Topic E - Test-Time Training with Masked Autoencoders

Lucas Versini

January 10, 2025

Abstract

Test-Time Training is a method designed to improve generalization by leveraging information about the data distribution found in the test samples, enabling the model to update its parameters during testing in a self-supervised way. The goal of this project is to investigate the effectiveness of the Test-Time Training technique described in [1], while exploring ways to enhance it, such as online adaptation method.

1. Introduction and background

1.1. Distribution shift

Distribution shift occurs when the data distribution at test time differs from the training distribution, which is a common challenge in real-world applications.
For models trained under such shifts, the performance is often low on the test data due to the models' inability to properly generalize to these new, unseen conditions.

1.2. Test-Time Training

Test-Time Training (TTT) [3] is a way for a model to keep learning features during test time.
The idea of [1] is to use a model that is Y-shaped, and which consists of:

- A feature extractor (f): Encodes the input data into a feature representation.
- A self-supervised head (g): Optimizes for a self-supervised task, such as reconstructing masked inputs.
- A main task head (h): Responsible for the primary prediction task, which is classification in our case.

During training, f and h are trained on the classification task, resulting in f_0 and h_0.
Then, during test time, h is frozen, and f and g are optimized on a self-supervised task. Specifically, given an image x, $g \circ f$ takes as input a masked version of x, and its goal is to reconstruct x:

$$f_x, g_x = \underset{f,g}{\operatorname{argmin}}\, \ell_s(g \circ f(\text{mask}(x)), x). \tag{1}$$

Then, classification is performed on x using $h_0 \circ f_x$.
Finally, f_x and g_x are discarded, and only f_0 and h_0 are kept: the model trained on x is not used for the next images.

Algorithm 1 contains a simplified version of the algorithm.

Algorithm 1 TTT Algorithm

for each test image x **do**
  Generate augmented variations of x
  Approximate f_x and g_x using the augmented images and (1), starting from f_0, g_0
  Predict the class of x using $h_0 \circ f_x$
  Discard f_x and g_x
end for

1.3. Dataset

For evaluation, TTT-MAE [1] uses the ImageNet-C dataset [2], which was designed to measure model robustness to common corruptions. It contains the original ImageNet validation set subjected to 15 types of corruptions (e.g., noise, blur, weather distortions) across 5 levels of severity. Each corruption includes 50 images per class for $1,000$ classes, hence a total of $15 \times 5 \times 1,000 \times 50 = 3,750,000$ images.

2. Results of TTT-MAE

2.1. Comparison to the baseline

To compare the baseline to TTT-MAE, we focused on seven types of corruption: *contrast*, *elastic transformation*, *gaussian noise*, *impulse noise*, *JPEG compression*, *pixelate* and *shot noise*, each with the maximum level of severity (5).
Due to computational limitations, we restricted ourselves to a batch size of 16 (compared to 128 in the paper), and considered only the first 5 images of each class, resulting in a total of $5,000$ images for each corruption type.
We reproduced part of Table 9 from [1], as shown in Table 1.
We obtained results similar to those from the article, though with a lower accuracy, likely due to our lower batch size.

2.2. Impact of the number of steps

The impact of the number of steps on the accuracy for some types of corruption is shown in Fig. 1.
In general, increasing the number of steps improves the accuracy. For *contrast* however, the accuracy stabilizes after

	cont	gauss	impul	pixel	shot	elast	jpeg
Baseline	7.0	17.2	18.3	50.0	17.9	!!	!!
TTT-MAE (article)	9.2	25.1	27.0	59.9	27.0	45.5	59.9
TTT-MAE (us)	8.8	21.2	22.4	59.0	23.0	36.8	55.3

Table 1. Accuracy (%) on ImageNet-C, level 5, after 10 steps of TTT, our results (batch size 16; $5,000$ images) and the article's results (batch size 128; $50,000$ images)

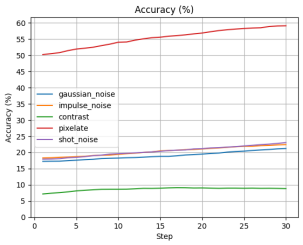

Figure 1. Accuracy (%) versus number of steps

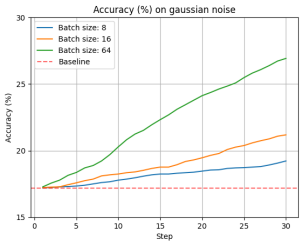

Figure 2. Impact of the batch size on the accuracy (gaussian noise)

approximately 15 steps, and slightly decreases after about 25 steps.

2.3. Impact of the batch size

Fig. 2 illustrates the effect of different batch sizes ($8, 16$ and 64) on the accuracy for the gaussian noise.

As expected, a larger batch size yields a higher accuracy. After 30 steps, the respective accuracies for a batch size of $8, 16$ and 64 were $19.22\%, 21.18\%$ and 26.92% respectively. In comparison, the baseline accuracy was 17.18%.

3. Online TTT

As explained previously, in the version of TTT used in [1], f_x and g_x are discarded once classification has been performed on image x.

However, we can implement an online version of TTT, where for an image x_{t+1}, we optimize (1) using f_{x_t} and g_{x_t} as starting points. Specifically, in Algorithm 1, we omit line 5 (Discard f_x and g_x), and in line 3 we start from f_{x_t} and g_{x_t}.

The idea is that in order to classify x_{t+1}, the model does not rely solely on x_{t+1}, but also on the images previously seen during test time $x_1, \dots, x_t$.

We implemented this version of TTT and explored different settings: now that the models are not discarded after each image, the order of the images can affect the results.

We focus on the *digital* transformations, composed of *contrast*, *elastic transformation*, *pixelate*, and *JPEG compression*.

3.1. Experiments on a single type of corruption

First, we consider a single type of corruption, *elastic transformation*, with the highest severity level, 5.

We compare three different settings:

- `TTT`: The images are processed class by class. f_x and g_x are *never* discarded.
- `TTT 1 model per class`: f_x and g_x are discarded when changing of class.
- `TTT random order`: The images are processed in a random order. f_x and g_x are *never* discarded.

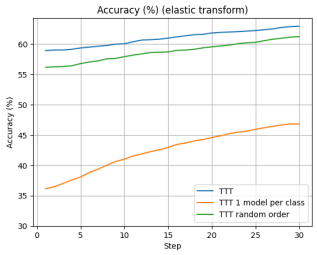

Figure 3. Online TTT for *elastic transform* with severity 5, batch size 16, 5 images per class, 30 steps

Fig. 3 shows the results of TTT on *elastic transform* under these three settings.

We see that using a different model for each class (`TTT 1 model per class`) yields poor results, with the accuracy dropping from more than 60% to less than 50% compared to `TTT`. This shows that reinitializing the model for each class fails to properly use the accumulated information from previously seen images.

Additionally, we see that processing images class by class (`TTT`) leads to a higher accuracy than processing images in random order (`TTT random order`). This suggests that even for a given corruption, when similar images are grouped together, the model benefits from incremental learning on similar data, hence an improved performance.

3.2. Single corruption type, several severities

We still focus on *elastic transform*, but this time we consider all severities between 1 and 5. We consider the two following settings:

- The images are seen by *increasing severity*.
- The images are seen in a random order.

In each case, we only used $1,000$ images for each severity, hence a total of $5,000$ images.

Fig. 4 shows the accuracy obtained after 30 steps for each severity, for both configurations.

We see that processing images with increasing severity is not necessarily beneficial.

We also ran an experiment with basic online TTT with only severity 5, as in the previous subsection, but with only $1,000$ images, so that we can compare the results with

Figure 4. Online TTT for *elastic transform* with all five severities, batch size 16, 1,000 images per severity, 30 steps. Results with ordered and shuffled severities.

the experiment considered here. The results presented in Table 2 clearly show that using other severities helps the model for severity 5.

	Severity 5	Severities 1-5 (Ordered)	Severities 1-5 (Shuffled)
Accuracy	56.3	70.7	70.9

Table 2. Accuracy (%) on *elastic transformation*, severity 5, when processing only severity 5 images, or all images.

3.3. Several corruption types

Now, we investigate the impact of the corruption type, by using all corruptions from *digital*, each with severity 5. Once again, we consider two configurations:

- The images are seen in a random order.
- The images are seen corruption type after corruption type.

The results are presented in Table 3. The results are not really encouraging. Even though these different corruptions all belong to *digital*, they may not be close enough to each other to make it interesting to train a same model on all of them.

	cont	elast	pixel	jpeg
Ordered	4.4	50	60.6	65.2
Shuffled	22.9	53.7	61.3	63.2

Table 3. Accuracy (%) on *digital*, severity 5, when processing images in a random order, or corruption after corruption.

4. Image analysis

To analyze failures cases, we used the setting `TTT random order` from 3.1, and used all 50 images from 100 classes, subject to *elastic transformation* with severity 5. The distribution of the accuracies is shown in Fig. 5, and images and accuracy for two specific classes are shown in Fig. 6 and Fig. 7. What we observed in practice, based on the observations for many classes and several corruption types, is that the classes for which the accuracy is particularly low contain images in which the object of interest is small, like in Fig. 6. Consequently, when applying a mask to the image, the object of interest can be totally masked, preventing the model from correctly classifying the image.

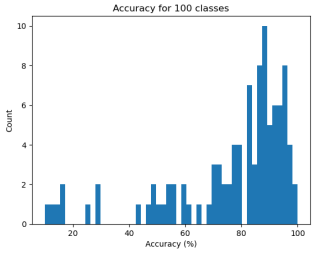

Figure 5. Accuracy (%) of online TTT on 100 classes, 50 images per class.

5. Choice of the masked regions

In [1] and in our previous experiments, the areas masked in the images were chosen randomly.
We tried to analyze the impact of choosing specific areas, such as edges, corners, flat areas, etc. However, our experiments did not yield any interesting results.
We also tried to use different mask proportions. The results are shown in Table 4.

Mask proportion (%)	50	55	60	65	70	75	80	85	90
Accuracy (%)	58.0	57.2	61.0	61.8	62.6	62.8	62.7	58.1	56.6

Table 4. Impact of the mask proportion (%) on the accuracy (%).

We see that the mask proportion should not be too low nor too large, otherwise the task would be too easy or too hard. This probably explains why the authors of [1] used a mask proportion of 75%, which is approximately the proportion that yields the best results.

6. Conclusion

As a conclusion, it appears that using TTT-MAE indeed improves the accuracy.
We have seen that using online learning yields even better results, conditioned to the fact that the test images are subject to similar corruptions (but potentially with different levels of severity).
A possible idea would be to try and use other self-supervised tasks, and to analyze the impact of these tasks on the obtained accuracy.

A. Appendix

The version of the code we used for this project can be found on:
https://github.com/lucas-versini/RecVis.

Figure 6. Some images of the class *Tarantula*. With online TTT, an accuracy of 14% is reached.

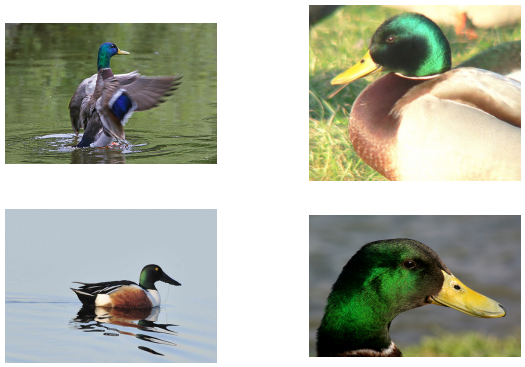

Figure 7. Some images of the class *Drake*. With online TTT, an accuracy of 100% is reached.

References

[1] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

[2] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[3] 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.