

Single Image Super Resolution (SISR)

Achint Chaudhary
D Awadhesh Tanwar



SISR

- Recover/Construct finer detail from Low-Resolution
- Obtain a learned LR to HR Image mapping using Convolutional Neural Network based approach
- Pixel-wise loss minimization between LR & HR images (Mean Square Error).
- PSNR Metric – Peak Signal To Noise Ratio
 - $PSNR = 10 \times \log_{10} \left(\frac{MAX^2}{MSE} \right) = 20 \times \log_{10} \left(\frac{MAX}{\sqrt{MSE}} \right)$
 - $MAX = 255$



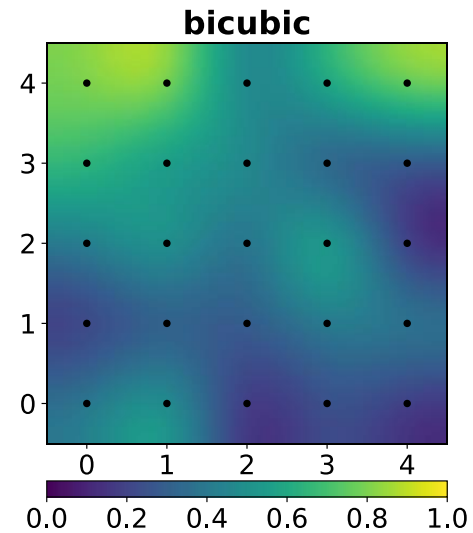
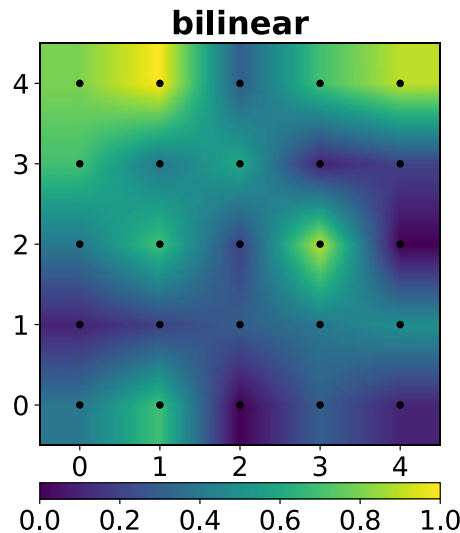
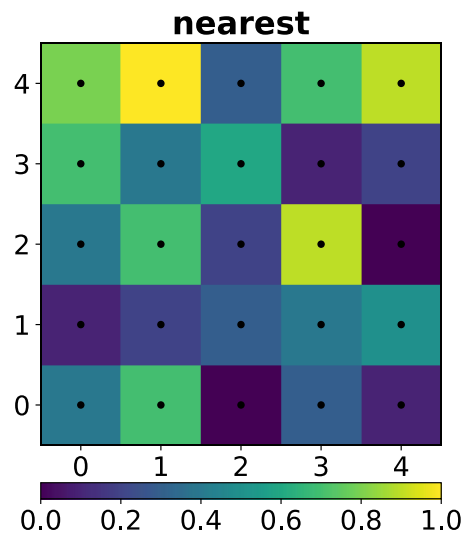
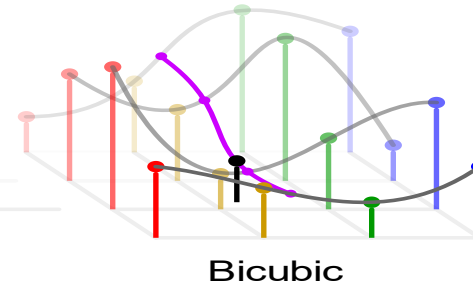
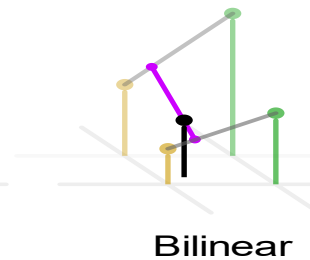
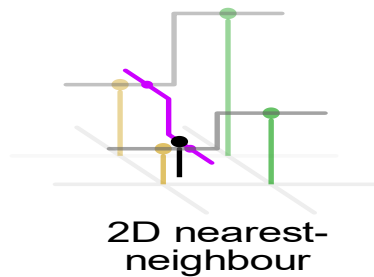
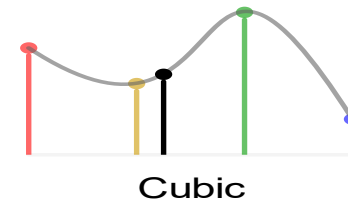
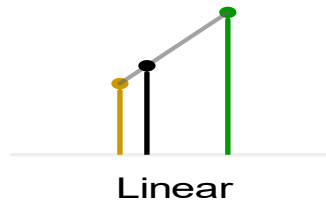
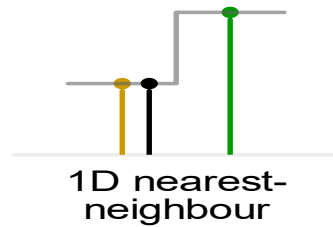
Applications of SISR

- Satellite Imaging
- Medical Imaging
- Security Imaging





Classic Interpolation





Training Datasets

- DIV2K: 800 images of 2K Resolution
- Flickr2K: 2650 2K images
- Challenge on Learned Image Compression (CLIC):
 - CLIC-professional: 585 images of 480p to 2K
 - CLIC-mobile: 1048 images of 320p to 2K



Testing Datasets

- PIRM Perceptual Image Super-Resolution Challenge
- BSDS100: 100 images
- Set5 & Set14: common SISR evaluation datasets
- Urban 100: FHD images of Urban scene
- Manga 109: Images of Anime comic covers
- Validation sets of DIV2K, CLIC



Data Preparation

- Datasets are of High-Resolution images
- Obtained Low Resolution via bicubic down-sampling
- Scale Factor: 4
- Patches
 - LR: 32x32
 - HR: 128x128
- Images of different resolution and formats



Experimental Setup

- Intel i7 processor
- 16 GB DDR4 RAM
- 12 GB NVIDIA 1080Ti GPU

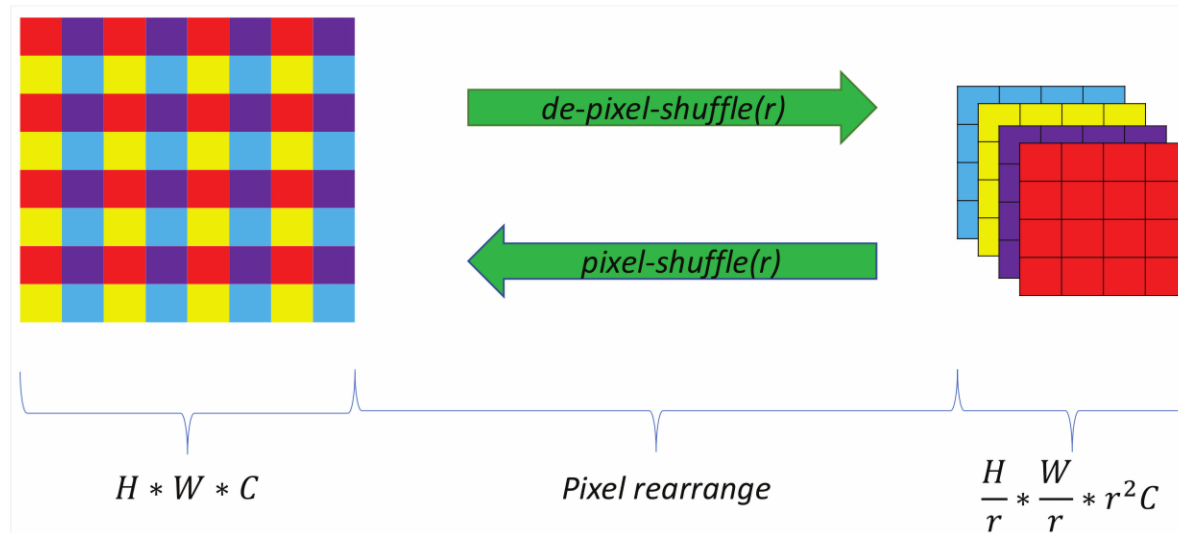
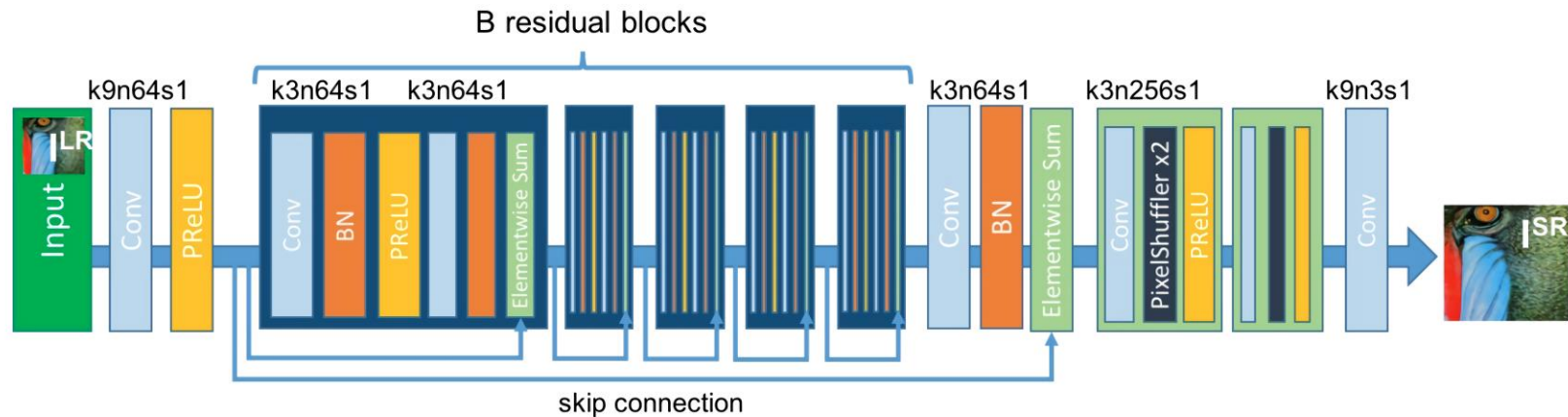


Model Selection (Exp 0.)

- Type 1: Takes interpolated LR image as input
 - High resource intensive & difficult to converge
- Type 2: Takes LR input, works on LR feature space, later find a learned up-sampling
 - Less resource intense & converges easily
- SRResNet is best in terms of accuracy & resource usage



SRResNet Architecture



Output images & PSNR values



30.83 dB
LR
(Bicubic)



32.50 dB
SRResNet



∞ dB
Ground Truth

PSNR Gain 1.67 dB



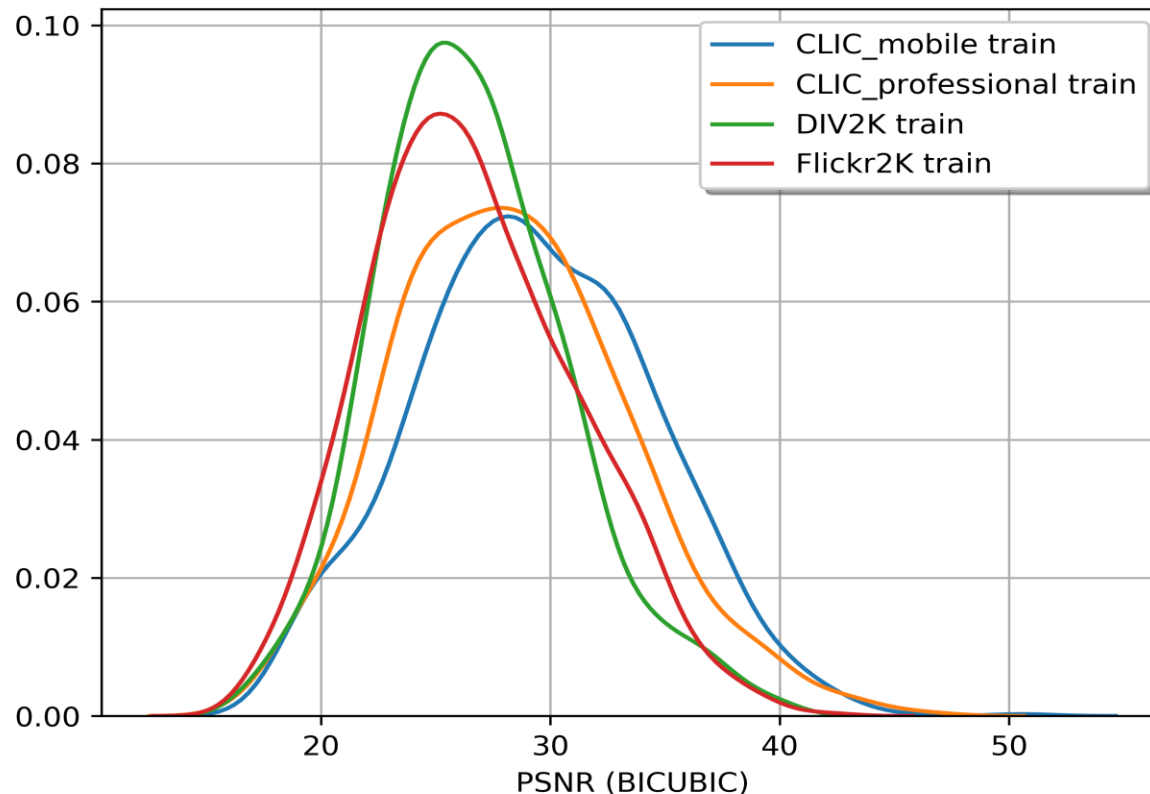
Mean PSNR Gain over Bicubic on separate SRResNet instances

Test/Valid Dataset	DIV2K	Flickr2K	CLIC-professional	CLIC-mobile	Min Diff
CLIC-mobile	1.43	1.28	1.34	1.16	0.09
CLIC-professional	1.64	0.88	1.49	1.20	0.15
DIV2K	1.55	1.11	1.27	1.12	0.28
BSDS100	1.73	1.39	1.44	1.32	0.29
Manga109	4.01	3.75	3.39	3.03	0.26
PIRM	1.40	1.27	1.25	1.08	0.13
PIRM	1.37	1.24	1.19	1.02	0.13
Set5	1.79	1.65	1.57	1.38	0.14
Set14	2.77	2.33	2.45	1.98	0.32
Urban100	2.00	1.86	1.78	1.57	0.14



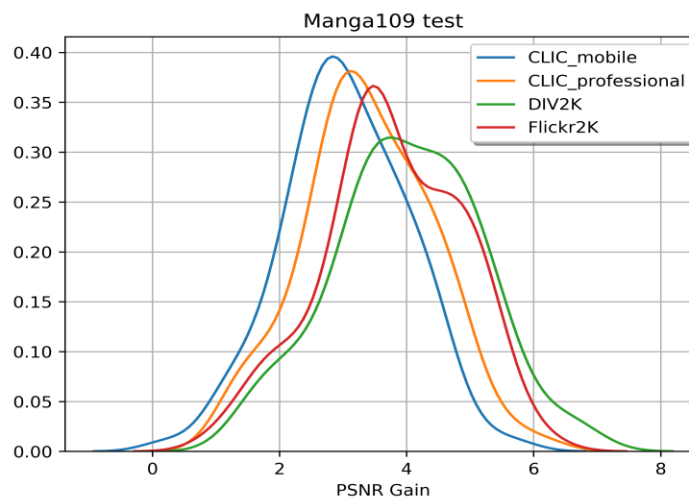
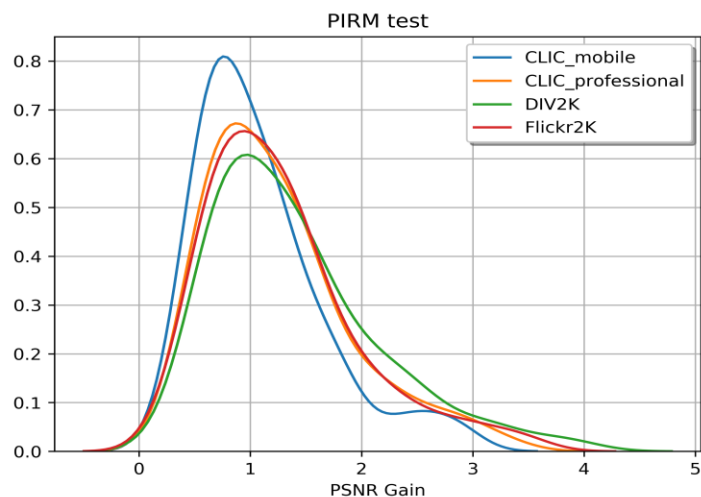
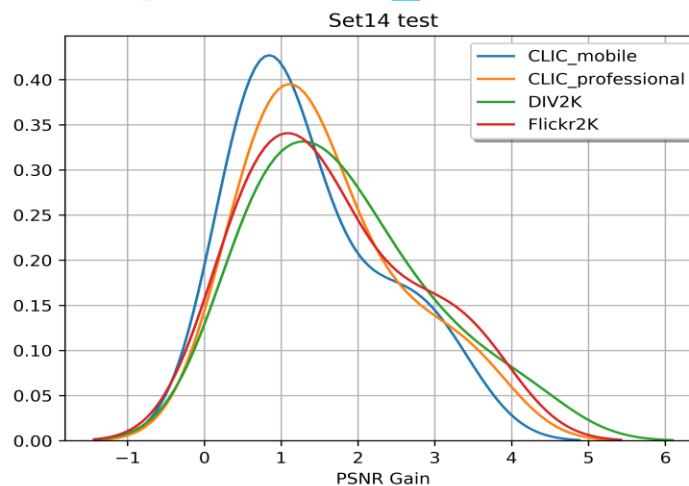
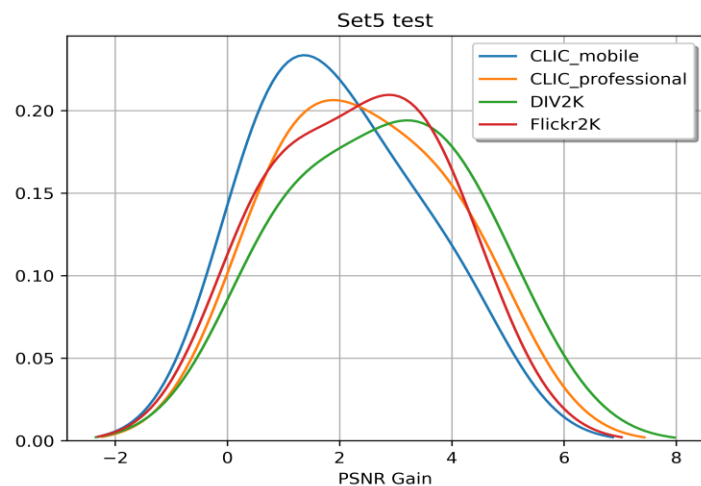
Data-Set Ranking (Exp 1.)

- Why (& How) training on different dataset matter?
- Fine details are difficult to recover from Bicubic





Data-Set Ranking (Exp 1.)

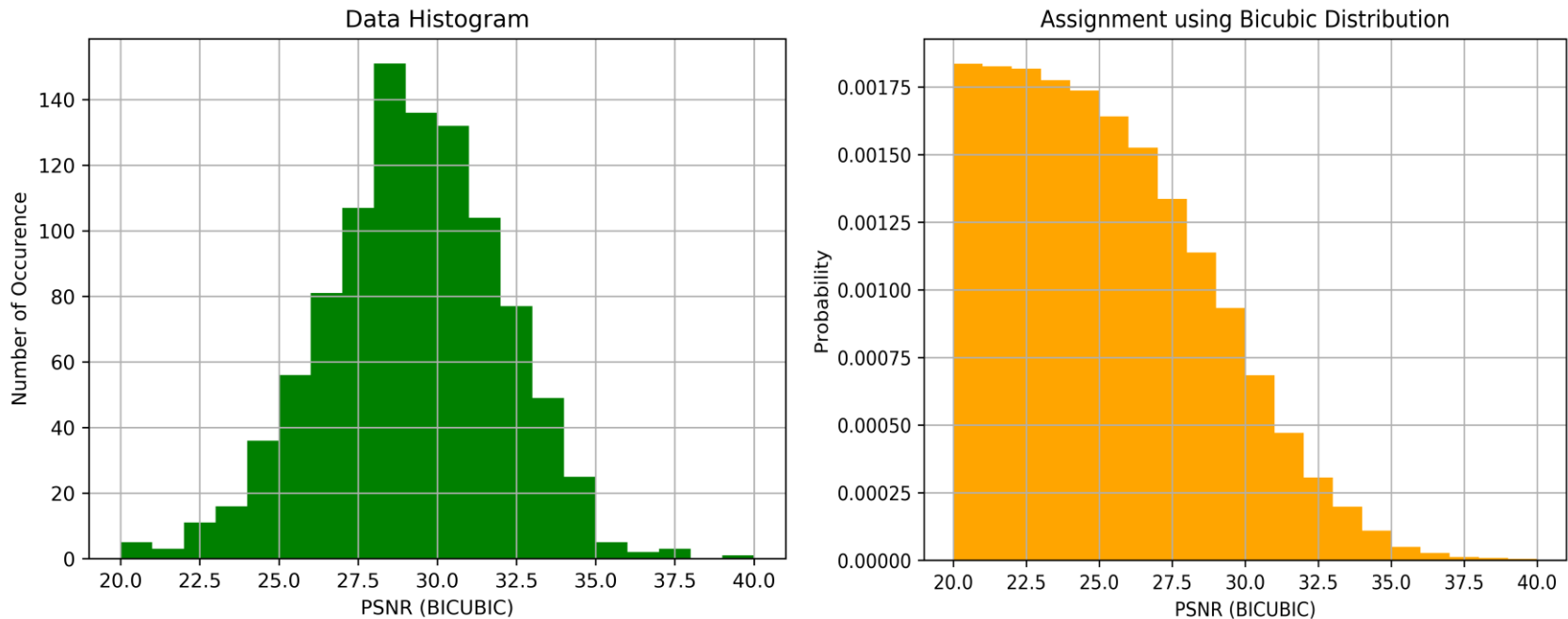




Data Sampling (Exp 2.)

- Batches from disk are taken uniformly for training
- Observation from Exp 1., motivates us to use some **specific sampling** mechanism
- Like Exp 1. does rank datasets, some **Scoring** can be assigned to individual images/patches

Sampling Algorithm



- Observed data distribution of patches is Gaussian
- Less detailed patches are easier to reconstruct using bicubic
 - Results in relatively high PSNR value
 - and vice-versa
- Thus, low PSNR patches should be preferred over high PSNR ones



Algorithm Pseudocode

Input: K, N be chosen sample size & number of patches

Step 0: Assign each patch, uniform probability

$$P_i = \frac{1}{N}$$

While (Model not Converges)

Step 1: Sample K patches, proportional to values in P for each patch

Step 2: Train model on above collected sample

Step 3: Calculate the PSNR gain over these patches

step 4: $P_{\text{sample}} = \sum_{i \in \text{Sample}} P_i$

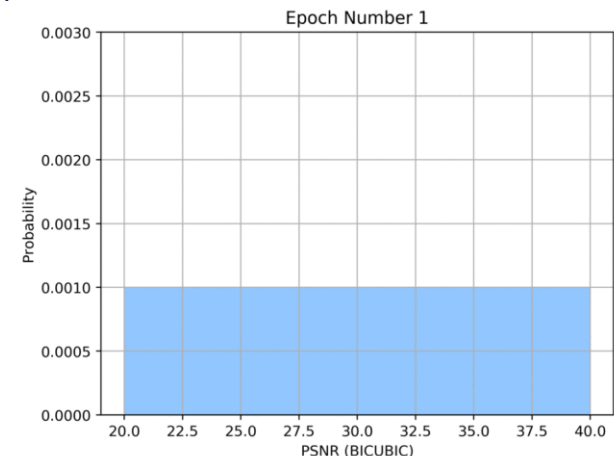
Create Histogram of B bins, using PSNR Gain distribution

Patch i is mapped to bin $b(i)$; frequency of corresponding bin is $f(b(i))$

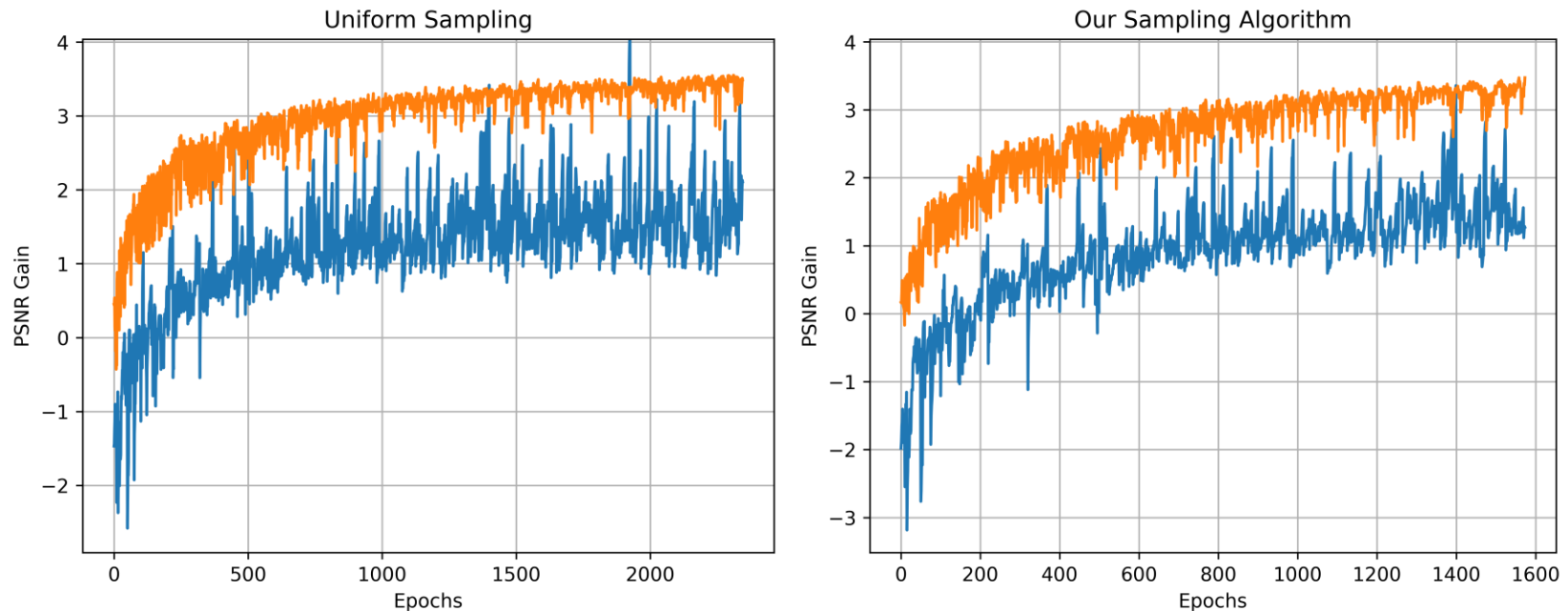
$$\text{val}(i) = \sum_{j=b(i)}^B \text{width}(j) * f(j)$$

$$\text{AggArea} = \sum_{i \in \text{Sample}} \text{val}(i)$$

$$\forall i \in \text{Sample} (P_i = \frac{\text{val}(i)}{\text{AggArea}} * P_{\text{sample}})$$



Convergence Performance



- Our sampling algorithm results in **faster convergence**
- Using Patch level approach instead of full image allows
 - Fine grained control
 - No need of extra inference from SRResNet



Core Findings

- Dataset-Ranking (Inter-Dataset) approach helps obtaining better performance
- Data-Sampling (Intra-Dataset) approach further enables faster convergence
- Combining all datasets will saturate in practice
- Techniques like our algorithm can be used to train vision models on extremely large datasets



Conclusion

- Analysed & Benchmarked various SISR models from the literature
- Characterize training datasets for best performance
- Proposed & empirically verified a data sampling algorithm for faster convergence of the model



Future Work

- Similar **Dataset Ranking & Data Sampling** approach can be used for other low-level vision applications like Compression Artefacts Removal, De-noising.,
- Reduction in training time can be further obtained by other probability initialization unlike uniform



References

- CLIC Challenge - <https://www.compression.cc/challenge/>
- Agustsson, E., & Timofte, R. (2017). NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 126-135).
- Matsui, Yusuke, et al. "Sketch-based manga retrieval using manga109 dataset." *Multimedia Tools and Applications* 76.20 (2017): 21811-21838.
- Lai, Wei-Sheng, et al. "Deep laplacian pyramid networks for fast and accurate super-resolution." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- Martin, David, et al. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics." Vancouver:: lccv, 2001.
- Blau, Yochai, et al. "The 2018 PIRM challenge on perceptual image super-resolution." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- Yang, Wenming, et al. "Deep learning for single image super-resolution: A brief review." *IEEE Transactions on Multimedia* (2019).
- Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- Du, Chen, et al. "Orientation-Aware Deep Neural Network for Real Image Super-Resolution." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.



Thank You!

Questions?



Vision Framework

- No common framework to compare vision models
 - Written in different languages (MATLAB, R, Python, Lua)
- So we created a common framework in python
 - Even higher level abstraction on top of Keras, TensorFlow
 - Define model as dictionary
 - Model architecture can be defined over default values
 - Layers & Functions in literature, not in Keras, are added
- Snapshot of two SISR models on next slide
 - Just 56 lines for VDSR & SRResNet models from literature
 - All training & testing arguments are via command line

Vision Framework

```

58 def vdsr_block(X_input):
59     X = Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='same', use_bias=True, kernel_regularizer=keras.regularizers.l2(0.0001))(X_input[0])
60     X = ReLU()(X)
61     return X
62
63 def vdsr_last_block(X_input):
64     X = Conv2D(filters=len(channel_indx), kernel_size=(3,3), strides=(1,1), padding='same', use_bias=True, kernel_regularizer=keras.regularizers.l2(0.0001))(X_input[0])
65     X = ReLU()(X)
66     return X
67
68 VDSR = { 'name' : 'VDSR' ,
69         # Block Links
70         'block' : [(1,1+1) for i in range(B)] + [(B,B+1),],
71         'block_sub': ['vdsr_block' for i in range(B)] + ['vdsr_last_block',],
72         # Merge Links
73         'merge' : [(0,B+1,B+2),],
74         'merge_sub': ['Add'],
75     }
76
77
78 def gen_residual_block(X_input):
79     X = Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='same', use_bias=True)(X_input[0])
80     X = BatchNormalization()(X)
81     X = PReLU(shared_axes=(1,2))(X)
82     X = Conv2D(filters=64, kernel_size=(3,3), strides=(1,1), padding='same', use_bias=True)(X)
83     X = BatchNormalization()(X)
84     X = Add()([X, X_input[0]])
85     return X
86
87 def gen_last_block(X_input):
88     X = Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='same', use_bias=True)(X_input[0])
89     X = UpSampling2D(size=2, interpolation='upsample_interpolation')(X)
90     # X = PixelShuffle(size=2)(X)
91     X = PReLU(shared_axes=(1,2))(X)
92     return X
93
94 SRResNet = { 'name' : 'SRResNet',
95             # Convolution Links
96             'conv' : [(0,1), (B+2,B+3), (B+5+int(np.ceil(np.log2(scale))), B+5+int(np.ceil(np.log2(scale)))+1),],
97             'conv_sub': ['Conv2D', 'Conv2D', 'Conv2D',],
98             'conv_par': [{'filters':64, 'kernel_size':(9,9), 'padding':repr('same'), 'use_bias':True},
99                         {'filters':64, 'kernel_size':(3,3), 'padding':repr('same'), 'use_bias':True},
100                        {'filters':3, 'kernel_size':(9,9), 'padding':repr('same'), 'use_bias':True},],
101             # Advanced Activation Links
102             'aactv' : [(1,2),],
103             'aactv_sub': ['PReLU',],
104             'aactv_par': [{'shared_axes':(1,2)},],
105             # Block Links
106             'block' : [(2+1,2+1+1) for i in range(B)] + [(B+5+1,B+5+1+1) for i in range(int(np.ceil(np.log2(scale)))]],
107             'block_sub': ['gen_residual_block' for i in range(B)] + ['gen_last_block' for i in range(int(np.ceil(np.log2(scale)))]],
108             # BatchNormalization Links
109             'btnrm' : [(B+3,B+4),],
110             # Merge Links
111             'merge' : [(2,B+4,B+5),],
112             'merge_sub': ['Add'],
113         }
114

```



Patching Approaches

- HR Image is first Down-sampled and divided into patches for further processing
- HR Image is first divided into patches of desired size and then Down-sampled for further processing
- First approach is used