Test-Time Training for Image Super-Resolution

Zeeshan Patel*, Yossi Gandelsman
UC Berkeley
{zeeshanp, yossi_gandelsman}@berkeley.edu

Abstract

We introduce a novel approach for Test-Time Training (TTT) in Single Image Super-Resolution (SR), which enhances the performance of deep learning SR models in an online setting without requiring external data for pretraining. Our TTT for SR method consists of two primary components: self-supervised fine-tuning and model patching. The baseline SwinIR transformer model is fine-tuned for every new test image encountered, using a self-supervised learning task that generates data solely from the test image. By employing an iterative updating strategy, our model adapts to new test examples as they arrive, allowing knowledge from previous test examples to influence future inferences. To obtain the best results, we merge the the pretrained and fine-tuned TTT models using a linear interpolation of their learned weights. Our approach is applicable to various vision models and has the potential to advance generalization for several vision tasks.

1. Introduction

Generalization, a crucial aspect of supervised learning and intelligence, is often studied under the assumption that training and test data share the same distribution. However, this is rarely true in real-world scenarios, leading to models that struggle with adapting to distribution shifts during deployment. This issue inspires an alternative approach to generalization: adapting to the future as it unfolds rather than preparing for all possible scenarios [11, 13, 14, 30]. Test-time training (TTT) embodies this approach, leveraging the test input as a hint about the new distribution and modifying the model accordingly [3, 12, 29].

Although the test input lacks a ground truth label, self-supervised learning can generate labels from the input itself, enabling the model to adapt and better handle previously unseen distributions. The self-supervised learning task that we attempt to solve is crucial to TTT. We do not want to solve a task that is too simple or complicated since we want to ensure that there is some useful signal that can be derived from the test input [10, 29]. Therefore, we aim to identify a task with moderate difficulty that is also general enough to generate valuable features from the test input for our model.

In this work, we investigate how we can apply TTT to the problem of single-image Super-Resolution (SR). In recent years, we have seen deep learning techniques boost SR performance drastically over traditional methods [8, 18, 19]. Additionally, with new transformer models such as SwinIR, there has been a major change in performance for image restoration tasks using learning tehcniques [6, 21]. Deep learning techniques have been able to see massive improvements in SR due to well-engineered CNN and transformer architectures that have been trained on large labeled datasets. However, when these models are tested on images that are outside of the training distribution or have been downscaled in non-ideal conditions, most deep learning SR approaches tend to underperform [2]. Specifically, most models for SR are trained using high-quality, realistic images that were generated using a known downscaling kernel (i.e. bicubic kernel with antialiasing). With real low-resolution (LR) images though, we often see sensor noise, compression artifacts, or other disturbances and aberrations that can create problems during inference for SoTA deep learning SR techniques [2, 21]. Additionally, there are theoretically an extremely large number of test LR images that a SR model would encounter when deployed in real-time. The test distribution will continually change, which is why we need a generalization approach that can be adapted in real-time for optimal SR results.

In this paper, we introduce an algorithm for Test-Time training for single-image SR, which utilizes the power of self-supervised learning to generalize pretrained deep learning models for SR on unseen test distributions. At a high level, we set up a *one-sample learning problem* and use self-supervised learning to "overfit" our model to the LR test input. This process allows us to look into the future by generating a fine-tuned version of our model that will be adapted to the self-supervised task and will also maintain the previous knowledge of the SR task it learned from pretraining. In addition to the test-time training step, we linearly interpolate the weights of the pretrained model and TTT model by a weight α. This technique, which is known as model patch-

ing [15], creates a new model that is able to improve accuracy on specific tasks without worsening its performance on tasks where it already produces desirable results. In our algorithm, we run the model patching procedure on a variety of α values for each test image and choose the model that produces the best results in terms of PSNR. For our experiments, we used a pretrained SwinIR transformer as our baseline for $\times 4$ image SR [21]. In our experiments, we noticed that TTT outperforms the baseline SwinIR model in specific regions of the test input, and on average, TTT provides similar results to the pretrained SwinIR or slightly better results on certain test sets.

2. Related Work

A variety of approaches have utilized Test-Time Training (TTT) for generalization under distribution shifts across various problems. Furthermore, self-supervised image SR has also been explored with notable results. In this section, we first examine related works in this specific context, followed by a discussion on the two components of our algorithm: TTT and model patching.

2.1. Zero-Shot Image SR

Within the realm of zero-shot image super-resolution, an influential work worth mentioning is *"Zero-Shot" Super-Resolution with Deep Internal Learning* (ZSSR) by Shocher *et al*. In ZSSR, the authors train a lightweight CNN on a single test-image, specializing it for super-resolution of that specific test image [2]. This pioneering attempt at zero-shot SR demonstrated that high-quality super-resolved images surpassing state-of-the-art models can indeed be generated using no external data.

To justify their results, the authors employed the concept of Deep Internal Learning (DIL) in the ZSSR technique. DIL leverages the natural recurrence of internal information within natural images to train an accurate model using a single test image [32]. This recurrence of internal information in small image patches yields powerful internal image statistics that surpass general external statistics learned from an external training dataset [2, 32]. This crucial insight is also applicable to TTT for SR: by beginning with a pretrained model and exposing it to the test image's internal statistics, the model's performance can be enhanced at test-time. In ZSSR, the authors generated a training dataset for the self-supervised learning problem using a single test-image. They obtained this dataset by downscaling the LR test image I by the desired scaling factor s, and augmenting the lower-resolution image using rotations in 90° increments, as well as vertical and horizontal mirroring [2]. This approach resulted in substantial improvements in SR quality, with approximately $0.2dB$ gains in PSNR.

2.2. Test-Time Training

Test-Time Training (TTT) has gained traction in recent years, as more research has explored its potential to maintain robust models in real-time settings. There are numerous applications of continual model training during deployment, including facial recognition [16], video segmentation [27], object tracking [9], and medical imaging [17]. TTT offers a continual learning framework that addresses the generalization problem in supervised models. With TTT, a new model is trained for every test data point using self-supervision, enabling the pretrained baseline to adapt to any underlying distribution shifts.

In practice, TTT is similar the technique of Unsupervised Domain Adaptation (UDA). In UDA, we use labeled data from the training domain and unlabeled test data from the target domain to learn how to complete various tasks in the target domain [5,7,23,24]. In terms of super-resolution, this would mean using some LR-HR image pairs from the training data and some unlabled test data to fine-tune our pretrained model to learn how to adapt to our new test images. In TTT, we use a similar approach that is much more powerful: we reduce the target sample to the test image itself. This approach is much more precise and efficient as we no longer have the need to generalize to other data points from the test distribution, and we also do not need to generate a set of target samples to fine-tune our models. We can simply use the test image to generate a variety of LR-HR image pairs for our self-supervised SR task, and with this data, our model can learn to "overfit" on this test image [2, 10]. This process allows our pretrained model to continually update its parameters such that it can provide better inferences on test data in real-time.

Another noteworthy advantage of TTT is its ease of deployment and applicability in workflows requiring on-the-fly inferences. To deploy a TTT-based generalization mechanism, simply set up a self-supervision task using the test input and fine-tune the model for a small number of gradient steps [10,29]. This process can be efficiently parallelized to accelerate training and inference generation.

2.3. Model Patching

Model patching is founded on the idea that interpolating the weights of two distinct models, one pretrained model and another fine-tuned version of that same model for a specific task, can yield superior results for zero-shot performance [15]. The term *patching* stems from the belief that interpolating the weights of the pretrained and fine-tuned models will create a new model capable of improving performance on specific tasks without compromising accuracy on tasks where it already achieves satisfactory results. In [15], the authors demonstrated substantial performance improvements using their model patching technique on several zero-shot vision tasks. This work supports the idea of

employing model patching at test-time to merge our pretrained and TTT models, retaining the knowledge gained by the model from the training distribution while introducing new context regarding the unknown test distribution. Through model patching, a new model can be created that incorporates knowledge from the self-supervised SR task derived from the LR test image, as well as retaining knowledge obtained from pretraining to perform SR on training images.

3. Method

There are two primary components in our TTT for SR approach: self-supervised fine-tuning and model patching. The majority of our design effort was dedicated to developing a suitable self-supervised learning problem for TTT.

3.1. Model Architecture

The central idea of TTT is to fine-tune a pretrained model for every new test image encountered. For SR, we opted for the SwinIR [21] transformer model as our baseline pretrained model. SwinIR has established itself as the SoTA in several image restoration tasks, including SR. Nonetheless, our approach is not tied to a specific model architecture and can be applied to various vision models.

3.2. TTT Setup

To implement TTT, we first need to establish a self-supervised learning task for fine-tuning our model. Consequently, the initial step in setting up TTT is devising a mechanism to generate a dataset from a single test image for fine-tuning. We employ a strategy similar to [2] to exploit the internal image statistics from our test inputs. To generate our TTT training set, we begin by taking a sliding window of size 48×48 across the entire test image, forming a set of image patches. This initial set of patches ensures that all parts of the image are considered during TTT. We then randomly generate additional patches of the same size from the test image. These generated patches will serve as the HR targets for the self-supervised task. We downscale these patches by the desired scaling factor s to create the LR inputs for the SwinIR model during TTT. We also randomly augment these LR-HR pairs using rotations in 90° increments and vertical and horizontal mirroring.

Once the training set is generated, we proceed to fine-tune our model. Let M_{t_i} be the i^{th} TTT model, X be the set of input LR image patches for the self-supervised learning problem, and Y be the set corresponding original LR image patches from which we generated X. In other words, X_i is a downscaled version of Y_i, which was the orignal LR test image patch.

Then, our loss for TTT is defined as follows:

$$\mathcal{L}(M_{t_i}, X, Y) = \frac{1}{n}\sum_{i=1}^{n}(M_{t_i}(X_i) - Y_i)^2. \quad (1)$$

Here, we generate n training examples (X_i, Y_i) from the original LR input image. Our loss $\mathcal{L}$ is simply the mean-squared error (MSE) between the TTT model output $M_{t_i}(X_i)$ and the corresponding target Y_i. This loss function is suitable for our self-supervised task as we aim to optimize our PSNR results, which depend on the MSE between the LR and HR images.

Using this loss function, we can define the test error on our unknown test distribution Z. During TTT, we are updating the parameters θ_{t_i} of our i^{th} TTT model M_{t_i} by training on the test input X. Thus, our expected test error is the following:

$$\text{TTT Test Error} = \mathbb{E}_Z\left[\mathcal{L}(M_{t_i}, X, Y); \theta_{t_i}(X)\right]. \quad (2)$$

To select an optimizer, we experimented with several learning rates for stochastic gradient descent (SGD) with momentum, Adam [20], and AdamW [25]. We found that Adam and AdamW caused issues with TTT since our performance deteriorated after too many steps. With SGD, we observed improvements in performance even after 40 steps of TTT for certain test images. Using SGD enables our TTT model to learn to adapt to all distribution shifts without any early stopping mechanism.

To adapt our method for an online setting, we iteratively update our TTT models as new test examples arrive. In an online setting with test examples $X_1, X_2, \ldots, X_n$, we train our pretrained baseline model M_p with parameters θ_p and fine-tune it using TTT on X_1 to generate our first TTT model M_{t_1} with parameters θ_{t_1}. Then, we use TTT to fine-tune M_{t_1} on X_2, and we iteratively continue this process till we generate our final model M_{t_n} with parameters θ_{t_n}, which has been fine-tuned for input X_n. Using this strategy, our model can use knowledge from other test examples to influence its inference on future test inputs as they arrive [29].

3.3. Model Merging

Before generating inferences, we aim to merge the outputs of the pretrained model M_p and the fine-tuned TTT model M_t to obtain the "best of both worlds." During our experiments, we attempted to determine a theoretical upper bound for the best PSNR results after merging the pretrained and TTT outputs for our test images. Since we had access to the HR ground truth for our test sets, we merged the outputs of M_p and M_t by selecting pixels from each inference image with a lower MSE compared to the corresponding pixels in the HR image. This process demonstrated that TTT performs better in certain regions of our

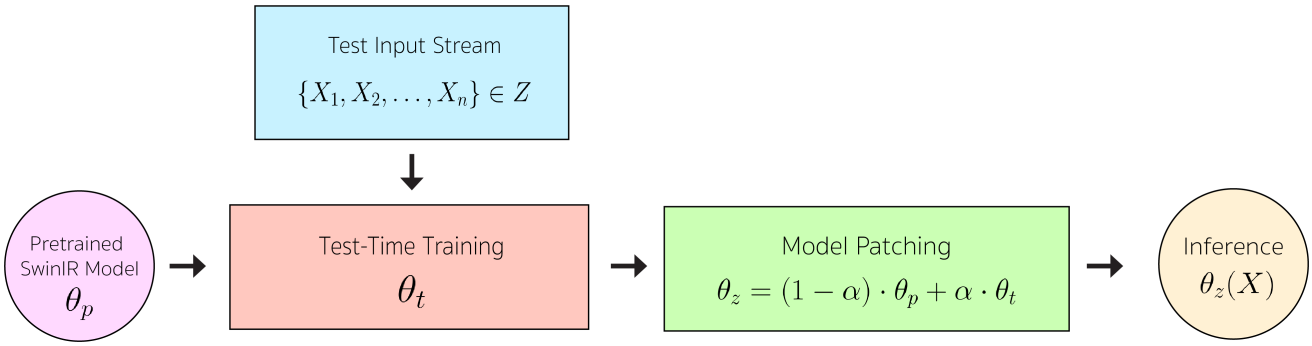

Figure 1. TTT for Image SR process.

test image than the baseline SwinIR model and vice versa. Initially, we planned to design another learning problem where we would train an additional model to learn where M_t performs better than M_p on our test image. However, this is a challenging problem to solve due to the large variance in our test images and the inconsistency in which regions of the test image M_t outperforms M_p. Instead, we chose to use model patching [15] to develop a cleaner, more generalizable approach for adapting our model to the test distribution while preserving knowledge from pretraining.

We merge the models M_p and M_t by taking an interpolation of their learned weights. Let θ_p be the pretrained model weights and θ_t be the TTT model weights. Then, we merge the two model weights as follows:

$$\theta_z = (1-\alpha) \cdot \theta_p + \alpha \cdot \theta_t, \tag{3}$$

where θ_z are the weights of the final merged model, M_z, and $\alpha \in [0, 1]$. We test several values for α and choose the model that generates the inference image with the best PSNR. The process of selecting α could be learned or decided through validation for a specific test distribution. The general process for our method is illustrated in Fig. 1.

4. Experiments and Results

4.1. Implementation Details

In all of our experiements, we use a SwinIR transformer as our baseline model that has been trained for 500K iterations on the DIV2K [1] dataset. This pretrained SwinIR model is fine-tuned with each individual test image in our test sets. Specifically, we tested our technique for $\times 4$ SR on the Set5 [4], Set14 [31], and BSD100 [26] datasets. After generating each of the fine-tuned TTT models, we use a weight merging mechanism similar to [15] to linerly interpolate the weights of the pretrained and fine-tuned models by some α. We use the performance of the inference versus the ground truth in our validation set to determing the best value of alpha for each model. We used the original implementation of the SwinIR model for TTT.

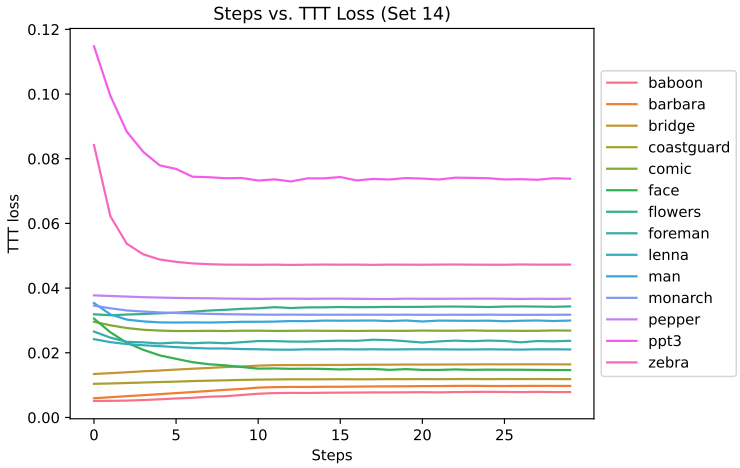

Figure 2. MSE loss during TTT for Set14 [31]. SGD keeps improving or maintains performance even after 30 steps.

We run TTT for 30 epochs for every test-image using SGD with a momentum of 0.9, weight decay of 0.2, and an initial learning rate of 1e-4. The learning rate is periodically updated by a learning rate scheduler that decreases the learning rate whenever a metric stops improving. We chose to perform TTT for 30 epochs since empirically, we noticed that TTT performance gains with more steps are marginal and would not impact the results significantly as seen in Fig. 2. Additionally, we needed to determine a suitable size for the dataset we generated for the self-supervised learning task during TTT. Since we needed to ensure our model is able to see all regions of the image, we first generated patches using a sliding window, and then we sampled additional patches from the image randomly. In order to truly "overfit" on the test image, we used 100 LR-HR pairs generated from the LR test input using the method described in Sec. 3.2. This provided us with sufficient data to learn internal image statistics and ensured that the model was learning new information during TTT. We use one NVIDIA GeForce RTX 2080 Ti GPU that performs TTT on each test image

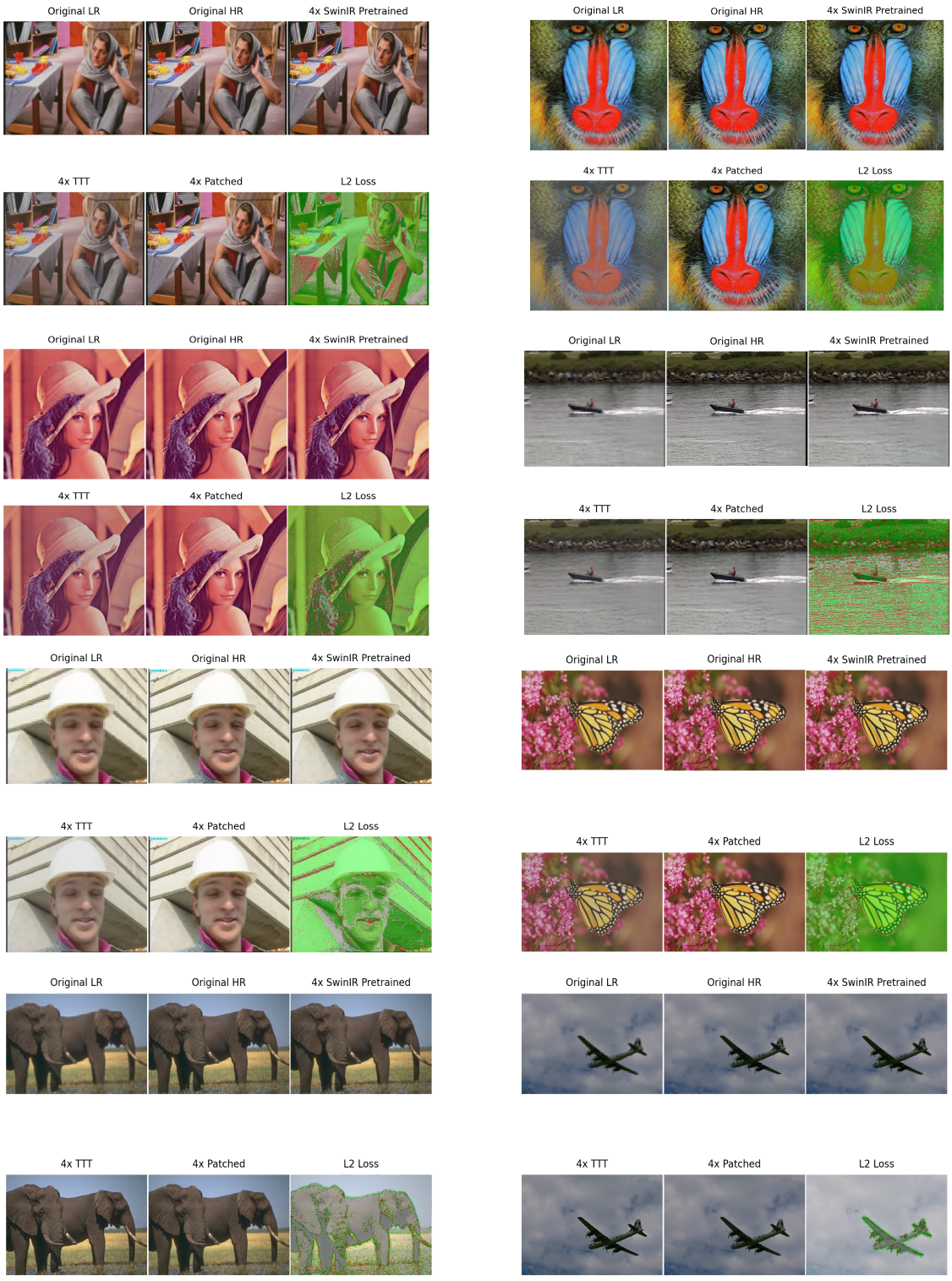

Figure 3. **SwinIR vs. TTT outputs**. Green pixels indicate SwinIR has lower error; Red pixels indicate TTT has lower error. TTT is able to perform better in areas with finer details. However, on some images, it does not improve the error and SwinIR outperforms TTT overall. Uncolored pixels on the ℓ_2 loss map indicate equal error between the SwinIR and TTT outputs. Patched images are generated by using the patched models merged with the best interpolation coefficient $\alpha \in [0, 1]$. We use validation on the same test image to choose α.

Model	Set5 [4]		Set14 [31]		BSD100 [26]	
	PSNR (Y)	SSIM (Y)	PSNR (Y)	SSIM (Y)	PSNR (Y)	SSIM (Y)
SwinIR baseline	32.72	0.9021	28.94	0.7914	27.83	0.7459
TTT (ours)	32.83	0.9024	29.05	0.7917	27.85	0.7459
Patched Model (ours)	32.75	0.9022	29.09	0.7917	27.87	0.7461

Table 1. Comparison of PSNR (Y) and SSIM (Y) for $\times 4$ single image SR. Metrics calculated in the Y-channel of the YCbCr space.

sequentially. However, TTT training can benefit from parallelism across multiple GPUs to improve training speed.

The final component of our method is model patching. Using the general approach outlined in [15], we merged the model weights for every value of alpha that we tested for each pair of SwinIR pretrained and TTT models as in Eq. (3). This process can be slow, so we implemented a distributed hyperparameter search to find the best value of α for every TTT model. This significantly improved model patching performance, especially for larger test sets such as BSD100 [26]. For model patching, we used 10 NVIDIA GeForce RTX 2080 Ti GPUs in parallel to decrease running time, but it is possible to obtain results in a reasonable amount of time with fewer GPUs.

4.2. Results

We generated results for Set5 [4], Set14 [31], and BSD100 [26] for a $\times 4$ scaling factor using our TTT method. We chose $\times 4$ specifically as it is reasonably hard to extract internal information as downscaling factor is increased, which would serve as an adeuquate challenge for TTT. However, in theory, our results would be similar for other scaling factors.

In Tab. 1, we show PSNR and SSIM results for the SwinIR baseline, TTT, and patched models. In Fig. 3, we display differences between the SwinIR baseline TTT, and patched model outputs. In these images, red pixels indicate that TTT outperformed the baseline SwinIR with respect to the ground truth, and the green pixels indicate that SwinIR performed better than the corresponding TTT models. Pixels that are not highlighted in green or red had similar performance in both SwinIR and TTT outputs. We used the ℓ_2 loss to determine which pixel performed better in the SwinIR and TTT outputs with respect to the ground truth.

Overall, we didn't see massive improvements in PSNR or SSIM using our method, on average. However, for specific test images, our method significantly improves the perceptual quality of the image. In Tab. 1, we can see that on average, our technique provided $0.1dB$ gains in PSNR for Set5 [4] and Set14 [31], but we did not see any significant difference in BSD100 [26]. Additionally, for all the test sets, we saw minimal differences in the SSIM.

In Fig. 3, we can see in the first four images being compared, the TTT model is able to capture finer details in the image better than the SwinIR model. For example, in the very first image, we can see that the TTT model has a lower error on the tablecloth and the woman's clothes. However, in other images, such as the image of the elephants from BSD100 [26], we see that the SwinIR model performs better on the outline of the elephants and TTT does not contribute in lowering the error by any significant amount. Although TTT is able to capture more information in some images, it tends to perform as well as the SwinIR model in many cases.

As mentioned in Sec. 3.3, we attempted to find a loose theoretical upper bound on the best PSNR/SSIM we could achieve by merging the SwinIR baseline outputs with the TTT outputs based off which corresponding pixels had a lower error with respect to the ground truth HR image. From this process, we determined that it would be possible to see upwards of approximately $1.3dB$ increases in PSNR, on average. In our real experiments though, we were not able to attain such results. We believe that this is due to model patching not accurately transferring the new knowledge learned during TTT. In addition to model patching, we also tried to merge the outputs of the SwinIR baseline and TTT models using different techniques such as classification, regression, or generative modeling. However, these approaches ultimately worsened our results and were unable to provide any performance improvements over the baseline. We also tested performance with varying numbers of TTT steps, but we did not notice any performance gains after 30 steps of TTT. However, we did notice the double descent phenomena [28] occur when testing TTT with AdamW. Specifically, we saw TTT loss increase after approximately 40 steps of TTT on Set14 [31] images. It eventually decreased below the previous minimum loss after about 80 steps. When testing these models, we did not see any performance improvements over TTT models that were trained for 30 steps.

We also noticed that TTT did not improve images with motion artifacts. However, the SwinIR baseline model also exhibited similar results. This can potentially mean that TTT cannot improve performance under massive distribution shifts [22] or that our self-supervised learning problem was not effectively helping the model learn new information in those situations.

Although our results are not significantly better than the

baseline model, we still believe that TTT can help with distribution shifts for SR tasks. We believe that our self-supervised learning problem is the main reason why our results were not significantly better than the baseline model. There are several ways to frame the self-supervised learning problem, and it is possible that extracting other image statistics from the test image can provide more useful information during TTT. This remains part of our future work.

5. Conclusion

In this work, we introduce Test-Time Training for Image Super-Resolution, which enables on-the-fly enhancement of deep learning-based SR models in an online setting. Our method eliminates the need for external data during pre-training and relies exclusively on the test image. With a self-supervised learning strategy, we fine-tune the baseline model at test-time, using data generated exclusively from the test image. This approach not only improves the perceptual quality of SR inferences at test-time but also offers an efficient mechanism for implementing continual learning for deep learning SR models. We anticipate that our contribution will inspire further advancements in Test-Time Training approaches for image restoration.

6. Acknowledgements

We would like to thank Alexei Efros for valuable technical discussions and his feedback on the paper. We also thank UC Berkeley for providing the computational resources for our experiments.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1122–1131, 2017. 4

[2] Michal Irani Assaf Shocher, Nadav Cohen. Zero-shot super-resolution using deep internal learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2, 3

[3] Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. Self-supervised test-time learning for reading comprehension, 2021. 1

[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie line Alberi Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of the British Machine Vision Conference*, pages 135.1–135.10. BMVA Press, 2012. 4, 6

[5] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. 2

[6] Marcos V. Conde, Ui-Jin Choi, Maxime Burchi, and Radu Timofte. Swin2sr: Swinv2 transformer for compressed image super-resolution and restoration, 2022. 1

[7] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey, 2017. 2

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015. 1

[9] Yang Fu, Sifei Liu, Umar Iqbal, Shalini De Mello, Humphrey Shi, and Jan Kautz. Learning to track instances without video annotations, 2021. 2

[10] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. Test-time training with masked autoencoders, 2022. 1, 2

[11] Robert Geirhos, Carlos R. Medina Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks, 2020. 1

[12] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment, 2021. 1

[13] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization, 2021. 1

[14] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *CoRR*, abs/1903.12261, 2019. 1

[15] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights, 2022. 2, 4, 6

[16] Vidit Jain and Erik Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR 2011*, pages 577–584, 2011. 2

[17] Neerav Karani, Ertunc Erdil, Krishna Chaitanya, and Ender Konukoglu. Test-time adaptable neural networks for robust medical image segmentation. *Medical Image Analysis*, 68:101907, feb 2021. 2

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2015. 1

[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution, 2016. 1

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[21] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer, 2021. 1, 2, 3

[22] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and

J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21808–21820. Curran Associates, Inc., 2021. 6

[23] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks, 2015. 2

[24] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks, 2017. 2

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 3

[26] David R. Martin, Charless C. Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2:416–423 vol.2, 2001. 4, 6

[27] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for efficient video inference, 2020. 2

[28] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt, 2019. 6

[29] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts, 2020. 1, 2, 3

[30] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks, 2017. 1

[31] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, 2010. 4, 6

[32] Maria Zontak and Michal Irani. Internal statistics of a single natural image. *CVPR 2011*, pages 977–984, 2011. 2