FSRCNN: Fast Super-Resolution Convolutional Neural Network

CS 7180 Advanced Perception Project Report

Author: Sukhrobbek Ilyosbekov
Date: September21, 2025

Abstract

This project implements the Fast Super-Resolution Convolutional Neural Network (FSRCNN) introduced in the "Accelerating the Super-Resolution Convolutional Neural Network" by Dong et al, a deep learning model for single image super-resolution. FSRCNN accelerates the traditional SRCNN approach by performing upsampling at the end of the network using a deconvolutional layer, rather than preprocessing the input image. The implementation includes a complete training pipeline with the DIV2K dataset, evaluation on standard benchmarks, and comprehensive utilities for image processing and metric computation. The model architecture consists of five main components: feature extraction, shrinking, mapping, expanding, and deconvolution layers. This implementation demonstrates significant improvements in inference speed while maintaining competitive image quality metrics (PSNR/SSIM) compared to traditional bicubic interpolation methods.

Introduction and Prior Work

This project is based on the paper "Accelerating the Super-Resolution Convolutional Neural Network" by Chao Dong, Chen Change Loy, and Xiaoou Tang (2016). The paper introduced FSRCNN as an improvement over the original SRCNN (Super-Resolution Convolutional Neural Network) by addressing the computational bottlenecks inherent in the original design. The original SRCNN where introduced in the paper "Learning a Deep Convolutional Network for Image Super-Resolution" by Dong et al. had several limitations: used large filter sizes throughout network and had significant computational overhead.

FSRNN tries to resolve these limitations by using a deconvolutional layer at the end instead of preprocessing, reducing feature dimensions in the middle layers and using smaller filters and fewer parameters. As a result, this significantly reduces inference time, which can be used in real-time applications.

Methods

Model Architecture

The FSRCNN model consists of five distinct components:

1. **Feature Extraction Layer**
    - 5×5 convolutional layer with d output channels (default: 56)
    - PReLU activation function
    - Extracts low-level features from the input image
2. **Shrinking Layer**
    - 1×1 convolutional layer reducing channels from d to s (default: 12)
    - PReLU activation
    - Reduces computational complexity for mapping layers
3. **Mapping Layers**
    - m sequential 3×3 convolutional layers (default: 4 layers)
    - Each layer maintains s channels with PReLU activation
    - Performs non-linear mapping of features
4. **Expanding Layer**
    - 1×1 convolutional layer expanding channels from s back to d
    - PReLU activation
    - Prepares features for final upsampling
5. **Deconvolution Layer**
    - 9×9 transposed convolutional layer
    - Stride equal to the scale factor (2, 3, or 4)
    - Performs end-to-end upsampling to target resolution

CLI commands

The project includes PDM scripts accessible via **pyproject.toml** and these commands are used for training and model evaluation:

- **pdm train** command executes training for T91 dataset using default parameters and epoch 1000.
- **pdm train_d64_s16_m6** command executes training for T91 dataset overrides model parameters *d=64*, *s=16* and *m=6* where *d* is number of feature maps for the feature extraction/expanding layers, *s* number of feature maps for shrinking/mapping layers and *m* number of mapping layers (3x3 convs) operating on s channels.
- **pdm train_div2k** executes training for DIV2K dataset and using overridden parameters *d=64*, *s=16* and *m=6* and epoch 5000.
- **pdm eval** command executes a model inference on the specified folder where it contains images for evaluation. It will show metrics for each image and inference time.
- **pdm infer** command executes a model inference for a single image or directory where it contains images and save output images in the specified path. It can also generate bicubic images to compare

These commands use Python scripts defined in the **scripts** folder. The full list of command parameters is documented in these files (train.py, infer.py, and eval.py).

Training Pipeline

1. Load and preprocess training images (DIV2K or T91 datasets)
2. Extract random patches during training
3. Convert RGB to YCbCr and use Y channel
4. Forward pass through FSRCNN model
5. Compute MSE loss against ground truth
6. Backpropagation and parameter updates
7. Periodic validation on DIV2K and test on Set5 datasets

Inference Process

The inference pipeline processes full-resolution images:

1. Input processing: load RGB image, apply modular crop, convert to YCbCr color space and extract Y channel for supre-resolution
2. Model inference: normalize Y channel to [0, 1] range and forward pass-through trained model
3. Output: upscale Cb and Cr channels using bicubic interpolation, combine super-resolved Y with upscaled CbCr, convert back to RGB and save image file.

Hyperparameters

The model has the following hyperparameters:

- ***scale*** - Upscale factor (2, 3, 4). Default is 4.
- ***d*** - Number of feature maps for the feature extraction/expanding layers. Default is 56
- ***s*** - Number of feature maps for shrinking/mapping layers. Default is 12.
- ***m -*** Number of mapping layers (3x3 convs) operating on s channels. Default is 4.

Results

The model was evaluated on standard super-resolution benchmarks such Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) using the enhanced evaluation script. The model was trained on DIV2K and T91 datasets.

Training results

- DIV2K dataset with default parameters (d=64, s=16, m=6, scale=4) and 5000 epoch:
  - Best PSNR: 28.00 dB
- T91 dataset with default parameters (d=56, s=12, m=4, scale=4) and 1000 epoch:
  - Best PSNR: 24.94 dB
- T91 dataset with fine-tuned parameters (d=64, s=16, m=6, scale=4) and 1000 epoch:
  - Best PSNR: 25.45 dB

Model evaluation

Set14 dataset results (upscaling factor=4):

Image	FSRCNN PSNR (dB)	FSRCNN SSIM	Bicubic PSNR (dB)	Bicubic SSIM	Inference time (s)
baboon.png	21.42	0.5201	21.15	0.4668	0.2658
barbara.png	24.45	0.7302	23.88	0.6858	0.0036
bridge.png	23.78	0.6375	23.23	0.5822	0.0040
coastguard.png	24.76	0.5415	24.19	0.4997	0.0433
comic.png	21.29	0.6884	20.46	0.6125	0.0037
face.png	31.03	0.7705	30.35	0.7445	0.0034
flowers.png	25.74	0.7830	24.33	0.7279	0.0032
foreman.png	30.49	0.8970	28.17	0.8566	0.0010
lenna.png	30.12	0.8371	28.67	0.8054	0.0008
man.png	25.46	0.7272	24.48	0.6754	0.0012
monarch.png	28.55	0.9154	26.34	0.8810	0.0034
pepper.png	31.54	0.8527	29.44	0.8224	0.0010
ppt3.png	22.74	0.8744	20.75	0.8130	0.0029

zebra.png	24.80	0.7584	23.02	0.6927	0.0029
Average	**26.15**	**0.7524**	**24.89**	**0.7047**	**0.0243**

Set14 FSRCNN improvements over Bicubic:

- **PSNR**: +1.26 dB (5.1%)
- **SSIM**: +0.0477 (6.8%)

comic.png original

comic.png bicubic 20.46 dB

comic.png FSRCNN 21.29 dB

Figure 1: The "comic" image from Set14 with upscaling factor 4

The “comic” image has noticeable improvements over bicubic: excellent edge sharpening and minimal artifacts in flat regions. It is hard to see these differences here need to open these images on full screen.

Set5 dataset results (upscaling factor=4):

Image	FSRCNN PSNR (dB)	FSRCNN SSIM	Bicubic PSNR (dB)	Bicubic SSIM	Inference time (s)
baby.png	31.99	0.8849	30.68	0.8571	0.1233
bird.png	31.28	0.9158	29.10	0.8737	0.0312
butterfly.png	23.50	0.8449	21.00	0.7388	0.0022
head.png	31.09	0.7724	30.38	0.7458	0.0022
woman.png	27.36	0.8835	25.35	0.8324	0.0020
Average	**29.05**	**0.8603**	**27.30**	**0.8095**	**0.0322**

Set5 FSRCNN improvements over Bicubic:

- **PSNR**: +1.78 dB (6.4%)
- **SSIM**: 0.0507 (6.3%)

Figure 2: bird.png original

bird.png bicubic 29.10 dB

bird.png FSRCNN 31.28 dB

The "bird" image in Set5 shows a complex natural scene with fine detail after upscaling by 4x. Overall, the improvement is good, with some loss of very fine detail on the petals. The FSRNN outperforms the bicubic model by approximately +2 dB. When these images are opened fullscreen, we see noticeable visual differences, such as sharp textures.

Model strengths:

- **Sharp Edge Preservation:** The model effectively preserves and enhances edge details in natural images
- **Texture Enhancement**: Fine textures in an image like "bird.png" show clear improvement over bicubic interpolation.
- **Fast inference**: Real-time processing capability suitable for practical applications

Model limitations:

- **Smooth Region Artifacts:** Occasional over-smoothing in uniform regions
- **Color Consistency**: Minor color shift artifacts when reconstructing from Y channel

Reflection

This project provided me with valuable experience. Over the course of its implementation, I acquired technical skills in working with PyTorch and the convolutional neural network (CNN) architecture. I learned about creating a high-quality codebase for a deep learning project and applying best practices. I used the PDM dependency manager to install packages, especially the PyTorch CUDA version. I learned how to implement end-to-end image processing workflows, optimize and reduce training time, and understand the importance of the right metrics and datasets for benchmarks. I gained experience solving

problems such as convergence and hyperparameter tuning, creating efficient data loading and augmentation systems, and working with the YCbCr color space for luminance-focused training.

Acknowledgements

This project was completed with assistance from various resources:

- **PyTorch Documentation**: Essential for understanding framework capabilities. https://docs.pytorch.org/docs/stable/index.html
- **Original FSRCNN Paper**: Primary reference for architectural details and training procedures. https://arxiv.org/pdf/1608.00367
- **Original FSRCNN Implementation Repository**: explored how the model was implemented originally and compared with custom implementation https://github.com/yjn870/FSRCNN-pytorch
- **Super-Resolution Datasets**: DIV2K, T91, Set5, and Set14 datasets for training, evaluation and testing. https://www.kaggle.com/datasets/soumikrakshit/div2k-high-resolution-images
- **OpenCV Library**: Image processing utilities for color space conversion. https://docs.opencv.org/4.x/index.html
- **PDM**: Setting up PDM for dependency management and development environment. https://pdm-project.org/en/latest

Project Extensions

This implementation includes several significant extensions beyond the basic FSRCNN requirements, qualifying for additional points:

- **Enhanced Dataset Implementation**: The original paper used T91 and 100 BMP images for training. In addition to T91, I trained/evaluated with **DIV2K** (which has 900 high resolution images) and standard test sets (**Set5/Set14**), yielding more robust results than the classic T91-only setup.
- **Fine-tuned Model Architecture**: Systematically experimented with architectural parameters *d* (feature channels), *s* (shrinking channels), and *m* (mapping layers) and achieved model improvement over the original paper implementation.
- **Comprehensive Evaluation**: Added per image PSNR, SSIM, and inference timing metrics.
- **Engineering Quality**: The codebase follows a clear **src/** layout, uses **PDM** for environments, **Black** for formatting, and thorough docstrings/headers across modules and scripts for reproducibility.