Deep Learning Based Super-Resolution: SRCNN Model Comparison with Own Dataset

Ömer Aysal –2106102004

Abstract

This study investigates the effect of input type and frame selection on image super-resolution performance using three different models based on SRCNN (Super Resolution Convolutional Neural Network) architecture. Model 1 is designed to work on RGB and luminance (Y) Channel, while Model 2 is designed to work only on luminance (Y) channel. Among the Y-channel models, Model 1 is developed in TensorFlow and Model 2 in PyTorch. Despite the differences in input and implementation, all models share a comparable three-layer network structure and are trained using the same dataset. Model performance is evaluated through training loss measurements and Peak Signal-to-Noise Ratio (PSNR) values. For consistency in visualization, the output images are reconstructed in RGB format after estimation. Experimental results provide a comparative analysis of how input channel selection and deep learning frameworks affect super-resolution results.

Keywords*: Super Resolution, Deep Learning, Image Enhancement, Luminance Channel, TensorFlow, PyTorch, PSNR*

1.Introduction

Image Super Resolution (SR) has become an increasingly important research area, especially in the fields of medicine, remote sensing, security, and digital media, with the increasing demand for high-quality visual content. The main goal of SR is to obtain high-resolution images from low-resolution images, thereby significantly improving detail and visual fidelity. Traditionally, SR has been performed through interpolation methods or hand-designed enhancement algorithms. However, these approaches often struggle to preserve sharp edges and complex patterns, which limits the quality of the results obtained.

The rapid development of deep learning, especially Convolutional Neural Networks (CNNs), has brought significant progress in image enhancement tasks such as SR. CNN-based methods provide more accurate and visually pleasing reconstructions compared to traditional methods, thanks to their ability to learn complex transformations from large datasets. These networks automatically learn and recover the features required for reconstructing high-resolution images.

Among the leading CNN-based approaches for SR, Super Resolution Convolutional Neural Network (SRCNN) stands out with its simple design and efficient performance [1]. The SRCNN framework learns an end-to-end mapping from low-resolution input to high-resolution output and achieves this through a series of convolution layers. Although its architecture has been widely implemented and evaluated, there is limited research that directly compares the impact of different input channels, such as full-color RGB and only luminance (Y) channel, and different implementation frameworks, such as TensorFlow and PyTorch, on its performance. Understanding these factors is critical for optimizing SRCNN implementations and selecting the most appropriate approach for specific applications.

This work aims to systematically investigate and compare the performance of three different SRCNN-based models under different experimental conditions. The specific objectives are: to evaluate the performance of an SRCNN model trained on full RGB images; to evaluate the performance of an SRCNN model trained only on the Y (luminance) channel of the input images; and to compare the performance of two SRCNN models trained on the

Y channel (one implemented with TensorFlow and the other with PyTorch) to identify potential framework-induced differences. This research aims to analyze how the choice of input modality (RGB and Y) and the choice of deep learning library affect the overall performance of SRCNN on SR tasks based on quantitative evaluation metrics.

To achieve these goals, a comparative experimental approach is adopted. Three different SRCNN models are developed and trained using the same dataset and a combined three-convolution layer architecture. The first model will process full RGB color images. The second and third models will process only the Y (luminance) channel of the input images. To study the effect of the implementation framework, the model processing the Y channel will be implemented in both TensorFlow and PyTorch. The performance evaluation will be primarily based on the training loss and the Peak Signal-to-Noise Ratio (PSNR) of the reconstructed images. These metrics will provide quantitative information about the learning efficiency of the models and the perceptual quality of the resulting high-resolution images. By maintaining consistent variables across all experiments, this study will provide a focused and illuminating analysis of how data representation and software tools affect the effectiveness of SRCNN-based super-resolution models.

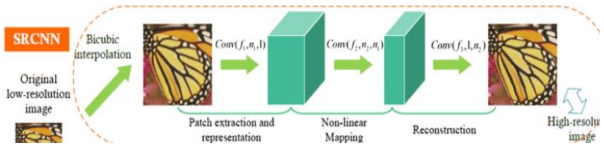

Figure 1.1 [2]

2. Methodology

2.1 Data

In this study, a dataset consisting of high-resolution (HR) images and their low-resolution (LR) counterparts obtained by bicubic interpolation was used. The dataset was divided into separate folders (train, val, test) for the purpose of training, validation and testing the model. This structure enables super-resolution learning by using LR images as input and determining the corresponding HR images as targets.

Different preprocessing steps were applied according to the input type of the model. In models using RGB input, normalization was performed as [(image - mean) / standard deviation] using the mean and standard deviation values calculated separately for each channel. In models working on the brightness (Y) channel, a simpler normalization was applied by scaling the pixel values to the range of 0 and 1 (pixel values / 255). These preprocessing steps are critical for meeting the data expectations of the model architectures used and increasing the stability of the training process.

2.2 Methods

In this study, the Super-Resolution Convolutional Neural Network (SRCNN) model, which is a basic deep learning method for the image super-resolution (SR) problem, is implemented using the PyTorch framework. SRCNN has a structure derived from traditional convolutional neural networks (CNN) and is optimized to produce high-resolution images from low-resolution images.

The model architecture consists of three convolutional layers with kernel sizes of 9x9, 1x1 and 5x5, respectively. The first layer provides feature extraction, the second layer provides feature maps rearrangement, and the last layer produces high-resolution estimation. ReLU is used as the activation function. The activation function is not used in the last layer of the model, because the estimated image values are intended to be linear.

Only the luminance (Y) channel is used as input. Since the human eye is more sensitive to brightness than color components, only the Y channel is generally used in super-resolution problems.[3] The input images were transformed from RGB space to YCbCr space, then only the Y channel was given as input to the model. Since the model output corresponds to only the Y channel, the estimated Y channel was combined with the original Cb and Cr channels to create the final images and converted to RGB format.

All input images were normalized and scaled to the range [0, 1]. This normalization process was provided by the ToTensor() transformation, and the model outputs were re-extruded to the range [0, 255] to obtain the final visual outputs.

The model was trained with a training dataset [4] containing a total of 684 images. The training process was continued for 10 epochs and mini-batches of 16 images were used in each iteration. The training and test datasets consist of low and high resolution conjugate images.

The loss function used in training is Mean Squared Error (MSE), and the error is calculated over the mean squared difference between the high resolution estimate and the true high resolution. Different learning rates were assigned to the three convolution layers of the model; 0.0001 was used for the first two layers and a smaller value of 0.00001 was used for the last layer. This strategy aims to provide more precise tuning of the last layer of the model.

The performance of the model was evaluated with both training loss and Peak Signal-to-Noise Ratio (PSNR) metrics. PSNR is a metric that measures how close the output image is to the noise-free reference image and is expressed in dB. The average PSNR value was calculated on the test dataset at the end of each epoch.

All stages of this study were performed using the Python programming language and the deep learning model was written with the PyTorch library. PIL (Python Imaging Library) was used for image operations and the Torchvision library was used for tensor transformations. GPU support was provided during the training process and calculations were accelerated using CUDA technology (if support is available).

3. Results

To evaluate the performance of the three SRCNN models, Peak Signal-to-Noise Ratio (PSNR) was used as the evaluation metric. PSNR is a widely used metric in image reconstruction tasks as it quantifies the similarity between the reconstructed high-resolution image and the real image. The following table shows the average PSNR values obtained from the test set for each model:

Model	***Input Type***	***Batch Size***	***Epoch***	***Average PSNR (dB)***
Model-1	*RGB Channel*	*16*	*10*	*30.37*
Model-1	*Y Channel*	*16*	*10*	*27.99*
Model-2	*Y Channel*	*16*	*10*	*24.48*

Table 3.1

Figure 3.1 High Resolution Image

Figure 3.2 Model-1 Image with RGB Channel

Figure 3.3 Model-1 Image with Y Channel

Figure 3.4 Model-2 Image with Y Channel

The performance of three different Super Resolution Convolutional Neural Network (SRCNN) configurations was analyzed using the Peak Signal-to-Noise Ratio (PSNR) metric, a widely used quantitative measure to evaluate the similarity between reconstructed and real high-resolution images. Table 3.1 summarizes the average PSNR values obtained on the test dataset after a batch size of 16 and 10 epochs of training for each model.

Model-1 was trained separately with two different input types using the TensorFlow library. In the first scenario, only the RGB channel was used as input and this configuration showed the highest performance with an average PSNR value of 30.37 dB. In this model, normalization was performed by extracting the channel mean of each pixel value and dividing it by its standard deviation (x = (x - mean) / std). In the second scenario, Model-1 was trained only with the Y (luminance) channel and an average PSNR value of 27.99 dB was obtained. This result shows that the Y channel successfully represents the structural details of the image; however, the lack of color (chrominance) information reduces the overall quality.

Model-2 was developed using PyTorch and trained only with the Y channel. This model differs from Model-1 both in terms of architecture and frame differences. In addition, in this configuration, the input images were normalized only as x = x / 255.0. Model-2 showed the lowest success with an average PSNR value of 24.48 dB. These results reveal that not only the choice of the input channel (RGB or Y), but also the deep learning library and normalization strategy used have decisive effects on the super-resolution performance.

4.Discussion

When the PSNR results obtained from three different SRCNN models applied on the test data in this study were compared, it was seen that Model-1, which is based on TensorFlow and uses the RGB channel as input, gave the most successful result with a value of 30.37 dB. Among the models that use only the Y channel, it was determined that Model-1, which was also trained with TensorFlow, exhibited a higher PSNR performance (27.99 dB vs. 24.48 dB) compared to Model-2 developed on PyTorch platform. It is thought that this performance difference may be due to technical differences in the weight initialization methods, optimization algorithms and data preprocessing approaches of the frameworks used. All models were trained under the same conditions, for 10 epochs and with a batch size of 16. This situation

reveals that both the input type (RGB or Y channel) and the development environment used (TensorFlow or PyTorch) play a decisive role in super-resolution performance.

There are some limitations to this study. First of all, the models were trained for only 10 epochs, and this limited training time may not fully reflect the learning capacity of the model. In addition, the relatively small size of the dataset used in training limited the generalization capabilities of the models. It may be possible to achieve higher performance in studies conducted with larger and more diverse datasets.

Various technical and structural difficulties were experienced during the model development process. Since some of the codes used were incompatible with current library versions, errors were encountered, especially on the PyTorch side, due to version differences, and these codes were rearranged in accordance with modern APIs. On the other hand, size incompatibilities were found between the input and target images in the dataset used; in order to eliminate these problems, operations such as resizing and cropping were carefully applied in the preprocessing stage

5.Conclusion

This study systematically evaluated the impact of input type (RGB and luminance Y-channel) and deep learning frameworks (TensorFlow and PyTorch) on the performance of SRCNN models for image super-resolution. The results showed that the model trained on full RGB images using TensorFlow achieved the highest PSNR value and exhibited the best reconstruction quality among them. Models trained on only Y-channel also showed inferior performances to RGB, although they exhibited lower PSNR scores, highlighting the importance of chrominance information in super-resolution tasks. Furthermore, the comparison between TensorFlow and PyTorch implementations of the Y-channel model revealed significant performance differences, likely influenced by factors such as framework-specific optimizations and preprocessing strategies.

Throughout the project, significant insights were gained into how data representation and software environment can significantly impact deep learning results for image enhancement. The study also revealed challenges related to dataset size, training time, and software compatibility, which collectively affect model performance and generalization.

For future work, extending the training period beyond 10 epochs and using larger, more diverse datasets may improve model robustness and accuracy. Additionally, exploring alternative normalization techniques, different network architectures, or hybrid input strategies that combine luminance and color information may further improve super-resolution results. Finally, deeper examination of framework-specific implementation details may provide valuable guidance for optimizing SRCNN and other CNN-based super-resolution models.

6. References

[1] https://ieeexplore.ieee.org/document/9849813

[2] https://www.researchgate.net/figure/Schematic-diagram-of-FSRCNN-model-and-SRCNN-model_fig9_328186859

[3] https://en.wikipedia.org/wiki/YCbCr

[4] https://github.com/eugenesiow/super-image-data?tab=readme-ov-file

[5] https://github.com/artifacia/Super-Resolution/tree/master

[6] https://github.com/Mirwaisse/SRCNN