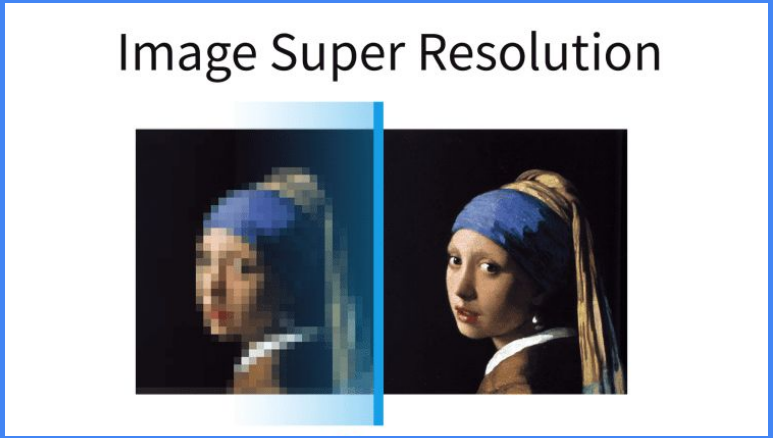
Super Resolution via Learning Sparse Representation

Haoze Wu
James Yu
Thomas Yu



Problem

- Low-resolution images exist (pixelated when scaling up)
- Implications in various fields
  - medical imaging
  - surveillance
  - satellite imagery analysis
- Super-resolved images -> more accurate analysis and decision-making

Solution

- Image super-resolution
  - Generating high-resolution images from low-resolution inputs

- Some previous attempts pretty successful
- Use previous models as benchmark for our model
- Research area always looking for improvements to super-resolved images

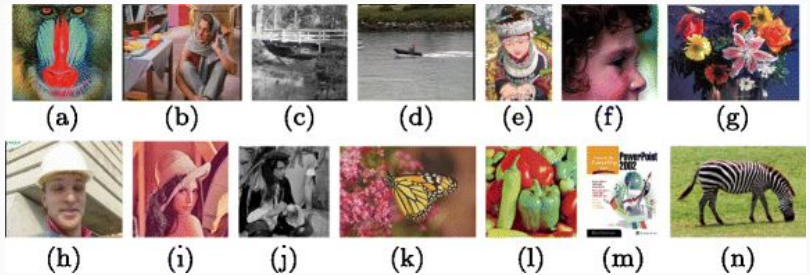
Project Goal

- Task: generating a super-resolution (SR) image from a single low-resolution input image.
- Scope of approach: (sparse) dictionary learning

Metrics and Datasets Used

Quantitative Metrics:

- RMSE for validation
- PSNR (peak signal-to-noise ratio) for benchmark comparison
  - Typical score: 20-50 dB
  - Higher PSNR = better quality of super-resolved image
- SSIM (structural similarity index measure)
  - Range -1 to 1 with scores above 0.9 indicating excellent similarity

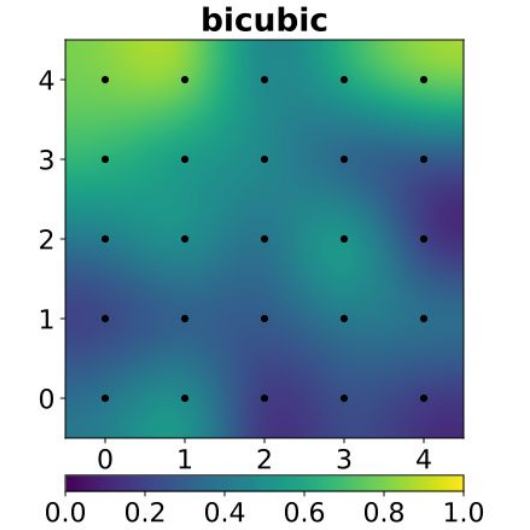


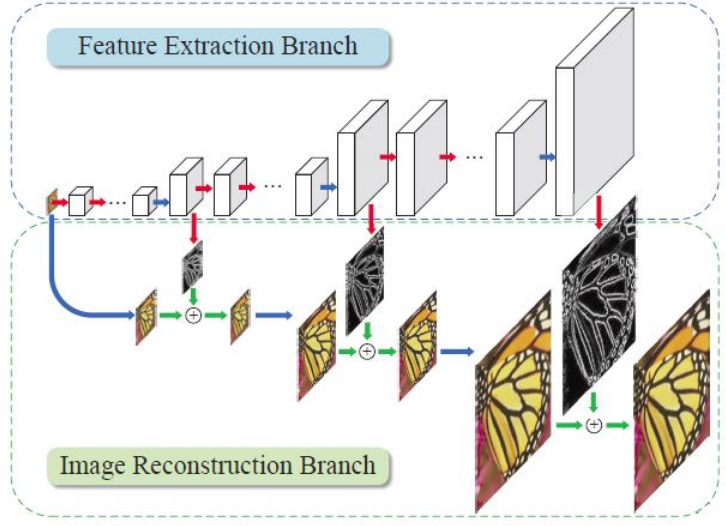
Datasets:

- Kaggle natural dataset
- Set14
- Set5

Benchmark Models

- Bicubic interpolation
- LapSRN (Laplacian Pyramid Super-Resolution Network)
  - State of the art
  - Increases image resolution by 8x





Sparse Representation

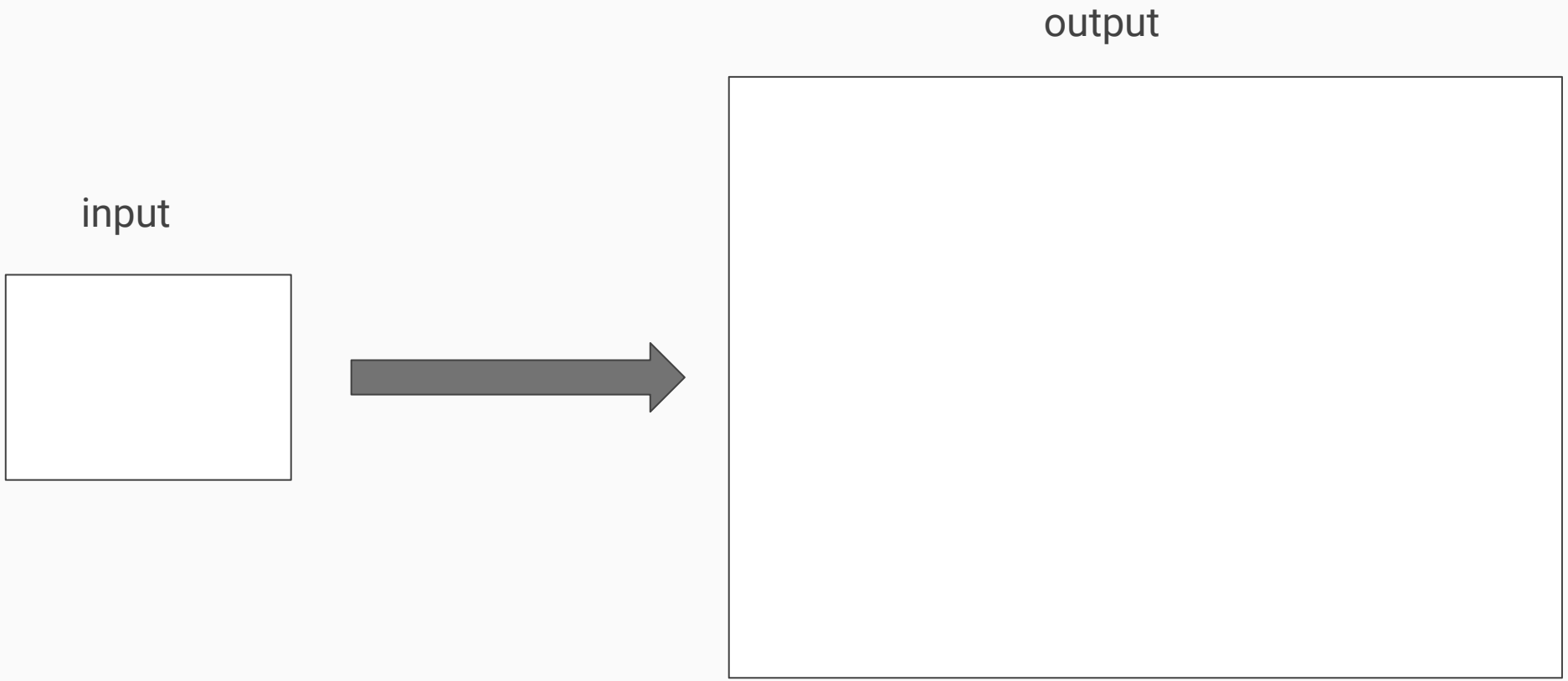
High-resolution image loses information when downscaling

downsample

Recovering high-resolution image requires prior knowledge

recover

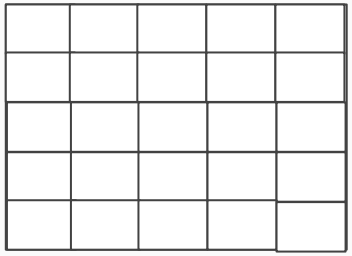
Step 1: divide the image into small patches (3x3, 5x5, etc.)



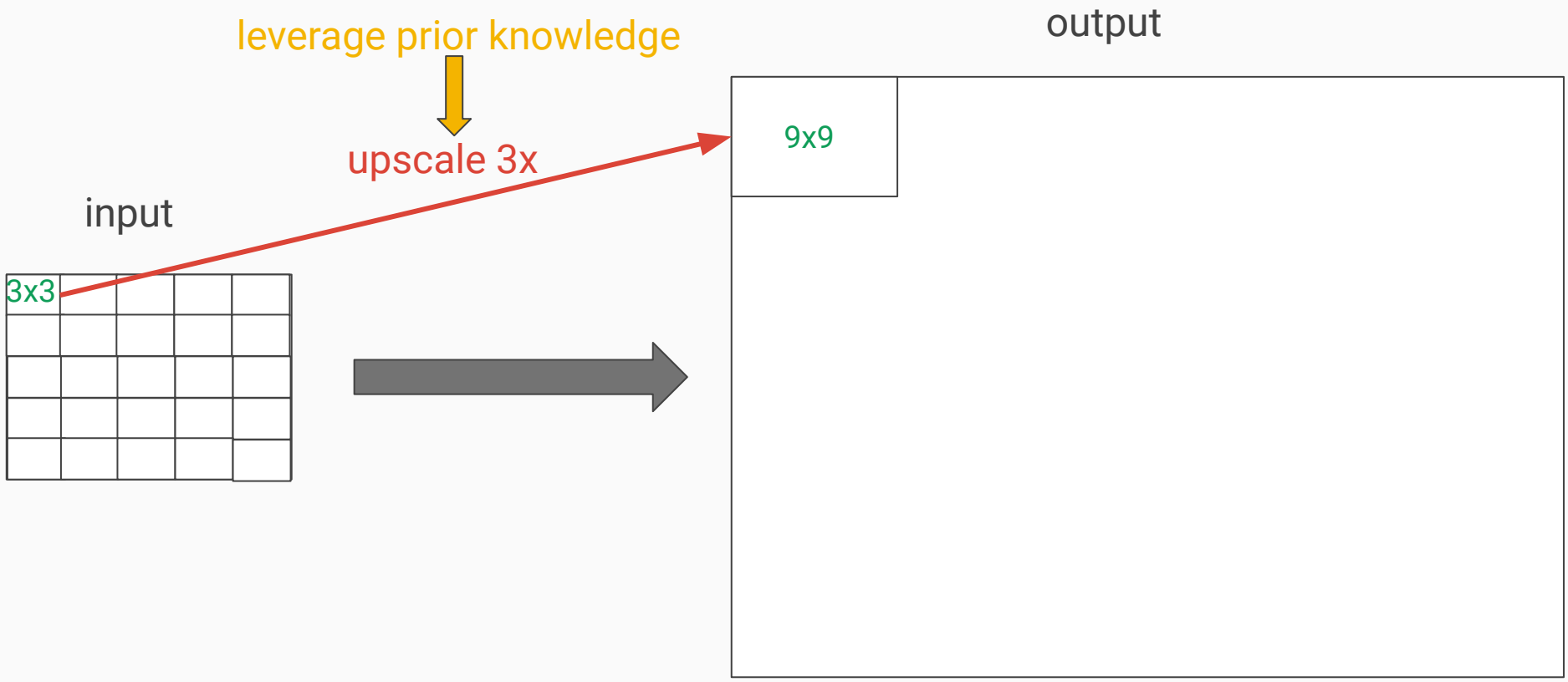
Step 1: divide the image into small patches (3x3, 5x5, etc.)

output

input

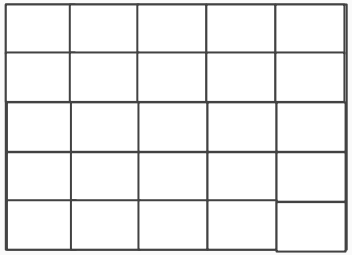


Step 2: leverage prior knowledge to upscale a patch

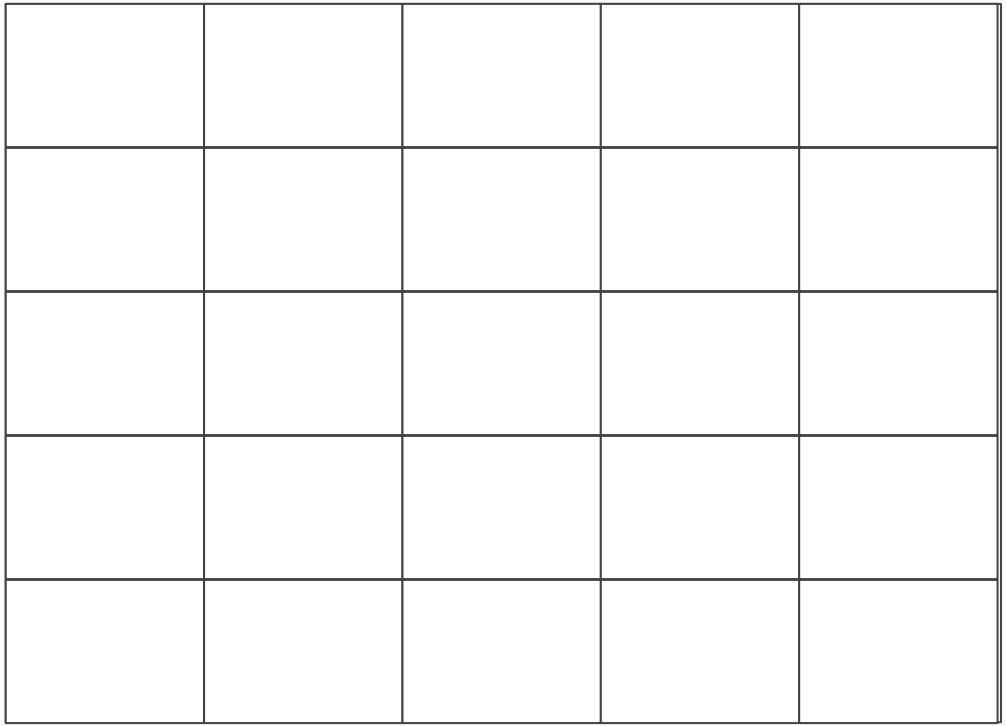


Step 3: repeat for all patches (stitch high resolution patches)

input



output

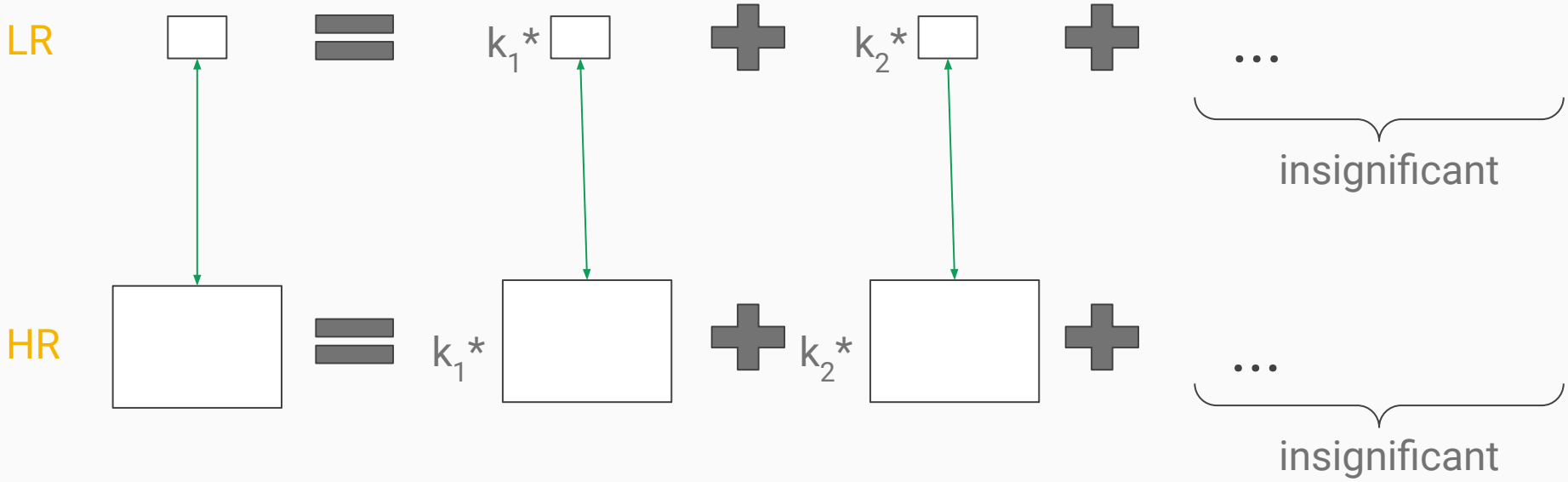


A high-level algorithm description

- Parameter: scale_factor, patch_size
- Input: train_hr, train_lr, validation_lr
- Output: validation_hr
- Algorithm:
  - train a model to convert a small patch (height = patch_size) to a larger patch (height = patch_size * scale_factor)
  - use the model to recover lr patch to hr patch
  - stitch the hr patches (make it smooth) to get the hr image result

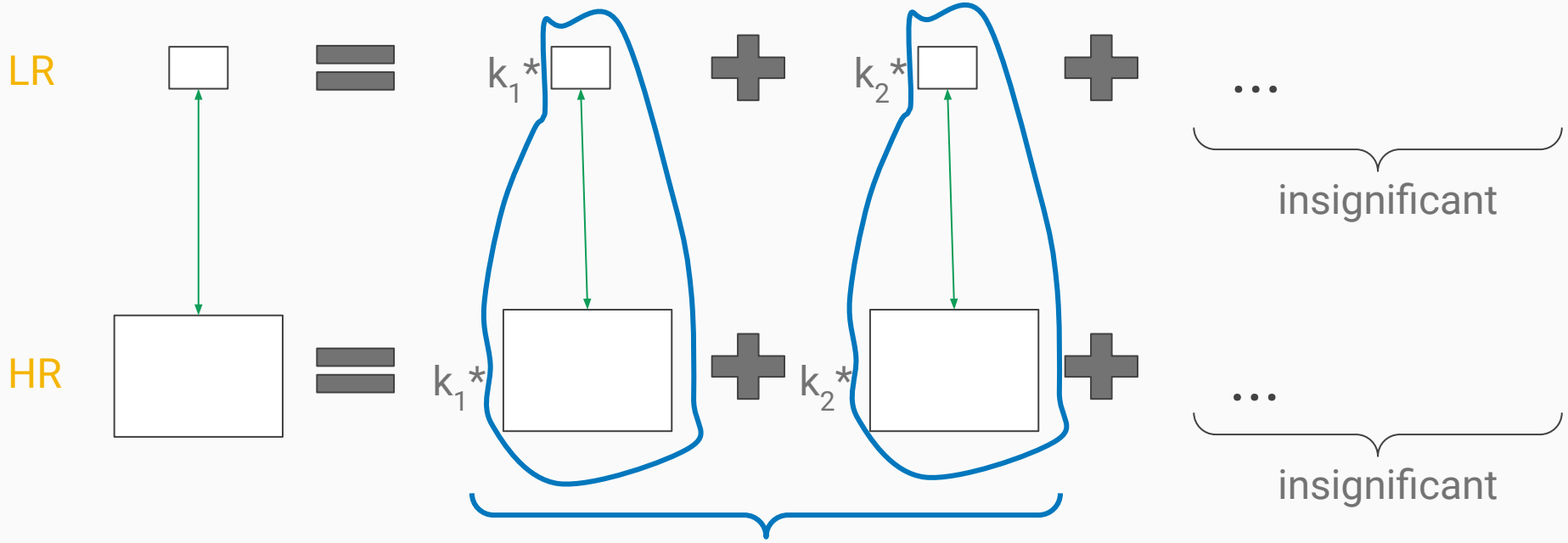
Intuition: most information is captured by a sparse representation

a pair of patches (LR & HR) in training data:



Intuition: most information is captured by a (linear) sparse representation

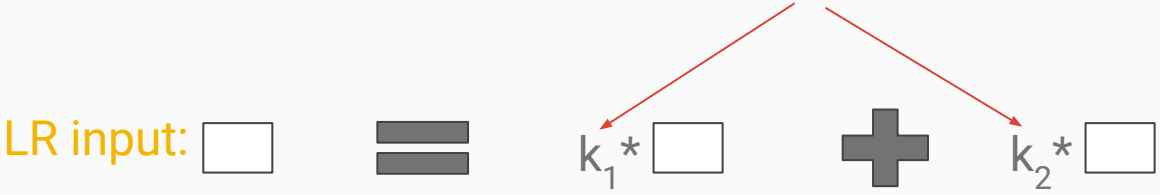
a pair of patches (LR & HR) in training data:



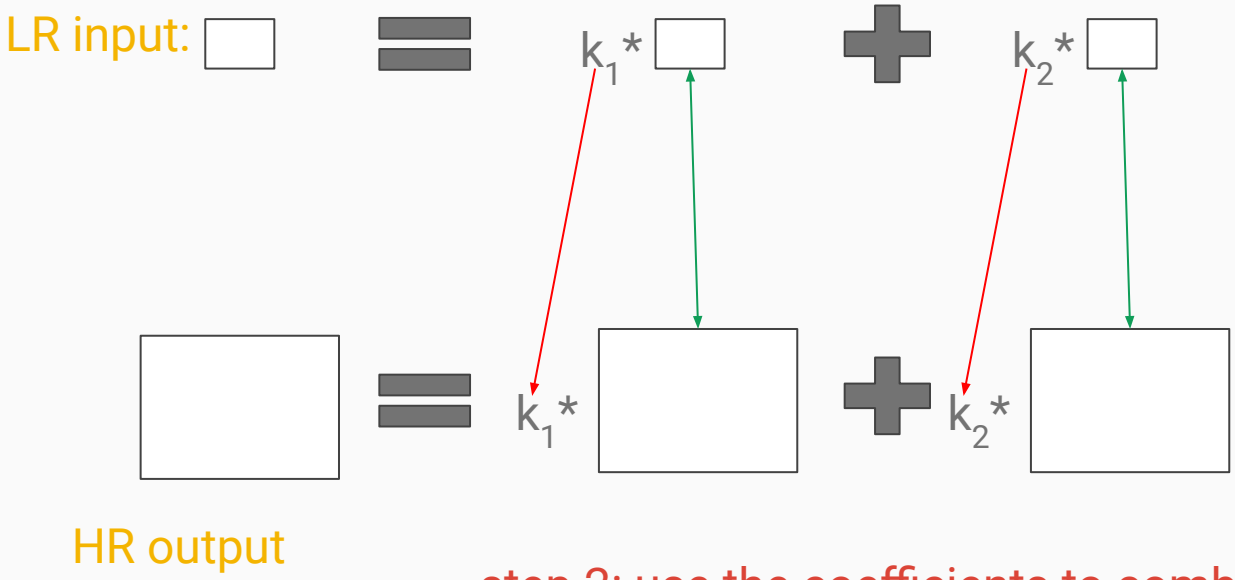
The (sparse) prior knowledge learned from the training data

Use the learned linear combination to recover the HR patch

step 1: compute the coefficients for the LR atoms



Use the learned linear combination to recover the HR patch

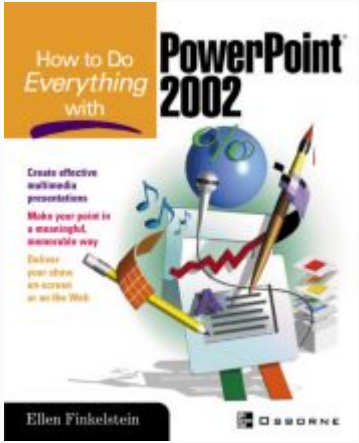


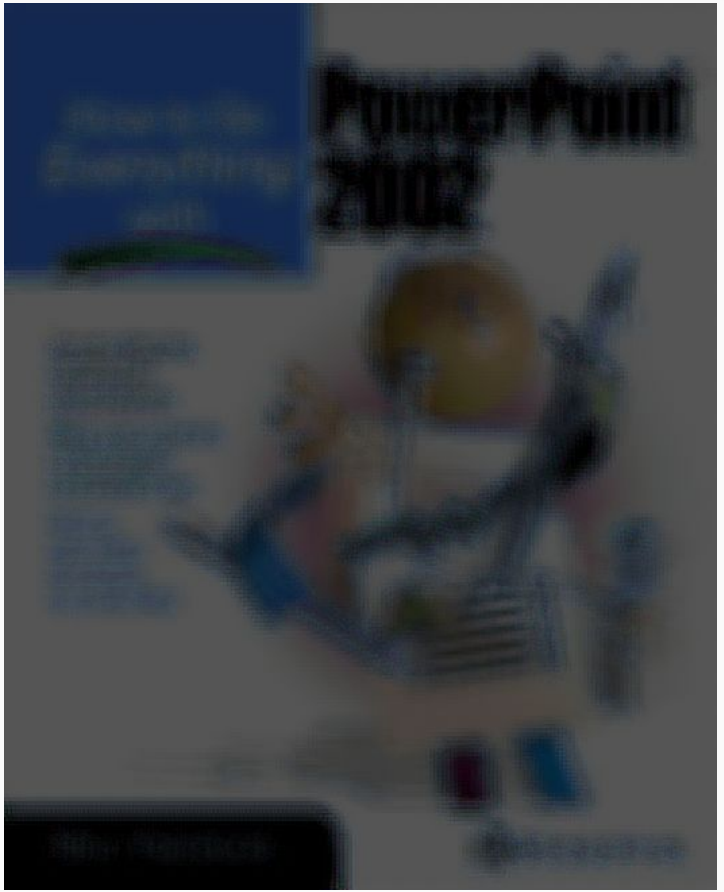
step 2: use the coefficients to combine the HR atoms

Implementation and Performance

- Aforementioned algorithm is used as K-SVD (but the paper is for the denoise task)
- We implement with sklearn in Python
- We use the average stitching to fuse the computed patches
- As a baseline
  - Randomly sampled 10,000 patches from the Kaggle natural dataset to train 100 atoms
  - Applied validation on the set14 dataset with scale factor = 3

Example





unsatisfactory resolution,
color changes,
...

Improvement

- Image Super-Resolution as Sparse Representation of Raw Image Patches
- Improvements:
  - In the training, the goal is set to get the SR to minimize the difference of the *features*
  - Improve the fused high-resolution image, by finding the closest image satisfying the reconstruction constraint: Y = DHX
- Advantages:
  - Better resolution
  - Requires much less samples in training dataset

Results

Bicubic Interpolation

bicubic

RMSE: 8.68
PSNR: 29.63
SSIM: 0.9825

STOA: LapSRN

LapSRN

RMSE: 7.37
PSNR: 30.95
SSIM: 0.9858

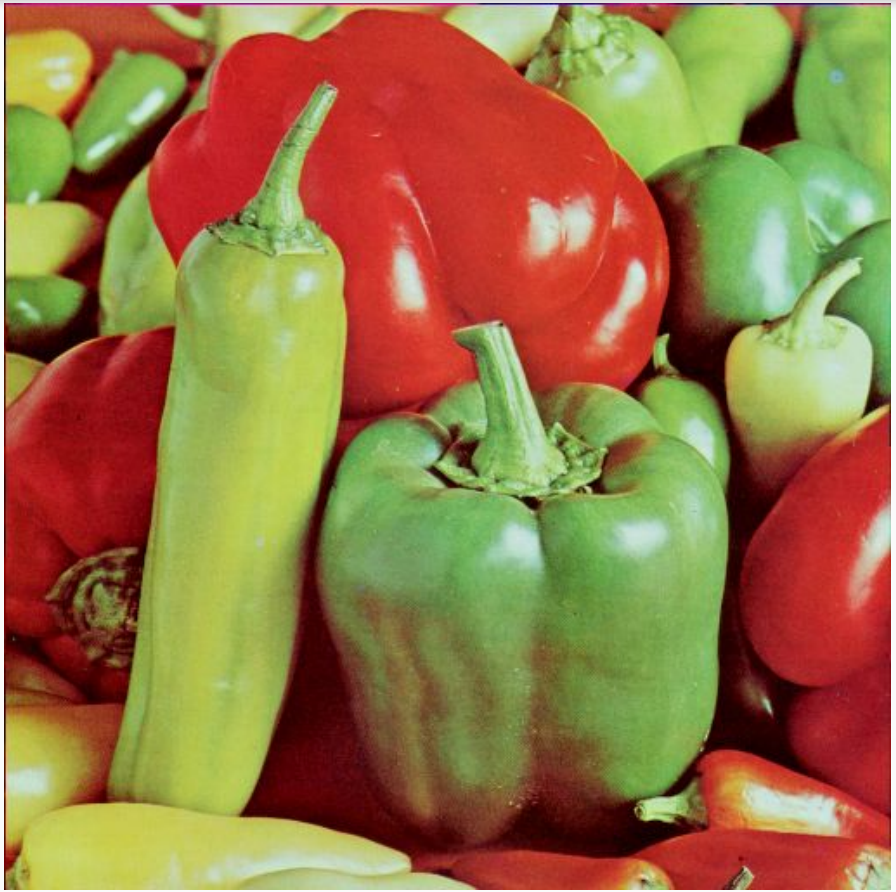
Sparse Representation

SR

RMSE: 7.585
PSNR: 21.85
SSIM: 0.9372

Original Pepper Image

HR Image



LR Image

Comparison for Pepper Image

Bicubic Interpolation
RMSE: 8.68
PSNR: 29.63
SSIM: 0.9825

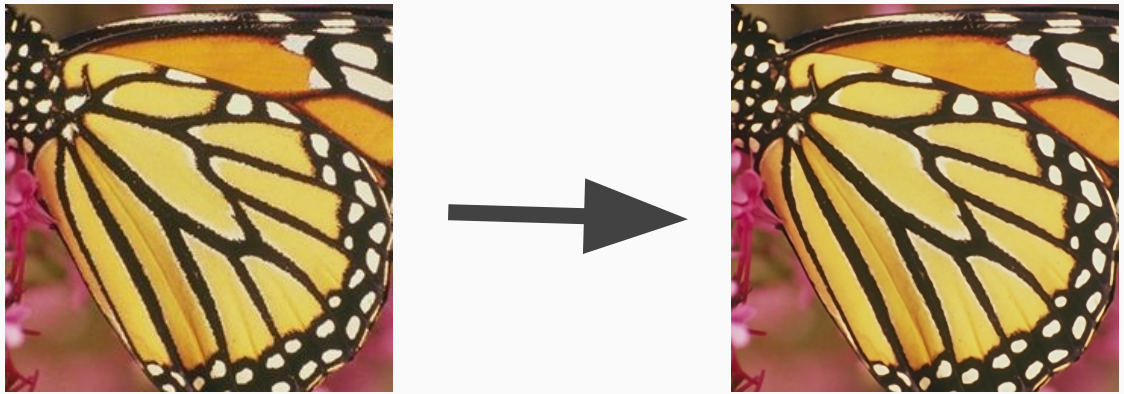
LapSRN
RMSE: 7.37
PSNR: 30.95
SSIM: 0.9858

Sparse Representation
RMSE: 7.585
PSNR: 21.85
SSIM: 0.9372

PSNR and SSIM Scores (Updated in paper)

Set14 Image	Bicubic (PSNR)	LapSRN (PSNR)	Sparse (PSNR)	Bicubic (SSIM)	LapSRN (SSIM)	Sparse (SSIM)
flowers	24.33	36.80	14.93	0.8514	0.9923	0.7140
monarch	26.74	37.65	23.46	0.9641	0.9978	0.9493
baboon	20.66	33.82	18.85	0.6766	0.9868	0.6665

LapSRN example:



Discussion

- SR is better than LapSRN in RMSE, but worse in PSNR and SSIM. Why?
  - RMSE is sensitive to pixel-wise differences and may be influenced more by sharpness improvements.
  - PSNR and SSIM are more sensitive to local structural changes.
- We plan to make adjustments to our model to try and improve the performance, and do further comparison and analysis in the Set14 and Set5 datasets.

Thank you!

Question?

PSNR and SSIM Scores

Set5 Image	Bicubic (PSNR)	LapSRN (PSNR)	Sparse (PSNR)	Bicubic (SSIM)	LapSRN (SSIM)	Sparse (SSIM)
baby.png	44.78	43.75	?	0.9986	0.9977	?
bird.png	43.40	41.20	?	0.9983	0.9975	?
butterfly.png	35.63	32.16	?	0.9967	0.9928	?
head.png	41.99	42.81	?	0.9941	0.9936	?
woman.png	40.11	37.72	?	0.9981	0.9970	?

LapSRN example: