Wide Activation for Efficient and Accurate Image Super-Resolution

Yu et al. (2018)

UVA CS 6316: Machine Learning : 2019 Fall
Course Project: Deep2Reproduce @ https://github.com/qiyanjun/deep2reproduce/tree/master/2019Fall

Machine Learning 6316 - Fall 2019

Professor Qi

Team: Skynet
Colin Price, Jacob Dineen, Kazi Ashik Islam, A S M Ahsan-Ul Haque

Motivation

- Imaging modalities are limited by resolution at some point by hardware
  - Diffraction limit in microscopy
  - Pixel count in digital cameras

- In cases where greater resolution is required than hardware allows, it is desirable to increase resolution after image acquisition
  - Can an image be reliably up-sampled using machine learning without introducing noise/false information?
  - How can model parameters and design choices impact performance through tuning?

Related Work

- Super-resolution network architecture utilizes:
  - Upsampling layers (FSRCNN showed parametric deconvolution to improve implicit S^2 runtime)[7]
  - Deep, recursive neural networks (Generally 10 layers with 3x3 kernels, risk of overfitting)[7]
  - Skip connections (Shown to preserve low-level features in images)[5]
- Choice of convolutional method
  - Flattened, Group, and Depthwise Seperable convolution are popular choices[8,9,10]

Background

- CNN's are state of the art and are proven to upsample[1]
- Applications include security, medical, and satellite imaging[2]
- Early Single Image Super Resolution (SISR) use shallow CNNs (3-5 layers)[3]
  - Deep CNNs with low-level feature preserving skip-layers have been shown to be superior
  - These include SRDenseNet, RDN, and MemNet[4,5,6]
- In SISR, topics of interest include activation width, normalization methods, and convolutional kernels

Claim

Main Claims:

- Non-linear Relu activations stop info propagation from shallow layers to deeper layers
- Expanding features before nonlinear activations improves SISR performance over architecture overhauls due to an increased likelihood of low level (super resolution) features flowing through network toward final layer
  - Wide activation - efficient ways to expand features before nonlinear activations
  - Using CNNs for mapping low resolution images to their high res. counterparts

Target Task

- To improve upon SoTA SISR performance by introducing this concept of wideness/wider activations to be computed by expanding features before activation.

- Achieving higher resolution mappings without increasing model complexity (Deep NN parameters) - Real time processing - Need to keep computational overhead/parameters down.

An Intuitive Figure Showing WHY Claim

Figure 10: Comparisons of 'monarch' in Set14 for scale 2 with Gaussian kernel degradation. We can see that, given the degradation mismatch with that of training, the performance of EDSR decreases drastically.

3.4 Network Structure

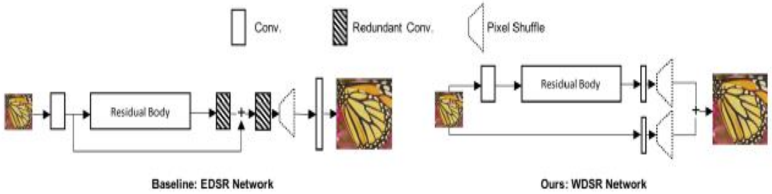


Figure 2: Demonstration of our simplified SR network compared with EDSR [19].

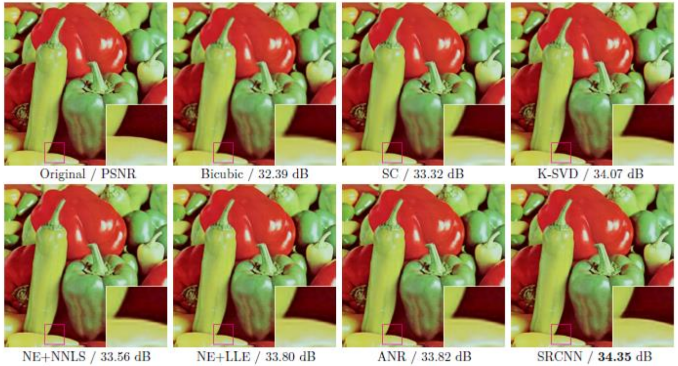
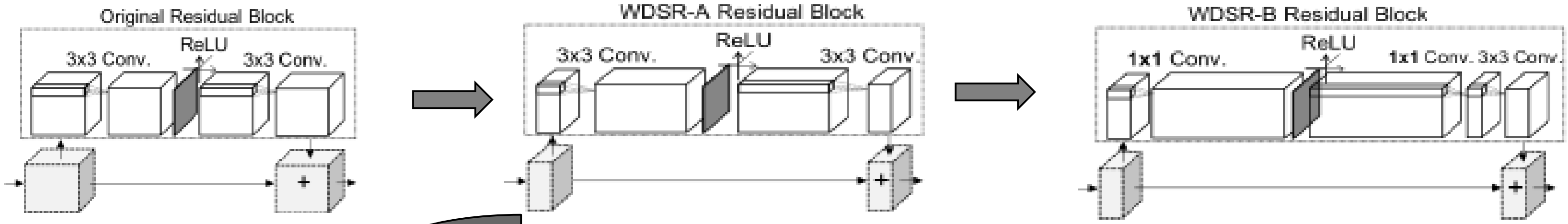


Fig. 7. "Pepper" image from Set14 with an upscaling factor 3.

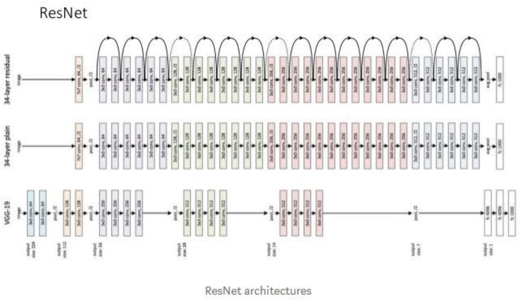
<u>Left Top:</u> Overview of results by SotA SISR models in 2018 [12]
<u>Left Bottom:</u> Yu et al. proposal to simplify existing SISR network architectures [13] - This is the network we will be reproducing.
<u>Right Top:</u> Early work on SISR [11]

More Figures + Proposed Solution



Novelty: Wider feature space before activation layer. Low level information isn't lost as network is propagated through.

Novelty: Wider feature space before activation layer than WDSR-A + low-rank convolution stack.



Proposed Solution #1

Wide Activation: WDSR-A

a. Slim identity mapping pathway with wider channels before activation in each residual block.
    i. Wide activation - efficient ways to expand features before nonlinear activations. Need quick upsampling techniques for real time processing.
    ii. Why? Existing architectures were over-parameterized.
    iii. What methods were carried over? Skip Connections are essential. Batchnorm is neglected as accuracy was too sensitive.

Proposed Solution #2

Efficient Wider Activation: WDSR-B

a. Expands on WDSR-A.
b. Tried: additional feature expansion via group convolution and depthwise separable convolution.
    i. Group convolution: Convolving over a portion of the input channels and concatenating.
    ii. Depthwise separable convolution: Each channel is kept separate when convolving + a 1x1 spatial filter.
c. Produced low-rank convolution coupled with even wider activation
    i. Results suggested this decayed feature activations, backing hypothesis.
d. Wider activation > baselines, given different parameter budgets
e. Loosely inspired by Inverted Residuals (A parameter efficient convolution method)

Data Summary

- Model trained on DIV2K (DIVerse 2K Resolution images) dataset
- Default split of DIV2K dataset
  - 800 training images
  - 100 validation images
  - 100 testing images (not publicly available)
- Authors used 800 images for training, 10 images for validation during training
- Trained models evaluated on 100 validation images

DIV2K: https://data.vision.ee.ethz.ch/cvl/DIV2K/

Implementation

- Cropped 96x96 RGB input patches and bicubic downsampled image (both from HR image) used as training output-input pair
- Training data augmented with random horizontal flips and rotations
- Mean RGB values of training images subtracted from the input images
- PSNR (Peak Signal-to-Noise Ratio) is used as metric for validation
- ADAM optimizer used
- Batch size set to 16, learning rate initialized with maximum convergent value, halved every 2x105 iterations

REF: https://github.com/JiahuiYu/wdsr_ntire2018

Experimental Results

Efficiency and **Accuracy** comparison in terms of **NO. of parameters** and **validation PSNR** respectively, among baseline model EDSR and proposed models (with same NO. of residual blocks)

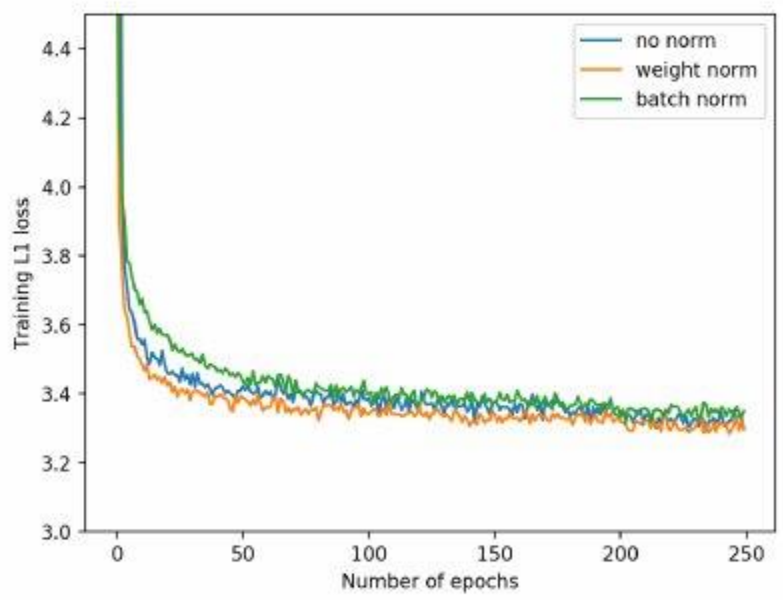
Residual Blocks	1			3		
Networks	EDSR	WDSR-A	WDSR-B	EDSR	WDSR-A	WDSR-B
Parameters	0.26M	**0.08M**	**0.08M**	0.41M	**0.23M**	**0.23M**
DIV2K (val) PSNR	33.210	33.323	33.434	34.043	34.163	34.205

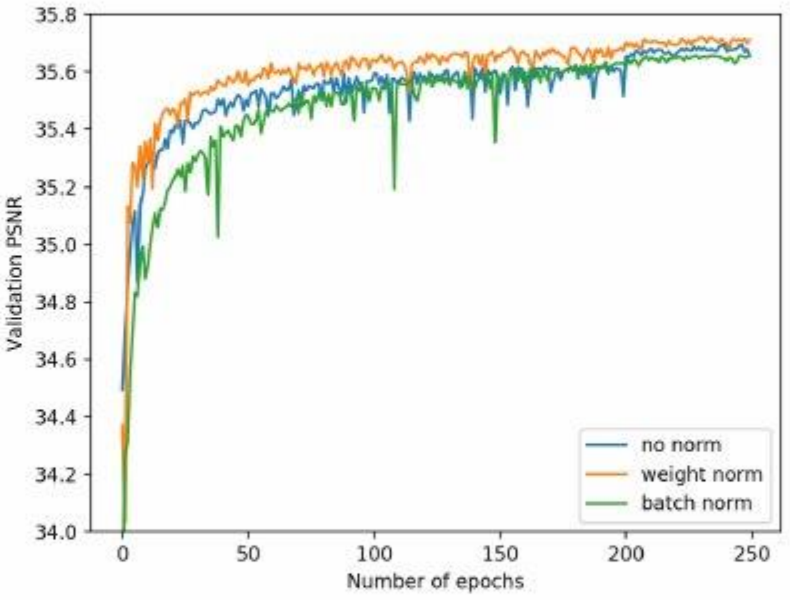
Residual Blocks	5			8		
Networks	EDSR	WDSR-A	WDSR-B	EDSR	WDSR-A	WDSR-B
Parameters	0.56M	**0.37M**	**0.37M**	0.78M	**0.60M**	**0.60M**
DIV2K (val) PSNR	34.284	34.388	34.409	34.457	34.541	34.536

*Higher PSNR value is better

Experimental Results (Continued)

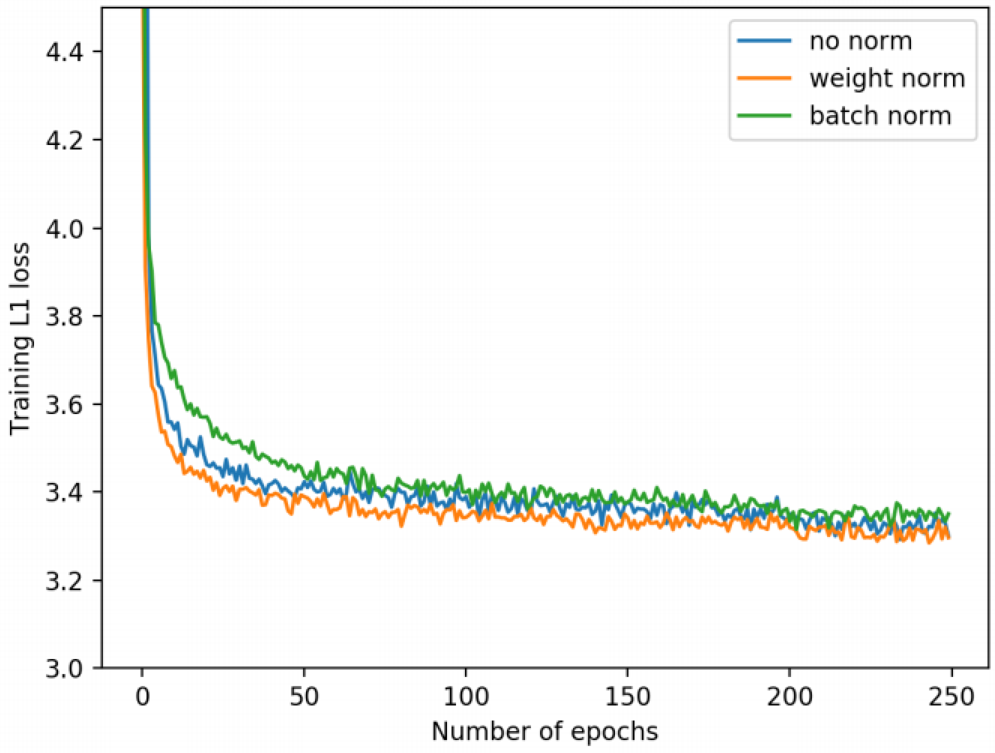
Effect of weight normalization compared to batch and no normalization (during training)





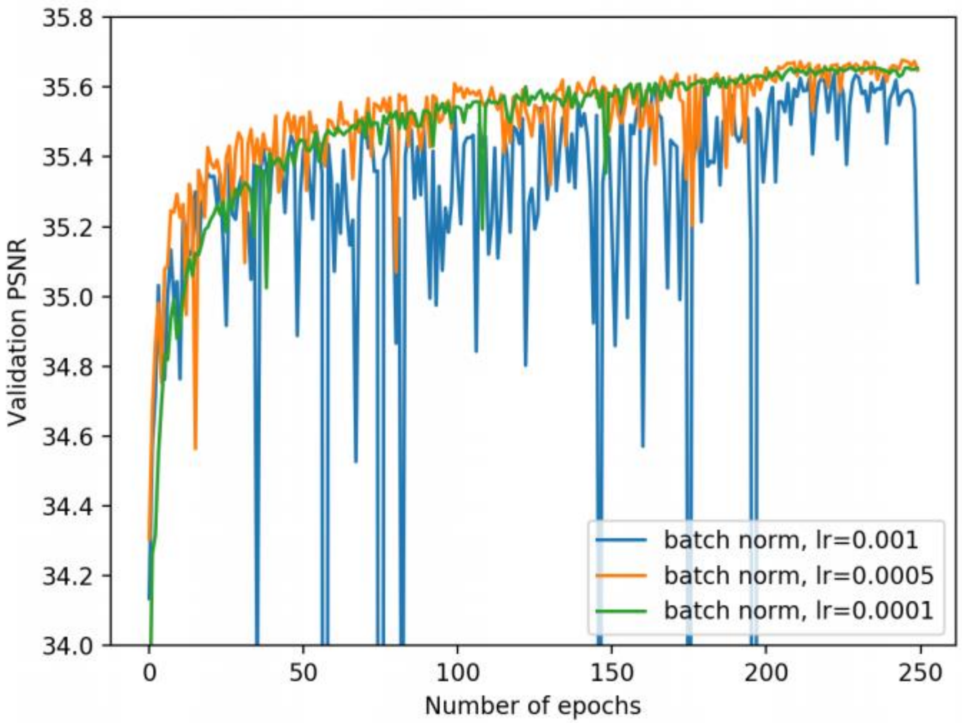
Experimental Analysis

- Weight normalization has faster convergence
- Batch normalization is unstable during testing
  - Possibly due to different mean, variance in test and batch-train data



Experimental Analysis (Cont’d)

- Batch normalization is unstable during testing
  - Not because of Learning Rate
  - Tried a variety of LR
  - Unstable PSNR for every LR



Conclusion

- Proposed 2 SR networks:
  - *WDSR-A:* Image features expanded before ReLU
  - WDSR-B: Image channels expanded using 1x1 convolution

- Experimented on DIV2K dataset
  - Weight Normalization works better than Batch-Norm or no norm

- Achieved better accuracy, keeping the same:
  - Parameters
    - Model complexity

Paper Reconstruction

CODE ADAPTED FROM: https://github.com/krasserm/super-resolution

Data - DIV2K Dataset

- Bicubic-Downsample each image in data set
    - Produces the highest quality downsample through weighted averaging of neighboring pixels
- Compare the super-resolved down-sampled image to the original image using peak signal-to-noise ratio (PSNR) as loss function
- 800 train, 100 test, 100 validation

Training Pipeline

- Calculate loss through mapping downsampled low resolution to high resolution image (x4 upscaled)
- Training images randomly flipped, roated, and cropped
- Loss Function: PSNR
  - Related to MSE
- Train EDSR, WSDR-A, WSDR-B models
- Adam Optimizer with learning rate schedule (PiecewiseConstantDecay)

$$\mathrm{MSE} = \sum_{\{\mathrm{all}\ i\}} \sum_{\{\mathrm{all}\ j\}} \left(\mathrm{Orig}(i,j) - \mathrm{Rcvd}(i,j)\right)^2$$

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}}\right)$$

EDSR - The Previous State-of-the-Art

```
def edsr(scale, num_filters=64, num_res_blocks=8, res_block_scaling=None):
    x_in = Input(shape=(None, None, 3))
    x = Lambda(normalize)(x_in)

    x = b = Conv2D(num_filters, 3, padding='same')(x)
    for i in range(num_res_blocks):
        b = res_block(b, num_filters, res_block_scaling)
    b = Conv2D(num_filters, 3, padding='same')(b)
    x = Add()([x, b])

    x = upsample(x, scale, num_filters)
    x = Conv2D(3, 3, padding='same')(x)

    x = Lambda(denormalize)(x)
    return Model(x_in, x, name="edsr")
```

WSDR

```
def wdsr(scale, num_filters, num_res_blocks, res_block_expansion, res_block_scaling, res_block):
    x_in = Input(shape=(None, None, 3))
    x = Lambda(normalize)(x_in)

    # main branch
    m = Conv2D(num_filters, 3, padding='same')(x)
    for i in range(num_res_blocks):
        m = res_block(m, num_filters, res_block_expansion, kernel_size=3, scaling=res_block_scaling)
    m = Conv2D(3 * scale ** 2, 3, padding='same', name=f'conv2d_main_scale_{scale}')(m)
    m = Lambda(pixel_shuffle(scale))(m)

    # skip branch
    s = Conv2D(3 * scale ** 2, 5, padding='same', name=f'conv2d_skip_scale_{scale}')(x)
    s = Lambda(pixel_shuffle(scale))(s)

    x = Add()([m, s])
    x = Lambda(denormalize)(x)

    return Model(x_in, x, name="wdsr")
```

WSDR-A Proposed Solution 1

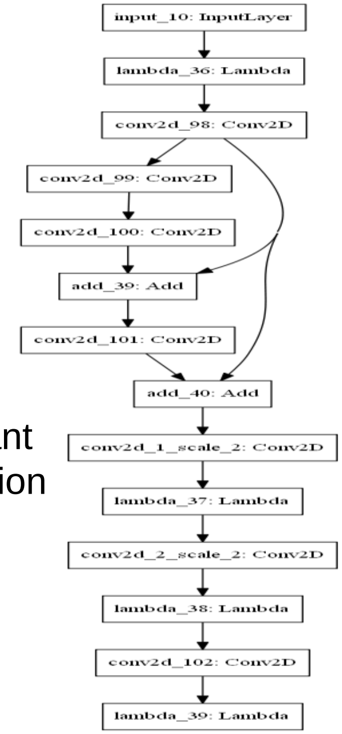
```
def res_block_a(x_in, num_filters, expansion, kernel_size, scaling):
    x = Conv2D(num_filters * expansion, kernel_size, padding='same', activation='relu')(x_in)
    x = Conv2D(num_filters, kernel_size, padding='same')(x)
    if scaling:
        x = Lambda(lambda t: t * scaling)(x)
    x = Add()([x_in, x])
    return x
```

WSDR-B Proposed Solution 2

```
def res_block_b(x_in, num_filters, expansion, kernel_size, scaling):
    linear = 0.8
    x = Conv2D(num_filters * expansion, 1, padding='same', activation='relu')(x_in)
    x = Conv2D(int(num_filters * linear), 1, padding='same')(x)
    x = Conv2D(num_filters, kernel_size, padding='same')(x)
    if scaling:
        x = Lambda(lambda t: t * scaling)(x)
    x = Add()([x_in, x])
    return x
```

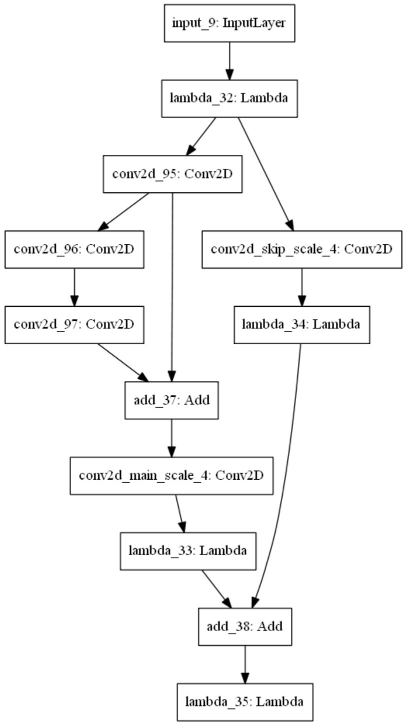
Model Architecture Samples

EDSR

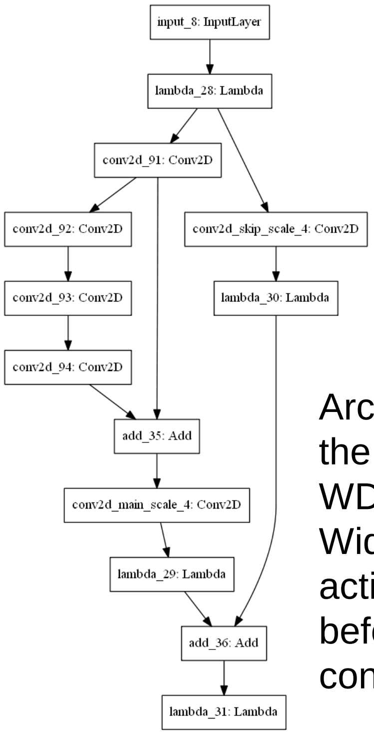


Redundant Convolution al stack.

WDSRa

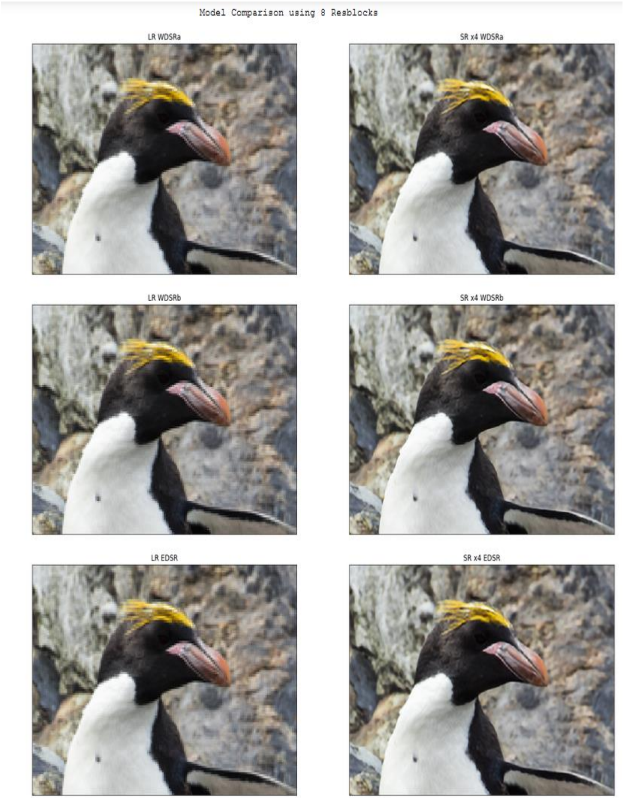


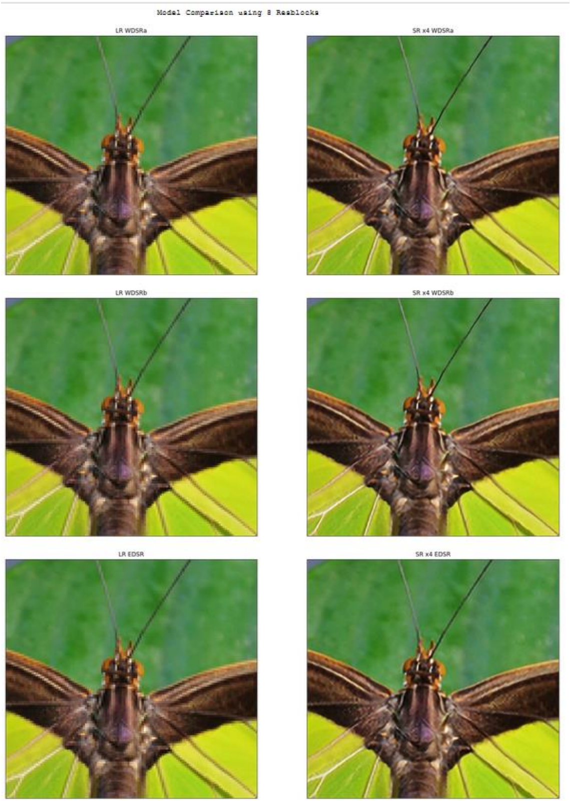
WDSRb

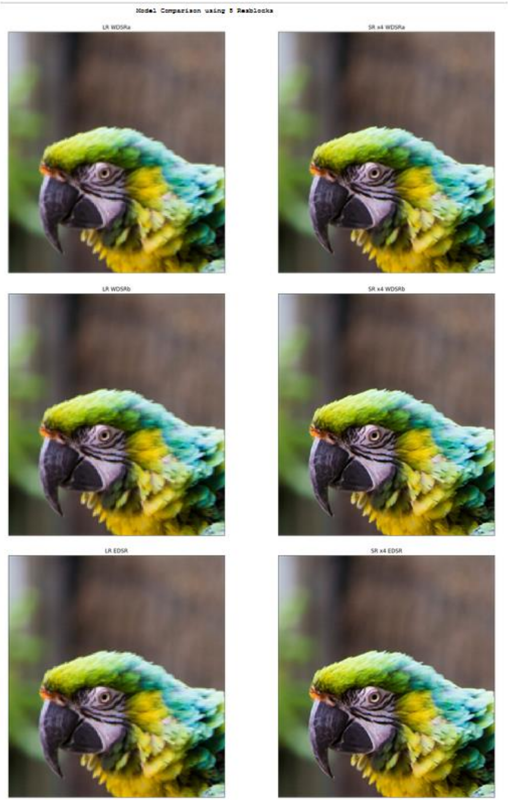


Architecture the same as WDSRa. Wider channel activation before skip connection.

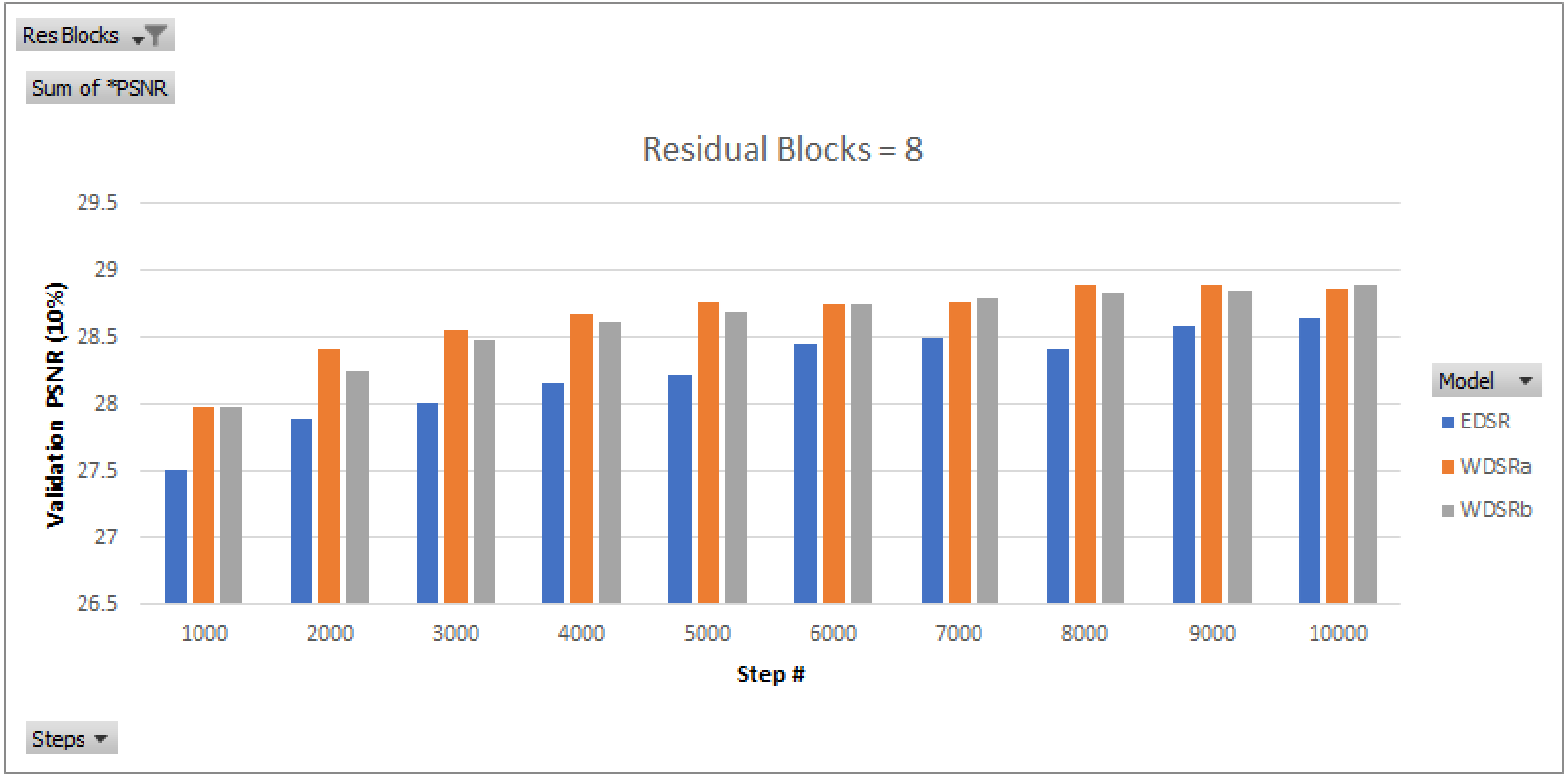
Example Visual Results







Comparison of Models



PSNR on 10 Validation Images During Training

Comparison of Models

Residual Blocks	1			3		
Networks	EDSR	WDSR-A	WDSR-B	EDSR	WDSR-A	WDSR-B
Parameters	409731.0	92304.0	36809.0	557443.0	240080.0	73595.0
DIV2k PSNR (validation)	27.5	27.8	27.7	27.8	28.1	28.0

Residual Blocks	5			8		
Networks	EDSR	WDSR-A	WDSR-B	EDSR	WDSR-A	WDSR-B
Parameters	705155.0	387856.0	110381.0	926723.0	609520.0	165560.0
DIV2k PSNR (validation)	27.9	28.2	28.1	27.9	28.2	28.2

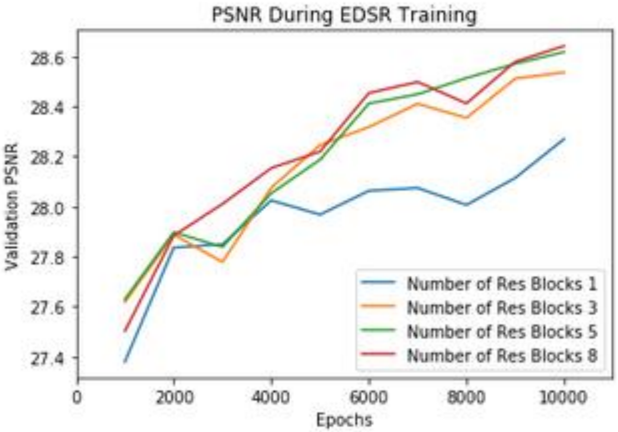
Comparison of Models

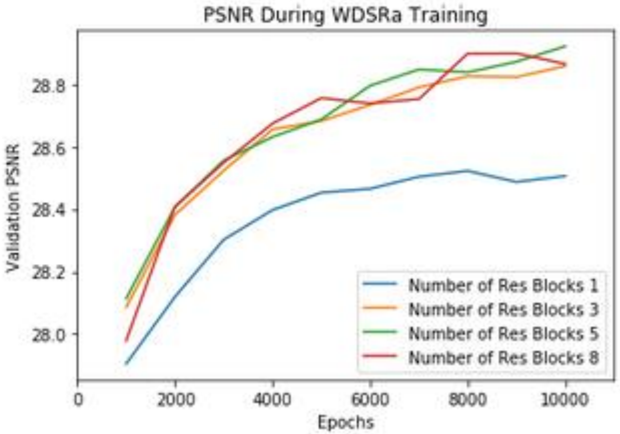
Residual Blocks	1			3		
Networks	EDSR	WDSR-A	WDSR-B	EDSR	WDSR-A	WDSR-B
% Parameter Increase	-	22.53%	8.98%	-	43.07%	13.20%
% Val PSNR Improvement	-	1.05%	0.82%	-	1.03%	0.72%

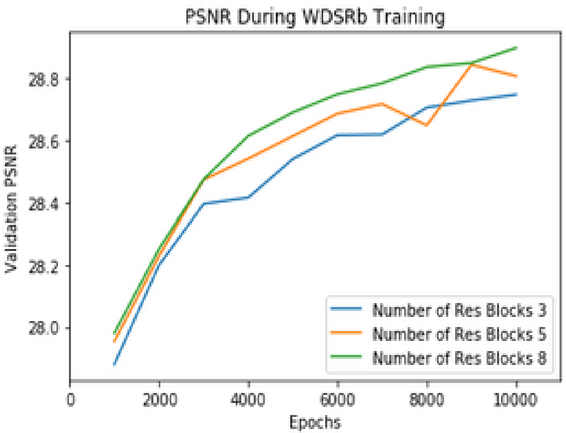
Residual Blocks	5			8		
Networks	EDSR	WDSR-A	WDSR-B	EDSR	WDSR-A	WDSR-B
% Parameter Increase	-	55.00%	15.65%	-	65.77%	17.87%
% Val PSNR Improvement	-	1.18%	0.86%	-	0.87%	0.98%

Key Takeaways

- Models likely trained far shorter (no reference in paper).
- WDSRa or WDSRb always outperform EDSR
- WDSRa and B always run at a fraction of the total number of params.
- WDSRb with 8 resblocks runs @ the highest Validation PSNR with 18% of the number of trainable parameters of EDSR.

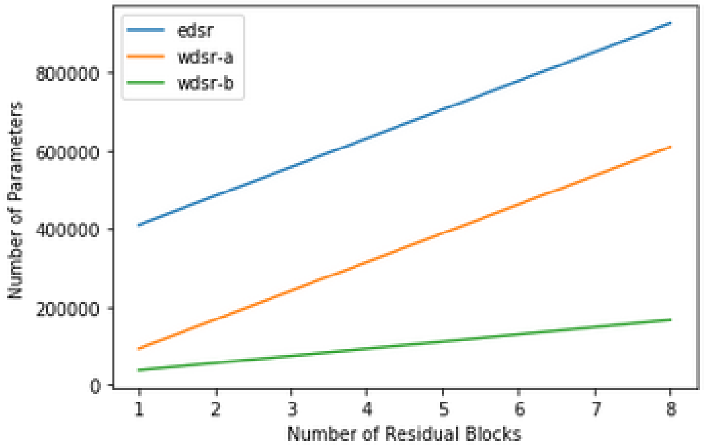


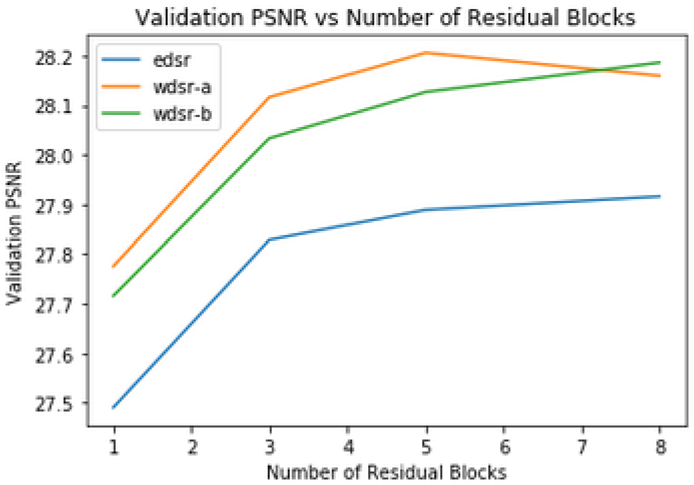




More Results

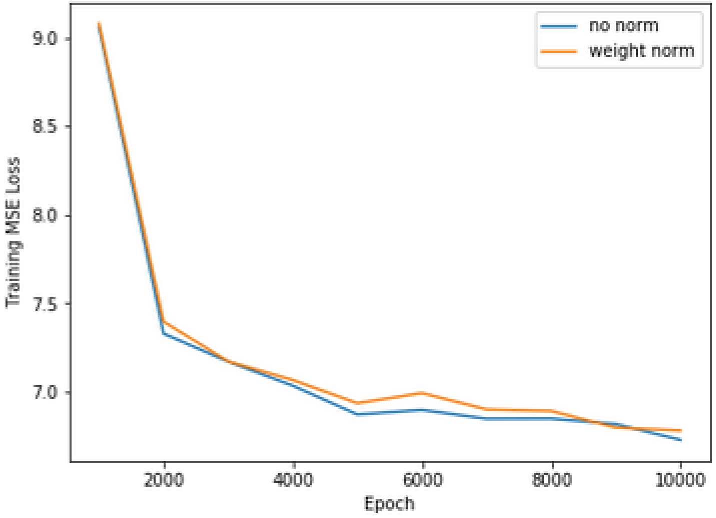
- PSNR mostly increases as a function of model depth (# of residual blocks)
- WDSRb parameter growth is much flatter than EDSR and WDSRa, even moreso than described in paper.

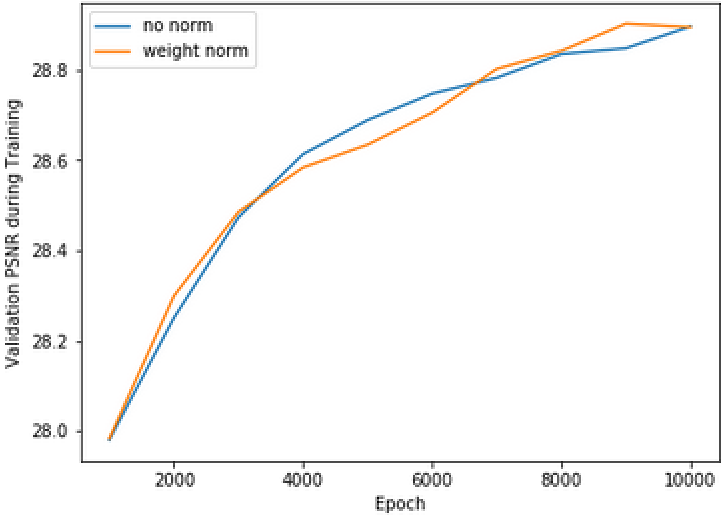




Normalization Effects

- No drastic difference between utilization of weight normalization layers against no normalization.
- Tests ran with Batchnorm proved to be unstable (PSNR ~16 throughout training)





References

1. Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Accurate image super-resolution using very deep convolutional networks”.
2. Sharon Peled and Yehezkel Yeshurun. “Superresolution in MRI: application to human white matter fiber tract visualization by diffusion tensor imaging”.
3. Chao Dong et al. “Learning a deep convolutional network for image super-resolution”.
4. Tong Tong et al. “Image Super-Resolution Using Dense Skip Connections”.
5. Y. Zhang et al. “Residual Dense Network for Image Super-Resolution”.
6. Ying Tai et al. “Memnet: A persistent memory network for image restoration”.
7. Chao Dong, Chen Change Loy, and Xiaoou Tang. “Accelerating the super-resolution convolutional neural network”.
8. Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. “Flattened convolutional neural networks for feedforward acceleration”.
9. Saining Xie et al. “Aggregated residual transformations for deep neural networks”.
10. Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”.
11. Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang. Learning a Deep Convolutional Network for Image Super-Resolution, in Proceedings of European Conference on Computer Vision (ECCV), 2014
12. W. Yang, X. Zhang, Y. Tian, W. Wang, J. Xue and Q. Liao, "Deep Learning for Single Image Super-Resolution: A Brief Review," in IEEE Transactions on Multimedia.
13. Yu, Jiahui et al. “Wide Activation for Efficient and Accurate Image Super-Resolution.” ArXiv abs/1808.08718 (2018)

Assignments

https://docs.google.com/spreadsheets/d/1CoJuoJgLHKp-m18c3Te0fyVIXqRbnKhyoAq3JYwUds8/edit#gid=1386834576