Final Project - Test-Time Training with Masked Autoencoders (E)

January 2025
Marc Boëlle and Tess Breton

Project Overview

For this final project, we worked on the paper "Test-Time Training with Masked Autoencoders" by Gandelsman *et al*. [1]. In this report, we first briefly introduce the problem and explain the approach developed by the authors. Then, we present the results of our experiments and finally go though our online implementation of the method.

Contributions: Tess worked on reproducing ImageNet-C results and analyzing failure cases, while Marc focused on understanding the method and implementing the online version.

1. Problem

The paper addresses the challenge of generalization under distribution shifts between the training and test data. In real-world applications of deep learning, the training and test distributions often differ. For instance, this can occur with sensor data, where the sensor's response may change over time due to wear and tear. However, most models are frozen after training. Thus, they need to be robust to any possible distribution shift (*e.g*. noise in images) they might encounter in the test data.

In computer vision, a common solution consists in artificially broadening the training distribution, *e.g*. with data augmentation. Although this technique often improves performance, it requires guessing future distribution shifts. If the shifts observed were not considered during training, then the model is likely to perform poorly. In addition, using data augmentation may degrade performance on specific distributions.

Another approach used in the paper is test-time training (TTT), which dynamically adapts the model to each test input.

2. Method

2.1. Test-Time Training

Test-time training aims at improving the robustness of the model to distribution shifts in the test data, by adapting it on-the-fly at test time. Yet test inputs do not come with ground-truth labels. Then, the idea is to guide the adaptation using a self-supervised task. The self-supervised task needs to be carefully designed so that it brings useful information for the main task.

During training, the main task (*e.g*. classification) and the self-supervised task are jointly optimized. At test time, the model adapts to each test input on the self-supervised task, and then makes a prediction on the main task.

2.2. TTT with Masked Autoencoders

In the paper, the main task is image classification and the distribution shifts correspond to visual corruptions such as noise or blur (see Sec. 3.1). The self-supervised task is image reconstruction with a masked autoencoder (MAE). Given an input image with masked patches, the MAE aims at reconstructing the missing patches to recover the original image.

Regarding the training setup, the autoencoder is first pre-trained for masked reconstruction on ImageNet-1k. Then, the classification head is trained with the frozen updated encoder. The encoder, decoder and classification head are all vision transformers.

At test time, the MAE is first updated to minimize the reconstruction loss on the test image considered. The MAE is trained for a few steps, using batches of random masks on the same input. Then, the frozen head is used for classification, taking as input the features from the updated encoder. Finally, the weights of the MAE are reset to the pre-trained version for the next test image.

Figures 6, 7 and 8 in the Supplementary Material provide visual representations of these different phases.

3. Results on ImageNet-C

3.1. Dataset

ImageNet-C was built from the ImageNet-1k validation set, with various corruptions applied. Each corruption has 5 levels of intensity and we focused on level 5, the most severe one. Due to limited resources, we decided to work with fewer corruptions, and picked gaussian noise, motion blur and elastic transform shown in Fig. 1.

Figure 1. ImageNet-C corruptions

3.2. Experiments

For our experiments setup, we built on the authors' code and pretrained models, but used only 10% of the data due to limited resources. We kept the same hyperparameters, except for the test-time training batch size to speed up runs. We took 32 instead of 128 and even so, making one full pass through our 5,000 images took 12 hours.

The results obtained on all three corruptions are shown in Fig. 2. As in the paper, we do get better results with test-time training, but our accuracies are not as good as the authors'. We suspect that this is due to our smaller batch-size, which likely affected the quality of our encoded features.

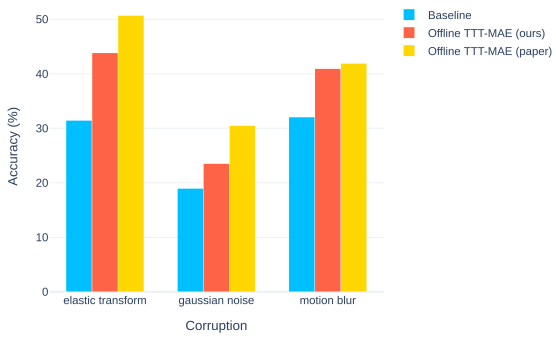

Figure 2. Accuracies of the baseline and TTT-MAE

These results were obtained with 20 TTT steps, but we also investigated the impact of this number of steps on performance. The accuracies obtained at each step for all three corruptions are displayed in Fig. 3. For each corruption, the accuracy gradually increases through the steps. It keeps going up at 20 steps, suggesting that further steps could improve the results.

Another interesting metric to look at is how many times the prediction changes over the 20 TTT steps. The results obtained are shown in Fig. 4. Notably, the predicted class never changes through the 20 steps for 62.3% of the images corrupted with elastic transform, 76.6% with gaussian noise and 61.6% with motion blur. Hence for a majority of images, the TTT steps do not impact the prediction. And when the prediction does change, it is mostly no more than a few times. Another qualitative observation we made in that sense is that most of the time, several steps are required to trigger a change in the prediction. Concrete examples and visualizations are provided in the Supplementary Material, in Figures 9 and 10.

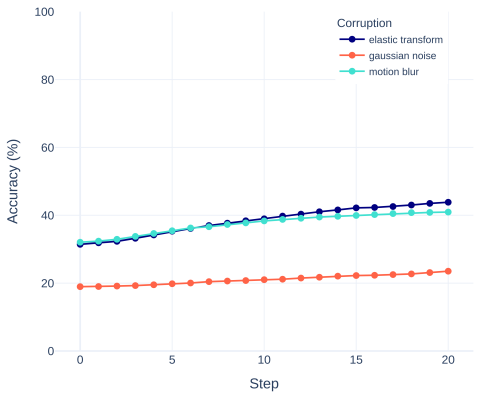

Figure 3. TTT-MAE accuracy for each number of steps

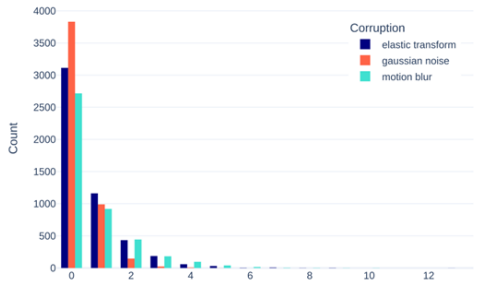

Figure 4. Number of prediction changes through the 20 steps

3.3. Failure cases

We saw in the previous section that TTT-MAE considerably improved classification results on ImageNet-C compared to the baseline model. Yet there are still many failure cases, which we also investigate. To understand the failure cases, we first have to mention some challenges inherent to the dataset. First, some classes in ImageNet are very similar, such as different dog breeds. Many images contain multiple objects, making it unclear which one actually represents the class of the image. In ImageNet-C, severe corruptions can make classification hard, even for humans. These challenges are responsible for many of the failure cases that we observed.

More specifically, we investigate cases in which the baseline outperforms TTT-MAE. These cases represent less than 2% of the data, as reported in Tab. 1. Hence, images on which both the baseline and TTT-MAE fail make up most failure cases. Examples of both types of failure cases are provided in the Supplementary Material, in Figures 11, 12 and 13.

Table 1. Failure cases comparison on baseline and TTT-MAE

Metric	Elastic	Gaussian	Motion
TTT-MAE correct and baseline wrong	13.42%	5.32%	10.66%
TTT-MAE wrong and baseline correct	1.02%	0.76%	1.78%

4. Online version

In the authors' method, the weights of the MAE are reset to the pre-trained version for every test image. The method is then offline, since it forgets the information learned from each test input. But in some cases, an online approach that would not reset the weights could be interesting. This is notably the case when the distribution shift does not affect a single test input, but the whole test set. Once again, a telling example is sensors, whose observations change over time as a result of gradual deterioration. Thus, we modified the authors' code to create an online version of TTT-MAE. To evaluate its performance, we ran experiments under two setups: one with a single corruption, and the other with two different corruptions.

4.1. Single corruption

For the first experiment, we used a single corruption at test time. For each of the three corruptions, we ran the online inference on all test images. Since only one corruption type is encountered at test time, we expect the model to improve as it processes more images.
To explore the impact of image order on performance, we tried with both the default class order (where the five images from each class are seen in a row) and a random order. We expected the default order to perform slightly better, as the model processes images by class. The results obtained are shown in Fig. 5, and the accuracies are reported in Tab. 3 in the Supplementary Material. As expected, the online model clearly outperforms the offline version. However, presenting the images in the default class order or randomly does not seem to have a substantial impact on performance.

4.2. Two corruptions

We also studied the performance of the online version in the presence of two different distribution shifts. Intuitively, when there are different corruptions, dynamically updating the model could potentially degrade performance. To explore this idea, we ran the following experiment. First, we chose two different corruptions: gaussian noise and elastic transform. We then split the images evenly and randomly in two sets, and applied gaussian noise on the first one and elastic transform on the other. To evaluate the impact of the order of images, we considered 3 settings: one where all images with elastic transform come first, one where gaussian noise comes first, and one with a random order. Note that within a corruption group, images are shuffled to get rid of any potential bias due to class order.

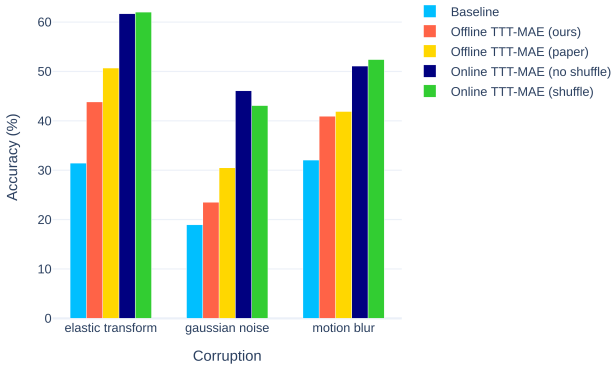

Figure 5. Accuracies of baseline, offline and online TTT-MAE

The results obtained are reported in Tab. 2. They do not show any major difference in accuracy. Yet, we noticed that the accuracy was a little higher for each corruption when it was processed first at test time. But the differences remain very small, and the accuracies are still far above the offline ones. We also visualized the moving average accuracy, shown in the Supplementary Material in Fig. 14.

Table 2. Results of the online training with two corruptions for different image orders.

Metric	Acc. Elastic	Acc. Gaussian	Average Acc.
Elastic first	**60.2%**	35.1%	47.6%
Gaussian first	60.0%	**35.7%**	47.8%
Random order	59.4%	35.4%	47.4%

Conclusion

In accordance with the authors' results, we can conclude that the TTT-MAE approach outperforms the baseline on ImageNet-C, at the cost of a longer inference time. Moreover, our online version produced even better results than the offline method, on both of our experiments.
For further exploration of the impact of the test distribution's structure, we suggest considering gradual corruption shifts. For instance, we could evaluate the performance of the model when test inputs are corrupted with an increasing gaussian noise.

References

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

Final Project - Test-Time Training with Masked Autoencoders (E)

Supplementary Material

5. Model Architecture and Training Setup

Figures 6, 7 and 8 provide schematic representations of the training steps presented in Sec. 2.2.

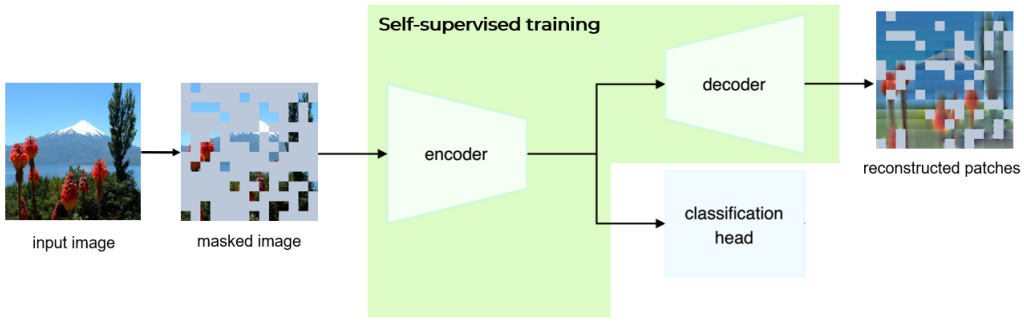

Figure 6. Pre-training of the encoder-decoder for masked image reconstruction on ImageNet-1k

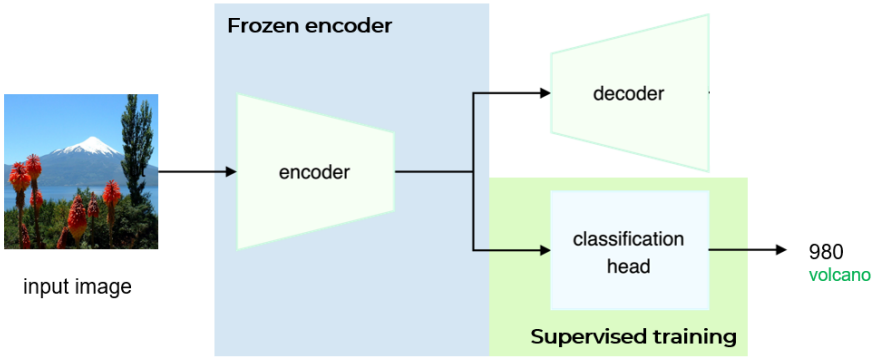

Figure 7. Training of the classification head with a frozen encoder (ViT probing)

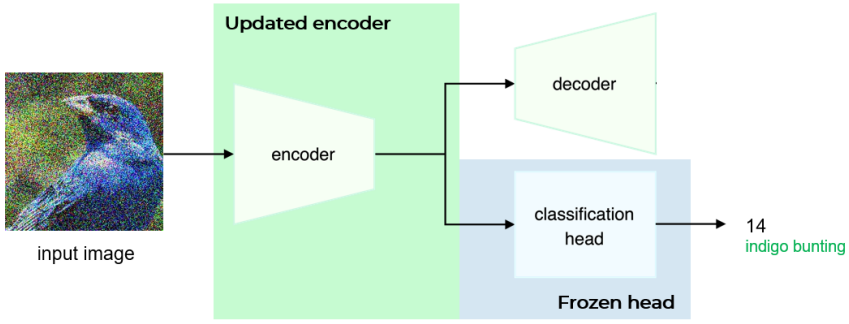

Figure 8. Test-time training : classification of the test image using the updated encoder and frozen classification head

6. Success Cases Examples

Figures 9 and 10 provide visual representations of the evolution of the predicted class through the TTT-MAE steps. They both correspond to images on which TTT-MAE outperforms the baseline. In the first case, the prediction becomes correct in the first few steps. For the second input, more steps are required to reach the true class.

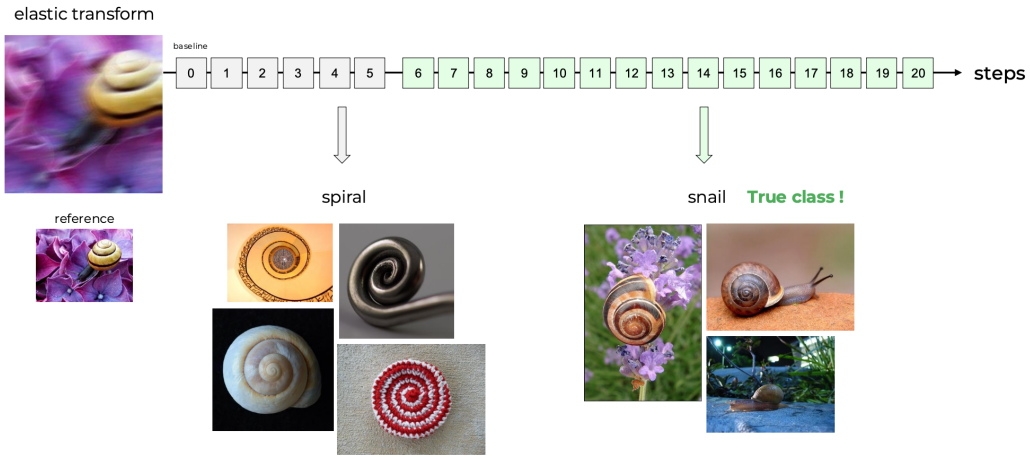

Figure 9. Example of success case

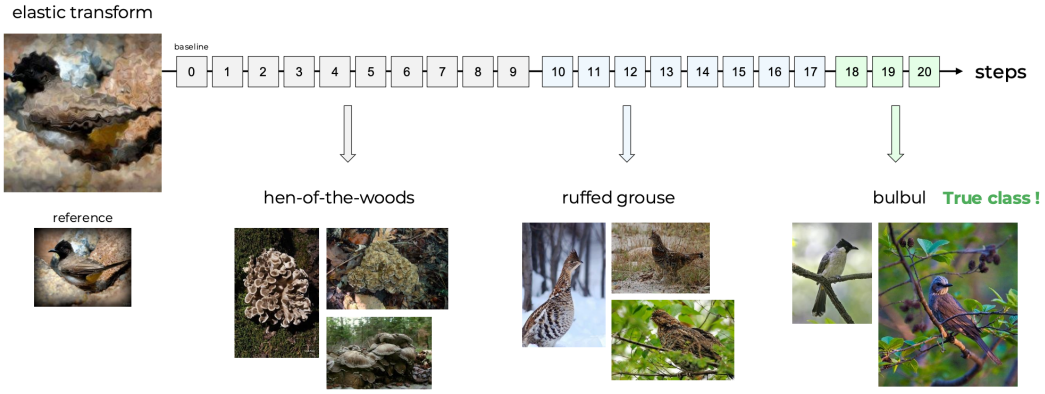

Figure 10. Example of success case

7. Failure Cases Examples

Figures 11, 12 and 13 show the evolution of the prediction on images on which TTT-MAE fails. In the first two, the baseline outperforms TTT-MAE. The case of Fig. 12 is less usual, with a prediction oscillating between two different classes. In Fig. 13, the prediction is always wrong. We guess that it might be because gaussian noise makes the background look like sand, but this in only hypothetical.

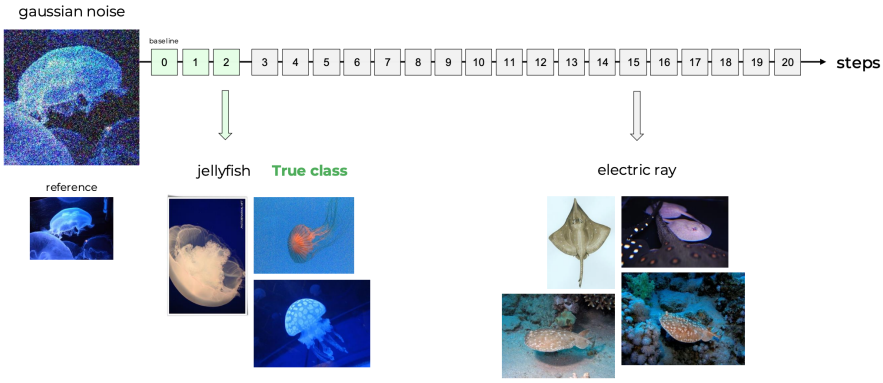

Figure 11. Example of failure case with close prediction

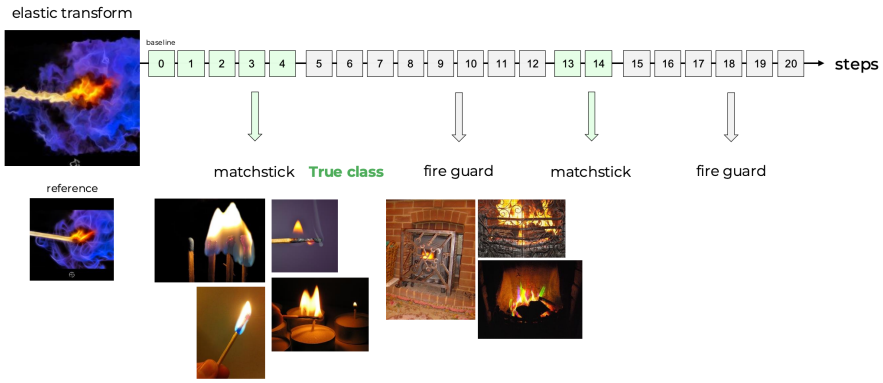

Figure 12. Example of failure case with oscillating prediction

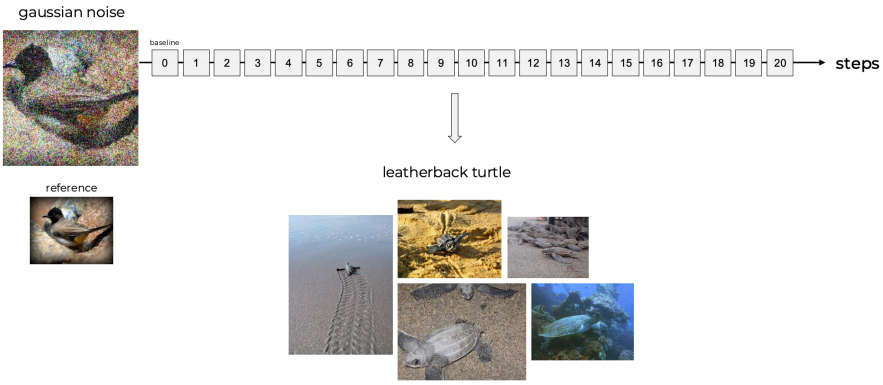

Figure 13. Example of failure case with constant prediction

8. Additional Online Results

Tab. 3 provides the numerical values of the accuracies used to build the histograms in Figures 2 and 5.

Table 3. Accuracies obtained with baseline, offline and online versions

Model	Elastic transform	Gaussian noise	Motion blur
Baseline	31.4%	19.0%	32.1%
TTT-MAE (ours)	43.8%	23.5%	40.9%
TTT-MAE (paper)	50.7%	30.5%	41.9%
TTT-MAE online (no shuffle)	61.6%	**46.1**%	51.1%
TTT-MAE online (shuffle)	**62.02**%	43.1%	**52.42%**

Fig. 14 shows the moving average accuracy with a window size of 300 during our experiment introduced in Sec. 4.2.

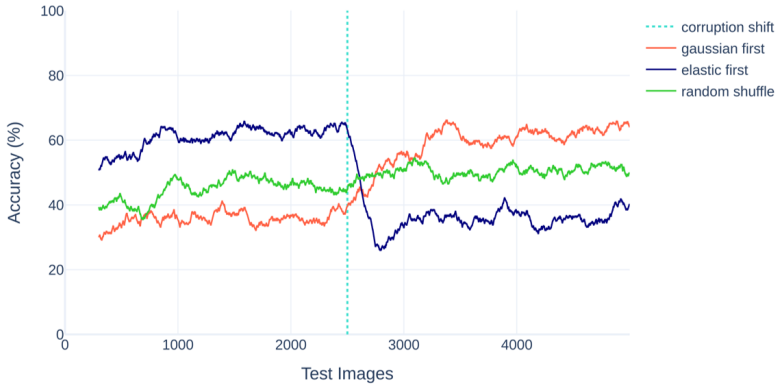

Figure 14. Moving average accuracy for the three orders with a window size 300