even though they are essentially black and white when viewed visually, these tumor photos should have random forms and be represented in RGB. It's important to note that the photos have the structure (x, y, 3), suggesting that they have three color channels. Despite their RGB format, they appear in black and white in a visual environment. This disparity highlights the necessity of understanding the data format as well as pre-processing requirements, such as transforming the visual representation to black and white.

4.2 Image Pre-Processing

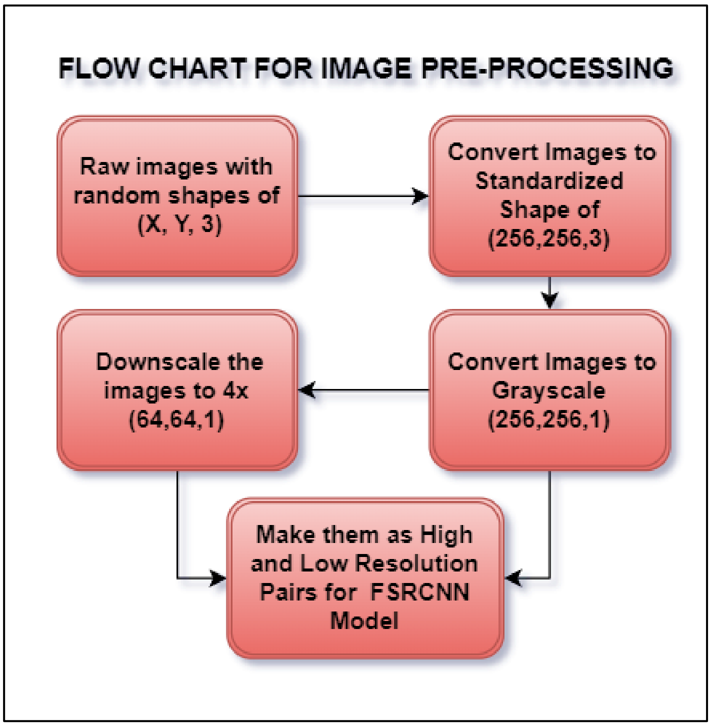


Fig 2 Flow Chart For Image Pre-Processing

To prepare high-resolution and Low-resolution brain tumor MRI images for training the FSRCNN (Fast Super-Resolution Convolutional Neural Network) model. The initial images are of random shapes (x, y, 3), and the goal is to standardize them into a consistent format, resize to (256, 256, 3), convert to grayscale (256,256,1) and then generate low-resolution counterparts for subsequent model training.

The first preprocessing step involves resizing the randomly shaped MRI images to a fixed dimension of (256, 256, 3). Standardizing the size facilitates consistent input for downstream processing and model training.

Subsequently, the standardized high-resolution images are downsampled by a factor of 4x to generate corresponding low-resolution counterparts with dimensions (64, 64, 3). This mimics the lower quality input that the FSRCNN model will be trained to enhance during the super-resolution

Following resizing, the High and low resolution images are converted to grayscale, reducing the channel dimension to 1 and simplifying the data for computational efficiency. This step ensures uniformity in image representation and prepares the data for the subsequent steps. The resulting dataset now comprises 500 pairs, each consisting of a high-resolution image (256, 256, 1) and its corresponding low-resolution counterpart (64, 64, 1). These pairs serve as the training and Testing data for the FSRCNN model. **(Fig 2)**

4.3 Image Transformation

The Image generator is utilized for data augmentation, a technique employed to prevent overfitting in machine learning models, especially when dealing with image datasets. Overfitting occurs when a model learns to perform well on the training data but fails to generalize effectively to new, unseen data. Data augmentation helps address this issue by applying various transformations to the input images during the training process, thereby diversifying the dataset and exposing the model to different perspectives of the same objects.

The ImageDataGenerator class typically allows for a range of image transformations, such as rotation, scaling, shearing, and flipping. These transformations simulate real-world variations in the input data and enable the model to become generalize better to unseen examples. By generating augmented images on-the-fly during training, the model becomes less prone to memorizing specific features of the training set, resulting in improved performance on new, unseen data.

4.4 Model Training

4.4.1 FSRCNN Architecture Requirements

To improve the resolution of images, a deep learning architecture utilizing convolutional layers is employed. Below Mentioned parameters of the convolutional layers play a crucial role in enhancing the image resolution effectively.

Conv2D

Conv2D stands for Convolutional 2D, which is a type of layer used in convolutional neural networks (CNNs). It performs a 2-dimensional convolution operation on the input, which is commonly used for image processing tasks. Convolution involves sliding a various filter (also known as a kernel) for differnet layers over the input data to extract features.

Activation (PRELU, ReLU)

Activation functions introduce non-linearities to the neural network, enabling it to learn complex patterns. ReLU (Rectified Linear Unit) and PReLU (Parametric Rectified Linear Unit) are activation functions. ReLU sets all negative values to zero, while PReLU allows a small negative

slope for negative values, which can sometimes help in training deep networks.

Padding (same)

Padding is the process of adding extra pixels around the input data before applying a convolution operation. “Same” padding means that the input is padded in such a way that the output has the same height and width as the input. This is often done to prevent the spatial dimensions from shrinking too quickly during convolutional operations.

Kernel Size

The kernel size refers to the dimensions of the convolutional filter. In Conv2D layers, the kernel is a small matrix that slides over the input data. The kernel size determines the receptive field of the convolutional operation and influences the types of patterns the layer can learn, For our model the usage kernal size varies from layer to layer

Strides

Strides determine the step size at which the convolutional filter moves across the input data. A larger stride reduces the spatial dimensions of the output, while a smaller stride preserves more information. Strides influence the spatial resolution of the learned features.

Conv2DTranspose

Conv2DTranspose is used for upsampling in neural networks. It is the opposite of Conv2D. Instead of reducing the spatial dimensions of the input, it increases them. This is commonly used in tasks like image segmentation and generating high-resolution images.

Input()

In the context of neural networks, Input() is a function or layer that defines the shape of the input data that will be fed into the model. The Shape of a Low resolution (64,64,1) of a image is act as a input of the layer

Dropout

Dropout is a regularization technique used to prevent overfitting in neural networks. During training, randomly selected neurons are ignored (dropped out) at each update, forcing the network to learn more robust features. It helps prevent the model from relying too heavily on specific neurons.

4.4.2 FSRCNN ARCHITECTURE

```
Model: "FSRCNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64, 64, 1)]       0

 conv2d (Conv2D)             (None, 64, 64, 256)       6656

 p_re_lu (PReLU)             (None, 64, 64, 256)       1048576

 conv2d_1 (Conv2D)           (None, 64, 64, 128)       32896

 p_re_lu_1 (PReLU)           (None, 64, 64, 128)       524288

 dropout (Dropout)           (None, 64, 64, 128)       0

 conv2d_2 (Conv2D)           (None, 64, 64, 256)       295168

 p_re_lu_2 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_3 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_3 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_4 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_4 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_5 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_5 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_6 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_6 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_7 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_7 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_8 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_8 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_9 (Conv2D)           (None, 64, 64, 256)       590080

 p_re_lu_9 (PReLU)           (None, 64, 64, 256)       1048576

 conv2d_10 (Conv2D)          (None, 64, 64, 256)       590080

 p_re_lu_10 (PReLU)          (None, 64, 64, 256)       1048576

 conv2d_11 (Conv2D)          (None, 64, 64, 256)       590080

 p_re_lu_11 (PReLU)          (None, 64, 64, 256)       1048576

 conv2d_12 (Conv2D)          (None, 64, 64, 256)       65792

 p_re_lu_12 (PReLU)          (None, 64, 64, 256)       1048576

 conv2d_transpose (Conv2DTra  (None, 256, 256, 1)      20737
 nspose)

=================================================================
Total params: 18,839,169
Trainable params: 18,839,169
Non-trainable params: 0
_________________________________________________________________
```

Fig 3 FSRCNN Model Architecture

4.4.3 Model Compiling And Training

Batch Size

The batch size is the number of training examples used in a single iteration.The batch size determines how many examples are used in each iteration.Larger batch sizes can speed up training by utilizing parallelism in modern hardware (such as GPUs), but they may require more memory.Smaller batch sizes may have a regularizing effect and require less memory, but training may take longer. We use a Batch size of 16 in our model.

Epochs

An epoch is one full pass through the training dataset.The model sees every example in the training set once during an epoch.The process of training a neural network involves minimizing a loss function over multiple epochs.The model's weights are updated at each epoch based on the gradients of the loss function with respect to the model parameters.The number of epochs is a hyperparameter that you specify before beginning training. It denotes how many times the learning algorithm will go through the entire training dataset. We use an epoch size of 50 in our model.

compile()

used to configure the learning process of the model. It requires specifying an optimizer and a loss function, and optionally, metrics to monitor during training and evaluation.

Adam Optimizer

Adam is an optimization algorithm commonly used for training neural networks. It adapts the learning rates of individual parameters by computing adaptive learning rates for each parameter. By passing Adam() without any arguments, the default learning rate is used.

steps_per_epoch

This variable represents the number of steps (batches) to be processed in each epoch. It is calculated based on the length of your low-resolution images dataset divided by the batch size. It determines how many batches are processed in each epoch.

augmented_data_gen

This variable seems to be a generator that yields batches of augmented data. It is likely created using a custom generator function paired_data_generator, which takes low-resolution (lr_images) and high-resolution (hr_images) image pairs, along with the batch size. Data augmentation is a technique to artificially increase the size of the training dataset by applying random transformations to the input data, which can help improve model generalization.

ReduceLROnPlateau

This callback monitors the training loss and reduces the learning rate when a metric has stopped improving. In this case, it monitors the loss, reduces the learning rate by a factor of 0.3 if no improvement is seen for two consecutive epochs, and has a minimum delta of 0.001 to qualify as an improvement.

ModelCheckpoint

This callback saves the model's weights to the specified path whenever the monitored metric (loss, in this case) improves. The save_best_only True parameter ensures that only the best model is saved.

model.fit(...)

This is Point where the actual training takes place. The fit method is called on your model (**FSRCNN**) with the generator (augmented_data_gen). The training is performed for a specified number of epochs (50). The callbacks parameter is used to pass a list of callbacks, including the learning rate reduction and model checkpoint callbacks.

Loss

This parameter defines the loss function to be used during training. In this case, the mean squared error (MSE) is selected as the loss function. MSE is often used for regression problems, where the goal is to minimize the squared difference between the predicted Image pixels and the actual target Image Pixels (**Fig 4-6**).

In the process of training a Fast Super-Resolution Convolutional Neural Network (FSRCNN) model architecture, the choice of activation function plays a crucial role in shaping the model's capacity to learn intricate features from input data. Three distinct activation functions ReLU (Rectified Linear Unit), PReLU (Parametric Rectified Linear Unit), and a combination of both offer diverse avenues for exploring non-linear transformations. The hybrid approach of combining ReLU and PReLU aims to harness the strengths of both functions, allowing the model to capture subtle features effectively. By experimenting with these activation functions during training, uncover insights into the trade-offs between model expressiveness and computational efficiency, facilitating the fine-tuning of the FSRCNN architecture for optimal super-resolution performance.

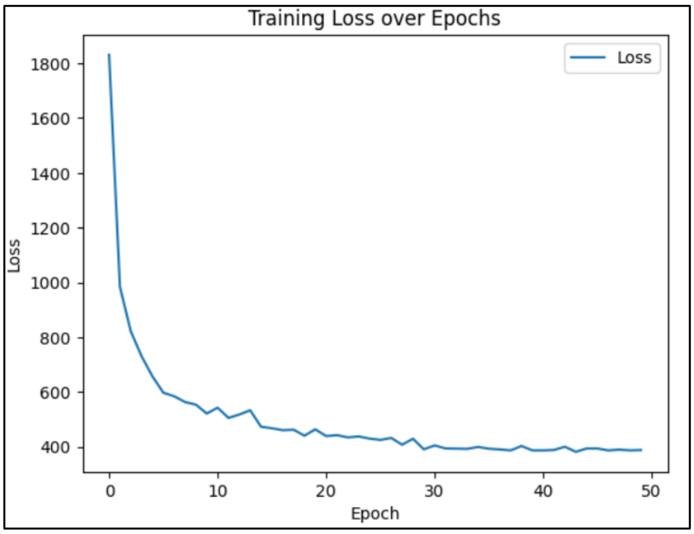


Fig 4 Training Loss For FSRCNN PReLU

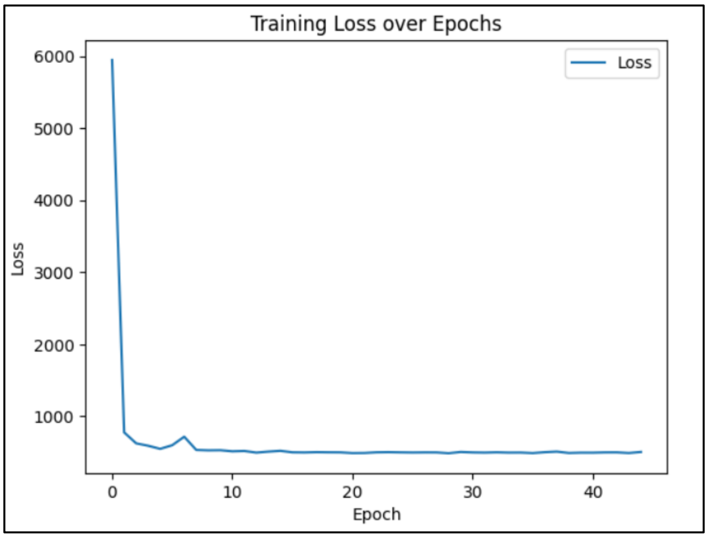


Fig 5 Training Loss For FSRCNN ReLU

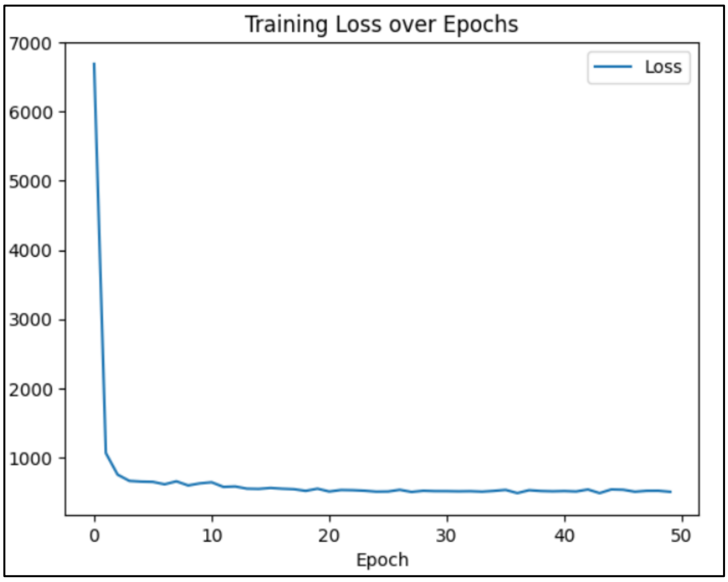


Fig 6 Training Loss For FSRCNN with ReLU and PReLU

4.5 Evaluation

4.5.1 Evaluation Metrics

The FSRCNN (Fast Super-Resolution Convolutional Neural Network) is a deep learning model designed for image super-resolution tasks. When evaluating the performance of FSRCNN, several metrics are commonly used to assess the quality of the generated high-resolution images compared to the ground truth.

Here are three commonly used metrics:

1. Mean Squared Error (MSE)
2. Peak Signal-to-Noise Ratio (PSNR)
3. Structural Similarity Index (SSIM)

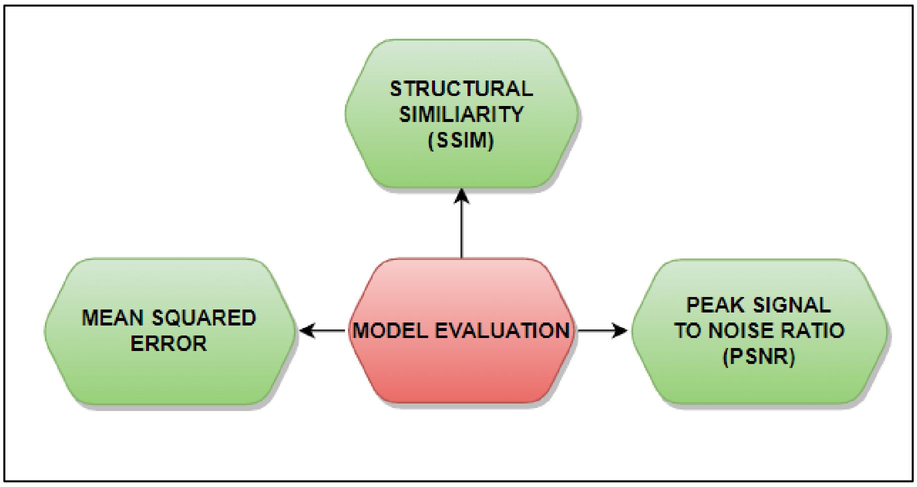


Fig 7 Evaluation Metrices

4.5.1.1 Mean Squared Error (MSE)

MSE measures the average squared difference between the pixels of the super-resolved image and the ground truth. Lower MSE values indicate better performance.

$$\mathbf{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(I_{SR}(i) - I_{HR}(i)\right)^2$$

Where,

N is the total number of pixels

I_{SR} is the pixel value of the super-resolved image (Predicted Image)

I_{HR} is the pixel value of the high-resolution ground truth image.

The formula involves calculating the squared difference between each corresponding pixel in the super-resolved (I_{SR}) and high-resolution (I_{HR}) images.The average of these squared differences is taken, providing a measure of how much the pixel values deviate on average. MSE is a straightforward metric, but it tends to be sensitive to outliers. A large error in a single pixel can significantly impact the overall MSE.

When predicting high-resolution images using FSRCNN, the pixel values often extend beyond the standard 0-255 range. To address this, a crucial preprocessing step involves scaling through normalization. This is achieved by applying the img_as_ubyte method, ensuring that the pixel values are appropriately rescaled to fit within the 0-255 range, vital for accurate and visually reasonable high-resolution image generation

$$\boldsymbol{Normalized\ image} = \frac{img() - img.\min()}{img.\max() - img.\min()}$$

Where,

img is current pixel value,

img.min() is minimum Pixel value of an image,

img.max() is maximum pixel value of an image.

4.5.1.2 Peak Signal-to-Noise Ratio (PSNR)

PSNR is a widely used metric that measures the ratio of the maximum possible power of a signal to the power of corrupting noise. It's expressed in decibels (dB), and higher PSNR values indicate better image quality.

$$\boldsymbol{PSNR} = 20.log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right)$$

Where,

MAX is the maximum possible pixel value (255 for an 8-bit image)

MSE is the mean squared error of an predicted and actual image pixel values

4.5.1.3 Structural Similarity Index (SSIM)

SSIM compares the structural information of the super-resolved image to the ground truth. It takes into account luminance, contrast, and structure. SSIM values range from -1 to 1, where 1 indicates perfect similarity. SSIM goes beyond pixel-wise differences and considers perceptual aspects of the images. It involves the mean (μ) and standard deviation (σ) of pixel values, as well as constants (C1 and C2) to stabilize the division.

SSIM is particularly useful when the goal is to evaluate the perceptual similarity between images. It's more aligned with human visual perception

$$SSIM\ (I_{SR}, I_{HR}) = \frac{(2_{\mu SR \mu HR} + C_1)(2_{\sigma SRHR} + C_2)}{(\mu_{SR}^2 + \mu_{HR}^2 + C_1)(\sigma_{SR}^2 + \sigma_{HR}^2 + C_2)}$$

Where,

$\boldsymbol{I_{SR}}$ is the pixel value of the super-resolved image (Predicted Image),

$\boldsymbol{I_{HR}}$ is the pixel value of the high-resolution ground truth image,

$\boldsymbol{\mu SR}$ is the pixel sample mean Value of SR,

$\boldsymbol{\mu HR}$ is the pixel sample mean Value of HR,

$\boldsymbol{\sigma^2 SR}$ is the variance of SR,

$\boldsymbol{\sigma^2 SR}$ is the variance of HR,

$\boldsymbol{\sigma SRHR}$ is the covariance of SR and HR,

C1 and C2 are constants to avoid division by zero errors.

4.5.2 Evaluation

Evaluate a image reconstruction metrices by Predicting with different activation models for a new High resolution Image from any low resolution image **(Fig 8)** and Evaluate with the Existing High resolution images of different **(Fig 9)** and we Obtained metrics values for image assesment assessment.**(Fig 10)**

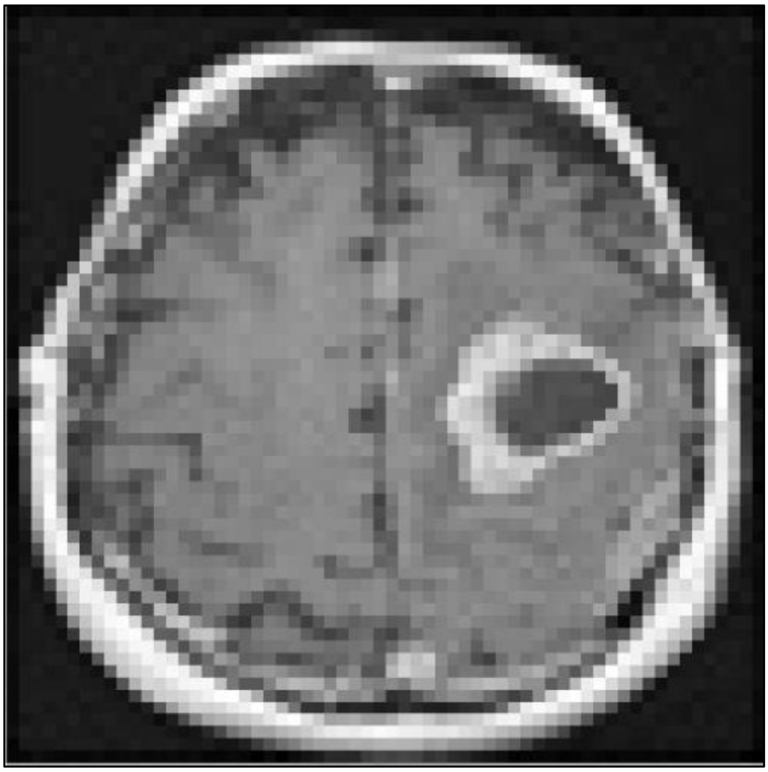
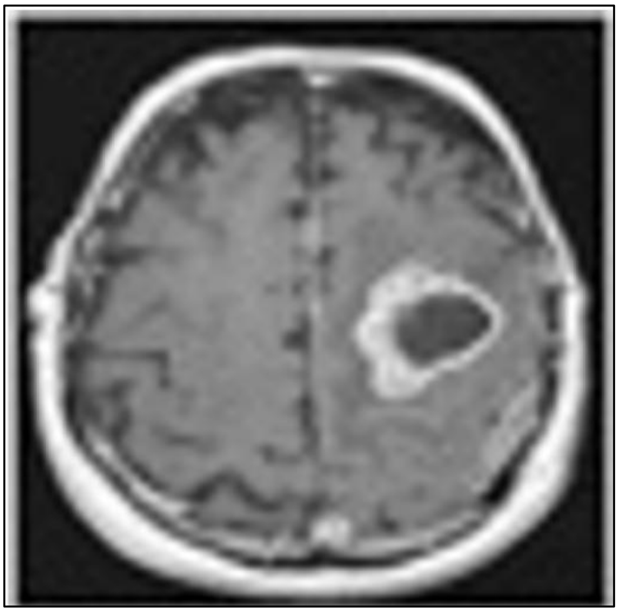
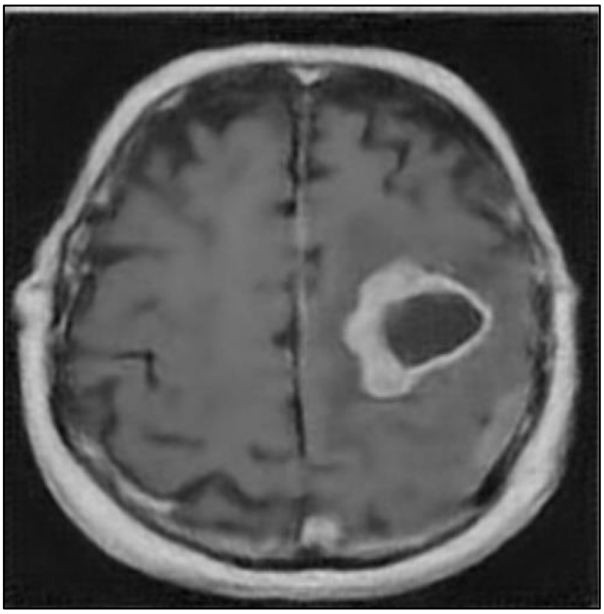


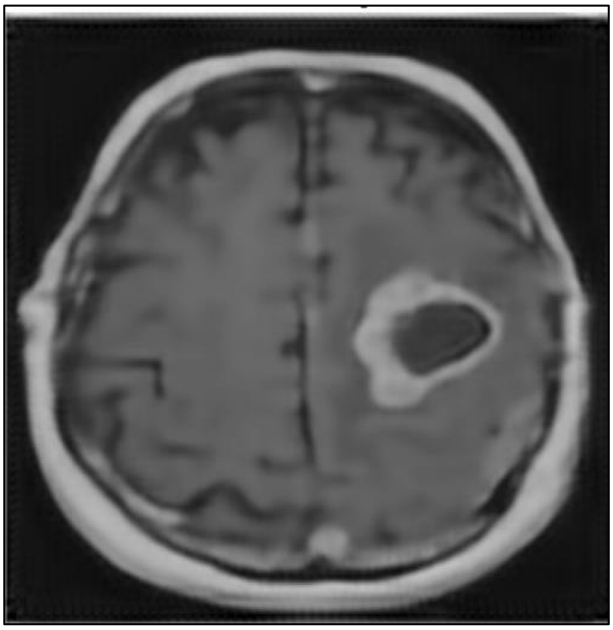
Fig 8 Low Resolution Image



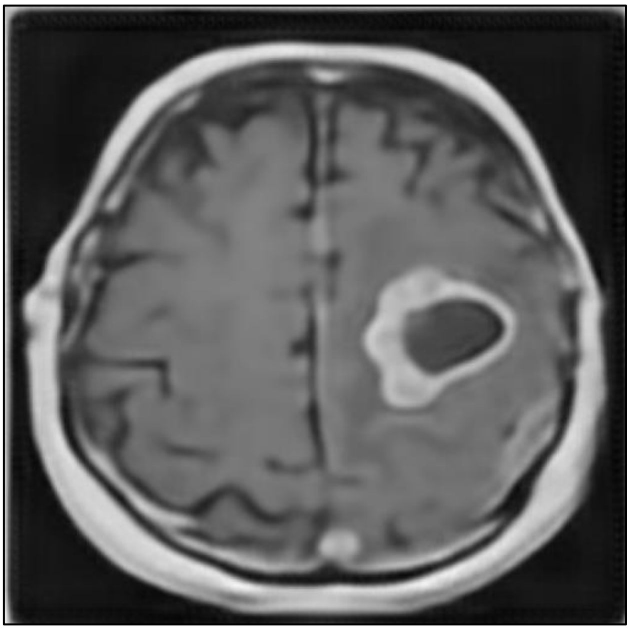
Bicubic Interpolation



FSRCNN PReLU



FSRCNN ReLU



FSRCNN with ReLU and PReLU

Fig 9 New Predicted Images

```
1/1 [==============================] - 1s 960ms/step
MSE value: 90.66
PSNR value: 28.56 dB
SSIM: 0.81
```

FSRCNN PReLU

```
1/1 [==============================] - 0s 480ms/step
MSE value: 101.002
PSNR value: 28.088 dB
SSIM: 0.80
```

FSRCNN ReLU

```
1/1 [==============================] - 1s 524ms/step
MSE value: 90.663
PSNR value: 28.557 dB
SSIM: 0.82
```

FSRCNN with ReLU and PReLU

Fig 10 Evaluation of Predicted images

CHAPTER – V

RESULT AND DISCUSSION

Model	Metrics	PSNR	SSIM	MSE
FSRCNN PReLU		28.56	0.81	90.66
FSRCNN ReLU		28.08	0.8	101.02
FSRCNN with ReLU and PReLU		28.557	0.82	90.663

FSRCNN PReLU:

The Mean Squared Error (MSE) of 90.66 is relatively high, indicating a considerable average discrepancy between the original and reconstructed images. The PSNR of 28.56 dB is decent, suggesting considerable image quality. The Structural Similarity Index (SSIM) of 0.81 is relatively good, indicating well-preserved structural information in the reconstructed image.

FSRCNN ReLU:

The Mean Squared Error (MSE) of 101.02 is relatively high, indicating a significant average error between the original and reconstructed images. The PSNR of 28.08 dB suggests decent image quality, and the Structural Similarity Index (SSIM) of 0.80 indicates reasonably well-preserved structural information.

FSRCNN Hybrid:

The Mean Squared Error (MSE) of 90.663 is relatively high, suggesting a considerable average discrepancy. The PSNR of 28.557 dB indicates decent image quality, and the Structural Similarity Index (SSIM) of 0.82 suggests well-preserved structural information in the reconstructed image.

CHAPTER – VI
MODEL DEPLOYMENT

Machine learning deployment is the process of deploying a machine learning model in a live environment. The model can be deployed across a range of different environments and will often be integrated with apps through an API. Deployment is a key step in an organization gaining operational value from machine learning. However, deploying from a local environment to a real-world application can be complex. Models may need specific infrastructure and will require close monitoring to ensure ongoing effectiveness. For this reason, ML deployment must be properly managed.

There are various sources to deploy the model that we developed; some of the easiest deployment sources from Python are Streamlit, Flask API, etc. ML models will usually be deployed in an offline or local environment, so they will need to be deployed to be used with live data. A data scientist may create many different models, some of which never make it to the deployment stage. Deploying these models can be very resource-intensive. Fig. 16 shows some of the backend processes of the API's work.

Stramlit:

Streamlit is an open-source app framework for Machine Learning and Data Science teams. It helps Data Scientists and ML engineers build beautiful web apps for their models quickly and easily, using only Python. Streamlit is a Python library that makes it easy to build interactive data science web applications. It has a simple and intuitive API, and it is compatible with many popular Python libraries, such as scikit-learn, Keras, and NumPy.Streamlit can be used to build a variety of data science web applications, such as:

- Exploratory data analysis (EDA): dashboards
- Machine learning model training and evaluation: tools
- Data visualization: tools
- Interactive reports

Streamlit is a powerful tool for data scientists and ML engineers who want to build interactive data science web applications quickly and easily. It is also a great tool for teaching data science and machine learning concepts. The **Fig 11** shows the Rough Model Deployment using Streamlit

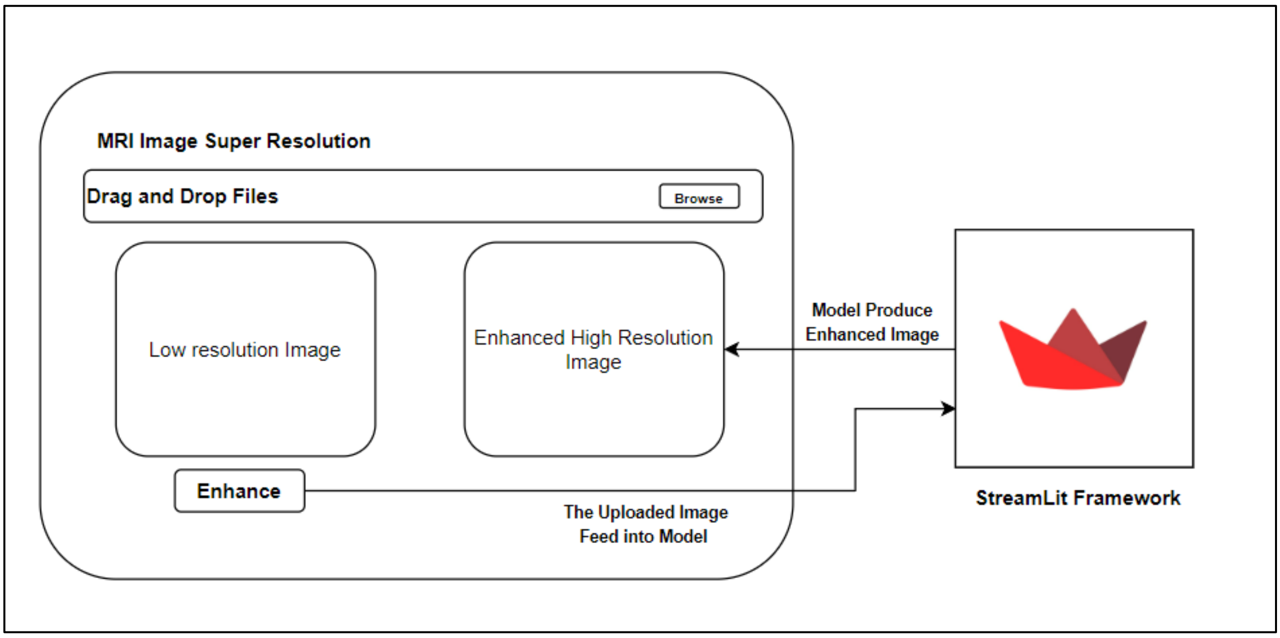


Fig 11 Rough Model Deployment using Streamlit

Python code is for a web application using Streamlit, designed to enhance the resolution of MRI images. Users can upload a low-resolution MRI image through the web interface. The application then displays the input image on the left side of the page. When users click the "Enhance" button, the code processes the uploaded image using a pre-trained deep learning model specifically designed for this task. The enhanced, high-resolution version of the MRI image is then displayed on the right side of the page. The application utilizes the Streamlit library for creating a user-friendly web interface, and the model for image enhancement is loaded using the Keras library. To use this application, you need to run the app.py file locally, and it will provide a link **(Fig 12)** to access the web app in our browser.

And the **Fig 13** and **Fig 14** shows the deployment.

```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.16.202.142:8501
```

Fig 12 Local Host Deployment link

.

Fig 13 Uploading Low Resolution MRI image

Fig 14 Enhanced Low Resolution image

CHAPTER -VII

CONCLUSION

In conclusion, the FSRCNN project focused on enhancing MRI image resolution through the implementation of a convolutional neural network trained on 500 augmented images. the FSRCNN Hybrid variant and shows slightly better performance in terms of PSNR and SSIM compared to the other variants, But the FSRCNN PRelu Variant shows indicating better image quality and preservation of structural information in viewers point of view.

However, it is crucial to acknowledge that the limitations of this project stem from the relatively small size of the training dataset and the inherent challenges posed by the low pixel size of the input images (64x64x1), compared to the higher resolution target images (256x256x1). The constrained number of training samples and smaller image shape leading to less no of feature extraction and subsequent image reconstruction.

Moving forward, expanding the training dataset and considering larger input image sizes could potentially address these limitations, enabling the model to capture more intricate features and nuances in the data. Additionally, optimizing hyperparameters, such as learning rates, could contribute to enhanced performance. Despite the current constraints, the project lays a foundation for future improvements and highlights the importance of addressing dataset size and pixel resolution for more robust and accurate image super-resolution tasks in medical imaging.

BIBLIOGRAPHY

[1] Adrian Bulat, Jing Yang, Georgios Tzimiropoulos. (2018). To learn image super-resolution, use a GAN to learn how to do image degradation.

[2] Rewa Sood, Mirabela Rus. (2019). Anisotropic Super Resolution in Prostate MRI Using Super Resolution Generative Adversarial Networks.

[3] Xuhao Jiang, Yifei Xu, Pingping Wei, Zhuming Zhou. (2020). CT Image Super Resolution Based on Improved SRGAN.

[4] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, Chen Change Loy. (2021). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks

[5] Minjie Cao, Zhe Liu, Xueting Huang, Zhuoxuan Shen. (2021). Research for Face Image Super-Resolution Reconstruction Based on Wavelet Transform and SRGAN.

[6] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

[7] Junchao Li, Liming Wu, Shiman Wang, Wenhao Wu. (2019). Super Resolution Image Reconstruction of Textile Based on SRGAN.

[8] Longgang Wang, Mana Zheng, Wenbo Du, Menglin Wei. (2019). Super-resolution SAR Image Reconstruction via Generative Adversarial Network.