Math Foundations: Exercise 6

Orion Junkins

May 27, 2024

1 Introduction

This lab presents an approach to image super-resolution using Convolutional Neural Networks. Two approaches are implemented: a Basic approach using 10 Convolutional blocks and a more advanced Residual model that introduces a residual connection. Training curves and final validation accuracy are compared across a range of learning rates for both approaches.

2 Implementation

The attached implementation follows the architecture outlined in the assignment specification with two key additions.

First, an additional *patches_per_image* argument is made available in the *SRDataset* Class constructor. This argument controls the number of patches sampled from each image. For training data, this value is set to 1, yielding the exact performance of the approach outlined in the assignment. This value is set higher for evaluation data, causing multiple patches to be sampled from each image. This enables the usage of a batch size larger than 9, allowing a more robust estimation of validation accuracy with a single epoch.

Second, all CNN layer weights are initialized with values sampled from a normal distribution with a standard deviation of 0.05. This fairly high random initialization helps improve the accuracy of the res net model. Without it, the model was often stuck in a local minima relatively early; it would simply output the bilinear upsampling, achieve reasonable loss, and never improve further. This random initialization ensures the initial model is very poor, forcing it to learn a more complex mapping. This problem is still present but mitigated (see discussion).

All code is contained within the `code` directory. The naive and Residual models are implemented in `code/sr_model_basic.py` and `code/sr_model_res.py` respectively.

To train, run `code/train.py` as follows:

```
$ python code/train.py [--epochs E] [--lr LR] [--model M]
```

Optionally, provide epochs (the default is 10), learning rate (the default is 0.0001), and model type ("basic" or "res"). This will train a model, save the final weights, and save a sample prediction from the evaluation set.

Similarly, to run inference and evaluation, run `inference.py` as follows:

```
$ python code/inference.py [--interp I]
```

Optionally, provide an interpolation scheme (either "Bilinear," "Bicubic," or "Nearest," default is "Bilinear"). This will load the best-performing model for both Basic and Residual model classes, load the evaluation images, downsample the evaluation images using the provided scheme, and evaluate the models. L1, PSNR, and SSIM are printed, and an image containing sample outputs is saved. Metrics are also computed and printed for naive Bilinear and Bicubic baselines.

3 Training

The Basic and Residual models were trained for 1000 epochs with 1e-02, 1e-03, 1e-04, 1e-05, and 1e-06 learning rates. Figures 1 and 3 below show the evolution of L1 Loss, PSNR, and SSIM metrics over the 1000 epochs for all learning rates. For both the Basic and Residual models, accuracy diverged substantially with a learning rate of 1e-02. So, figures 2 and 4 exclude 1e-02 to allow for a better comparison of the other three values. Final metrics appear in the "Value" column below each plot.

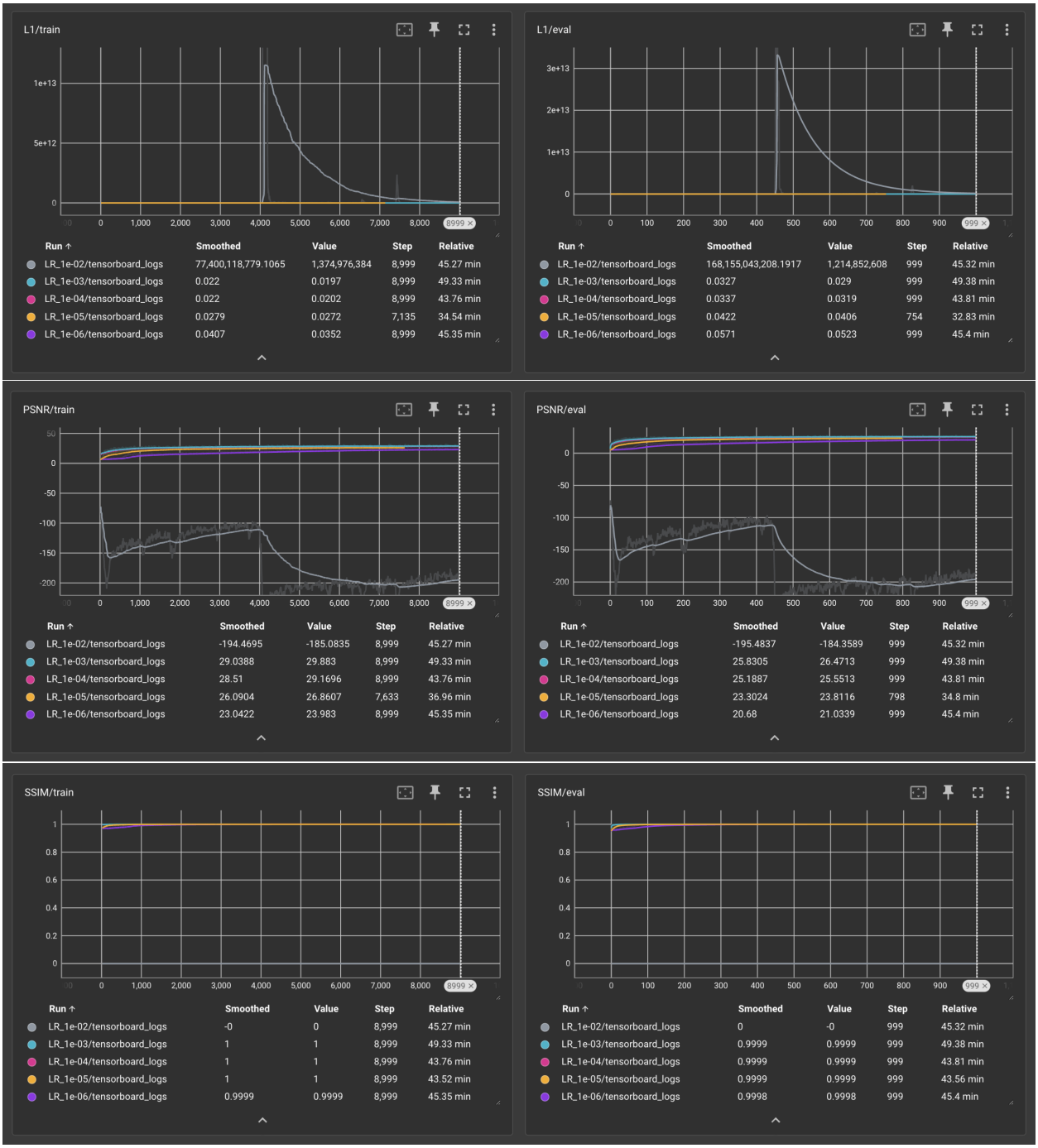

Figure 1: Basic model error metrics for all learning rates

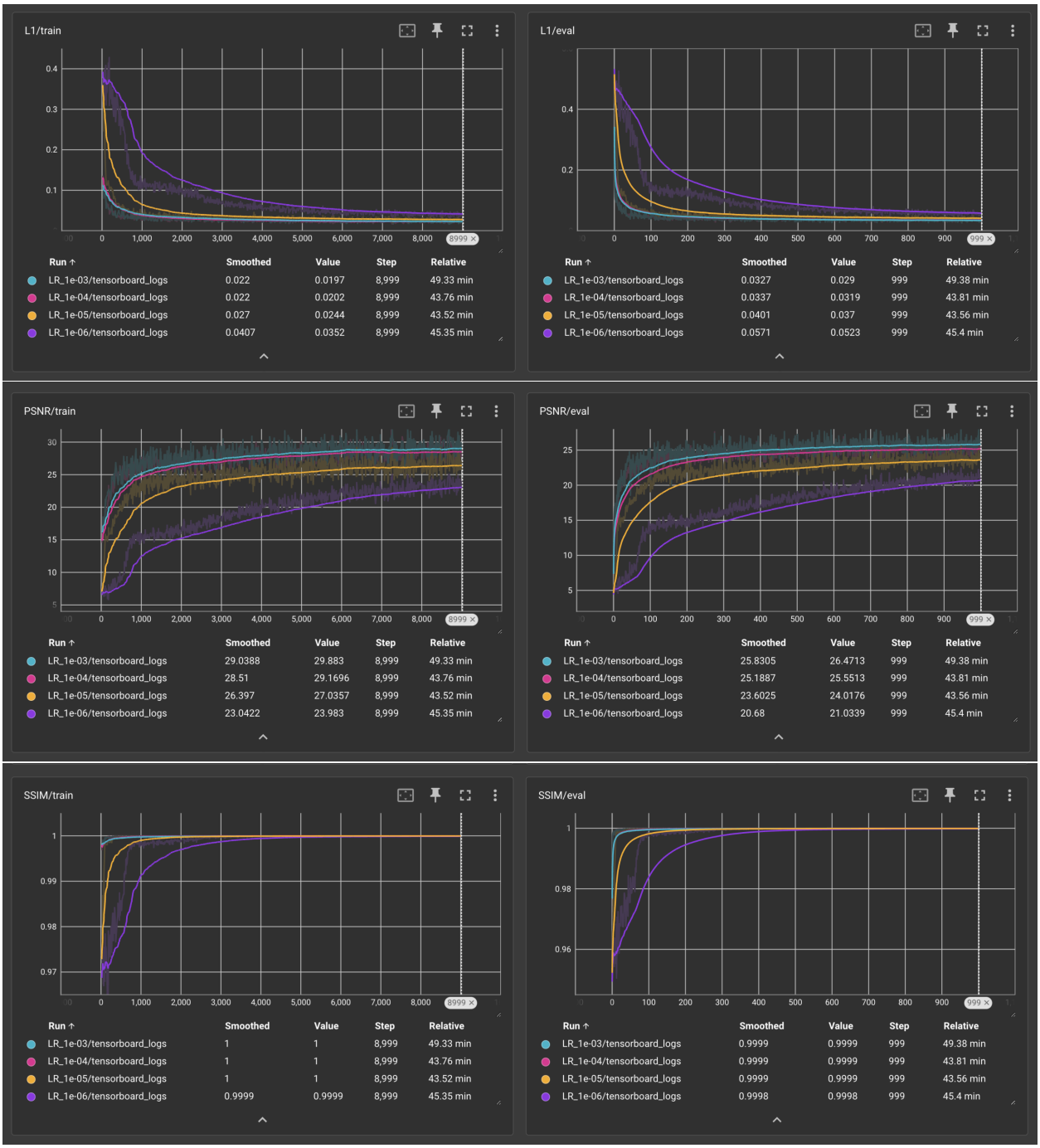

Figure 2: Basic model error metrics for LR of 1e-03, 1e-05, and 1e-06

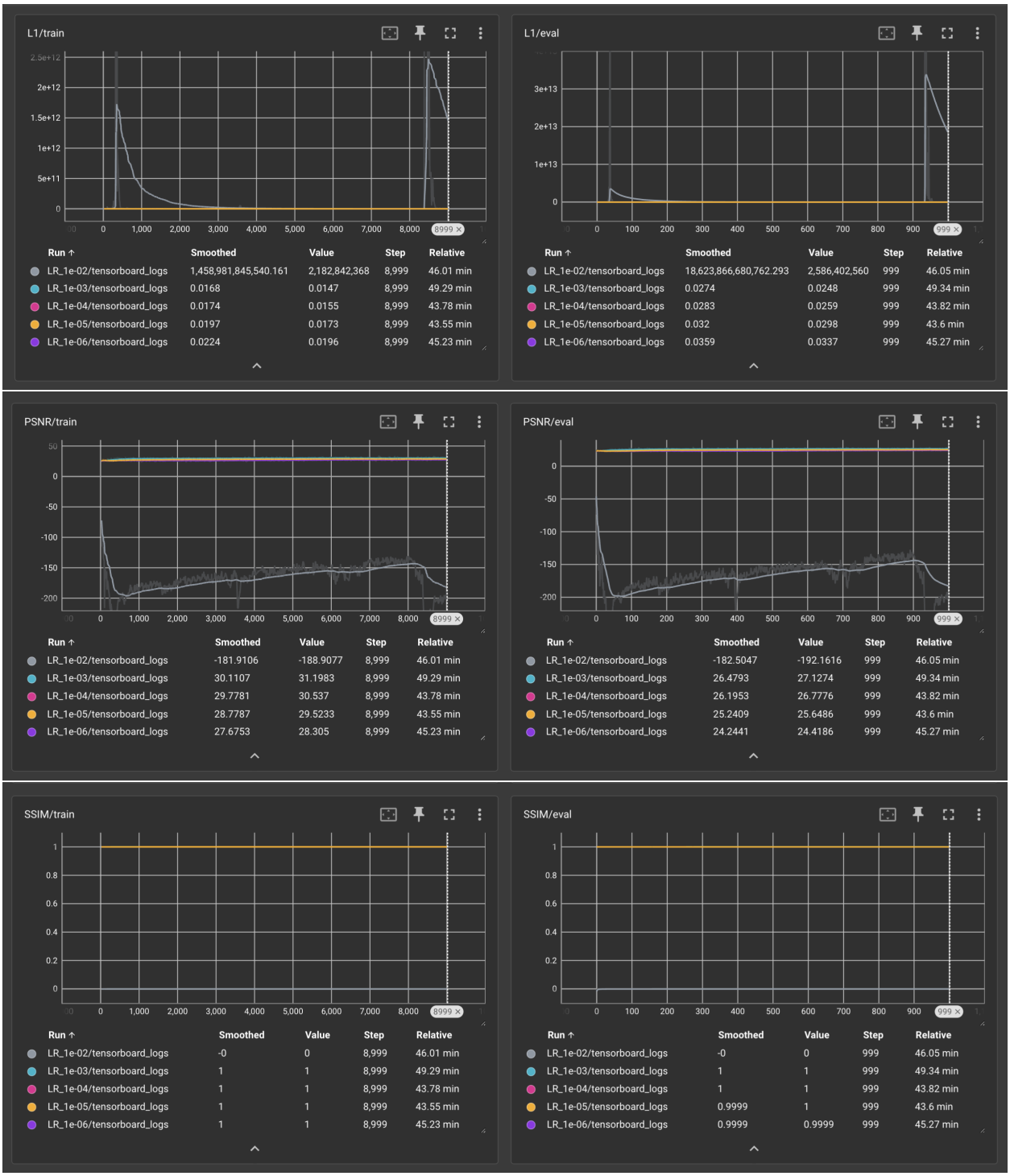

Figure 3: Residual model error metrics for all learning rates

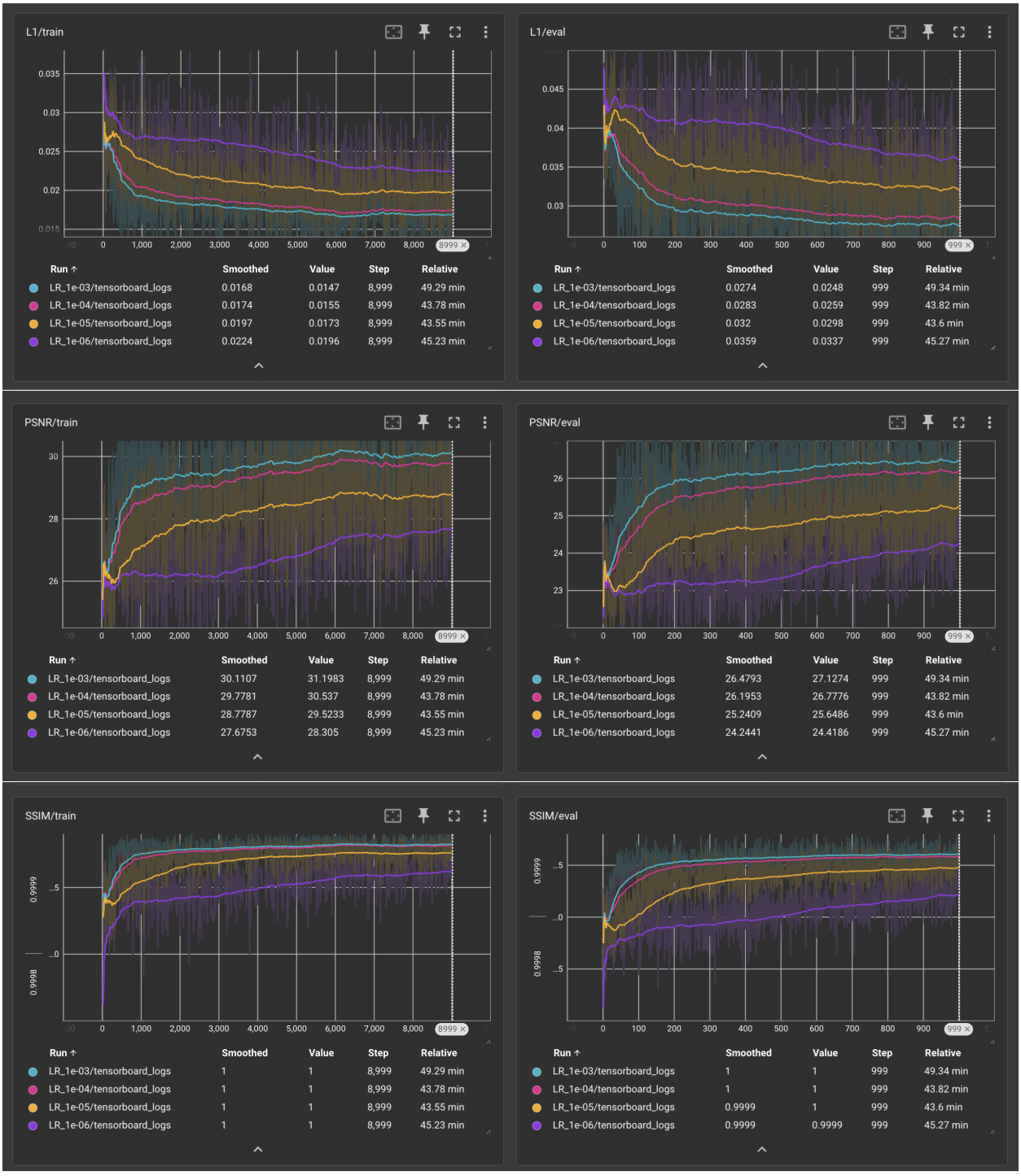

Figure 4: Residual model error metrics for LR of 1e-03, 1e-04, 1e-05, and 1e-06

4 Evaluating Effect of Downsampling

Evaluation scores for all models appear beneath their training curves. For further evaluation, this section takes a closer look at the fully trained 1e-03 learning rate Basic and Residual models as they performed best across all learning rates. Specifically, it explores how the models perform when the input data is downsampled with different schemes.

To test performance, the evaluation dataset is loaded with eight patches sampled from each image (as described in the Implementation section above) for a more robust error estimate. The evaluation dataset is downsampled using Bilinear, Bicubic and Nearest Neighbor downsampling. Section 4.1 presents quantitative results for both models across the three downsampling schemes. As a baseline, Bilinear and Bicubic upsampling are also included. Section 4.2 presents example outputs across all three downsampling schemes.

4.1 Quantitative Results

	L1	PSNR	SSIM
Residual Model	0.0285	26.9664	1.0000
Basic Model	0.0332	26.2978	0.9999
Bilinear	0.0434	23.4675	0.9999
Bicubic	0.0389	24.3338	0.9999

Table 1: Error metrics for Basic and Residual models compared against baselines when evaluation data is downsampled using Bilinear interpolation

	L1	PSNR	SSIM
Residual Model	0.0337	25.6939	0.9999
Basic Model	0.0372	25.3472	0.9999
Bilinear	0.0396	24.1639	0.9999
Bicubic	0.0360	25.0307	0.9999

Table 2: Error metrics for Basic and Residual models compared against baselines when evaluation data is downsampled using Bicubic interpolation

	L1	PSNR	SSIM
Residual Model	0.0860	17.1475	0.9994
Basic Model	0.0850	17.4134	0.9994
Bilinear	0.0447	22.5244	0.9998
Bicubic	0.0467	22.1106	0.9998

Table 3: Error metrics for Basic and Residual models compared against baselines when evaluation data is downsampled using Nearest Neighbor Interpolation

4.2 Sample Outputs

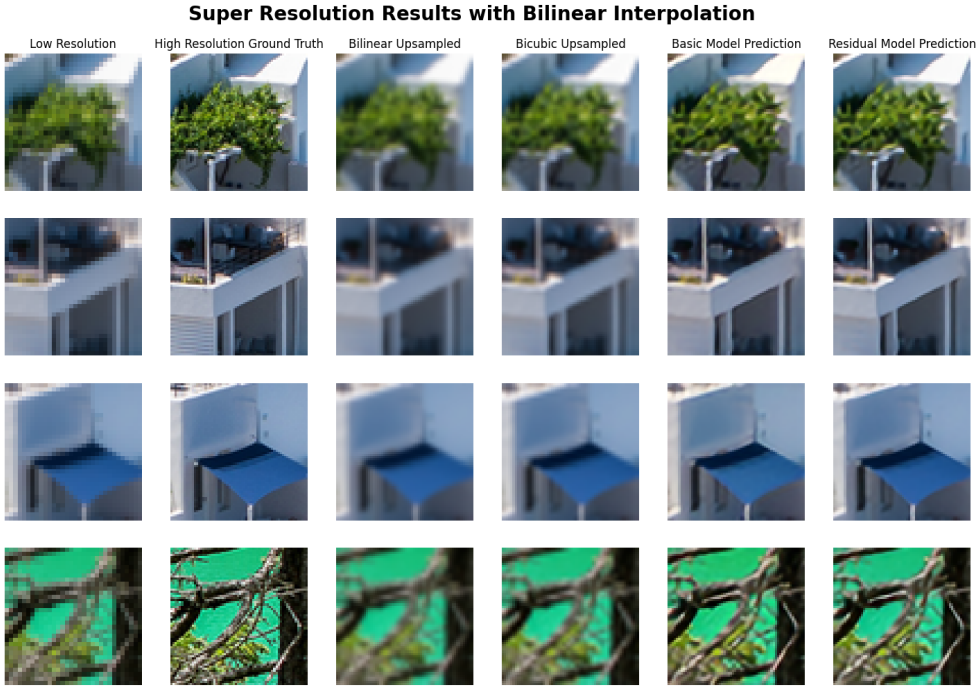

Figure 5: Results on evaluation patches downsampled with the Bilinear scheme

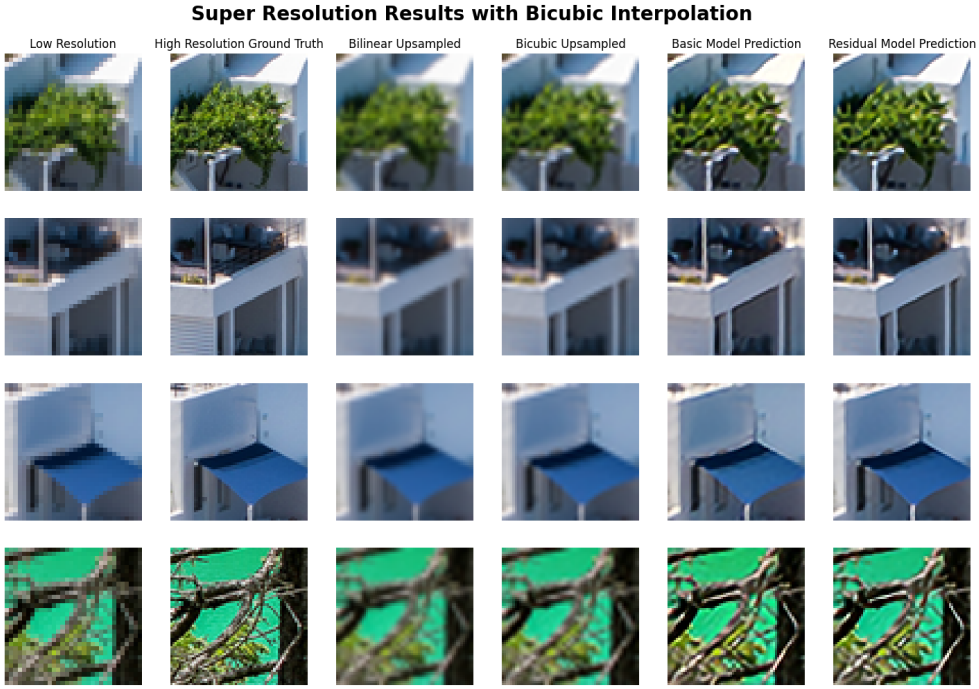

Figure 6: Results on evaluation patches downsampled with the Bicubic scheme

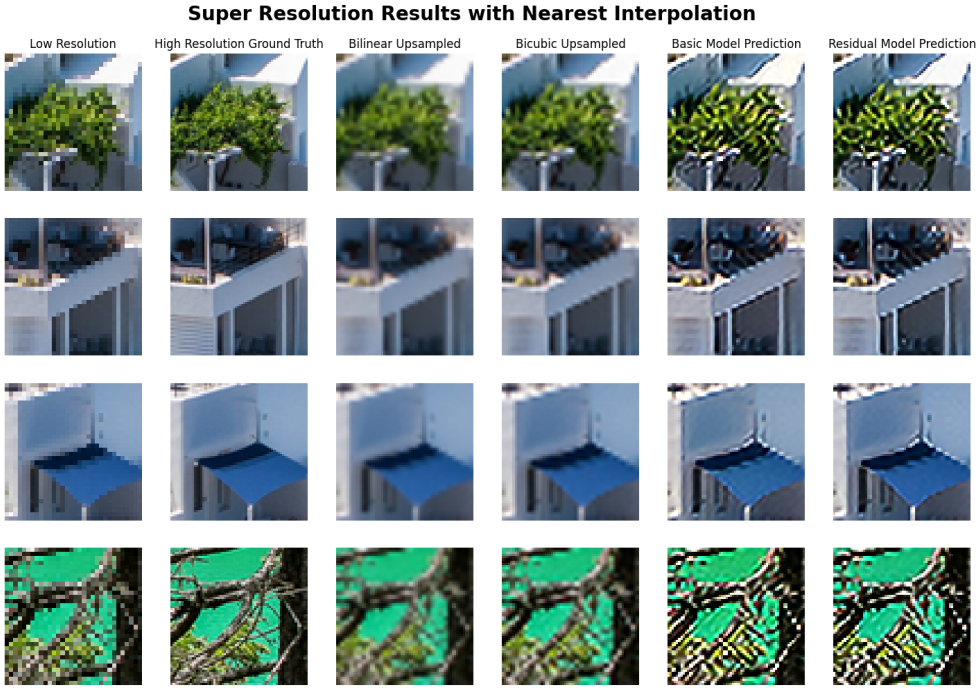

Figure 7: Results on evaluation patches downsampled with the Nearest Neighbor scheme

5 Discussion

5.1 Effect of Learning Rate

Overall, a learning rate of 1e-03 consistently performed as well or better than other values on both train and evaluation sets across all metrics. The learning rate directly affects the step size taken in the loss landscape. Learning rates that are too low will be slower to converge and potentially become stuck in local minima as the step size becomes too small. Increasing the learning rate can avoid these issues, causing the model to take larger steps, converging faster and "stepping over" or "stepping out of" local minima. However, increasing the learning rate too far will cause divergence, as the model will consistently "overshoot" optimal points, leading to remote, poor-performing areas of the loss landscape.

Thus, finding a sweet spot is key. For this dataset, it appears to be approximately 1e-03. Exploring nearby values could further optimize this.

One final interesting conclusion in the loss landscapes is an apparent local minimum for the Residual model that must be escaped. Performance is relatively stagnant from approximately 5 to 35 epochs, even worsening performance. Beyond 40 epochs, the model escapes this minimum and continues improving. Presumably, this minimum corresponds to directly passing the Bilinearly upsampled version of the image without modification. This naive approach would yield reasonable loss scores, and improving upon this accuracy is non-trivial. Adding more aggressive random initialization (as discussed in the Implementation section above) helps to mitigate this issue by forcing the model to be worse initially. However, it does not avoid it completely. Ultimately, this does not affect the final outcome but is an interesting observation regarding what the model is doing under the hood.

5.2 Comparison Between Models

Both models ultimately generate high-quality outputs. For Bilinear and Bicubic downsampled data, the residual model clearly outperforms the basic model, achieving better scores across all metrics. Furthermore, the Basic model is substantially slower to train, taking many more epochs to reach reasonable performance. This result is perfectly logical as the residual connection leverages Bilinear upsampling to get a head start. Rather than learning the entire upsampling process from scratch, the Residual model has a much easier task; it must learn only the aspects of upsampling that naive Bilinear upsampling fails to capture.

Presumably, if given enough capacity and training data, the basic model could eventually converge to competitive performance. But, it is much simpler and more efficient to take advantage of this easy head start.

5.3 Role of Downsampling Method

These results are highly dependent on the type of downsampling used. When Bilinear downsampling is used, both models substantially outperform the baselines. When Bicubic downsampling is used, both models outperform the baselines but with less margin. When Nearest Neighbor downsampling is used, neither model beats the baseline, and the Basic model outperforms the Residual model slightly.

This degradation is reasonable given that the network is trained solely using Bilinearly downsampled data. In other words, the network has effectively learned to invert Bilinear downsampling. When switching to a slightly different downsampling scheme (Bicubic), performance degrades slightly since the training data is less representative of this case. When switching to a substantially different downsampling scheme (Nearest Neighbor), performance degrades substantially.

In other words, the model is "overfit" to Bilinear downsampling, hurting generalization. Furthermore, the Residual model is more overfit than the Basic model as the drop in performance is more substantial. This is reasonable given its' head start discussed above, it has effectively had more epochs to optimize itself, overfitting to a greater degree.

6 Conclusion

For this approach to be useful in an application, it is crucial to consider what kind of downsampling is present in the final use case. The current approach is viable if the use case is truly to undo Bilinear downsampling (e.g., to undo compression). However, if the use case is to upsample arbitrary low-resolution images (e.g., those captured with a poor-quality camera), the training dataset may need to be crafted more carefully. Using a different downsampling scheme for training data or combining multiple schemes may help the model generalize as desired.