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

Melisa Aslan -2106102003

Abstract

In this project, the performances of three different models developed with the Super Resolution Convolutional Neural Network (SRCNN) architecture were compared. The first model works directly with RGB and Y channel [6], while the other model [7] is designed to take only Y (luminance) channel as input. One of the models working with the Y channel was implemented using TensorFlow, and the other using PyTorch. All models were trained with the same dataset and a similar three-layer architecture was used. The comparison was made on the training loss and PSNR (Peak Signal-to-Noise Ratio) values. Although the models only worked on the Y channel, the output images were converted to RGB format and saved. The obtained results reveal the effects of the input type (RGB or Y channel) and the library used (TensorFlow or PyTorch) on the super resolution performance.

Keywords: CNN , High Resolution , Low Resolution, PNSR ,Y Channel

1. Introduction

Super Resolution (SR) is a technique used to increase the resolution of low-quality images and produce higher resolution images from lower resolution images. This task plays an important role in various fields such as medical imaging and satellite imagery, where detailed and clear images are vital. In recent years, deep learning models, especially Convolutional Neural Networks (CNNs), have made significant strides in solving SR problems by providing better results compared to traditional methods due to their ability to learn complex mappings and recover fine image details that hand-crafted filters usually fail to capture.

One of the most popular deep learning-based architectures for super resolution is the Super Resolution Convolutional Neural Network (SRCNN). SRCNN leverages the power of CNNs to reconstruct high-resolution images from low-resolution inputs by learning a mapping from low-resolution images to high-resolution images [1]. The model has been widely adopted due to its simplicity and efficiency in restoring image details.

However, the impact of different input types (full RGB and Olly luminance Y channel) and deep learning frameworks (TensorFlow and PyTorch) on SRCNN performance has not been studied extensively. This is the main motivation behind our project.

In this project, we investigate the performance differences between three SRCNN implementations, each designed to handle different types of input data. The first model takes full RGB images as input, while the second and third models are designed to work only with the luminance (Y) channel of the images. The second model is implemented using TensorFlow, while the third model is implemented using PyTorch. Despite using different frameworks, all three models have a similar architecture consisting of three convolutional layers.

The aim of this study is to evaluate how the input format (RGB and Y channel) and the implementation framework (TensorFlow and PyTorch) affect the performance of SRCNN models trained on the same dataset. The aim is to understand which approach provides better results for super-resolution tasks and whether the choice of framework or the type of input channel has a significant impact on the performance.

To ensure a fair comparison, all models are trained on the same pre-processed datasets and their performance is evaluated based on training loss and Peak Signal-to-Noise Ratio (PSNR). The models share the same architecture and differ only in the input format and framework implementation.

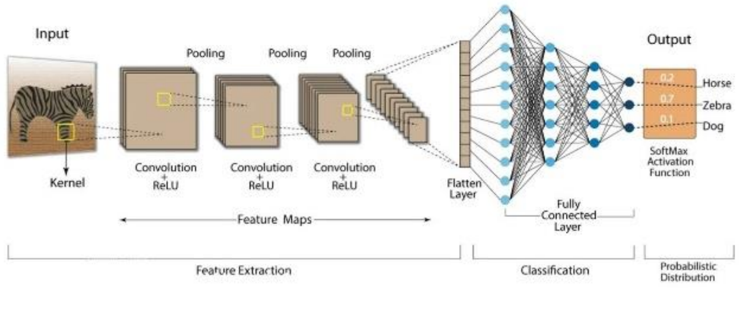

Figure 1.1 CNN Architecture [3

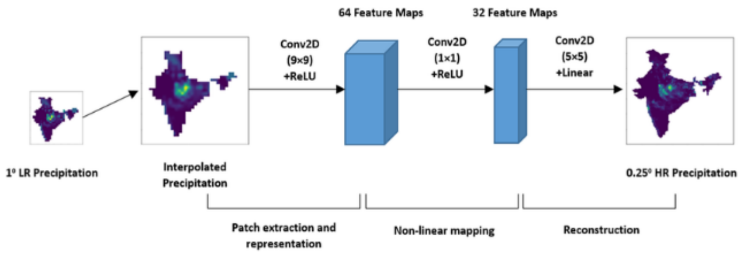

Figure 1.2 SRCNN Architecture [2]

2. Methodology

2.1 Data

The dataset used in this study is divided into three separate folders: training (train), validation (val), and test (test) [5]. Each folder contains high-resolution (HR) images and their corresponding low-resolution (LR) versions, which are created by downscaling HR images using the bicubic interpolation method. In other words, LR images are directly obtained by downscaling HR images by a certain scale factor (e.g. ×2 or ×4). Thanks to this structure, the model performs super-resolution learning by taking the LR image as input and using the corresponding HR image as a target.

Depending on the model, two different normalization strategies are used. For models using RGB input, images are usually normalized using the channel-wise mean and standard deviation in the following format: [(image - mean) / std], where the mean and standard values are calculated from the dataset. On the other hand, models working in the brightness (Y) channel use a simpler normalization by scaling the pixel values to the range [0, 1]. ToTensor(), which divides all pixel values by 255. These preprocessing choices align with the data expectations of the respective model architectures and help ensure stable training and convergence.

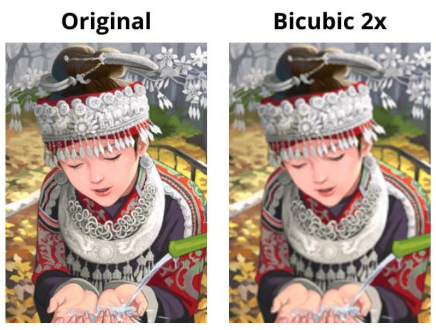

Figure 2.1.1 Bicubic Interpolation [4]

2.1 Methods

In this study, the deep learning-based SRCNN (Super-Resolution Convolutional Neural Network) architecture was applied for the super-resolution problem, which aims to reconstruct low-resolution images with higher quality. Unlike classical convolutional neural networks (CNN), SRCNN is a structure that directly takes a low-resolution image as input and directly estimates the high-resolution counterpart of this image. While classical CNNs are generally used in tasks such as classification or object recognition, SRCNN works based on regression and aims to make each pixel clearer.

Two different models were structured in this study: Model 1 was trained on both the RGB space and only the Y channel of the YCbCr space. Model 2 was developed using only the Y channel (luminance) in the YCbCr color space. The Y channel carries the brightness information that the human visual system is most sensitive to. Based on this principle, Model 2 focused only on the Y channel, estimated the Y channel as output, combined it with the original Cb and Cr channels to create the final color image.

The SRCNN model consists of three layers: the first layer extracts low-level features from the input image with a large kernel size (9×9); the second layer transforms these features with a 1×1 kernel size; and the third layer reproduces the high-resolution image with a 5×5 kernel. A total of 684 training images were used in the training process of both models, the training was carried out for 10 epochs, and a batch size of 16 images was used in each iteration. Mean Square Error (MSE) was used as the loss function, and optimization was performed with the Adam algorithm by applying different learning rates to different layers.

At the end of the training process, the model performance was evaluated with the PSNR (Peak Signal-to-Noise Ratio) metric. PSNR measures the difference between the high-resolution real image and the model output and is expressed in dB (decibel); the higher this value, the better the quality of the model output. At the end of each epoch, PSNR was calculated on the test data and the development was observed. After the training was completed, the validation images were given to the model; Those that made predictions in the RGB space produced direct output, while those that made predictions only in the Y channel converted the image back to RGB and saved the results.

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 2.2.1

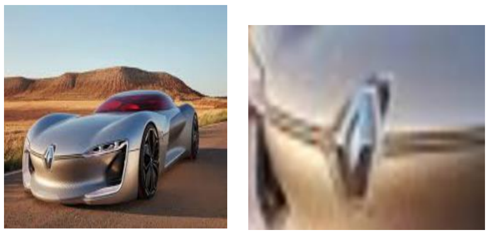

Figure 3.1 High Resolution Image

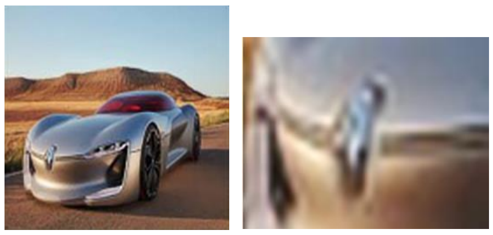

Figure 3.2 Predicted Image by Model-1 RGB

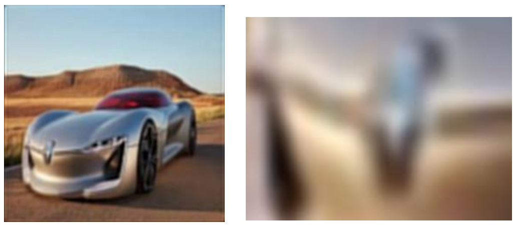

Figure 3.3 Predicted Image by Model-1 Y

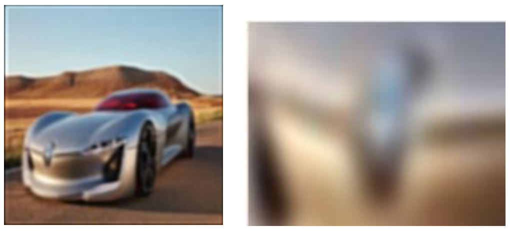

Figure 3.4 Predicted Image by Model-2

TensorFlow-based Model-1 performed better when trained on RGB images (30.37 dB)(Figure 3.1])compared to when trained only on the Y channel (Figure 3.2)(27.99 dB). This contrasts with some literature where Y-channel inputs often outperform RGB; however, in this particular implementation, RGB inputs allowed the model to learn from both structural and chromatic features, potentially improving reconstruction.

Both Model-1 and Model-2 use the Y channel, but Model-1 (TensorFlow) achieved significantly better PSNR (27.99 dB) than Model-2 (PyTorch), which scored 24.48 dB. This suggests that the TensorFlow implementation of SRCNN may be more effective, potentially due to differences in initialization, optimization strategy, or preprocessing steps.

All models were trained under the same conditions: 10 epochs with a batch size of 16, ensuring a fair comparison. The results indicate that both input type and framework implementation play a crucial role in final SR performance.

4. Discussion

When the PSNR results obtained by the three different SRCNN models applied in this study on the test data were evaluated, it was observed that Model-1 (30.37 dB) based on TensorFlow and using RGB input showed the highest success. On the other hand, when the models trained using only Y channel were compared, Model-1, again based on TensorFlow, achieved a higher PSNR value than Model-2 developed with PyTorch (27.99 dB vs. 24.48 dB). This difference may be due to the differences in weight initialization methods and optimization algorithms of different frameworks. All models were trained with the same epoch (10) and batch size (16). These results reveal that both the input type (RGB or Y channel) and the framework used (TensorFlow or PyTorch) have a decisive effect on the final performance in super-resolution problems.

This study has some limitations. First of all, the models were trained for only 10 epochs. In addition, the evaluation was only made on the PSNR metric. The small amount of training data limited the generalization ability of the model and limited the performance compared to studies conducted with larger datasets.

Various technical and structural difficulties were encountered during the model training and development process. First, some of the codes used were old and incompatible with current library versions. Especially due to errors caused by version differences in PyTorch libraries, the codes were converted to suit modern APIs. Some problems were also encountered on the dataset side. In old datasets widely used in the literature, the dimensions of the input and label images were usually arranged to be 256x256. However, there were incompatibilities in the dimensions of the images in the special dataset used in this study. Preprocessing steps such as resizing and cropping were carefully applied to ensure full alignment between the input and target images.

5. Conclusion

In this study, three separate SRCNN models developed using different input types (RGB and Y channel) and different frameworks (TensorFlow and PyTorch) were compared. The performance of the models was evaluated using the PSNR (Peak Signal-to-Noise Ratio) metric, which is widely used in image super-resolution problems. According to the results, Model-1, which is based on TensorFlow and uses the RGB channel, became the most successful model by achieving the highest PSNR value (30.37 dB) compared to other models. When the models using only the Y channel were compared, Model-1 developed with TensorFlow again showed higher performance than Model-2 based on PyTorch (27.99 dB vs. 24.48 dB).

During this project, important information was gained about the impact of different data formats on model performance, technical differences between frameworks, and implementation details. In addition, valuable experience was gained in dealing with practical challenges such as modernization of old codes, alignment of data sizes, and model training under limited hardware conditions.

By increasing the model training time (with more epochs and data augmentation), higher accuracy can be achieved. Visual quality can be evaluated more comprehensively by using not only PSNR but also more advanced quality metrics such as SSIM (Structural Similarity Index). The generalization capacity of the models can be tested by training with larger and more diverse datasets. In addition, comparative analysis with different super-resolution architectures (e.g. EDSR, VDSR, ESPCN) will reveal the advantages and disadvantages of SRCNN more clearly.

References

[1] https://paperswithcode.com/paper/image-super-resolution-using-deep

[2]https://www.researchgate.net/figure/Layered-structure-of-SRCNN-model_fig1_365759432

[3] https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

[4] https://debuggercafe.com/srcnn-implementation-in-pytorch-for-image-super-resolution/

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

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

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