SRNEXT

CNNS STRIKE BACK ON SUPER-RESOLUTION

A PREPRINT

Dmitri Shevchenko
Dmitri.Shevchenko.au@gmail.com

2024-06-12

ABSTRACT

Recent developments in transformer alternatives have provided an opportunity to research their effectiveness in the broader vision domain. One such task, super-resolution, depends on rich feature representations to recover high quality images from low-quality inputs. Current state of the art models such as SwinIR heavily relies on attention based mechanisms to achieve this, which are infamous for their computational complexity and memory usage. While ConvNext has shown to outperform attention-based models primarily in classification tasks, we hope to show that the same method can be applied in the upscaling domain. We propose SRNext; a model that can outperform SwinIR in lightweight super-resolution tasks while using less memory and delivering higher throughput, while only being based on conventional convolutional neural networks.

1 Introduction

Image super resolution is an image-to-image process that requires adding width and height data to an input. Initially, this task was delegated to convolutional neural networks (CNNs) due to their image processing abilities, in particular SRCNN [9] and the ESR family [10][11]. More recently however, state of the art models utilize attention mechanisms (SwinIR [6], HAT [7], DCRT [8]) to achieve a global context [12][6] which previous CNN solutions lacked. This global context allows these models to leverage long-range dependencies within the input to better identify spatial information, improving performance and simultaneously efficiency [12].

However, there is an ever-growing interest in finding attention alternatives due to their complexity and computational demands [25] [26]. One such alternative, ConvNext, has been shown to outperform attention-based models for classification, image segmentation, and object detection. It achieves this by "modernizing" a ConvNet, following improvements discovered in the attention field; giving itself the name "A ConvNet for the 2020s". An opportunity presents itself here as ConvNext outperformed the previous state of the art, Swin Transformer, which is the same mechanism that drives current state of the art upscalers, such as those previously mentioned.

For a successful experiment into the impact of ConvNext for upscaling, we took SwinIR and replaced its Swin Transformer layers with ConvNext blocks. This allows us to directly compare the effects that replacing Swin Transformer layers with ConvNext has on training convergence and on final test set metrics. We hypothesized that we would have similar results seen in the classification space, namely 50% faster inference and 2-5% improved accuracy (depending on model size).
One immediate benefit is reduced complexity and operations, as the image and its feature representations don't need additional processing that are otherwise required for Swin Transformers to operate.

We ultimately aim to support further research into ConvNext for upscaling, and for other image-to-image tasks such as image generation or possibly to expand into the video domain.

2 Related Work

2.1 Super Resolution

The task of super resolution is commonly split into two primary subcategories - "Lightweight" and "Real World". Both methods try to achieve upscaling, however real world super resolution simultaneously tackles additional degradation including noise, blurring, and often compression artifacts such as jpeg. Alternatively, lightweight upscaling

focuses on creating more efficient models that are not expected to reverse degradation, resulting in smaller models for embedded devices or real-time applications.

In our experiment we were faced with selecting one of these methods; and decided to implement BSRGAN style degradation for real world super-resolution [16] in an attempt to investigate a "worst case scenario" where the most efficient model would produce the best results as it would learn more than the other and be more effective as a result.

2.2 SwinIR

To fully explain why SwinIR was so effective over its CNN predecessors, we must first acknowledge the adept flexibility of transformers and by extension, Attention.

Attention was introduced in Google Deep Mind's 2017 paper: "Attention is all you need" [12], where it was initially used for text-based translation tasks. Attention was developed to address parallelisation issues with RNNS, while also improving performance by identifying long-form relationships within input sequences.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where:

- Q represents the query matrix.
- K represents the key matrix.
- V represents the value matrix.
- d_k represents the dimensionality of the keys.

This equation computes attention scores, which identify long range relationships by examining the similarity between each pair of tokens in a sequence. These attention scores are then used to weight the importance of different tokens when generating a representation of the input.

Similarly, Transformers can be thought of as an extended form of attention [12] [13], allowing for abstract input sequences to be converted into a processed output sequence with a refined representation.

This then led to Vision Transformers, which as the name suggests, processes images with transformers by first converting them into patches so that they are compatible with the attention mechanism [14]. This was used in classification tasks for state of the art (SOTA) performance.

Finally, this leads to Swin Transformers, which improves the performance of Vision Transformers by applying a hierarchical shifted window across the input, significantly reducing memory requirements [15]. This also achieved SOTA in classification by surpassing Vision Transformers.

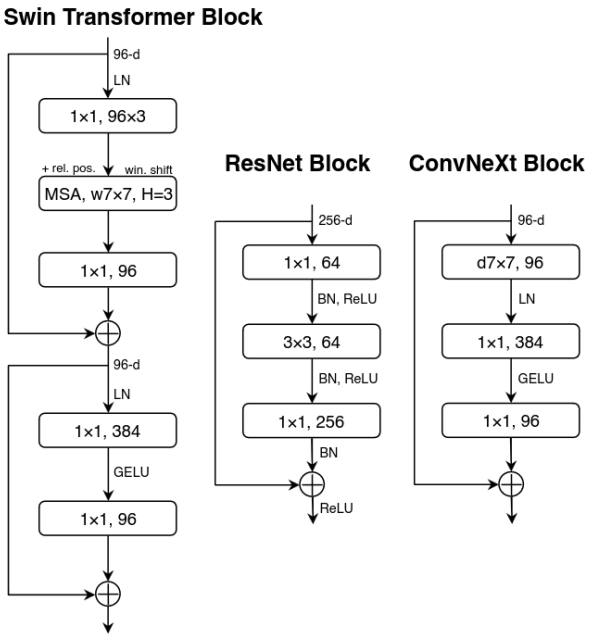


Figure 1: ConvNext structure compared to SwinTransformer and ResNet

For our research we focus on SwinIR; which is a super-resolution and image restoration technique that leverages the performance of Swin Transformer layers (which we will refer to as the Swin method in this paper) to achieve substantially better performance than its counterparts [6]. This ultimately spawned a family of swin-based super-resolution models, namely DRCT, which currently maintains state of the art in super-resolution as of writing [2].

2.3 ConvNext

ConvNext modernizes conventional convolutional neural networks, arguing that the performance of Swin Transformers is not intrinsic to the underlying attention mechanism. Instead, they provide insights on "updating" CNN compatible improvements such as kernel size, bottlenecks, and activation functions, to re-evaluate modern conventions. This resulted in attention-based like performance, without using attention [5]. Equally as important however, they show that by not using attention, throughput increases substantially, as much as 49% against an equivalent Swin-based model [5].

ConvNext's speed and efficiency was attributed to its simpler design as it fogos the complexities of attention. Unfortunately however, the paper does not go into great detail in the exact mechanism that allows ConvNext to become faster than Swin, more so at higher resolutions.
Nontheless, we expect ConvNext to perform fairly similar for super resolution as their experiments suggest superior performance in non-sparse vision tasks, which aligns with our task.

3 Method

3.1 Computation

Because of our limited compute, our experiment had to account for the likely event of incomplete results as most super-resolution solutions typically require non-consumer or hard to acquire hardware for extended periods of time to fully train. For this reason, we settled on training SwinIR and our solution in the same compute-limited environment in an attempt to produce fairer results.

3.2 Dataset

To train the models, we require both a high and low quality image for super-resolution. The low quality image is fed into the model to be processed and upsampled, where it is then compared to the high quality ground truth to calculate L1 loss:

$$L_1(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2}$$

where:

- $\hat{y}$ is the predicted value,
- y is the true value,
- n is the number of samples.

For our high quality source, we chose Flickr2k + DIV2k due to its academic popularity [6] [7] [8] [10] [11].
The method we used to apply degradation to the high resolution samples follows BSRGAN. This is relatively popular technique that applies multiple rounds of degradation (noise, blurring & artifacts) in different orders, simulating real-world deterioration [16] that is difficult to mimic otherwise. We first create the 256x256 high quality sample, then apply BSRGAN degredation to produce the 64x64 low quality pair, normalised to [0, 1].

3.3 Evaluation

For determining the performance of the model, we used the Set14, Urban100, BSD100, and MANGA109 testing datasets. Low quality samples are produced from this, which the model attempts to reverse. The model prediction and the ground truth are compared using the Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) index score for each sample, and then taking the average for the final result of each set.
We use these metrics as they allow us to determine the quality of the outputs, as well as allowing us to compare our results against other papers.

PSNR is used to determine per-pixel performance, and is defined as:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right) \tag{3}$$

Where:

- MAX is the maximum possible pixel value of the image.
- MSE is the Mean Squared Error between the original and reconstructed images.

Alternatively, SSIM is used for texture/edge quality, and is calculated as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4}$$

where:

- x and y are the compared images,
- μ_x and μ_y are the means of x and y,
- σ_x^2 and σ_y^2 are the variances of x and y,
- σ_{xy} is the covariance of x and y,
- c_1 and c_2 are constants to stabilize the division with weak denominator.

To identify the training progress of our models, we calculate the PSNR and SSIM results of Set14 at the end of each epoch.

We use these datasets due to their popularity, which should also allow us to compare our results with other papers. Please note that our PSNR implementation for the training logs was incomplete and differs from academic standards. Our mistake was calculating the luminance channel in place of the red, but not removing the excess green and blue channels for calculations. This was a mistake that could not be fixed without retraining, and given our time restraints, we had to leave it as is. It should still provide meaningful relative performance, but unfortunately these specific calculations are not compatible between other research papers. The final PSNR / SSIM were calculated with a third party library [17] to ensure consistent results.

3.4 Implementations

For our experiment, we implemented the upscaling training pipeline in pytorch version 2.3.0, which allows for "Channel Last" memory speed up for ConvNext. This also allows us to use the code of SwinIR and ConvNext provided in their respective github repositories with no further modifications. This should prevent any fatal implementation mistakes, while also enabling us to load provided pre-trained models for testing.

3.5 Architecture

To begin with the design of SRNext, we first take the original SwinIR implementation from github and used the "lightweight sr" configuration, namely setting the embedding dim to 60, mlp ratio to 2, window size to 8,

and upsampler to “pixel shuffle direct”.

This flavor of SwinIR was chosen because it had a pretrained model that we could evaluate for testing purposes, and because it was the smallest version that we had the resources to train.We then create a ConvNext based alternative from this, simultaneously trying to preserve at least most of these configurations to make it a suitable substitute.
While we implemented the same embedding dim, mlp ratio, and upsampler, we did make a architectural deviation that was shown to improve performance early in training. Primarily, we removed layer normalization within the ‘STL’-equivalent blocks, as it otherwise formed strange "blobs", likely similar to case that RealESR had [22], the reasons for which were unexplained.

Figure 2 shows the general structure that both SRNext and SwinIR had in our experiment.

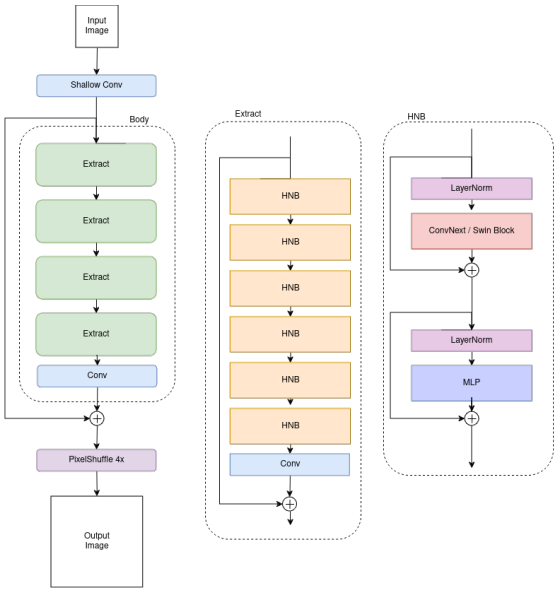


Figure 2: Generalised structure for SRNext & Lightweight SwinIR

Inadvertently, SRNext ended up with a slightly lower parameter count than SwinIR, which was necessary to validate our results
If SRNext had the parameter advantage and outperformed SwinIR, then one could argue that the extra parameters were necessary for it to gain those results. However, because SRNext has the parameter disadvantage, if it were to outperform SwinIR then it must fundamentally be more effective.

In the possible scenario that SRNext cannot outperform SwinIR with fewer parameters, then it would simply suggest poorer efficiency.

4 Experiment

4.1 Control

The control model was made to validate our work. It is very similar to SRNext and SwinIR, but contains no Swin or ConvNext blocks. If this control matched or outperformed SwinIR, it would be an indicator of a failed experiment as it would not align with results found in other well researched papers. This was only used during training as an early indicator rather than as an integral part of our research.

4.2 Training Parameters

We trained each model for 100k iterations (up to 10 hours on a RTX 3090) with a batch size of 32. We used the AdamW optimizer, with a learning rate of 1e-4, betas=(0.9, 0.999), eps=1e-08, and weight decay=0.01. This is mildly different from the original SwinIR paper which uses Adam. We chose AdamW because it was shown to “substantially improve Adam’s generalization performance” [23].

5 Results

5.1 Profiles

Here we are profiling SRNext against SwinIR to gain a better understanding about the computation required for each model to perform a single inference. These results are summarized from torchstat [18], which hooks and traces models layers to obtain the following values.

Parameters describes the number of trainable weights each model has. More parameters allow for better generalization, at the cost of slower inference speeds [19]. Generally a model with fewer parameters that performs the same is considered to be more efficient.

Memory is the amount of gpu memory necessary to perform an inference. Less memory allows smaller systems to run the model.

GMAdd is the number of gigga (billion) multiplications and additions needed to perform an inference. Fewer GMAdds typically results in faster inferences.

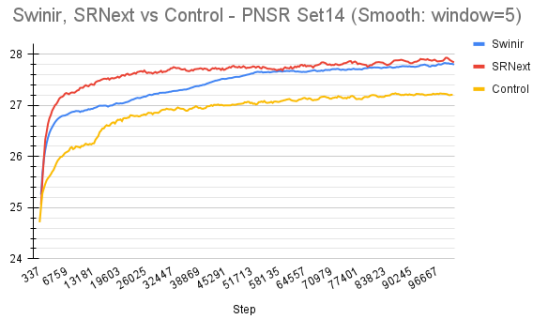
Flops is the number of floating point operations needed to perform an inference. Fewer GMAdds typically results in faster inferences.
These values are in respect to an 256x256 input image. Lower is better.

	Parameters	Memory	GMAdd	Flops
SwinIR	897,228	3540MB	437	218
SRNext	840,840	1434MB	94	47

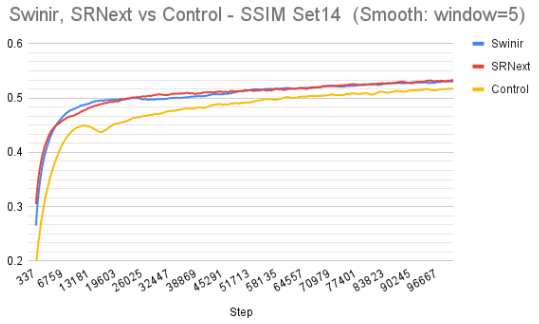
Table 1: Model characteristics

5.2 Training

The training PSNR and SSIM values on Set14 can be seen in figure 3.



(a) PSNR



(b) SSIM

Figure 3: PSNR & SSIM Set14 results during training

5.3 Final Metrics

These are the PSNR/SSIM values, calculated by an external library [17]. Higher is better.

Set (PSNR)	SwinIR	SRNext
Set14	24.7218	25.9241
BSD100	24.3544	24.8355
URBAN100	23.4033	24.6494
MANGA109	23.2317	23.9694

Table 2: PSNR Results

Set (SSIM)	SwinIR	SRNext
Set14	0.5294	0.6116
BSD100	0.4314	0.5285
URBAN100	0.4566	0.5863
MANGA109	0.5153	0.6404

Table 3: SSIM Results

5.4 Thoughput

These values are the average number of images each model can produce per second on a RTX 3090 (repeat=5), higher is better.

Image Size	64x64	128x128	256x256
SwinIR	119k	39k	12k
SRNext	185k	96k	20k

Table 4: Avg. Samples Per Sec

6 Discussion

6.1 Expectations

To fully identify if SRNext is a suitable alternative to SwinIR, we must first address a fundamental trade off that almost every model encounters, that of accuracy versus performance. Fewer parameters typically allows for faster inferencing, but places a limit on the maximum throughput of a model. On the other hand, adding too many parameters tends to lead to diminishing returns, while simultaneously slowing training and inference speed [18] [19].

6.2 Comparisons

However, despite using 7% fewer parameters, not only does our solution outperform an equivalent SwinIR model in each provided test dataset, but can do so with a throughput 1.6x faster at an input resolution of 128x128 while using 40% the memory. These results are futher supported by theory as they align extremely close to the results gained in the original ConvNext paper [5].

An additional benefit that we can identify in the training PSNR curve is that SRNext converges faster and at a higher value than SwinIR. The SwinIR paper discusses how their model was able to converge “faster and better than RCAN [a CNN based solution] which is contradictory to previous observations that Transformer-based models often suffer from slow model convergence”.

As an aside, the convergence of transformers in comparison to conventional methods is a seemingly undocumented yet somehow widely agreed upon phenomena. As an example, DETR (image detection) is infamous for its arduous training in comparison to FasterRCNN, requiring additional techniques for better performance [1][3]. Regardless, our results show what is normally to be “expected” when compared against an attention based solution.

We also observe far fewer operations occurring as indicated by the lower MAdd and Flops, which makes

Figure 4: Set14 Samples, SRNext top, SwinIR bottom

SRNext a more viable option for mobile devices as there are fewer operations needed to inference the model.

It may be possible to utilize this model for real-time applications as it is more suited to run as a shader. This technique is somewhat experimental, but was recently popularized by the likes of RealESR/Video2x[22], FSR [20], and DLSS[21] for use in video games and movies.

6.3 Limitations

It is difficult to judge if the compute available was adequate to fully evaluate SwinIR as unfortunately the original paper does not directly state number of epochs/iterations, batch size, or the hardware that was used to train the lightweight version.

We are also unable to test if SRNext scales in the same manner as SwinIR with more parameters due to compute and time constraints. It is possible that at higher parameter counts SwinIR may train more optimally, as it has been similarly shown that RNNs can outperform transformers, but only on smaller scales [4].

For training, we utilized the lightweight flavor of SwinIR as it was the easist to train, however the paper argues that 8x8 window sizes reduces real-world performance which they attribute to jpeg compression artifacts also being 8x8 [6].

The OpenCV interpolation that we used to train the models does not apply anti-aliasing, which can impact the quality of the downsampled input images.

6.4 Future work

As previously addressed, our research lacks the compute to properly identify if SRNext can maintain its lead with larger parameter counts, but we hope to show that despite not utilizing attention, SRNext is a suitable contender in the super-resolution space. We hope this ignites the re-evaluation of CNN based solutions, either by building upon our work or developing entirely new solutions. It may be possible to apply the same improvements that DCRT did over SwinIR to feasibly achieve state of the art performance. Unfortunately this is all speculation until the compute requirements are addressed.

7 Conclusion

By placing SwinIR and SRNext in the same training environment and showing that SRNext outperforms SwinIR substantially, we hope to shed light on the potential of attention alternatives to provide improvements in lightweight, and possibly real-world super-resolution tasks.
Despite our results indicating state of the art performance, we are not confident without further evaluation due to minor "hiccups" as a result of severe restraints that prevented our research to reach its maximum potential. Regardless, we hope it shows promise with strong theoretical and practical grounds to pursue further research.

8 Ethical

We consumed roughly 50 hours of gpu processing, releasing around 5kg of CO2. We also acknowledge that the dataset may contain copyrighted material, however for our research purposes, this public data was a necessary step to create a overall beneficial impact.

References

[1] Yu, P., Chen, Y., Jin, Y. & Liu, Z. Improving Vision Transformers for Incremental Learning. (2021)

[2] Hsu, C., Lee, C. & Chou, Y. DRCT: Saving Image Super-resolution away from Information Bottleneck. (2024)

[3] Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L. & Wang, J. Conditional DETR for Fast Training Convergence. (2021)

[4] Izsak, P., Guskin, S. & Wasserblat, M. Training Compact Models for Low Resource Entity Tagging using Pre-trained Language Models. (2019)

[5] Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T. & Xie, S. A ConvNet for the 2020s. (2022)

[6] Liang, J., Cao, J., Sun, G., Zhang, K., Gool, L. & Timofte, R. SwinIR: Image Restoration Using Swin Transformer. (2021)

[7] Chen, X., Wang, X., Zhou, J., Qiao, Y. & Dong, C. Activating More Pixels in Image Super-Resolution Transformer. (2022)

[8] Hsu, C., Lee, C. & Chou, Y. DRCT: Saving Image Super-resolution away from Information Bottleneck. (2024)

[9] Dong, C., Loy, C., He, K. & Tang, X. Image Super-Resolution Using Deep Convolutional Networks. (2014)

[10] Wang, X., Xie, L., Dong, C. & Shan, Y. Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data. (2021)

[11] Kim, J., Lee, J. & Lee, K. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. (2015)

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. Attention Is All You Need. (2017)

[13] Turner, R. An Introduction to Transformers. (2023)

[14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. & Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. (2020)

[15] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. & Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. (2021)

[16] Zhang, K., Liang, J., Gool, L. & Timofte, R. Designing a Practical Degradation Model for Deep Blind Image Super-Resolution. (2021)

[17] Chen, C. & Mo, J. IQA-PyTorch: PyTorch Toolbox for Image Quality Assessment. ([Online]. Available: https://github.com/chaofengc/IQA-PyTorch,2022)

[18] 'Swall0w' torchstat ([Online]. Available: https://github.com/Swall0w/torchstat,2018)

[19] Khan, Z., Akella, K., Namboodiri, V. & Jawahar, C. More Parameters? No Thanks!. (2021)

[20] GPUOpen by AMD. (2024, March 11). AMD FidelityFXTM Super Resolution 1 (FSR 1) - AMD GPUOpen. AMD GPUOpen. https://gpuopen.com/fidelityfx-superresolution/

[21] DLSS 2.0 -IMAGE RECONSTRUCTION FOR REAL-TIME RENDERING WITH DEEP LEARNING. (n.d.). http://behindthepixels.io/assets/files/DLSS2.0.pdf

[22] Intaniyom, T., Thananporn, W. & Woraratpanya, K. Enhancement of Anime Imaging Enlargement using Modified Super-Resolution CNN. *2021 13th International Conference On Information Technology And Electrical Engineering (ICITEE)*. pp. 226-231 (2021)

[23] Loshchilov, I. & Hutter, F. Decoupled Weight Decay Regularization. (2017)

[24] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena and Chinmay Hegde. On The Computational Complexity of Self-Attention, 2022; arXiv:2209.04881.

[25] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2023; arXiv:2312.00752.

[26] Kaplan, J., McCandlish, S., Henighan, T., Brown, T., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. & Amodei, D. Scaling Laws for Neural Language Models. (2020)

9 Appendix

In this appendix, we further evaluate SRNext's performance directly against current state of the art models in the light-weight super-resolution space.

9.1 Preface

We prefix the following results by stating that we are writing this paper close to its deadline. Unfortunately, while we wish for more concrete results, there is simply no more time left to train any more models or to get higher quality datasets/testsets. It is for this reason that for almost all of the limitations that follow, it was mostly due to time constraints.

9.2 Limitations

Super resolution models are primarily trained with bicubic interpolation. This bicubic interpolation is almost exclusively done in MatLab, likely due to its academic popularity and performance, as well as for backwards compatibility.

However, we were unable to gain access to MatLab. This is a problem as the bicubic interpolation implementations that PyTorch and OpenCV freely provides are substantially different to MatLab. The closest, most available, and popular implementation we could find was from BasicSR.
https://github.com/XPixelGroup/BasicSR/issues/15

However, despite all this, BasicSR/Matlab's bicubic interpolation still produces slightly different outputs. Even though these differences seem practically insignificant, they can substantially affect the performance of upscalers. This will become evident in the Metrics section.

9.3 Training

We take our SRNext model that was trained alongside our version of SwinIR, and continue training for 100k iterations. Our initial learning rate was set to 2e-4, halving every 50 epochs to a minimum of 1e-5. Rather than using BSRGAN degradation, we trained for lightweight upscaling with BasicSR interpolation to produce the low quality (64x64) and high quality (256x256) pairs. This is what will allow us to compare against official state of the art models.

9.4 Set14 Metrics

As discussed in the limitations, the interpolation function used to train our SRNext model is different from SwinIR. In order to evaluate exactly what kind of effect this had and if it invalidates our results, we can analyze SRNext's performance from each one. The Set14 MatLab images were taken from kaggle.
https://www.kaggle.com/datasets/ll01dm/set-5-14-super-resolution-dataset

9.5 Values

SRNext	Set14 (BasicSR)	Set14 (Matlab)
PSNR	28.0285	28.0505
SSIM	0.7678	0.7574

Table 5: SRNext Set14 Results

As you can see in table 5, the difference in results from using BasicSR vs MatLab interpolation is extremely minimal, which is to be expected. It is even surprising to see that the Matlab interpolation testset resulted in better PSNR performance, despite SRNext never being trained on it. However, when performing the same analysis with SwinIR, we get the following results in table 6:

SRNext	Set14 (BasicSR)	Set14 (Matlab)
PSNR	28.2085	28.3028
SSIM	0.7685	0.7653

Table 6: SwinIR Set14 Results

The PSNR discrepancy between SRNext and SwinIR is noticeably larger, 0.022dB vs 0.0943dB, or about 4x greater. Somewhat surprising is the SSIM marginally going down!

The most reasonable, and likely explanation, is that because of SwinIR "greater performance", it might be more sensitive to changes in the interpolation function. On the other hand, SSIM, which tries to measure texture/quality, is less sensitive to these differences and the worser results are a fluke.

Unfortunately, we could not find other verifiably MatLab interpolated test sets, so a deeper investigation cannot be made.

Final results

The following are done with BasicSR's bicubic interpolation on the samples from: https://www.kaggle.com/datasets/jesucristo/super-resolution-benchmarks

Each of the models were taken from their respective repositories, using the official checkpoints that were similarly provided.

Bolded results are the best of the set

PSNR	Set14	BSD100	URBAN100	MANGA109
IMDN	27.8040	24.0524	27.2880	23.4331
CARN	28.0326	24.0659	27.3436	23.4023
SwinIR	**28.2085**	24.0346	**27.7790**	23.4692
SRNext	28.0364	**24.1766**	27.1437	**23.7909**

Table 7: Final PSNR Results

PSNR	Set14	BSD100	URBAN100	MANGA109
IMDN	0.7508	0.7002	0.8201	0.8176
CARN	0.7490	0.6730	0.7945	0.7885
SwinIR	**0.7685**	0.7080	**0.8397**	0.8297
SRNext	0.7677	**0.7103**	0.8263	**0.8338**

Table 8: Final SSIM Results

As you can see, SwinIR and SRNext perform extremely similarly, which is anticipated from what the research we performed would suggest, given the expected higher efficiency but lower parameter count.

However, there is a very important problem that sadly invalidates these results - All of them do not align with their respective papers' results, roughly by about 0.2-0.5dB for PSNR and 0.05 for SSIM, even though the relative performance and order seems about correct.

This would more than likely indicate that the PSNR/SSIM calculations are incorrect, but we are using a popular 3rd party library [17] to calculate these values. Frustratingly, there is no clear answer to explain all these discrepancies, especially without access to MatLab and even more so against our time constraints.

Final Remarks

We still hope to show the potential that SRNext provides once these unknowns are solved, as we are still confident that it can provide potential insight for future papers to build upon.

Contribution

While this project was initially meant as a collaborative effort, this paper and it's supporting materials were created by it's sole author, Dmitri Shevchenko.