Deep Optimized Super-Resolution with Generative Adversarial Networks (DOSRGAN)

S.M. Faiaz Mursalin
Computer Science Department
Missouri State University
Springfield, Missouri, USA
sm943s@missouristate.edu

Igor Tisma
Computer Science Department
Missouri State University
Springfield, Missouri, USA
it669s@missouristate.edu

***Abstract*—Single-image super-resolution (SISR) is a computer vision task that reconstructs a super-resolution (SR) image from a low-resolution (LR) image. It could be used in a variety of applications such as medical imaging, security, and surveillance imaging. In this work, we are proposing a novel deep learning framework that focuses on perceptual quality, structural fidelity, and adversarial realism. Our approach integrates feature extraction with enhanced residual blocks featuring channel expansion and dual pooling, refined channel attention, and a 3-layer MLP design. Multi-scale feature fusion and dense connections with local and global skip connections strengthen the generator-discriminator interaction. Progressive upsampling is guided by a three-phase training strategy: pixel loss (0–50 %), pixel + content loss (50–80%), and minimal GAN loss (80–100%). Evaluated on Set5 and Set14, our model achieves PSNR scores of at least 34 for 2× scaling and 29 for 4× scaling, demonstrating superior perceptual quality and quantitative performance through adversarial refinement and adaptive loss optimization.**

***Index Terms*—CNN, super-resolution, GAN, DL, ML**

I. Introduction

In this work, we propose an optimized single-image super-resolution framework that incorporates residual blocks, channel attention mechanisms, and a progressive training strategy. The generator network focusses on to extract features and super resolved image reconstruction which is later supported by a lightweight discriminator network that enhances and refines the perceptual and adverserial realism. The training follows in three phases pixel-based accuracy, textre recovery via content loss and on the final phase including the adverserial learning.

We evaluated our model on the widely used Set5 and Set14 datasets and later compared the metrics with baseline models. Our findings highlights the capability of our model to enhance image sharpness while having robust performance on standard benchmarks.

This work advances the field of image super-resolution by presenting a refined model design which is deep enough to catch all the reatures and effective strategy while training along with balancing multiple loss function for superior results.

II. Data Preparation

This section talks about details about the data the we are using, and how we prepared it to best fit our model

A. Training Data

For training the model we used the General100 dataset. This dataset contains 100 high resolution images. To make our model more robust, we included various techniques for data manipulation. First, we did patch extraction. We made patches from these high resolution images, by using a sliding window. All patches were 96x96 in pixel size with a stride value of 48. This enabled us not to have overlapping regions. Methods that we used for creating low resolution images are Gaussian blur and downscale. The Gaussian blur is used to avoid distortion during downscaling. Downscaling is done by bicubic interpolation by using two scale factors (x2 and x4). Another method that we used is data augmentation. By using it, we had more generalization for training our model. We used rotations, vertical and horizontal flips.

B. Test and Validation Data

We used widely used set5 and set14 datasets for evaluation and validation. We did preprocessing on this data as well. Methods like Gaussian blur and downscale are used to generate low resolution images.

III. Methodology

This section talks about details about the methodology in details that is used by the proposed model.

A. Feature Extraction

For feature extraction, we used the first 35 layers of the feature extraction module of pre trained VGG19 model. The purpose of this part is to capture low level features of the image. We froze the parameters for this part. That allows us to keep them static during the network training. It also makes the network faster, because of the lower number of computations. Input is tensor, and output is feature map.

B. Residual Blocks

The residual block consists of three convolutional layers, three batch normalizations, and two Leaky ReLU activation functions. Number of filters is 64. The Figure 1 is the representation of it.

1) First Convolution: This convolution consists of a 3x3 kernel size, padding is set to 1, and there are 64 input channels and 128 output channels. It is doing initial preprocessing.

2) Second Convolution: This convolution consists of a 3x3 kernel size, padding is set to 1, and there are 128 input channels and 128 output channels. It is enhancing features.

3) Third Convolution: This convolution consists of a 3x3 kernel size, padding is set to 1, and there are 128 input channels and 64 output channels. It is reducing the channels.

4) Batch Normalization: We apply batch normalization, after each convolution layer. It normalizes feature maps during the training.

5) Activation Functions: The activation function that we used is a Leaky ReLU. The activation function is applied after first and second convolution layer, and it introduces non linearity.

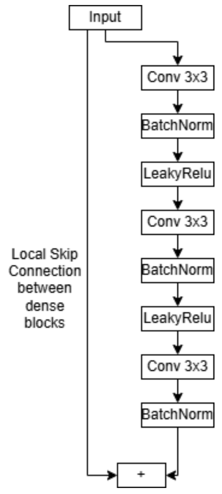


Fig. 1: Residual Block

C. Channel Attention with dual pooling

The Channel Attention enhances and changes the importance of different channels in a feature map. It is doing it by learning weights through training. It consists of average pooling, max pooling, and a fully connected network.

1) Adaptive Pooling Layers: We used two different pooling layers for feature maps. One is average pooling, and the other one is max pooling. They are producing a feature map with shape [batch size, channels, 1, 1]. By reducing spatial dimensions, we are capturing global information about the channels, and important features.

2) Fully Connected Layers: There are three fully connected layers. The first one is reducing the dimensionality from $2c$ to $\frac{c}{\text{reduction}}$. Second one keeps it at $\frac{c}{\text{reduction}}$. This allows for less computation and faster training. The final fully connected layer is there to restore the dimensionality to the original number of channels.

3) Activation Functions: For the introduction of non linearity the Leaky ReLU is used. It helps the model to learn difficult relations while extracting the features. At the end, there is a sigmoid activation function. The purpose of it is to flatten the output.

This helps in an enhanced channel attention weight extraction which helps in generating an enhanced feature map which is basically an element-wise product of the input feature map, x, and the attention map. The overall network diagram is provided in fig 2

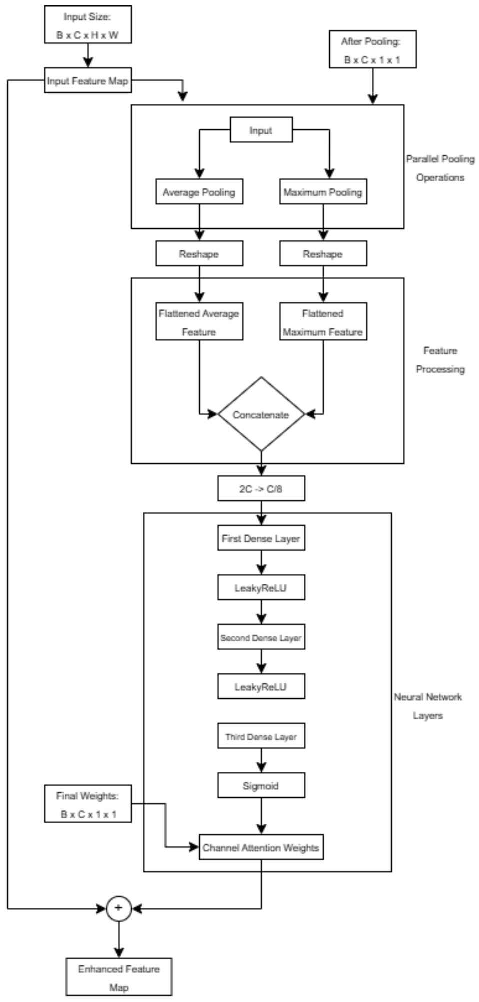


Fig. 2: Channel Attention Block

D. Generator Network

This section talks about the main network for our model.

1) Feature Extraction: The generator network begins with a feature extraction module. The architecture starts with feature extraction. After the features are extracted it performs a 9x9 convolution layer with a stride of 1 and padding of 4. After that Leaky ReLU is applied for non linearity. After that there is one more 3x3 convolution layer, and Leaky ReLU activation function.

2) Residual Blocks and Channel Attention: The head part of the network is a series of 23 residual blocks, that are upgraded with channel attention and dense connections. The residual blocks make learning effective. The Channel Attention Mechanism is giving the network the ability to focus on important features. The dropout layer with a 10% dropout rate improves generalization and prevents overfitting.

3) Multi Scale Feature Fusion: After residual blocks, the network employs a multi scale fusion component with a 1×1 convolutional layer that reduces the high-dimensional dense features. After that, there is a 3×3 convolutional layer followed by a Leaky ReLU activation function. The fusion module combines information from all residual blocks. That means that it captures both local and global features.

4) Global Residual Learning: A global skip connection is involved to propagate initial features through the network. The 3×3 convolutional layer transforms the fused features. After that, the transformed features are added element-wise to the features extracted in the initial stage. The global residual connection let the network to learn the difference between the input and high-resolution target image, making training more stable and efficient.

5) Progressive Upsampling: The progressive upsampling module is used to upscale the input. The upsampling is divided in two parts. At both stages there is a 3x3 convolutional layer. This layer expands the channels by predefined factor. The pixel shuffle moves around the expanded channels into a higher-resolution spatial grid, doubling the spatial resolution. The non-linearity is introduced by using a Leaky ReLU activation function. The Channel Attention Mechanism boosts feature importance. This process is repeated until the desired resolution is achieved.

6) Final Reconstruction: The final part of the network is a final reconstruction. It starts with a 3×3 convolutional layer that reduces the number of channels. It is followed by Leaky ReLU. After that, there is a 9×9 convolutional layer that is reconstructing the final output. The Tanh activation function is applied to normalize output. It is returning high resolution image.

7) Forward Pass: The forward pass has these steps. Extract initial features from the input image. Pass features through 23 residual blocks with channel attention. Fuse the dense outputs into a single feature representation. Add a global residual connection. Upscale the fused features progressively using convolution, pixel shuffling, and channel attention. Reconstruct the final super-resolved image.

E. Discriminator Network

The purpose of the proposed discriminator is to differentiate between high-resolution images that are generated by the generator network and real high-resolution images. The discriminator netwok is shown in 4

1) Feature Extraction: The main block for descriminator network is build with a 3x3 convolutional layer. After that the there is a normalization layer for stabilization. After that there is a Leaky Relu for non linearity. The network build as five discriminator blocks. The fist block has 3 input channels and 64 output channels. It uses stride of 2. The second block has 64 input channels and 128 output channels. The third block has 128 input channels and 256 output channels. The fourth block has 256 input channels and 512 output channels. The fifth block has 512 input channels and 512 output channels.

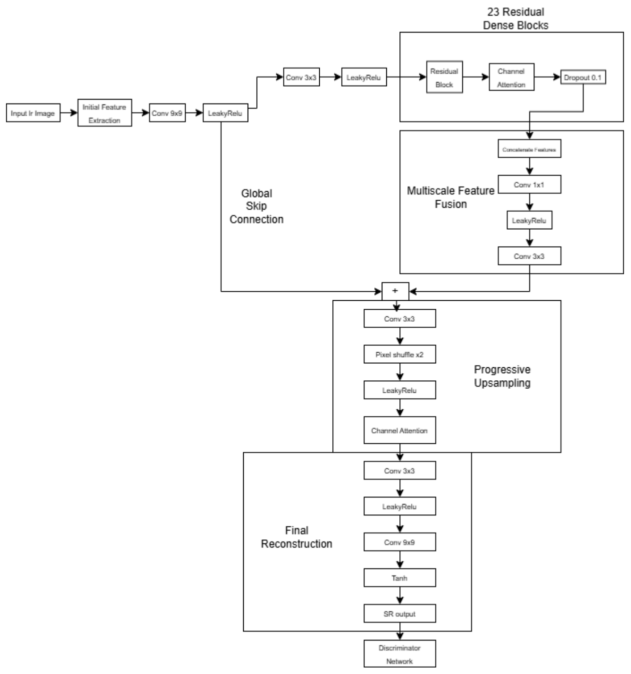


Fig. 3: DOSRGAN

2) Classifier: The classifier is made of adaptive average pooling. The purpose of it is that reduces the spatial dimensions of the feature map to 1×1, creating a global representation of the input. After that we have a 1×1 convolutional layer that are mapping 512 channel feature map to 1024 channels. The Leaky ReLU activation introduces non-linearity. After that we have one more 1×1 convolutional layer that reduces the dimensionality to a single channel, producing the final validity score.

IV. Network Training

The training is based on a generator and discriminator networks. Along with them we add features like dynamic loss weighting, gradient accumulation, and learning rate schedulers. We trained the model for 1000 epochs for two different super-resolution factors (x2, x4).

A. Experiment Setup

1) Loss Functions: We are using three different loss functions. Pixel Loss is focusing on reducing the pixel-wise difference generated and high-resolution images. Content Loss evaluates perceptual similarity. Adversarial Loss gives a chance to the generator network to produce images that the discriminator cannot distinguish from real HR images.

2) Training Process: Training of the generator network consists of three phases. In the phase number one there is pixel loss. In the phase number two content loss is added. In the phase number three adversarial loss is added. The losses while training for x2 and x4 is shown in fig 5.

3) Evaluation: After each epoch, the model is evaluated on validation set, and both Set5 and Set14 test sets using PSNR and SSIM metrics.

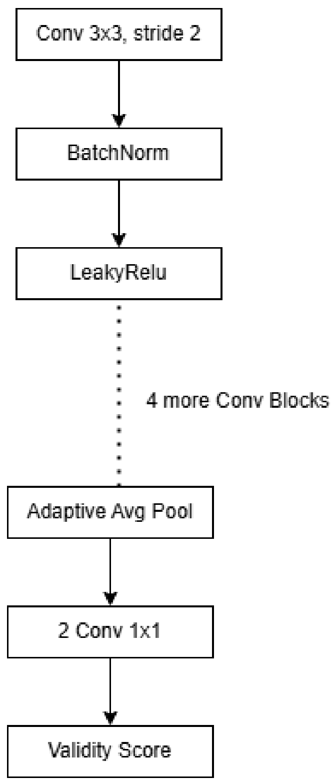
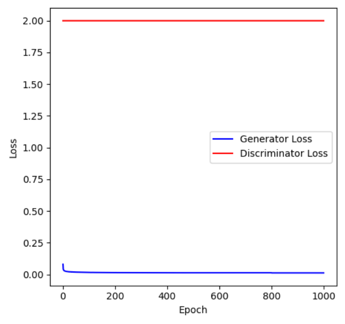
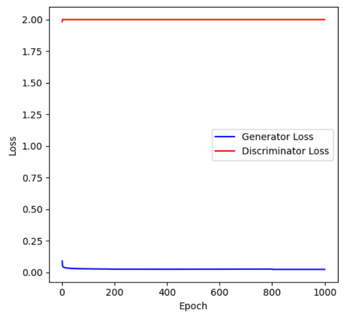


Fig. 4: Discriminator Block



(a) Training Loss for x2.



(b) Training Loss for x4.

Fig. 5: Training losses for factors x2 and x4.

V. Results and Comparison

The best result for factor x2 for set5: PSNR is 35.04, SSIM: 0.9279. The best result for factor x2 for set14: PSNR is 30.44, SSIM: 0.8853. The best result for factor x4 for set5: PSNR is 29.29, SSIM: 0.8853. The best result for factor x4 for set14: PSNR is 25.60, SSIM: 0.7375.

.

A. Qualitative Analysis

While training the model, we saved its instance with the best PSNR value. We used it to generate super resolution image and compare it visually with other models. You can see our results in figures from 7a to 8e.

Eval. Mat.	Scale	Bicubic	SRCNN	FSRCNN	EDSR	DOSRGAN
PSNR	2	33.66	36.66	37.00	38.11	**35.04**
	4	28.42	30.49	30.71	32.46	**29.29**
SSIM	2	0.9299	0.9542	0.9558	0.9601	**0.9279**
	4	0.8104	0.8628	0.8657	0.8968	**0.8472**

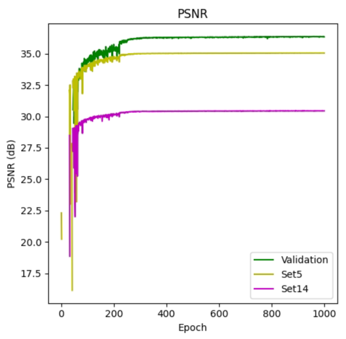
TABLE I: Comparison of Methods for PSNR and SSIM on Set5

Eval. Mat.	Scale	Bicubic	SRCNN	FSRCNN	EDSR	DOSRGAN
PSNR	2	30.23	32.45	32.63	33.92	**30.44**
	4	26.00	27.50	27.59	28.80	**25.60**
SSIM	2	0.8687	0.9067	0.9088	0.9195	**0.8853**
	4	0.7019	0.7513	0.7535	0.7876	**0.7375**

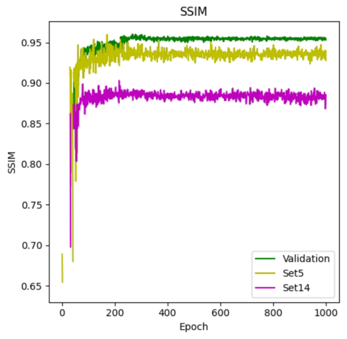
TABLE II: Comparison of Methods for PSNR and SSIM on Set14

B. Conclusion and Future Works

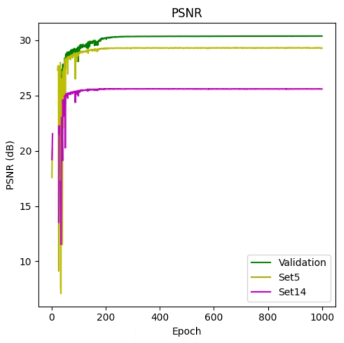
Although we achieved baseline psnr for set 5 for both x2 and x4 super resolution, further work should be done in optimizing the training such that we can get better result for other dataset like set14, BSD100 and URBAN100 datasets. Moreover, we need to test with x3 upsampling to get similar results. As we are not training the discriminator netowork of the code for this reason the discriminator loss function is not coming down which needs to be tested in the future. In conclusion, we can say that this model has potential to reach better values as the results we got are somewhat in the competitive part.



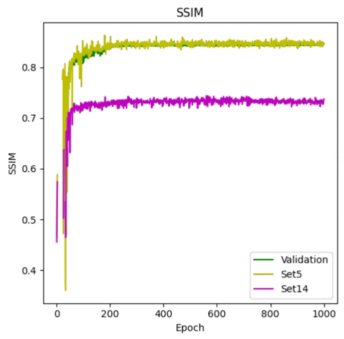
(a) Detailed training curves for factor x2 and PSNR.



(b) Detailed training curves for factor x2 and SSIM.

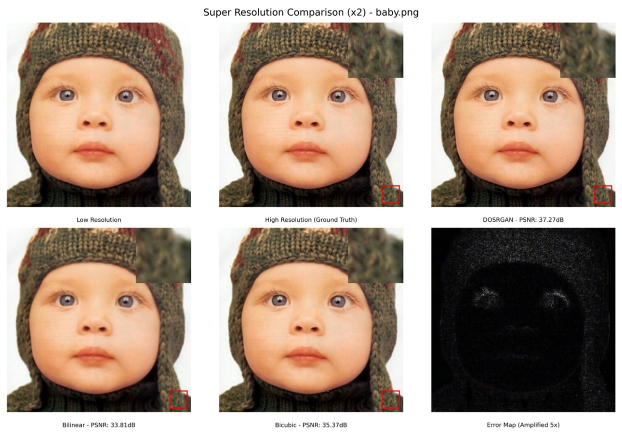


(c) Detailed training curves for factor x4 and PSNR.

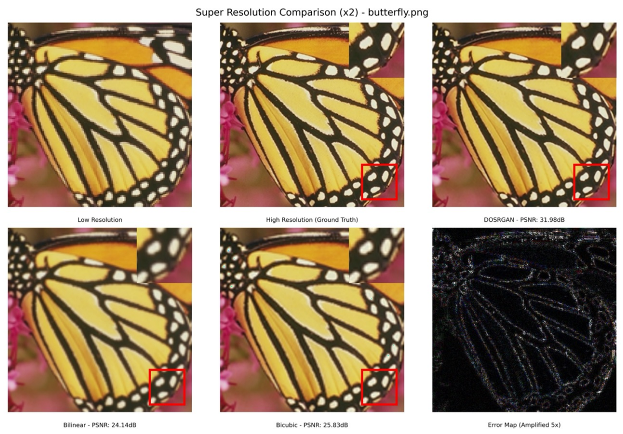


(d) Detailed training curves for factor x4 and SSIM.

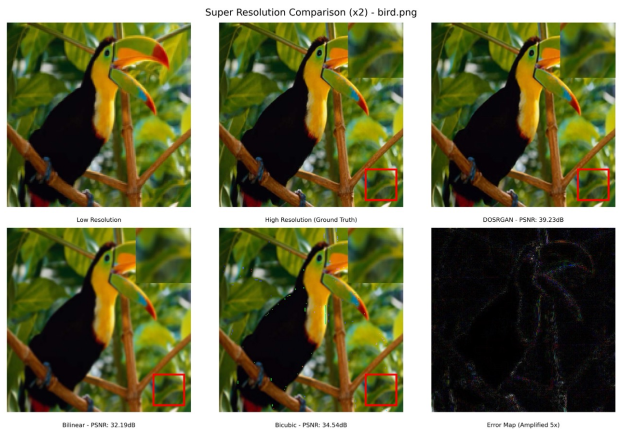
Fig. 6: Training curves for factors x2 and x4.



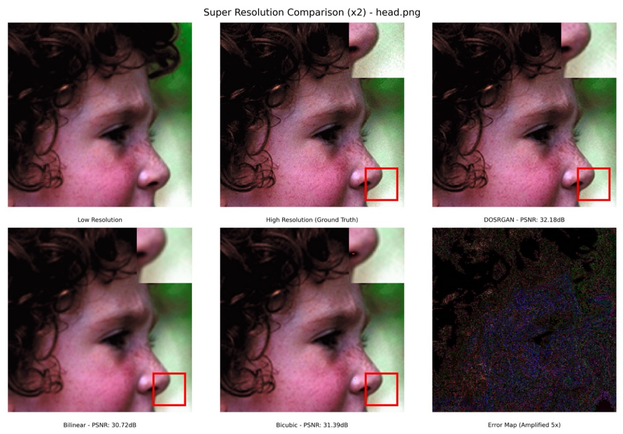
(a) x2 baby



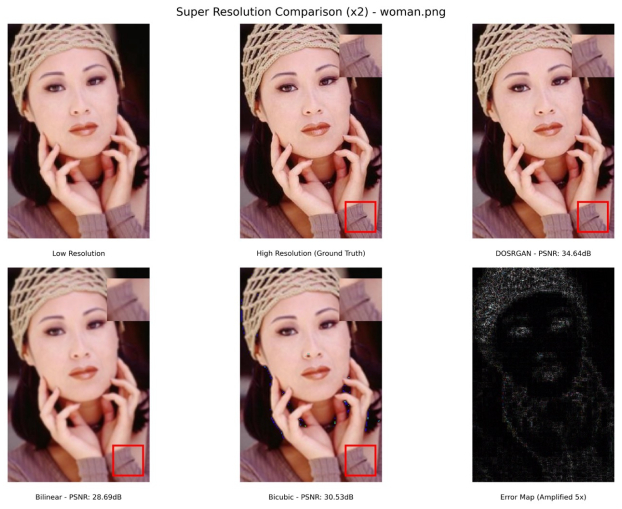
(b) x2 butterfly



(c) x2 bird

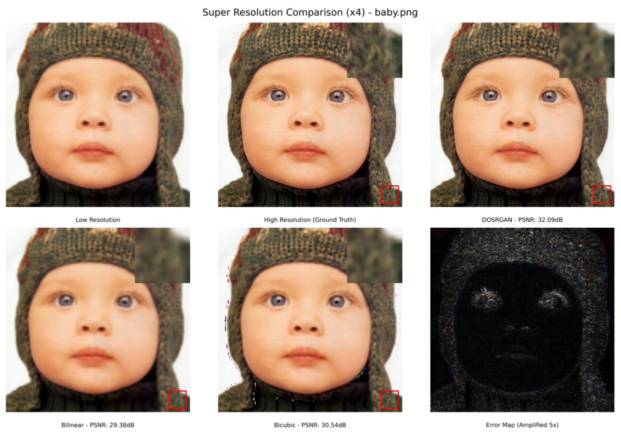


(d) x2 head

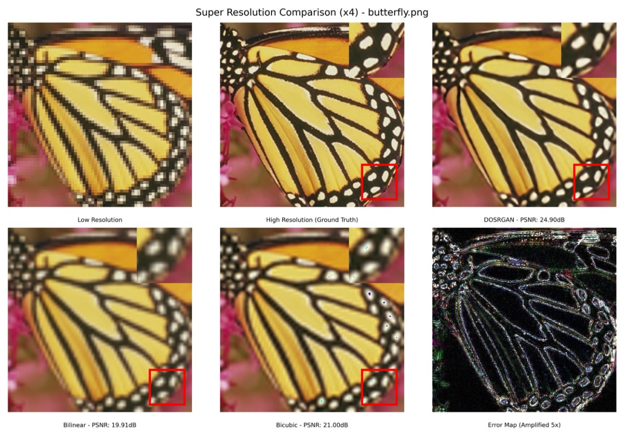


(e) x2 woman

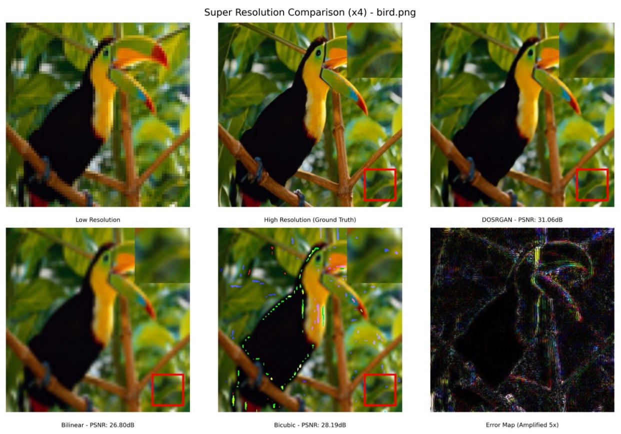
Fig. 7: Comparison of various images at x2 scaling.



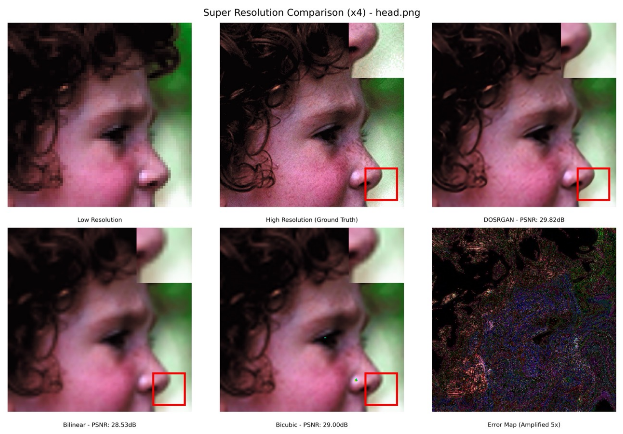
(a) x4 baby



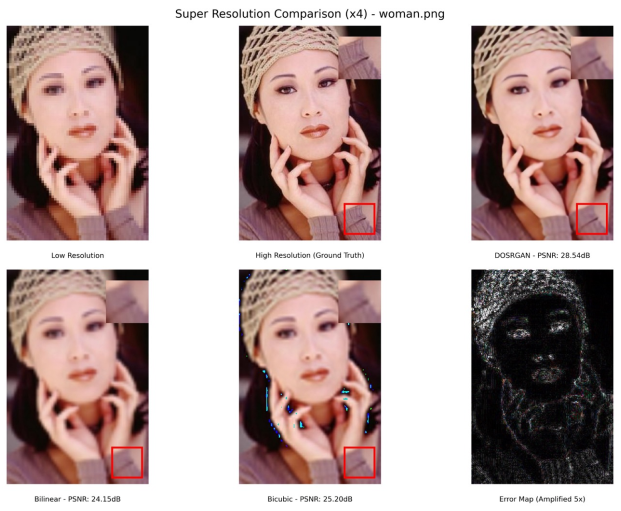
(b) x4 butterfly



(c) x4 bird



(d) x4 head



(e) x4 woman

Fig. 8: Comparison of various images at x4 scaling.