Project: Scale-wise Convolution for image restoration

Mayank Pratap

Purdue University
West Lafayette, USA
pratapm@purdue.edu

Abstract

This project investigates the use of a deep learning approach for single-image resolution enhancement, which focuses on enhancing image quality in high-scale reconstructions. The project utilizes a Scale-Wise Convolution Network (SCN) architecture. It explores multiple resolution pathways that integrate scale-aware attention mechanisms. The multi-scale design allows the model to selectively emphasize relevant image features across varying spatial resolutions, improving details in the reconstructed high-resolution images. Our experiments are conducted on standard benchmarks such as DIV2K and Set5, evaluating the model's performance in terms of Peak Signal-to-Noise Ratio (PSNR). Two similar models are compared, where one model is slightly more complex and their results are compared, effectively observing image enhancement for low-resolution images. The project code can be found at the link given here.

Code — https://github.com/Mayankpr04/ai_scn.git
Datasets — https://data.vision.ee.ethz.ch/cvl/DIV2K/

Introduction

In an age where digital images are everywhere, from social-media to medical imaging, the demand for high-quality visuals is constantly rising. Unfortunately, not all images that are captured are at high resolutions, due to either limitations in hardware or the need to conserve space, as a very high resolution image can take up significant amount of space. Image resolution offers a way to enhance the quality of such low-resolution images by reconstructing a high-resolution version with better details and clearer edges. This is a significant challenge as the model may have to predict fine details that aren't visible in the low-resolution images. A good solution to image enhancement is through the use of Convolutional Neural Networks (CNN). However, traditional CNN-based methods can struggle to capture detailed structures. This project addresses the limitations by implementing a Scale-Wise Convolution Network, originally introduced in the The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20). The paper titled 'Scale-Wise Convolution for Image Restoration'(Fan et al. 2020) introduces a novel approach, which is inspired from spatial-wise convolution for shift-invariance. The scale-wise convolution is proposed to convolve across multiple scales for scale-invariance. In the authors' scale-wise convolutional network (SCN), the input image is first mapped to the feature space of the image and then a feature pyramid representation is built via bi-linear down-scaling progressively. The feature pyramid is then passed to a residual network with scale-wise convolutions. The proposed scale-wise convolution learns to dynamically activate and aggregate features from different scales in each residual building block, in order to exploit contextual information on multiple scales. The project aims to create a similar model and enhance image resolution. We explore dynamic learning rate adjustments which contributes to efficient and effective training. The results highlight the SCN's potential to improve visual clarity across various applications from everyday photo enhancement to specialised and sensitive fields such as military and medical imaging.

Related Work

Scale-invariance in image processing has incited the creation of network architectures that incorporate it. A good example of such a network is highlighted in (Xu et al. 2014). The proposed Scale-invariant Convolution Neural Network (SiCNN) was designed to incorporate multi-scale feature extraction and classification within a multi-column structure. Each column utilizes a shared set of weights, transformed to address different scales, enhancing image classification performance by capturing features across multiple scales. Some other popular multi-scale architectures, like U-Net (Ronneberger, Fischer, and Brox 2015) and SegNet (Badrinarayanan, Kendall, and Cipolla 2015) leverage multiple scales across network stages with the help of interconnections. They have proved to be highly-effective for tasks such as image segmentation. Another important model is the Feature Pyramid Network (Lin et al. 2017) that uses a top-down architecture with lateral connections which refers to connections between layers of different depths and resolutions within the network, allowing high-level semantic feature mapping across scales with the ability to generate rich multi-scale feature maps. This is very useful for tasks like object detection.

Different techniques have been developed to integrate multi-scale information into Neural Network architectures

to improve image restoration. A specific example would be the Dual-State Recurrent Network (DSRN) (Han et al. 2018) proposed in 2018 which introduced a novel approach for super-resolution by utilizing signals from low-resolution and high-resolution scales withing the same network. It has a unique dual-scale structure, where recurrent connections allow information to flow between the two different scales in both directions. This is possible due to the use of a mechanism of delayed feedback. This design allows the model to refine details iteratively across scales, enhancing the quality in terms of the resolution for the output image. Another architecture is the Deep Back-Projection Network (DBPN) (Haris, Shakhnarovich, and Ukita 2018) introduced in 2018. It has a sophisticated iterative process, using multiple up and down-sampling layers to continously refine the image. Each layer in DBPN functions as a back-projection mechanism, correcting errors introduced during scaling and providing increasingly accurate reconstructions. This iterative structure allows the network to maintain finer details and generate higher quality images through repeated interactions between different resolutions, making it particularly effective for high-fidelity image restoration tasks.

Problem Defintion

The challenge of reconstructing high-quality, high-resolution images from low-resolution inputs is complex due to the need for accurately capturing fine details across varying scales. To address the task of increasing or enhancing resolution in image processing, the project focuses on constructing a neural networks model that enhances the image quality by learning how to reconstruct high-frequency details from low resolution inputs. As mentioned earlier, traditional CNN's are limited in handling images with varying spatial frequencies and scale invariance, which are essential for accurate image enhancement. The primary research challenge is how to design a model that can effectively capture multi-scale contextual information across multiple resolutions. This involves developing an architecture that dynamically takes into account various features at different scales while preserving fine details in high-resolution reconstructions. Specifically, this project aims to replicate and evaluate a Scale-Wise Convolutional Network (SCN), which uses a scale-aware attention mechanism to aggregate information across a pyramid of progressively downscaled image features. This model leverages both high and low-resolution signals, allowing the network to learn scale-invariant features crucial for precise detail reconstruction. To give a precise statement on what the project aims to achieve : Can a Scale-Wise Convolution Network (SCN) effectively reconstruct high-resolution images from low-resolution inputs by exploiting multi-scale contextual information and dynamic feature aggregation across varying resolutions?

Methodology

The project implements two models, with different complexities and then compares their performance on the Div2k and Set14 dataset, which are standard benchmark datasets for image resolution and restoration. The first model is called the 'Original' model and the second model is called 'Adaptive'. Let us now understand the architecture for each model.

Scale-Wise-Network (original)

The 'Original' Scale-Wise-Network is designed to improve image resolution by reconstructing high-resolution (HR) images from low-resolution (LR) inputs using multi-scale feature extraction. This model uses a scale-wise approach to convolution, where feature extraction occurs across different scales, followed by residual learning. Here's a detailed breakdown of each component in the model and the techniques applied.

ScaleWiseConv This class defines the ScaleWiseConv layer and is designed to process images at different scales and reconstruct a single high-resolution feature map by merging information across those scales. it consists of the following structure:

- **Convolutional Processing Across Scales:** For each downsampled version, a 2D convolution operation is applied, capturing features specific to that scale. A 3×3 kernel with padding is used to preserve the dimensions of each feature map after convolution.
- **Feature Pyramid:** Each scale-specific feature map is upsampled back to the original input size using bilinear interpolation. The upsampled maps are collected in a feature pyramid list, resulting in a set of feature maps, each corresponding to a different scale. The multi-scale features from the pyramid are aggregated by element-wise summation, producing a final feature map that integrates information across scales.

  This can be defined mathematically as follows:

$$\text{Output} = \sum_{s=0}^{S-1} \text{Upsample}(\text{Conv}_s(F(x)_s))$$

  where $F(x)_s$ represents the features extracted at scale s, S is the total number of scales, and Conv_s is the convolution layer corresponding to scale s.

ResidualBlock The ResidualBlock class introduces residual learning, which addresses the issue of vanishing gradients in deeper networks. By focusing on refining residual features rather than reconstructing all features from scratch, the block enables the model to better capture fine details.

- **Scale-Wise Convolution:** Each residual block contains two sequential ScaleWiseConv layers, which process the input feature maps across multiple scales. These layers help the model analyze information at different resolutions within each block.
- **ReLU Activation:** Between the two scale-wise convolution layers, a ReLU activation function is applied to introduce non-linearity. This activation enhances the model's capacity to capture complex patterns within the image, which is essential for high-quality image reconstruction.

- **Residual Addition:** The output of the final scale-wise convolution layer is added to the original input of the block, forming a residual connection. This addition retains the initial features while incorporating new details learned by the convolutional layers, allowing the network to focus on refining finer details within the image. The output of a residual block, where x is the input and $F(x)$ represents the transformation performed by the two scale-wise convolution layers, can be expressed as:

$$\text{Output} = x + F(x)$$

ScaleWiseNetwork : The ScaleWiseNetwork class consists of the entire architecture for image resolution enhancement.

- **Initial Convolution Layer:** The initial convolutional layer transforms the input image from input channels to an internal representation with the number of features channels. This layer serves as a preparatory step, allowing the model to focus on essential features before further transformations are applied.
- **Residual Blocks:** The network consists of a sequence of ResidualBlock layers. Each block contains ScaleWiseConv layers, enabling feature extraction across multiple scales in a progressive manner. By stacking these residual blocks, the model develops a deeper understanding of hierarchical features within the image.
- **Final Convolution and Pixel Shuffle:** A final convolutional layer is used to adjust the output dimensions to match the target resolution. The number of output channels is set to

$$out_{channels} \in N : out_{channels} = c_{out} \times s^2$$

- **PixelShuffle Operation:** The PixelShuffle operation rearranges the channels to upscale the spatial resolution by the scale factor, producing the final high-resolution output image. The mathematical transformation of the main model can be described by passing the input X through the initial convolution, the sequence of residual blocks, and finally the PixelShuffle upscaling operation. This process can be formulated as:

$$\begin{aligned}\text{Output} = \text{PixelShuffle}(&\text{FinalConv}(\\ &\text{ResidualBlocks}(\text{InitialConv}(X))))\end{aligned}$$

Adaptive Scale-wise convolution model

- **Scale Attention Module:** The 'ScaleAttention' class takes the outputs from each scale of the 'ScaleWiseConv' module and applies a 1x1 convolution, followed by a softmax activation function across the different scales. This attention mechanism computes weights for each scale, allowing the model to prioritize certain scales based on the input features. Mathematically, let F_s represent features at scale s. The attention-weighted output for a given scale is:

$$\text{Output} = \sum_{s=0}^{S-1} \alpha_s \cdot F_s$$

where α_s are attention weights computed for each scale.

- **ScaleWiseConv Module with Attention:** This module comprises multiple convolutional layers, each applied to a downscaled version of the input image. After convolution at each scale, instance normalization is applied to each feature map independently. Each normalized feature map is then upsampled to match the original input size, and an attention mechanism aggregates features across scales for a refined multi-scale feature representation. Let $F_s(x)$ represent the convolutional operation at scale s for input x. The output of the ScaleWiseConv layer can be formulated as:

$$\text{ScaleWiseConv Output} = \text{Attention}\left(\{\text{Upsample}(F_s(x))\}_{s=0}^{S-1}\right)$$

This approach enhances scale-awareness and facilitates focused feature integration.

- **ResidualBlock Module:** Each residual block comprises two sequential ScaleWiseConv layers with a ReLU activation in between. The block includes a skip connection, which retains the original input features and combines them with the transformed features after convolution. Letting $F(x)$ denote the transformation within the residual block, the output can be represented as:

$$\text{Output} = x + F(x)$$

- **Initial Convolution:** An initial convolutional layer maps the input image to a feature space with a specified number of channels, `num_features`, preparing the input for further processing in the residual blocks.
- **Stacked Residual Blocks:** The network stacks multiple 'ResidualBlock' layers, as defined by `num_blocks`. Each block integrates multi-scale features progressively, aiding in hierarchical feature extraction across scales.
- **Final Convolution and Pixel Shuffle:** A final convolutional layer adjusts the output channels, preparing for the 'PixelShuffle' operation, which increases spatial resolution by the scale factor, producing the high-resolution output image. Formally, for an input X, the transformation from input to output is described as:

$$\begin{aligned}\text{Output} = \text{PixelShuffle}(&\text{FinalConv}(\\ &\text{ResidualBlocks}(\text{InitialConv}(X))\end{aligned}$$

Comparison of Model 1 and Model 2

Scale Attention Mechanism

- **Model 1:** Combines features from each scale through direct summation, without attention. Each scale contributes equally to the final output.
- **Model 2:** Uses a scale-wise attention mechanism with a softmax-based module that assigns weights to each scale, allowing certain scales to be prioritized for more detailed reconstructions.

Normalization Layers

- **Model 1:** Lacks normalization layers, keeping the architecture simpler but possibly less stable.
- **Model 2:** Includes instance normalization in each `ScaleWiseConv` layer, which helps stabilize training across scales.

Feature Pyramid Construction and Aggregation

- **Model 1:** Constructs the feature pyramid by upsampling and summing feature maps from each scale without adaptive weighting.
- **Model 2:** Aggregates the feature pyramid with scale-specific attention, enabling adaptive control over which scales are emphasized.

Residual Learning in Blocks

- **Model 1:** Applies basic residual connections for simpler feature retention.
- **Model 2:** Adds instance normalization within residual blocks for improved stability while learning complex patterns.

Complexity and Parameter Count

- **Model 1:** Simpler and faster due to fewer parameters, but may lack the ability to capture intricate details.
- **Model 2:** More complex with higher parameter count, aiming to enhance performance through additional attention and normalization layers.

Upsampling and Final Output

- Both models employ a final convolution and pixel shuffle for upscaling, but Model 2's attention mechanism may enhance the quality of upsampled results.

Summary of Key Differences

- **Attention Mechanism:** Model 2 introduces scale-wise attention; Model 1 does not.
- **Normalization:** Model 2 uses instance normalization for stability, absent in Model 1.
- **Feature Aggregation:** Model 2 has adaptive weighted feature aggregation.
- **Complexity:** Model 2 is more complex with a higher parameter count, targeting improved resolution enhancement.

Evaluation and Loss Functions

When training and testing our image enhancement models, we use specific methods to measure how well they perform. We combine two types of "loss functions" during training to help the model learn effectively, and we use two measurement tools (PSNR and PSNR-Y) to check how good the enhanced images look.

Training Loss: Combined L1 and Smooth L1 Loss

We use two loss functions together to get the best results: 1. L1 loss: Helps ensure each pixel is as accurate as possible 2. Smooth L1 loss: Helps handle unusual or extreme values better

We combine these functions like this:

$$\text{Loss} = \alpha \cdot \text{L1\_Loss(SR,HR)} + \beta \cdot \text{Smooth\_L1\_Loss(SR,HR)} \tag{1}$$

Where:

- L1 Loss is calculated as:

$$\text{L1\_Loss(SR,HR)} = \frac{1}{N} \sum_{i=1}^{N} |SR_i - HR_i| \tag{2}$$

- Smooth L1 Loss is defined as:

$$\text{Smooth\_L1\_Loss}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{3}$$

  where $x = SR - HR$
- We set $\alpha = 1$ and $\beta = 0.5$ as weights

Evaluation Metrics: PSNR and PSNR-Y

We use two ways to measure how good our enhanced images look:

Peak Signal-to-Noise Ratio (PSNR) PSNR tells us how close our enhanced image is to the perfect high-resolution version. Higher value means better quality. Here's how we calculate it:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{4}$$

Where:

- MAX is 255 (the highest possible pixel value for standard images)
- MSE (Mean Squared Error) is:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (SR_i - HR_i)^2 \tag{5}$$

PSNR-Y (Brightness-based PSNR) PSNR-Y looks specifically at how well we maintain the brightness of the image, which is particularly important for human viewers. It's calculated like regular PSNR but only looks at the brightness channel:

$$\text{PSNR-Y} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}_Y} \right) \tag{6}$$

Dynamic Learning Rate with Cosine Annealing

To help our model learn better, we gradually change how fast it learns (the learning rate). We use a method called Cosine Annealing, which smoothly adjusts the learning rate over time:

$$\eta(t) = \eta_{\min} + 0.5 \cdot (\eta_{\max} - \eta_{\min})(1 + \cos(\frac{t}{T}\pi)) \quad (7)$$

Where:

- $\eta_{\max}$ is the starting learning rate
- $\eta_{\min}$ is the smallest learning rate we'll use
- t is which training round we're on
- T is the total number of training rounds

Experiment Results

The two models were initially trained for 10 epochs using only L1 loss. The training was done one the div2k dataset (Timofte et al. 2017) using the X2 low resolution images and the corresponding high resolution images. There were 800 images for training and 100 for testing. The results are as follows

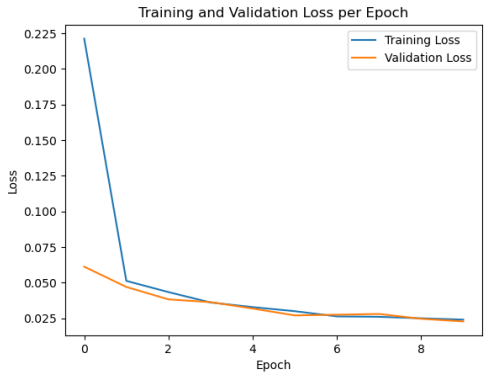

Figure 1: Original model trained for 10 epochs with only L1 loss

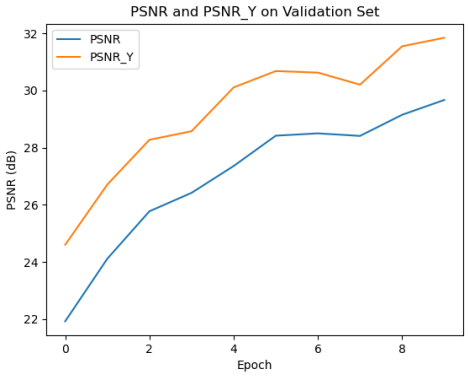

Figure 2: Original model trained for 10 epochs: PSNR

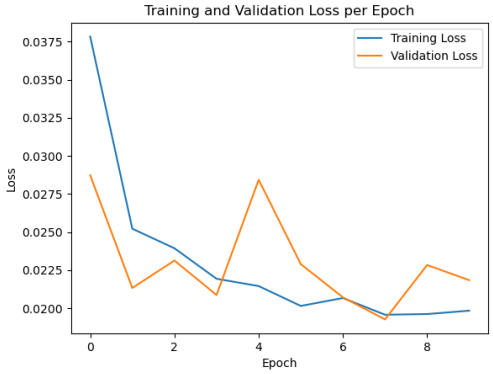

Figure 3: Adaptive model trained for 10 epochs using L1-loss

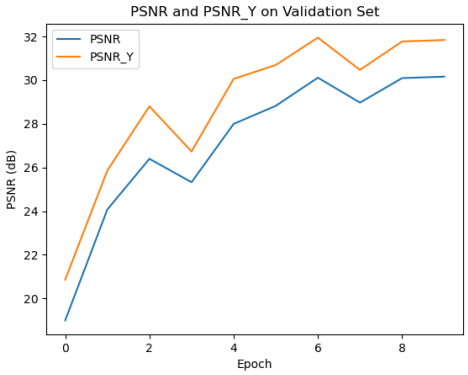

Figure 4: Original model trained for 10 epochs: PSNR

Change in loss function The loss function (Stack Exchange 2018) was then changed to a weighted combination of L_1 loss and Smooth L_1 loss with values of $\alpha = 1$ and $\beta = 0.5$, where α and β correspond to the weights of L_1 and Smooth L_1 loss, respectively. The combined loss function $\mathcal{L}$ can be represented as follows:

$$\mathcal{L} = \alpha \cdot \text{L1\_Loss} + \beta \cdot \text{SmoothL1\_Loss}$$

where:

- L1_Loss is the L1 loss between the predicted and target images.
- SmoothL1_Loss is the Smooth L1 loss, which is robust to outliers than the standard L1 loss.

The original model was trained for 20 epochs, and the adaptive model was trained for 30 epochs.

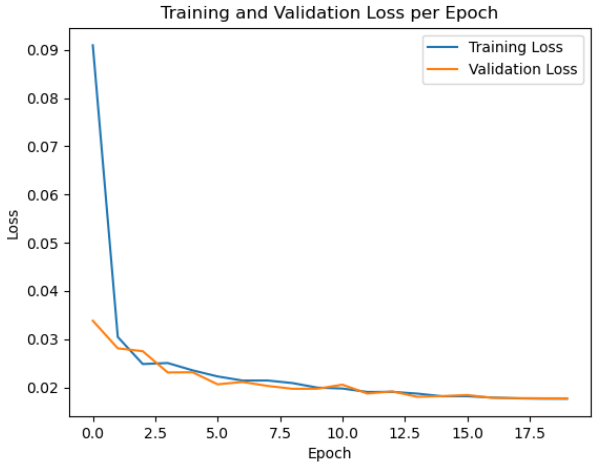

Figure 5: Original model trained for 20 epochs with weighted Loss function

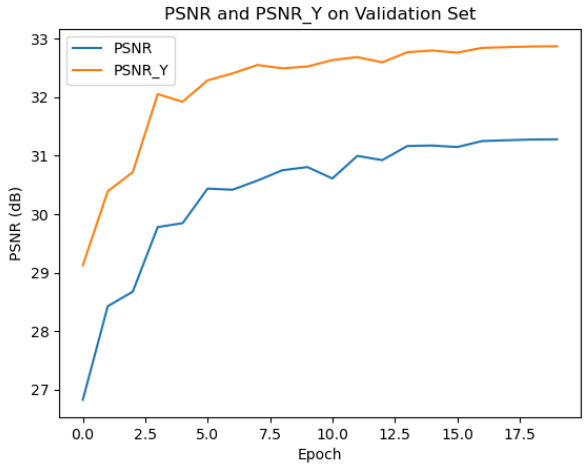

Figure 6: Original model trained for 20 epochs: PSNR

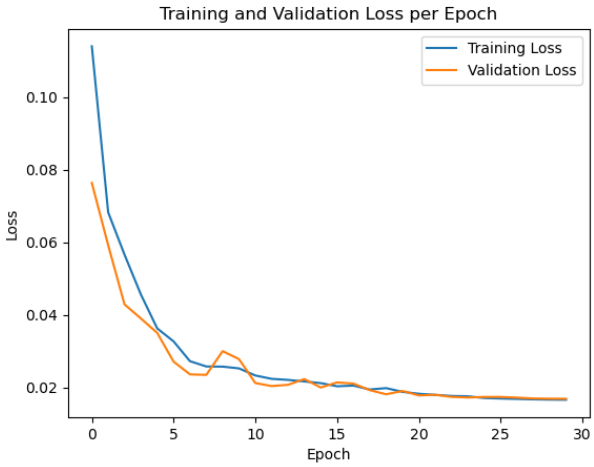

Figure 7: Adaptive model trained for 30 epochs with new loss function

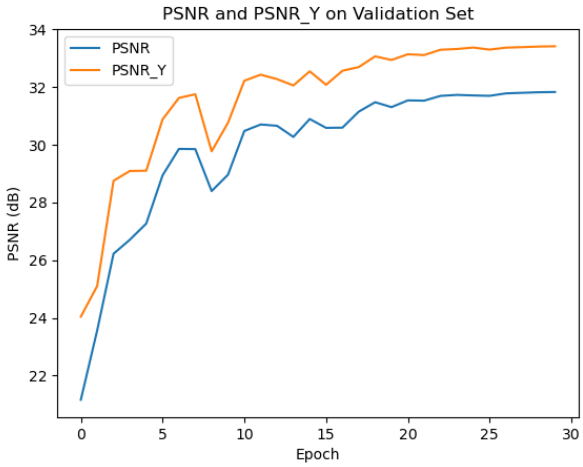

Figure 8: Adaptive model trained for 30 epochs : PSNR

Inference We can infer the training results as follows: From the training and PSNR of the 10 epoch models, while the PSNR looks relatively good, our model could be trained for a few more epochs, as indicated by the adaptive models' loss results, whereas we see a sharp drop in the learning of the first model. These models were trained with a fix learning rate of 0.001. The models were then training with a dynamic learning rate and a weighted loss function, and we can see learning improves for the adaptive model whereas the original model converges quickly. Both models have relatively good PSNR values on dataset, with Adaptive model performing slightly better.

Some sample images Here are some images from the evaluation of both models on div2k dataset

Figure 9: Original model

Figure 10: Enhanced model

Inference While both models perform resolution enhancement, there are certain sublte differences which can be observed. We observe that the output from the 'original' model contains more artifact problem compared to the adaptive model.

Testing on a single image The models were also tested on images from different datasets, such as the Set14 dataset which is part of the set5 dataset. A single image was taken "lenna.png" and downscaled to half the resolution of the original image and then the models were run on it to obtain the resolution enhanced images and are shown below:

Figure 11: Original model output on lenna.png

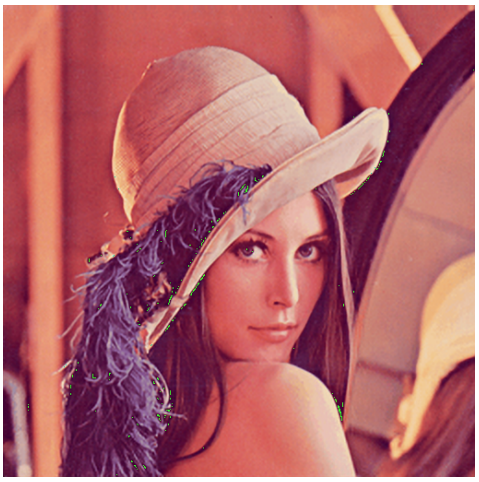

Figure 12: Adaptive model output on lenna.png

Inference Similar to the lotus image, while both models perform resolution enhancement, We observe that the output from the 'original' model has more artifact problems compared to the adaptive model.

Comparing Results We have the following comparison between the two models and the author's model on the DIV2K and Set14 datasets.

Dataset	Author	Original	Adaptive
DIV2K x2	35.19	31.28	31.83
Set14 x2	34.14	36.90	36.72

Table 1: Comparison of results between the author's model, the original model, and the adaptive model on the DIV2K and Set14 datasets.

Future Directions

Artifact Removal

Objective: High-resolution image reconstruction can often introduce artifacts, as visible in our results, such as ringing or blurring around edges. Integrating specialized techniques for artifact removal could lead to cleaner high-resolution images.

Approach: One approach, as we have done, is to use a combination of loss functions specifically designed to detect and reduce artifacts. The Smooth L1 loss (also known as Huber loss) is robust to outliers and can help reduce sharp discontinuities in pixel intensities. Additionally, adding a separate artifact detection and removal module within the network could focus specifically on mitigating these imperfections. The objective function for artifact removal could be formulated as:

$$\mathcal{L}_{\text{artifact}} = \alpha \cdot \text{L1Loss} + \beta \cdot \text{SmoothL1Loss}, \quad (8)$$

where α and β are weighting factors that can be adjusted to control the contribution of each loss term to the overall objective.

Noise Reduction

Objective: Many low-resolution images, especially those captured under challenging conditions, contain noise. Enhancing their resolution without amplifying this noise is crucial for clearer outputs.

Approach: A dedicated denoising block can be added to the network, processing images before the main feature extraction layers. Techniques such as non-local means or wavelet-based denoising filters could be integrated into the model for this purpose. Additionally, the use of joint loss function—combining L1 loss for super-resolution with MSE loss for denoising could encourage the model to minimize noise. Given an input image X with noise, the objective function for noise reduction could be represented as:

$$\mathcal{L}_{\text{noise}} = \gamma \cdot \text{L1Loss}(Y, X) + \delta \cdot \text{MSELoss}(Y, X), \quad (9)$$

where Y is the target high-resolution output, and γ and δ are weights that control the contribution of super-resolution and denoising to the objective function.

References

Badrinarayanan, V.; Kendall, A.; and Cipolla, R. 2015. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39: 2481–2495.

Fan, Y.; Yu, J.; Liu, D.; and Huang, T. S. 2020. Scale-Wise Convolution for Image Restoration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07): 10770–10777.

Han, W.; Chang, S.; Liu, D.; Yu, M.; Witbrock, M.; and Huang, T. S. 2018. Image Super-Resolution via Dual-State Recurrent Networks. *CoRR*, abs/1805.02704.

Haris, M.; Shakhnarovich, G.; and Ukita, N. 2018. Deep Back-Projection Networks For Super-Resolution. *CoRR*, abs/1803.02735.

Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segme ntation. *CoRR*, abs/1505.04597.

Stack Exchange. 2018. How to interpret Smooth L1 Loss? https://stats.stackexchange.com/questions/351874/how-to-interpret-smooth-l1-loss. Accessed: 2024-11-10.

Timofte, R.; Agustsson, E.; Van Gool, L.; Yang, M.-H.; Haris, M.; et al. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Xu, Y.; Xiao, T.; Zhang, J.; Yang, K.; and Zhang, Z. 2014. Scale-Invariant Convolutional Neural Networks. *CoRR*, abs/1411.6369.