Computer vision methods adapting to new domains with few samples

by

David OSOWIECHI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, "15-09-2025"

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Christian Desrosiers, Thesis supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Ismail Ben Ayed, Thesis Co-Supervisor
Department of System Engineering, École de technologie supérieure

M. Marco Pedersoli, Chair, Board of Examiners
Department of System Engineering, École de technologie supérieure

M. Jose Dolz, Member of the Jury
Department of Software and IT Engineering, École de technologie supérieure

M. Christian Gagné, External Independent Examiner
Department of Electrical and IT Engineering, Université de Laval

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON 02-09-2025

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

À Maman, Papa, Yoni, parce que la distance et le temps n'ont aucune emprise sur ce que vous avez réussi à construire. Merci. Je vous aime pour l'éternité.

Thank you to Thomas — I haven't forgotten, and I never would have made it here without him.

To those without whom these past four years wouldn't have had the same flavor — thank you.

First and foremost, a special thank you to Christian, without whom my life would have taken a completely different path. He gave me the opportunity to come to Canada, to grow, to travel, and to flourish — all thanks to the freedom and trust he placed in me.
I'm also deeply grateful to him and Ismail for always being there, for listening, and for offering invaluable advice. Your guidance has truly meant the world to me.
Thank you to Arnaud and Steve — I truly appreciate your leadership: always with a smile, empathy, and kindness. Even in high-pressure situations, you created a reassuring atmosphere. Your calm, understanding presence during meetings always helped ease our stress.

To the entire lab team — Gus, Mehrdad, Ali, Moslem, Sonia, Milad, Sam — working alongside you over these four years has been a true joy. I'm proud to call you not only colleagues but friends. The pressure wouldn't have tasted the same without your support, laughter, and camaraderie. Thank you for accepting me as I am, with real open-mindedness, even with our differences.
Within this incredible team, a few people played an especially pivotal role in my journey:
A heartfelt thanks to Gus, whose unwavering support since day one has felt like that of an older brother — always offering wise advice and meaningful, genuine conversations.
And to Mehrdad — it was truly a pleasure working with him. I'm grateful for everything he shared with me — from teaching me Farsi to his generosity, energy, and all the unforgettable moments we created together.
I hope we will be able to make some more bangers together soon.

Finally, to all my lab friends — thank you. I'm deeply grateful, and incredibly proud, to now call people from all over the world my friends.

VI

Dans l'ordre de rencontre : Vitto, Ben, Cléclé, Hugz, Hec, Tom, Toinou, Jullos, Alex, Yas, Eva, Vlad, Aliette, Gud, Abel, PM, Manon, Jeanno, Yas, Mart, Margot, Gio, Ju, Mat, et tous les autres.

À ceux qui m'ont donné envie, à ceux qui m'ont poussé à le faire.
À ceux qui m'ont appris à dépasser mes peurs, à ceux qui m'ont rappelé comment vivre.
À ceux qui m'ont appris à rêver, à ceux qui m'ont fait les réaliser.
À ceux qui ont loué une partie de mon cœur, à ceux qui en sont devenus propriétaires.
À tous les gens de passage, et ceux qui restent.
À tous ceux qui m'ont influencé et permis de devenir.

Merci pour tout, je vous aime.

Je suis Docteur Dav.

Méthodes de vision par ordinateur s'adaptant à de nouveaux domaines avec peu d'échantillons

David OSOWIECHI

RÉSUMÉ

Les modèles d'apprentissage profond ont connu un succès remarquable dans un large éventail de tâches de vision par ordinateur, allant de la classification à la segmentation, et au-delà. Cependant, ces modèles sont souvent entraînés sous l'hypothèse que les données rencontrées lors de l'inférence proviendront de la même distribution que celles du jeu de données d'entraînement. Dans des scénarios réels, cette hypothèse est rarement vérifiée. Même de subtiles variations dans l'éclairage, les propriétés des caméras, les textures de fond ou l'apparence des objets peuvent entraîner des décalages importants dans la distribution des données, ce qui se traduit par une chute brusque des performances du modèle. Cette vulnérabilité aux changements de distribution soulève des préoccupations majeures quant à la robustesse et à la fiabilité des systèmes de vision déployés dans le monde réel.

Contrairement aux réseaux de neurones profonds, la perception humaine fait preuve d'une résilience naturelle face à de tels changements. Nous reconnaissons les personnes et les objets dans des conditions variées sans avoir besoin de réentraîner ou de superviser explicitement, guidés par des indices contextuels et des connaissances générales. Inspirée par cela, cette thèse déplace son attention du paradigme traditionnel centré sur l'entraînement vers une approche centrée sur l'inférence, où les modèles ne sont pas réentraînés ou ajustés hors ligne, mais s'adaptent dynamiquement et de manière non supervisée au moment de l'inférence. Ce cadre, couramment appelé Adaptation au Temps d'Inférence (*Test Time Adaptation*–TTA), présente des défis uniques, tels que l'absence de données étiquetées, l'inaccessibilité des données d'entraînement sources en raison de contraintes de confidentialité ou de stockage, et la nécessité d'une adaptation rapide et en temps réel.

Pour relever ces défis, cette thèse présente une série de méthodes modulaires et indépendantes de l'architecture pour adapter les modèles de vision pendant l'inférence, en mettant l'accent sur la robustesse, l'efficacité computationnelle et la large applicabilité.

En première contribution, nous introduisons NC-TTT : un cadriciel d'entraînement contrastif au temps d'évaluation, adapté aux réseaux de neurones convolutifs. Plutôt que de minimiser l'entropie ou de mettre à jour les statistiques de lot, notre méthode repose sur une tâche contrastive auxiliaire qui apprend à distinguer les augmentations bruitée des représentations de caractéristiques. Cela permet au modèle de renforcer sa compréhension des caractéristiques appartenant à la distribution tout en supprimant le bruit hors distribution, sans étiquettes ni accès aux données sources. NC-TTT démontre de solides performances sous divers types de changements à la distribution, notamment les entrées corrompues ainsi que les différences entre les domaines synthétiques et réels.

Dans la deuxième contribution, nous étendons l'adaptation au temps d'évaluation aux Modèles Vision-Langage (MLV), en particulier CLIP, qui a gagné en popularité grâce à ses capacités de faire des prédictions pour des nouvelles classes (*zero-shot*). Nous introduisons CLIPArTT, une méthode qui exploite la compositionnalité inhérente au langage pour adapter les invites textuelles (*text prompts*) au moment de l'évaluation. Plutôt que d'utiliser des invites fixes et préétablies, nous proposons une stratégie pour construire dynamiquement des invites en utilisant les propres prédictions top-K du modèle et les scores de similarité multi-modaux. Cette adaptation guidée par pseudo-étiquettes permet à CLIP de réaligner ses prédictions avec les distributions cibles, améliorant ainsi la précision de classification sur divers ensembles de données corrompus et décalés de domaine.

Notre troisième contribution principale explore le potentiel de l'adaptation multi-modèle dans les MVL à travers une méthode appelée WATT (*Weight Averaged Test-Time Adaptation*). Ici, nous adaptons CLIP à la distribution cible en utilisant plusieurs modèles textuels divers, chacun représentant une formulation linguistique différente des classes. Plutôt que de sélectionner un seul meilleur modèle, nous agrégeons les poids du modèle appris en utilisant une stratégie d'agrégation par moyenne des poids. Le résultat est un modèle plus stable et généralisable qui exploite la diversité des invites tout en évitant le sur-apprentissage de toute perspective linguistique unique.

Ensemble, ces contributions forment une approche cohérente et prospective pour construire des modèles de vision robustes, adaptatifs et prêts pour le déploiement. En minimisant la dépendance aux données étiquetées et à l'accès au domaine source, et en privilégiant la modularité et la flexibilité architecturale, cette thèse ouvre la voie à une nouvelle génération de systèmes intelligents qui apprennent non seulement pendant l'entraînement, mais évoluent également continuellement pendant l'inférence.

Mots-clés: Adaptation au Temps d'Inférence, Entrainement au Temps d'Inférence, Changement de Distribution, Modèles Vision-Langage

Computer vision methods adapting to new domains with few samples

David OSOWIECHI

ABSTRACT

Deep learning models have achieved remarkable success in a wide array of computer vision tasks, from classification to segmentation and beyond. However, these models are often trained under the assumption that the data encountered at test time will be drawn from the same distribution as the training set. In real-world scenarios, this assumption rarely holds. Even subtle variations in lighting, camera properties, background textures, or object appearances can lead to significant shifts in data distribution—resulting in a sharp drop in model performance. This vulnerability to distribution shifts raises critical concerns about the robustness and reliability of vision systems deployed in the wild.

In contrast to deep neural networks, human perception exhibits a natural resilience to such changes. We recognize people and objects under varying conditions without retraining or explicit supervision, guided by contextual cues and generalizable priors. Inspired by this, the focus of this thesis shifts from the traditional training-centered paradigm to an inference-centered approach, where models are not retrained or fine-tuned offline, but instead adapt dynamically and unsupervised at test time. This setting—commonly referred to as Test-Time Adaptation (TTA)—presents unique challenges, such as the absence of labeled data, the inaccessibility of source training data due to privacy or storage constraints, and the need for rapid, on-the-fly adaptation.

To address these challenges, this thesis presents a series of modular, architecture-agnostic methods for adapting vision models during inference, with a focus on robustness, computational efficiency, and broad applicability.

As a first contribution, we introduce NC-TTT: a noise contrastive test-time training framework tailored for convolutional neural networks. Instead of minimizing entropy or updating batch statistics, our method relies on an auxiliary contrastive task that learns to distinguish between noisy augmentations of feature representations. This allows the model to reinforce its understanding of in-distribution features while suppressing out-of-distribution noise—without any labels or access to source data. NC-TTT demonstrates strong performance under various types of distribution shift, including corrupted inputs and synthetic-to-real domain gaps.

In the second contribution, we extend test-time adaptation to Vision-Language Models (VLMs)—particularly CLIP, which has gained prominence for its zero-shot capabilities. We introduce CLIPArTT, a method that leverages the inherent compositionality of language to adapt textual prompts at test time. Rather than using fixed, handcrafted prompts, we propose a strategy for dynamically constructing prompts using the model's own top-K predictions and multi-modal similarity scores. This pseudo-label-guided adaptation enables CLIP to realign its predictions with target distributions—improving classification accuracy across various corrupted and domain-shifted datasets.

Our third core contribution explores the potential of multi-template adaptation in VLMs through a method called WATT (Weight Averaged Test-Time Adaptation). Here, we adapt CLIP to the target distribution using several diverse textual templates, each representing a different linguistic framing of the classes. Rather than selecting a single best template, we aggregate their learned model weights using a principled weight averaging strategy. The result is a more stable and generalizable model that leverages the diversity of prompts while avoiding overfitting to any single linguistic perspective.

Together, these contributions form a cohesive and forward-looking approach to building robust, adaptive, and deployment-ready vision models. By minimizing reliance on labeled data and source domain access, and by prioritizing modularity and architectural flexibility, this thesis paves the way for a new generation of intelligent systems that learn not only during training—but also evolve continually during inference.

Keywords: Test-Time Adaptation, Test-Time Training, Distribution Shift, Vision-Language Models

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

TTA	Test-Time Adaptation
TTT	Test-Time Training
DA	Domain Adaptation
DG	Domain Generalization
NCE	Noice-Constrative Estimation
OOD	Out-Of-Distribution
CNN	Convolutional Neural Network
VLM	Vision Language Model
ViT	Vision Transformer
BN	Batch Normalization
LN	Layer Normalisation
PTBN	Prediction Time Batch Normalization
TENT	Test-Time Adaptation by Entropy Minimization
SAR	Sharpness-Aware and Reliable Entropy Minimization
LAME	Laplacian Adjusted Maximum-likelihood Estimation
TPT	Test-time Prompt Tuning
TDA	Test-time Dynamic Adapter
CE	Cross-Entropy
MI	Mutual Information
Acc	Accuracy

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathcal{X}_s$	Input space of the source domain
$\mathcal{Y}_s$	Label space of the source domain
$\mathcal{X}_t$	Input space of the target domain
$\mathcal{Y}_t$	Label space of the target domain
$P(\mathcal{X}_s, \mathcal{Y}_s)$	Joint distribution of inputs and labels in the source domain
$P(\mathcal{X}_t, \mathcal{Y}_t)$	Joint distribution of inputs and labels in the target domain
$g(\cdot)$	Shared feature extractor (encoder)
$\pi_m(\cdot)$	Main task predictor
$\pi_{ss}(\cdot)$	Self-supervised task predictor
Θ	Complete set of model parameters
Θ_g	Parameters of the shared encoder
Θ_m	Parameters specialized for the main task
Θ_{ss}	Parameters specialized for the self-supervised task
$\mathcal{L}_{\text{train}}$	Training loss combining main and self-supervised objectives
$\mathcal{L}_{\text{m}}$	Main task loss
$\mathcal{L}_{\text{ss}}$	Self-Supervised loss
$\mathcal{L}_{\text{test}}$	Test-time adaptation loss (e.g., self-supervised loss or entropy loss)
$\mathcal{L}_{\text{ent}}$	Entropy loss used for unsupervised test-time adaptation
$\mathcal{L}_{\text{CL}}$	Contrastive learning loss

$\mathcal{L}_{\text{align}}$	Feature alignment loss between source and target domains
μ_s, μ_t	Mean feature vectors of source and target distributions
σ_s, σ_t	Standard deviation of feature vectors of source and target distributions
$p(\hat{y})$	Softmax probability output of the model prediction
γ, β	Scale and shift parameters of Batch Normalization layers
μ_t, σ_t^2	Batch mean and variance at test time for BN adaptation
τ	Temperature parameter in contrastive learning
z	Latent feature representation
$p_\phi(\cdot)$	Linear projection head in contrastive test-time training
$q_\phi(\cdot)$	Classifier head in contrastive test-time training
β	Noise ratio parameter in NC-TTT (σ_o/σ_s)
σ_s, σ_o	Standard deviations for in-distribution and out-of-distribution noise in NC-TTT
$\mathbf{S}_v$	Image-to-image similarity matrix
$\mathbf{S}_t$	Text-to-text similarity matrix
$\mathbf{Q}$	Soft pseudo-label matrix
$\hat{\mathbf{P}}$	Image-to-text prediction matrix
$\mathbf{Z}_v$	Image embeddings (features)
$\mathbf{Z}_t$	Text embeddings (features)

INTRODUCTION

Over the past decade, deep learning has revolutionized the field of computer vision, achieving unprecedented performance across a wide range of tasks such as image classification, semantic segmentation, object detection, and more. These advancements have largely been driven by the availability of large annotated datasets and the computational power to train complex neural networks. In controlled experimental settings, modern deep models often reach or even surpass human-level accuracy. However, these successes are typically obtained under the assumption that the training and test data are drawn from the same underlying distribution—an assumption that seldom holds in real-world scenarios.

In practice, data distributions are rarely stationary. Natural changes in lighting, background, camera quality, context, or even the time of acquisition can induce what is known as a **distribution shift** between training and deployment conditions. Alarmingly, even subtle shifts in the input distribution can lead to significant degradations in performance (Hendrycks & Dietterich, 2019). This sensitivity not only limits the practical applicability of deep learning models but also raises questions about their robustness, generalization capacity, and trustworthiness when faced with new, unseen environments (see Figure 0.1).

In contrast, human perception exhibits a remarkable ability to adapt to distribution shifts. Whether it's recognizing a friend under different lighting conditions or interpreting handwritten notes in a variety of styles, humans can reliably maintain performance across a broad spectrum of input variations without requiring retraining. This robustness stems from our capacity to form abstract, high-level representations of the world, which are invariant to many superficial changes in the input.

While current deep models do not yet match this level of flexibility, they often acquire useful and general knowledge during training—knowledge that should not be discarded even when the model struggles under distribution shifts. Instead of resetting or retraining models from scratch

whenever new data domains emerge, it is increasingly important to explore **adaptation strategies** that preserve the valuable inductive biases and learned representations already embedded in the model. Doing so not only saves computational resources but also acknowledges the growing consensus that models trained on rich data sources encode transferable features that remain relevant beyond the original training distribution.

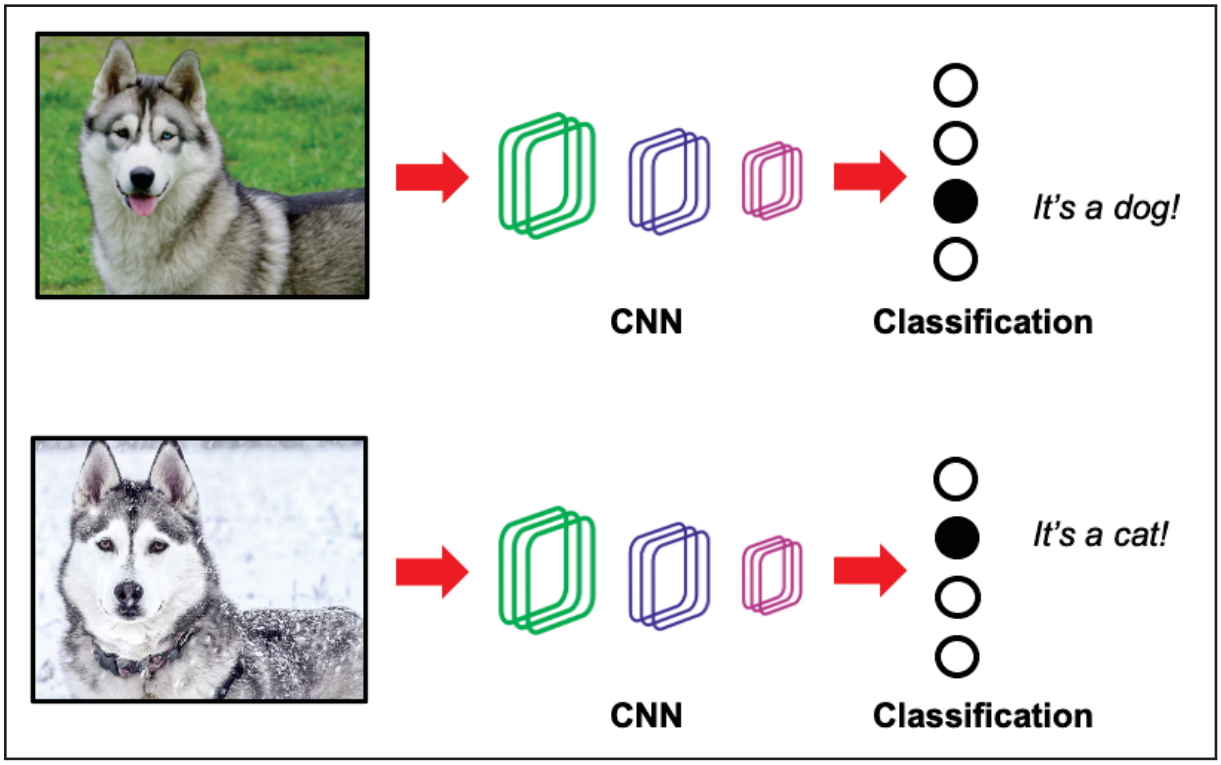


Figure 0.1 Problem with distribution shift. Initially, the dog is well classified by the model, then with only snow as distribution shift it classified it as a cat

In recent years, the emergence of **foundation models** has marked a significant paradigm shift in computer vision. Among these, **Vision-Language Models** (VLMs) have gained particular prominence, as they are trained on vast, often web-scale datasets using weak or noisy supervision, with the aim of learning general-purpose visual representations that are transferable across a wide array of downstream tasks. Rather than training a new model for each specific problem, foundation models are pre-trained once—typically with contrastive, multi-modal, or

self-supervised objectives—and then adapted or prompted for many different applications such as classification, retrieval, segmentation, captioning, or even zero-shot learning. A seminal example is CLIP (Contrastive Language–Image Pretraining) (Radford *et al.*, 2021), which demonstrated that aligning images and natural language descriptions in a shared embedding space enables powerful zero-shot classification without task-specific fine-tuning. Building on this idea, SigLIP (Zhai, Mustafa, Kolesnikov & Beyer, 2023) and its successor SigLIP-2 (Zhai, Mustafa, Kolesnikov & Beyer, 2025) introduced architectural and training improvements that significantly enhance visual-textual alignment and robustness. Other notable models such as ALIGN (Jia *et al.*, 2021), and BLIP (Li, Li, Xiong & Hoi, 2022) further illustrate the potential of this approach. These foundation models demonstrate a surprising degree of robustness to distribution shifts and unseen categories, in part because they are exposed to a broader diversity of data during training. However, while they perform impressively in zero-shot settings, their performance can still degrade under significant domain changes, and adapting them to specific downstream distributions without losing their generalization abilities remains an open challenge.

To address the challenge of distribution shifts, the research community has proposed a wide range of strategies aimed at increasing the robustness and generalization capacity of deep learning models. Among these, two prominent paradigms have emerged: **Domain Generalization** (DG) and **Domain Adaptation** (DA). Domain Generalization (Volpi *et al.*, 2018; Prakash *et al.*, 2019; Zhou, Yang, Qiao & Xiang, 2020; Kim, Wang, Sclaroff & Saenko, 2022; Wang *et al.*, 2022a; Noori *et al.*, 2024; Cheraghalikhani *et al.*, 2024; Noori *et al.*, 2025) seeks to train models that can perform well on unseen domains by leveraging data from multiple, diverse source domains. The underlying intuition is that exposure to a variety of training distributions enables the model to learn invariant features that are less sensitive to domain-specific artifacts. However, the effectiveness of DG methods hinges on the availability of sufficiently rich and diverse domain-labeled datasets, which may be expensive or impractical to collect in many

real-world applications. Moreover, even with broad coverage during training, there is no formal guarantee that the model will generalize to entirely novel test-time conditions.

In contrast, Domain Adaptation methods (Chidlovskii, Clinchant & Csurka, 2016; Liang, Hu & Feng, 2020; Wilson & Cook, 2020) aim to adapt a model trained on one source domain to perform well on a different, often unlabeled, target domain. This framework is particularly appealing in scenarios where only limited information about the target domain is available at test time. Adaptation can be performed either in a supervised, unsupervised, or semi-supervised manner, depending on whether the target domain provides labeled, unlabeled, or partially labeled data. Unlike DG, DA focuses directly on minimizing the performance gap caused by distribution mismatch between the source and target domains, often using alignment techniques or representation adaptation. However, many DA approaches assume access to the original labeled source data during adaptation, which may not always be feasible due to storage constraints, privacy concerns, or licensing restrictions. This limitation has led to increased interest in source-free and test-time adaptation settings, where models must adapt to new domains without revisiting the source training data.

More recently, the field of **Test-Time Adaptation** (TTA) has emerged as a promising direction for enhancing the robustness of deep networks to distribution shifts under even more constrained and realistic conditions. In contrast to Domain Generalization, which typically relies on data from multiple source domains during training, TTA assumes that the model is trained on a single source domain. Moreover, unlike conventional Domain Adaptation, which often requires access to the source data and sometimes even target supervision, TTA operates under a stricter setting: the model must adapt at test time, without access to the original training data, and typically without any labeled examples from the target domain.

This setup poses significant challenges. Because source data is unavailable, the model cannot explicitly compare source and target distributions to guide adaptation. Instead, it must rely

entirely on the structure of the target data it encounters at test time—often just a small batch of unlabeled samples—to refine its predictions or internal representations. Despite these constraints, test-time adaptation remains an appealing research direction, as it aligns closely with real-world deployment scenarios where labeled target data is scarce or unavailable, and data privacy or memory limitations prevent the reuse of training data. A successful TTA method would enable models to continually adjust to changing environments—such as evolving lighting conditions, sensor noise, or even new data modalities—while retaining the general-purpose capabilities acquired during pretraining.

Research Objectives and Contributions

This thesis addresses the fundamental challenge of **adapting computer vision models to new domains under distribution shift with minimal supervision and few target samples**. While deep learning models have demonstrated remarkable success in standard settings, their performance often deteriorates significantly when deployed in real-world scenarios where the data distribution differs from that of the training set. To mitigate this, we explore methods that enable models to dynamically adapt at test time, preserving the knowledge acquired during training while improving robustness to previously unseen conditions.

To achieve this, we focus on **unsupervised** adaptation objectives, which guide the adaptation process without reliance on labeled target samples or explicit domain alignment with the source distribution. A key challenge in this setting is to ensure adaptation mechanisms remain effective across diverse test-time scenarios, where the nature and extent of distribution shift are unknown. Thus, our approach prioritizes **robustness**—seeking to maintain or even improve model performance consistently across a wide range of deployment conditions. Moreover, a crucial requirement for practical applicability is ensuring **compatibility** with off-the-shelf, pre-trained models, without the need for retraining, architectural modifications, or access to the

original training data. This constraint aligns with real-world deployment scenarios, where storage limitations, data privacy concerns, and computational efficiency are major considerations.

Ultimately, this work aims to advance test-time adaptation techniques that enhance the adaptability of deep learning models, enabling them to operate reliably across diverse and evolving environments.

This is a thesis by articles. The core contributions are:

- In Chapter 2, we address the standard problem of Test-Time Training (TTT)—a specific for of TTA— for image classification within the framework of convolutional neural networks. Specifically, we propose a novel unsupervised loss function inspired by noise contrastive estimation, designed to enhance model adaptation during inference.

  Related publication:

  NC-TTT: A Noise Contrastive Approach for Test-Time Training, published at the IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR) 2024, Seattle. (Osowiechi *et al.*, 2024a).
- In Chapter 3, we extend the problem of test-time adaptation to vision-language models. More precisely, we introduce an approach for automatic text prompt construction during inference, enabling their use as text supervision.

  Related publication: CLIPArTT: Adaptation of CLIP to New Domains at Test-Time, published at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2025, Tucson. (Vargas Hakim *et al.*, 2024)
- Finally, in Chapter 4, we explore an alternative approach to test-time adaptation for vision-language models. In this framework, we leverage a diverse set of templates for text prompts, extending the existing capabilities of CLIP. Model predictions serve as pseudo-labels for adaptation, followed by weight averaging to consolidate the learned information globally.

  Related publication: WATT: Weight Average Test-Time Adaptation of CLIP, published at

The Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS) 2024, Vancouver. (Osowiechi *et al.*, 2024b)

To support further research and enhance the reproducibility of results, the code associated with the papers discussed above is publicly available.

Additional contributions

Additional contributions to test-time adaptation and test-time training were made throughout this thesis, which are outlined below:

- We present the first paper that employs an unsupervised auxiliary task for test-time training, using Normalizing Flows to model the normal distribution of latent features and detect domain shifts in test examples.

  Related publication: TTTFlow: Unsupervised Test-Time Training with Normalizing Flow, published at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2023, Hawaii.(Osowiechi *et al.*, 2023)
- We introduce a novel unsupervised TTT technique based on the maximization of Mutual Information between multi-scale feature maps and a discrete latent representation.

  Related publication: ClusT3: Information Invariant Test-Time Training, published at the IEEE/CVF International Conference on Computer Vision (ICCV) 2023, Paris. (Vargas Hakim *et al.*, 2023)
- We extend test-time adaptation to vision-language models within the new paradigm of Open-Vocabulary Semantic Segmentation, where we employ a Multi-Template, Multi-Level entropy minimization approach based on all tokens.

  Related publication: Test-Time Adaptation of Vision-Language Models for Open-Vocabulary Semantic Segmentation, *submitted to The International Conference on Machine Learning (ICML) 2025*

- Finally, we focus our work on histopathology vision-language models, where we introduce a framework for corrupted domain shifts and propose a method for adapting to these shifts at test-time.

  Related publication: Histopath-C: Towards Realistic Domain Shifts for Histopathology Vision-Language Adaptation, *submitted to The International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) 2025*

This thesis was conducted in collaboration with Zebra Technologies / Matrox Imaging.

CHAPTER 1

LITERATURE REVIEW

In this thesis, we consider the general problem of test-time adaptation, where a model trained on a source domain must generalize to a target domain whose data distribution differs from that of the source, without access to source data, ground-truth labels, or large batches of target samples, which is summarized in Figure 1.1. Let $P(\mathcal{X}_s, \mathcal{Y}_s)$ denote the joint distribution over inputs $\mathcal{X}_s$ and labels $\mathcal{Y}_s$ in the source domain, and $P(\mathcal{X}_t, \mathcal{Y}_t)$ the corresponding distribution in the target domain. Specifically, we focus on the setting of likelihood shift, where the conditional input distributions given the labels differ between domains, i.e., $P(\mathcal{X}_s|\mathcal{Y}_s) \neq P(\mathcal{X}_t|\mathcal{Y}_t)$, while the label space is shared across domains ($\mathcal{Y}_s = \mathcal{Y}_t$). This formulation captures a broad class of real-world scenarios and provides the foundational context for the methods and assumptions explored in the following literature.

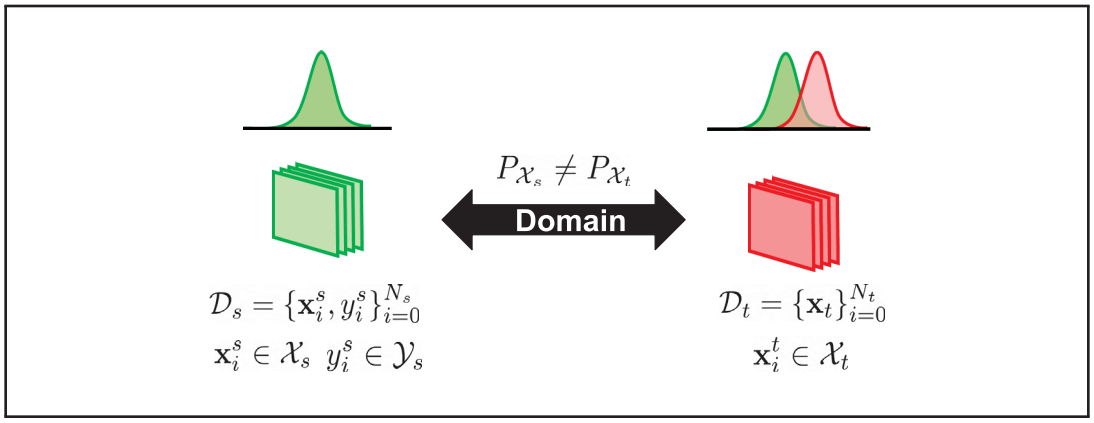


Figure 1.1 Diagram of the general problem of TTA

1.1 Batch Normalization Adaptation

Batch Normalization (BN) has become a ubiquitous component in modern deep networks, widely adopted for its ability to accelerate training and stabilize convergence. By normalizing the activations within a mini-batch to have zero mean and unit variance, BN reduces internal covariate shift and allows for the use of higher learning rates. However, this reliance on batch-level statistics becomes problematic at test time under domain shift, as the original batch

statistics collected during training may no longer reflect the distribution of the target data. This mismatch can degrade model performance, particularly when deploying models in environments that differ significantly from the training conditions.

To address this issue, several works have proposed adapting the batch normalization statistics at test time to better match the target distribution. A notable early contribution is Prediction-Time Batch Normalization (PTBN) (Nado *et al.*, 2021), which proposes to recompute the batch mean and variance using only the current batch of test samples:

$$\mu_t = \frac{1}{B}\sum_{i=1}^{B} f(x_i), \;\; \sigma_t^2 = \frac{1}{B}\sum_{i=1}^{B}(f(x_i) - \mu_t)^2 \tag{1.1}$$

These recalculated statistics are then used in place of the stored running estimates from training, as follows:

$$\mathrm{BN}(f(x_i)) = \gamma \cdot \frac{f(x_i) - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}} + \beta \tag{1.2}$$

The intuition is that by realigning the normalization process to the test distribution, the model's internal representations can be better calibrated, thereby improving prediction accuracy in shifted domains.

However, PTBN faces limitations when the test batch is small, a common scenario in online or streaming applications. The estimated mean and variance may become highly unreliable due to the high variance inherent in small sample statistics. To mitigate this, subsequent approaches have proposed hybrid strategies that blend training and test-time statistics. For example, Schneider *et al.* (2020) propose a weighted average between the batch statistics computed during training and those observed at test time. This approach serves as a trade-off between the stability of the learned distribution and the adaptability to the target domain.

Building on this line of work, SITA (Single Instance Test-time Adaptation) (Khurana, Paul, Rai, Biswas & Aggarwal, 2021) extends the idea to the extreme case where only a single test instance is available. To estimate reliable statistics in this setting, SITA generates a pseudo-batch

by applying a series of stochastic augmentations to the test image. These augmented views simulate the statistical diversity of a true mini-batch, enabling the computation of meaningful batch normalization statistics. This approach allows for per-sample adaptation without requiring access to additional data, making it particularly appealing in low-latency or privacy-sensitive applications.

Overall, the adaptation of normalization layers—whether through batch-level recalibration as in PTBN or augmentation-based estimation as in SITA—has proven to be an effective and lightweight strategy for mitigating performance degradation due to domain shift, especially in scenarios where model parameters must remain fixed.

1.2 Test-Time Training

Methods based on **Test-Time Training (TTT)** update the model during inference in order to mitigate domain shift. The key idea is to jointly train the network on two complementary tasks: the main task, denoted as π_m, and a self-supervised task, denoted as π_{ss}. In this framework, a shared encoder g is used to extract features that serve both tasks. During training, the model is optimized jointly on both the primary and self-supervised objectives. At test time, while the main task branch remains frozen, the shared encoder is updated using the self-supervised loss, thereby adapting the model to the new data distribution.

Figure 1.2 illustrates a common architecture for TTT methods. The architecture follows a Y-structure: the bottom part is the shared encoder g, and the two branches represent the main task and the self-supervised task. More formally, let $\Theta = (\theta_1, \theta_2, \ldots, \theta_K)$ denote the full set of model parameters distributed across K layers. The shared encoder corresponds to $\Theta_g = (\theta_1, \ldots, \theta_\kappa)$ for some $\kappa \in \{1, \ldots, K\}$, while the remaining parameters are specialized for the main task and the self-supervised task, respectively. Specifically, the parameters for the main task are denoted as $\Theta_m = (\theta_\kappa, \theta_{\kappa+1}, \ldots, \theta_K)$ and those for the self-supervised task as $\Theta_{ss} = (\theta'_\kappa, \theta'_{\kappa+1}, \ldots, \theta'_K)$.

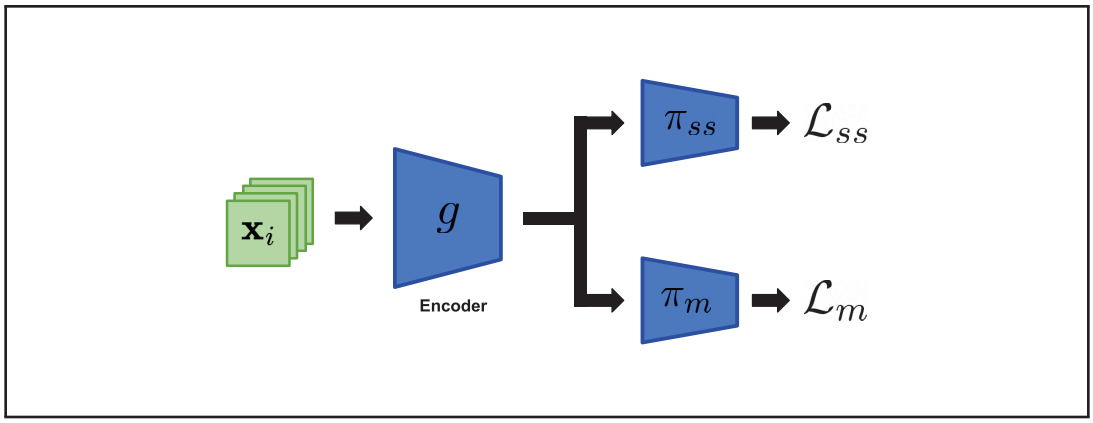


Figure 1.2 Common architecture for Test-Time Adaptation through Self-Supervised Learning

Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, the training objective is to minimize a composite loss that accounts for both tasks:

$$\mathcal{L}_{\text{train}} = \frac{1}{N} \sum_{i=1}^{N} \Big[\mathcal{L}_m(\pi_m(g(x_i)), y_i) + \mathcal{L}_{ss}(\pi_{ss}(g(x_i))) \Big], \tag{1.3}$$

where $\mathcal{L}_m$ is the loss function associated with the main task, and $\mathcal{L}_{ss}$ is the self-supervised loss. Common choices for the self-supervised task include rotation prediction as made in TTT (Sun *et al.*, 2020), where images are rotated by one of several fixed angles (e.g., 0°, 90°, 180°, and 270°) and the model must predict the rotation angle.

After joint training, the model undergoes test-time adaptation by freezing Θ_m and updating only the parameters of the shared encoder Θ_g using the self-supervised loss $\mathcal{L}_{ss}$ computed on incoming target samples:

$$\mathcal{L}_{\text{test}} = \mathcal{L}_{ss}(\pi_{ss}(g(x_i))) \tag{1.4}$$

The implicit assumption is that adapting the encoder to the target domain without altering the classifier (or main task head) will yield better generalization, as the self-supervised loss drives the encoder to extract features that are invariant to the distributional changes.

Building on this basic framework, several extensions to TTT have been proposed. A key limitation of the original TTT framework is its reliance on a fixed self-supervised task—such as rotation prediction—that may not be sufficiently informative or aligned with the downstream task or the characteristics of the target domain. To address this issue, **TTT++** (Liu *et al.*, 2021) proposes two major improvements: (1) replacing the rotation prediction loss with a more expressive contrastive learning objective, and (2) introducing a novel feature alignment loss that explicitly bridges the gap between source and target distributions during inference.

In TTT++, the self-supervised task is reformulated using a contrastive loss inspired by SimCLR (Chen, Kornblith, Norouzi & Hinton, 2020):

$$\mathcal{L}_{\text{CL}} = -\log \frac{\exp(\text{sim}(\pi_{ss}(g(x_i), \pi_{ss}(g(x_i^+))/\tau)}{\sum_{j=1}^{2N} \mathbb{1}_{[j \neq i]} \exp(\text{sim}(\pi_{ss}(g(x_i), \pi_{ss}(g(x_j))/\tau)} \tag{1.5}$$

Given two augmentations of the same image, the model is encouraged to map them to nearby representations in the feature space, while pushing representations of different images apart. This improves the quality and generalizability of learned representations, especially when transferring across domains.

Moreover, TTT++ incorporates a source-target alignment loss computed at test time. Specifically, it compares the mean μ and standard deviation σ of the features extracted from the current test batch to those of the source domain, which are precomputed and stored during training:

$$\mathcal{L}_{\text{align}} = \|\mu_s - \mu_t\|_2^2 + \|\sigma_s - \sigma_t\|_2^2 \tag{1.6}$$

The goal is to ensure that the features of the target samples stay close to the distribution seen during training, thereby preserving compatibility with the frozen main task head. This regularization term effectively guides the encoder to stay within the manifold of source-like representations, preventing excessive drift that could degrade the classifier's performance.

Together, the contrastive self-supervised loss and the online feature alignment regularizer allow TTT++ to achieve more stable and effective adaptation, particularly in scenarios where the

domain shift is more pronounced or where the self-supervised signal alone may not suffice. As a result, TTT++ significantly improves over the original TTT approach across several standard benchmarks, highlighting the importance of both loss design and alignment in test-time training frameworks.

Another notable development is the introduction of meta-learning strategies at test time, as seen in MT3 (Bartler, Bühler, Wiewel, Döbler & Yang, 2021), where meta-training is performed on the self-supervised task to improve adaptation efficiency. Additionally, TTT-MAE (Gandelsman, Sun, Chen & Efros, 2022) integrates Masked Autoencoders (MAE) (He *et al.*, 2022) as the second branch for test-time training, further extending the approach to work effectively with Vision Transformers (Dosovitskiy *et al.*, 2020).

Collectively, these methods demonstrate the versatility of the TTT paradigm in leveraging self-supervised signals for model adaptation during inference. They provide a robust framework for addressing domain shifts, wherein the online adaptation of the shared encoder—with the main task branch kept fixed—ensures that the model remains well-calibrated to the target data, leading to improved performance in dynamic and real-world environments.

1.3 Test-Time Adaptation on CNNs

Test-Time Adaptation (TTA) has recently emerged as a compelling framework for addressing distribution shifts during inference, without requiring access to source data or any form of target supervision. The core objective of TTA is to adapt a pre-trained model to the target distribution *on the fly*, i.e., as batches of data arrive at test time. This setup reflects realistic deployment conditions, but also introduces several challenges: (1) the inaccessibility of source samples, which rules out direct domain alignment strategies commonly used in traditional Domain Adaptation; (2) the lack of ground-truth labels, which forces methods to rely on unsupervised or self-supervised signals; and (3) the incomplete visibility of the target distribution, as adaptation occurs sequentially over mini-batches, often without revisiting past data.

A seminal contribution to the field of TTA is the **TENT** method (Test-Time Entropy Minimization) (Wang, Shelhamer, Liu, Olshausen & Darrell, 2020), which introduced a simple yet effective strategy for adapting models during inference without access to source data (see Figure 1.3). TENT assumes that a model has been fully trained on a source domain and that only the trained model and incoming, unlabeled target samples are available at test time. The key idea is to adapt the model by minimizing the entropy of its predictions on the target data, encouraging the model to make confident predictions even in the presence of distribution shift.

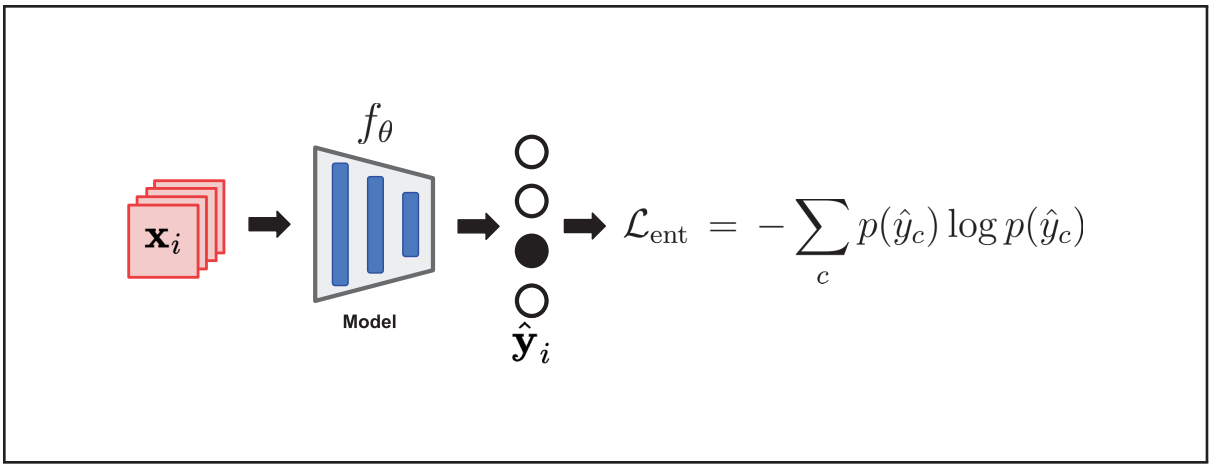


Figure 1.3 Architecture of TENT on CNNs. Only the Batch Normalization Layers are updated based on entropy minimization

Formally, for a given input batch x from the target domain and the whole model f, TENT computes the model's softmax output $p(\hat{y})$ where $\hat{y} = f(x)$ is the model prediction and minimizes the Shannon entropy:

$$\mathcal{L}_{\text{ent}} = -\sum_c p(\hat{y}_c) \log p(\hat{y}_c) \tag{1.7}$$

This objective is optimized with respect to a limited subset of the model's parameters: only the affine parameters of batch normalization layers (scale and shift) are updated during test time, while all other weights are kept frozen. The normalization statistics (mean and standard deviation) are also updated based on the target batch. This design choice ensures that the core

knowledge learned on the source domain is preserved, and adaptation is both lightweight and stable. Batch normalization layers are particularly well-suited for this kind of adaptation, as they control the feature distributions internally and have been shown to be sensitive to domain shift.

TENT operates in an online or episodic manner, adapting the model incrementally as target batches are observed. Importantly, it does not require access to the entire target dataset at once, making it suitable for streaming or real-time scenarios. It also requires no auxiliary tasks or pretext objectives—only entropy minimization over the target predictions.

Despite its simplicity, TENT has been shown to be highly effective across a range of benchmarks, including corrupted image datasets and synthetic-to-real domain shifts. However, the method does have certain limitations. First, it assumes that the target domain samples are drawn from a single, coherent distribution; if the test-time data is non-stationary or shifts significantly across time, performance can degrade. Second, because the method relies exclusively on entropy minimization, it can sometimes lead to degenerate solutions (e.g., overly confident but incorrect predictions), especially when the model is severely miscalibrated on the target domain.

Nevertheless, the simplicity and effectiveness of TENT have inspired a broad range of follow-up research aiming to improve or extend its underlying principles. A notable refinement was proposed by subsequent work (Mummadi *et al.*, 2021), which replaces the standard entropy loss with a log-likelihood ratio objective. This alternative formulation allows for a more calibrated optimization, better distinguishing between high- and low-confidence predictions. Additionally, this variant introduces improved estimation of the batch normalization statistics by explicitly modeling the distribution of the target batch, leading to more stable updates under distribution shift.

Other approaches build upon TENT by introducing mechanisms to selectively adapt only on target samples deemed reliable. For instance, EATA (Niu *et al.*, 2022) enhances test-time entropy minimization by integrating a confidence-based filtering mechanism that discards samples with low prediction reliability. By avoiding adaptation on noisy or ambiguous samples, EATA

mitigates the risk of error amplification, a common failure mode in unsupervised test-time adaptation.

In a different direction, MEMO (Zhang, Levine & Finn, 2022) explores the use of test-time data augmentation to improve robustness and estimation. For each test instance, MEMO generates a set of augmented views and aggregates their predictions to approximate a more accurate marginal distribution. This ensemble of predictions is then used to compute an entropy-based objective, which guides the adaptation of the model parameters. This augmentation-based marginal estimation enables the model to generalize better from individual target samples, even under severe domain shift.

The Sharpness-Aware and Reliable Entropy Minimization (**SAR**) method, proposed by Niu *et al.* (2023), addresses key challenges in TTA by improving the robustness of models to noisy target samples and guiding the model toward stable solutions during adaptation (see Figure 1.4). SAR is designed to operate under the constraints of test-time adaptation, where the model is applied to a stream of unlabeled target samples without access to the source data.

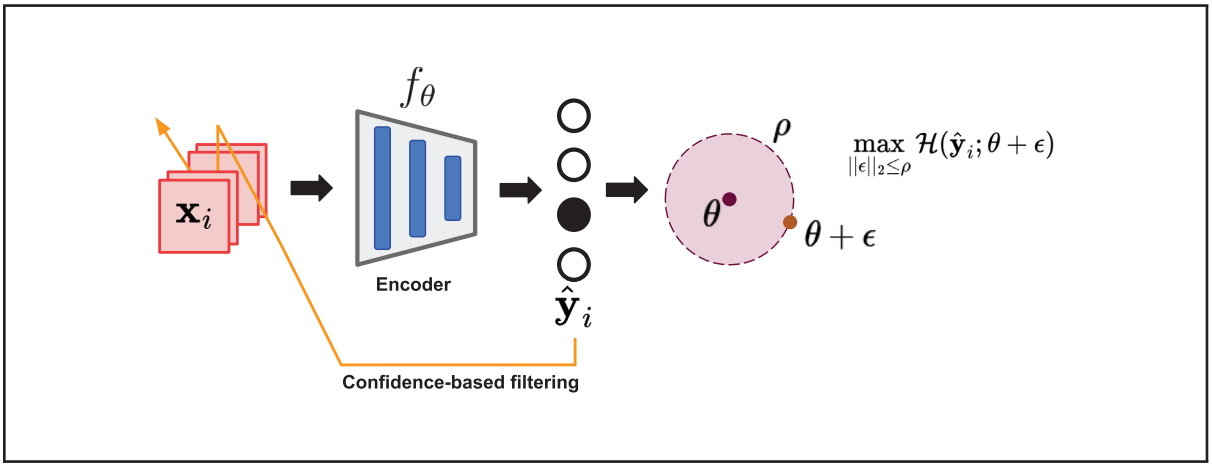


Figure 1.4 Architecture of SAR on CNNs. Only the Batch Normalization Layers are updated based on Sharpness-Aware and Reliable Entropy Minimization

A central concept in SAR is the idea of sharpness-aware regularization, which aims to make the model more robust by encouraging it to converge to a flat minimum in the loss landscape. The sharpness of a minimum refers to how sensitive the model's loss function is to small perturbations

in the parameters. A flatter minimum corresponds to a more stable and less sensitive solution, which is desirable when the model must generalize to unseen, noisy test-time data. In SAR, this is achieved by incorporating a sharpness-aware penalty that minimizes the loss function's sensitivity to small changes in the model's parameters. By encouraging the model to settle in a flat minimum, SAR improves generalization to out-of-distribution data, making the adaptation process more stable even in the presence of substantial domain shifts.

In addition to sharpness-aware regularization, SAR introduces a mechanism for removing noisy samples from the adaptation process. Noisy target samples—often characterized by large gradients—can lead to instability and degrade the performance of test-time adaptation. To mitigate this, SAR identifies and filters out samples that exhibit large gradients, which are typically indicative of outliers or noisy data points. By excluding these problematic samples, the method ensures that adaptation is performed on more reliable target data, preventing the model from overfitting to noisy or irrelevant information. This approach enhances the stability of the adaptation process and reduces the risk of the model being swayed by erroneous data during test-time adaptation.

The combination of sharpness-aware regularization and noise removal strategies makes SAR particularly effective in dynamic environments, where distribution shifts and noisy data are common. The method encourages the model to focus on the most relevant information from the target domain while maintaining the robustness required to adapt to unforeseen variations. By addressing both the optimization landscape's stability and the quality of target data, SAR provides a more reliable and stable approach to test-time adaptation, which is crucial for real-world applications where labeled target data is scarce or unavailable, and test-time conditions may vary significantly from training data.

Unlike these most existing approaches to test-time adaptation, which rely on updating the model's parameters, Laplacian Adjusted Maximum-likelihood Estimation (**LAME**) (Boudiaf, Mueller, Ben Ayed & Bertinetto, 2022) is a parameter-free method that adapts model predictions without modifying the underlying network weights. This design choice makes LAME particularly

appealing in scenarios where memory, computational constraints, or deployment requirements preclude test-time fine-tuning.

LAME operates by adjusting the model's output predictions using the local structure of the test data in feature space. For each incoming test sample, the method constructs a local neighborhood graph based on feature similarity between the current and recent samples. This structure is encoded using a graph Laplacian matrix, which is then used to regularize a maximum-likelihood estimation problem. The result is a refined set of predictions that are smoothed across nearby samples, promoting label consistency and robustness to domain shift.

Importantly, the adaptation process is fully analytical and requires no gradient descent, hyperparameter tuning, or access to source data. By working exclusively at the prediction level, LAME avoids the instability and overfitting risks associated with updating model parameters on small, unlabeled target batches. It is also well-suited for online settings, where test samples arrive sequentially and cannot be revisited. Despite its simplicity, LAME has shown strong performance on a range of domain adaptation benchmarks, offering a compelling balance between efficiency and accuracy in real-world deployment scenarios.

Pseudo-labels have also played a central role in several test-time adaptation methods. These approaches typically rely on the model's own predictions on unlabeled target samples as surrogate labels to guide adaptation. For instance, SHOT (Liang *et al.*, 2020) employs pseudo-labels in a regularization loss based on mutual information. It optimizes the entire feature encoder while keeping the classifier fixed, aiming to maximize the confidence and discriminability of predictions on the target data. CoTTA (Wang, Fink, Van Gool & Dai, 2022b) extends this idea within a student-teacher framework, using a consistency loss between predictions on original and augmented inputs. The teacher model is updated via exponential moving average of the student, promoting stable adaptation over time. PAD (Wu, Yue & Sangiovanni-Vincentelli, 2021) further enhances pseudo-label reliability by applying multiple augmentations to each input and aggregating predictions via voting, thereby filtering out uncertain predictions and reinforcing consistent ones. These methods illustrate how pseudo-labeling can effectively guide adaptation,

but also highlight the importance of mitigating confirmation bias and prediction noise during test-time optimization.

1.4 Test-Time Adaptation on VLMs

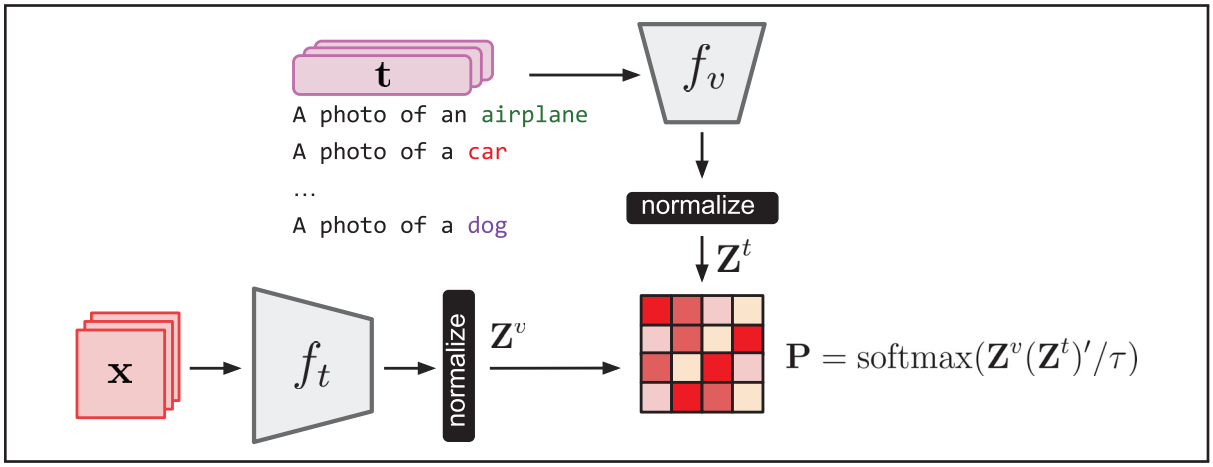


Figure 1.5 Architecture of CLIP for zero-shot classification. The cosine similarity between the text embeddings $\mathbf{Z}_t$ and the image embeddings $\mathbf{Z}_v$ is computed to determine the final prediction

In the realm of vision-language models, recent work has explored how to adapt foundation models such as CLIP (see Figure 1.5) to distribution shifts during test time. One of the early contributions in this direction is Test-time Prompt Tuning (**TPT**) (Shu *et al.*, 2022) (see Figure 1.6). The core idea is to modify the textual prompts used by the model to better align with the distribution of unseen target data, while keeping the visual encoder fixed. To achieve this, TPT treats the textual prompt as a learnable parameter, referred to as a soft prompt, and optimizes it by minimizing the prediction entropy on the target data. The entropy loss encourages the model to make confident predictions and implicitly aligns the text features with the test distribution.

To enhance robustness, TPT applies multiple data augmentations to each test input and jointly optimizes the prompt across these transformations. The final classification is obtained by averaging the model's predictions over all augmentations, thus mitigating the risk of overfitting to a specific view. While effective, TPT requires iterative gradient-based optimization of the

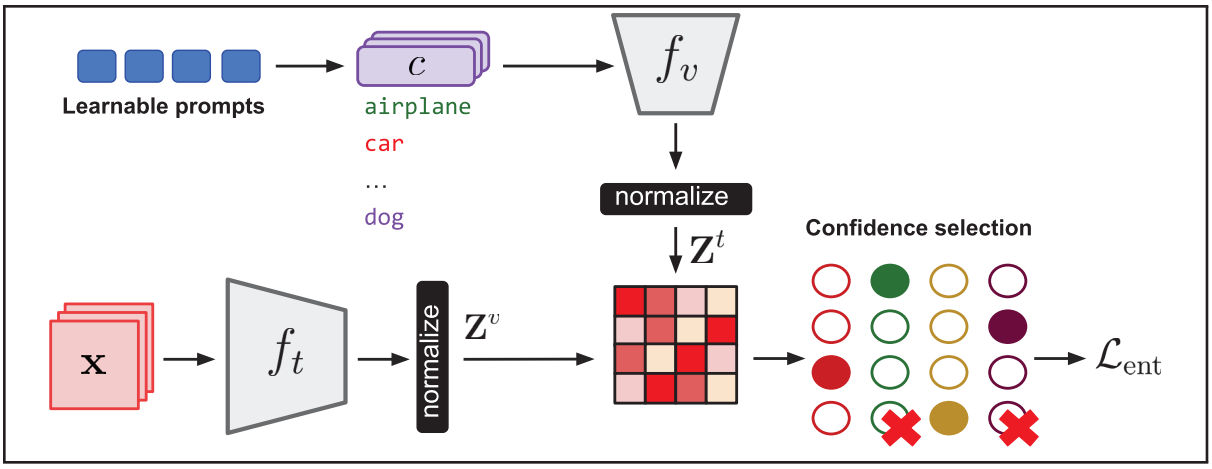


Figure 1.6 Architecture of TPT. The text and the image features are fixed, only the prompt is tuned based on entropy minimization after a confidence selection

prompt embeddings at inference time, which results in substantial computational overhead. Moreover, since the model does not observe any ground-truth labels from the target domain, the optimization landscape can be unstable, making the method sensitive to the prompt initialization and learning rate.

DiffTPT (Feng, Yu, Liu, Khan & Zuo, 2023) extends TPT by addressing two of its limitations: limited diversity of augmentations and the noisiness of entropy-based updates. In addition to standard input augmentations (e.g., flips, crops, color jitter), DiffTPT generates semantically diverse augmented views of each test image using a pre-trained diffusion model. These diffusion-based augmentations enable the model to explore a wider space of visual contexts during adaptation, potentially improving generalization to novel domains.

To avoid introducing noise from poorly aligned augmentations, DiffTPT employs a filtration mechanism based on cosine similarity in the embedding space. Specifically, it compares the feature vectors of augmented views and retains only those with high similarity, thereby filtering out ambiguous or semantically irrelevant augmentations. The retained views are then used to compute a weighted entropy loss, which guides the optimization of the prompt embeddings.

This strategy enhances the reliability of the pseudo-supervision signal during adaptation and improves robustness to spurious features in the target domain. However, as with TPT, DiffTPT still requires gradient-based optimization of the prompts, and the inclusion of a diffusion model increases computational cost. Nevertheless, it provides a more stable and effective approach for adapting CLIP-like models to unseen distributions.

In contrast to the optimization-based TPT and DiffTPT approaches, Test-time Dynamic Adapter (**TDA**) (Karmanov, Guan, Lu, El Saddik & Xing, 2024) offers a training-free alternative that adapts foundation models using lightweight operations at inference time. TDA builds upon the Tip-Adapter framework, which enhances CLIP predictions by computing similarity scores between test samples and a set of support features stored in a key-value cache. Each key in the cache corresponds to a feature vector from a support image, while the value encodes its pseudo-label. At inference, TDA computes the similarity between a test feature and the cache, and aggregates the top-k retrieved labels to refine the prediction.

To improve reliability, TDA introduces a dynamic confidence weighting mechanism that reweights each retrieved key-value pair based on its similarity to the query feature. This adaptive weighting increases robustness to noisy pseudo-labels and helps prevent over-reliance on outlier cache entries. Additionally, TDA refines pseudo-labels online by applying an entropy minimization loss on the retrieved support features and their similarity scores, improving the alignment between features and predictions.

Unlike prompt tuning methods, TDA does not require any gradient updates or modification of model parameters. This makes it highly efficient and scalable to large test sets. However, a key limitation is its sensitivity to the quality and size of the support cache. Similar to Tip-Adapter, the method requires tuning of key hyperparameters such as the weighting factor and number of top-k neighbors, which can vary significantly across datasets. Moreover, because the cache must be constructed from a representative set of support features, TDA may struggle in settings with highly dynamic or evolving data distributions unless cache updating mechanisms are implemented.

CALIP (Guo *et al.*, 2023) introduces a parameter-free test-time adaptation mechanism that jointly adapts both textual and visual features. It uses a cross-attention module to align the representations from the two modalities and then combines the refined features for final prediction. While CALIP does not require gradient-based optimization or explicit pseudo-labeling, it depends on access to the entire test set to perform hyperparameter search and calibration, which limits its application in streaming or real-time scenarios.

More recently, SwapPrompt (Ma, Zhang, Guo & Xu, 2024) proposes a dual-prompt architecture for self-supervised test-time adaptation. It relies on contrastive learning to swap between a fixed and a learnable prompt, encouraging the model to preserve consistency across augmented views. By avoiding the need for ground truth labels or heavy optimization, SwapPrompt achieves strong performance across domain shifts with minimal computational overhead.

1.5 Discussion and Limitations

While a wide range of Test-Time Adaptation methods have emerged in recent years, several key limitations remain unresolved. Many approaches depend on entropy minimization or pseudo-labeling strategies that are inherently unstable when applied to miscalibrated or noisy test-time distributions. Batch normalization-based methods adapt only affine BN parameters and often degrade in performance when batch sizes are small or the target distribution is non-stationary. Self-supervised test-time training strategies improve robustness but are typically limited by their reliance on a fixed pretext task (e.g., rotation prediction), which may not align well with the downstream objective or domain-specific characteristics.

In the context of vision-language models, most adaptation methods operate at the level of prompts or prediction smoothing, often keeping the visual encoder completely frozen. Methods such as TPT and DiffTPT update only the text encoder or prompt embeddings, limiting their capacity to adapt visual features to the shifted distribution. Cache-based strategies like TDA sidestep model updates altogether, which may hinder adaptation depth when the domain gap is large.

In response to these challenges, this thesis introduces a series of contributions aimed at enhancing the robustness and applicability of TTA in practical scenarios. We begin with NC-TTT, a contrastive objective tailored to convolutional models that avoids the pitfalls of entropy minimization. We then extend test-time adaptation to vision-language models where we explicitly adapt the vision encoder at test time via Layer Normalization updates. CLIPArTT introduces a novel strategy for dynamically constructing text prompts, which guide the adaptation of LayerNorm parameters in the vision encoder using pseudo-labels derived from CLIP's own similarity scores. Finally, we propose WATT, a multi-template weight averaging framework for vision-language models that consolidates adaptation across diverse linguistic cues. Together, these methods push the boundaries of test-time adaptation toward more modular, lightweight, and generalizable solutions for deployment in real-world conditions.

CHAPTER 2

NC-TTT: A NOISE CONSTRASTIVE APPROACH FOR TEST-TIME TRAINING

David Osowiechi*, Gustavo A. Vargas Hakim*
Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah,
Ismail Ben Ayed, Christian Desrosiers

Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

*Equal Contribution
Spotlight at the IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)
June 2024.

Abstract

Despite their exceptional performance in vision tasks, deep learning models often struggle when faced with domain shifts during testing. Test-Time Training (TTT) methods have recently gained popularity by their ability to enhance the robustness of models through the addition of an auxiliary objective that is jointly optimized with the main task. Being strictly unsupervised, this auxiliary objective is used at test time to adapt the model without any access to labels. In this work, we propose Noise-Contrastive Test-Time Training (NC-TTT), a novel unsupervised TTT technique based on the discrimination of noisy feature maps. By learning to classify noisy views of projected feature maps, and then adapting the model accordingly on new domains, classification performance can be recovered by an important margin. Experiments on several popular test-time adaptation baselines demonstrate the advantages of our method compared to recent approaches for this task. The code can be found at: https://github.com/GustavoVargasHakim/NCTTT.git

2.1 Introduction

A crucial requirement for the success of traditional deep learning methods is that training and testing data should be sampled from the same distribution. As widely shown in the literature (Recht, Roelofs, Schmidt & Shankar, 2018; Peng *et al.*, 2018), this assumption rarely holds in practice and a model's performance can drop dramatically in the presence of domain shifts. The

field of Domain Adaptation (DA) has emerged to address this important issue, proposing various mechanisms that adapt learning algorithms to new domains.

In the realm of domain adaptation, two notable directions of research have surfaced: Domain Generalization and Test-Time Adaptation. Domain Generalization (DG) approaches (Volpi *et al.*, 2018; Prakash *et al.*, 2019; Zhou *et al.*, 2020; Kim *et al.*, 2022; Wang *et al.*, 2022a) typically train a model with an extensive source dataset encompassing diverse domains and augmentations, so that it can achieve a good performance on test examples from unseen domains, without retraining. Conversely, Test-Time Adaptation (TTA) (Wang *et al.*, 2020; Khurana *et al.*, 2021; Boudiaf *et al.*, 2022) entails the dynamic adjustment of the model to test data in real-time, typically adapting to subsets of the new domain, such as mini-batches. TTA presents a challenging, yet practical problem as it functions without supervision for test samples or access to the source domain data. While they do not require training data from diverse domains as DG approaches, TTA methods are often susceptible to the choice of unsupervised loss used at test time, a factor that can substantially influence their overall performance. Test-Time Training (TTT), as presented in (Sun *et al.*, 2020; Liu *et al.*, 2021; Gandelsman *et al.*, 2022; Osowiechi *et al.*, 2023; Vargas Hakim *et al.*, 2023), offers a compelling alternative to TTA. In TTT, an auxiliary task is learned from the training data (source domain) and subsequently applied during test-time to refine the model. Generally, unsupervised and self-supervised tasks are selected for their capacity to support an adaptable process, without relying on labeled data. Finally, employing a dual-task training approach in the source domain allows the model to be more confident at test time, as it is already familiar with the auxiliary loss.

Motivated by recent developments in machine learning using Noise-Contrastive Estimation (NCE) (Mnih & Kavukcuoglu, 2013; Oord, Li & Vinyals, 2018; Aneja, Schwing, Kautz & Vahdat, 2021), we introduce a Noise-Contrastive Test-Time-Training (NC-TTT) method that efficiently learns the distribution of sources samples by contrasting it with a noisy distribution. This is achieved by training a discriminator that learns to distinguish noisy out-of-distribution (OOD) features from in-distribution ones. At test time, the output of the discriminator is used to guide

the adaptation process, modifying the parameters of the network encoder so that it produces features that match in-distribution ones. Our contributions can be summarized as follows:

- We present an innovative Test-Time Training approach inspired by the paradigm of Noise-Contrastive Estimation (NCE). While NCE was initially proposed for generative models as a way to learn a data distribution without having to explicitly compute the partition function (Gutmann & Hyvärinen, 2010; Mnih & Kavukcuoglu, 2013), and later employed for unsupervised representation learning (Aneja *et al.*, 2021; Oord *et al.*, 2018), our work is the first to show the usefulness of this paradigm for test-time training.
- We motivate our method with a principled and efficient framework deriving from density estimation, and use this framework to guide the selection of important hyperparameters.
- In a comprehensive set of experiments, we expose our NC-TTT method to a variety of challenging TTA scenarios, each featuring unique types of domain shifts. Results of these experiments demonstrate the superior performance of our method compared to recent approaches for this problem.

The subsequent sections of this paper are structured as follows. Section 2.2 reviews prior research on TTA, TTT, and NCE. Section 2.3 presents our NC-TTT method along with the experimental framework for its evaluation, detailed in Section 2.4. Section 2.5 offers experimental results and discussions, while Section 2.6 concludes the paper with final remarks.

2.2 Related work

Test-Time Adaptation. TTA is the challenging problem of adapting a pre-trained model from a source domain to an unlabeled target domain in an online manner (i.e., on a batch-wise basis). In this problem, it is assumed that the model no longer has access to source samples, making the setting more realistic and applicable as an *off-the-shelf* tool. Finally, the online nature of TTA also limits the possibility of computing accurate target data distributions, specially when the number of samples is low.

Two classic TTA methods have prevailed in the literature, Prediction Time Batch Normalization (PTBN) (Nado *et al.*, 2021) and Test-Time Adaptation by Entropy Minimization (TENT) (Wang *et al.*, 2020). The former consists in simply recomputing the statistics from each batch of data inside the batch norm layers, instead of using the frozen source statistics. The latter goes one step further by minimizing the entropy loss on the model's predictions and updating only the affine parameters of the batch norm layers. Recently, LAME (Boudiaf *et al.*, 2022) introduced a closed-form optimization mechanism that acts on the model's predictions for target images. This method is based on the Laplacian of the feature maps, which enforces their clustering based on similarity. A more detailed presentation of TTA approaches can be found in (Liang, He & Tan, 2023).

Test-Time Training. TTA methods assume the existence of an implicit property in the model that can be linked to accuracy and can be used for adaptation at test time (e.g., entropy (Wang *et al.*, 2020)). In contrast, TTT techniques explicitly introduce a given property by learning a secondary task alongside the main classification task at training. As seminal work in the field, TTT (Sun *et al.*, 2020) introduced a Y-shaped architecture allowing for a self-supervised rotation prediction task. This sub-network can be attached to any layer of a CNN. Formally, the overall TTT objective is composed of a supervised loss $\mathcal{L}_{sup}$ (e.g., cross-entropy) and an auxiliary, task-dependent loss $\mathcal{L}_{aux}$, as follows:

$$\mathcal{L}_{TTT} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{aux} \tag{2.1}$$

The auxiliary loss is used at test time to update the model's encoder, reconditioning the features into being more similar to those from the source domain. TTT++ (Liu *et al.*, 2021) proposed using contrastive learning as the secondary task, while also preserving statistical information from the source domain's feature maps to align the test-time features. Similarly, TTT-MAE (He *et al.*, 2022) used Masked Autoencoder (MAE) (Gandelsman *et al.*, 2022) image reconstruction as the auxiliary task. Normalizing Flows (NF) (Dinh, Sohl-Dickstein & Bengio, 2016; Kingma & Dhariwal, 2018) have also been employed in TTTFlow (Osowiechi *et al.*, 2023), adapting the feature encoder at test time by approximating a likelihood-based domain shift

detector. Unlike previous approaches, TTTFlow requires two separate training procedures for the original model and the NF network, which makes source training more complex. Recently, ClusT3 (Vargas Hakim *et al.*, 2023) introduced an unsupervised secondary task where the projected features of a given layer are clustered using a mutual information maximization objective. Although ClusT3 achieves competitive results, the hyperparameters of this method (e.g., number clusters) are dataset dependent, which limits its generalization capabilities.

Noise-contrastive estimation (NCE). Our work is also related to NCE, a useful tool to model unknown distributions by *comparison* (Gutmann & Hyvärinen, 2010). In NCE, a dataset is contrasted against a set of noisy points drawn by an arbitrary distribution. A discriminator is then trained to distinguish between both sets, thereby learning the original dataset's properties. This approach has been employed to learn word embeddings (Mnih & Kavukcuoglu, 2013), training Variational Autoencoders (Aneja *et al.*, 2021), and self-supervised learning (InfoNCE) (Oord *et al.*, 2018), among others. To our knowledge, this work is the first to investigate the potential of NCE for test-time training. We hypothesize that NCE is well suited to estimate the source domain distribution at training time, and that this estimation can be used in an unsupervised manner at test time to adapt a model to target domain samples.

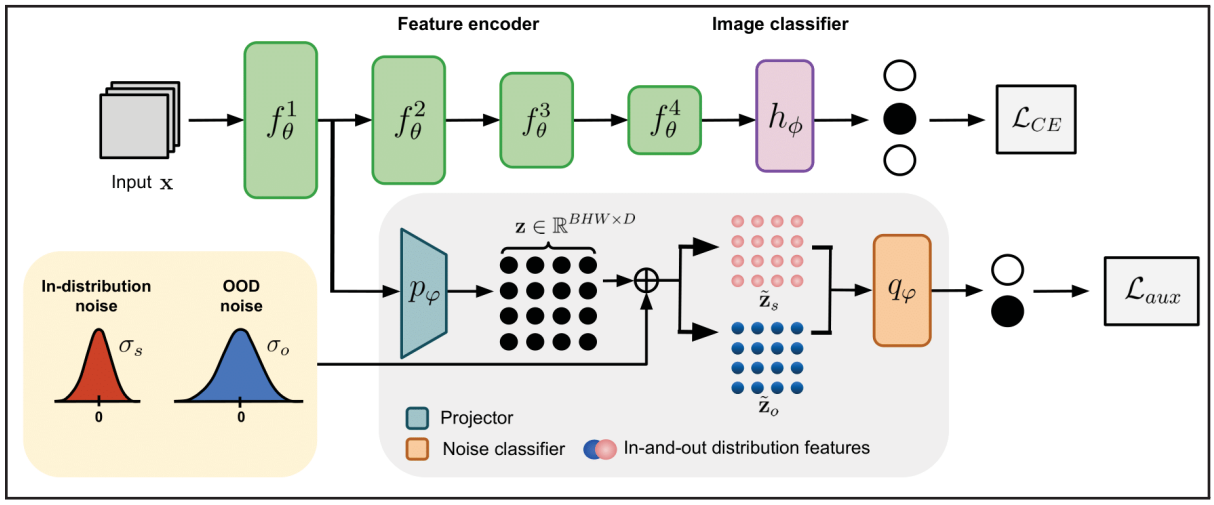


Figure 2.1 Overview of our Noise-Contrastive Test-Time-Training (NC-TTT) method. The auxiliary module comprises a linear projector p_φ that reduces the scale of features, and a classifier q_φ to discriminate between two different noisy views of the reduced features

2.3 Methodology

We begin by presenting an overview of our NC-TTT method for Test-Time Training. We then proceed to detail the Noise-Contrastive Estimation framework on which it is grounded.

2.3.1 The proposed method

The problem of Test-Time Training can be formally defined as follows. Let the source domain be represented by a joint distribution $\mathcal{P}(\mathcal{X}_s, \mathcal{Y}_s)$, where $\mathcal{X}_s$ and $\mathcal{Y}_s$ correspond to the image and labels spaces, respectively. Likewise, denote as $\mathcal{P}(\mathcal{X}_t, \mathcal{Y}_t)$ the target domain distribution, with $\mathcal{X}_t$ and $\mathcal{Y}_t$ as the respective target images and labels. Following previous research, we consider the likelihood shift (Boudiaf *et al.*, 2022) between source and target datasets, expressed as $\mathcal{P}(\mathcal{X}_s|\mathcal{Y}_s) \neq \mathcal{P}(\mathcal{X}_t|\mathcal{Y}_t)$, and assume the label space to be the same between domains ($\mathcal{Y}_s = \mathcal{Y}_t$). Given a model $F : \mathcal{X} \rightarrow \mathcal{Y}$ trained on source data $(\mathbf{x}, y) \in \mathcal{X}_s \times \mathcal{Y}_s$, the goal of TTT is to adapt this model to target domain examples from $\mathcal{X}_t$ at test time, without having access to source samples or target labels.

As shown in Fig. 2.1, our NC-TTT model follows the same Y-shaped architecture as in previous works, with the first branch corresponding to the main classification task and the second one to the auxiliary TTT task. The classification branch can be defined as $F_{\theta,\phi} = (h_\phi \circ f_\theta)$ where $f_\theta = (f_\theta^L \circ \ldots \circ f_\theta^1)$ is an encoder that transforms images into feature maps via L convolutional layers (blocks) and h_ϕ is a classification head that takes features from the last encoder layer and outputs the class probabilities. This branch is trained with a standard cross-entropy loss $\mathcal{L}_{CE}$

Following recent TTT approaches (Osowiechi *et al.*, 2023; Vargas Hakim *et al.*, 2023), our auxiliary task operates on the features of the encoder. Without loss of generality, we suppose that the features come from layer ℓ of the encoder and denote as $f_\theta^\ell(\mathbf{x}) \in \mathbb{R}^{B\times W\times H\times D}$ the D feature maps of size $W\times H$ for a batch of B images. We first reshape these feature maps to a $(BWH)\times D$ feature matrix and then use a linear projector to reduce its dimensionality, giving projected features $\mathbf{z} = p_\varphi(f_\theta^\ell(\mathbf{x})) \in \mathbb{R}^{BWH\times d}$ with $d \ll D$. Next, we generate two noisy versions of $\mathbf{z}$, an in-distribution version $\widetilde{\mathbf{z}}_s = \mathbf{z} + \boldsymbol{\epsilon}_s$, $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 I)$, and an out-of-distribution (OOD) version

$\widetilde{\mathbf{z}}_o = \mathbf{z} + \boldsymbol{\epsilon}_o$, $\boldsymbol{\epsilon}_o \sim \mathcal{N}(\mathbf{0}, \sigma_o^2 I)$ where $\sigma_o > \sigma_s$. These noisy features are fed into a discriminator q_φ which predicts in-distribution probabilities $[0, 1]^{BWH}$. This discriminator, which is built using two linear layers with ReLU in between, is trained by minimizing loss $\mathcal{L}_{aux}$ computing the binary cross-entropy between the predicted probabilities and *soft-labels* which will be described in the next section. To update the encoder parameters at test-time, as we do not have class labels, we only compute gradients from $\mathcal{L}_{aux}$.

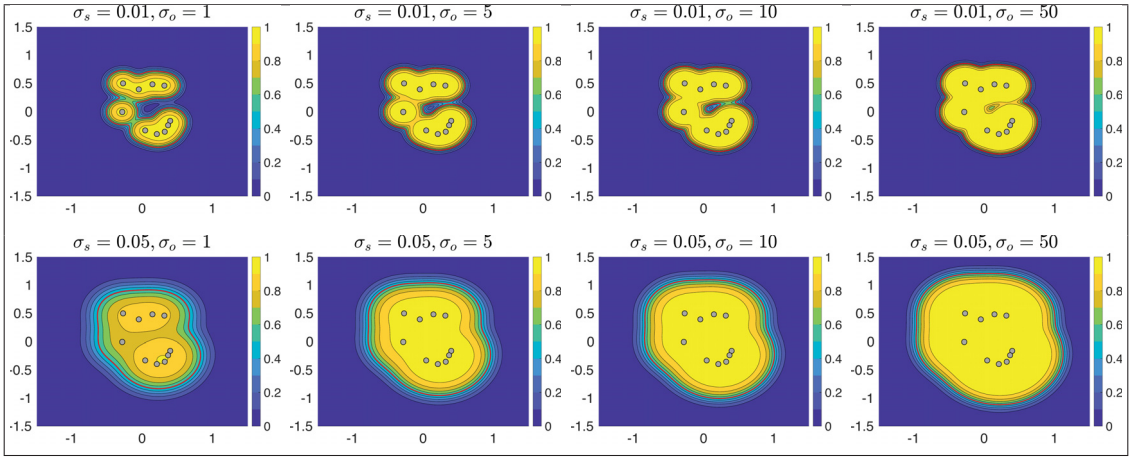


Figure 2.2 Posterior probability $p(y_s = 1|\mathbf{z})$ of 2D points with different pairs (σ_s, σ_o). The in-domain *influence* expands by increasing σ_o for a fixed σ_s (see difference row-wise). Furthermore, this region is more regular when σ_s increases when σ_o is fixed (see difference column-wise)

2.3.2 Noise-contrastive Test-time Training

We now present our noise-contrastive strategy for test-time training. Let us denote as $p_s(\mathbf{z})$ the probability of features from the source domain. Our method employs a density estimation strategy to learn $p_s(\mathbf{z})$ from training source examples $\mathcal{D}_s = \{\mathbf{z}_i\}_{i=1}^{N_s}$, where $N_s = BWH$. Afterwards, it uses the estimated distribution $\widehat{p}_s(\mathbf{z})$ to adapt the model to distribution shifts at test time.

Estimating the source distribution. We consider the well-known kernel density estimation approach to model $p_s(\mathbf{z})$. This approach puts a small probability mass around each training example $\mathbf{x}_i \in \mathcal{D}_s$, in the shape of a D-dimensional Gaussian with isotropic variance $\Sigma_s = \sigma_s^2 I$,

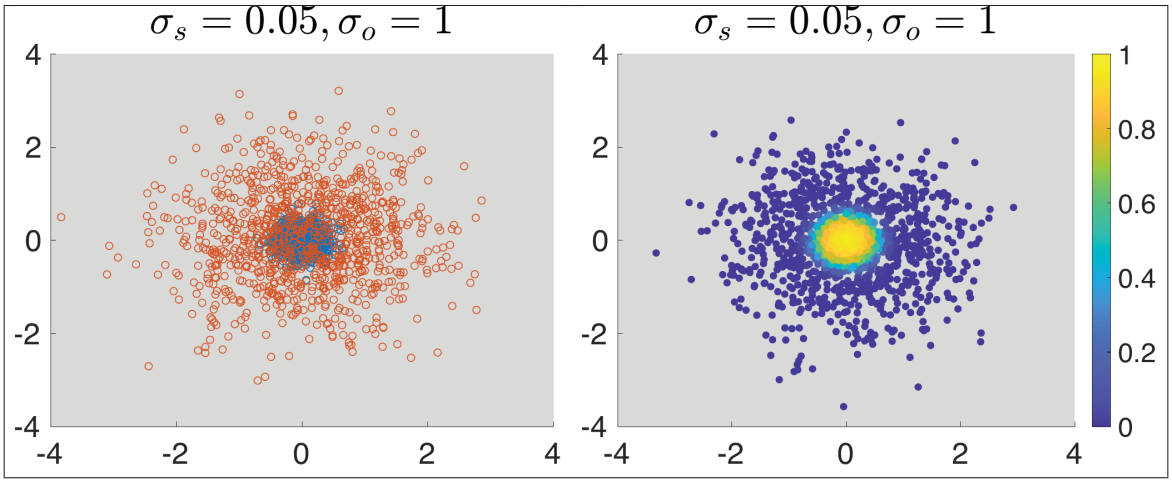


Figure 2.3 Noise 2D vectors sampled with $\sigma_s = 0.05$ and $\sigma_o = 1$ (*left*). The overlapping of both distributions can be overcome by assigning a probability to each point based on our threshold method

and then estimates the distribution as

$$\widehat{p}_s(\mathbf{z}) = \frac{1}{N_s(2\pi)^{D/2}\sigma_s^D} \sum_{i=1}^{N_s} \exp\left(-\frac{1}{2\sigma_s^2}\|\mathbf{z} - \mathbf{z}_i\|^2\right) \tag{2.2}$$

At test-time, one could use this probability estimation to define an adaption objective $\mathcal{L}_{aux}$ that minimizes the negative log-likelihood of test examples $\mathcal{D}_t = \{\mathbf{z}_j\}_{j=1}^{N_t}$:

$$\mathcal{L}_{aux} = -\frac{1}{N_t} \sum_{j=1}^{N_t} \log \widehat{p}_s(\mathbf{z}_j). \tag{2.3}$$

However, this simple approach faces two important issues. First, estimating the density in high-dimensional space is problematic since moving away from a training example quickly reduces the probability to zero. Second, the training examples from the source domain are no longer available at test time, hence the density of samples in Eq. (2.2) cannot be evaluated.

To overcome these issues, we propose a *noise contrastive* approach, which uses a discriminator to learn feature distribution $p_s(\mathbf{z})$. Toward this goal, we contrast $p_s(\mathbf{z})$ with an out-of-domain distribution $p_o(\mathbf{z})$ which is also estimated using Eq. (2.2) but replacing the variance with σ_o^2,

where $\sigma_o > \sigma_s$. Let y_s be a domain indicator variable such that $y_s = 1$ if an example is from the source domain, else $y_s = 0$. Assuming equal priors $p(y_s = 1) = p(y_s = 0)$, we can use Bayes' theorem to get the posterior

$$p(y_s = 1 \mid \mathbf{z}) \;=\; \frac{\widehat{p}_s(\mathbf{z})}{\widehat{p}_s(\mathbf{z}) \;+\; \widehat{p}_o(\mathbf{z})}. \tag{2.4}$$

To illustrate this model, we show in Figure 2.2 the probability $p(y_s = 1 \mid \mathbf{z})$ obtained for different values of σ_s and σ_o, when training with randomly-sampled 2D points. For a fixed σ_s, increasing σ_o expands the in-domain region around the training samples. Likewise, for the same σ_o, using a greater σ_s gives a larger and more regular (less determined by individual points) in-domain region.

Training the disciminator. To train the discriminator $q_\varphi(\cdot)$, for each training example $\mathbf{z}_i \in \mathcal{D}_s$, we generate $2M$ samples $\widetilde{\mathbf{z}}_{i,m} = \mathbf{z}_i + \boldsymbol{\epsilon}_{i,m}$, the first M from the in-domain distribution, i.e. $\boldsymbol{\epsilon}_{i,m} \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 I)$, and the other M ones from the noisier out-of-domain distribution, i.e. $\boldsymbol{\epsilon}_{i,m} \sim \mathcal{N}(\mathbf{0}, \sigma_o^2 I)$. For these samples, we assume that $\exp(-\|\widetilde{\mathbf{z}}_{i,m} - \mathbf{z}_j\|_2^2 / 2\sigma_s^2) \approx 0$, for $j \neq i$, hence the posterior simplifies to

$$p(y_s = 1 \mid \widetilde{\mathbf{z}}_{i,m}) \;= \frac{\sigma_s^{-D} \exp\!\left(-\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}_{i,m}\|^2\right)}{\sigma_s^{-D} \exp\!\left(-\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}_{i,m}\|^2\right) + \sigma_o^{-D} \exp\!\left(-\frac{1}{2\sigma_o^2}\|\boldsymbol{\epsilon}_{i,m}\|^2\right)} \tag{2.5}$$

where $\boldsymbol{\epsilon}_{i,m} = \widetilde{\mathbf{z}}_{i,m} - \mathbf{z}_i$. For large values of D, this formulation is numerically unstable it leads to *division by zero* errors. Instead, we use an equivalent formulation $p(y_s = 1 \mid \widetilde{\mathbf{z}}) = \text{sigmoid}(u)$, where pre-activation "logit" u is given by

$$u \;=\; \frac{1}{2}\left(\frac{1}{\sigma_o^2} - \frac{1}{\sigma_s^2}\right)\|\boldsymbol{\epsilon}_{i,m}\|^2 \;+\; D \log\left(\frac{\sigma_o}{\sigma_s}\right) \tag{2.6}$$

See Appendix A in the supplementary material for a proof. The in-domain region, $p(y_s = 1 \mid \widetilde{\mathbf{z}}) \geq 0.5$, which corresponds to the case where $u \geq 0$, is thus defined by the following

condition:

$$\|\boldsymbol{\epsilon}_{i,m}\| \leq \sigma_s \sigma_o \sqrt{\frac{2D}{(\sigma_s^2 - \sigma_o^2)} \log\left(\frac{\sigma_s}{\sigma_o}\right)} \tag{2.7}$$

Figure 2.3 shows examples of noise vectors $\boldsymbol{\epsilon}$ sampled with $\sigma_s = 0.05$ and $\sigma_o = 1$ (*left*), and their corresponding posterior probability (*right*). As can be seen, the posterior probability correctly separates in-distribution samples from OOD ones. Doing so, it overcomes the problem of having OOD samples that are similar to in-distribution ones (red circles near the center), which would confuse the discriminator during training.

Using these samples $\widetilde{\mathbf{z}}_{i,m}$, we train the discriminator $q_\varphi(\cdot)$ by minimizing the cross-entropy between its prediction and the soft-label $\widetilde{p}_{i,m} = p(y_s = 1 \,|\, \widetilde{\mathbf{z}}_{i,m})$:

$$\begin{aligned} \mathcal{L}_{aux} = -\frac{1}{2MN_s} \sum_{i=1}^{N_s} \sum_{m=1}^{2M} & \widetilde{p}_{i,m} \log q_\varphi(\widetilde{\mathbf{z}}_{i,m}) \\ & + \left(1 - \widetilde{p}_{i,m}\right) \log\left(1 - q_\varphi(\widetilde{\mathbf{z}}_{i,m})\right) \end{aligned} \tag{2.8}$$

Adapting the model at test time. During inference, we adapt the parameters of the encoder in layers where the auxiliary loss is computed, as well as those of preceding layers. The adaptation modifies the encoder so that the trained discriminator $q_\varphi(\cdot)$ perceives the encoded features $\{\mathbf{z}_j\}_{j=1}^{N_t}$ of test examples as being in-distribution. This is achieved by minimizing the following test-time loss:

$$\mathcal{L}_{aux}^{test} = -\frac{1}{N_t} \sum_{j=1}^{N_t} \log q_\varphi(\mathbf{z}_j) \tag{2.9}$$

As illustrated in Fig. 2.4, our method models the in-distribution probability $p(y_s = 1 \,|\, \mathbf{z})$ using NCE and then approximates this distribution with discriminator $q_\varphi(\cdot)$. At test time, the encoder is updated to move OOD features (white point) toward the source distribution, making them more suitable for the source-trained classifier. Thanks to the non-zero in-distribution noise ($\sigma_s > 0$), we avoid over-adapting the encoder (the white point stops at the border of the in-distribution region and not at a training sample), a problem often found in other TTT approaches.

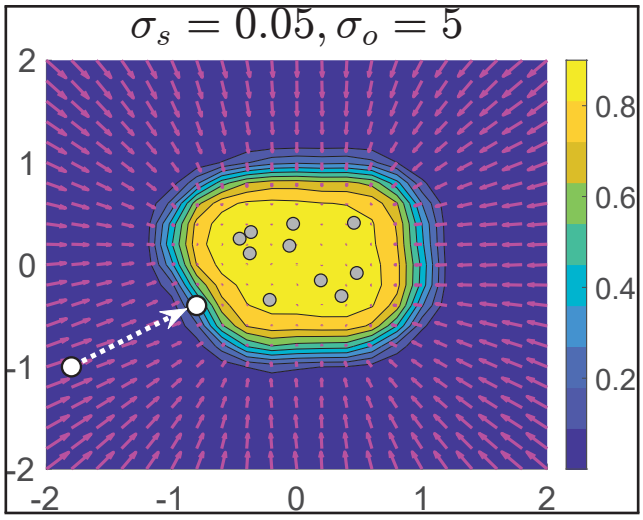


Figure 2.4 Heatmap of in-distribution probabilities, i.e., $p(y_s = 1 \mid \mathbf{z})$ approximated by $q_\varphi(\mathbf{z})$ in our model, and spatial gradient of log-likelihood function, i.e. $\nabla \log q_\varphi(\mathbf{z})$, which is used as test-time adaptation objective. The arrow shows how an OOD test sample (white point) is adapted toward the source distribution

2.3.3 Selecting the distribution variances

Our model requires to specify the in-distribution variance σ_s^2 and the OOD variance σ_o^2. In this section, we present how these can be chosen. The OOD variance should be greater than the in-distribution, hence we can write $\sigma_o = \beta\sigma_s$, with $\beta = \sigma_o/\sigma_s > 1$. Hence, β is a measure of noise ratio for the in-distribution and OOD samples. Using this relationship, Eq. (2.6) simplifies to

$$u = -\frac{1}{2\sigma_s^2}\left(\frac{\beta^2 - 1}{\beta^2}\right)\|\boldsymbol{\epsilon}\|^2 + D\log\beta \tag{2.10}$$

For OOD samples, the expected value of "logit" u is then given by

$$\begin{aligned}\overline{u}_\beta &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\sigma_o^2 I)}\left[-\frac{1}{2\sigma_s^2}\left(\frac{\beta^2-1}{\beta^2}\right)\|\boldsymbol{\epsilon}\|^2 + D\log\beta\right] \\ &= -\frac{1}{2\sigma_s^2}\left(\frac{\beta^2-1}{\beta^2}\right)\underbrace{\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\sigma_o^2 I)}\left[\|\boldsymbol{\epsilon}\|^2\right]}_{\sigma_o^2=\beta^2\sigma_s^2} + D\log\beta \\ &= -\frac{1}{2}\left(\beta^2-1\right) + D\log\beta\end{aligned} \tag{2.11}$$

Figure 2.5 show how the expected in-distribution prediction $\mathbb{E}[y_s \mid \mathbf{z}] = \text{sigmoid}(\overline{u}_\beta)$ varies as function of β, for $D = 16$ (the dimension used in our experiments). In this case, to have near-zero probability for OOD samples, one can choose any $\beta > 1.5$. In our experiments, we selected $\beta = 2$.

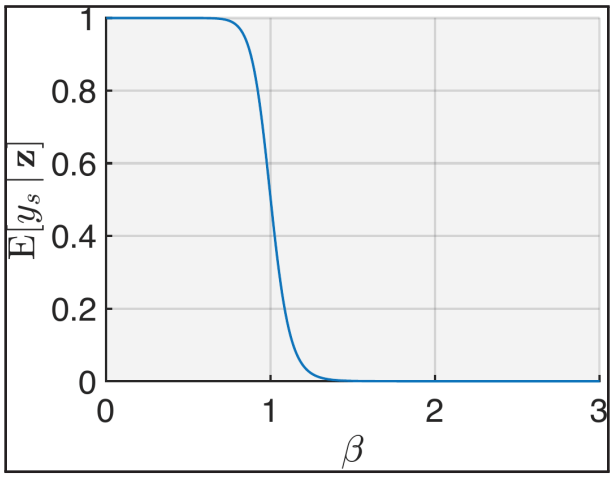


Figure 2.5 Expected in-distribution label as a function of noise ratio $\beta = \sigma_o/\sigma_s$

2.4 Experimental Settings

We evaluate NC-TTT on several TTT datasets, following the protocol of previous works. These benchmarks emulate different challenging domain shift scenarios, which help evaluating the effectiveness of our approach. As in (Sun *et al.*, 2020; Vargas Hakim *et al.*, 2023), these benchmarks are categorized as *common corruptions*, and *sim-to-real* domain shift.

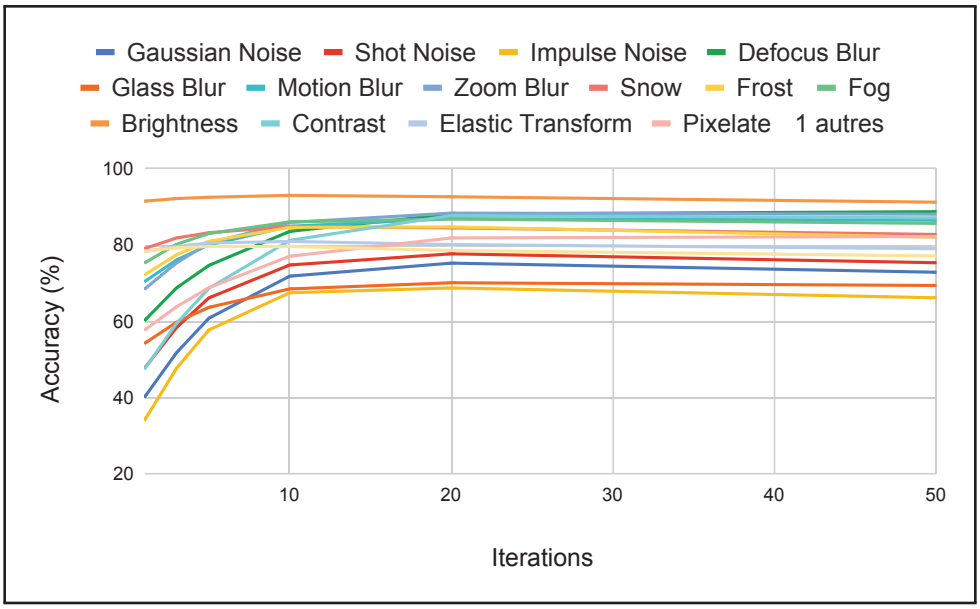


Figure 2.6 Evolution of accuracy on all corruptions in CIFAR-10-C

For *common corruptions*, we evaluate our method on CIFAR-10-C and CIFAR-100-C (Hendrycks & Dietterich 2019). This family of domain shifts include 15 different corruptions such as Gaussian noise, JPEG compression, among others. Each corruption has 5 different levels of severity with 10,000 images, which amounts to 75 different testing scenarios. For each of the aforementioned datasets, CIFAR-10 and CIFAR-100 are used as source domains, with 10 and 100 classes respectively. Finally, the challenging large-scale VisDA-C (Peng *et al.*, 2018) dataset corresponds to the *sim-to-real domain shift*. The source domain comprises a training set of 152,397 images of 3D renderings from 12 different classes, while the test set consists in 72,372 video frames of the same categories.

Source training. The cross-entropy and auxiliary losses are jointly trained on the source dataset. We explored different architectural choices for each setting. For common corruptions (i.e. CIFAR-10/100-C), we define the projector as a 1×1 convolutional layer that reduces the number of channels to $D = 96$ to later be flattened for classification. We utilize a discriminator composed of two linear layers with a Batch Norm layer and Leaky ReLU in between, and a hidden dimension of 1024 in the intermediate layer. For this particular case, we use the tuple

Table 2.1 Accuracy (%) on CIFAR-10-C dataset with Level 5 corruption for NC-TTT compared to previous TTA and TTT methods

	ResNet50	LAME	PTBN	TENT	TTT	TTT++	ClusT3	NC-TTT (ours)
Gaussian Noise	21.01	22.90 ±0.07	57.23 ±0.13	57.15 ±0.19	66.14 ±0.12	75.87 ±5.05	**76.01 ±0.19**	75.30 ±0.04
Shot noise	25.77	27.11 ±0.13	61.18 ±0.03	61.08 ±0.18	68.93 ±0.06	77.18 ±1.36	77.67 ±0.17	**77.74 ±0.05**
Impulse Noise	14.02	30.99 ±0.15	54.74 ±0.13	54.63 ±0.15	56.65 ±0.03	**70.47 ±2.18**	69.76 ±0.15	68.80 ±0.11
Defocus blur	51.59	45.16 ±0.13	81.61 ±0.07	81.39 ±0.22	88.11 ±0.08	86.02 ±1.35	87.85 ±0.11	**88.77 ±0.09**
Glass blur	47.96	36.58 ±0.06	53.43 ±0.11	53.36 ±0.14	60.67 ±0.06	69.98 ±1.62	**71.34 ±0.15**	70.15 ±0.16
Motion blur	62.30	55.41 ±0.15	78.20 ±0.28	78.04 ±0.17	83.52 ±0.03	85.93 ±0.24	86.10 ±0.11	**86.93 ±0.05**
Zoom blur	59.49	51.48 ±0.20	80.29 ±0.13	80.26 ±0.22	87.25 ±0.03	**88.88 ±0.95**	86.68 ±0.05	88.40 ±0.06
Snow	75.41	66.14 ±0.12	71.59 ±0.21	71.59 ±0.04	79.29 ±0.05	82.24 ±1.69	83.71 ±0.09	**84.92 ±0.08**
Frost	63.14	50.03 ±0.22	68.77 ±0.25	68.52 ±0.20	79.84 ±0.11	82.74 ±1.63	83.69 ±0.03	**84.79 ±0.05**
Fog	69.63	64.56 ±0.19	75.79 ±0.05	75.73 ±0.10	84.46 ±0.09	84.16 ±0.28	85.12 ±0.13	**86.85 ±0.10**
Brightness	90.53	84.27 ±0.10	84.97 ±0.05	84.77 ±0.13	91.23 ±0.08	89.07 ±1.20	91.52 ±0.02	**93.05 ±0.03**
Contrast	33.88	31.46 ±0.23	80.81 ±0.15	80.70 ±0.15	**88.58 ±0.09**	86.60 ±1.39	84.40 ±0.11	87.78 ±0.15
Elastic transform	74.51	64.23 ±0.10	67.14 ±0.17	67.13 ±0.10	75.69 ±0.10	78.46 ±1.83	**82.04 ±0.17**	80.99 ±0.11
Pixelate	44.43	39.32 ±0.08	69.17 ±0.31	68.70 ±0.29	76.35 ±0.19	**82.53 ±2.01**	82.03 ±0.09	82.26 ±0.11
JPEG compression	73.61	66.19 ±0.02	65.86 ±0.05	65.83 ±0.07	73.10 ±0.19	81.76 ±1.58	**83.24 ±0.10**	79.66 ±0.06
Average	53.82	49.06	70.05	69.93	77.32	81.46	82.08	**82.43**

Table 2.2 Accuracy (%) on CIFAR-100-C dataset with Level 5 corruption for NC-TTT and the works from the *state-of-the-art*

	ResNet50	LAME	PTBN	TENT	TTT	ClusT3	NC-TTT (ours)
Gaussian Noise	12.67	10.55 ±0.08	43.00 ±0.16	43.17 ±0.24	33.99 ±0.11	**49.77 ±0.18**	46.03 ±0.12
Shot noise	14.79	12.58 ±0.04	44.57 ±0.16	44.47 ±0.23	36.55 ±0.08	**50.54 ±0.16**	47.04 ±0.14
Impulse Noise	6.47	5.83 ±0.07	36.76 ±0.11	36.64 ±0.28	26.87 ±0.08	**44.35 ±0.31**	41.53 ±0.11
Defocus blur	29.97	29.07 ±0.11	66.68 ±0.06	66.74 ±0.06	65.96 ±0.14	64.40 ±0.12	**67.00 ±0.09**
Glass blur	21.36	19.58 ±0.02	45.17 ±0.08	45.09 ±0.06	34.90 ±0.01	**50.78 ±0.24**	48.08 ±0.07
Motion blur	39.60	41.26 ±0.09	62.61 ±0.17	62.54 ±0.23	57.10 ±0.10	62.62 ±0.15	**64.31 ±0.02**
Zoom blur	35.75	34.93 ±0.02	65.36 ±0.03	65.29 ±0.05	62.90 ±0.07	63.81 ±0.08	**66.24 ±0.25**
Snow	42.05	43.58 ±0.20	52.82 ±0.27	52.31 ±0.16	54.97 ±0.03	55.84 ±0.12	**58.70 ±0.10**
Frost	31.44	32.67 ±0.12	51.92 ±0.09	51.79 ±0.23	54.60 ±0.16	55.46 ±0.06	**58.55 ±0.11**
Fog	30.96	35.95 ±0.12	55.78 ±0.05	55.91 ±0.28	55.80 ±0.09	51.39 ±0.07	**57.73 ±0.17**
Brightness	61.80	64.84 ±0.03	66.20 ±0.06	66.47 ±0.06	73.25 ±0.06	66.71 ±0.11	**71.36 ±0.10**
Contrast	12.31	15.50 ±0.04	60.84 ±0.15	60.91 ±0.19	60.97 ±0.09	54.67 ±0.05	**61.53 ±0.20**
Elastic transform	53.06	51.32 ±0.13	56.38 ±0.04	56.43 ±0.33	53.51 ±0.04	59.44 ±0.27	**60.25 ±0.04**
Pixelate	26.08	27.65 ±0.02	58.21 ±0.14	58.19 ±0.22	50.39 ±0.05	60.75 ±0.09	**61.17 ±0.33**
JPEG compression	52.19	49.95 ±0.07	51.65 ±0.16	51.30 ±0.16	49.62 ±0.09	**59.94 ±0.12**	55.69 ±0.09
Average	31.37	31.68	54.53	54.48	51.43	56.70	**57.68**

($\sigma_s = 0, \sigma_o = 0.015$), which was experimentally determined as it produced the best performance. The model is trained using 128 images per batch for 350 epochs using SGD, an initial learning rate of 0.1, and a multi-step scheduler with a decreasing factor of 10 at epochs 150 and 250. Due to the challenging nature of the *sim-to-real* domain shift from VisDA-C, we escalate the architecture to make it able to learn more source domain information. We utilize a 1×1 convolutional

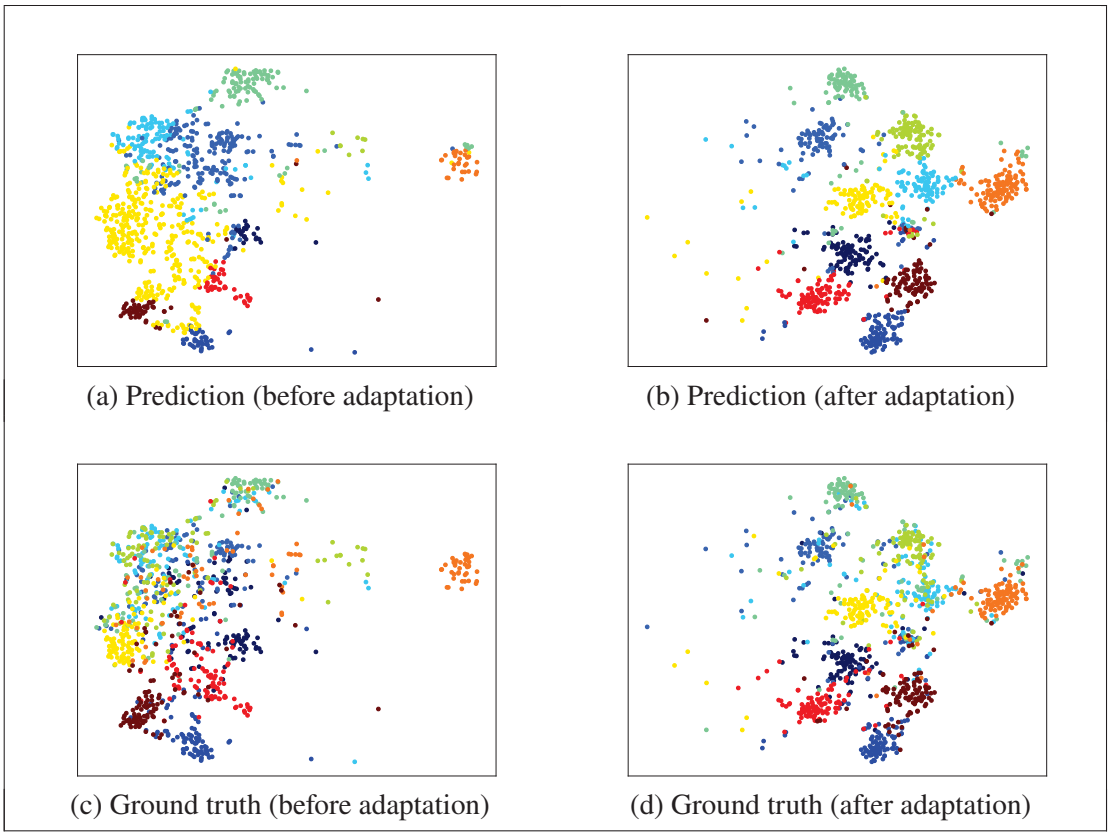
projector with an output number of channels of $D = 16$. As opposed to flattening the features, we also employ two 1×1 convolutional layers for the discriminator, with an intermediate number of channels of 1024. The noise values are sampled with ($\sigma_s = 0.025, \sigma_o = 0.05$) and added *pixel-wise* to the projected feature maps. Following related works' protocol for VisDA-C, we use an ImageNet-pre-trained model (Deng *et al.*, 2009) as a warm start, to then perform the source training with a batch size of 50 for 100 epochs with SGD and a learning rate of 0.01. ResNet50 (He, Zhang, Ren & Sun, 2016) is the chosen architecture for all datasets.

Test-time adaptation. Adaptation is performed on the encoder's blocks (including BatchNorm layers). If the auxiliary task is plugged to the third layer block, for instance, the weights of all the previous blocks will be optimized. The source training on CIFAR-10 is used to adapt for CIFAR-10-C. In an analog way, CIFAR-100 is utilized to adapt for CIFAR-100-C. For all this cases, the ADAM optimizer with a learning rate of 10^{-5} is used in batches of 128 images. As for VisDA-C, a batch size of 50 is employed with a learning rate of 10^{-4}. The weights of the source model are restored after each batch.

Benchmarking. We compare the performance of NC-TTT with previous works from the *state-of-the-art* in TTT and TTA. Chosen works in TTA include PTBN (Nado *et al.*, 2021), TENT (Wang *et al.*, 2020), and LAME (Boudiaf *et al.*, 2022), whereas for TTT we consider TTT (Sun *et al.*, 2020), TTT++ (Liu *et al.*, 2021), and ClusT3 (Vargas Hakim *et al.*, 2023). We utilize the source model (named ResNet50 in our results) without adaptation to measure accuracy gains.

2.5 Results

In this section, we present the experimental results obtained from NC-TTT and compare them against the *state-of-the-art*. In accordance with previous TTT research, we also offer insights on the working mechanisms that take part in the success of our technique.



(a) Prediction (before adaptation)

(b) Prediction (after adaptation)

(c) Ground truth (before adaptation)

(d) Ground truth (after adaptation)

Figure 2.7 t-SNE visualizations depict shot noise characteristics in the features extracted from NC-TTT. Panels (a) and (b) illustrate the model predictions without and with 20 iterations of adaptation, respectively. Panels (c) and (d) showcase the ground truth labels in the absence of adaptation and for the adapted representations, respectively

2.5.1 Image classification on common corruptions

We assess the performance of ClusT3 using the CIFAR-10/100-C dataset, considering 15 distinct corruptions. Subsequently, our experiments concentrate exclusively on Level 5, recognized as the most demanding adaptation scenario. Comprehensive results for all severity levels are provided in the Supplementary material.

The data presented in Fig 2.6 reveals that peak accuracy is typically reached around 20 iterations, depending on the specific corruption type. Remarkably, accuracy remains stable even beyond the 20th iteration. In the case of certain corruptions, specifically the ones with noise such as Impulse

Noise which significantly degrade the image quality, we observe a decline in performance with an increase in the number of adaptation iterations.

As shown in Table 2.1, NC-TTT achieves an average improvement of 30.61% with respect to the baseline (i.e. ResNet50), and obtains a considerable advantage in all the different corruptions. Moreover, our method achieves to outperform ClusT3 in most corruptions and in average for the whole dataset. It is worth noticing that, besides the strong relation of NC-TTT to Gaussian-like noise, the performance on the *Gaussian Noise* corruption is not necessarily the highest, which could be due to the fact that the auxiliary task does not bias the model towards any type of domain shift. Table 2.2 shows a more surprising trend on CIFAR-100-C, as our technique outperforms the closest competitor on the majority of the corruptions, and obtains an average improvement of 26.31% with respect to ResNet50. Based on the above, NC-TTT can approximate the source information even when: a) the number of classes increases, and b) the auxiliary task works at a smaller scale as the main classification task.

Figure 2.7 demonstrates the impact of NC-TTT during adaptation through t-SNE plots showcasing the target feature maps before and after adaptation, along with the associated model predictions. The challenging corruption of shot noise becomes more manageable with the assistance of NCE, contributing to improved predictions by refining the clustering of diverse class samples within the target dataset.

2.5.2 Image classification on sim-to-real domain shift

For adaptation on VisDA-C, the first encoder's layer block is chosen for the auxiliary task. The obtained results concur with previous works (Sun *et al.*, 2020; Vargas Hakim *et al.*, 2023), in that the first layers of the network's encoder are sufficient for adaptation.

As shown in Table 2.3, NC-TTT obtains a competitive performance with respect to previous works on VisDA-C. The severe domain shift in this dataset makes it a very challenging scenario, as can be seen when testing the source model. NC-TTT obtains a gain of 16.19% in accuracy, and surpasses previous methods by an important margin.

Table 2.3 Results on VisDA-C

Method	**Acc. (%)**
ResNet50	46.31
LAME-L (Boudiaf *et al.*, 2022)	22.02 ±0.23
LAME-K (Boudiaf *et al.*, 2022)	42.89 ±0.14
LAME-R (Boudiaf *et al.*, 2022)	19.33 ±0.11
PTBN (Nado *et al.*, 2021)	60.33 ±0.04
TENT (Wang *et al.*, 2020)	60.34 ±0.05
TTT (Sun *et al.*, 2020)	40.57 ±0.02
ClusT3 (Vargas Hakim *et al.*, 2023)	61.91 ±0.02
NC-TTT (ours)	**62.71 ±0.09**

2.6 Conclusions

We proposed NC-TTT, a Test-Time Training method based on the popular theory of Noise-Contrastive Estimation. Our method learns a proximal representation of the source domain by discriminating between noisy views of feature maps. The entire model can be added on top of any given layer of a CNN's encoder, and comprises only a linear projector and a classifier.

The proposed experiments support already established hypothesis of TTT, which states that adaptation in the first encoder's layer blocks (e.g. first or second) is often sufficient to recover the model's performance on a new domain. NC-TTT is evaluated on different challenging benchmarks, and its performance is compared against recent *state-of-the-art* methods in the field.

This work leads to interesting questions that can be addressed as future work. First, different types of added noise could be explored to analyze their impact in the learning of the auxiliary task. A similar framework can eventually be derived for different distributions. Moreover, and as an open question partaking all the existent TTT methods, the exact mechanisms that allow auxiliary tasks to learn domain-related information are unclear. This is especially intriguing considering that the scale of such tasks is small compared to the classification task. Their properties and their relation with the models' performance a suitable research direction.

CHAPTER 3

CLIPARTT: ADAPTATION OF CLIP TO NEW DOMAINS AT TEST TIME

David Osowiechi*, Gustavo A. Vargas Hakim*
Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah,
Ismail Ben Ayed, Christian Desrosiers

Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

*Equal Contribution
Poster at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)
February 2025.

Abstract

Pre-trained vision-language models (VLMs), exemplified by CLIP, demonstrate remarkable adaptability across zero-shot classification tasks without additional training. However, their performance diminishes in the presence of domain shifts. In this study, we introduce CLIP Adaptation duRing Test-Time (CLIPArTT), a fully test-time adaptation (TTA) approach for CLIP, which involves automatic text prompts construction during inference for their use as text supervision. Our method employs a unique, minimally invasive text prompt tuning process, wherein multiple predicted classes are aggregated into a single new text prompt, used as *pseudo label* to re-classify inputs in a transductive manner. Additionally, we pioneer the standardization of TTA benchmarks (e.g., TENT) in the realm of VLMs. Our findings demonstrate that, without requiring additional transformations nor new trainable modules, CLIPArTT enhances performance dynamically across non-corrupted datasets such as CIFAR-100, corrupted datasets like CIFAR-100-C and ImageNet-C, alongside synthetic datasets such as VisDA-C. This research underscores the potential for improving VLMs' adaptability through novel test-time strategies, offering insights for robust performance across varied datasets and environments. The code can be found at: https://github.com/dosowiechi/CLIPArTT.git

3.1 Introduction

Combining vision and language modalities for learning, namely a Vision Language model (VLM), has demonstrated an outstanding performance in different vision tasks (Radford *et al.*, 2021; Jia *et al.*, 2021; Kirillov *et al.*, 2023). Remarkably, these models are surprisingly effective at zero-shot generalization, where a new task can be outside the original scope of the training set, without any additional supervision to fine tune the model. Models such as CLIP (Radford *et al.*, 2021) have then been employed in fields as diverse as video recognition (Lin *et al.*, 2022), audio (Guzhov, Raue, Hees & Dengel, 2022), and medical imaging (Liu *et al.*, 2023).

As in other more traditional deep architectures (e.g., CNNs), CLIP is prone to performance degradation on domains to which it was not originally exposed. Recent research trends suggest that domain adaptation mechanisms can play an important role in deploying CLIP (Lai *et al.*, 2023; Shu *et al.*, 2022). The challenge, however, is adapting the model to new domains in an efficient manner, so that its attractive zero-shot capabilities is maintained without the need of retraining.

In this paper, CLIP is contextualized in the setting of Test-Time Adaptation, a challenging yet practical scenario of domain adaptation. In this scenario, a model needs to adapt to new data *on-the-fly* to cope with unknown distribution shifts, and without using any class supervision. Although an experimental boilerplate was standardized in recent years, CLIP has not been integrated to it yet. Additionally, we introduce a powerful adaptation technique that achieves *state-of-the-art* performance without a significant computational overhead. Comprehensive studies are performed on multiple datasets containing different types of domain shifts on several levels of severity, resulting in a total of 59 evaluation scenarios. Our main contributions can be summarized as follows:

- We propose CLIPArTT, a Test-Time Adaptation method for CLIP that adapts the VLM by updating normalization-layer parameters. This is achieved by combining multiple classes into a single new text prompt, which is then used as a pseudo-label.

- We introduce a new benchmark for Test-Time Adaptation on Vision-Language Models by implementing representative baselines, such as TENT.
- Through comprehensive experiments, we subject our CLIPArTT methodology to diverse and challenging Test-Time Adaptation scenarios, each characterized by distinct types of domain shifts. The outcomes of these experiments highlight the superior performance of our approach when compared against other methodologies addressing similar challenges.

3.2 Related work

Test-Time Adaptation. TTA is a particular setting of Domain Adaptation, encompassing two main characteristics: (*a*) adapting a model to a target domain, with inputs coming as unlabeled data streams (i.e., batches), (*b*) without any access to the source domain samples. The former challenge complicates accurately estimating the target domain's distribution, while the latter impedes solving the problem by directly comparing measures of the domain distributions (e.g., feature means). Despite the challenging nature of the problem, the field has gained important momentum in recent years, providing insights on the possibilities and limitations of adapting pre-trained models.

A key focus in TTA methods is adapting batch normalization layers, which retain important source domain information. PTBN (Nado *et al.*, 2021) adjusts batch statistics at test time, while TENT (Wang *et al.*, 2020) refines the affine parameters using entropy minimization on predictions. Entropy minimization, used in various methods (Goyal, Sun, Raghunanthan & Kolter, 2022; Zhang *et al.*, 2022; Lin, Lai, Pan & Yin, 2023; Lee, Das, Choo & Choi, 2023), enhances model confidence without label supervision but often depends on image augmentations or large batches. For example, Text-Prompt Tuning (TPT) (Shu *et al.*, 2022) learns text prompt adapters for CLIP using entropy minimization. However, this approach needs to perform several augmentations for each test sample, making it computationally expensive. Test-Time Distribution Normalization (TTDN) (Zhou, Ren, Li, Zabih & Lim, 2023) normalizes test data to match the training distribution, but needs access to source data or approximations of the mean of each

test batch. Our method fine-tunes normalization layers, leverages prediction confidence for text supervision, and avoids input augmentations.

Recent techniques have also sought to adapt CLIP in a gradient-free manner. CALIP (Guo *et al.*, 2023) adapts the visual and text features bidirectionally through a parameter-free attention module. The features are later combined to obtain refined predictions. Although efficient, this method needs the features of the entire test set for conducting a hyperparameter search, which limits its generalization to very large datasets. TDA (Karmanov *et al.*, 2024) dynamically builds a positive and a negative cache to adapt features through the Tip-Adapter strategy (Zhang *et al.*, 2021). This method however requires to find specific hyperparameters for each dataset, as in the original Tip-Adapter approach.

Pseudo-labels have been central to previous TTA methods. SHOT (Liang *et al.*, 2020) uses them in a regularization loss based on mutual information, optimizing the entire feature encoder. CoTTA (Wang *et al.*, 2022b) employs pseudo-labels in a student-teacher model with consistency loss between original and augmented inputs. PAD (Wu *et al.*, 2021) enhances pseudo-labels by augmenting inputs and voting on predictions. Instead of simple class pseudo-labels, our CLIPArTT method exploits the text supervision to better predict the correct class. By combining visual and text information into a single pseudo-label guided by the most probable classes, we can direct the model towards a more certain prediction.

Other methods take less conventional approaches to TTA. LAME (Boudiaf *et al.*, 2022) refines classifier predictions transductively using feature similarity through the Laplacian. Test-time training (TTT) methods (Sun *et al.*, 2020; Liu *et al.*, 2021; Osowiechi *et al.*, 2023; Gandelsman *et al.*, 2022; Vargas Hakim *et al.*, 2023) train a sub-branch alongside the main network in an unsupervised manner to update the model, requiring training from scratch on the source domain. Like LAME, our method relates to Laplacian regularization but applies it differently. While LAME updates predictions without altering the model, our method uses it in a test-time adaptation loss to enforce consistency between embeddings of related batch samples. Unlike

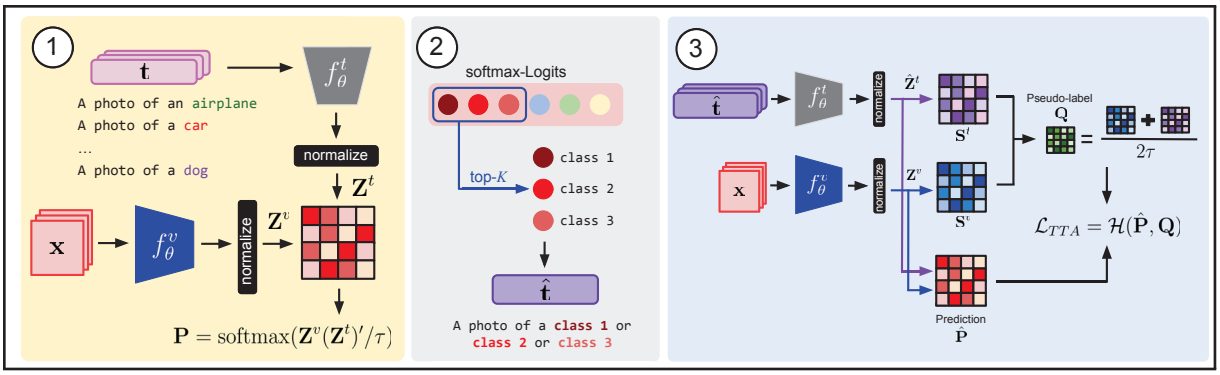


Figure 3.1 CLIPArTT pipeline overview: 1) Computing predictions from Image-Text Similarity, 2) generating a new text prompt by filtering the top-K class predictions, 3) with the new prompts, a pseudo-label $\mathbf{Q}$ is obtained by averaging the image-to-image and text-to-text similarity scores, while the prediction $\hat{\mathbf{P}}$ is computed as the image-to-text similarity. Cross-entropy is then used as the TTA loss

TTT methods, we do not require additional branches in the network or training this network from scratch.

Conformal Learning. CLIPArTT is also related to the field of conformal learning, where intervals of confidence for new predictions are derived from previous experience (Shafer & Vovk, 2008). In a conformal prediction, a given level of certainty is assigned to a set $C = \{c_1, ..., c_K\}$ with the K most plausible classes that an input can belong to (Angelopoulos & Bates, 2021). We draw inspiration from this concept and build a conformal set of class predictions that can help adapting CLIP towards an accurate top-1 prediction. Our technique, however, stands out by not relying on image transformations such as in (Shu *et al.*, 2022; Lai *et al.*, 2023) to filter out predictions.

3.3 Methodology

We start by presenting the vanilla CLIP model for classification and explain how it can be extended to test-time adaptation using entropy minimization. Building on the limitations of this approach, we then introduce our CLIPArTT method that leverages class uncertainty and the relationship between samples in a batch.

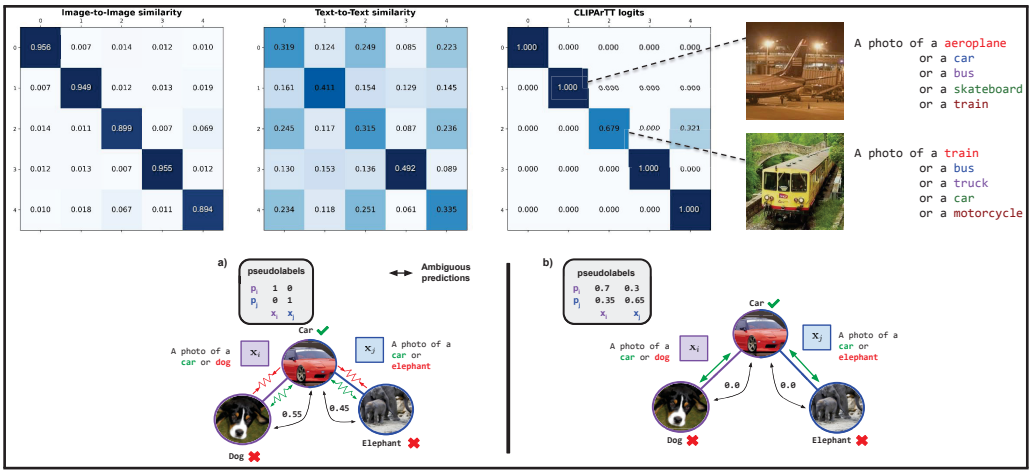


Figure 3.2 **Top**: Example of similarity matrices ($\mathbf{S}^v$, $\mathbf{S}^t$) and CLIPArTT softmax probabilities ($\mathbf{Q}$) for a batch of 5 examples and using $K = 5$ classes. **Bottom**: a) When using the identity matrix as pseudo-label for contrastive learning, the correct prediction is ambiguous, as the images are forced to both approaching and moving away from the right class. b) CLIPArTT uses soft pseudo-labels that smoothly guides the prediction towards the correct class by reducing the impact of ambiguities in the prompts

3.3.1 CLIP-based classification

Contrastive Language-Image Pre-training (CLIP) (Radford *et al.*, 2021) consists of a visual encoder $f_\theta^v(\cdot)$, mapping an image $\mathbf{x}$ to visual features $\mathbf{z}^v \in \mathbb{R}^D$, and a text encoder $f_\theta^t(\cdot)$ transforming text prompts $\mathbf{t}$ to text features $\mathbf{z}^t \in \mathbb{R}^D$. The visual and text encoders are trained jointly with a contrastive loss so that the feature embeddings of training images and their corresponding text prompt are close to each other, while those of different training examples are pushed apart.

In a classification task with K fixed classes, CLIP can be used to perform inference by encoding a pre-defined text prompt for each class, for example $\mathbf{t}_k =$ "`a photo of a {class k}`". For a new image $\mathbf{x}_i$, the probability of belonging to class k is then estimated based on cosine similarity,

$$p_{ik} = \frac{\exp\left(\cos(\mathbf{z}_i^v, \mathbf{z}_k^t)/\tau\right)}{\sum_j \exp\left(\cos(\mathbf{z}_i^v, \mathbf{z}_j^t)/\tau\right)}, \quad \cos(\mathbf{z}, \mathbf{z}') = \frac{\mathbf{z}^\top \mathbf{z}'}{\|\mathbf{z}\|_2 \cdot \|\mathbf{z}'\|_2}, \tag{3.1}$$

where τ is a suitable softmax temperature.

The model in Eq. (3.1) can be used for test-time adaptation in various ways, the simplest one being entropy minimization as in TENT. This approach, which relies on the principle that the decision boundary lies in a low-density region of space, giving rise to a low prediction entropy, adapts the model parameters by minimizing entropy on a test batch of size B:

$$\mathcal{L}_{\text{TENT}}(\theta) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{k=1}^{K} p_{ik}\log p_{ik} \tag{3.2}$$

However, this approach, as similar ones based on pseudo-labels (Liang *et al.*, 2020; Wang *et al.*, 2022b; Wu *et al.*, 2021), suffers from two important limitations. First, due to domain shifts, the model's prediction may be unreliable (e.g., giving the highest probability to the wrong class) and techniques such as entropy minimization or standard pseudo-label will only reinforce these errors during adaptation. Secondly, they assume that samples in a test batch are independent and do not directly leverage their semantic relationships.

Table 3.1 Accuracy (%) on CIFAR-10/100 and CIFAR-10/100-C datasets with Level 5 corruption for the top-1 or the top-3 predicted classes

	CIFAR10		CIFAR100	
	Top-1	Top-3	Top-1	Top-3
Original	88.74	97.79	61.68	80.92
Corrupted (-C)	59.22	82.43	29.43	46.61

3.3.2 Our CLIPArTT method

The proposed CLIPArTT method (see Figure 3.1), addresses the above-mentioned limitations using two key insights. The first insight, inspired by conformal learning, is that the correct class is often among the top most probable ones, although the model's most confident prediction may not be always correct. This claim is supported by the results in Table 3.1 (first row: *Original*), showing that the correct class is within the top-3 predictions 97.79% of the times for CIFAR-10

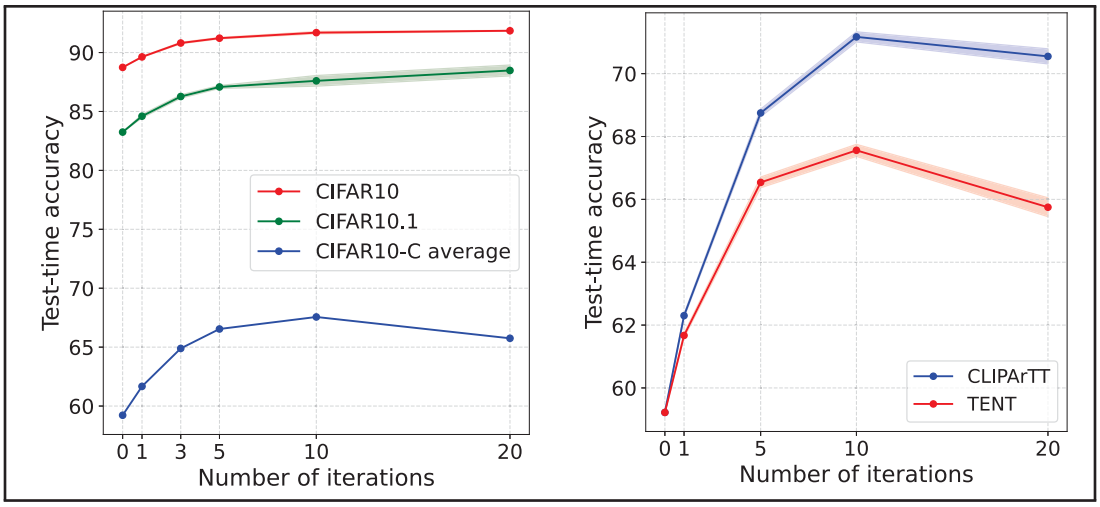


Figure 3.3 Evolution of CLIPArTT's accuracy during test-time adaptation. **Left**: For different versions of CIFAR10. **Right**: Compared to TENT on CIFAR10-C

(versus 88.74% within the top-1), and 80.92% of the times for CIFAR-100 (versus 61.68% within the top-1). A way to include information about multiple classes could, therefore, help adapting the model at test time. The second insight is that the similarity between the batch samples could be evaluated based on their visual and/or text embeddings. As we will show below, such similarities could be exploited in a strategy related to Stochastic Neighbor Embedding (SNE) and graph-Laplacian regularization.

Instance-specific multi-class prompt. Inspired by our first insight and recent work investigating CLIP's ability to perform compositional logical reasoning (Brody, 2023), we devise a novel technique to generate instance-specific prompts from the top-k predictions, with $1 \leq k \ll K$. Specifically, for an image $\mathbf{x}_i$, we estimate the CLIP-based class probabilities using Eq. (3.1) and then generate a new text prompt as $\hat{\mathbf{t}}_i$ = `a photo of a {class` i_1`} or ... or {class` i_k`}`", where `{class` i_j`}` is the name of the class with j-th highest probability.

Transductive TTA. Next, we design a test-time adaptation loss that accounts for semantic relationships between batch samples. Let $\mathbf{Z}^v \in \mathbb{R}^{B\times D}$ and $\hat{\mathbf{Z}}^t \in \mathbb{R}^{B\times D}$ denote the *normalized* visual and instance-specific text embeddings of the samples within the test batch, respectively.

We compute an image-to-image similarity matrix, $\mathbf{S}^v = \mathbf{Z}^v(\mathbf{Z}^v)^\top \in [-1, 1]^{B\times B}$, and a text-to-text similarity matrix, $\mathbf{S}^t = \hat{\mathbf{Z}}^t(\hat{\mathbf{Z}}^t)^\top \in [-1, 1]^{B\times B}$. The former measures the affinity between each pair of samples within the batch in terms of their visual characteristics (shapes, textures, etc.). The latter captures common (or related) classes in the top-k predictions of two samples, since it is computed using the instance-specific multi-class prompts. As illustrated in Figure 3.2 (*top*), a broad range of similarity values are obtained with this approach.

We deploy these two pairwise similarity matrices to compute pseudo-labels as follows:

$$\mathbf{Q} = \text{softmax}\big((\mathbf{S}^v + \mathbf{S}^t)/2\tau\big) \in [0, 1]^{B\times B} \tag{3.3}$$

where the softmax operation is applied column-wise and the temperature $\tau = 0.01$ is used in all our experiments.

Let $\hat{\mathbf{P}}$ denote the zero-shot prediction matrix using our instance-specific multi-class text prompts:

$$\hat{\mathbf{P}} = \text{softmax}\big(\mathbf{Z}^v(\hat{\mathbf{Z}}^t)^\top/\tau\big), \;\; \hat{p}_{ij} = \frac{\exp\big(\cos(\mathbf{z}_i^v, \hat{\mathbf{z}}_j^t)/\tau\big)}{\sum_k \exp\big(\cos(\mathbf{z}_i^v, \hat{\mathbf{z}}_k^t)/\tau\big)} \tag{3.4}$$

This matrix, along with the pairwise pseudo-labels we introduced in Eq. (3.3), yield our final TTA loss based on cross-entropy:

$$\mathcal{L}_{\text{TTA}}(\theta) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} q_{ij} \log \hat{p}_{ij}. \tag{3.5}$$

Unlike recent approaches like TPT (Shu *et al.*, 2022), which adapt CLIP by learning a text prompt, we instead update the normalization-layer parameters of the *visual* encoder, which yields a light-weight, computationally efficient TTA method. We note that TPT requires to generate multiple augmentations during the forward pass, incurring a substantial overhead.

The mechanism of CLIPArTT is illustrated in Fig. 3.2. Our soft pseudo-label reduces the risk of increasing the ambiguity of the predictions. Using the identity matrix (i.e., hard pseudo-labels)

leads to uncertain class assignments, as the ambiguity of the text prompts would both attract and repel the right class if it is present in the same text prompt for two different images.

3.3.3 Link to existing techniques

An important element of our TTA method is that it exploits the similarities between pairs of samples within the batch, in terms of visual features and features of text prompts representing the top-k classes. In this section, we draw a connection between the proposed method and two well-known techniques in machine learning: Stochastic Neighbor Embedding (Hinton & Roweis, 2002) and graph-Laplacian regularization (Chapelle, Scholkopf & Zien, 2006; Iscen, Tolias, Avrithis & Chum, 2019).

Stochastic Neighbor Embedding (SNE). This popular technique for dimensionality reduction estimates the local probability of a point $\mathbf{x}_j$ given a point $\mathbf{x}_i$ based on their Euclidean distance $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$:

$$p_{ij} = \frac{\exp\left(-d_{ij}^2\right)}{\sum_{k \neq i} \exp\left(-d_{ik}^2\right)} \tag{3.6}$$

The goal is to find, for each $\mathbf{x}_i$, a low-dimensional embedding $\mathbf{y}_i$ such that the local probabilities q_{ij} computed using Eq. (3.6) on these embeddings, is similar to p_{ij}. This is achieved by minimizing the row-wise KL divergence between distribution matrices $\mathbf{P}$ and $\mathbf{Q}$. Our CLIPArTT method can be linked to SNE since $\cos(\mathbf{x}_i, \mathbf{x}_j) = -d_{ij}^2 + 2$ when $\mathbf{x}_i$, $\mathbf{x}_j$ are unit normalized, and KL divergence between $\mathbf{P}$ and $\mathbf{Q}$ is equal to the cross-entropy between these matrices minus the entropy of $\mathbf{P}$. In summary, our TTA loss in Eq. (3.5) ensures that the inter-modality (text-to-image) similarities of batch samples are aligned with their intra-modality ones (text-to-text and image-to-image).

Graph-Laplacian regularization is commonly-used in the context semi-supervised learning techniques (Chapelle *et al.*, 2006) and label propagation (Iscen *et al.*, 2019). In our case, the

Laplacian regularizer is computed over a set of B nodes with predictions $\mathbf{Z} \in \mathbb{R}^{B\times D}$ as:

$$\mathcal{L}_{\text{reg}}(\mathbf{Z}) = \text{trace}\big(\mathbf{Z}^\top \mathbf{L}_\text{W} \mathbf{Z}\big) \;=\; \sum_{i,j} w_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2, \tag{3.7}$$

where $\mathbf{L}_\text{W} \in \mathbb{R}^{B\times B}$ is the Laplacian matrix defined from edge weight matrix $\mathbf{W}$ as follows:

$$[\mathbf{L}_\text{W}]_{ij} = \begin{cases} d_{ii} = \sum_j w_{ij}, & \text{if } i = j \\ -w_{ij} & \text{else.} \end{cases} \tag{3.8}$$

This connection is made in the following proposition.

Proposition 1. *The TTA loss in Eq. (3.5) can be expressed as a Laplacian regularization over a bipartite graph with one set of nodes for image embeddings and another for text embeddings.*

Proof. Expanding the softmax in Eq. (3.5), we get

$$\mathcal{L}_\text{TTA} \;=\; -\frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{B} q_{ij} \log \frac{\exp\left((\mathbf{z}_i^v)^\top \mathbf{z}_j^t / \tau\right)}{\sum_k \exp\left((\mathbf{z}_i^v)^\top \mathbf{z}_k^t / \tau\right)} \tag{3.9}$$

$$= \frac{1}{\tau B} \sum_i \sum_j q_{ij} \Big(-(\mathbf{z}_i^v)^\top \mathbf{z}_j^t + \tau \underbrace{\log \sum_k \exp\left((\mathbf{z}_i^v)^\top \mathbf{z}_k^t / \tau\right)}_{\text{LogSumExp (LSE)}} \Big) \tag{3.10}$$

Since the feature embeddings are normalized, we have $\|\mathbf{z}_i^v\|_2 = \|\mathbf{z}_j^t\|_2 = 1$ and thus $(\mathbf{z}_i^v)^\top \mathbf{z}_j^t = 2 - \|\mathbf{z}_i^v - \mathbf{z}_j^t\|_2^2$. Moreover, following the scaled LogSumExp (LSE) rule, we can bound the right-side term as follows:

$$\max_k \{(\mathbf{z}_i^v)^\top \mathbf{z}_k^t\} \;<\; \tau LSE \;\leq\; \max_k \{(\mathbf{z}_i^v)^\top \mathbf{z}_k^t\} \;+\; \tau \log B \tag{3.11}$$

For a small τ (we use a value of 0.01 in our method), the bounds tighten and we get that

$$LSE \;\approx\; \tfrac{1}{\tau}\max_k \{(\mathbf{z}_i^v)^\top \mathbf{z}_k^t\} \;=\; \tfrac{1}{\tau}(\mathbf{z}_i^v)^\top \mathbf{z}_i^t \tag{3.12}$$

The last equality comes from the hypothesis that, in a well trained model, the maximum similarity occurs between the image embedding of a sample and its corresponding text embedding. Our TTA loss can then be expressed as

$$\begin{aligned}\mathcal{L}_{\text{TTA}} \;&\approx\; \frac{1}{\tau B}\Big(\sum_{i,j} q_{ij}\|\mathbf{z}_i^v - \mathbf{z}_j^t\|_2^2 \;+\; \sum_i (\mathbf{z}_i^v)^\top \mathbf{z}_i^t\Big) \;+ const \\ &=\; \frac{1}{\tau B}\text{trace}\big((\mathbf{Z}^v)^\top(\mathbf{L}_\text{Q} + \mathbf{I})\,\mathbf{Z}^t\big) \;+ const\end{aligned} \tag{3.13}$$

where $\mathbf{I}$ is the identity matrix. The modified Laplacian $\mathbf{L}_\text{Q} + \mathbf{I}$ enforces nodes with a high connection weight (q_{ij}) to have similar embeddings, while also avoiding embeddings to collapse into a single vector. □

3.4 Experimental Settings

We evaluate CLIPArTT's performance across a variety of TTA datasets, covering scenarios such as natural images, common corruptions, simulated images, video, and Domain Generalization benchmark. This comprehensive framework allows for a robust assessment of the model's adaptability to various challenges, including domain shifts and corruptions. For a detailed description of the datasets, please refer to the supplementary materials.

Test-time adaptation. For test-time adaptation, model updates are applied to all the Layer Normalization (LN) layers within the visual encoder. We employ the Adam optimizer with a fixed learning rate set to 10^{-3}. Throughout our experiments, a consistent batch size of 128 is utilized to maintain uniformity and enable effective comparisons across different scenarios. As in previous TTA works, a smaller learning rate of 10^{-4} is preferred to adapt to the 3D renderings split (Vargas Hakim *et al.*, 2023), as it represents a more aggressive shift.

Table 3.2 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with Level 5 corruption for different number of K selected classes to create pseudo-labels

	$K = 1$	$K = 3$	$K = 4$
CIFAR-10	89.80 ±0.05	90.04 ±0.13	**90.41 ±0.07**
CIFAR-10.1	85.37 ±0.17	**86.35 ±0.27**	86.07 ±0.21
CIFAR-10-C	70.79 ±0.15	**71.17 ±0.16**	70.99 ±0.15

Table 3.3 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with Level 5 corruption for different number of iterations to update the model at test-time

	Iter = 1	Iter = 5	Iter = 10	Iter = 20
CIFAR-10	89.59 ±0.01	**90.54 ±0.09**	90.04 ±0.13	88.32 ±0.12
CIFAR-10.1	84.78 ±0.02	**86.67 ±0.06**	86.35 ±0.27	84.33 ±0.31
CIFAR-10-C	62.30 ±0.06	68.75 ±0.12	**71.17 ±0.16**	70.55 ±0.24

Benchmarking. We compare CLIPArTT with *state-of-the-art* methods. Specifically, we utilize adapted versions of TENT (Wang *et al.*, 2020) and LAME (Boudiaf *et al.*, 2022) tailored to CLIP. While the classifier logits are used in previous TTA research involving CNNs, we follow the standard practice and use the image-to-text similarity to obtain the final logits in CLIP. Only the visual encoder is optimized where needed. TENT now updates only the affine parameters in LN layers through entropy minimization directly on these logits. Notably, we employ 10 iterations for TENT adaptation, a choice informed by our findings, as evidenced in Fig 3.3. In LAME, the Laplacian regularizer is applied on the similarity between image features (obtained from the visual encoder). Finally, we include CLIP-tailored methods TTDN (Zhou *et al.*, 2023), TPT (Shu *et al.*, 2022), TDA (Karmanov *et al.*, 2024) and CALIP (Guo *et al.*, 2023). Both TDA and CALIP require hyperparameter tuning for optimal performance. To ensure fairness in our benchmarking process, we keep the hyperparameters of these approaches as reported in their original works. The batch-size for TPT is reduced to 32 due to its reliance on image augmentations and high memory requirements.

Table 3.4 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with different targets

	Avg. Prompt	Image	Text	Image + Text
CIFAR-10	76.94 ±0.41	**90.18 ±0.02**	89.05 ±0.14	90.04 ±0.13
CIFAR 10.1	67.83 ±0.49	86.25 ±0.37	84.85 ±0.40	**86.35 ±0.27**
CIFAR-10-C	43.08 ±0.08	70.98 ±0.15	70.03 ±0.21	**71.17 ±0.16**

3.5 Results

In this section, we outline the experimental outcomes derived from CLIPArTT and provide a comprehensive analysis of results. We first show a series of exploratory ablations that later help conduct the final experiments on the different datasets and compare with *state-of-the-art* approaches.

3.5.1 Ablation Studies

Number of classes for new prompts. Determining the number of classes in building new prompts presents a critical aspect in our methodology. Table 3.1 illustrates various scenarios where $K = 3$ emerges as a favorable choice, exhibiting a remarkable accuracy above 90% across multiple instances of CIFAR-10-C datasets. This initial observation is further corroborated by the findings presented in Table 3.2. However, the decision becomes more nuanced when applied to CIFAR-100 datasets. While leveraging the top 3 classes contributes to enhanced accuracy, it does not guarantee optimal performance. Similarly, adopting a strategy akin to CIFAR-10, wherein 30% of the classes are chosen, proves impractical due to resulting lengthy sentences that are impossible to tokenize. Nonetheless, our analysis shows that $K = 3$ consistently yields superior performance, particularly evident in the average results across CIFAR-100-C. Consequently, we maintain $K = 3$ as the preferred configuration for all subsequent experiments on CIFAR datasets.

Comparison against prompt averaging. The combination of K classes inside the template `{class` i_1`} or ... or {class` i_k`}`, poses a sensible question: how comparable is this

Table 3.5 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with Level 5 corruption for different number of batch-size

	CLIP	BS = 16	BS = 32	BS = 64	BS = 128
CIFAR-10	88.74	85.89 ±0.19	88.25 ±0.15	89.48 ±0.15	90.04 ±0.13
CIFAR-10.1	83.25	81.55 ±0.53	84.00 ±0.31	85.40 ±0.08	86.35 ±0.27
CIFAR-10-C	59.22	64.72 ±0.23	67.70 ±0.23	69.82 ±0.20	71.17 ±0.16

Table 3.6 Accuracy (%) on CIFAR-100-C and ImageNet-C datasets with ViT-B/32 as visual encoder

	CIFAR-100-C				ImageNet-C			
	CLIP	TENT	TDA	CLIPArTT	CLIP	TENT	TDA	CLIPArTT
Gaussian Noise	14.80	14.38 ±0.14	8.20 ±0.35	**25.32 ±0.14**	12.27	12.28 ±0.05	11.54 ±0.05	**20.18 ±0.61**
Shot noise	16.03	17.34 ±0.27	9.58 ±0.43	**27.90 ±0.05**	12.20	8.46 ±0.08	12.13 ±0.13	**20.94 ±0.08**
Impulse Noise	13.85	10.03 ±0.13	7.63 ±0.19	**25.62 ±0.09**	12.89	15.71 ±0.04	12.12 ±0.08	**19.95 ±0.02**
Defocus blur	36.74	49.05 ±0.07	25.59 ±0.41	**49.88 ±0.23**	21.60	20.61 ±7.05	21.39 ±0.08	**24.51 ±0.13**
Glass blur	14.19	3.71 ±0.07	9.83 ±0.56	**27.89 ±0.03**	10.84	17.20 ±0.08	10.74 ±0.06	**19.51 ±0.02**
Motion blur	36.14	46.62 ±0.27	28.92 ±0.18	**47.93 ±0.14**	17.85	24.60 ±0.04	18.45 ±0.06	**25.80 ±1.16**
Zoom blur	40.24	51.84 ±0.15	31.08 ±0.36	**52.70 ±0.06**	16.38	21.13 ±0.08	16.83 ±0.03	**24.00 ±0.06**
Snow	38.95	46.71 ±0.21	32.94 ±0.12	**49.72 ±0.01**	21.91	24.19 ±0.09	22.97 ±0.05	**27.26 ±0.06**
Frost	40.56	44.90 ±0.27	34.84 ±0.25	**49.63 ±0.12**	23.33	23.34 ±0.06	24.68 ±0.06	**26.56 ±0.08**
Fog	38.00	47.31 ±0.04	31.13 ±0.15	**48.77 ±0.04**	26.39	28.58 ±0.04	27.72 ±0.05	**34.68 ±0.02**
Brightness	48.18	60.58 ±0.18	42.36 ±0.10	**61.27 ±0.08**	45.87	47.16 ±0.01	48.06 ±0.04	**47.21 ±0.07**
Contrast	29.53	45.90 ±0.11	18.03 ±0.07	**48.55 ±0.24**	16.16	21.62 ±0.01	16.09 ±0.05	**23.38 ±0.12**
Elastic transform	26.33	33.09 ±0.08	18.88 ±0.24	**37.45 ±0.08**	16.04	17.17 ±0.01	17.75 ±0.01	**23.95 ±0.04**
Pixelate	21.98	26.47 ±0.09	14.59 ±0.30	**33.88 ±0.14**	28.00	33.56 ±1.71	30.03 ±0.05	**34.12 ±0.09**
JPEG compression	25.91	29.89 ±0.07	17.56 ±0.11	**36.07 ±0.32**	27.44	31.89 ±0.02	29.36 ±0.01	**32.23 ±0.12**
Average	29.43	35.19	22.08	**41.51**	20.61	23.17	21.32	**26.95**

alternative against a simple prompt averaging? In CLIPArTT, we hypothesize that combining the names of the most probable classes in a single text prompt better leverages CLIP's language understanding capabilities. Experiments conducted on CIFAR-10 and its variants are presented in Table 3.4, showing that our template, while simple, gives rise to an important accuracy gain with respect to prompt averaging.

Number of iterations. In line with our approach for TENT, we investigate the optimal number of adaptation iterations. As demonstrated in Table 3.3, iterating 10 times strikes the most favorable balance, exhibiting superior performance across the overall average on CIFAR-10-C

and yielding optimal results for numerous corruption types. Nonetheless, it is imperative to acknowledge that employing a lower number of iterations (e.g., Iter = 5) may yield improved outcomes in scenarios with minimal or no distribution shift, while a higher number of iterations (e.g., Iter = 20) may be more effective in mitigating severe distribution shifts. As a result, 10 iterations are used for all forthcoming experiments.

Pseudo-label selection for adaptation. For the target, we employed a linear combination of both image-to-image and text-to-text similarities to encapsulate the information derived from both modalities. This combination proved to be more effective than each modality separately (see Table 3.4). While selecting solely the text-to-text similarity for adaptation performs well, it falls short of achieving optimal results. Conversely, utilizing only the image-to-image similarity mainly improves for the original CIFAR-10 dataset and few corruption types.

Performance of the model for different batch-sizes. TTA methods have historically exhibited limitations when applied with small batch sizes. In our study, we investigate the performance of our model in this regard. As depicted in Table 3.5, performance improves significantly with increasing batch size. However, beyond a batch size of 8, no further performance gains are observed. This phenomenon can be attributed to the fact that, with smaller batch sizes, our method can potentially introduce uncertainty by using multiple classes. This leads to a lower performance, especially when the model is already confident. In subsequent experiments, we maintain a batch size of 128, consistent with prevailing practices in *state-of-the-art* methodologies.

3.5.2 Comparison on different datasets

In standard TTA, the concept of domain shift is strictly related to the source dataset used for pretraining. In the context of CLIP, the presence of domain shifts is less evident, as the source data contained an enormous amount of images that likely include different domains. For this reason, we categorize our experiments per dataset type.

Table 3.7 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-B/32 as visual encoder. The Shift column denotes a domain shift from standard images recognized by CLIP: corruption in CIFAR-10-C, CIFAR-100-C and ImageNet-C datasets, synthetic images in VisDA-C (3D)

	Shift	CLIP	LAME	TENT	TPT (BS=32)	TTDN	TDA	CALIP	CLIPArTT
CIFAR-10	✗	88.74	89.36 ±0.06	**91.69 ±0.10**	88.06 ±0.06	89.06 ±0.02	84.09 ±0.04	86.79 ±0.01	90.04 ±0.13
CIFAR-10.1	✗	83.25	81.22 ±0.33	**87.60 ±0.45**	81.80 ±0.27	83.77 ±0.06	78.98 ±0.37	81.02 ±0.03	86.35 ±0.27
CIFAR-100	✗	61.68	58.27 ±0.17	69.74 ±0.16	63.78 ±0.28	64.10 ±0.05	60.32 ±0.06	61.94 ±0.01	**69.79 ±0.04**
ImageNet	✗	61.81	61.17 ±0.02	62.13 ±0.02	60.74 ±0.16	61.85 ±0.02	**64.49 ±0.01**	58.50 ±0.00	61.39 ±0.09
VisDA-C (YT)	✗	84.45	84.72 ±0.01	84.41 ±0.02	82.95 ±0.01	83.29 ±0.03	84.47 ±0.08	**85.02 ±0.00**	83.46 ±0.03
PACS	✗	93.65	93.68 ±0.04	93.81 ±0.03	93.23 ±0.12	**94.77 ±0.07**	92.94 ±1.08	94.11 ±0.00	93.95 ±0.06
Office Home	✗	77.53	75.46 ±0.12	77.68 ±0.06	77.20 ±0.24	**79.76 ±0.04**	77.70 ±0.13	78.75 ±0.01	77.56 ±0.09
CIFAR-10-C	✓	59.22	50.57 ±0.30	67.56 ±0.19	56.80 ±0.22	60.13 ±0.07	48.00 ±0.68	56.58 ±0.0	**71.17 ±0.16**
CIFAR-100-C	✓	29.43	26.23 ±0.12	35.19 ±0.16	30.46 ±0.14	31.90 ±0.04	22.08 ±0.29	29.92 ±0.01	**41.51 ±0.15**
ImageNet-C	✓	20.61	20.02 ±0.16	23.17 ±0.54	21.10 ±0.06	21.28 ±0.02	21.32 ±0.06	21.00 ±0.00	**26.95 ±0.12**
VisDA-C (3D)	✓	84.51	84.75 ±0.17	83.85 ±0.03	78.45 ±0.05	80.45 ±0.01	84.22 ±0.10	83.81 ±0.08	**87.24 ±0.04**

Natural images. In the different Tables 3.7 and 3.8, CLIPArTT consistently enhances accuracy across CIFAR-10, CIFAR10.1, and CIFAR-100 datasets compared to the baseline (+2%, +3%, +8% respectively with ViT-B/32). This improvement suggests that uncorrupted datasets, which may not have been seen by the model during training, can benefit from adaptation. Consequently, a zero-shot model can leverage adaptation at test-time, relying solely on the model's predictions. However, while CLIPArTT outperforms the baseline, as well as LAME, TPT and TTDN, it is worth noting that TENT may yield superior results depending on the visual encoder and the number of classes. Entropy minimization used solely on confident results proves to be also effective. Interestingly, the corruptions in which TENT achieves a better performance tend to have a higher accuracy before adaptation, limiting the impact of error reinforcement on this approach. For a larger dataset such as ImageNet, most methods perform similarly, with the exception of TDA and CALIP, which improve CLIP performance by over 2%. Notably, both TDA and CALIP require hyperparameter tuning, and we used the recommended settings optimized for ImageNet.

Varied styles and textures. Next we compare methods on two datasets, PACS and OfficeHome, which contain images from very different domains and are often used as benchmark in domain generalization. Once again, these domains do not constitute real distribution shifts for CLIP

which was trained with a broad range of datasets. As can be seen, the best performance is achieved by methods such as TTDN for this setting. Nevertheless, it is important to highlight CLIPArTT's robustness, as it consistently maintains or improves model performance after adaptation. For instance, on the PACS dataset, CLIPArTT achieves an accuracy of 93.95%, compared to 93.65% for the original CLIP.

Table 3.8 Accuracy (%) on CIFAR-10, CIFAR-10.1, CIFAR-10-C, CIFAR-100 and CIFAR-100-C datasets with ViT-B/16 and ViT-L/14 as visual encoders

	Backbone	CLIP	TENT	CLIPArTT
CIFAR-10.1	ViT-B/16	84.00	88.52 ±0.33	**88.72 ±0.33**
	ViT-L/14	91.20	**92.22 ±0.25**	91.02 ±0.02
CIFAR-10-C	ViT-B/16	60.15	68.00 ±0.18	**73.22 ±0.17**
	ViT-L/14	76.04	**79.18 ±0.10**	78.06 ±0.15
CIFAR-100-C	ViT-B/16	32.01	37.90 ±0.18	**40.08 ±0.18**
	ViT-L/14	44.59	50.14 ±0.15	**52.52 ±0.18**

Common corruptions. In the various referenced tables, including Tables 3.7 and 3.8, CLIPArTT consistently outperforms compared approaches across all visual encoders, resulting in enhancements of up to 13% on average. A more detailed examination reveals that CLIPArTT frequently surpasses TENT, particularly in instances where the baseline performs poorly. Indeed, TENT tends to compromise the model's performance under conditions of low baseline confidence. For instance, in Table 3.6, TENT experiences a 3% decrease compared to the baseline under *Impulse Noise*, while CLIPArTT exhibits a 12% improvement. In Figure 3.3 (*right*), it is evident that CLIPArTT achieves better performance more rapidly across varying numbers of iterations. Conversely, when the baseline is confident, CLIPArTT consistently maintains a high performance. Finally, our method performs effectively on larger datasets with a greater number of classes, consistently improving upon the baseline by 6% on ImageNet-C.

Simulated images and video. The performance of out method on the YT dataset is suboptimal, a trend that can be observed in scenarios where the CLIP model already exhibits high confidence. However, our observations indicate that on the 3D dataset, CLIPArTT demonstrates a notable

competitive advantage over other Test-Time Adaptation (TTA) methods and the baseline, achieving an improvement of nearly 3%. Additionally, while TENT – often considered one of the more robust methods in handling various corruptions – results in a performance decline relative to the baseline, CLIPArTT maintains its effectiveness.

3.5.3 Limitations

Our results indicate that CLIPArTT performs particularly well under significant domain shifts such as corruptions, though it is slightly less effective on natural images. While the performance of our method in these cases is not as strong as that of TENT, it remains stable without deteriorating CLIP's capabilities, whereas the performance of TENT degrades under severe domain shifts. It is important to mention that any dataset not included in CLIP's pre-training can be considered a potential domain shift. However, natural images are more likely to have been seen by CLIP during pre-training, making adaptation for them prone to overfitting. This may explain why CLIPArTT excels in more challenging scenarios. To further explore the potential limitations of our method, we have included two additional scenarios in the supplementary material: *a)* evaluation in a highly imbalanced setting, where only C randomly chosen classes are present in a batch at random, and *b)* open-set classification, where out-of-distribution images (i.e., from classes not present in the text prompts) are introduced into the batches.

3.6 Conclusion

We introduce CLIPArTT, a novel Test-Time Adaptation framework designed specifically for VLMs. By leveraging the model's predictions as new pseudo-labels, we effectively minimize cross-entropy at test-time, thereby enhancing model performance even within a zero-shot setting.

A comprehensive ablation study determined optimal hyperparameters and helped gain deeper insights into the various model configurations. Our experimental results demonstrate that CLIPArTT achieves highly competitive performance across TTA datasets, surpassing *state-of-the-art* approaches. While TENT remains a strong competitor, our model offers enhanced

versatility by effectively addressing both natural and severe domain shifts, thus exhibiting robustness across various scenarios.

Exploring the potential of text prompts in classification is a promising avenue for future research. Investigating alternative methods to fine-tune these text prompts could yield valuable insights. Moreover, extending the study of Test-Time Adaptation to other scenarios, such as segmentation or object detection with Vision-Language Models (VLMs), holds significant promise for advancing our understanding of model adaptability and performance across diverse tasks.

CHAPTER 4

WATT: WEIGHT AVERAGE TEST-TIME ADAPTATION OF CLIP

David Osowiechi*, Mehrdad Noori*, Gustavo A. Vargas Hakim
Moslem Yazdanpanah, Ali Bahri, Milad Cheraghalikhani, Sahar Dastani
Farzad Beizaee, Ismail Ben Ayed, Christian Desrosiers

Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

*Equal Contribution
Poster at The Thirty-Eighth Annual Conference on Neural Information Processing Systems
(NeurIPS) December 2024

Abstract

Vision-Language Models (VLMs) such as CLIP have yielded unprecedented performances for zero-shot image classification, yet their generalization capability may still be seriously challenged when confronted to domain shifts. In response, we present Weight Average Test-Time Adaptation (WATT) of CLIP, a new approach facilitating full test-time adaptation (TTA) of this VLM. Our method employs a diverse set of templates for text prompts, augmenting the existing framework of CLIP. Predictions are utilized as pseudo labels for model updates, followed by weight averaging to consolidate the learned information globally. Furthermore, we introduce a text ensemble strategy, enhancing the overall test performance by aggregating diverse textual cues. Our findings underscore the effectiveness of WATT across diverse datasets, including CIFAR-10-C, CIFAR-10.1, CIFAR-100-C, VisDA-C, and several other challenging datasets, effectively covering a wide range of domain shifts. Notably, these enhancements are achieved without the need for additional model transformations or trainable modules. Moreover, compared to other TTA methods, our approach can operate effectively with just a single image. The code is available at: https://github.com/Mehrdad-Noori/WATT.

4.1 Introduction

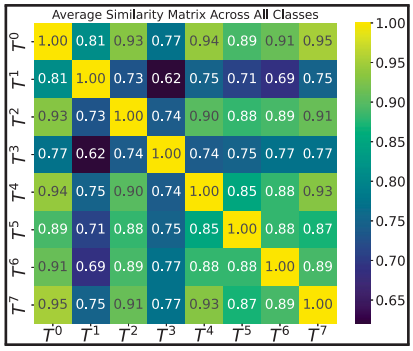
The integration of vision and language modalities into a unified learning framework, known as Vision Language models (VLM), has shown remarkable effectiveness in a broad range of vision-related tasks (Radford *et al.*, 2021; Jia *et al.*, 2021; Kirillov *et al.*, 2023). Notably, these models excel in zero-shot generalization scenarios, where they demonstrate proficiency in tasks beyond their original training scope, without requiring additional fine-tuning supervision. Applications of models like CLIP (Radford *et al.*, 2021) extend across diverse domains including video recognition (Lin *et al.*, 2022), audio processing (Guzhov *et al.*, 2022), and medical imaging (Liu *et al.*, 2023). These advancements underscore the pivotal role of such methods in shaping the trajectory of future research and applications in machine learning.

Despite its powerful capabilities, CLIP, like other traditional deep architectures such as Convolutional Neural Networks (CNNs), experiences performance degradation when confronted with domains it has not been trained on. Current research trends emphasize the importance of domain adaptation mechanisms in the deployment of CLIP (Lai *et al.*, 2023; Shu *et al.*, 2022). However, a significant challenge remains: swiftly and effectively adapting the model to new domains while preserving its attractive zero-shot capabilities, thus obviating the need for retraining.

To tackle this challenge, we investigate the impact of different text prompt templates, listed in Figure 4.1a, on model adaptation. Despite these templates being related, as reported in Figure 4.1b, the cosine similarity between their embeddings varies greatly, suggesting that they encode complementary information about the classes. The variability of information in different templates can also be observed in Figure 4.1c, where the classification accuracy obtained with these templates on a subset of CIFAR-10 corruptions fluctuates by up to 3%. Given this insight, finding an effective way to leverage the knowledge from different text templates would be useful to yield a better adaptation. This motivates our work proposing Weight Average adaptation during Test-Time (WATT).

	Template
T^0:	"a photo of a {class k}"
T^1:	"itap of a {class k}"
T^2:	"a bad photo of the {class k}"
T^3:	"a origami {class k}"
T^4:	"a photo of the large {class k}"
T^5:	"a {class k} in a video game"
T^6:	"art of the {class k}"
T^7:	"a photo of the small {class k}"

a)



b)

	T^0	T^1	T^2	T^3	T^4	T^5	T^6	T^7	WATT
Original	89.80	90.37	90.50	88.42	89.93	89.95	90.13	88.54	**91.05**
Gaussian Noise	60.19	61.01	61.17	58.24	58.84	58.35	59.62	61.13	**63.84**
Defocus Blur	77.23	77.07	78.00	75.98	76.39	77.45	77.08	75.59	**78.94**
Snow	76.57	77.36	77.93	75.08	77.45	77.09	77.05	75.57	**79.79**
JPEG Compression	64.65	65.36	65.24	64.16	64.18	64.36	64.78	65.32	**67.36**

c)

Figure 4.1 (a) The different templates used during our experiments. (b) Matrix of cosine similarity between each template, averaged over all classes of the CIFAR-10 dataset. (c) Comparison of accuracy (%) using cross-entropy (CE) on CIFAR-10 and some corruptions of CIFAR-10-C datasets using different templates and our weight average strategy

By strategically averaging the adapted weights derived from multiple text prompt templates, our method aims to harness the complementary strengths of individual templates, resulting in robust and enhanced performance across a wide range of domain shifts. To further illustrate this point, Figure 4.2 presents the test loss and adaptation error surfaces for three models that are separately adapted using three templates (T^0, T^1, T^2) under the Gaussian noise corruption of the CIFAR-10-C dataset[1]. The central point in these landscapes, representing the final model obtained by averaging the weights of the separate models, demonstrates a convergence towards

[1] To visualize the loss and error surface, we use weight vectors from models adapted with text templates T^0, T^1, and T^2, denoted as w_0, w_1, and w_2. Following (Garipov, Izmailov, Podoprikhin, Vetrov & Wilson, 2018), we define $u = w_1 - w_0$ and $v = (w_2 - w_0) - \frac{(w_2 - w_0)\cdot(w_1 - w_0)}{\|w_1 - w_0\|^2}(w_1 - w_0)$. The normalized vectors $\hat{u} = \frac{u}{\|u\|}$ and $\hat{v} = \frac{v}{\|v\|}$ form an orthonormal basis in the plane of w_0, w_1, and w_2. We create a Cartesian grid in this basis and evaluate the networks at each grid point. A point P with coordinates (x, y) in the plane is given by $P = w_0 + x \cdot \hat{u} + y \cdot \hat{v}$. To plot all in the same plane, we used the average of the three templates' text embeddings.

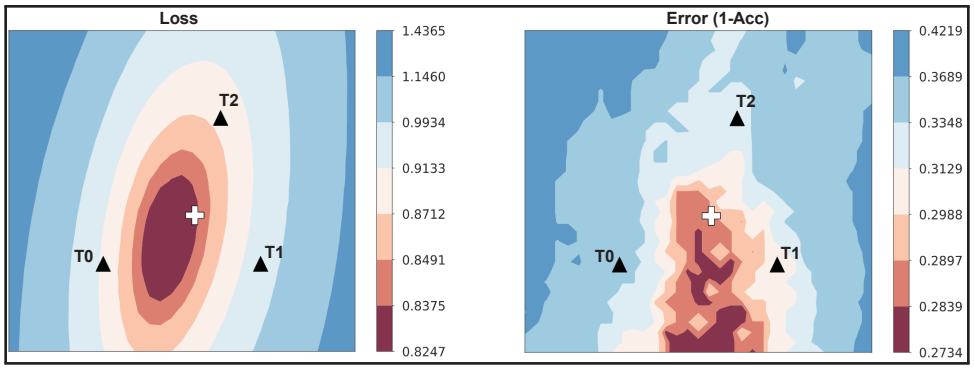


Figure 4.2 Loss and Error surfaces on model parameters for the Gaussian noise corruption of the CIFAR-10C dataset. Points T^0, T^1, and T^2 represent models adapted with different text templates (please see Fig. 4.1a). The central point (cross) shows the model obtained by averaging these weights, demonstrating improved performance

lower loss and error, highlighting the potential of weight averaging for test-time adaptation. Moreover, inspired by recent advancements in machine learning utilizing train-time weight averaging techniques (Cha *et al.*, 2021; Zhang *et al.*, 2024), the proposed WATT method can dynamically adjust to new data to tackle unforeseen distribution shifts without relying on class supervision.

We outline the main contributions of our work as follows:

- We introduce a novel Test-Time Adaptation method for CLIP, which leverages weight averaging across various text templates at test-time.
- Our WATT strategy yields highly competitive performances in comparisons to current TTA methods, and could bring improvement using only a single image at test time, a capability not present in previous approaches.
- We rigorously evaluate our WATT methodology through comprehensive evaluations across different datasets characterized by diverse types and degrees of domain shifts, encompassing **a total of 155 evaluation scenarios**. Our experiments demonstrate the robustness and efficacy of WATT compared to existing adaptation methods.

4.2 Related work

Test-Time Adaptation (TTA) is crucial in domain adaptation, particularly with unlabeled target domain data and no access to source domain samples. The challenge lies in estimating the target domain's distribution and comparing domain characteristics indirectly. Recent advancements have highlighted the potential and limitations of adapting pre-trained models. A key focus has been on leveraging batch normalization layers for adaptation due to their ability to retain source domain information. Methods such as PTBN (Nado *et al.*, 2021) and TENT (Wang *et al.*, 2020) recalibrate batch statistics and optimize affine parameters via entropy minimization, though they often require image augmentations or large batches. MEMO (Zhang *et al.*, 2022) proposes a simple approach that does not require multiple test points; it uses single-test-point data augmentations and minimizes the marginal entropy of the model's average output to enforce consistency and improve robustness. LAME (Boudiaf *et al.*, 2022) introduces a closed-form optimization strategy that refines model predictions for target images by leveraging the Laplacian of feature maps to encourage clustering, thereby emphasizing feature similarities. In another study, SAR (Niu *et al.*, 2023) addresses TTA instability by using batch-agnostic norms (e.g., group and layer norms) and a sharpness-aware entropy minimization approach. It filters noisy samples and guides the model to flat minima, improving robustness under mixed domain shifts and small batch sizes.

Recently, Test-Time Training (TTT) methodologies have emerged as prominent contenders in TTA (Sun *et al.*, 2020; Osowiechi *et al.*, 2023; Gandelsman *et al.*, 2022; Vargas Hakim *et al.*, 2023; Osowiechi *et al.*, 2024a). This approach involves training a supplementary sub-branch alongside the primary network in an unsupervised manner, subsequently leveraging it to refine the model. Unlike previous methods, our approach operates on individual image batches, offering a significant advantage in TTA by avoiding the necessity of training additional branches from scratch.

In natural language processing, TPT (Shu *et al.*, 2022) introduced entropy minimization for adapting models like CLIP, albeit with high computational costs due to learning an adapter at the text prompt with multiple transformations. DiffTPT (Feng *et al.*, 2023) extends this by

leveraging pre-trained diffusion models to generate diverse and informative augmented data, combining conventional augmentation methods used in TPT with diffusion-generated data to enhance adaptability. It also introduces a cosine similarity-based filtration technique for improved prediction fidelity. TDA (Karmanov *et al.*, 2024) offers a training-free approach with a dynamic adapter, utilizing a lightweight key-value cache and pseudo label refinement, making it computationally efficient. CLIPArTT (Vargas Hakim *et al.*, 2024), fine-tunes normalization layers with minimal disruption to the model's knowledge, enhancing text supervision by introducing pseudo labels. Existing methods often lag behind supervised prompt adaptation techniques in performance. SwapPrompt (Ma *et al.*, 2024) bridges this gap by leveraging self-supervised contrastive learning, employing a dual prompt paradigm. In comparison, our method combines prompt augmentation and fine-tuning of normalization layers, highlighting its effectiveness in test-time adaptation.

Weight Averaging (WA) is a powerful train-time technique for improving deep neural network generalization. Stochastic Weight Averaging (SWA) (Izmailov, Podoprikhin, Garipov, Vetrov & Wilson, 2018) averages weights of multiple models sampled from different training epochs, aiding smooth optimization trajectory and convergence to points with superior generalization. SWAD (Cha *et al.*, 2021) refines SWA by densely sampling weights throughout training, enhancing generalization and robustness across tasks. This train-time refinement enhances WA's effectiveness in producing models with improved generalization. The Lookaround (Zhang *et al.*, 2024) optimizer iterates between an "around step" and an "average step", building on SWAD's advancements. In the "around step", independently trained models using various data augmentations explore a broader loss landscape to find flatter minima. In the "average step," weights of these models are then averaged, guiding optimization towards lower loss regions. This method enhances robustness and generalization across tasks, improving upon SWA and SWAD by providing a more effective weight averaging process.

In contrast to existing approaches, WATT leverages varied text prompts to adapt vision-language models such as CLIP during testing. Our method also harnesses the benefits of weight averaging

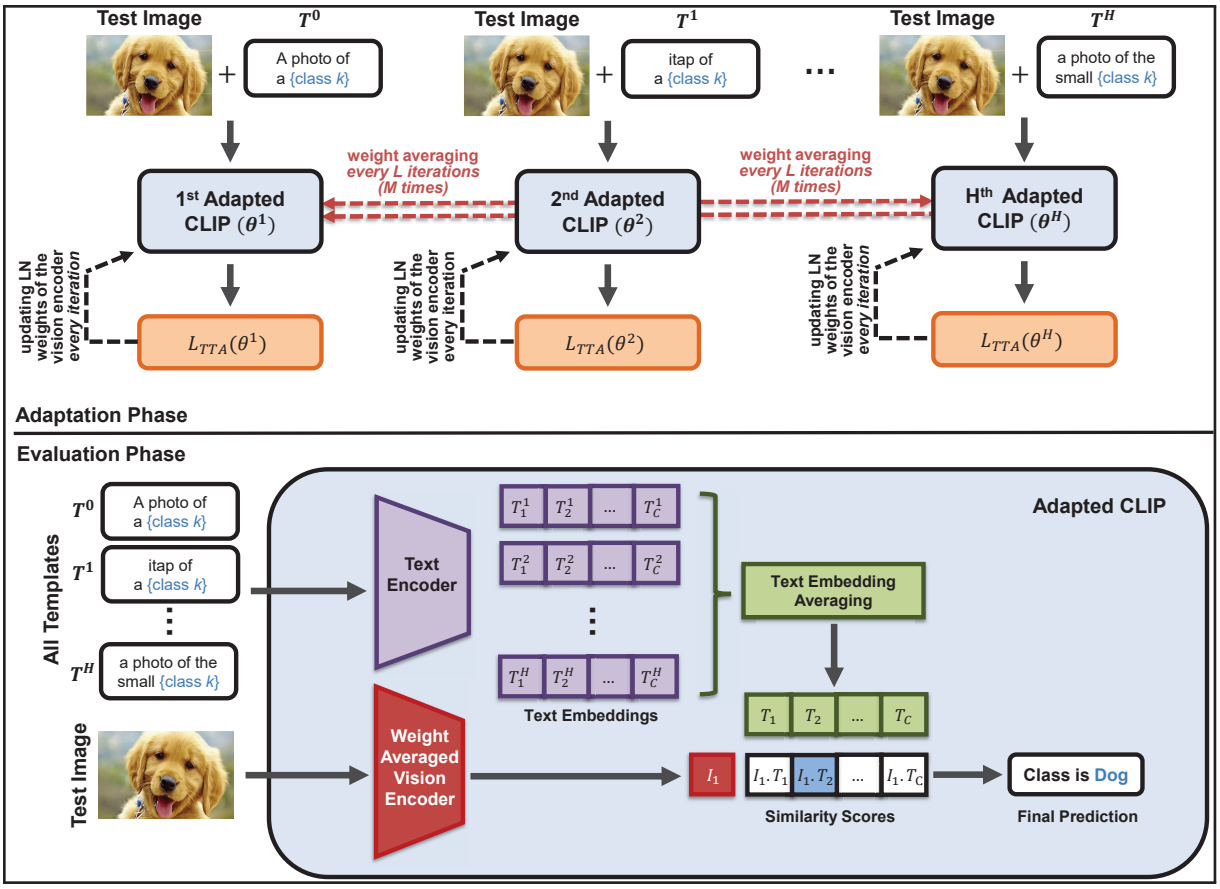


Figure 4.3 Overview of the proposed WATT method. In the Adaptation Phase, the model is adapted using different text templates (T^0, T^1, ..., T^H), with weight averaging performed periodically. In the Evaluation Phase, the adapted CLIP model uses averaged text embeddings from all templates and the weight averaged model to predict the class of the test image

while addressing domain shifts without additional model transformations or trainable modules, thereby setting a new precedent in test-time adaptation.

4.3 Method

The proposed WATT method, summarized visually in Figure 4.3 comprises three main components, the first two in the Adaptation Phase and the third in the Evaluation Phase: **1)** a light-weight transductive TTA strategy that adapts CLIP's visual encoder effectively by considering the similarity between *all* batch samples in terms of their visual and text features; **2)**

a weight-averaging strategy using multiple text templates to generate diverse model hypotheses during adaptation; **3)** an ensembling technique that boosts performance during evaluation by averaging the embedding of different text templates.

4.3.1 Transductive TTA loss

While our method can be employed with any TTA framework, in this work, we implement a strategy inspired by the transductive TTA approach of CLIPArTT (Vargas Hakim *et al.*, 2024) which effectively incorporates semantic relationships among batch samples.

Initially, our process involves executing inference using CLIP, a system comprising a visual encoder $f_\theta^v(\cdot)$ that translates an image $\mathbf{x}$ into visual features $\mathbf{z}^v \in \mathbb{R}^D$, and a text encoder $f_\theta^t(\cdot)$ which converts text prompts $\mathbf{t}$ into text features $\mathbf{z}^t \in \mathbb{R}^D$. During inference, we employ pre-defined text prompts assigned to each class within a dataset, such as $\mathbf{t}_k^0$ = "`a photo of a {class k}`". For a new image $\mathbf{x}_i$, the likelihood of belonging to class k is then computed using cosine similarity:

$$p_{ik} = \frac{\exp\left(\cos(\mathbf{z}_i^v, \mathbf{z}_k^t)/\tau\right)}{\sum_j \exp\left(\cos(\mathbf{z}_i^v, \mathbf{z}_j^t)/\tau\right)}, \quad \cos(\mathbf{z}, \mathbf{z}') = \frac{\mathbf{z}^\top \mathbf{z}'}{\|\mathbf{z}\|_2 \cdot \|\mathbf{z}'\|_2}, \tag{4.1}$$

where τ is a softmax temperature parameter set to 0.01 in this work. This prediction is then stored to be used as pseudo labels for the model.

Denoting the normalized visual embeddings of the samples within the test batch as $\mathbf{Z}^v \in \mathbb{R}^{B\times D}$ and the instance-specific text embeddings as $\mathbf{Z}^t \in \mathbb{R}^{B\times D}$, we compute an image-to-image similarity matrix $\mathbf{S}^v = \mathbf{Z}^v(\mathbf{Z}^v)^\top \in [-1,1]^{B\times B}$ modeling pairwise relationships in terms of image characteristics. Similarly, we construct a text-to-text similarity matrix $\mathbf{S}^t = \mathbf{Z}^t(\mathbf{Z}^t)^\top \in [-1,1]^{B\times B}$, capturing the semantic relationships among text embeddings within the batch. Utilizing the computed pairwise similarity matrices, we generate pseudo labels $\mathbf{Q} = \text{softmax}\big((\mathbf{S}^v + \mathbf{S}^t)/2\tau\big) \in [0,1]^{B\times B}$ which are used with cross-entropy in our transductive

Table 4.1 Accuracy (%) with different text ensembles at test time

Dataset	single_temp	text_avg
CIFAR-10	90.87 ±0.10	**91.08** ±0.06
CIFAR-10.1	86.80 ±0.19	**86.85** ±0.18
CIFAR-10-C	72.08	**72.66**
CIFAR-100	69.79 ±0.20	**70.30** ±0.11
CIFAR-100-C	41.79	**42.24**

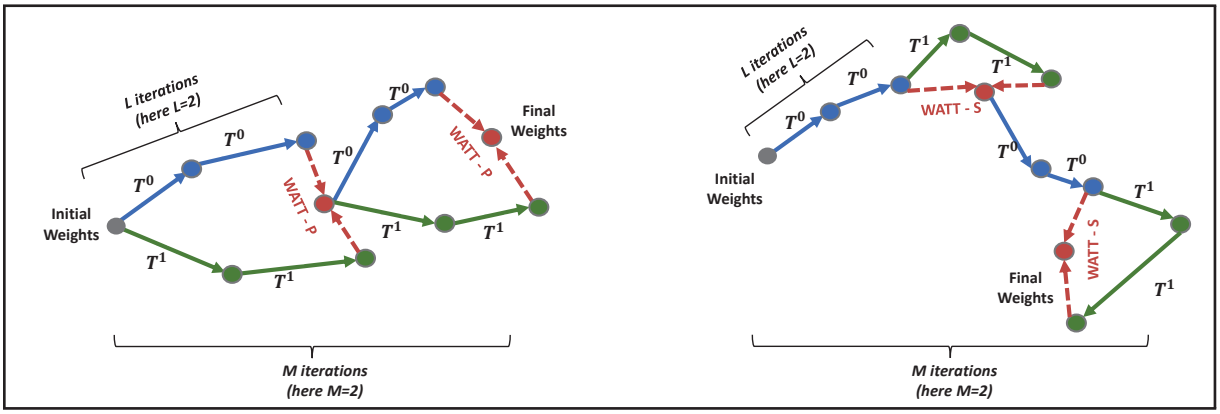


Figure 4.4 Visual comparison of the Parallel **(left)** and Sequential **(right)** approaches for multi-template weight averaging during adaptation

TTA loss:

$$\mathcal{L}_{\text{TTA}}(\theta) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} q_{ij}\log p_{ij}. \tag{4.2}$$

Drawing a link with the Stochastic Neighbor Embedding (SNE) method for dimensionality reduction (Hinton & Roweis, 2002), which minimizes the KL divergence between distributions modeling pairwise distances, our TTA loss ensures that the inter-modality (text-to-image) similarities of batch samples are aligned with their intra-modality ones (text-to-text and image-to-image).

4.3.2 Multi-Template Weight Averaging

We explore various text prompt templates suggested in the CLIP paper and detailed in Fig. 4.1a. As reported in Table 4.1c, these prompts achieve varying performance across different corruption

types of CIFAR-10-C. We formulate prompts of the form $\mathbf{t}_k^h$ = `template h(class k)`, where $h \in \{1, 2, \ldots, H\}$, encompassing a spectrum of textual cues tailored to elicit diverse responses from the model.

Two different approaches are investigated for our multi-template weight averaging (MTWA) strategy. The first one denoted as **Parallel MTWA (WATT-P)**, which follows recent optimization approaches like Lookaround (Zhang *et al.*, 2024), performs the adaptation separately for each text template, starting from the same parameters, and then averages the resulting adapted weights. The second one, called **Sequential MTWA (WATT-S)**, instead considers text templates sequentially without resetting the weights. These two approaches, which we illustrate and compare in Fig. 4.4, are detailed below.

Parallel MTWA. This approach optimizes the TTA loss in (4.2) separately for H different models, each utilizing a distinct template. Starting from the same visual encoder parameters θ, these models are updated in parallel for L iterations, resulting in updated parameters θ'_h, with $h \in \{1, \ldots, H\}$. The parameters are reset after each update, enabling each model to restart the adaptation from the same initial point. Subsequently, we aggregate the weights obtained from these H models by computing their average: $\theta_{\text{avg}} = \frac{1}{H} \sum_{h=1}^{H} \theta'_h$. We repeat this step M times, and denote the overall process as "(after L iter) $\times$ M".

Sequential MTWA. Our Sequential MTWA approach is inspired from the work of (Izmailov *et al.*, 2018), where the averaging of weights across various stages of the training process is employed to mitigate variance and enhance generalization capabilities. Instead of resetting parameters for each model, we update parameters after each template's iteration. To ensure impartiality in the update sequence of templates, a random selection process is implemented, thereby disregarding any predetermined order.

4.3.3 Evaluation Phase

At test-time, predictions are computed using Equation 4.1 through two distinct methodologies. In the first approach, the text features $\mathbf{z}_0^t$ are derived from the initial text prompt $\mathbf{t}_k^0$ = "`a photo`

Table 4.2 Accuracy (%) of our method for different batch sizes compared to CLIP

Dataset	CLIP	BS = 1	BS = 2	BS = 4	BS = 8	BS = 16	BS = 32	BS = 64	BS = 128
CIFAR-10	88.74	89.87	89.39 ±0.02	89.16 ±0.07	88.93 ±0.16	89.14 ±0.04	89.51 ±0.12	90.16 ±0.13	91.05 ±0.06
CIFAR-10.1	83.25	84.55	84.32 ±0.15	83.88 ±0.17	84.12 ±0.37	84.35 ±0.21	84.87 ±0.16	85.52 ±0.30	86.98 ±0.31
CIFAR-10-C	59.22	61.26	63.60	63.47	63.94	65.66	68.34	71.21	73.82

`of a class` k", denoted as single_temp. Conversely, the second method aggregates the text features from all templates by computing their mean, resulting in the prediction $\mathbf{z}^t_{\text{ens}} = \frac{1}{H}\sum_{h=1}^{H}\mathbf{z}^t_h$, denoted as text_avg (see Fig. 4.3).

4.4 Experimental Setup

Settings. In line with prior TTA methodologies, adjustments are made to all Layer Normalization layers within the visual feature extractor for test-time adaptation. The Adam optimizer is employed with a fixed learning rate of 10^{-3}, wheras a smaller learning rate of 10^{-4} is chosen for adaptation to the 3D renderings split, as it reflects a more pronounced shift. Throughout our experimentation process, a consistent batch size of 128 is maintained to ensure uniformity and facilitate meaningful comparisons across various scenarios.

Datasets. Following (Vargas Hakim *et al.*, 2024), we rigorously evaluate WATT's performance across diverse TTA datasets using established assessment techniques. These datasets simulate intricate domain shifts, providing nuanced insights into our approach's effectiveness. Additionally, we explore WATT's adaptability on the original dataset through zero-shot test-time adaptation. To ensure a thorough examination, we extend our analysis to include various domain generalization datasets, exposing our method to a broad spectrum of image categories for comprehensive evaluation. Our evaluation framework encompasses *natural images*, *common corruptions*, *simulated shifts*, *video shifts*, *texture shifts* and *style shifts*.

In our assessment of natural image analysis, we include CIFAR-10, CIFAR-10.1, and CIFAR-100, each comprising 10,000 images and offering varied data distributions. CIFAR-10.1 (Hendrycks & Dietterich, 2019) introduces a natural shift from CIFAR-10, providing a

comprehensive evaluation of our model's performance. We also incorporate the CIFAR-10-C and CIFAR-100-C datasets (Hendrycks & Dietterich, 2019), augmented with 15 distinct corruptions across 5 severity levels, resulting in 75 common corruption scenarios. This comprehensive augmentation assesses the model's resilience effectively.

Our investigation also extends to the VisDA-C dataset (Peng *et al.*, 2018), challenging models with simulated and video shifts across diverse imagery types. Additionally, we evaluate our method on three datasets mostly used in the field of domain generalization: PACS (Li, Yang, Song & Hospedales, 2017), VLCS (Fang, Xu & Rockmore, 2013), and OfficeHome (Venkateswara, Eusebio, Chakraborty & Panchanathan, 2017) datasets, instrumental in understanding texture and style variations. These evaluations effectively demonstrate the generalizability of our method across distinct domain shifts.

Benchmarking. We conduct a comparative analysis of WATT against contemporary methods using ViT-B/32 as the backbone. Specifically, we incorporate an adapted version of TENT (Wang *et al.*, 2020), customized for CLIP by the authors of (Vargas Hakim *et al.*, 2024), using 10 iterations. We also include SAR (Niu *et al.*, 2023) and MEMO (Zhang *et al.*, 2022) in the same manner as TENT. Additionally, we compare with TPT (Shu *et al.*, 2022), a novel adaptation technique for CLIP that heavily relies on image augmentations, as well as DiffTPT (Feng *et al.*, 2023), an extension that incorporates diffusion models. Lastly, we include TDA (Karmanov *et al.*, 2024), which employs a dynamic adapter, and CLIPArTT (Vargas Hakim *et al.*, 2024), a recent approach that utilizes pseudo labels generated through conformal learning.

4.5 Results

In this section, we present empirical findings from our WATT method through a series of ablation studies aimed at understanding the impacts of individual components. These studies inform subsequent experiments across diverse datasets. Leveraging insights from ablations, we conduct comprehensive experiments, benchmarking WATT against state-of-the-art techniques across various datasets.

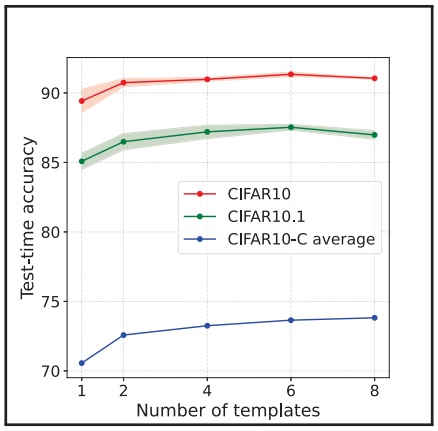


Figure 4.5 Evolution of the accuracy for different numbers of random template on 5 test-time runs

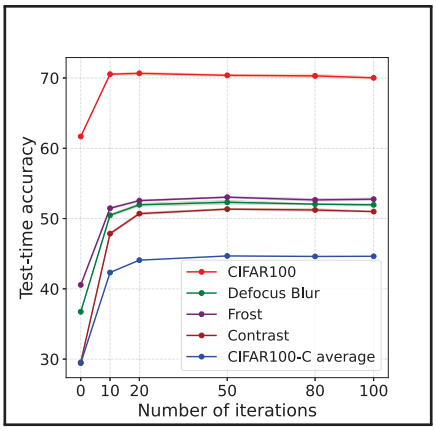


Figure 4.6 Evolution of accuracy on CIFAR-100 corruptions with the Parallel MTWA method

4.5.1 Ablation Studies

In this section, unless otherwise specified, we focus on the Sequential MTWA variant of our method (see Section 4.3.2) and will use these findings as a reference for the Parallel MTWA method.

Comparison of the template used during testing. After updating the model, we proceed to compute the similarity between the image features and the text embeddings, enabling prediction. Typically, text embeddings originate from the text prompt "`a photo of a class` k". However, by employing multiple templates, we have the flexibility to alter this text embedding through the averaging of all text embeddings from each template. Table 4.1 conducts a comparative analysis revealing that this averaged text embedding consistently yields superior results across all scenarios. Hence, we adopt this approach for next experiments.

Comparison of the number of templates. In Figure 4.5, we examine the performance variation relative to the number of utilized templates. In this investigation, we conduct 5 runs wherein the templates are randomly selected from a pool of 8 distinct templates (as outlined in Fig. 4.1a). Notably, when the distribution shift is minimal, as observed in CIFAR-10 and CIFAR-10.1, optimal performance is attained using 6 templates, with performance gradually

diminishing thereafter. Conversely, in scenarios characterized by substantial corruptions, such as CIFAR-10-C, employing all 8 templates proves advantageous. Consequently, our focus moving forward will be on utilizing all 8 templates in our work.

Table 4.3 Accuracy (%) obtained with different averaging strategies

Dataset	Text avg.	Output avg.	Weight avg. (ours)		
			(after 10 iter)×1	(after 1 iter)×10	(after 2 iter)×5
CIFAR-10	90.58 ±0.03	90.90 ±0.03	91.08 ±0.06	**91.39 ±0.14**	91.05 ±0.06
CIFAR-10.1	85.78 ±0.25	86.77 ±0.08	86.85 ±0.18	**88.02 ±0.18**	86.98 ±0.31
CIFAR-10-C	71.41	72.60	72.66	73.66	**73.82**
CIFAR-100	69.46 ±0.13	70.32 ±0.1	70.3 ±0.11	**70.85 ±0.08**	70.74 ±0.20
CIFAR-100-C	41.37	42.68	42.24	45.32	**45.57**

Text Averaging vs Output Averaging vs Weight Averaging. Utilizing the averaging method within a VLM offers several possibilities, including averaging the weights, the outputs or the text embeddings before computing the logits. In Table 4.3, a comparison between these approaches is presented. It becomes evident that weight averaging consistently outperforms text embedding averaging across various datasets, showcasing a superiority of approximately 1% even with the less effective weight averaging method. This performance advantage is observed across CIFAR-10, CIFAR-10.1, and CIFAR-10-C, and persists even with larger numbers of classes, such as in CIFAR-100 and CIFAR-100-C. When concentrating on output averaging, the results may be less evident with less effective weight averaging methods. However, they remain valid and even more accurate with superior weight averaging techniques. Therefore, our focus for future experiments will be on weight averaging as the preferred approach.

Best moment to do the Weight Averaging. Examining Table 4.3, it is evident that the parameters L and M discussed in Section 4.3 are crucial. Specifically, a large L (e.g., 10) combined with a small M (e.g., 1) is ineffective. Conversely, setting $L = 1$ and $M = 10$ yields optimal results for small distribution shift datasets, while $L = 2$ and $M = 5$ perform best on highly corrupted datasets. Given that TTA typically encounters substantial distribution shifts, we will use $L = 2$ and $M = 5$ in our subsequent experiments.

Performance over the number of iterations. In this section, we focus on the method incorporating a Parallel MTWA mechanism and examine the impact of the number of iterations on performance. As illustrated in Figure 4.6, the accuracy stabilizes after approximately 20 iterations. Although there is a slight improvement in performance beyond 50 iterations, the difference is marginal. Based on these observations, we have opted to use 50 iterations for our experiments.

Model performance across various batch sizes. In our investigation, we delve into the performance implications of TTA methods when operating under small batch sizes, a historical challenge in the field. Table 4.2 provides insights into this aspect, revealing substantial performance enhancements with increasing batch sizes. Notably, our WATT model showcases remarkable adaptability, demonstrating performance improvements even with a single image input contrary to alternative methods. Specifically, we observe enhancements of approximately 1% for CIFAR-10 and CIFAR-10.1, and an impressive 2% for CIFAR-10-C when compared to baseline. Moving forward, we maintain a batch size of 128 in our experiments, aligning with prevalent practices observed in contemporary state-of-the-art methodologies.

4.5.2 Comparison to SOTA methods

Performance evaluation with small batches. In existing TTA works that study small batch size scenarios, such as SAR (Niu *et al.*, 2023) and MEMO (Zhang *et al.*, 2022), we implement these methods in the context of CLIP. We compare our method with other TTA approaches using a batch size of 1. As shown in Table 4.5, WATT-P achieves the highest accuracy across all cases, outperforming SAR by 2.23% and MEMO by 0.75% on CIFAR-10.1. Notably, this improvement is achieved without any image augmentation, which is a common practice in previous TTA approaches that deal with small batches.

Performance evaluation in the presence of natural or no domain shift. In Table 4.4, results show consistent performance enhancements with WATT, both with the Parallel and Sequential MTWA strategies, alongside the baseline. On CIFAR-10, performance improves by 2.67% with

Table 4.4 Accuracy (%) on CIFAR-10, CIFAR-10.1, CIFAR-10-C, CIFAR-100, and CIFAR-100-C datasets. WATT-P refers to our method with Parallel MTWA and WATT-S to the Sequential MTWA variant of WATT

	Dataset	CLIP	TENT	TPT	TDA	DiffTPT	SAR	CLIPArTT	WATT-P	WATT-S
	CIFAR-10	88.74	**91.69** ±0.10	88.06 ±0.06	84.09 ±0.04	83.07 ±0.05	89.05 ±0.02	90.04 ±0.13	91.41 ±0.17	91.05 ±0.06
	CIFAR-10.1	83.25	87.60 ±0.45	81.80 ±0.27	78.98 ±0.37	76.50 ±0.29	83.65 ±0.04	86.35 ±0.27	**87.78** ±0.05	86.98 ±0.31
	CIFAR-10-C	59.22	67.56	56.80	48.00	56.77	60.45	71.17	72.83	**73.82**
	CIFAR-100	61.68	69.74 ±0.16	63.78 ±0.28	60.32 ±0.06	52.80 ±0.08	64.44 ±0.01	69.79 ±0.04	70.38 ±0.14	**70.74** ±0.20
CIFAR-100-C	Gaussian Noise	14.80	14.38 ±0.14	14.03 ±0.10	8.20 ±0.35	21.40 ±0.08	15.85 ±0.03	25.32 ±0.14	31.28 ±0.03	**32.07** ±0.23
	Shot noise	16.03	17.34 ±0.27	15.25 ±0.17	9.58 ±0.43	24.17 ±0.49	17.41 ±0.05	27.90 ±0.05	33.44 ±0.11	**34.36** ±0.11
	Impulse Noise	13.85	10.03 ±0.13	13.01 ±0.13	7.63 ±0.19	16.87 ±0.24	14.90 ±0.09	25.62 ±0.09	29.40 ±0.11	**30.33** ±0.03
	Defocus blur	36.74	49.05 ±0.07	37.60 ±0.17	25.59 ±0.41	20.30 ±0.29	42.00 ±0.06	49.88 ±0.23	52.32 ±0.28	**52.99** ±0.16
	Glass blur	14.19	3.71 ±0.07	16.41 ±0.02	9.83 ±0.56	15.57 ±0.46	13.84 ±0.08	27.89 ±0.03	31.20 ±0.12	**32.15** ±0.30
	Motion blur	36.14	46.62 ±0.27	37.52 ±0.23	28.92 ±0.18	21.00 ±0.64	39.52 ±0.01	47.93 ±0.14	49.72 ±0.15	**50.53** ±0.12
	Zoom blur	40.24	51.84 ±0.15	42.99 ±0.11	31.08 ±0.36	25.53 ±0.05	45.40 ±0.05	52.70 ±0.06	54.72 ±0.04	**55.30** ±0.22
	Snow	38.95	46.71 ±0.21	42.35 ±0.13	32.94 ±0.12	28.83 ±0.37	41.85 ±0.08	49.72 ±0.01	51.79 ±0.04	**52.77** ±0.15
	Frost	40.56	44.90 ±0.27	43.31 ±0.14	34.84 ±0.25	31.10 ±0.36	42.20 ±0.04	49.63 ±0.12	53.04 ±0.08	**53.79** ±0.31
	Fog	38.00	47.31 ±0.04	38.81 ±0.17	31.13 ±0.15	16.60 ±0.43	40.14 ±0.00	48.77 ±0.04	50.78 ±0.24	**51.49** ±0.21
	Brightness	48.18	60.58 ±0.18	50.23 ±0.11	42.36 ±0.10	38.13 ±0.29	52.77 ±0.10	61.27 ±0.08	62.65 ±0.25	**63.57** ±0.21
	Contrast	29.53	45.90 ±0.11	28.09 ±0.09	18.03 ±0.07	7.70 ±0.22	34.40 ±0.10	48.55 ±0.24	51.34 ±0.10	**52.76** ±0.27
	Elastic transform	26.33	33.09 ±0.08	28.12 ±0.15	18.88 ±0.24	21.60 ±0.51	28.44 ±0.07	37.45 ±0.08	39.97 ±0.06	**40.90** ±0.43
	Pixelate	21.98	26.47 ±0.09	20.43 ±0.14	14.59 ±0.30	22.83 ±0.31	22.91 ±0.07	33.88 ±0.14	39.59 ±0.09	**40.97** ±0.16
	JPEG compression	25.91	29.89 ±0.07	28.82 ±0.09	17.56 ±0.11	31.77 ±0.45	27.20 ±0.06	36.07 ±0.32	38.99 ±0.16	**39.59** ±0.08
	Mean	29.43	35.19	30.46	22.08	22.89	31.92	41.51	44.68	**45.57**

Table 4.5 Comparison of different TTA methods with a batch size equal to 1

Dataset	CLIP	TPT	SAR	MEMO	CLIPArTT	WATT-P
CIFAR-10	88.74	88.29	87.41	89.29	88.76	89.87
CIFAR 10.1	83.25	82.85	82.32	83.80	83.15	84.55
CIFAR-10-C	59.22	59.03	58.70	61.15	59.18	61.26

Parallel MTWA and 2.31% using Sequential MTWA. On CIFAR-10.1, improvements reach 4.53% and 3.73%, and on CIFAR-100, enhancements are 8.70% and 9.06%. While WATT consistently outperforms the baseline, TPT, and CLIPArTT, TENT yields superior results on CIFAR-10. WATT's effectiveness often correlates with the number of classes, showing better performance with more classes, indicating its strength in lower-confidence scenarios.

Performance evaluation in the presence of common corruptions. Table 4.4 shows that both WATT variants consistently outperform alternative methods across various corruptions

and class numbers. Notably, WATT with Parallel MTWA improves performance by 16.48% on CIFAR-100 *Gaussian Noise* and by 17.01% on *Glass Blur* compared to the baseline, while WATT with Sequential MTWA shows improvements of 17.27% and 17.96% respectively. On common corruptions, the Sequential MTWA variant surpasses Parallel MTWA, with improvements of 0.99% on CIFAR-10 and 0.89% on CIFAR-100. According to the TDA supplementary materials, we selected the weighting factor alpha as 5.0 and the sharpness ratio beta as 2.0, which are stated as optimal. However, these values did not appear to be the best choice for more challenging datasets like CIFAR-10-C or CIFAR-100-C that contain various corruptions. Adjusting these parameters based on the dataset would not be consistent with the principles of a fully TTA method, which might explain their suboptimal performance in our results. Regarding DiffTPT, it generates 64 images per test image, making it challenging to use in a real-world TTA scenario. Similar to TDA, DiffTPT requires carefully chosen parameters to fit the dataset, whereas our method does not require dataset-specific tuning. This highlights the robustness and practicality of our approach in diverse real-world applications.

Performance analysis under simulated and video shifts. Results on the 3D (simulated shift) and YT (video shift) splits of VisDA-C demonstrate a significant improvement in accuracy with our proposed WATT method compared to pure CLIP. The Sequential MTWA variant achieves the highest accuracy on both the 3D and YT splits, with scores of 85.36% and 84.69%, respectively, surpassing other adaptation methods including TENT, TPT, and CLIPArTT (see Table 4.6).

Performance analysis under texture and style shifts. Results on the OfficeHome, PACS, and VLCS datasets are presented in Table 4.6. On average, our proposed WATT method, with Parallel and Sequential MTWA variants, improves performance across the different domains of OfficeHome, PACS, and VLCS compared to other methods. This highlights its robustness in addressing texture and style shifts, which are especially challenging compared to other domain shift variants.

Table 4.6 Accuracy (%) on different domains of VisDA-C, OfficeHome, PACS and VLCS datasets

Dataset	Domain	CLIP	TENT	TPT	CLIPArTT	WATT-P	WATT-S
VisDA-C	3D (trainset)	84.43	84.86 ±0.01	79.35 ±0.04	85.09 ±0.01	**85.42 ±0.03**	85.36 ±0.01
	YT (valset)	84.45	84.68 ±0.01	83.57 ±0.04	84.40 ±0.01	84.57 ±0.00	**84.69 ±0.01**
	Mean	84.44	84.77	81.46	84.75	85.00	**85.03**
OfficeHome	Art	73.75	74.03 ±0.27	**75.76 ±0.27**	73.84 ±0.20	75.65 ±0.27	**75.76 ±0.39**
	Clipart	63.33	63.42 ±0.04	63.08 ±0.31	63.54 ±0.06	**66.23 ±0.13**	65.77 ±0.11
	Product	85.32	**85.51 ±0.08**	84.07 ±0.28	85.23 ±0.16	85.41 ±0.09	85.41 ±0.01
	Real World	87.71	87.74 ±0.05	85.89 ±0.33	87.61 ±0.05	88.22 ±0.15	**88.37 ±0.05**
	Mean	77.53	77.68	77.20	77.56	**78.88**	78.83
PACS	Art	96.34	**96.65 ±0.05**	95.52 ±0.20	96.57 ±0.09	96.31 ±0.00	96.39 ±0.00
	Cartoon	96.08	96.22 ±0.05	94.77 ±0.20	96.00 ±0.02	96.52 ±0.02	**96.62 ±0.02**
	Photo	99.34	99.40 ±0.00	99.42 ±0.06	99.28 ±0.00	99.48 ±0.03	**99.52 ±0.00**
	Sketch	82.85	82.96 ±0.12	83.22 ±0.14	83.93 ±0.14	**86.92 ±0.04**	86.65 ±0.12
	Mean	93.65	93.81	93.23	93.95	**94.81**	94.80
VLCS	Caltech101	**99.51**	**99.51 ±0.00**	99.36 ±0.06	**99.51 ±0.00**	99.43 ±0.00	**99.51 ±0.00**
	LabelMe	68.15	67.89 ±0.13	54.88 ±0.12	67.96 ±0.04	66.67 ±0.21	**68.49 ±0.12**
	SUN09	68.85	69.27 ±0.04	67.30 ±0.49	68.68 ±0.09	72.61 ±0.15	**73.13 ±0.17**
	VOC2007	84.13	**84.42 ±0.15**	76.74 ±0.28	84.09 ±0.02	82.30 ±0.16	83.41 ±0.17
	Mean	80.16	80.27	74.57	80.06	80.25	**81.14**

4.6 Conclusion

We introduce WATT, a Test-Time Adaptation method tailored for Vision-Language Models. Our approach harnesses Weight Averaging with different text prompts and incorporates text embeddings averaging to bolster prediction accuracy.

Through an extensive ablation study, we scrutinized the efficacy of employing varied text prompts and weight averaging. Comparative evaluations across Test-Time Adaptation and Domain Generalization datasets underscored the superiority of our method, particularly in scenarios involving distribution shifts and zero-shot performance enhancements compared to state-of-the-art approaches.

Looking forward, investigating the potential of text prompts and weight averaging in classification opens up promising avenues for future exploration. Our methodology, with its focus on template manipulation, suggests potential avenues for extension, such as incorporating alternative class

descriptors, yielding valuable insights for future research. Moreover, expanding Test-Time Adaptation to encompass diverse scenarios, including segmentation or object detection with Vision-Language Models, holds significant potential for advancing our comprehension of model adaptability and performance across varied tasks.

CONCLUSION AND RECOMMENDATIONS

Throughout this thesis, we have explored the challenge of adapting vision models to distribution shifts that arise at test time, under constraints where neither labeled target data nor source domain data are accessible. Importantly, our focus extended to scenarios where only a very small amount of test data is available at any given time — reflecting realistic conditions faced during online deployment. Motivated by the limitations of traditional training-time solutions and by the increasing deployment of large pre-trained models across real-world environments, we proposed a set of strategies focused on enhancing test-time adaptability — with a particular emphasis on methods that operate during inference, without reliance on supervision, prior assumptions about the shift, or access to the original source data.

Rather than aiming for a one-size-fits-all solution, we developed model-specific adaptation techniques, each tailored to a particular class of architectures — from convolutional neural networks to vision-language models. These approaches are designed to be efficient, practical, and complementary to the strengths and inductive biases of their target models, enabling robust adaptation without retraining or architectural changes.

We summarize the main insights of this work in the following three takeaways:

First, we show that both Test-Time Training and Test-Time Adaptation, when properly designed, can significantly improve model performance under distribution shift. Our contribution NC-TTT falls under the Test-Time Training paradigm: it requires preparing an auxiliary contrastive task during training to enable adaptation at inference. While effective, TTT frameworks like NC-TTT inherently assume that potential distribution shifts are anticipated during training, making them less flexible in unforeseen deployment scenarios. In contrast, our contributions CLIPArTT and WATT belong to the Test-Time Adaptation paradigm: they operate without any modification at training time, adapting purely based on target data observed during inference. Importantly, we also demonstrate that TTA strategies can improve performance even in the

absence of explicit domain shift, by enabling models to specialize and calibrate dynamically to their specific deployment conditions.

The second contribution of this thesis has been to surface the limitations and fragility of TTA methods. While powerful, these methods are not universally stable; they depend heavily on optimization dynamics, pseudo-label reliability, and the characteristics of the shift itself. Our work highlights the importance of thoughtful design — including careful objective selection, filtering mechanisms, and regularization — to avoid degeneracies during adaptation.

Third, and more broadly, this thesis advocates for a transition from training-centered to inference-centered paradigms in machine learning. As foundation models and large pre-trained architectures become the norm, retraining models for every new distribution is becoming impractical. While Test-Time Training offers improvements, it remains constrained by training-time assumptions. Test-Time Adaptation, on the other hand, offers a more future-proof, modular, and scalable solution: adapting pre-trained models dynamically during inference, without requiring access to training data or model retraining. TTA opens the way for self-improving, context-aware AI systems — systems that continuously specialize, calibrate, and improve post-deployment, even under constraints of privacy, storage, and time.

In this light, TTA is not merely a reactive strategy — it is a proactive and indispensable evolution of model deployment. By treating adaptation as an inference-time capacity, rather than a retraining step, we move toward AI systems that are continually evolving, more resilient to the unknown, and ultimately better suited to the complexity of real-world environments. We hope that the methods and insights presented in this thesis will help pave the way toward a new generation of adaptive, lifelong learning AI systems — systems that not only learn during training but continue to learn and adapt whenever and wherever they are deployed.

APPENDIX I

NC-TTT: A NOISE CONSTRASTIVE APPROACH FOR TEST-TIME TRAINING - SUPPLEMENTARY MATERIAL

David Osowiechi*, Gustavo A. Vargas Hakim*
Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah,
Ismail Ben Ayed, Christian Desrosiers

Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

*Equal Contribution
Spotlight at Computer Vision and Pattern Recognition (CVPR) June 2024

1. Deriving the posterior of Equation (6)

We start with the definition of the posterior in Eq. (5):

$$p(y_s = 1 \,|\, \widetilde{\mathbf{z}}_{i,m}) = \frac{\sigma_s^{-D} \exp\left(-\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}\|^2\right)}{\sigma_s^{-D} \exp\left(-\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}\|^2\right) + \sigma_o^{-D} \exp\left(-\frac{1}{2\sigma_o^2}\|\boldsymbol{\epsilon}\|^2\right)}, \tag{A I-1}$$

where $\boldsymbol{\epsilon} = \widetilde{\mathbf{z}}_i - \mathbf{z}_i$. Multiplying the numerator and denominator by $\sigma_s^D \exp\left(\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}\|^2\right)$ gives

$$p(y_s = 1 \,|\, \widetilde{\mathbf{z}}_{i,m}) = \frac{1}{1 + \left(\sigma_s/\sigma_o\right)^D \exp\left(-\frac{1}{2}(1/\sigma_s^2 - 1/\sigma_o^2)\|\boldsymbol{\epsilon}\|^2\right)} \tag{A I-2}$$

We then use the following equality

$$\begin{aligned} \left(\sigma_s/\sigma_o\right)^D &= \exp\left(\log\left(\sigma_s/\sigma_o\right)^D\right) \\ &= \exp\left(D \log\left(\sigma_s/\sigma_o\right)\right) \end{aligned} \tag{A I-3}$$

Table-A I-1 Accuracy (%) on CIFAR-10-C dataset with Level 4 corruption for NC-TTT compared to *state-of-the-art*

	ResNet50	LAME	PTBN	TENT	TTT	TTT++	ClusT3	NC-TTT (ours)
Gaussian Noise	28.02	26.22 ±0.21	61.39 ±0.10	61.19 ±0.26	70.63 ±0.04	78.70 ±4.28	**79.14 ±0.03**	78.09 ±0.02
Shot noise	38.33	37.06 ±0.17	66.57 ±0.06	66.2 ±0.18	75.18 ±0.04	80.12 ±0.12	**81.51 ±0.15**	81.33 ±0.10
Impulse Noise	46.12	45.03 ±0.17	63.56 ±0.20	62.98 ±0.19	65.91 ±0.04	70.64 ±0.53	**76.95 ±0.07**	75.52 ±0.08
Defocus blur	67.33	67.70 ±0.07	85.48 ±0.12	85.32 ±0.18	**91.95 ±0.02**	81.75 ±0.43	90.33 ±0.09	91.91 ±0.05
Glass blur	34.42	32.63 ±0.09	52.26 ±0.04	52.08 ±0.15	60.44 ±0.05	62.85 ±0.50	**71.09 ±0.17**	69.95 ±0.09
Motion blur	63.71	64.00 ±0.01	80.78 ±0.12	80.75 ±0.09	86.29 ±0.10	68.42 ±1.08	87.87 ±0.11	**89.02 ±0.10**
Zoom blur	61.27	62.12 ±0.21	83.33 ±0.11	83.28 ±0.10	89.90 ±0.04	70.74 ±2.05	88.86 ±0.04	**90.96 ±0.05**
Snow	72.15	72.18 ±0.04	73.25 ±0.16	73.17 ±0.25	81.25 ±0.02	52.43 ±0.56	84.30 ±0.07	**85.36 ±0.06**
Frost	62.27	61.72 ±0.06	73.41 ±0.22	73.54 ±0.16	83.83 ±0.04	52.80 ±2.67	87.17 ±0.07	**88.08 ±0.02**
Fog	81.86	82.07 ±0.03	83.88 ±0.06	83.81 ±0.09	90.62 ±0.05	41.75 ±0.09	90.03 ±0.02	**92.07 ±0.07**
Brightness	87.58	87.64 ±0.08	86.81 ±0.05	86.81 ±0.23	92.87 ±0.09	50.95 ±2.19	**92.99 ±0.06**	**94.41 ±0.03**
Contrast	68.62	69.02 ±0.11	84.16 ±0.09	84.23 ±0.29	90.94 ±0.07	45.28 ±0.55	**89.24 ±0.07**	**91.52 ±0.03**
Elastic transform	67.84	68.32 ±0.03	76.44 ±0.18	76.21 ±0.08	84.03 ±0.11	35.53 ±1.51	**86.74 ±0.04**	86.39 ±0.04
Pixelate	56.3	55.94 ±0.08	76.34 ±0.10	76.40 ±0.16	84.92 ±0.15	33.64 ±0.83	87.93 ±0.03	**88.06 ±0.20**
JPEG compression	70.62	70.44 ±0.18	69.64 ±0.03	69.54 ±0.05	76.46 ±0.04	28.01 ±1.75	**85.11 ±0.06**	82.73 ±0.07
Average	60.43	60.14	74.48	74.37	81.68	56.91	85.28	**85.69**

to obtain

$$p(y_s = 1 \,|\, \widetilde{\mathbf{z}}_{i,m}) \;=\; \frac{1}{1 \;+\; \exp\!\Big(\!-\!\frac{1}{2}(1/\sigma_s^2 - 1/\sigma_o^2)\|\boldsymbol{\epsilon}\|^2 \;+\; D\log\left(\sigma_s/\sigma_o\right)\Big)} \tag{A I-4}$$

Last, since $\log\left(\sigma_s/\sigma_o\right) = -\log\left(\sigma_o/\sigma_s\right)$, we finally get $p(y_s = 1 \,|\, \widetilde{\mathbf{z}}) = 1/\big(1 + \exp(-u)\big)$ with

$$u \;=\; \frac{1}{2}\left(\frac{1}{\sigma_o^2} - \frac{1}{\sigma_s^2}\right)\|\boldsymbol{\epsilon}\|^2 \;+\; D\log\left(\frac{\sigma_o}{\sigma_s}\right) \quad \square \tag{A I-5}$$

2. Results on different levels of CIFAR-10-C corruptions

We evaluate NC-TTT on the remaining severity levels of CIFAR-10-C (see Tables I-1-I-4). Accuracy decreases on all methods as the severity augments, but NC-TTT outperforms the closest competitors from the *state-of-the-art* on the majority of the corruptions and in average for the whole dataset. To be more precise, when considering NC-TTT at a lower severity level (refer to Table I-4), it demonstrates superior performance across all corruptions, with the exception of *Gaussian Noise* and *JPEG compression*. This is noteworthy, given that Gaussian noise is introduced to features during training.

Table-A I-2 Accuracy (%) on CIFAR-10-C dataset with Level 3 corruption for NC-TTT compared to *state-of-the-art*

	ResNet50	LAME	PTBN	TENT	TTT	TTT++	ClusT3	NC-TTT (ours)
Gaussian Noise	33.99	32.37 ±0.16	64.55 ±0.13	64.67 ±0.17	74.10 ±0.09	80.29 ±0.81	**81.55 ±0.09**	81.09 ±0.09
Shot noise	46.35	45.83 ±0.14	69.82 ±0.08	70.04 ±0.14	78.43 ±0.07	82.46 ±0.37	**84.12 ±0.02**	84.03 ±0.09
Impulse Noise	59.90	59.43 ±0.13	72.08 ±0.14	71.95 ±0.33	76.32 ±0.10	79.20 ±0.38	**83.75 ±0.01**	83.73 ±0.05
Defocus blur	79.29	79.67 ±0.11	87.62 ±0.17	87.39 ±0.05	93.25 ±0.06	87.68 ±0.38	91.74 ±0.07	**93.51 ±0.08**
Glass blur	47.29	46.36 ±0.10	63.29 ±0.11	63.26 ±0.21	72.09 ±0.11	72.52 ±0.56	**79.78 ±0.02**	79.25 ±0.08
Motion blur	63.42	63.72 ±0.07	81.13 ±0.13	80.99 ±0.08	86.48 ±0.09	69.59 ±1.38	88.02 ±0.10	**89.02 ±0.02**
Zoom blur	67.86	68.23 ±0.08	84.57 ±0.11	84.34 ±0.06	91.00 ±0.02	73.23 ±2.33	89.90 ±0.07	**91.86 ±0.10**
Snow	74.93	74.78 ±0.05	75.08 ±0.14	75.14 ±0.19	83.90 ±0.07	57.96 ±1.02	86.22 ±0.07	**87.17 ±0.05**
Frost	64.54	64.16 ±0.08	74.15 ±0.04	73.98 ±0.14	84.13 ±0.10	49.94 ±3.53	**87.37 ±0.07**	88.05 ±0.05
Fog	85.73	85.98 ±0.16	86.57 ±0.09	86.38 ±0.15	92.19 ±0.08	52.89 ±4.13	91.83 ±0.01	**93.55 ±0.01**
Brightness	88.93	88.67 ±0.08	87.50 ±0.19	87.44 ±0.01	93.53 ±0.09	57.96 ±1.32	93.31 ±0.04	**94.61 ±0.01**
Contrast	79.66	79.99 ±0.05	85.63 ±0.05	85.46 ±0.08	91.85 ±0.09	53.44 ±2.37	90.83 ±0.05	**92.54 ±0.09**
Elastic transform	75.67	75.96 ±0.14	82.72 ±0.14	82.56 ±0.15	90.09 ±0.10	36.49 ±3.72	89.33 ±0.11	**90.95 ±0.03**
Pixelate	74.83	75.12 ±0.04	82.17 ±0.14	81.91 ±0.13	89.30 ±0.10	33.41 ±3.02	90.23 ±0.06	**91.44 ±0.05**
JPEG compression	73.70	73.66 ±0.16	71.54 ±0.09	71.54 ±0.15	78.95 ±0.09	28.82 ±2.74	**86.55 ±0.06**	85.10 ±0.01
Average	67.74	67.60	77.89	77.80	85.04	61.06	87.64	**88.39**

Table-A I-3 Accuracy (%) on CIFAR-10-C dataset with Level 2 corruption for NC-TTT compared to *state-of-the-art*

	ResNet50	LAME	PTBN	TENT	TTT	TTT++	ClusT3	NC-TTT (ours)
Gaussian Noise	50.53	50.02 ±0.24	71.31 ±0.16	71.43 ±0.08	81.18 ±0.11	85.41 ±2.26	**86.07 ±0.08**	85.37 ±0.08
Shot noise	69.27	69.47 ±0.22	78.97 ±0.19	79.02 ±0.17	87.54 ±0.10	88.79 ±0.44	89.77 ±0.04	**90.15 ±0.09**
Impulse Noise	68.57	68.68 ±0.08	77.09 ±0.13	77.03 ±0.15	82.20 ±0.13	84.27 ±0.29	86.60 ±0.03	**86.89 ±0.11**
Defocus blur	87.45	87.46 ±0.14	88.20 ±0.11	88.06 ±0.06	93.67 ±0.06	90.85 ±0.42	92.87 ±0.01	**94.32 ±0.01**
Glass blur	43.26	42.04 ±0.19	62.66 ±0.09	62.55 ±0.11	71.33 ±0.04	71.60 ±1.95	78.81 ±0.11	**79.86 ±0.06**
Motion blur	72.98	73.14 ±0.06	83.51 ±0.16	83.46 ±0.10	89.57 ±0.07	77.38 ±1.12	89.78 ±0.13	**91.23 ±0.04**
Zoom blur	74.89	75.23 ±0.18	85.81 ±0.21	85.79 ±0.05	92.05 ±0.10	80.30 ±1.45	90.82 ±0.04	**92.76 ±0.10**
Snow	71.11	70.78 ±0.12	74.73 ±0.11	74.69 ±0.22	82.96 ±0.08	68.56 ±1.36	86.30 ±0.04	**87.55 ±0.05**
Frost	76.67	76.46 ±0.02	79.54 ±0.15	79.41 ±0.27	87.67 ±0.03	63.66 ±3.39	90.27 ±0.10	**91.09 ±0.03**
Fog	88.51	88.55 ±0.08	87.62 ±0.10	87.60 ±0.17	93.23 ±0.04	64.26 ±3.37	93.07 ±0.04	**94.46 ±0.02**
Brightness	89.75	89.52 ±0.01	88.09 ±0.03	87.97 ±0.14	93.69 ±0.08	67.19 ±1.23	93.64 ±0.01	**94.97 ±0.04**
Contrast	84.58	84.87 ±0.07	86.19 ±0.17	86.41 ±0.04	92.50 ±0.12	62.90 ±1.93	92.00 ±0.01	**3.50 ±0.08**
Elastic transform	82.10	82.17 ±0.10	83.69 ±0.13	83.68 ±0.08	90.98 ±0.12	50.06 ±2.37	90.37 ±0.01	**91.60 ±0.05**
Pixelate	81.04	80.96 ±0.13	82.92 ±0.14	83.01 ±0.07	90.61 ±0.15	43.33 ±3.31	91.28 ±0.09	**92.46 ±0.06**
JPEG compression	76.06	75.92 ±0.09	73.63 ±0.02	73.56 ±0.13	81.37 ±0.11	28.26 ±2.78	**87.86 ±0.08**	86.27 ±0.03
Average	74.45	74.35	80.26	80.24	87.37	68.45	89.30	**90.17**

3. Hyperparameter search on VisDA-C

In order to choose the best configuration for VisDA-C, we performed a hyperparameter search considering the four different layer blocks from ResNet50 as well as four different in-distribution noise standard deviation values (i.e., 0.01, 0.015, 0.025, 0.05). The results are obtained across three executions per combination. We show in Fig.I-1 that the best performance can be generally

Table-A I-4 Accuracy (%) on CIFAR-10-C dataset with Level 1 corruption for NC-TTT compared to *state-of-the-art*

	ResNet50	LAME	PTBN	TENT	TTT	TTT++	ClusT3	NC-TTT (ours)
Gaussian Noise	71.38	71.35 ±0.05	79.22 ±0.13	79.52 ±0.12	88.38 ±0.12	90.14 ±1.05	**90.35 ±0.05**	90.29 ±0.01
Shot noise	80.39	80.32 ±0.07	82.21 ±0.05	82.18 ±0.15	90.43 ±0.02	90.89 ±0.29	91.42 ±0.02	**92.25 ±0.02**
Impulse Noise	80.04	79.98 ±0.09	82.39 ±0.08	82.48 ±0.15	88.23 ±0.02	87.76 ±0.06	90.51 ±0.06	**91.07 ±0.05**
Defocus blur	90.17	89.9 ±0.06	88.28 ±0.04	88.26 ±0.15	93.89 ±0.04	91.51 ±0.48	93.72 ±0.09	**95.12 ±0.02**
Glass blur	40.96	39.87 ±0.16	63.19 ±0.05	63.22 ±0.15	71.12 ±0.07	72.12 ±2.13	79.01 ±0.21	**79.78 ±0.05**
Motion blur	82.78	82.81 ±0.11	85.99 ±0.09	85.89 ±0.08	91.97 ±0.05	84.11 ±0.91	91.50 ±0.13	**93.15 ±0.07**
Zoom blur	78.58	79.03 ±0.06	86.19 ±0.06	86.23 ±0.04	92.21 ±0.08	81.76 ±1.38	90.87 ±0.04	**92.60 ±0.07**
Snow	83.45	83.32 ±0.11	82.94 ±0.13	82.84 ±0.35	88.90 ±0.04	75.89 ±0.75	90.33 ±0.02	**91.57 ±0.03**
Frost	84.84	84.44 ±0.10	83.88 ±0.15	83.71 ±0.24	91.17 ±0.03	71.54 ±3.13	92.19 ±0.06	**93.16 ±0.08**
Fog	90.15	90.05 ±0.05	88.31 ±0.13	88.05 ±0.06	93.71 ±0.09	70.58 ±1.29	93.64 ±0.01	**95.11 ±0.03**
Brightness	90.35	90.24 ±0.06	88.28 ±0.09	88.35 ±0.25	93.90 ±0.06	64.40 ±2.69	93.83 ±0.05	**95.28 ±0.02**
Contrast	89.52	89.57 ±0.07	87.98 ±0.09	87.93 ±0.08	93.61 ±0.05	53.60 ±3.80	93.61 ±0.03	**94.95 ±0.06**
Elastic transform	82.46	82.72 ±0.06	83.29 ±0.17	83.28 ±0.27	90.55 ±0.09	39.92 ±1.52	90.33 ±0.06	**91.62 ±0.07**
Pixelate	87.27	87.18 ±0.08	85.79 ±0.12	85.81 ±0.17	92.24 ±0.01	36.04 ±3.47	92.74 ±0.04	**93.84 ±0.03**
JPEG compression	82.03	81.66 ±0.07	79.72 ±0.10	79.82 ±0.14	86.86 ±0.08	30.90 ±1.18	**90.90 ±0.01**	90.18 ±0.05
Average	80.96	80.83	83.17	83.17	89.81	69.41	91.00	**92.00**

obtained on the first layer of the network, consistent with previous results in the field (Sun *et al.*, 2020; Liu *et al.*, 2021; Osowiechi *et al.*, 2023; Vargas Hakim *et al.*, 2023). Furthermore, an in-distribution standard deviation $\sigma_s = 0.025$ is found to perform the best across all the different layers.

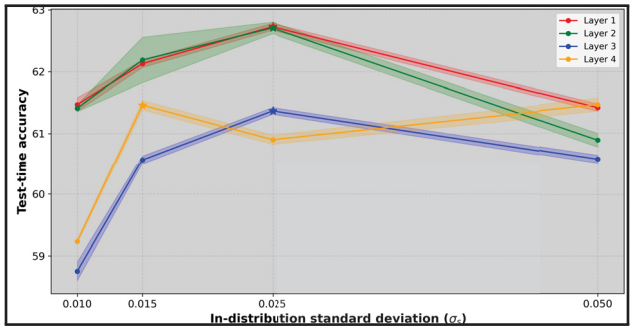


Figure-A I-1 Test-time accuracy on different layer blocks with different in-distribution standard deviation

APPENDIX II

CLIPARTT: ADAPTATION OF CLIP TO NEW DOMAINS AT TEST TIME - SUPPLEMENTARY MATERIAL

David Osowiechi*, Gustavo A. Vargas Hakim*
Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah,
Ismail Ben Ayed, Christian Desrosiers

Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

*Equal Contribution
Poster at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)
February 2025

1. Dataset Details

We evaluate CLIPArTT's performance across diverse TTA datasets using established methodologies. These datasets simulate challenging scenarios, offering insights into our approach's efficacy. Additionally, we explore CLIPArTT's adaptability on other datasets through zero-shot test-time adaptation.

Our evaluation framework encompasses *natural images*, *varied styles and textures images*, *common corruptions*, *simulated images*, and *video* providing a comprehensive assessment of the model's performance across diverse challenges.

In our evaluation of *natural images* (also known as zero-shot scenario), we utilize CIFAR-10, CIFAR-10.1, and CIFAR-100 datasets, each comprising 10,000 images and featuring 10 and 100 classes, respectively. These datasets represent natural imagery and are novel to the model under scrutiny. Notably, CIFAR-10.1 introduces a natural domain shift from CIFAR-10, thereby enriching our assessment with varied and nuanced data distributions. We also evaluate our method on ImageNet and extend our investigation on two datasets mostly used in the field of domain generalization: PACS Li *et al.* (2017) and OfficeHome Venkateswara *et al.* (2017) datasets, instrumental in understanding *texture and style variations*. The PACS dataset consists

of 9,991 images across four domains (Art, Cartoons, Photos, Sketches) and seven classes. Lastly, the OfficeHome dataset includes 15,588 images across four domains (Art, Clipart, Product, Real) and 65 classes. Evaluating across these distinct scenarios showcases the generalizability of our method.

Transitioning to our investigation of *common corruptions*, we turn to the CIFAR-10-C and CIFAR-100-C Hendrycks & Dietterich (2019). These datasets offer a diverse range of 15 distinct corruptions, including elastic transform and impulse noise, among others. Each corruption is characterized by 5 severity levels, yielding a total of 75 unique testing scenarios per dataset. Within each severity level, there are 10,000 images, contributing to a comprehensive evaluation of the model's robustness against a variety of corruption types and intensities. Finally, we test our method on ImageNet-C to evaluate its performance on a larger dataset with 1,000 classes.

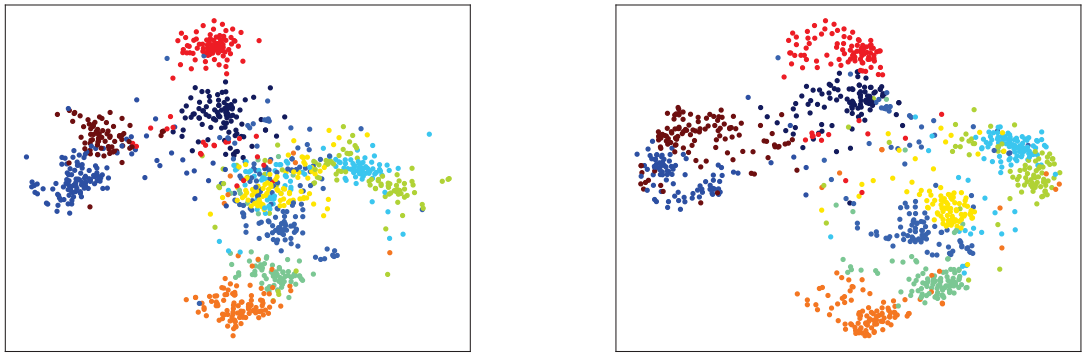
Finally, we examine the VisDA-C dataset's Peng *et al.* (2018) two domain shifts: *simulated* (3D) and *video* (YT). The former comprises a set of 152,397 images rendered in 3D across 12 different classes. The latter includes 72,372 YouTube video frames spanning the same categories. This dataset presents an important challenge, as it bridges the gap of the type of imagery that a model can be applied on.

2. Unsupervised clustering

In Fig. II-1, tSNE visualizations of data points are shown. We show how the distribution of data points change after adaptation, which improves the accuracy of class predictions and facilitates the assignment of ground truth labels.

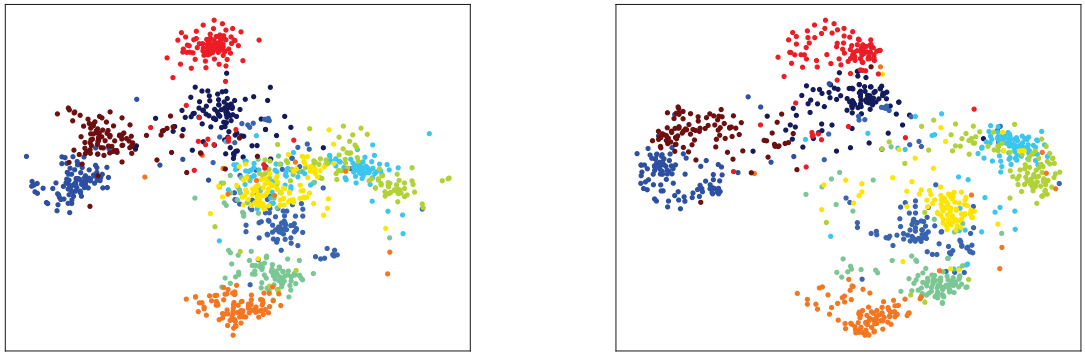
3. Additional settings

We explore additional experimental settings to further explore the strengths and weaknesses of CLIPArTT. Although these scenarios deviate from the standard practices in the TTA literature, they are useful to evaluate a method's performance in potentially challenging real-world



(a) Prediction (before adaptation)

(b) Prediction (after adaptation)



(c) Ground truth (before adaptation)

(d) Ground truth (after adaptation)

Figure-A II-1 The t-SNE visualizations exhibit discernible attributes of brightness within the visual features derived from CLIPArTT. Panels (a) and (b) present the model's predictions before and after 10 iterations of adaptation, respectively. Panels (c) and (d) demonstrate the actual labels in the absence of adaptation and following adaptation of the representations, respectively

applications. For most of the following experiments, we utilize the CIFAR-10 dataset's variants, unless otherwise is mentioned.

3.1 Imbalanced batch instances

CLIPArTT adapts to batch data in a trandsductive manner by computing the image-to-image similarity. The diversity of the batches cannot be ensured due to their finite size. The opposite case naturally arises in large scale classification datasets such as ImageNet, where including at

least one image per class in a batch of size 128 is impossible. On the other hand, adaptation to highly imbalanced batches is understudied.

In this experiment, we force extreme imbalances in the batches, by allowing only C randomly chosen classes to be present in the batch. As CLIPArTT is dependant also on the most probable predictions, it is expected that the miss-classification due to the imbalance would drive the model to directions opposite to the actual correct classes. In this experiment, we evaluate with $C = \{2, 3, 4, 5\}$ as the possible number of classes to be present in the batches, and compare CLIPArTT against TENT. Final results are shown in Table II-1).

It can be observed that TENT shows a higher robustness to class imbalance, as for each prediction, it depends less on the other images' predictions (i.e., induction). Interestingly, we observe a trend where TENT's performance decrease when more classes are present (e.g., $C = 5$ instead of $C = 2$), whereas CLIPArTT increase its performance as C grows.

Table-A II-1 Accuracy on imbalance datasets with different numbers of available classes at each batch

	CLIPArTT				TENT			
	2	3	4	5	2	3	4	5
Original	76.99	81.12	84.37	86.92	97.07	96.14	95.27	94.09
CIFAR-10.1	74.21	78.96	82.57	85.14	96.47	94.33	92.56	90.68
Gaussian Noise	47.12	50.83	54.18	58.63	63.76	49.49	49.92	47.15
Shot Noise	48.67	53.30	56.37	60.24	69.89	55.83	56.68	53.55
Impulse Noise	45.98	49.82	53.15	57.48	66.87	54.47	54.77	52.88
Defocus Blur	60.80	67.78	71.24	74.85	89.19	85.39	83.05	81.44
Glass Blur	47.82	53.07	56.45	60.12	77.11	65.34	62.72	58.78
Motion Blur	60.85	66.82	70.67	73.43	83.33	79.88	79.28	75.89
Zoom Blur	56.78	66.52	70.43	72.94	86.88	84.15	81.89	81.35
Snow	59.20	66.38	71.29	73.97	91.84	87.60	85.32	82.45
Frost	60.02	68.08	72.12	75.15	92.50	88.62	86.18	84.25
Fog	58.57	66.11	70.54	73.12	91.26	87.01	84.07	83.12
Brightness	72.29	77.62	80.75	83.55	95.45	93.82	92.29	91.15
Contrast	62.18	71.68	75.08	77.64	89.75	87.11	85.72	83.25
Elastic Transform	50.85	58.91	62.12	65.12	88.17	82.05	78.13	76.05
Pixelate	51.75	57.20	60.42	63.08	77.35	71.38	69.22	66.85
JPEG Compression	49.63	56.08	59.32	62.05	79.98	72.34	69.85	68.17

3.2 Open-Set classification

Another interesting scenario is open-set classification, where images from classes that are not considered are present in the batch. These images could potentially affect performance, specially in transductive methods and particularly in CLIPArTT, as the image-to-image and text-to-text similarities are combined. The only available text prompts are wrongly assigned to the out-of-distribution (OOD) samples.

To measure the effectiveness of our method in this scenario, we utilize SVHN-C, a corrupted version of the SVHN Netzer *et al.* (2011) dataset, analogous to CIFAR-10.C. For each batch of 128 image, 128 additional OOD images are included. Both parts are used at the same time for adaptation, and accuracy is only measured on the first half.

As seen in Table II-2, CLIPArTT defeats TENT's open-set accuracy by a significant margin. This encouraging result suggests that our method is robust to semantic out-of-distribution perturbations.

Table-A II-2 Open-set accuracy on CIFAR-10-C when using SVHN-C as the OOD dataset

	CLIPArTT	TENT
Gaussian Noise	59.72 ±0.05	41.27 ±0.03
Shot Noise	62.11 ±0.07	48.14 ±0.05
Impulse Noise	55.23 ±0.10	47.86 ±0.07
Defocus Blur	75.44 ±0.07	72.03 ±0.05
Glass Blur	60.12 ±0.07	42.08 ±0.08
Motion Blur	74.60 ±0.05	65.71 ±0.05
Zoom Blur	76.15 ±0.05	71.45 ±0.06
Snow	74.69 ±0.05	73.99 ±0.08
Frost	77.72 ±0.05	74.49 ±0.09
Fog	72.87 ±0.03	70.57 ±0.04
Brightness	85.70 ±0.09	82.34 ±0.05
Contrast	76.49 ±0.04	71.67 ±0.04
Elastic Transform	67.42 ±0.01	65.99 ±0.12
Pixelate	62.02 ±0.11	54.25 ±0.08
JPEG Compression	60.81 ±0.07	54.09 ±0.10
Average	69.41 ±8.86	62.40 ±13.20

3.3 Generalization after adaptation

In TTA, the classification performance is evaluated directly on the adapted set of images in an episodic manner. However, a high accuracy on this set does not necessarily guarantee a good performance in a separate set of images unknown to the model.

Using a batch size of 160 images, we measure CLIPArTT's generalization by separating 25% of each batch as a test-time validation split. The remaining 75% is used for adaptation prior to testing on the validation split. The accuracies on the adaptation split and the validation split are both reported and contrasted against TENT. Results are shown in Table II-3.

While TENT obtains a higher accuracy on the adaptation splits of natural images (i.e., CIFAR-10 and CIFAR-10.1), following a similar trend as in the main experiments' results, the difference in the generalization accuracy is smaller with respect to CLIPArTT's. Moreover, our method demonstrates a consistently better generalization and adaptation accuracy on corrupted samples, representing stronger domain shifts.

Table-A II-3 Generalization results on CIFAR-10 variants. Each batch is divided into an adaptation split and a validation split. Accuracy (Acc.) is measured on the former after adaptation, whilst the generalization accuracy (Acc. Gen.) is measured on the later

	CLIPArTT		TENT	
	Acc.	Acc. Gen.	Acc.	Acc. Gen.
Original	95.43 ±0.01	71.23 ±0.0008	97.02 ±0.02	72.68 ±0.0023
CIFAR-10.1	88.75 ±0.85	65.87 ±0.8083	90.49 ±0.38	67.07 ±1.1
Gaussian Noise	63.79 ±0.20	46.87 ±0.0008	45.30 ±0.05	30.83 ±0.0030
Shot Noise	66.38 ±0.10	79.80 ±0.0038	50.65 ±0.18	35.85 ±0.0025
Impulse Noise	59.31 ±0.19	44.56 ±0.0023	51.91 ±0.20	37.99 ±0.0037
Defocus Blur	81.26 ±0.03	60.13 ±0.0016	84.48 ±0.07	60.32 ±0.0028
Glass Blur	65.38 ±0.04	49.47 ±0.0020	56.68 ±0.26	41.32 ±0.0035
Motion Blur	80.41 ±0.12	60.08 ±0.0022	75.70 ±0.13	55.72 ±0.0013
Zoom Blur	81.97 ±0.05	60.63 ±0.0009	81.14 ±0.13	59.33 ±0.0007
Snow	82.06 ±0.14	60.83 ±0.0014	83.08 ±0.13	61.08 ±0.0008
Frost	83.85 ±0.19	61.73 ±0.0009	84.70 ±0.11	66.75 ±0.0006
Fog	90.04 ±0.27	59.48 ±0.0027	81.56 ±0.09	60.69 ±0.0033
Brightness	91.73 ±0.04	68.01 ±0.0017	92.93 ±0.03	68.80 ±0.0018
Contrast	82.63 ±0.07	61.28 ±0.0025	83.53 ±0.06	62.97 ±0.0009
Elastic Transform	74.04 ±0.22	54.87 ±0.0011	74.08 ±0.10	54.92 ±0.0009
Pixelate	70.58 ±0.02	51.53 ±0.0042	67.35 ±0.18	50.19 ±0.0004
JPEG Compression	67.32 ±0.06	49.56 ±0.0023	66.25 ±0.14	48.85 ±0.0021
Average	**75.38** ±9.04	55.92 ±6.62	71.78 ±14.14	52.77 ±11.10

4. Computational Cost

In this section, we compare the computational cost of CLIPArTT with other TTA methods through a thorough evaluation under consistent conditions, using an NVIDIA A6000 GPU within the same Python environment. The provided table II-4 compares adaptation time, memory usage, and the number of learnable parameters across various TTA methods, including our proposed CLIPArTT. The results demonstrate that CLIPArTT maintains competitive adaptation time and memory usage relative to other approaches, such as TENT and TPT.

Table-A II-4 Comparison of Computational Cost

Method	Adaptation Time	Memory	Pct. of Learnable Parameters
TENT	0.28 s	1.5 GB	0.026%
TPT	0.26 s	1.7 GB	0.001%
CLIPArTT	0.55 s	1.7 GB	0.026%

5. Comprehensive experimental results

We present comprehensive tables containing all the detailed information about results that was summarized in the main paper.

Table-A II-5 Accuracy (%) on CIFAR-10/100 and CIFAR-10/100-C datasets with Level 5 corruption for the top 1 or the top 3 predicted classes

	CIFAR10		CIFAR100	
	Top 1	Top 3	Top 1	Top 3
Original	88.74	100.00	61.68	97.34
Gaussian Noise	35.27	99.87	14.8	63.66
Shot noise	39.67	99.99	16.03	67.02
Impulse Noise	42.61	100.00	13.85	64.4
Defocus blur	69.76	100.00	36.74	90.14
Glass blur	42.40	100.00	14.19	61.66
Motion blur	63.97	100.00	36.14	90.36
Zoom blur	69.83	100.00	40.24	91.27
Snow	71.78	100.00	38.95	91.40
Frost	72.86	100.00	40.56	92.23
Fog	67.04	99.98	38.00	91.51
Brightness	81.87	100.00	48.18	93.10
Contrast	64.37	100.00	29.53	84.67
Elastic transform	60.83	100.00	26.33	78.96
Pixelate	50.53	100.00	21.98	75.65
JPEG compression	55.48	100.00	25.91	80.81
Average	59.22	99.99	29.43	81.12

Table-A II-6 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with Level 5 corruption for different number of K selected classes to create pseudo-label

	K = 1	K = 3	K = 4
Original	89.8 ±0.05	90.04 ±0.13	90.41 ±0.07
CIFAR 10.1	85.37 ±0.17	86.35 ±0.27	86.07 ±0.21
Gaussian Noise	60.20 ±0.24	59.90 ±0.36	59.71 ±0.15
Shot noise	62.08 ±0.11	62.77 ±0.07	62.17 ±0.16
Impulse Noise	54.33 ±0.07	56.02 ±0.16	56.27 ±0.15
Defocus blur	77.16 ±0.02	76.74 ±0.05	76.79 ±0.11
Glass blur	61.91 ±0.15	61.77 ±0.16	61.72 ±0.23
Motion blur	74.94 ±0.15	76.01 ±0.19	76.33 ±0.10
Zoom blur	76.84 ±0.13	77.40 ±0.20	77.15 ±0.04
Snow	76.87 ±0.05	77.29 ±0.16	76.56 ±0.16
Frost	77.81 ±0.04	79.20 ±0.08	78.42 ±0.04
Fog	75.83 ±0.28	75.74 ±0.14	75.65 ±0.06
Brightness	85.55 ±0.12	86.59 ±0.16	86.83 ±0.10
Contrast	78.02 ±0.18	77.82 ±0.14	78.27 ±0.14
Elastic transform	69.42 ±0.07	70.20 ±0.01	69.81 ±0.20
Pixelate	66.07 ±0.09	66.52 ±0.13	66.45 ±0.08
JPEG compression	64.82 ±0.26	63.51 ±0.14	62.72 ±0.25
Average	70.79	71.17	70.99

Table-A II-7 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with Level 5 corruption for different number of K selected classes to create pseudo-labels

	$K = 1$	$K = 3$	$K = 5$	$K = 7$	$K = 10$	$K = 20$
Original	69.00 ±0.22	69.79 ±0.04	69.68 ±0.07	69.56 ±0.02	69.78 ±0.02	**69.93 ±0.08**
Gaussian Noise	**26.05 ±0.11**	25.32 ±0.14	24.69 ±0.03	24.60 ±0.03	24.70 ±0.15	23.73 ±0.07
Shot noise	**28.88 ±0.11**	27.90 ±0.05	27.25 ±0.26	26.75 ±0.07	26.83 ±0.26	25.73 ±0.13
Impulse Noise	24.04 ±0.09	**25.62 ±0.09**	25.12 ±0.14	25.25 ±0.14	24.95 ±0.23	24.57 ±0.12
Defocus blur	49.03 ±0.15	**49.88 ±0.23**	49.75 ±0.11	49.74 ±0.25	49.62 ±0.10	49.49 ±0.07
Glass blur	26.77 ±0.14	**27.89 ±0.03**	27.76 ±0.23	27.28 ±0.07	26.57 ±0.07	25.52 ±0.09
Motion blur	46.50 ±0.09	**47.93 ±0.14**	47.48 ±0.21	47.57 ±0.10	47.53 ±0.09	47.36 ±0.09
Zoom blur	52.08 ±0.12	52.70 ±0.06	52.22 ±0.10	52.10 ±0.24	52.62 ±0.24	**52.82 ±0.09**
Snow	49.24 ±0.07	**49.72 ±0.01**	48.98 ±0.08	48.87 ±0.08	49.13 ±0.08	49.54 ±0.13
Frost	**49.91 ±0.07**	49.63 ±0.12	48.43 ±0.17	48.11 ±0.06	48.72 ±0.08	49.13 ±0.10
Fog	47.15 ±0.04	48.77 ±0.04	48.95 ±0.18	48.78 ±0.22	**49.06 ±0.05**	48.74 ±0.36
Brightness	60.01 ±0.08	**61.27 ±0.08**	60.77 ±0.16	60.89 ±0.19	60.98 ±0.18	61.03 ±0.19
Contrast	46.90 ±0.21	**48.55 ±0.24**	49.01 ±0.14	49.07 ±0.03	**49.27 ±0.09**	49.08 ±0.12
Elastic transform	36.32 ±0.10	37.45 ±0.08	**37.63 ±0.12**	37.31 ±0.09	37.13 ±0.16	36.94 ±0.11
Pixelate	32.52 ±0.17	33.88 ±0.14	34.40 ±0.15	34.38 ±0.16	**34.65 ±0.09**	34.32 ±0.02
JPEG compression	35.81 ±0.11	**36.07 ±0.32**	35.77 ±0.01	35.60 ±0.10	35.63 ±0.10	35.29 ±0.14
Average	40.75	**41.51**	41.21	41.09	41.16	40.89

Table-A II-8 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with Level 5 corruption for different number of iterations to update the model at test-time

	Iter = 1	Iter = 5	Iter = 10	Iter = 20
Original	89.59 ±0.01	**90.54 ±0.09**	90.04 ±0.13	88.32 ±0.12
CIFAR 10.1	84.78 ±0.02	**86.67 ±0.06**	86.35 ±0.27	84.33 ±0.31
Gaussian Noise	39.75 ±0.04	53.79 ±0.08	**59.90 ±0.36**	59.33 ±0.20
Shot noise	43.80 ±0.04	57.16 ±0.24	62.77 ±0.07	**63.08 ±0.43**
Impulse Noise	45.19 ±0.07	52.44 ±0.14	56.02 ±0.16	**56.73 ±0.01**
Defocus blur	72.93 ±0.07	76.36 ±0.10	**76.74 ±0.05**	75.33 ±0.13
Glass blur	46.61 ±0.06	57.45 ±0.13	61.77 ±0.16	**62.01 ±0.27**
Motion blur	67.89 ±0.03	74.34 ±0.14	**76.01 ±0.19**	75.94 ±0.28
Zoom blur	73.24 ±0.05	77.03 ±0.07	**77.40 ±0.20**	75.42 ±0.13
Snow	73.81 ±0.07	76.51 ±0.08	**77.29 ±0.16**	76.18 ±0.21
Frost	74.80 ±0.04	78.13 ±0.12	**79.20 ±0.08**	77.44 ±0.30
Fog	69.81 ±0.06	74.16 ±0.09	**75.74 ±0.14**	74.66 ±0.02
Brightness	84.16 ±0.01	**86.61 ±0.10**	86.59 ±0.16	85.14 ±0.42
Contrast	67.75 ±0.03	74.85 ±0.04	**77.82 ±0.14**	77.75 ±0.11
Elastic transform	63.15 ±0.08	68.53 ±0.19	**70.20 ±0.01**	68.48 ±0.24
Pixelate	54.20 ±0.02	61.87 ±0.04	66.52 ±0.13	**67.13 ±0.15**
JPEG compression	57.46 ±0.09	62.00 ±0.13	63.51 ±0.14	**63.64 ±0.20**
Average	62.30	68.75	**71.17**	70.55

Table-A II-9 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with different targets

	Image	Text	Image + Text
Original	**90.18 ±0.02**	89.05 ±0.14	90.04 ±0.13
CIFAR 10.1	86.25 ±0.37	84.85 ±0.40	**86.35 ±0.27**
Gaussian Noise	59.05 ±0.30	59.29 ±0.27	**59.90 ±0.36**
Shot noise	62.11 ±0.18	61.89 ±0.31	**62.77 ±0.07**
Impulse Noise	55.43 ±0.12	55.48 ±0.10	**56.02 ±0.16**
Defocus blur	76.88 ±0.09	76.25 ±0.10	**76.74 ±0.05**
Glass blur	61.56 ±0.03	61.28 ±0.33	**61.77 ±0.16**
Motion blur	**76.32 ±0.15**	75.37 ±0.27	76.01 ±0.19
Zoom blur	**77.66 ±0.09**	76.29 ±0.11	77.40 ±0.20
Snow	77.28 ±0.08	76.03 ±0.16	**77.29 ±0.16**
Frost	78.80 ±0.18	78.05 ±0.21	**79.20 ±0.08**
Fog	75.69 ±0.17	73.70 ±0.15	**75.74 ±0.14**
Brightness	**86.62 ±0.20**	84.69 ±0.04	86.59 ±0.16
Contrast	77.43 ±0.12	74.52 ±0.02	**77.82 ±0.14**
Elastic transform	69.63 ±0.02	69.33 ±0.20	**70.20 ±0.01**
Pixelate	66.33 ±0.16	64.86 ±0.29	**66.52 ±0.13**
JPEG compression	**63.92 ±0.13**	63.44 ±0.20	63.51 ±0.14
Average	70.98	70.03	**71.17**

Table-A II-10 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with Level 5 corruption for different batch sizes

	CLIP	BS = 8	BS = 16	BS = 32	BS = 64	BS = 128
Original	88.74	82.50 ±0.13	85.89 ±0.19	88.25 ±0.15	89.48 ±0.15	90.04 ±0.13
CIFAR 10.1	83.25	77.2 ±0.92	81.55 ±0.53	84.00 ±0.31	85.40 ±0.08	86.35 ±0.27
Gaussian Noise	35.27	47.30 ±0.37	50.91 ±0.35	54.23 ±0.28	57.89 ±0.13	59.90 ±0.36
Shot noise	39.67	49.62 ±0.26	53.1 ±0.27	56.88 ±0.23	60.56 ±0.12	62.77 ±0.07
Impulse Noise	42.61	47.24 ±0.22	50.24 ±0.48	52.7 ±0.21	54.88 ±0.17	56.02 ±0.16
Defocus blur	69.76	68.24 ±0.35	72.22 ±0.04	75.09 ±0.16	75.97 ±0.27	76.74 ±0.05
Glass blur	42.40	49.49 ±0.30	53.27 ±0.04	57.18 ±0.24	60.12 ±0.14	61.77 ±0.16
Motion blur	63.97	65.22 ±0.06	69.02 ±0.30	72.54 ±0.27	74.71 ±0.18	76.01 ±0.19
Zoom blur	69.83	67.69 ±0.20	71.33 ±0.11	74.53 ±0.11	76.35 ±0.07	77.40 ±0.20
Snow	71.78	68.68 ±0.42	72.37 ±0.11	74.93 ±0.18	76.53 ±0.41	77.29 ±0.16
Frost	72.86	70.35 ±0.25	73.93 ±0.34	76.81 ±0.23	78.22 ±0.13	79.20 ±0.08
Fog	67.04	66.25 ±0.31	69.71 ±0.24	72.36 ±0.23	73.96 ±0.21	75.74 ±0.14
Brightness	81.87	77.36 ±0.17	81.20 ±0.20	84.07 ±0.08	85.58 ±0.25	86.59 ±0.16
Contrast	64.37	65.12 ±0.07	69.02 ±0.12	72.60 ±0.46	75.79 ±0.24	77.82 ±0.14
Elastic transform	60.83	59.61 ±0.11	63.67 ±0.13	66.36 ±0.26	68.74 ±0.07	70.20 ±0.01
Pixelate	50.53	56.78 ±0.24	60.01 ±0.06	62.57 ±0.19	64.64 ±0.03	66.52 ±0.13
JPEG compression	55.48	57.59 ±0.26	60.78 ±0.12	62.63 ±0.06	63.43 ±0.16	63.51 ±0.14
Average	59.22	61.10	64.72	67.70	69.82	71.17

Table-A II-11 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-B/16 as visual encoder

	CLIP	TENT	TPT (BS=32)	CLIPArTT
Original	89.25	**92.75 ±0.17**	89.80 ±0.05	92.61 ±0.05
CIFAR 10.1	84.00	88.52 ±0.33	83.75.0.21 ±	**88.72 ±0.33**
Gaussian Noise	37.75	31.04 ±0.38	35.35 ±0.15	**60.89 ±0.26**
Shot noise	41.10	40.54 ±0.41	41.03 ±0.19	**65.19 ±0.21**
Impulse Noise	51.71	58.03 ±0.16	54.86 ±0.07	**67.55 ±0.09**
Defocus blur	70.07	77.57 ±0.03	70.29 ±0.02	**78.92 ±0.12**
Glass blur	42.24	47.16 ±0.05	37.86 ±0.17	**57.18 ±0.20**
Motion blur	65.81	76.16 ±0.05	67.43 ±0.11	**76.59 ±0.06**
Zoom blur	72.50	**79.64 ±0.12**	72.91 ±0.02	79.62 ±0.11
Snow	73.23	**81.68 ±0.03**	72.98 ±0.32	81.13 ±0.29
Frost	76.52	**83.22 ±0.05**	75.87 ±0.16	81.24 ±0.08
Fog	68.35	**80.78 ±0.15**	69.13 ±0.27	78.47 ±0.19
Brightness	83.36	**89.85 ±0.11**	83.67 ±0.14	88.66 ±0.15
Contrast	61.90	**79.24 ±0.19**	62.16 ±0.06	75.15 ±0.07
Elastic transform	53.16	62.54 ±0.08	51.26 ±0.23	**69.49 ±0.08**
Pixelate	48.48	67.08 ±0.24	44.65 ±0.21	**71.80 ±0.16**
JPEG compression	56.05	65.42 ±0.05	56.73 ±0.07	**66.42 ±0.25**
Average	60.15	68.00	59.75	**73.22**

Table-A II-12 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-L/14 as visual encoder

	CLIP	TENT	TPT (BS=32)	CLIPArTT
Original	95.36	**96.13 ±0.06**	95.18 ±0.02	95.16 ±0.03
CIFAR 10.1	91.20	**92.22 ±0.25**	91.32 ±0.12	91.02 ±0.02
Gaussian Noise	64.64	68.87 ±0.20	64.44 ±0.11	**70.04 ±0.31**
Shot noise	67.82	**71.95 ±0.06**	66.81 ±0.19	71.44 ±0.16
Impulse Noise	78.21	**80.22 ±0.19**	76.46 ±0.17	79.42 ±0.15
Defocus blur	80.73	**83.10 ±0.03**	79.01 ±0.23	81.75 ±0.19
Glass blur	50.29	57.12 ±0.07	49.64 ±0.23	**58.13 ±0.23**
Motion blur	80.75	**82.69 ±0.11**	78.85 ±0.04	80.76 ±0.12
Zoom blur	82.75	**84.91 ±0.08**	82.32 ±0.13	83.39 ±0.05
Snow	83.01	**85.99 ±0.11**	82.69 ±0.10	84.48 ±0.07
Frost	84.90	**87.15 ±0.12**	84.63 ±0.08	85.21 ±0.06
Fog	78.44	**81.30 ±0.07**	77.56 ±0.17	79.27 ±0.07
Brightness	91.67	**93.07 ±0.04**	90.94 ±0.04	91.87 ±0.09
Contrast	84.20	**87.93 ±0.04**	82.88 ±0.09	86.19 ±0.06
Elastic transform	65.45	**69.96 ±0.12**	64.81 ±0.14	67.43 ±0.24
Pixelate	75.10	**77.89 ±0.05**	72.92 ±0.12	77.11 ±0.10
JPEG compression	72.58	**75.49 ±0.07**	71.18 ±0.19	74.46 ±0.11
Average	76.04	**79.18**	75.01	78.06

Table-A II-13 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with ViT-B/16 as visual encoder

	CLIP	TENT	TPT (BS=32)	CLIPArTT
Original	64.76	**71.73 ±0.14**	67.15 ±0.23	71.34 ±0.07
Gaussian Noise	15.88	12.28 ±0.20	15.43 ±0.03	**19.01 ±0.24**
Shot noise	17.49	15.07 ±0.21	16.88 ±0.07	**20.27 ±0.21**
Impulse Noise	21.43	13.13 ±0.16	**22.12 ±0.15**	17.66 ±0.10
Defocus blur	40.10	**50.35 ±0.03**	41.08 ±0.22	49.86 ±0.13
Glass blur	13.48	4.84 ±0.14	**18.43 ±0.15**	18.34 ±0.31
Motion blur	39.82	49.85 ±0.37	40.85 ±0.26	**50.00 ±0.09**
Zoom blur	45.45	**54.76 ±0.04**	46.77 ±0.06	54.13 ±0.08
Snow	42.77	52.38 ±0.18	47.24 ±0.18	**52.80 ±0.27**
Frost	45.39	**51.66 ±0.04**	48.61 ±0.14	49.56 ±0.08
Fog	38.98	**50.74 ±0.14**	39.92 ±0.16	49.92 ±0.11
Brightness	52.55	**64.26 ±0.09**	55.83 ±0.10	63.76 ±0.13
Contrast	33.32	**48.69 ±0.08**	33.13 ±0.16	47.86 ±0.02
Elastic transform	24.39	**33.56 ±0.28**	27.36 ±0.10	32.93 ±0.23
Pixelate	21.89	36.20 ±0.28	21.26 ±0.10	**39.49 ±0.21**
JPEG compression	27.21	30.80 ±0.05	30.97 ±0.10	**35.56 ±0.23**
Average	32.01	37.90	33.73	**40.08**

Table-A II-14 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with ViT-L/14 as visual encoder

	CLIP	TENT	TPT (BS=16)	CLIPArTT
Original	73.28	78.03 ±0.08	76.85 ±0.06	**79.42 ±0.08**
Gaussian Noise	30.55	36.93 ±0.03	36.10 ±0.11	**41.46 ±0.15**
Shot noise	34.58	40.96 ±0.16	38.23 ±0.13	**44.27 ±0.09**
Impulse Noise	44.89	49.09 ±0.14	49.69 ±0.21	**51.44 ±0.23**
Defocus blur	48.88	55.23 ±0.07	50.43 ±0.19	**56.55 ±0.22**
Glass blur	23.46	27.02 ±0.23	24.35 ±0.22	**30.47 ±0.14**
Motion blur	50.83	56.03 ±0.20	51.94 ±0.04	**56.98 ±0.18**
Zoom blur	56.02	61.19 ±0.10	56.96 ±0.16	**62.56 ±0.04**
Snow	49.03	55.60 ±0.09	54.89 ±0.11	**58.81 ±0.11**
Frost	53.27	58.21 ±0.15	58.15 ±0.33	**60.38 ±0.23**
Fog	48.51	53.37 ±0.25	49.26 ±0.13	**54.38 ±0.04**
Brightness	60.53	67.34 ±0.17	66.60 ±0.10	**69.63 ±0.14**
Contrast	50.24	59.91 ±0.13	53.64 ±0.24	**63.39 ±0.13**
Elastic transform	35.07	38.49 ±0.12	35.72 ±0.09	**39.57 ±0.39**
Pixelate	43.86	48.37 ±0.17	44.32 ±0.10	**50.45 ±0.16**
JPEG compression	39.11	44.42 ±0.09	43.44 ±0.11	**47.45 ±0.14**
Average	44.59	50.14	47.58	**52.52**

Table-A II-15 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-B/32 as visual encoder

	CLIP	LAME	TENT	TPT (BS=32)	CLIPArTT
Original	88.74	89.36 ±0.06	**91.69 ±0.10**	88.06 ±0.06	90.04 ±0.13
CIFAR 10.1	83.25	81.22 ±0.33	**87.60 ±0.45**	81.80 ±0.27	86.35 ±0.27
Gaussian Noise	35.27	24.71 ±0.11	41.27 ±0.27	33.90 ±0.08	**59.90 ±0.36**
Shot noise	39.67	27.44 ±0.09	47.20 ±0.23	38.20 ±0.02	**62.77 ±0.07**
Impulse Noise	42.61	31.38 ±0.15	48.58 ±0.31	37.66 ±0.20	**56.02 ±0.16**
Defocus blur	69.76	62.45 ±0.44	**77.12 ±0.16**	67.83 ±0.28	76.74 ±0.05
Glass blur	42.40	29.96 ±0.06	52.65 ±0.30	38.81 ±0.12	**61.77 ±0.16**
Motion blur	63.97	54.00 ±0.36	71.25 ±0.09	63.39 ±0.13	**76.01 ±0.19**
Zoom blur	69.83	61.97 ±0.36	76.20 ±0.19	68.95 ±0.16	**77.40 ±0.20**
Snow	71.78	64.61 ±0.48	**78.29 ±0.20**	70.16 ±0.10	77.29 ±0.16
Frost	72.86	65.17 ±0.17	**79.84 ±0.09**	72.39 ±0.22	79.20 ±0.08
Fog	67.04	59.13 ±0.49	**77.39 ±0.01**	64.31 ±0.28	75.74 ±0.14
Brightness	81.87	80.05 ±0.23	**87.78 ±0.03**	81.30 ±0.18	86.59 ±0.16
Contrast	64.37	56.91 ±0.37	**79.47 ±0.11**	62.26 ±0.31	77.82 ±0.14
Elastic transform	60.83	53.89 ±0.20	70.00 ±0.25	56.43 ±0.27	**70.20 ±0.01**
Pixelate	50.53	39.67 ±0.34	63.74 ±0.18	42.80 ±0.40	**66.52 ±0.13**
JPEG compression	55.48	47.24 ±0.14	62.64 ±0.14	53.67 ±0.25	**63.51 ±0.14**
Average	59.22	50.57	67.56	56.80	**71.17**

Table-A II-16 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with ViT-B/32 as visual encoder

	CLIP	LAME	TENT	TPT (BS=32)	CLIPArTT
Original	61.68	58.27 ±0.17	69.74 ±0.16	63.78 ±0.28	**69.79 ±0.04**
Gaussian Noise	14.80	12.72 ±0.04	14.38 ±0.14	14.03 ±0.10	**25.32 ±0.14**
Shot noise	16.03	13.78 ±0.08	17.34 ±0.27	15.25 ±0.17	**27.90 ±0.05**
Impulse Noise	13.85	7.82 ±0.14	10.03 ±0.13	13.01 ±0.13	**25.62 ±0.09**
Defocus blur	36.74	33.38 ±0.11	49.05 ±0.07	37.60 ±0.17	**49.88 ±0.23**
Glass blur	14.19	9.00 ±0.05	3.71 ±0.07	16.41 ±0.02	**27.89 ±0.03**
Motion blur	36.14	32.79 ±0.13	46.62 ±0.27	37.52 ±0.23	**47.93 ±0.14**
Zoom blur	40.24	37.57 ±0.15	51.84 ±0.15	42.99 ±0.11	**52.70 ±0.06**
Snow	38.95	35.49 ±0.18	46.71 ±0.21	42.35 ±0.13	**49.72 ±0.01**
Frost	40.56	37.22 ±0.21	44.90 ±0.27	43.31 ±0.14	**49.63 ±0.12**
Fog	38.00	35.94 ±0.09	47.31 ±0.04	38.81 ±0.17	**48.77 ±0.04**
Brightness	48.18	44.93 ±0.08	60.58 ±0.18	50.23 ±0.11	**61.27 ±0.08**
Contrast	29.53	27.52 ±0.06	45.90 ±0.11	28.09 ±0.09	**48.55 ±0.24**
Elastic transform	26.33	24.01 ±0.02	33.09 ±0.08	28.12 ±0.15	**37.45 ±0.08**
Pixelate	21.98	19.55 ±0.13	26.47 ±0.09	20.43 ±0.14	**33.88 ±0.14**
JPEG compression	25.91	21.77 ±0.14	29.89 ±0.07	28.82 ±0.09	**36.07 ±0.32**
Average	29.43	26.23	35.19	30.46	**41.51**

APPENDIX III

WATT: WEIGHT AVERAGE TEST-TIME ADAPTATION OF CLIP - SUPPLEMENTARY MATERIAL

David Osowiechi*, Mehrdad Noori*, Gustavo A. Vargas Hakim
Moslem Yazdanpanah, Ali Bahri, Milad Cheraghalikhani, Sahar Dastani
Farzad Beizaee, Ismail Ben Ayed, Christian Desrosiers

Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

*Equal Contribution
Poster at The Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS) December 2024

1. Implementation

Our proposed WATT method is implemented in Python using the PyTorch (version 2.0.1) framework. All experiments were conducted on an NVIDIA V100 32 GB GPU. However, due to the effectively and lightweight nature of our method, it can be executed on less powerful GPUs. Specifically, the adaptation with a batch size of 128 using the ViT/B32 backbone requires up to 4 GB of memory, making it feasible to use on a wide range of GPUs. For fairness and consistency, we re-implemented and ran all other methods, including CLIPArTT, TENT, and TPT, in the same environment. Each experiment was performed three times to ensure reliability (three trials per experiment). To facilitate reproducibility, we provide the original implementation and detailed step-by-step instructions in our repository, accessible via *this github link*.

2. Template details

The CLIP paper identifies 80 templates that enhance model robustness and performance. They ultimately conclude that 7 of these templates best summarize their model (see *this link*). In our work, we use these 7 generic templates and add the common one, "`a photo of {}`," based on their optimization. These templates are not specifically linked to the content of the images.

3. Dataset details

In our investigation, we use the VisDA-C dataset, which challenges models with two distinct domain shifts: the simulated shift and the video shift. The simulated shift includes 152,397 3D-rendered images across 12 diverse classes, while the video shift comprises 72,372 YouTube video frames spanning the same categories. This dataset addresses the diversity of imagery types applicable to a model, posing a significant challenge.

Moreover, we evaluate our proposed method on three other datasets: PACS, VLCS, and OfficeHome. These datasets help understand various domain shifts, including texture and style variations. The PACS dataset consists of 9,991 images across four domains (Art, Cartoons, Photos, Sketches) and seven classes. The VLCS dataset contains 10,729 images across four domains (Caltech101, LabelMe, SUN09, VOC2007) and five classes. Lastly, the OfficeHome dataset includes 15,588 images across four domains (Art, Clipart, Product, Real) and 65 classes. Evaluating across these distinct domain shifts showcases the generalizability of our method. These datasets are more representative of real-world scenarios compared to CIFAR, with complex domain shifts.

4. Computational Cost

We conduct a thorough evaluation under consistent conditions using an NVIDIA A6000 GPU within the same Python environment. The Table III-1 provided compares the adaptation time, memory usage, and the number of learnable parameters for various TTA methods, including our proposed WATT method.The table clearly demonstrates that WATT-S, a sequential implementation of WATT, maintains competitive adaptation time and memory usage compared to other methods like TENT and ClipArTT, which are efficient but lack the robustness of WATT's method. Additionally, the table highlights that WATT-P, with parallel model training, offers a faster adaptation time than WATT-P with a for-loop implementation, albeit at the cost of higher memory usage. It's important to note that methods like DiffTPT Feng *et al.* (2023) and MEMO Zhang *et al.* (2022), which show significantly higher adaptation times,

employ off-the-shelf diffusion models and AugMix augmentation, respectively, resulting in time-consuming processes that may be impractical for real-world scenarios. In contrast, the effectiveness of our WATT-S method makes it better suited for scenarios where a robust, rapid, and resource-efficient adaptation is crucial.

Table-A III-1 Comparison of computational cost of different methods

Method	Adaptation Time	Memory	Percentage of Learnable Parameters
WATT-S	2.34 s	1.5 GB	0.026%
WATT-P	23.2 s (23.2/8)	1.5 GB (8 x 1.5 GB)	0.026% (x8)
TENT	0.28 s	1.5 GB	0.026%
ClipArTT	0.55 s	1.7 GB	0.026%
SAR	0.42 s	1.4 GB	0.026%
MEMO	165 s	2 GB	0.026%
DiffTPT	8.2 + 0.26 s	8.7 GB + 1.7 GB	0.001%

5. Pseudo-code of our both methods

In Algorithms 1 and 2, we compare the two variants of WATT: one with Parallel MTWA (WATT-P) and the other with Sequential MTWA (WATT-S). The WATT-P model recalibrates its parameters for each template using the average parameters of $m-1$, whereas the WATT-S model updates its parameters solely at the start of each new iteration m.

Algorithm 1 WATT-P - model f, parameter θ

for $m \in \{1, 2, \ldots, M\}$ **do**
  $\theta_{\text{avg}} \leftarrow \frac{1}{H}\sum_{h=1}^{H}\theta_h$
  for $h \in \{1, 2, \ldots, H\}$ **do**
    $f \leftarrow f_{\theta_{\text{avg}}}$
    for $l \in \{1, 2, \ldots, L\}$ **do**
      $\theta_h \leftarrow \mathcal{L}_{\text{TTA}}(f_{\theta_{\text{avg}}}(\texttt{template}_h))$
    end for
  end for
end for

Algorithm 2 WATT-S - model f, parameter θ

for $m \in \{1, 2, \ldots, M\}$ **do**
  $\theta_{\text{avg}} \leftarrow \frac{1}{H}\sum_{h=1}^{H}\theta_h$
  $f \leftarrow f_{\theta_{\text{avg}}}$
  for $h \in \{1, 2, \ldots, H\}$ **do**
    for $l \in \{1, 2, \ldots, L\}$ **do**
      $\theta_h \leftarrow \mathcal{L}_{\text{TTA}}(f_{\theta_{\text{avg}}}(\texttt{template}_h))$
    end for
  end for
end for

6. Additional Ablation Studies

Cross Entropy vs Entropy Minimization. Two unsupervised loss functions were integrated into previous TTA methods: classical entropy minimization and the loss introduced by CLIPArTT Vargas Hakim *et al.* (2024), where predictions are utilized as pseudo labels for cross-entropy computation. In Table III-2, a comparison between these two loss functions is presented across the original CIFAR-10 dataset and various corruptions from CIFAR-10-C. It is observed that, for these specific corruptions, entropy minimization generally outperforms with the different templates employed, except for *Gaussian Noise*. However, upon assessing the weighted average accuracy, computed after 10 iterations for each template, cross-entropy consistently emerges as the superior option. The marginal impact of the weighted average on entropy minimization suggests that, irrespective of the template used, the model updates in a consistent direction to enhance confidence, rendering cross-entropy the preferred choice for subsequent experiments.

Table-A III-2 Comparison of accuracy (%) using entropy minimization (TENT) or cross-entropy (CE) on CIFAR-10 and some corruptions of CIFAR-10-C datasets with ViT-B/32 encoder on different templates (please see Fig. 4.1a) and the weight average

Dataset	Loss	T^0	T^1	T^2	T^3	T^4	T^5	T^6	T^7	WA
CIFAR-10	TENT	91.69	91.97	91.69	90.28	91.16	92.11	91.98	89.14	90.60
	CE	89.8	90.37	90.5	88.42	89.93	89.95	90.13	88.54	**91.05**
Gaussian Noise	TENT	41.27	37.16	46.39	51.31	39.27	32.51	49.7	42.96	47.08
	CE	60.19	61.01	61.17	58.24	58.84	58.35	59.62	61.13	**63.84**
Defocus Blur	TENT	77.12	77.13	78.7	76.09	76.85	76.59	77.86	74.31	76.21
	CE	77.23	77.07	78	75.98	76.39	77.45	77.08	75.59	**78.94**
Snow	TENT	78.29	79.54	80.09	75.39	78.97	78.52	78.78	75.82	77.24
	CE	76.57	77.36	77.93	75.08	77.45	77.09	77.05	75.57	**79.79**
JPEG Compression	TENT	62.64	65.83	64.27	59.49	62.78	64.19	62.62	63.39	65.31
	CE	64.65	65.36	65.24	64.16	64.18	64.36	64.78	65.32	**67.36**

7. Experiments with another VLM

We investigate if our TTA method is working with other VLMs like SigLip, so we compare if after adaptation with our method the performance improves compared to the baseline in Table III-3.

Table-A III-3 Performance comparison of SigLip and WATT-S on different datasets

Dataset	SigLip	WATT-S
CIFAR-10	66.35	75.02 $\pm$0.05
CIFAR-10.1	57.30	65.87 $\pm$0.21
CIFAR-10-C	37.52	45.29 $\pm$0.13
CIFAR-100	33.97	65.87 $\pm$0.21
CIFAR-100-C	14.43	20.05 $\pm$0.05

8. Experiments on other Visual Encoders

We replicated the experiments from the main paper using alternative visual encoders, ViT-B/16 and ViT-L/14.

Performance evaluation in the presence of natural or no domain shift. As indicated in the main results, employing WATT, both with Parallel and Sequential MTWA, consistently enhances performance alongside the baseline. This pattern persists across different visual encoders, as shown in Tables **??**. Although WATT consistently outperforms the baseline and TPT, TENT and CLIPArTT may occasionally yield superior results depending on the visual encoder used.

Performance evaluation in the presence of common corruptions. Table **??** demonstrates a consistent trend where both WATT methods consistently outperform alternative methods across various corruptions and class numbers. Upon closer examination of Table **??**, specifically with ViT-B/16 as the visual encoder, Sequential MTWA exhibits a significant performance advantage, surpassing the baseline by 16.07% and the leading method in the *state-of-the-art* by 3.00%. This trend becomes even more pronounced with an increased number of classes, where Sequential MTWA surpasses the baseline and CLIPArTT by 16.94% and 8.87%, respectively.

Performance analysis under simulated and video shifts. Our study reveals substantial accuracy improvements on the 3D (simulated shift) and YT (video shift) partitions of VisDA-C when employing different backbones. This enhancement is particularly notable with our proposed WATT method compared to pure CLIP. Notably, the WATT-S variant achieves the highest accuracy across both the 3D and YT partitions, outperforming various adaptation approaches including TENT, TPT, and CLIPArTT. Detailed comparisons can be found in Tables III-5 and III-6.

Performance analysis under texture and style shifts. Findings across the OfficeHome, PACS, and VLCS datasets are detailed in Tables III-5 and III-6. Across these varied domains, our WATT method demonstrates consistent performance enhancements, as evidenced by both its WATT-P and WATT-S variants. These improvements underscore the efficacy of our approach in mitigating the complexities of texture and style shifts, which pose particular challenges compared to other forms of domain shift.

Table-A III-5 Accuracy (%) on different domains of VisDA-C, OfficeHome, PACS and VLCS datasets with ViT-B/16 as visual encoder

Dataset	Domain	CLIP	TENT	TPT	CLIPArTT	WATT-P	WATT-S
VisDA-C	3D (trainset)	87.16	87.57 ±0.01	84.04 ±0.03	87.58 ±0.00	87.61 ±0.01	**87.72 ±0.02**
	YT (valset)	86.61	**86.81 ±0.00**	85.90 ±0.11	86.60 ±0.01	86.66 ±0.00	86.75 ±0.04
	Mean	86.89	87.19	84.97	87.09	87.14	**87.24**
OfficeHome	Art	79.30	79.26 ±0.14	**81.97 ±0.17**	79.34 ±0.05	80.37 ±0.25	80.43 ±0.09
	Clipart	65.15	65.64 ±0.05	67.01 ±0.21	65.69 ±0.11	**68.59 ±0.13**	68.26 ±0.11
	Product	87.34	87.49 ±0.02	**89.00 ±0.06**	87.35 ±0.07	88.15 ±0.07	88.02 ±0.08
	Real World	89.31	89.50 ±0.04	89.66 ±0.06	89.29 ±0.03	**90.18 ±0.03**	90.14 ±0.06
	Mean	80.28	80.47	**81.91**	80.42	81.82	81.71
PACS	Art	97.44	97.54 ±0.02	95.10 ±0.41	97.64 ±0.02	97.49 ±0.08	**97.66 ±0.08**
	Cartoon	97.38	97.37 ±0.04	91.42 ±0.22	97.37 ±0.02	97.47 ±0.04	**97.51 ±0.02**
	Photo	**99.58**	**99.58 ±0.00**	98.56 ±0.40	**99.58 ±0.00**	**99.58 ±0.00**	**99.58 ±0.00**
	Sketch	86.06	86.37 ±0.05	87.23 ±0.06	86.79 ±0.04	**89.73 ±0.16**	89.56 ±0.14
	Mean	95.12	95.22	93.08	95.35	96.07	**96.08**
VLCS	Caltech101	**99.43**	**99.43 ±0.00**	97.62 ±0.12	**99.43 ±0.00**	99.36 ±0.00	99.36 ±0.00
	LabelMe	67.75	67.31 ±0.14	49.77 ±0.03	67.74 ±0.10	67.55 ±0.39	**68.59 ±0.25**
	SUN09	71.74	71.57 ±0.15	71.56 ±0.86	71.67 ±0.01	74.75 ±0.07	**75.16 ±0.12**
	VOC2007	84.90	**85.10 ±0.11**	71.17 ±0.70	84.73 ±0.08	82.53 ±0.10	83.24 ±0.05
	Mean	80.96	80.85	72.53	80.89	81.05	**81.59**

Table-A III-6 Accuracy (%) on different domains of VisDA-C, OfficeHome, PACS and VLCS datasets with ViT-L/14 as visual encoder

Dataset	Domain	CLIP	TENT	TPT	CLIPArTT	WATT-S
VisDA-C	3D (trainset)	91.24	91.40 ±0.01	90.65 ±0.00	91.34 ±0.00	**91.71 ±0.00**
	YT (valset)	85.62	85.77 ±0.01	85.41 ±0.06	85.61 ±0.01	**86.80 ±0.01**
	Mean	88.43	88.59	88.03	88.48	**89.26**
OfficeHome	Art	82.47	82.61 ±0.15	**86.76 ±0.26**	82.35 ±0.19	84.43 ±0.20
	Clipart	72.20	72.51 ±0.03	74.76 ±0.07	72.41 ±0.06	**75.43 ±0.08**
	Product	90.94	90.97 ±0.02	**92.42 ±0.07**	90.94 ±0.06	91.88 ±0.05
	Real World	92.72	92.75 ±0.02	92.95 ±0.16	92.63 ±0.03	**94.06 ±0.02**
	Mean	84.58	84.71	86.72	84.58	86.45
PACS	Art	98.68	**98.83 ±0.00**	94.82 ±0.34	98.76 ±0.02	98.68 ±0.00
	Cartoon	97.74	97.74 ±0.00	95.65 ±0.19	97.74 ±0.00	**97.90 ±0.02**
	Photo	99.54	99.54 ±0.03	99.44 ±0.03	**99.54 ±0.00**	**99.64 ±0.00**
	Sketch	93.28	93.51 ±0.04	92.72 ±0.15	93.26 ±0.02	**93.80 ±0.02**
	Mean	97.31	97.41	95.66	97.33	**97.51**
VLCS	Caltech101	99.43	99.43 ±0.00	97.86 ±0.43	99.43 ±0.00	**99.51 ±0.00**
	LabelMe	69.22	69.07 ±0.12	52.54 ±0.20	**69.32 ±0.15**	62.76 ±0.13
	SUN09	68.06	68.23 ±0.03	69.49 ±0.32	67.89 ±0.07	**72.21 ±0.15**
	VOC2007	83.99	84.08 ±0.15	76.16 ±0.63	**83.89 ±0.13**	83.02 ±0.12
	Mean	80.18	**80.20**	74.01	80.13	79.38

9. Detailed Experimental Findings

This section provides extensive tables with detailed information on the results, which were summarized in the main body of the paper.

Table-A III-7 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with different text ensemble at test time. (WA after 10 iter)×1

Dataset		single_temp	text_avg
CIFAR-10		90.87 ±0.10	**91.08 ±0.06**
CIFAR-10.1		86.80 ±0.19	**86.85 ±0.18**
CIFAR-10-C	Gaussian Noise	61.20 ±0.05	**62.09 ±0.15**
	Shot noise	63.16 ±0.09	**63.51 ±0.03**
	Impulse Noise	55.29 ±0.22	**56.04 ±0.16**
	Defocus blur	78.03 ±0.12	**78.66 ±0.07**
	Glass blur	62.7 ±0.24	**63.35 ±0.25**
	Motion blur	76.33 ±0.11	**76.96 ±0.14**
	Zoom blur	78.29 ±0.05	**79.08 ±0.15**
	Snow	78.65 ±0.21	**78.95 ±0.05**
	Frost	79.49 ±0.11	**79.95 ±0.06**
	Fog	77.21 ±0.03	**77.72 ±0.09**
	Brightness	86.60 ±0.07	**86.98 ±0.06**
	Contrast	79.22 ±0.01	**79.62 ±0.04**
	Elastic transform	71.17 ±0.25	**71.61 ±0.09**
	Pixelate	67.59 ±0.08	**68.70 ±0.15**
	JPEG compression	66.26 ±0.00	**66.72 ±0.01**
	Mean	72.08	**72.66**

Table-A III-8 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with different text ensemble at test time. (WA after 10 iter)×1

Dataset		single_temp	text_avg
CIFAR-100		69.79 ±0.20	**70.30 ±0.11**
CIFAR-100-C	Gaussian Noise	27.17 ±0.22	**28.08 ±0.21**
	Shot noise	29.69 ±0.20	**30.47 ±0.19**
	Impulse Noise	25.28 ±0.10	**26.37 ±0.28**
	Defocus blur	49.83 ±0.11	**50.52 ±0.04**
	Glass blur	27.83 ±0.03	**28.25 ±0.06**
	Motion blur	47.77 ±0.21	**47.89 ±0.21**
	Zoom blur	52.90 ±0.16	**53.05 ±0.10**
	Snow	50.31 ±0.03	**50.22 ±0.17**
	Frost	50.79 ±0.07	**51.08 ±0.09**
	Fog	48.70 ±0.16	**48.48 ±0.21**
	Brightness	61.22 ±0.06	**61.56 ±0.16**
	Contrast	47.87 ±0.17	**47.90 ±0.14**
	Elastic transform	37.55 ±0.13	**37.93 ±0.19**
	Pixelate	33.81 ±0.06	**34.56 ±0.14**
	JPEG compression	36.09 ±0.13	**37.30 ±0.18**
	Mean	41.79	**42.24**

Table-A III-9 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with different averaging

Dataset		Text avg.	Output avg.	Weight avg. (ours)		
				(after 10 iter)×1	(after 1 iter)×10	(after 2 iter)×5
CIFAR-100		69.46 ±0.13	70.32 ±0.10	70.3 ±0.11	**70.85 ±0.08**	70.74 ±0.20
CIFAR-100-C	Gaussian Noise	27.67 ±0.11	28.58 ±0.04	28.08 ±0.21	31.67 ±0.10	**32.07 ±0.23**
	Shot noise	30.18 ±0.06	31.05 ±0.13	30.47 ±0.19	34.26 ±0.28	**34.36 ±0.11**
	Impulse Noise	25.79 ±0.02	26.86 ±0.07	26.37 ±0.28	30.12 ±0.12	**30.33 ±0.03**
	Defocus blur	49.51 ±0.04	51.04 ±0.02	50.52 ±0.04	52.76 ±0.25	**52.99 ±0.16**
	Glass blur	27.88 ±0.22	28.72 ±0.08	28.25 ±0.06	31.95 ±0.08	**32.15 ±0.30**
	Motion blur	46.68 ±0.05	48.30 ±0.19	47.89 ±0.21	50.46 ±0.10	**50.53 ±0.12**
	Zoom blur	52.05 ±0.07	53.72 ±0.11	53.05 ±0.10	55.13 ±0.29	**55.30 ±0.22**
	Snow	49.40 ±0.18	51.01 ±0.13	50.22 ±0.17	52.60 ±0.26	**52.77 ±0.15**
	Frost	49.68 ±0.04	51.50 ±0.06	51.08 ±0.09	53.30 ±0.21	**53.79 ±0.31**
	Fog	47.36 ±0.17	48.67 ±0.22	48.48 ±0.21	51.35 ±0.08	**51.49 ±0.21**
	Brightness	60.42 ±0.12	61.74 ±0.31	61.56 ±0.16	63.23 ±0.12	**63.57 ±0.21**
	Contrast	46.86 ±0.05	48.14 ±0.10	47.90 ±0.14	52.40 ±0.23	**52.76 ±0.27**
	Elastic transform	37.00 ±0.37	38.55 ±0.23	37.93 ±0.19	40.97 ±0.11	**40.90 ±0.43**
	Pixelate	33.65 ±0.12	34.63 ±0.17	34.56 ±0.14	40.32 ±0.08	**40.97 ±0.16**
	JPEG compression	36.38 ±0.11	37.67 ±0.23	37.30 ±0.18	39.35 ±0.19	**39.59 ±0.08**
	Mean	41.37	42.68	42.24	45.32	**45.57**

Table-A III-10 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with different averaging

Dataset		Text avg.	Output avg.	Weight avg. (ours)		
				(after 10 iter)×1	(after 1 iter)×10	(after 2 iter)×5
CIFAR-10		90.58 ±0.03	90.90 ±0.03	91.08 ±0.06	**91.39 ±0.14**	91.05 ±0.06
CIFAR-10.1		85.78 ±0.25	86.77 ±0.08	86.85 ±0.18	**88.02 ±0.18**	86.98 ±0.31
CIFAR-10-C	Gaussian Noise	61.23 ±0.13	62.22 ±0.12	62.09 ±0.15	63.42 ±0.07	**63.84 ±0.24**
	Shot noise	62.88 ±0.15	63.98 ±0.17	63.51 ±0.03	64.93 ±0.13	**65.28 ±0.21**
	Impulse Noise	54.71 ±0.07	56.41 ±0.11	56.04 ±0.16	58.37 ±0.37	**58.64 ±0.11**
	Defocus blur	77.93 ±0.12	78.63 ±0.18	78.66 ±0.07	**79.11 ±0.17**	78.94 ±0.12
	Glass blur	62.37 ±0.18	63.32 ±0.07	63.35 ±0.25	64.67 ±0.18	**65.12 ±0.07**
	Motion blur	75.55 ±0.19	76.97 ±0.05	76.96 ±0.14	77.56 ±0.12	**77.81 ±0.14**
	Zoom blur	77.86 ±0.06	78.90 ±0.18	79.08 ±0.15	**79.76 ±0.03**	79.32 ±0.07
	Snow	77.77 ±0.03	78.92 ±0.03	78.95 ±0.05	**79.89 ±0.26**	79.79 ±0.06
	Frost	78.51 ±0.09	79.67 ±0.09	79.95 ±0.06	80.52 ±0.04	**80.54 ±0.12**
	Fog	76.04 ±0.17	77.54 ±0.10	77.72 ±0.09	78.44 ±0.21	**78.53 ±0.22**
	Brightness	86.08 ±0.13	86.75 ±0.04	86.98 ±0.06	**87.32 ±0.14**	87.11 ±0.11
	Contrast	77.87 ±0.02	79.48 ±0.07	79.62 ±0.04	80.77 ±0.35	**81.20 ±0.22**
	Elastic transform	69.98 ±0.16	71.20 ±0.22	71.61 ±0.09	72.52 ±0.19	**72.66 ±0.15**
	Pixelate	66.78 ±0.29	68.27 ±0.17	68.70 ±0.15	70.50 ±0.20	**71.11 ±0.13**
	JPEG compression	65.62 ±0.28	66.78 ±0.08	66.72 ±0.01	67.05 ±0.10	**67.36 ±0.28**
	Mean	71.41	72.60	72.66	73.66	**73.82**

Table-A III-11 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-B/16 as visual encoder for different number of batches

Dataset		CLIP	BS = 1	BS = 2	BS = 4	BS = 8	BS = 16	BS = 32	BS = 64	BS = 128
CIFAR-10		88.74	89.87	89.39 ±0.02	89.16 ±0.07	88.93 ±0.16	89.14 ±0.04	89.51 ±0.12	90.16 ±0.13	91.05 ±0.06
CIFAR-10.1		83.25	84.55	84.32 ±0.15	83.88 ±0.17	84.12 ±0.37	84.35 ±0.21	84.87 ±0.16	85.52 ±0.30	86.98 ±0.31
CIFAR-10-C	Gaussian Noise	35.27	38.55	43.85 ±0.26	45.41 ±0.10	47.95 ±0.15	51.79 ±0.27	56.35 ±0.11	60.87 ±0.33	63.84 ±0.24
	Shot noise	39.67	42.57	46.87 ±0.25	47.95 ±0.15	49.13 ±0.14	52.57 ±0.03	56.96 ±0.10	61.84 ±0.06	65.28 ±0.21
	Impulse Noise	42.61	42.92	47.94 ±0.29	48.20 ±0.18	48.69 ±0.11	50.53 ±0.18	53.32 ±0.19	55.81 ±0.11	58.64 ±0.11
	Defocus blur	69.76	72.29	72.80 ±0.13	72.95 ±0.13	72.57 ±0.20	73.71 ±0.18	75.28 ±0.18	77.37 ±0.08	78.94 ±0.12
	Glass blur	42.40	44.15	48.15 ±0.15	47.69 ±0.07	48.96 ±0.04	52.59 ±0.19	57.83 ±0.24	62.16 ±0.20	65.12 ±0.07
	Motion blur	63.97	66.37	67.53 ±0.07	67.22 ±0.01	68.00 ±0.12	69.20 ±0.11	71.60 ±0.06	74.75 ±0.09	77.81 ±0.14
	Zoom blur	69.83	71.50	72.60 ±0.14	72.30 ±0.04	72.39 ±0.01	73.19 ±0.06	75.01 ±0.09	77.03 ±0.27	79.32 ±0.07
	Snow	71.78	73.72	74.46 ±0.16	73.97 ±0.19	74.12 ±0.05	74.62 ±0.22	76.06 ±0.06	77.64 ±0.06	79.79 ±0.06
	Frost	72.86	75.67	76.50 ±0.23	75.98 ±0.11	75.55 ±0.16	76.32 ±0.13	77.67 ±0.03	78.82 ±0.20	80.54 ±0.12
	Fog	67.04	68.88	70.25 ±0.02	69.94 ±0.06	69.88 ±0.09	71.02 ±0.15	73.10 ±0.02	75.95 ±0.04	78.53 ±0.22
	Brightness	81.87	83.52	83.73 ±0.10	83.38 ±0.03	83.31 ±0.05	83.51 ±0.11	84.49 ±0.13	85.40 ±0.07	87.11 ±0.11
	Contrast	64.37	67.02	69.67 ±0.13	68.64 ±0.14	69.08 ±0.06	71.11 ±0.17	74.58 ±0.14	78.25 ±0.22	81.20 ±0.22
	Elastic transf.	60.83	62.04	64.25 ±0.13	63.50 ±0.40	63.46 ±0.10	64.65 ±0.28	66.63 ±0.21	69.58 ±0.18	72.66 ±0.15
	Pixelate	50.53	51.65	55.18 ±0.26	55.47 ±0.14	56.30 ±0.29	58.88 ±0.21	63.00 ±0.08	67.43 ±0.11	71.11 ±0.13
	JPEG compr.	55.48	58.12	60.17 ±0.04	59.44 ±0.18	59.74 ±0.04	61.20 ±0.09	63.15 ±0.15	65.32 ±0.16	67.36 ±0.28
	Mean	59.22	61.26	63.60	63.47	63.94	65.66	68.34	71.21	73.82

Table-A III-12 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-B/16 as visual encoder for different number of templates randomly picked over 5 runs

Dataset		T=1	T=2	T=4	T=6	T=8
CIFAR-10		89.42 ±0.84	90.74 ±0.30	90.98 ±0.14	91.34 ±0.16	91.05 ±0.06
CIFAR-10.1		85.08 ±0.59	86.49 ±0.59	87.20 ±0.48	87.53 ±0.21	86.98 ±0.31
CIFAR-10-C	Gaussian Noise	59.82 ±1.43	62.05 ±0.62	62.79 ±0.18	63.49 ±0.27	63.84 ±0.24
	Shot noise	62.32 ±1.32	63.35 ±0.43	64.73 ±0.31	65.02 ±0.10	65.28 ±0.21
	Impulse Noise	54.07 ±0.17	56.83 ±0.33	57.53 ±0.47	58.37 ±0.08	58.64 ±0.11
	Defocus blur	77.09 ±0.24	78.32 ±0.32	78.92 ±0.16	79.17 ±0.26	78.94 ±0.12
	Glass blur	60.64 ±0.29	63.77 ±0.43	64.42 ±0.68	64.64 ±0.39	65.12 ±0.07
	Motion blur	74.60 ±0.50	77.02 ±0.32	77.70 ±0.26	77.73 ±0.12	77.81 ±0.14
	Zoom blur	77.40 ±0.29	78.93 ±0.46	79.28 ±0.54	79.33 ±0.24	79.32 ±0.07
	Snow	76.96 ±1.04	78.83 ±0.31	79.47 ±0.29	79.69 ±0.33	79.79 ±0.06
	Frost	77.62 ±0.86	79.27 ±0.45	80.04 ±0.26	80.46 ±0.17	80.54 ±0.12
	Fog	75.32 ±0.57	77.27 ±0.39	78.00 ±0.17	78.55 ±0.29	78.53 ±0.22
	Brightness	85.13 ±0.58	86.74 ±0.22	87.07 ±0.20	87.13 ±0.21	87.11 ±0.11
	Contrast	77.18 ±0.68	79.74 ±0.31	80.32 ±0.07	80.69 ±0.12	81.20 ±0.22
	Elastic transform	69.39 ±0.39	71.40 ±0.24	72.25 ±0.14	72.28 ±0.34	72.66 ±0.15
	Pixelate	66.26 ±0.76	68.86 ±0.68	69.47 ±0.39	71.00 ±0.57	71.11 ±0.13
	JPEG compression	64.58 ±0.58	66.28 ±0.14	66.82 ±0.24	67.16 ±0.18	67.36 ±0.28
	Mean	70.56	72.58	73.25	73.65	73.82

Table-A III-15 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with ViT-B/16 as visual encoder

Dataset		CLIP	TENT	TPT (BS=32)	CLIPArTT	WATT-P	WATT-S
CIFAR-100		64.76	71.73 ±0.14	67.15 ±0.23	71.34 ±0.07	72.98 ±0.07	72.85 ±0.15
CIFAR-100-C	Gaussian Noise	15.88	12.28 ±0.20	15.43 ±0.03	19.01 ±0.24	34.23 ±0.03	35.95 ±0.27
	Shot noise	17.49	15.07 ±0.21	16.88 ±0.07	20.27 ±0.21	36.68 ±0.1	37.96 ±0.15
	Impulse Noise	21.43	13.13 ±0.16	22.12 ±0.15	17.66 ±0.10	43.17 ±0.35	44.62 ±0.2
	Defocus blur	40.10	50.35 ±0.03	41.08 ±0.22	49.86 ±0.13	53.13 ±0.12	53.80 ±0.12
	Glass blur	13.48	4.84 ±0.14	18.43 ±0.15	18.34 ±0.31	32.53 ±0.03	33.39 ±0.11
	Motion blur	39.82	49.85 ±0.37	40.85 ±0.26	50.00 ±0.09	51.63 ±0.06	52.72 ±0.30
	Zoom blur	45.45	54.76 ±0.04	46.77 ±0.06	54.13 ±0.08	56.81 ±0.11	57.51 ±0.09
	Snow	42.77	52.38 ±0.18	47.24 ±0.18	52.80 ±0.27	56.04 ±0.06	56.73 ±0.27
	Frost	45.39	51.66 ±0.04	48.61 ±0.14	49.56 ±0.08	56.00 ±0.11	56.48 ±0.34
	Fog	38.98	50.74 ±0.14	39.92 ±0.16	49.92 ±0.11	52.88 ±0.33	53.83 ±0.19
	Brightness	52.55	64.26 ±0.09	55.83 ±0.10	63.76 ±0.13	65.58 ±0.07	66.67 ±0.19
	Contrast	33.32	48.69 ±0.08	33.13 ±0.16	47.86 ±0.02	52.90 ±0.06	55.06 ±0.15
	Elastic transform	24.39	33.56 ±0.28	27.36 ±0.10	32.93 ±0.23	39.82 ±0.21	40.37 ±0.26
	Pixelate	21.89	36.20 ±0.28	21.26 ±0.10	39.49 ±0.21	45.10 ±0.06	47.02 ±0.04
	JPEG compression	27.21	30.80 ±0.05	30.97 ±0.10	35.56 ±0.23	41.43 ±0.18	42.13 ±0.21
	Mean	32.01	37.90	33.73	40.08	47.86	48.95

Table-A III-16 Accuracy (%) on CIFAR-10, CIFAR-10.1 and CIFAR-10-C datasets with ViT-L/14 as visual encoder

Dataset		CLIP	TENT	TPT (BS=32)	CLIPArTT	WATT-P	WATT-S
CIFAR-10		95.36	96.13 ±0.06	95.18 ±0.02	95.16 ±0.03	95.91 ±0.10	95.71 ±0.03
CIFAR-10.1		91.20	92.22 ±0.25	91.32 ±0.12	91.02 ±0.02	92.97 ±0.13	92.10 ±0.33
CIFAR-10-C	Gaussian Noise	64.64	68.87 ±0.20	64.44 ±0.11	70.04 ±0.31	72.81 ±0.09	72.73 ±0.03
	Shot noise	67.82	71.95 ±0.06	66.81 ±0.19	71.44 ±0.16	74.45 ±0.16	74.60 ±0.03
	Impulse Noise	78.21	80.22 ±0.19	76.46 ±0.17	79.42 ±0.15	81.36 ±0.09	80.95 ±0.15
	Defocus blur	80.73	83.10 ±0.03	79.01 ±0.23	81.75 ±0.19	83.20 ±0.10	83.15 ±0.18
	Glass blur	50.29	57.12 ±0.07	49.64 ±0.23	58.13 ±0.23	61.51 ±0.07	62.35 ±0.15
	Motion blur	80.75	82.69 ±0.11	78.85 ±0.04	80.76 ±0.12	82.60 ±0.13	82.61 ±0.12
	Zoom blur	82.75	84.91 ±0.08	82.32 ±0.13	83.39 ±0.05	85.76 ±0.06	85.44 ±0.13
	Snow	83.01	85.99 ±0.11	82.69 ±0.10	84.48 ±0.07	84.91 ±0.13	85.61 ±0.15
	Frost	84.90	87.15 ±0.12	84.63 ±0.08	85.21 ±0.06	87.17 ±0.13	86.88 ±0.04
	Fog	78.44	81.30 ±0.07	77.56 ±0.17	79.27 ±0.07	81.80 ±0.11	81.79 ±0.09
	Brightness	91.67	93.07 ±0.04	90.94 ±0.04	91.87 ±0.09	92.78 ±0.05	92.59 ±0.16
	Contrast	84.20	87.93 ±0.04	82.88 ±0.09	86.19 ±0.06	87.54 ±0.12	87.38 ±0.02
	Elastic transform	65.45	69.96 ±0.12	64.81 ±0.14	67.43 ±0.24	71.19 ±0.07	71.25 ±0.09
	Pixelate	75.10	77.89 ±0.05	72.92 ±0.12	77.11 ±0.10	77.88 ±0.13	77.67 ±0.16
	JPEG compression	72.58	75.49 ±0.07	71.18 ±0.19	74.46 ±0.11	75.88 ±0.16	75.84 ±0.18
	Mean	76.04	79.18	75.01	78.06	80.05	80.06

Table-A III-17 Accuracy (%) on CIFAR-100 and CIFAR-100-C datasets with ViT-L/14 as visual encoder

Dataset		CLIP	TENT	TPT,(BS=32)	CLIPArTT	WATT-P	WATT-S
CIFAR-100		73.28	78.03 ±0.08	76.85 ±0.06	79.42 ±0.08	79.33 ±0.05	78.85 ±0.19
CIFAR-100-C	Gaussian Noise	30.55	36.93 ±0.03	36.10 ±0.11	41.46 ±0.15	43.99 ±0.13	44.13 ±0.11
	Shot noise	34.58	40.96 ±0.16	38.23 ±0.13	44.27 ±0.09	46.28 ±0.22	46.63 ±0.17
	Impulse Noise	44.89	49.09 ±0.14	49.69 ±0.21	51.44 ±0.23	56.15 ±0.04	56.26 ±0.22
	Defocus blur	48.88	55.23 ±0.07	50.43 ±0.19	56.55 ±0.22	57.46 ±0.01	57.66 ±0.42
	Glass blur	23.46	27.02 ±0.23	24.35 ±0.22	30.47 ±0.14	32.54 ±0.12	33.54 ±0.16
	Motion blur	50.83	56.03 ±0.20	51.94 ±0.04	56.98 ±0.18	58.22 ±0.10	57.81 ±0.05
	Zoom blur	56.02	61.19 ±0.10	56.96 ±0.16	62.56 ±0.04	62.94 ±0.02	62.74 ±0.06
	Snow	49.03	55.60 ±0.09	54.89 ±0.11	58.81 ±0.11	60.68 ±0.06	61.04 ±0.27
	Frost	53.27	58.21 ±0.15	58.15 ±0.33	60.38 ±0.23	62.34 ±0.14	62.76 ±0.22
	Fog	48.51	53.37 ±0.25	49.26 ±0.13	54.38 ±0.04	54.71 ±0.31	54.70 ±0.13
	Brightness	60.53	67.34 ±0.17	66.60 ±0.10	69.63 ±0.14	71.52 ±0.07	71.60 ±0.09
	Contrast	50.24	59.91 ±0.13	53.64 ±0.24	63.39 ±0.13	62.77 ±0.22	63.95 ±0.04
	Elastic transform	35.07	38.49 ±0.12	35.72 ±0.09	39.57 ±0.39	41.28 ±0.25	41.27 ±0.15
	Pixelate	43.86	48.37 ±0.17	44.32 ±0.10	50.45 ±0.16	51.15 ±0.15	51.22 ±0.13
	JPEG compression	39.11	44.42 ±0.09	43.44 ±0.11	47.45 ±0.14	49.40 ±0.17	49.78 ±0.18
	Mean	44.59	50.14	47.58	52.52	54.10	54.34

BIBLIOGRAPHY

Aneja, J., Schwing, A., Kautz, J. & Vahdat, A. (2021). A contrastive learning approach for training variational autoencoder priors. *Advances in neural information processing systems*, 34, 480–493.

Angelopoulos, A. N. & Bates, S. (2021). A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*.

Bartler, A., Bühler, A., Wiewel, F., Döbler, M. & Yang, B. (2021). MT3: Meta Test-Time Training for Self-Supervised Test-Time Adaption. *CoRR*, abs/2103.16201. Retrieved from: https://arxiv.org/abs/2103.16201.

Boudiaf, M., Mueller, R., Ben Ayed, I. & Bertinetto, L. (2022). Parameter-free Online Test-time Adaptation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8344–8353.

Brody, J. (2023). On the Potential of CLIP for Compositional Logical Reasoning. *arXiv preprint arXiv:2308.15887*.

Cha, J., Chun, S., Lee, K., Cho, H.-C., Park, S., Lee, Y. & Park, S. (2021). Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34, 22405–22418.

Chapelle, O., Scholkopf, B. & Zien, A. (2006). Semi-supervised learning. 2006. *Cambridge, Massachusettes: The MIT Press View Article*, 2.

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. (2020, 13–18 Jul). A Simple Framework for Contrastive Learning of Visual Representations. *Proceedings of the 37th International Conference on Machine Learning*, 119(Proceedings of Machine Learning Research), 1597–1607. Retrieved from: https://proceedings.mlr.press/v119/chen20j.html.

Cheraghalikhani, M., Noori, M., Osowiechi, D., Hakim, G. A. V., Ayed, I. B. & Desrosiers, C. (2024). Structure-Aware Feature Stylization for Domain Generalization. *Computer Vision and Image Understanding*, 244, 104016.

Chidlovskii, B., Clinchant, S. & Csurka, G. (2016). Domain adaptation in the absence of source domain data. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 451–460.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255.

Dinh, L., Sohl-Dickstein, J. & Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fang, C., Xu, Y. & Rockmore, D. N. (2013). Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1657–1664.

Feng, C.-M., Yu, K., Liu, Y., Khan, S. & Zuo, W. (2023, October). Diverse Data Augmentation with Diffusions for Effective Test-time Prompt Tuning. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2704-2714.

Gandelsman, Y., Sun, Y., Chen, X. & Efros, A. A. (2022). Test-Time Training with Masked Autoencoders. *Advances in Neural Information Processing Systems*. Retrieved from: https://openreview.net/forum?id=SHMi1b7sjXk.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P. & Wilson, A. G. (2018). Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *Advances in Neural Information Processing Systems*, 31. Retrieved from: https://proceedings.neurips.cc/paper_files/paper/2018/file/be3087e74e9100d4bc4c6268cdbe8456-Paper.pdf.

Goyal, S., Sun, M., Raghunanthan, A. & Kolter, Z. (2022). Test-time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems*.

Guo, Z., Zhang, R., Qiu, L., Ma, X., Miao, X., He, X. & Cui, B. (2023). CALIP: zero-shot enhancement of CLIP with parameter-free attention. *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, (AAAI'23/IAAI'23/EAAI'23). doi: 10.1609/aaai.v37i1.25152.

Gutmann, M. & Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304.

Guzhov, A., Raue, F., Hees, J. & Dengel, A. (2022). Audioclip: Extending clip to image, text and audio. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 976–980.

He, K., Zhang, X., Ren, S. & Sun, J. (2016, June). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P. & Girshick, R. (2022). Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009.

Hendrycks, D. & Dietterich, T. (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.

Hinton, G. E. & Roweis, S. (2002). Stochastic neighbor embedding. *Advances in neural information processing systems*, 15.

Iscen, A., Tolias, G., Avrithis, Y. & Chum, O. (2019). Label propagation for deep semi-supervised learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5070–5079.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. & Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z. & Duerig, T. (2021). Scaling up visual and vision-language representation learning with noisy text supervision. *International conference on machine learning*, pp. 4904–4916.

Karmanov, A., Guan, D., Lu, S., El Saddik, A. & Xing, E. (2024, June). Efficient Test-Time Adaptation of Vision-Language Models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14162-14171.

Khurana, A., Paul, S., Rai, P., Biswas, S. & Aggarwal, G. (2021). Sita: single image test-time adaptation. *arXiv:2112.02355 [cs]*. Retrieved from: http://arxiv.org/abs/2112.02355. arXiv: 2112.02355.

Kim, D., Wang, K., Sclaroff, S. & Saenko, K. (2022). A Broad Study of Pre-training for Domain Generalization and Adaptation. *Computer Vision – ECCV 2022*, pp. 621–638.

Kingma, D. P. & Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. *Advances in Neural Information Processing Systems*, 31. Retrieved from: https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y. et al. (2023). Segment anything. *arXiv preprint arXiv:2304.02643*.

Lai, Z., Vesdapunt, N., Zhou, N., Wu, J., Huynh, C. P., Li, X., Fu, K. K. & Chuah, C.-N. (2023, October). PADCLIP: Pseudo-labeling with Adaptive Debiasing in CLIP for Unsupervised Domain Adaptation. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16155-16165.

Lee, J., Das, D., Choo, J. & Choi, S. (2023, October). Towards Open-Set Test-Time Adaptation Utilizing the Wisdom of Crowds in Entropy Minimization. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16380-16389.

Li, D., Yang, Y., Song, Y.-Z. & Hospedales, T. M. (2017). Deeper, broader and artier domain generalization. *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550.

Li, J., Li, D., Xiong, C. & Hoi, S. C. H. (2022). BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pp. 12888–12900.

Liang, J., Hu, D. & Feng, J. (2020). Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *International conference on machine learning*, pp. 6028–6039.

Liang, J., He, R. & Tan, T. (2023). A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*.

Lin, G., Lai, H., Pan, Y. & Yin, J. (2023). Improving Entropy-Based Test-Time Adaptation from a Clustering View. *arXiv preprint arXiv:2310.20327*.

Lin, Z., Geng, S., Zhang, R., Gao, P., de Melo, G., Wang, X., Dai, J., Qiao, Y. & Li, H. (2022). Frozen clip models are efficient video learners. *European Conference on Computer Vision*, pp. 388–404.

Liu, J., Zhang, Y., Chen, J.-N., Xiao, J., Lu, Y., A Landman, B., Yuan, Y., Yuille, A., Tang, Y. & Zhou, Z. (2023). Clip-driven universal model for organ segmentation and tumor detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21152–21164.

Liu, Y., Kothari, P., van Delft, B., Bellot-Gurlet, B., Mordan, T. & Alahi, A. (2021). TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive? *Neural Information Processing Systems (NeurIPS)*.

Ma, X., Zhang, J., Guo, S. & Xu, W. (2024). Swapprompt: Test-time prompt adaptation for vision-language models. *Advances in Neural Information Processing Systems*, 36.

Mnih, A. & Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26.

Mummadi, C. K., Hutmacher, R., Rambach, K., Levinkov, E., Brox, T. & Metzen, J. H. (2021). Test-Time Adaptation to Distribution Shift by Confidence Maximization and Input Transformation. arXiv. Retrieved from: https://arxiv.org/abs/2106.14999.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B. & Snoek, J. (2021). Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv:2006.10963 [cs, stat]*. Retrieved from: http://arxiv.org/abs/2006.10963. arXiv: 2006.10963.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. & Ng, A. Y. (2011). Reading Digits in Natural Images with Unsupervised Feature Learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. Retrieved from: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P. & Tan, M. (2022, 17–23 Jul). Efficient Test-Time Model Adaptation without Forgetting. *Proceedings of the 39th International Conference on Machine Learning*, 162(Proceedings of Machine Learning Research), 16888–16905. Retrieved from: https://proceedings.mlr.press/v162/niu22a.html.

Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P. & Tan, M. (2023). Towards Stable Test-Time Adaptation in Dynamic Wild World. Retrieved from: https://arxiv.org/abs/2302.12400.

Noori, M., Cheraghalikhani, M., Bahri, A., Hakim, G. A. V., Osowiechi, D., Yazdanpanah, M., Ayed, I. B. & Desrosiers, C. (2024). TFS-ViT: Token-Level Feature Stylization for Domain Generalization. *Pattern Recognition*, 149, 110213.

Noori, M., Cheraghalikhani, M., Bahri, A., Hakim, G. A. V., Osowiechi, D., Yazdanpanah, M., Ayed, I. B. & Desrosiers, C. (2025). FDS: Feedback-Guided Domain Synthesis with Multi-Source Conditional Diffusion Models for Domain Generalization. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Oord, A. v. d., Li, Y. & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Osowiechi, D., Vargas Hakim, G. A., Noori, M., Cheraghalikhani, M., Ayed, I. & Desrosiers, C. (2023, jan). TTTFlow: Unsupervised Test-Time Training with Normalizing Flow. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2125-2126. doi: 10.1109/WACV56688.2023.00216.

Osowiechi, D., Hakim, G. A. V., Noori, M., Cheraghalikhani, M., Bahri, A., Yazdanpanah, M., Ben Ayed, I. & Desrosiers, C. (2024a, June). NC-TTT: A Noise Constrastive Approach for Test-Time Training. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6078-6086.

Osowiechi, D., Noori, M., Vargas Hakim, G., Yazdanpanah, M., Bahri, A., Cheraghalikhani, M., Dastani, S., Beizaee, F., Ayed, I. & Desrosiers, C. (2024b). WATT: Weight Average Test Time Adaptation of CLIP. *Advances in Neural Information Processing Systems*, 37, 48015–48044. Retrieved from: https://proceedings.neurips.cc/paper_files/paper/2024/file/55d16334943f8728073e17139e5baa3d-Paper-Conference.pdf.

Peng, X., Usman, B., Kaushik, N., Wang, D., Hoffman, J. & Saenko, K. (2018, June). VisDA: A Synthetic-to-Real Benchmark for Visual Domain Adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Prakash, A., Boochoon, S., Brophy, M., Acuna, D., Cameracci, E., State, G., Shapira, O. & Birchfield, S. (2019). Structured domain randomization: Bridging the reality gap by context-aware synthetic data. *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7249–7255.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. et al. (2021). Learning transferable visual models from natural language supervision. *International conference on machine learning*, pp. 8748–8763.

Recht, B., Roelofs, R., Schmidt, L. & Shankar, V. (2018). Do CIFAR-10 Classifiers Generalize to CIFAR-10? *CoRR*, abs/1806.00451. Retrieved from: http://arxiv.org/abs/1806.00451.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W. & Bethge, M. (2020). Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 11539–11551. Retrieved from: https://proceedings.neurips.cc/paper/2020/file/85690f81aadc1749175c187784afc9ee-Paper.pdf.

Shafer, G. & Vovk, V. (2008). A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9(3).

Shu, M., Nie, W., Huang, D.-A., Yu, Z., Goldstein, T., Anandkumar, A. & Xiao, C. (2022). Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 35, 14274–14289.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A. & Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. *International Conference on Machine Learning (ICML)*.

Vargas Hakim, G. A., Osowiechi, D., Noori, M., Cheraghalikhani, M., Bahri, A., Ben Ayed, I. & Desrosiers, C. (2023). ClusT3: Information Invariant Test-Time Training. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6136–6145.

Vargas Hakim, G. A., Osowiechi, D., Noori, M., Cheraghalikhani, M., Bahri, A., Yazdanpanah, M., Ayed, I. B. & Desrosiers, C. (2024). CLIPArTT: Light-weight Adaptation of CLIP to New Domains at Test Time. *arXiv preprint arXiv:2405.00754*. Retrieved from: https://arxiv.org/abs/2405.00754.

Venkateswara, H., Eusebio, J., Chakraborty, S. & Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5018–5027.

Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V. & Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B. & Darrell, T. (2020). Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W. & Yu, P. (2022a). Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*.

Wang, Q., Fink, O., Van Gool, L. & Dai, D. (2022b). Continual test-time domain adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211.

Wilson, G. & Cook, D. J. (2020). A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5), 1–46.

Wu, Q., Yue, X. & Sangiovanni-Vincentelli, A. (2021). Domain-agnostic test-time adaptation by prototypical training with auxiliary data. *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

Zhai, X., Mustafa, B., Kolesnikov, A. & Beyer, L. (2023). Sigmoid Loss for Language Image Pre-Training. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1037–1047.

Zhai, X., Mustafa, B., Kolesnikov, A. & Beyer, L. (2025). SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding. *arXiv preprint arXiv:2502.14786*.

Zhang, J., Liu, S., Song, J., Zhu, T., Xu, Z. & Song, M. (2024). Lookaround Optimizer: *k* steps around, 1 step average. *Advances in Neural Information Processing Systems*, 36.

Zhang, M., Levine, S. & Finn, C. (2022). Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35, 38629–38642.

Zhang, R., Fang, R., Gao, P., Zhang, W., Li, K., Dai, J., Qiao, Y. & Li, H. (2021). Tip-Adapter: Training-free CLIP-Adapter for Better Vision-Language Modeling. *arXiv preprint arXiv:2111.03930*.

Zhou, K., Yang, Y., Qiao, Y. & Xiang, T. (2020). Domain Generalization with MixStyle. *International Conference on Learning Representations*.

Zhou, Y., Ren, J., Li, F., Zabih, R. & Lim, S. N. (2023). Test-Time Distribution Normalization for Contrastively Learned Visual-language Models. *Advances in Neural Information Processing Systems*, 36, 47105–47123. Retrieved from: https://proceedings.neurips.cc/paper_files/paper/2023/file/931db0b5a61f9db6c97c7e4bf068147d-Paper-Conference.pdf.