Reproduce Test Time Training (TTT++) algorithm on iWildCam dataset

Rui Huang*, Shanlan Li*, Xiaotian Su*
Supervised by Yuejiang Liu
{firstname.lastname}@epfl.ch
* denote equal contribution VITA Lab, EPFL, Switzerland

***Abstract*—The ability to handle distributional shift in the real-world data is one of the criteria of the robustness of the model. In this project, we aim to use an improved version of test-time training algorithm on an image dataset with 186 different classes containing distributional shift across camera traps.**

I. Introduction

While supervised learning models have achieved remarkable performance in a wide range of tasks simply by minimizing loss, a truly frustrating fact is that they fail to address the challenge of domain shift, even for state-of-the-art models. A minor distributional shift between training and testing data, e.g., image rotation and blur, may drastically decrease the prediction accuracy of models, undermining their utility when being deployed in the wild.

The reason why models suffer from this limitation, can be well-explained by a thought experiment discussed in [1]. Without any constraint, models are prone to recklessly fit training data such that they start to rely on domain-specific, spurious features, which reduces their generalization to new domains. The TTT++ [12] algorithm developed by the VITA Lab aims to tackle this issue. In this project, we aim to investigate the reproducibility of the newly proposed TTT++ algorithm by applying it into a new dataset with distributional shift.

In this report, we first briefly introduce TTT++ [12] as well as the datasets we use for evaluation. Then, we shed light on the details of our implementation. Finally, the results of experiments over TTT++ and other baseline models are compared and analyzed.

II. Background

Enabling models to generalize out-of-distribution (OOD) is deemed as a difficult yet crucial task. Motivated by the intuition mentioned above, a popular and heavily investigated approach is to learn from domain-invariant representations, e.g., CORAL [15] and domain adversarial training of neural networks [6]. One drawback of these methods is their demand for massive human annotations, since for invariant features to be fully captured, labeled data from various domains are required at training time. Besides, they also need to address the challenges of inflating sizes of datasets and increasing privacy concerns.

Recently, a novel learning paradigm, called test-time training (TTT) [16], has been proposed. It has the flexibility to adapt from one domain to another, without the need of domain-diverse training data. This new framework consists of a main learning task and a self-supervised learning task (SSL), both of which are trained jointly at training time. For test data with distributional shift, it can update the model with the help of SSL before prediction. Even though TTT has shown inspiring results, it might deteriorate empirical performance, due to unconstrained updates from SSL. TTT++ as an improved version of TTT has been proved outperforms state-of-the-art methods by a significant margin on multiple vision benchmarks.

In this section, we would explain the techniques used by TTT++.

A. Self-supervised task and contrastive learning

The idea of test-time training behind TTT can find its origin in many previous work. Several recent studies have adopted similar approaches to adapt their trained models to an unseen domain at test time, such as replacing batch normalization statistics of training data with those of shifted testing data[14]. Different from their counterparts, both of TTT and its variant - TTT++ turn to an auxiliary self-supervised task for online adaptation. A self-supervised branch allows them to learn rich data representations even at test time when labels are absent.

More specifically, TTT and TTT++ design their auxiliary branches based on contrastive learning paradigm. Contrastive learning is a machine learning technique used to learn the general features of a dataset without labels by teaching the model which data points are similar or different. It is quite suitable for mitigating the impact of distributional shift. TTT uses image rotation prediction task, while in comparison, TTT++ replaces it with SimCLR[4]. The feature of each image from an encoder is projected to a lower-dimensional space, where SSL task strives to discriminate negative samples, in contrast, and reduce the distance between positive ones measured by cosine similarity.

B. Feature alignment and online dynamic queue

A deeply investigated drawback of TTT is its lack of constraints on updates from the SSL task at test time, which may yield severe overfitting to the auxiliary task. Inspired by CORAL[15], the solution proposed by TTT++ is to align the statistics of features between training data and testing data during test-time training.

After training, TTT++ summarize the features of training data by calculating empirical mean and covariance matrix, and store feature summarization as part of the model. During test time, TTT++ dynamically calculates the moments over testing data and restrains its feature space to be as much closer as possible to that in the training domain. It is achieved by adding the difference between online moments and offline summarization into the loss function. Notably, TTT++ also adopts a dynamic feature queue from [7] to decouple batch size and sample size.

III. Dataset

They further validate the effectiveness of the method on the VisDA-C dataset [13], which is a classification dataset about visual Domain adaptation.

A. Datasets in Wilds

The Wilds benchmark contains 10 datasets across a diverse set of application areas, data modalities, and dataset sizes [9]. Each dataset comprises data from different domains, and it contains both in-distribution (ID) and out-of-distribution (OOD) evaluation sets.

Wilds bring up three different kinds of problem settings - domain generalization, subpopulation shift and a hybrid of the previous two. Out of 10 datasets, there are 5 sets about domain generalization, 1 under subpopulation shift, and the rest 4 are a mixture of these two types. From another perspective, the dataset in Wilds are related two separate ares in machine learning: computer vision and natural language processing.

B. The iWildCam2020-wilds dataset

The iWildCam2020-wilds dataset is a variant of the iWildCam 2020 dataset [2] which is a large collection of camera trap images.

Camera trapping [5] is a method for capturing wild animals on film when researchers are not present, and has been used in ecological research for decades. The cameras take thousands of images each day, and it takes a great amount of time for human experts to specify species. Hence, automatic species classification would help to increase the efficiency to a great deal.

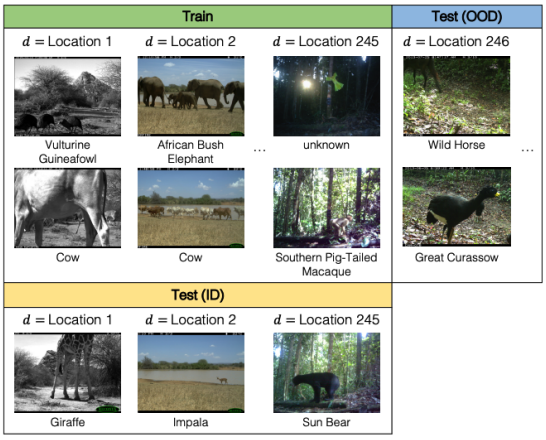

Fig. 1: Samples from iWildCam dataset

Usually, machine learning models are trained on photos from some existing camera traps and then they are used across new camera trap deployments. However, there exists radical disparity in camera angle, color, vegetation, and relative animal frequencies across different camera traps. And this would degrade the performance of models on new camera trap deployments [3].

This dataset is considered as a domain generalization problem by Wilds. The training and test sets comprise photos from disjoint sets of camera traps, that is $D^{train} \cap D^{test}$. For each camera, the dataset provides a series of remote sensing imagery that is tied to the location of the camera. It also has citizen science imagery from the set of species seen in the data. The challenge posed by the Wilds is to build models that generalize to photos from new camera traps that are not in the training set.

Fig 1 demonstrates some samples from the dataset where the input x is a photo from a camera trap, the label y is one of 186 animal species, and the domain d specifies the identity of the camera trap.

IV. Implementation

A. Experiment flow and model structure

To test TTT++ algorithm on iWildCam2020-wilds dataset, we import data using the data loading API provided by Wilds. Firstly, a standard "Resnet50" model imported from torchvision is applied as pre-training. Then, a joint model (Resent50 + Self-supervised learning head) is used to train as the "main task" in TTT++. There are two different ends being attached to the output of Resnet50 encoder in joint training. One is linear classification, which is trained with normal classification predictions. The other one is a three-layer neural network, which is trained by feeding transformed data and updated by contrastive learning loss. The transformation applied to simulate domain shifts are "random horizontal flip", "randome color jitter", "random grey scale". The contrastive learning algorithm applied in this experiment is "Supervised Contrastive Learning"[8] from google. After that, TTT++ algorithm is applied on the joint model with test data.

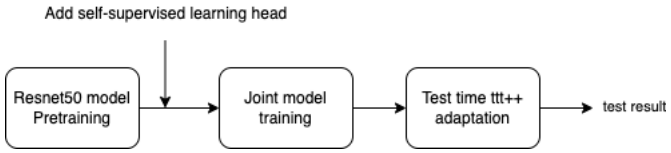

Fig. 2: Flowchart of reproducing ttt++ algorithm on iWildCam2020-wilds dataset

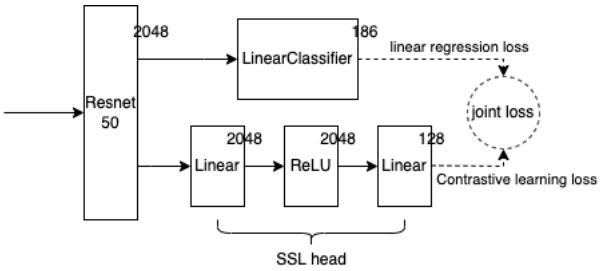

Fig. 3: Joint model structure (digits on the top right of each module indicate the output dimension)

B. Out of memory problem

One of the major issues we encountered is the OOM problem. The default dataset in the script of TTT++ is cifar [10] which contains image size of 32x32. But for the iWildCam dataset, it has images with size 224x224. So we need to do some adjustments before loading the data. We originally think we finished the work, since both resnet50 and the joint model are successfully trained over iWildCam dataset. However, when loading the pretrained model to TTT++ algorithm and performing test-time training, we get the OOM problem.

By using the Python debugger and printing out parameters and network layers, we find that the pretrained model and the model used in TTT++ have different structures. It turns out that the original implementation of resnet50 in TTT++ is specifically designed for cifar, which is not suitable for iWildCam. For example, in the first convolutional layer of pretrained model, kernel size $K = (7, 7)$, stride size $S = (2, 2)$. By having stride size 2, the resulting image becomes smaller. However, in TTT++, the corresponding kernel size $K = (3, 3)$, stride size $S = (1, 1)$ where the spatial input size did not decrease after the convolutional layer, leading to an explosion of feature maps and exhausting the CUDA memory.

To accommodate input data from iWildCam, we replace the implementation of resnet50 used in TTT++ with that in our pretrained model, which successfully solves the OOM problem. Besides, we rematch parameters between TTT++ and pretrained models for proper model loading.

V. Experiment

A. Baseline models

We test the dataset on TTT++ and compare the result with two other baseline models: TENT [17] and SHOT [11].

TENT[17] is short for test entropy minimization, it reduces generalization error for image classification on corrupted ImageNet and CIFAR-10/100.

SHOT[11] is the Source HypOthesis Transfer method, it freezes the classifier module and updates only the feature extraction module by exploiting the concepts of information maximization and self-supervised pseudo-labeling during testing.

According to the leaderboard on the wild website, the base line of a joint training model on the iWildCam dataset should have an accuracy around 70%. However, our model only gets an accuracy of 45%. Joint training on a single domain is also experimented as shown in the following table.

Location ID	95	101	120	187	288	all domains
Number of samples	1160	3020	3499	7600	8494	42,791
Joint accuracy	44%	19.4%	24.5%	98.4%	2.8%	45.5%

TABLE I: Joint training accuracy for different domains

Method	SHOT	TENT	TTT++
Error rate	74.69%	53.53%	57.81%

TABLE II: Error rate for different algorithms

VI. Results

Our results of pretrained resnet50 on iWildCam are shown in Fig 4 and Fig 5.

First, we pretrain Resnet50 as an image encoder on the pure classification task, which produces a 94.6% training accuracy and 45.5% validation accuracy on the whole training set from iWildCAM. In joint training, the training accuracy is decreased to 85%±3%. This is expected because in this step, random corruptions of the images are introduced to adapt the join model to be more resilient under various domain settings.

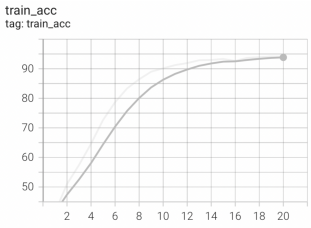

Fig. 4: Training accuracy on pretrained ResNet50

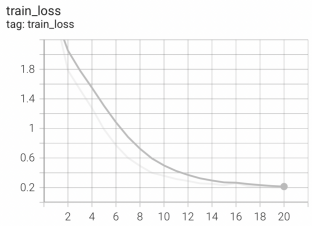

Fig. 5: Training loss on pretrained ResNet50

In the phase of testing time training, the result did not come as expected. We did not observe an improvement on testing accuracy. There may be several reasons for this result:

1. Pretrain model did not reach a sub-optimal stage.

One important assumption of the TTT++ algorithm is that it is built upon a model that is at least sub-optimal, so that the model can be fine-tuned to an optimal stage in test-time training. Test-time training algorithm lacks the theoretical support if the pre-existing model is not at a sub-optimal stage. From our current experiment result, there is a large variance between the pre-training results of data from different domains(2.8% - 98.4%). This implies that the pretrained model may be trained to fit some of the dominant domains only. The reason for that may be imbalanced or insufficient data per domain. Therefore, the pretrained model we obtained may not be suitable for testing on testing-time algorithm.

2. Unsuited self-supervised learning task or image transformation method.

It is observed that training accuracy is dropped during joint model training, and dropped further in test-time training phase. One possible reason could be that the self-supervised learning task we applied here do not simulate domain shifts in iWildCam dataset very well. Instead of simulating across domains, the added image transformation for contrastive learning simply add random noise to the test-time training. Further designs and tuning of the self-supervised learning task and image transformation method are ongoing.

VII. Conclusions and reflections

Our experiments of TTT++ on iWildCam yield surprising results, which fails to improve its accuracy in test domains, compared with those from CIFAR. One plausible explanation might be that the testing set of iWildCam has much more domain shifts and it leads to unpredictable learning directions. We plan to investigate its impact next by performing comparison experiments with respect to the number of test domains. The experiment is still ongoing, we will continue working on it and constantly update our GitHub repository.

Unaware of the difficulty of the task, we choose CivilComments-Wilds dataset which is a problem about NLP at first. After trials and errors, we realize that we could not figure out a way to make it work as deadline approaches. So we decided to revert to a CV dataset which is closer to the original problem. However, we still encounter a serious of unexpected problems since none of the team member has completed project in CV field.

Nevertheless, this is still a precious learning experience for all of us. Through which we not only gain a deeper understanding of CNN but also get in touch with the latest machine learning problem and become more familiar with the PyTorch library.

Lastly, we would like to express our gratitude to the professors and all the teaching assistants for the effort throughout the semester. And we really appreciate all the support and guidance from our supervisor Yuejiang Liu, without whom we could not have this attainment.

References

[1] Martin Arjovsky et al. "Invariant risk minimization". In: *arXiv preprint arXiv:1907.02893* (2019).

[2] Sara Beery, Elijah Cole, and Arvi Gjoka. "The iWildCam 2020 Competition Dataset". In: *CoRR* abs/2004.10340 (2020). arXiv: 2004.10340. URL: https://arxiv.org/abs/2004.10340.

[3] Sara Beery, Grant Van Horn, and Pietro Perona. "Recognition in terra incognita". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 456–473.

[4] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *CoRR* abs/2002.05709 (2020). arXiv: 2002.05709. URL: https://arxiv.org/abs/2002.05709.

[5] Wikipedia contributors. *Camera trap*. 2021. URL: https://en.wikipedia.org/wiki/Camera_trap.

[6] Yaroslav Ganin et al. "Domain-adversarial training of neural networks". In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.

[7] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *CoRR* abs/1911.05722 (2019). arXiv: 1911.05722. URL: http://arxiv.org/abs/1911.05722.

[8] Prannay Khosla et al. "Supervised contrastive learning". In: *arXiv preprint arXiv:2004.11362* (2020).

[9] Pang Wei Koh et al. "WILDS: A Benchmark of in-the-Wild Distribution Shifts". In: *CoRR* abs/2012.07421 (2020). arXiv: 2012.07421. URL: https://arxiv.org/abs/2012.07421.

[10] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[11] Jian Liang, Dapeng Hu, and Jiashi Feng. "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6028–6039.

[12] Yuejiang Liu et al. "TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive?" In: *Advances in Neural Information Processing Systems* 34 (2021).

[13] Xingchao Peng et al. "VisDA: The Visual Domain Adaptation Challenge". In: *CoRR* abs/1710.06924 (2017). arXiv: 1710.06924. URL: http://arxiv.org/abs/1710.06924.

[14] Steffen Schneider et al. "Improving robustness against common corruptions by covariate shift adaptation". In: *arXiv preprint arXiv:2006.16971* (2020).

[15] Baochen Sun and Kate Saenko. "Deep coral: Correlation alignment for deep domain adaptation". In: *European conference on computer vision*. Springer. 2016, pp. 443–450.

[16] Yu Sun et al. "Test-Time Training for Out-of-Distribution Generalization". In: *CoRR* abs/1909.13231 (2019). arXiv: 1909.13231. URL: http://arxiv.org/abs/1909.13231.

[17] Dequan Wang et al. "Tent: Fully test-time adaptation by entropy minimization". In: *arXiv preprint arXiv:2006.10726* (2020).