Test-Time Training with Masked Autoencoders

Anonymous Author(s)
Affiliation
Address
email

Abstract

Prior work has shown masked autoencoding as an effective self-supervised task across many visual distributions with sufficient training data. We use masked autoencoding to train on each unlabeled test sample as it arrives at test time, before making a prediction. This simple method improves generalization of predictive models on many visual benchmarks of unknown distributions, where input images are not covered by any training data. Our theoretical analysis explains how test-time training with autoencoding helps a linear model under distribution shifts. [1]

1 Introduction

The success story of deep learning has been about general methods that leverage data and computation instead of domain knowledge. When the test distribution is not restricted to be the same as that of training, test-time training [50, 22, 1, 49] is a paradigm of methods that takes this philosophy beyond the training data. It performs self-supervised learning for each unlabeled test input, before making a prediction on this input itself. Instead of anticipating a specific test distribution through prior knowledge, this paradigm generalizes to diverse kinds of distribution shifts by leveraging the extra input and computation at test time.

However, the generality of test-time training is still limited by its choice of the self-supervised task. Take the rotation prediction task used in [50], which hopes to form semantic features by learning the orientation of rotated images. For natural scenes, image rotation can be trivially predicted, without scene understanding, from the horizon's orientation alone. Even for centered objects, many classes, e.g. frogs and lizards, have mostly top-down views, for which every orientation looks equally plausible. In both cases, rotation prediction is unlikely to form semantic features as hoped for.

Latest advances in self-supervised learning suggest masked autoencoding as a potentially general-purpose task. This task is simple: remove parts of the data, and predict the removed content. In natural language processing, masked autoencoding [12] and the closely related task of autoregressive language modeling [39, 42, 4] have long been a staple of the community. Recent research in computer vision also shows that predicting missing patches on images can lead to state-of-the-art performance on a variety of downstream tasks and datasets [23, 2, 55].

Motivated by these successes, we experiment with masked autoencoding (MAE) as self-supervision for test-time training on four datasets. Our method leads to improvements in all of them. We also theoretically characterize the relationship between our method and the well known bias-variance trade-off in statistics.

[1] The code and the models will be made public upon acceptance.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

2 Related Work

2.1 Masked Autoencoders and Self-Supervised Learning

Key to the success of deep learning is the formation of semantic features that transfer across many tasks; the goal of self-supervised pre-training is to learn these transferable features without human annotations, by making up inputs and labels to an auxiliary pretext task. Given an original input image x, the pretext task of masked autoencoding [23] splits the image into many non-overlapping patches (e.g. 196), randomly masks out majority of the patches (e.g. 75%), and trains a model to reconstruct the missing patches, by optimizing the mean squared error between the reconstructed and the original pixels.

This model, called a masked autoencoder (MAE), is composed of two parts: a large encoder and a small decoder, both based on Vision Transformers [15]. For a downstream task, e.g. object recognition, the encoder takes in all the image patches and outputs the features, which then become input to a linear projection head. There are two common ways to combine the pre-trained encoder and untrained head. 1) Fine-tuning: both the encoder and head are trained, end-to-end, for the downstream task. 2) Linear probing: the encoder is frozen as a fixed feature extractor, and only the head is trained. We will refer back to these training options in Section 3.

2.2 Generalization under Distribution Shifts

Traditionally, training and test data are drawn i.i.d. from the same distribution; deep learning generalizes well by interpolating the training set with rather limited prior knowledge [56]. When training and test distributions are different, generalization is intrinsically hard since interpolation cannot cover many of the test samples [43, 44, 46, 14].

One way to turn extrapolation into interpolation again is to make the training distribution wider, and the training set larger, to cover the test distribution. This brute force approach, however, naturally falls short of scaling up to all possible distribution shifts in the world, even if they could be anticipated at training time. Training on many distribution shifts at once causes underfitting [57, 13]. The robustness obtained by training or fine-tuning on one type of distribution shift (e.g. Gaussian noise) also does not transfer to another (e.g. salt-and-pepper noise) [53, 18], even for visually similar corruptions.

To avoid the hardness of extrapolation, researchers work on alternative problem settings that anticipate the test distribution. For domain generalization [33, 32, 45, 40, 5, 17, 38, 21], a meta distribution generates many distribution shifts, some of which are used for training and others for testing. For domain adaptation [35, 36, 9, 6, 48, 28, 7, 20], a potentially unlabeled dataset from the test distribution is available for training. Solutions motivated by these settings share the same flaw as augmenting the training set: they are only effective on the particular meta / test distribution that was trained for.

2.3 Test-Time Training

Test-time training (TTT) [50] is a paradigm of methods that generalizes to distribution shifts without anticipating them, through self-supervised learning on each test input x. In [50], the main task is object recognition, and the self-supervised task is rotation prediction [18]: x is simply rotated in the image plane by multiples of 90 degrees to create a four-way classification problem.

The network for TTT is Y-shaped: a feature extractor f, followed simultaneous by a self-supervised head g and a main task head h. During training, labels are available for both the self-supervised task, using $g \circ f$, and the main task, using $h \circ f$; so both branches are trained jointly by summing their losses together for gradient descent. During testing, for each x, only the self-supervised task is labeled; so $g \circ f$ is trained, end-to-end, on this single input. This produces a different model, call it f_x (and g_x), initialized from f (and g), locally optimized for each test input x. After making a prediction on x, now with $h \circ f_x$, the algorithm never uses f_x (and g_x) again; the entire process repeats itself for the next input, again initializing from the old f (and g).

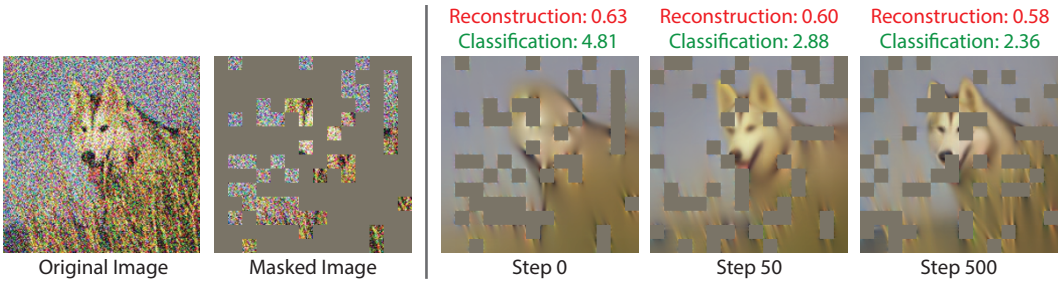

Figure 1: We train an MAE to reconstruct each test image at test time, with 75% of the input patches masked out. The three reconstructed images on the right visualize the progress of this single-input learning problem. Loss of the main task (green) – object recognition – keeps dropping even after 500 steps of gradient descent, while the network continues to optimize for reconstruction (red). The unmasked patches are not shown on the right since they are not part of the reconstruction loss.

TTT as explained above is not a new problem setting; it is a proposed solution to the canonical problem of generalization. Our paper stay in the same setting. Other followups have worked on new settings such as a batch [54] or dataset [34] of samples from the test distribution.

Before [50], single-image training has been performed for super-resolution [47]. The philosophy of TTT is also preceded by that of local learning [3] and transductive learning [52], although usually performed for support vector machines with multiple test samples [8, 29]. Since [50], TTT has been applied to many fields with domain-specific self-supervised tasks: vision-based reinforcement learning [22], model-based legged locomotion [49], tracking objects in videos [16], natural language question answering [1], and even medical imaging [31].

3 Method

At a high level, our method simply substitutes MAE [23] for the self-supervised part of TTT [50]. In practice, making this work involves many design choices.

Architecture. Like in [50], our architecture is Y-shaped. The feature extractor f is exactly the encoder of MAE, and g the decoder; each is a Vision Transformer (ViT) [15]. We intentionally avoid modifying them to make clean comparisons with results in [23]. Choice of the main task (object recognition) head h presents more opportunities. [23] uses a linear projection, from the encoder feature dimension to the number of classes; the authors discuss this as mostly a historic artifact of how people have been evaluating the quality of features in unsupervised learning. Therefore we also experiment with a more expressive main task head – an entire ViT-Base – to strengthen our baseline.

Training-time training. Following standard practice, we start from the MAE model provided by the authors, with a ViT-Large encoder, pre-trained for reconstruction on ImageNet-1k [11]. There are three ways to combine it with the untrained main task head: 1) Fine-tuning: train $h \circ f$ end-to-end. 2) Probing: train h only, with f frozen. 3) Joint training: train both $h \circ f$ and $g \circ f$, by summing the respective losses together. The first two are practiced by [23], only for linear h (see Subsection 2.1). The third is practiced by [50], for linear h, as well as a neural network (see Subsection 2.3). We experiment with all three, and choose probing with a ViT-Base head, a.k.a. *ViT probing*, as our default setup, based on the following considerations:

1) Fine-tuning does not work with TTT. As discussed in [23], reconstruction and recognition require very different features. Fine-tuning makes the encoder specialize to recognition, only to find it training for reconstruction again at test time and losing the recognition-only features that the head learned to rely on. This agrees with the intuition in [50] for joint training, instead of training only for recognition, as preparation for TTT.

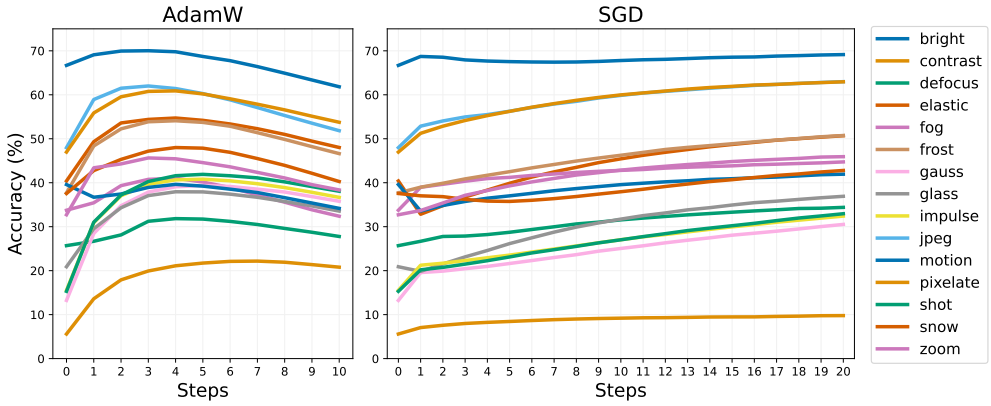

Figure 2: We experiment with two optimizers for TTT. MAE uses AdamW for pre-training, but using AdamW for TTT requires early stopping, which is unrealistic for generalization to unknown distributions. We instead use SGD, which keeps improving performance even after 20 steps.

2) ViT probing is much more lightweight than both fine-tuning and joint-training. It trains 3.5 times fewer parameters than even linear fine-tuning (86M vs. 306M). It also obtains higher accuracy than linear fine-tuning without aggressive augmentations on the ImageNet validation set, and accuracy similar to linear fine-tuning with augmentations, which [23] optimized heavily with many hyper-parameters that prevent overfitting. Altogether, we think that ViT probing strikes the right balance between the highly constrained linear probing and highly expressive linear / ViT fine-tuning.

3) Joint training is the recommended design in [50], but its accuracy on the ImageNet validation set, to the best of our ability, has been worse than the linear / ViT probing counterparts. This is most likely because we have not explored enough of the hyper-parameter space. As discussed in 2), fine-tuning is already heavyweight and hard to train; joint training is only heavier, and has more hyper-parameters. If future work finds a reliable recipe for joint training, our method will likely benefit from its setup. Meanwhile, we consider it our contribution to find ViT probing as a a lightweight substitute.

Augmentations. Our default setup, during training-time training, is to only use image cropping and horizontal flips for augmentation, follow the protocol in [23] for pre-training and linear probing. Fine-tuning in [23] (and [2, 55]), however, adds aggressive augmentations on top of the two above, including random changes in brightness, contrast, color and sharpness[2]. Many of them are in fact corruptions / distribution shifts contained in our evaluation benchmarks. Training with them is analogous to training on the test set, for the purpose of studying generalization to truly unknown distributions. Therefore, we choose not to use these augmentations, even though they would improve our results at face value.

Test-time training. The choice of optimizer for TTT is straightforward in [50]: it simply takes exactly the same optimizer setting as during the last epoch of training-time training of the self-supervised task. This choice, however, is not available for us, because the learning rate schedule of MAE reaches zero by the end of pre-training. We experiment with various learning rates for AdamW [37] and stochastic gradient descent (SGD) with momentum. Performance of both, using respectively the best learning rate, is shown in Figure 2.

AdamW is used in [23], but for TTT it hurts performance with too many iterations. On the other hand, more iterations with SGD consistently improve performance on all distribution shifts. Test accuracy keeps improving even after 20 iterations. This is very desirable for TTT: the single test

[2]Other augmentations used by fine-tuning in [23] are: interpolation, equalization, inversion, posterization, solarization, rotation, shearing, random erasing, and translation. These are taken from RandAugment [10].

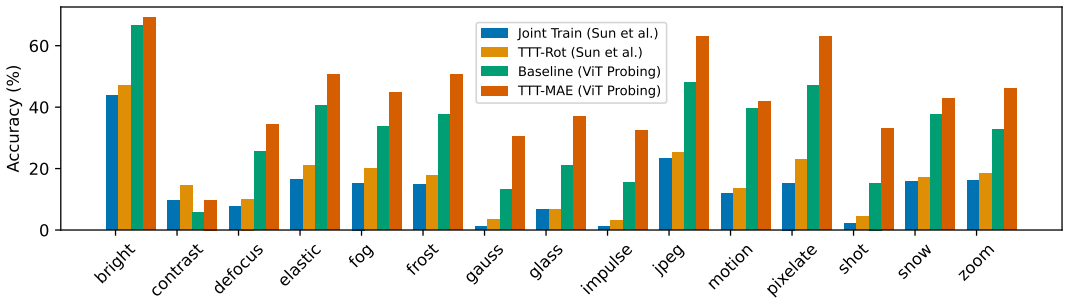

Figure 3: Results on ImageNet-C, see Subsection 4.2 for details. Our method, TTT-MAE, significantly improves accuracy on top of our baseline, which already outperforms the method of [50].

image is all we know about the test distribution, so there is no validation set to tune hyper-parameters, or monitor performance for early stopping. With SGD, we can simply keep training.

4 Empirical Results

4.1 Implementation Details

In all experiments we use the MAE based on ViT-Large, configured as in [23], pre-trained on ImageNet-1k [11] for 800 epochs. Our ViT-Base head takes as input the image features from the pre-trained MAE encoder. There is a linear layer in between that resizes those features to fit as inputs to the head, just like between the encoder and the decoder in [23]. A linear layer is also appended to the final class token of the head to produce the classification logits.

TTT is performed with SGD, as discussed, for 20 steps, using a momentum of 0.9, weight decay of 0.2, batch size of 128, and fixed learning rate of 5e-3. The choice of 20 steps is purely computational; more steps will likely improve performance marginally, judging from the trend observed in Figure 2. Most experiments are performed on one NVIDIA A100 GPU; hyper-parameter sweeps are ran on an industrial cluster with mostly V100 GPUs.

Like in pre-training, only tokens for non-masked patches are given to the encoder during TTT, whereas the decoder takes all tokens. Each image in a TTT batch has a different random mask. We use the same masking ratio as MAE [23] – 75%. We do not use any augmentation on top of random masking for TTT. Specifically, every iteration of TTT is performed on the same 224×224 center crop of the image as we later use to make a prediction; the only difference being that predictions are made on images without masking.

We optimize both the encoder and decoder weights during TTT, together with the class token and the mask token. We have experimented with freezing the decoder and found that the difference, even for multiple iterations of SGD, is negligible; see the appendix for detailed comparison. This is consistent with the observations in [50]. We also tried reconstruction loss both with and without normalized pixels, as done in [23]. Our method works well for both, slightly better for normalized pixels because the baseline is slightly better. We include these results in the appendix.

4.2 ImageNet-C

ImageNet-C [26] is a benchmark for object recognition under distribution shifts. It contains copies of the ImageNet [11] validation set with 15 types of corruptions, independent across images, each with 5 levels of severity.

In the spirit of treating these distribution shifts as truly unknown, we do not use training data, prior knowledge, or data augmentations derived from these corruptions. This complies with the stated rule of the benchmark, that the corruptions should be used only for evaluation, and the algorithms should not be corruption-specific [26]. This rule, in our opinion, is helpful for community progress:

	Frost		*JPEG*		*Brightness*		*Elastic*	
	[50]	Ours	[50]	Ours	[50]	Ours	[50]	Ours
Jellyfish	-3	**0**	-6	**5**	**2**	-1	-2	**3**
Holster	-1	**4**	-2	**6**	**1**	0	-1	**14**
Chiton	-1	**3**	-3	**8**	-1	**2**	-3	**8**
Fire salamander	1	**3**	-1	**3**	2	**5**	**2**	**2**
Spaghetti squash	-3	**4**	-2	**5**	2	**3**	1	**6**

Table 1: Changes after TTT in number of correctly classified images, out of 50 total for each category. On these rotation invariant classes, TTT-Rot [50] hurts performance, while TTT-MAE still helps, thanks to the generality of MAE as a self-supervised task; see details in main text. The four corruptions are selected for being the *most accurate* categories of the TTT-Rot baseline.

as explained in Subsection 2.2, the numerous distribution shifts outside of research evaluation cannot be anticipated, and algorithms that rely on information specific to the test distribution cannot scale.

Our main results on ImageNet-C appear in Figure 3. Following the convention in [50], we plot the accuracy only on the level-5 (most severe) corruptions. Results on the other levels are in the appendix. Our baseline is the fixed model without TTT, produced by ViT probing on a pre-trained MAE encoder as discussed, i.e. the default training method throughout this paper. Performing TTT-MAE on our baseline significantly improves performance.

Reconstruction vs. rotation prediction. The baseline for TTT-Rot, taken from [50], is called Joint Train in Figure 3. This is a ResNet [24] with 16-layers, trained jointly for rotation prediction and object recognition. Because our baseline is much more advanced than the ResNet baseline, the former already outperforms the reported results of TTT-Rot, for all corruptions except contrast. [3] So comparison to [50] is more meaningful in relative terms: TTT-MAE has higher performance gains in all corruptions than TTT-Rot, on top of their respective baselines. [4]

Rotation invariant classes. As discussed in the introduction, many self-supervised tasks like rotation prediction [18] are designed for specific domains. This can be especially limiting when the goal is, in fact, to generalize to unknown domains. Here we show examples in ImageNet where the assumptions behind rotation prediction do not hold.

We find rotation invariant classes through the following: 1) Take the ImageNet pre-trained ResNet-18 from the official PyTorch website [41]. 2) Run it on the original validation set, and one copy of it with 90-degree rotation. 3) Filter out classes for which the original accuracy is lower than 60%. 4) For each class, compare accuracy for the original and the rotate images, and choose the 5 classes for which the normalized difference is the smallest.

Sample images from these 5 classes are shown in Table 1. Not surprisingly, these images are usually taken from top-down views; rotation prediction on them can only memorize the generated labels,

[3] It turns out that for contrast, the ResNet baseline of TTT-Rot is somehow much better than our ViT baseline; TTT-MAE improves both baselines nevertheless.

[4] For fair comparison, we have also attempted to implement the rotation prediction task of [50] on top of our ViT probing baseline, as a simple substitution to the reconstruction task of [23]; the rotation prediction head here is the same as the main task head. However, even after joint training in the fashion of [50], TTT for this model with rotation still hurts performance on every corruption. Please see appendix for details.

	brigh	cont	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom
Joint Train	62.3	4.5	26.7	39.9	25.7	30.0	5.8	16.3	5.8	45.3	30.9	45.9	7.1	25.1	31.8
Fine-Tune	67.5	7.8	33.9	32.4	36.4	38.2	22.0	15.7	23.9	51.2	37.4	51.9	23.7	37.6	37.1
Probing	68.3	6.4	24.2	31.6	38.6	38.4	17.4	18.4	18.2	51.2	32.2	49.7	18.2	35.9	32.2
TTT-MAE	**69.1**	**9.8**	**34.4**	**50.7**	**44.7**	**50.7**	**30.5**	**36.9**	**32.4**	**63.0**	**41.9**	**63.0**	**33.0**	**42.8**	**45.9**

Table 2: Accuracy (%) of the three design choices discussed in Section 3; see details in main text.

without forming semantic features as intended. This causes TTT-Rot of [50] to harm performance, as seen in Table 1. Also not surprisingly, TTT-MAE is agnostic to rotation invariance and still helps.

Training-time training. In the synonymous paragraph in Section 3, we discussed our choice of ViT probing instead of fine-tuning or joint training. Table 2 compares accuracy of these three choices under our default setting. As discussed earlier, joint training does not achieve satisfactory performance in most categories; while fine-tuning is initially better than probing, it is not amenable to our method. In the end, TTT-MAE on top of probing outperforms fine-tuning for all the corruptions. Numbers in the last two rows are the same as in Figure 3.

4.3 Other ImageNet Variants

We evaluated TTT-MAE on two other popular benchmarks for distribution shifts. ImageNet-A [27] contains real-world, unmodified, and naturally occurring examples where popular ImageNet models perform poorly. ImageNet-R [25] contains renditions of ImageNet classes, e.g. art, cartoons and origami. Both TTT-MAE and the baseline are the same models and algorithms, with exactly the same hyper-parameters as for ImageNet-C (Subsection 4.2). Our method continues to outperform the baseline by large margins, see Table 4.3.

4.4 Portraits Dataset

The Portraits dataset [19] contains American high school yearbook photos labeled by gender, taken over more than a century. It contains real-world distribution shifts in appearances over the years. We sort the entire dataset by year and split it into four equal parts, with 5062 images each. Sample images from the four splits are shown in Figure 4. Between the four splits, there are visible low-level differences such as blurriness and contrast, as well as high-level change like hair-style and smile.

We create three experiments: for each we train on one of the first three splits and test on the fourth. Performance is in terms of accuracy in binary gender classification. As expected, performance increases slightly as the training split gets closer to test in time.

Our model is the same ImageNet pre-trained MAE + ViT head (with sigmoid for binary classification), and probing is performed on the training split. We use exactly the same hyper-parameters for TTT. Our results are shown in 4: our method improves on the baseline for all three experiments.

ImageNet-A		*ImageNet-R*	
Baseline	Ours	Baseline	Ours
15.3	**21.3**	31.3	**38.9**

Table 3: Accuracy (%) on ImageNet-A [27] and ImageNet-R [25]; see Subsection 4.3 for details. Our baseline is ViT-Probing, as default, trained on the original ImageNet.

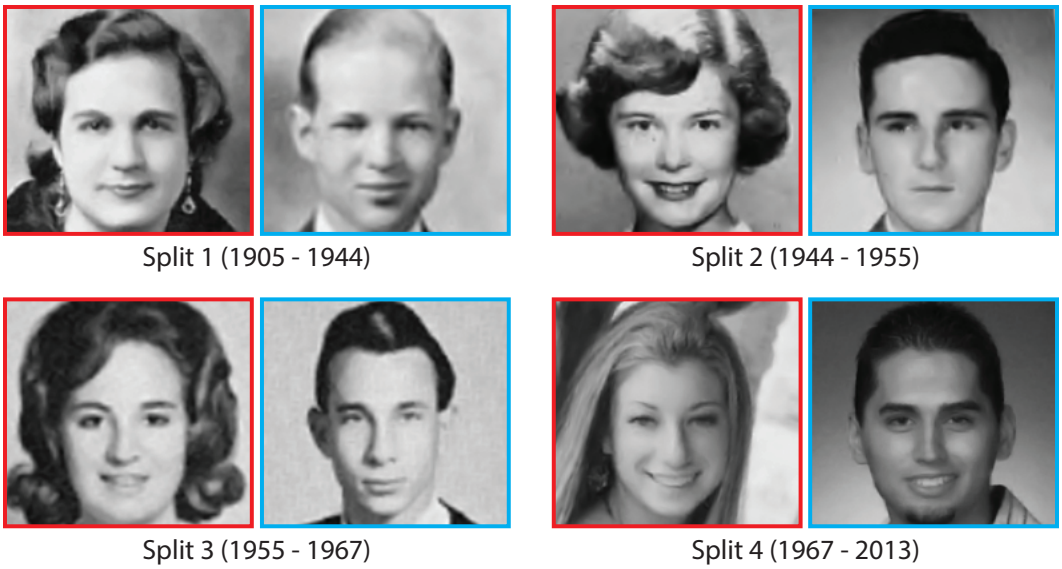

Figure 4: Sample images from our four splits, by year, of the Portraits Dataset.

Split (train)	1	2	3
Baseline	76.1	76.5	78.2
TTT-MAE	**76.4**	**76.7**	**79.4**

Table 4: Accuracy (%) for binary classification of gender. For each column, we train on the indicated split, and test on the fourth. Our method improves on the baseline for all three training splits.

5 Theoretical Results

Why does TTT help? The theory of [50] gives an intuitive but shallow explanation: because the self-supervised task happens to propagate gradients that correlate with those of the main task. But this evasive theory only delegates one unknown – the test distribution, to another – the magical self-supervised task, without really answering the question.

In this paper, we do not want to assume an oracle, i.e. domain expert, on the supposedly unknown test distribution, handing over a self-supervised task with the desirable properties. Algorithmically, we have experimented with masked autoencoding to demonstrate its generality. Theoretically, in this section, we explain why TTT with autoencoding helps in the linear world, in simple mathematical terms. This analysis is possible thanks to the well known equivalence between linear autoencoding and principle component analysis (PCA).

The most illustrative insight, in our opinion, is that under distribution shifts, test-time training finds a *better trade-off between bias and variance* than applying a fixed model. And in our context, bias means *dataset bias* in its colloquial sense [51]. We begin by setting up the problem.

Problem setup. Throughout this section, we denote a_i as the ith column of matrix A. Let $x, y \sim P$, the training distribution, and assume that the population covariance matrix $\Sigma = \mathrm{Cov}(x)$ is known. PCA performs spectral decomposition on $\Sigma = UDU^\top$, and takes the top k eigenvectors $u_1, \ldots, u_k$ to project $x \in \mathbb{R}^d$ to $\mathbb{R}^k$.

Assume that for each x, the ground truth y is a linear function of the PCA projection with $k = 1$:

$$y = w u_1^\top x, \tag{1}$$

for some known weight $w \in \mathbb{R}$. For mathematical convenience, we also assume the following about the eigenvalues, i.e. the diagonal entries of D:

$$\sigma_1 > \sigma_2 = \sigma_3 = \ldots = \sigma_d = \sigma. \tag{2}$$

At test time, nature draws a new $x, y \sim P$, but we can only see $\tilde{x}$, a corrupted version of x. We model a corruption as an unknown orthogonal transformation R, and $\tilde{x} = Rx$.

Algorithms. The fixed model baseline is to blithely apply our old procedure and predict

$$\hat{y} = wu_1^\top \tilde{x}, \tag{3}$$

as an estimate of y. Intuitively, this will be inaccurate if corruption is severe i.e. R is far from I. And here is the PCA version of TTT. We form a new (and rather trivial) covariance matrix with $\tilde{x}$, the only piece of information we have about the corruption. Let $\alpha \in [0, 1]$ be a hyper-parameter, and the linear combination of Σ with our new covariance matrix be

$$M(\alpha) = (1 - \alpha) \cdot \Sigma + \alpha \cdot \tilde{x}\tilde{x}^\top. \tag{4}$$

Denote its spectral decomposition as $M(\alpha) = V(\alpha) \cdot S(\alpha) \cdot V^\top(\alpha)$. We then predict

$$\hat{y} = wv_1^\top \tilde{x}. \tag{5}$$

Bias-variance trade-off. $\alpha = 0$ is equivalent to the baseline, and $\alpha = 1$ means that we exclusive use the one test input and completely forget about the training features. In statistical terms, α presents a bias-variance trade-off: $\alpha \downarrow 0$ means more bias, $\alpha \uparrow 1$ means more variance.

Theorem. Define the prediction risk as $\mathbb{E}[|\hat{y} - y|]$, where the expectation is taken over the corrupted test distribution. This risk is strictly dominated when $\alpha = 0$. That is, TTT with some hyper-parameter choice $\alpha > 0$ is, on average, strictly better than the baseline.

The proof is given in the appendix.

Remarks on assumptions. The assumption in Equation 1 is mild; it basically just says that PCA is at least helpful on the training distribution. It is also mild to assume that Σ and w are known in this context, since the estimated covariance matrix and weight asymptotically approach the population ground truth as the training set grows larger. The assumption on the eigenvalues in Equation 4 greatly simplifies the math, but a more complicated version of our analysis should exist without it. Lastly, we believe that our general statement should still hold for invertible linear transformation instead of orthogonal transformations, since they share the same intuition.

6 Discussion

Limitations. Due to the additional computation at test time, our method is slower than the baseline. Inference speed has not been the focus of this paper, and it can potentially be improved through better optimizers and hyper-parameters. In general, test-time training has been limited by the deep learning toolbox. Most tools, for architectural design, regularization and optimization, have been refined for training-time training on large datasets, especially for large models like ours; this is another direction for future work.

Broader impact. In his book *Thinking, Fast and Slow*, Nobel Prize Winner Daniel Kahneman [30] describes a dichotomy between two kinds of thinking: System 1, the reactive and intuitive, and System 2, the deliberate and logical. Consensus holds that deep learning systems today are almost exclusively System 1. Pushing them closer to System 2 has the potential to make artificial intelligence more reliable, interpretable and transparent, and better vehicles to understand human intelligence. We believe that our work is a small step towards this goal, by allowing machine learning model to slowly refine their predictions beyond a reactive feed-forward process. Choosing a general self-supervised learning task makes this kind of "thinking" less affected by biases of human experts, and more attentive to each individual at test time.

References

[1] Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. Self-supervised test-time learning for reading comprehension. *arXiv preprint arXiv:2103.11263*, 2021.

[2] Hangbo Bao, Li Dong, and Furu Wei. Beit: BERT pre-training of image transformers. *CoRR*, abs/2106.08254, 2021.

[3] Léon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural computation*, 4(6):888–900, 1992.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

[6] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in neural information processing systems*, pages 2456–2464, 2011.

[7] Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570, 2018.

[8] Ronan Collobert, Fabian Sinz, Jason Weston, Léon Bottou, and Thorsten Joachims. Large scale transductive svms. *Journal of Machine Learning Research*, 7(8), 2006.

[9] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[10] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719, 2019.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Samuel Dodge and Lina Karam. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119*, 2017.

[14] Samuel Dodge and Lina Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–7. IEEE, 2017.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[16] Yang Fu, Sifei Liu, Umar Iqbal, Shalini De Mello, Humphrey Shi, and Jan Kautz. Learning to track instances without video annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8680–8689, 2021.

[17] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[19] Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A. Efros. A century of portraits: A visual historical record of american high school yearbooks. *CoRR*, abs/1511.02575, 2015.

[20] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012.

[21] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[22] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] 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. *ICCV*, 2021.

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

[27] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

[28] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[29] Thorsten Joachims. *Learning to classify text using support vector machines*, volume 668. Springer Science & Business Media, 2002.

[30] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

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

[32] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. *arXiv preprint arXiv:1902.00113*, 2019.

[33] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018.

[34] 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? *Advances in Neural Information Processing Systems*, 34, 2021.

[35] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[36] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[38] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching. In *International Conference on Machine Learning*, pages 7313–7324. PMLR, 2021.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[40] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18, 2013.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Preprint*, 2018.

[43] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

[44] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019.

[45] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[46] Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. Do image classifiers generalize across time? *arXiv*, 2019.

[47] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, 2018.

[48] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint*, 2019.

[49] Yu Sun, Wyatt L Ubellacker, Wen-Loong Ma, Xiang Zhang, Changhao Wang, Noel V Csomay-Shanklin, Masayoshi Tomizuka, Koushil Sreenath, and Aaron D Ames. Online learning of unknown dynamics for model-based controllers in legged locomotion. *IEEE Robotics and Automation Letters*, 6(4):8442–8449, 2021.

[50] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[51] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[52] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[53] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016.

[54] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[55] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[56] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[57] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code and data will be released upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] Following the convention of [23] and many other large-scale experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] In appendix
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]