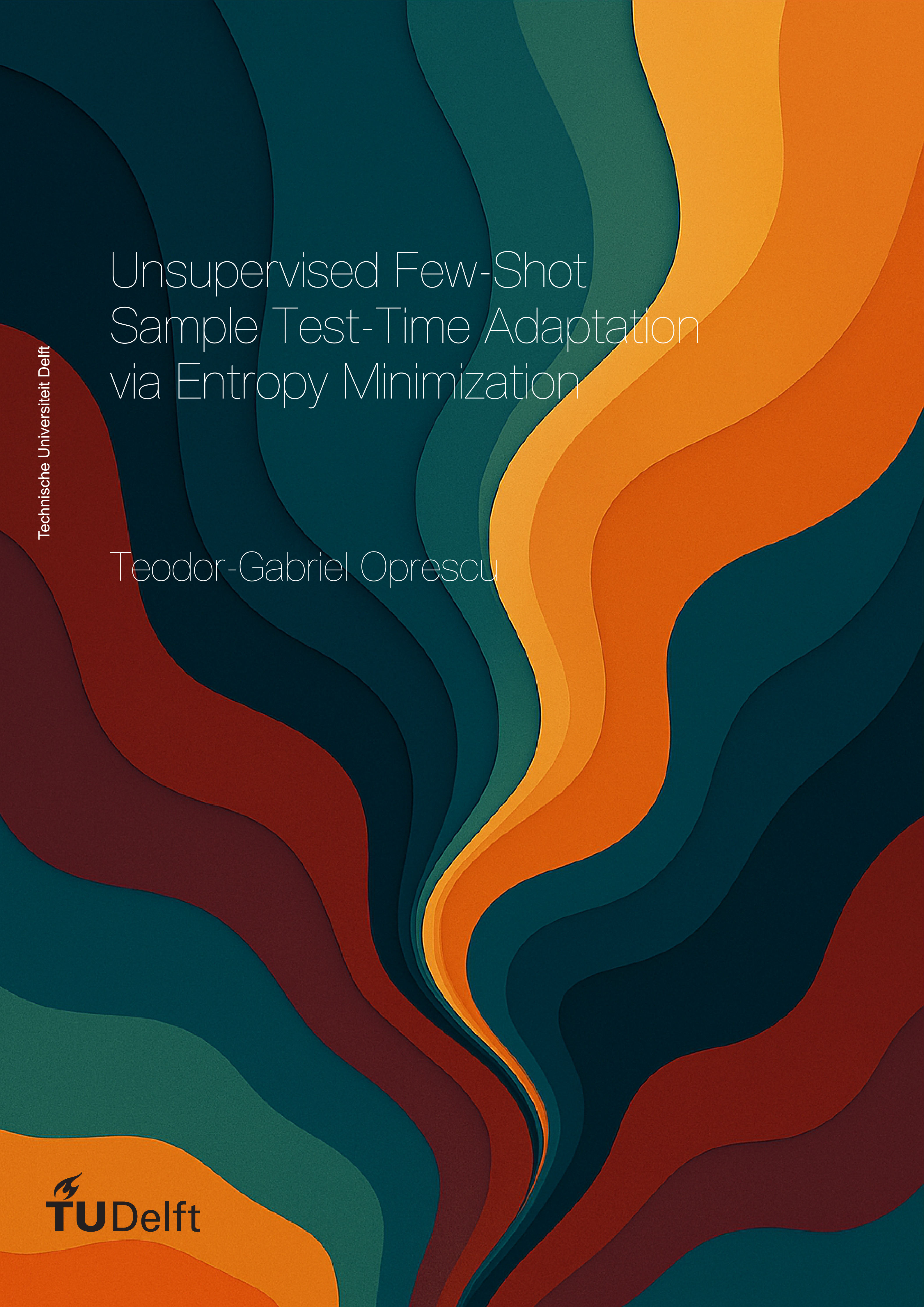


Technische Universiteit Delft

Unsupervised Few-Shot Sample Test-Time Adaptation via Entropy Minimization

Teodor-Gabriel Oprescu

TU Delft

Unsupervised Few-Shot Sample Test-Time Adaptation via Entropy Minimization

by

Teodor-Gabriel Oprescu

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 15, 2025 at 11:00 AM.

Supervisors:	Dr. J. Van Gemert
Graduation committee:	Dr. J. Van Gemert (Associate Professor, TU Delft)
	Dr. J. M. Castaneda (Associate Professor, TU Delft)
Project duration:	November, 2024 – July, 2025
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

Preface

This thesis represents the culmination of my journey to obtain the degree of Master of Science at TU Delft, a period that has been as intellectually demanding as it has been personally transformative.

I owe profound gratitude to my supervisor, Prof. Dr. Jan van Gemert, whose guidance has been invaluable. I will always remember his remarkable ability to express the most complex computer vision concepts and disarm entire arguments through the simplest of words. This clarity of thought and expression is something I aspire to master in my own career.

My deepest appreciation goes to the PhD students who have been my mentors throughout this journey: Sayak Mukherjee, Zhi-Yi Lin, and Thomas Markhorst. Their patience has been extraordinary, guiding me through countless iterations, debugging sessions, and moments of doubt. They supported me through my most erratic phases and challenging periods, always ready with constructive feedback and encouragement. Their mentorship transformed what could have been an overwhelming experience into a journey of genuine scientific discovery.

To my family, whose unwavering support has been my anchor, and to my friends in Delft who transformed a foreign city into a second home, your presence made this journey not just bearable, but meaningful.

Teodor-Gabriel Oprescu
Delft, July 2025

Contents

1 Introduction

Machine learning models operate under a fundamental assumption: test data will follow the same statistical distribution as training data. Distribution shift occurs when this assumption breaks, causing deployed models to encounter data systematically different from their training experience [1]. A medical imaging classifier trained on high-resolution MRI scans from research hospitals encounters lower-quality images from portable scanners in rural clinics. The feature distributions differ, decision boundaries no longer separate classes correctly, and diagnostic accuracy degrades. This degradation is not a failure of the model architecture or training procedure but an inevitable consequence of deployment in diverse, uncontrolled environments.

Test-time adaptation emerged to address distribution shift by modifying models during deployment. Unlike traditional approaches requiring retraining on labeled target data, test-time adaptation works with unlabeled samples as they arrive during inference. Methods update batch normalization parameters through entropy minimization [2], adjust feature statistics to match test distributions [3], or learn input-specific prompts [4]. Each approach restores performance by aligning the model with shifted data. The field assumes models can be modified at deployment time through gradient computation, parameter updates, or architectural access.

Real-world deployments prohibit such modifications. Medical devices certified under regulatory frameworks cannot change parameters without invalidating safety approvals. Cloud vision services operate through encrypted APIs that accept images and return predictions while keeping models completely opaque. Edge devices compile neural networks into optimized firmware where parameters become immutable. Military and financial systems enforce cryptographic verification ensuring models remain unmodified. These constraints define standard deployment practice, not exceptional cases.

We present Input Transformation via Entropy Minimization (ITEM), the first test-time adaptation method that preserves complete model integrity. Rather than modifying models to handle corrupted data, ITEM modifies corrupted data to match model expectations. The approach leverages a fundamental property of neural network confidence: well-calibrated models produce low-entropy predictions on familiar inputs and high-entropy predictions on out-of-distribution samples. By learning input transformations that minimize prediction entropy, we guide shifted samples back toward the training distribution without any model modification.

Our validation demonstrates ITEM's effectiveness under extreme data scarcity that mirrors real deployment constraints. Training models with minimal samples per class and adapting with single calibration examples, ITEM successfully recovers performance where parameter-based methods fail catastrophically. The approach achieves dramatically better sample efficiency than existing methods, revealing fundamental advantages of input-space optimization over parameter updates.

Chapter 2 presents our complete methodology and experimental validation through the research paper. Readers seeking background on neural networks, distribution shift, and existing adaptation methods should first read Chapter 3. These foundations explain why adapting inputs rather than models is both necessary for real deployments and theoretically sound. Together, the chapters show that test-time adaptation without model modification is not just possible but essential for practical computer vision systems.

2

Research Paper

Unsupervised Few-Shot Sample Test-Time Adaptation via Entropy Minimization

Teodor-Gabriel Oprescu Zhi-Yi Lin Sayak Mukherjee
Thomas Markhorst Jan van Gemert

Delft University of Technology

Abstract

Test-time adaptation methods assume privileged access to model internals: parameters for fine-tuning, statistics for recalibration, or architectural components for modification. This assumption fails when models are deployed as certified systems, encrypted services, or under regulatory constraints that prohibit parameter changes. We present ITEM (Input Transformation via Entropy Minimization), the first preparation-agnostic sample adaptation method for test-time adaptation. ITEM learns input transformations that minimize prediction entropy using only gradient signals through frozen models, exploiting the principle that well-calibrated models produce low-entropy outputs on familiar data. Unlike existing sample adaptation methods that require specialized training procedures or parameter updates, ITEM works with any pre-trained model without modification or preparation requirements. Using scalar transformations as proof of concept, we demonstrate adaptation under extreme data scarcity: models trained on 10 samples per class and adapted with single calibration samples. ITEM significantly reduces performance degradation from corruption while existing methods fail or show negligible improvement. Our results establish that effective test-time adaptation is possible without model modification, architectural knowledge, or training preparation, opening new possibilities for adapting deployed models under real-world constraints.

1 Introduction

Machine learning models operate under a fundamental assumption: the data encountered during deployment will follow the same distribution as the training data. This assumption, while mathematically convenient, rarely holds in practice. Deployed vision systems face distribution shifts arising from environmental changes, sensor degradation, demographic variations, and countless other factors. In medical imaging, models trained on high-quality data from specialized centers often experience significant accuracy drops when deployed in diverse clinical settings, with degradation patterns well-documented across domain adaptation studies [8]. Autonomous vehicles' perception systems, reliable in clear conditions, experience detection failures exceeding 40% in adverse weather [11]. These shifts affect critical applications where retraining is often impossible due to computational, regulatory, or data access constraints.

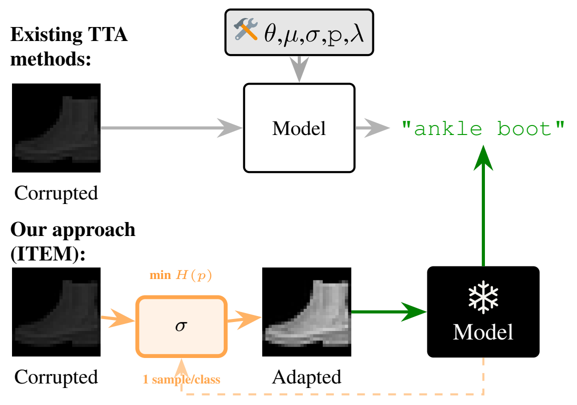


Figure 1. Test-time adaptation methods categorized by model internals requirements. **Top**: Existing approaches require modification of model internals. Model adaptation manipulates parameters (θ) via gradients (∇). Normalization adaptation modifies batch statistics (μ, σ). Inference adaptation adjusts prediction logits (p). Prompt adaptation requires architectural embedding points (λ). **Bottom**: ITEM uses sample adaptation, learning input transformation σ through entropy minimization feedback. Using single calibration samples per class, ITEM recovers accuracy comparable to methods requiring model internals while operating under strict deployment constraints, using only gradient signals through frozen models.

Test-time adaptation (TTA) has emerged as a promising approach to address distribution shift during deployment. By adapting models as they encounter new data, TTA methods can restore performance without requiring retraining or access to the original training data. The field has shown considerable progress, with methods adapting various model components to handle shifted distributions. However, these advances rest on a critical assumption: the ability to modify model internals. Methods update parameters [16, 21, 22, 28], adjust statistics [15, 20, 25], or access architectural components [9, 16]. This assumption fails for a large class of real-world deployments.

Real-world deployments increasingly constrain model modification through technical and regulatory requirements. Models in medical devices certified under FDA 510(k) or European CE marking cannot be modified post-deployment without invalidating safety certifications. Cloud vision services like Google Cloud Vision API [5], Amazon Rekognition [1], and Azure Computer Vision [19] provide state-of-the-art capabilities through RESTful endpoints that accept images and return predictions, but prohibit any parameter updates. Models embedded in encrypted firmware or compiled for edge deployment similarly prevent modification while allowing inference. In these settings, adaptation cannot rely on changing model parameters, accessing internal statistics, or modifying architectural components. The model exists as an immutable prediction function.

We pose a different question: rather than adapting the model to corrupted data, can we adapt the corrupted data to the model? This perspective shift leads to ITEM (Input Transformation via Entropy Minimization), the first preparation-agnostic sample adaptation method that preserves model integrity while achieving effective adaptation. Our key insight exploits a fundamental property of neural network calibration: models trained on clean data produce low-entropy predictions on familiar inputs and high-entropy predictions on shifted data. By learning input transformations that minimize prediction entropy, we guide corrupted samples toward the training distribution without any model modification Fig. 1. Unlike existing sample adaptation methods that require specialized training procedures or architectural knowledge, ITEM works with any pre-trained model, requiring only the ability to compute gradients with respect to input transformations.

Our contributions are threefold:

1. First preparation-agnostic sample adaptation method: We introduce ITEM, which adapts through input transformation without requiring specialized training, architectural constraints, or parameter modification. This enables adaptation for arbitrary pre-trained models while preserving certifications and deployment constraints.
2. Extreme sample efficiency: With just 1 sample per class, ITEM recovers performance lost to distribution shift, demonstrating that entropy signals from minimal data suffice for effective adaptation when optimization is properly constrained to the input space.
3. Insight on adaptation objectives: We discover that entropy-optimal transformations systematically undercorrect compared to pixel-reconstruction optima, revealing that models prioritize maintaining inputs within learned decision boundaries over perfect intensity restoration.

2 Related Work

2.1 Domain Adaptation

Domain adaptation addresses distribution shift between training and deployment phases [2, 4]. Classical approaches minimize distribution divergence through various techniques: Maximum Mean Discrepancy (MMD) [7] measures differences between feature distributions, adversarial training [4] learns domain-invariant representations, and optimal transport [3] finds minimal-cost mappings between distributions. While theoretically grounded for covariate shift [26], these methods require simultaneous access to source and target data during training, making them unsuitable for already-deployed models.

Source-free domain adaptation (SFDA) emerged to address scenarios where source data is unavailable [12, 16]. SHOT [16] generates pseudo-labels through entropy minimization and diversity regularization. Hypothesis transfer [17] extends this approach by transferring decision boundaries without source access. Neighborhood structure exploitation [32] leverages intrinsic data relationships, while [10] explores model distillation techniques. Despite eliminating source data requirements, all SFDA methods modify model parameters during inference, which is incompatible with deployment constraints that prohibit parameter updates.

2.2 Test-Time Adaptation

Test-time adaptation (TTA) modifies models during inference to handle distribution shift without access to source data. Given a model f_θ trained on source distribution p_s, TTA adapts to shifted target distribution p_t using only unlabeled target samples during deployment. Recent surveys [18, 31] categorize TTA methods along two primary axes: preparation requirements and adaptation target. Preparation requirements divide methods into three categories as shown in Table 1: preparation-agnostic methods that work with any pre-trained model, training-preparation methods requiring specific training procedures, and training-and-data preparation methods that additionally require augmented training data. The adaptation target determines what component is modified:

Table 1. Taxonomy of adaptation settings, extending Table 1 from Wang et al. [28] to incorporate the three test-time adaptation categories identified in recent surveys. We adopt the modern terminology introduced in [31] where "preparation-agnostic TTA" corresponds to TENT's "fully test-time" setting. Notation: (X^s, Y^s) and (X^t, Y^t) denote source and target data with labels; X^t denotes unlabeled target data; f_θ represents the model with parameters θ; $\mathcal{L}$ denotes the task loss (e.g., cross-entropy); aug denotes augmentation strategies during training. $\mathcal{L}_{\text{align}}$: domain alignment losses (e.g., MMD, adversarial). $\mathcal{L}_{\text{aux}}$: auxiliary self-supervised losses (e.g., rotation prediction). $\mathcal{L}_{\text{prep}}$: preparation-specific objectives including meta-learning. $\mathcal{S}$: method-specific statistics (e.g., running moments, prototypes). $\theta' \subseteq \theta$: parameter subset. **Model Access** indicates methods requiring modification of model internals (parameters, statistics, or architecture). ✓denotes violation of deployment constraints.

Setting	Source Data	Target Data	Training Loss	Test Loss	Model Access
Fine-tuning	(X^s, Y^s)	(X^t, Y^t)	$\mathcal{L}(f_\theta(X^s), Y^s)$	-	✓
Domain Adaptation	(X^s, Y^s)	X^t	$\mathcal{L}(f_\theta(X^s), Y^s) + \mathcal{L}_{\text{align}}$	-	✓
Training&Data-prep TTA	(X^s, Y^s) + aug	X^t	$\mathcal{L}(f_\theta(X^s), Y^s) + \mathcal{L}_{\text{prep}}$	$\mathcal{L}(f_\theta(X^t))$	✓
Training-prep TTA	(X^s, Y^s)	X^t	$\mathcal{L}(f_\theta(X^s), Y^s) + \mathcal{L}_{\text{aux}}$	$\mathcal{L}(f_\theta(X^t))$	✓
Preparation-agnostic TTA	-	X^t	-	$\mathcal{L}(f_\theta(X^t))$	✓
ITEM (Ours)	-	X^t	-	$\mathcal{L}(f_\theta(\sigma(X^t)))$	-

model parameters, inference procedures, normalization statistics, input samples, or prompts.

Preparation-agnostic methods adapt standard pretrained models without special training. Entropy minimization [28] updates BatchNorm parameters, while AdaBN [15] directly replaces source statistics with target statistics. Recent methods add complexity: sharpness awareness [22], Fisher regularization [21], and continual adaptation [29]. These methods achieve strong adaptation performance but fundamentally require parameter modification, violating deployment constraints in certified systems.

Sample adaptation methods modify individual test instances rather than model parameters. Existing approaches include DUA [20], which requires architectural modifications during training to integrate differentiable augmentation modules. Test-Time Training [27] necessitates auxiliary rotation prediction tasks during the training phase. Style transfer approaches [23] depend on domain-specific feature statistics extracted during training. MEMO [33] performs augmentation-based adaptation but updates full model parameters. Critically, all existing sample adaptation methods require either training preparation or parameter modification, preventing deployment on arbitrary pre-trained models under strict constraints.

No existing method provides preparation-agnostic adaptation without requiring parameter modification or architectural knowledge. TENT [28] requires gradient-based updates to BatchNorm parameters, AdaBN [15] needs direct access to normalization layers, and MEMO [33] performs full model updates through augmentation consistency. Every approach violates at least one requirement for constrained deployment scenarios. This gap motivates ITEM, which achieves effective adaptation through input transformation alone, working with any pre-trained model while preserving all deployment constraints Tab. 2.

Table 2. Deployment constraint violations by test-time adaptation methods. P: Parameter modification, A: Architectural knowledge/requirements, F: Feature/internal state access. ✓indicates violation. ITEM achieves adaptation without violating any deployment constraints.

Method	P	A	F
ITEM (Ours)	-	-	-
Parameter Modification Methods			
TENT [28]	✓	✓	-
SAR [22]	✓	-	-
EATA [21]	✓	✓	-
CoTTA [29]	✓	✓	-
MEMO [33]	✓	-	-
Architecture-Dependent Methods			
AdaBN [15]	-	✓	-
T3A [9]	✓	✓	✓
SHOT [16]	✓	✓	✓
Pseudo-Labeling Methods			
RPL [24]	✓	-	-
PseudoLabeling [14]	✓	-	-

3 Method

We formalize input-only adaptation for frozen models accessed through gradient computation but without parameter modification privileges. Unlike existing test-time adaptation methods that update model internals, our approach operates entirely through input transformation.

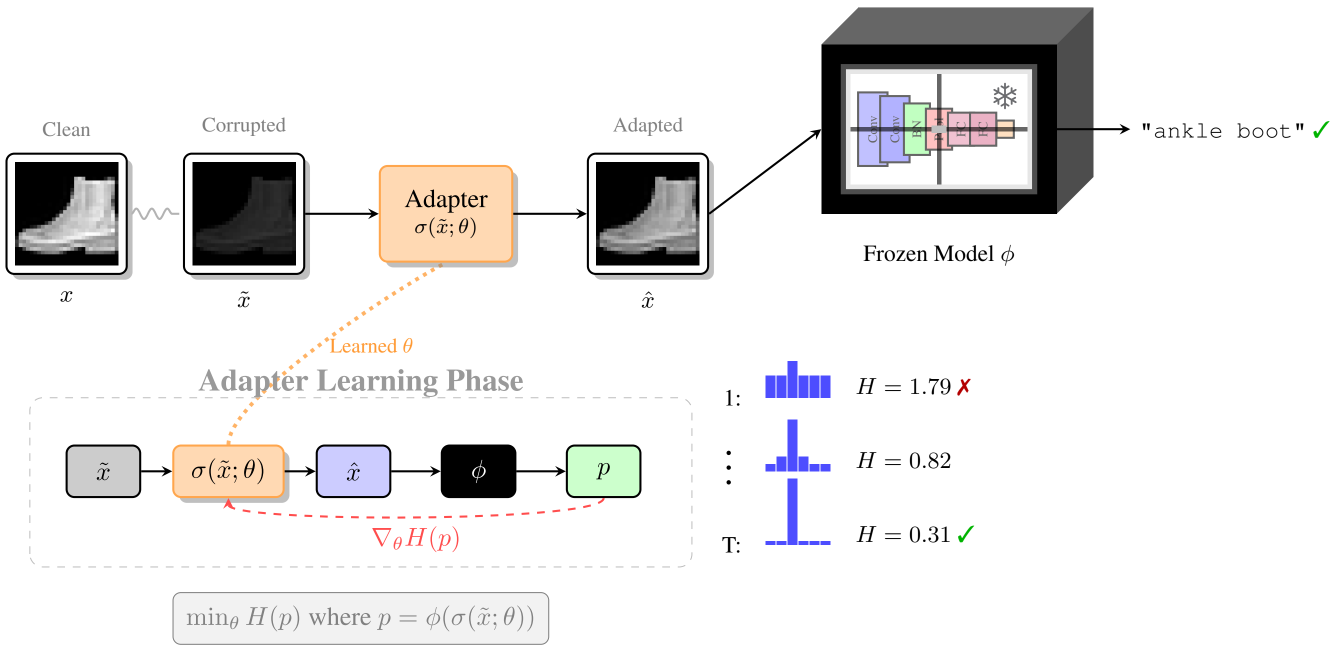


Figure 2. Input Transformation via Entropy Minimization (ITEM) adapts to distribution shifts through input transformation alone. When corrupted test images arrive, an adapter $\sigma(\cdot;\theta)$ transforms them before passing to the frozen model ϕ. The adapter parameters θ are optimized to minimize prediction entropy $H(p)$. The bottom panel shows the adapter learning phase: corrupted calibration samples $\tilde{x}$ are transformed by σ, passed through the frozen model to obtain predictions p, and the entropy gradient $\nabla_\theta H(p)$ updates the adapter without modifying ϕ. The entropy progression (right) demonstrates convergence from high uncertainty to confident predictions.

Algorithm 1 Input Transformation via Entropy Minimization (ITEM)

```
Input: Frozen model $f_\theta$, corrupted samples $\{\tilde{x}_i\}_{i=1}^n$, learning rate $\eta$
Output: Optimal transformation parameter $c^*$
1:  Initialize $c \leftarrow 1.0$                          // Identity transformation
2:  $c \leftarrow$ nn.Parameter($c$)                      // Make gradient-trackable
3:  optimizer $\leftarrow$ Adam([$c$], lr=$\eta$)
4:  for $t = 1$ to $T_{\max}$ do
5:      $\hat{x} \leftarrow \text{clip}(c \cdot \tilde{x}, 0, 1)$          // Apply transformation
6:      $p \leftarrow \text{softmax}(f_\theta(\hat{x}))$                // Get predictions
7:      $\mathcal{H}(p) \leftarrow -\sum_{i=1}^{C} p_i \log p_i$     // Compute entropy
8:      $\nabla_c \mathcal{H} \leftarrow$ autograd($\mathcal{H}$, $c$)   // Gradient w.r.t. $c$ only
9:      optimizer.step()                                 // Update $c$
10:     $c \leftarrow \text{clip}(c, 0.1, 10.0)$                // Bound parameter
11:     if $|\nabla_c \mathcal{H}(p)| < \tau$ then
12:         break                                        // Converged
13:     end if
14: end for
15: return $c^* = c$
```

3.1 Problem Formulation

Let $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^C$ be a classifier trained on source distribution $p_s(x, y)$. At test time, inputs arrive from shifted distribution $p_t(x, y)$ where $p_t(x) \neq p_s(x)$ but $p_t(y|x) \approx p_s(y|x)$ representing the covariate shift.

Deployment constraints: We operate under the following restrictions that reflect real-world deployment scenarios where models cannot be modified post-deployment:

- Cannot modify parameters θ
- Cannot access intermediate features or hidden representations
- Cannot exploit architectural knowledge or internal statistics
- Can only compute gradients with respect to input transformations

Traditional test-time adaptation solves:

$$\theta' = \arg\min_\theta \mathcal{L}(f_\theta, \mathcal{D}_t), \tag{1}$$

which requires modifying θ. Under deployment constraints, such modification violates certification.

We reformulate adaptation as input transformation learning:

$$\sigma^* = \arg\min_{\sigma \in \Sigma} \mathbb{E}_{x \sim p_t}[\mathcal{H}(f_\theta(\sigma(x)))] \tag{2}$$

where $\sigma : \mathcal{X} \rightarrow \mathcal{X}$ transforms inputs and $\mathcal{H}$ denotes entropy. This shifts optimization from protected parameter space to accessible input space.

3.2 Input Transformation Architecture

Figure 2 illustrates our adaptation pipeline consisting of three components:

1. Input Adapter $\sigma(\cdot;\psi)$: A parameterized transformation that modifies corrupted inputs. For our proof-of-concept, we use a scalar transformation $\sigma_c(\tilde{x}) = \text{clip}(c \cdot \tilde{x}, 0, 1)$ where $c \in \mathbb{R}_+$ is the learnable parameter. This simple transformation demonstrates the viability of input-space adaptation while leaving room for more expressive transformations in future work.
2. Frozen Model f_θ: The classifier receives adapted inputs and produces predictions. Crucially, θ remains completely frozen throughout adaptation, preserving certifications and deployment constraints.
3. Entropy Objective: The learning signal computed solely from model outputs guides transformation learning without requiring internal access.

3.3 Entropy Minimization in Input Space

Entropy minimization exploits the cluster assumption: decision boundaries lie in low-density regions [6]. All existing formulations minimize entropy in parameter space: $\min_\theta \mathcal{H}(f_\theta(x))$. We propose the dual formulation in input space: $\min_\sigma \mathcal{H}(f_\theta(\sigma(x)))$, where σ transforms inputs while θ remains fixed.

For prediction probabilities $p = \text{softmax}(f_\theta(x)) \in \Delta^{C-1}$, entropy is:

$$\mathcal{H}(p) = -\sum_{i=1}^{C} p_i \log p_i \tag{3}$$

Well-calibrated models exhibit a fundamental property: low entropy on training distribution samples, high entropy on out-of-distribution inputs. This entropy gap provides the learning signal for adaptation.

Given corrupted calibration samples $\{\tilde{x}_i\}_{i=1}^n \sim p_t$, we solve:

$$\min_\sigma \frac{1}{n} \sum_{i=1}^{n} \mathcal{H}(f_\theta(\sigma(\tilde{x}_i))) \tag{4}$$

The transformation σ effectively reverses distribution shift by mapping corrupted inputs toward regions where the model exhibits high confidence, presumably closer to the training distribution. We optimize the transformation using gradient descent on entropy, backpropagating only through the input while keeping model parameters frozen. Parameters are clipped to a valid range, and optimization stops when entropy gradients fall below a fixed threshold Algorithm 1.

4 Experiments

We evaluate ITEM under extreme constraints that reflect real deployment scenarios: severe data scarcity during training, minimal calibration samples at test time, and strictly preserved model parameters.

4.1 Experimental Setup

Datasets and Corruption. We evaluate on MNIST [13] and Fashion-MNIST [30], representing increasing complexity from handwritten digits to fashion items across 10 classes. We apply brightness corruption with factor $\alpha = 0.2$, reducing intensities by 80%: $\tilde{x} = 0.2 \cdot x$. This severe corruption simulates realistic scenarios including degraded sensors, poor lighting conditions, or domain shift between imaging modalities. The corruption magnitude was selected to create substantial performance degradation while maintaining some signal for adaptation.

Model Architecture. We employ SimpleCNNwithBN: two convolutional layers (32/64 filters) with BatchNorm, ReLU activations, 2×2 max pooling, followed by two fully connected layers (128 hidden units) with 0.5 dropout. We specifically choose BatchNorm architectures as they represent the majority of deployed models and enable comparison with normalization-based adaptation methods. The architecture balances expressiveness with the extreme training constraints.

Training Details. Models train on only 10 randomly sampled images per class (100 total) using Adam optimizer with learning rate 0.001, batch size 32, for 50 epochs. This extreme scarcity simulates domains where annotation is expensive (medical imaging), time-consuming (expert labeling), or limited by availability (rare defects). The constraint stress-tests adaptation when models are poorly calibrated due to limited training data.

Adaptation Protocol. Methods receive a calibration set with 1, 100, or 500 samples per class from the corrupted distribution. Calibration samples are selected randomly from the corrupted test set without replacement. ITEM uses learning rate $\eta = 0.05$, maximum 100 iterations with early stopping threshold $\tau = 10^{-6}$, computing gradients through automatic differentiation. We maintain identical protocols across all methods to ensure fair comparison.

Baseline Methods. We compare against four baseline configurations: (1) Original Model shows clean test accuracy before corruption, (2) Corrupted (No Adapt) shows accuracy on corrupted data without any adaptation, (3) Supervised adaptation uses labeled calibration data to directly optimize c via cross-entropy loss, providing an upper bound on unsupervised performance, and (4) Oracle uses perfect knowledge of the corruption factor

Table 3. Performance recovery under varying calibration sample sizes. Original Model shows clean accuracy, Corrupted Model shows performance under brightness corruption ($\alpha = 0.2$). ITEM achieves near-perfect recovery on MNIST while requiring only gradient computation through frozen models.

Method	**MNIST**			**Fashion-MNIST**		
	Calibration Samples per Class			Calibration Samples per Class		
	1	100	500	1	100	500
Original Model		0.832 ± 0.017			0.736 ± 0.017	
Corrupted Model		0.674 ± 0.130			0.344 ± 0.130	
ITEM (Ours)	**0.833 ± 0.016**	**0.833 ± 0.016**	**0.833 ± 0.016**	**0.730 ± 0.017**	**0.727 ± 0.015**	**0.727 ± 0.015**
Supervised	0.832 ± 0.018	0.832 ± 0.018	0.832 ± 0.018	0.718 ± 0.017	0.707 ± 0.021	0.706 ± 0.018
Oracle		0.832 ± 0.017			0.736 ± 0.017	

($c^* = 1/\alpha = 5.0$) without any optimization. Additionally, we evaluate against existing test-time adaptation methods from the literature, with selection rationale and hyperparameter details discussed in Section 4.3.

Evaluation Metrics. We report test accuracy on the full corrupted test set (10,000 images for MNIST/Fashion-MNIST). Results show means and standard deviations over 5 seeds controlling all randomness sources.

Hyperparameter Selection. For ITEM, we use fixed hyperparameters across all experiments without dataset-specific tuning: learning rate 0.05, convergence threshold 10^{-6}, and parameter bounds [0.1, 10.0]. For comparison methods, we perform grid search at 100 calibration samples to identify optimal configurations, then apply these settings consistently across all sample sizes to simulate realistic deployment where validation data may be unavailable.

4.2 Recovery Under Extreme Constraints

We investigate the effectiveness of input-space adaptation in recovering model performance under distribution shift using only forward passes through frozen models. Table 3 reveals critical findings about the effectiveness of input transformation for test-time adaptation.

First, ITEM achieves near-complete recovery on MNIST. From a corrupted baseline of 67.4%, our method recovers to 83.3%, within 0.1% of the original clean performance (83.2%). This recovery using only gradient signals through frozen models demonstrates that entropy minimization in input space can match the effectiveness of parameter-space methods. The result validates our core hypothesis that well-calibrated models provide sufficient feedback through prediction entropy to guide input transformations.

Second, performance saturates immediately with minimal calibration data. Accuracy remains constant at 83.3% across all calibration set sizes from 10 to 5000 samples. This saturation occurs because scalar transformation has only one degree of freedom: once the corruption magnitude is estimated from initial samples, additional data provides no further information.

Unsupervised adaptation closely matches supervised performance. ITEM achieves 83.3% compared to supervised adaptation's 83.2%, a negligible gap suggesting that entropy signals provide nearly as much useful information as true labels for this adaptation task

Fashion-MNIST presents a more challenging scenario with baseline corruption degrading accuracy from 73.6% to 34.4%. ITEM recovers to 73.0%, achieving 99.2% of original performance. The consistent sample efficiency pattern holds, with performance stabilizing at 72.7% across larger calibration sets. The slightly lower recovery compared to MNIST reflects Fashion-MNIST's reliance on fine texture details that are more severely affected by brightness reduction.

4.3 Comparison with Existing Methods

We compare ITEM against established test-time adaptation methods to evaluate its performance under conditions of extreme calibration sample scarcity. We selected representative methods from each TTA category: gradient-based (TENT [28], SAR [22], EATA [21]), statistics-based (AdaBN [15]), pseudo-labeling (SHOT [16], RPL [24]), and architectural-specific (T3A [9], CoTTA [29], MEMO [33]) approaches.

Table 4 demonstrates systematic failure of established test-time adaptation methods under extreme sample constraints, revealing fundamental assumptions that break when calibration data is scarce.

Gradient-based methods require statistical reliability. TENT shows virtually no improvement (67.4% throughout on MNIST) because single-sample batches

Table 4. Comparison with test-time adaptation baselines. ITEM operates without model access yet outperforms most methods requiring gradients, parameters, or architectural knowledge.

Method	MNIST			Fashion-MNIST		
	Calibration Samples per Class			Calibration Samples per Class		
	1	100	500	1	100	500
Original Model		0.832 ± 0.015			0.736 ± 0.015	
Corrupted (No Adapt)		0.674 ± 0.138			0.344 ± 0.138	
ITEM (Ours)	**0.833 ± 0.017**	**0.833 ± 0.017**	**0.833 ± 0.017**	**0.730 ± 0.018**	**0.727 ± 0.015**	**0.727 ± 0.016**
Methods violating our constraints						
TENT	0.674 ± 0.138	0.674 ± 0.138	0.671 ± 0.143	0.344 ± 0.138	0.344 ± 0.138	0.343 ± 0.138
AdaBN	0.832 ± 0.017	0.833 ± 0.017	0.834 ± 0.017	0.737 ± 0.018	0.733 ± 0.017	0.733 ± 0.016
SAR	0.674 ± 0.138	0.670 ± 0.144	0.646 ± 0.183	0.344 ± 0.138	0.343 ± 0.139	0.335 ± 0.143
EATA	0.674 ± 0.138	0.673 ± 0.140	0.667 ± 0.151	0.344 ± 0.138	0.343 ± 0.139	0.340 ± 0.146
CoTTA	0.640 ± 0.182	0.141 ± 0.080	0.101 ± 0.007	0.131 ± 0.067	0.100 ± 0.000	0.100 ± 0.000
SHOT	0.571 ± 0.158	0.684 ± 0.151	0.688 ± 0.145	0.290 ± 0.134	0.368 ± 0.151	0.373 ± 0.145
RPL	0.674 ± 0.138	0.674 ± 0.138	0.674 ± 0.138	0.374 ± 0.136	0.398 ± 0.096	0.414 ± 0.112
T3A	0.566 ± 0.132	0.678 ± 0.083	0.571 ± 0.079	0.299 ± 0.111	0.441 ± 0.080	0.269 ± 0.071
PseudoLabeling	0.802 ± 0.021	0.800 ± 0.018	0.800 ± 0.018	0.493 ± 0.100	0.483 ± 0.085	0.485 ± 0.091
MEMO	0.402 ± 0.117	0.404 ± 0.119	0.403 ± 0.118	0.212 ± 0.100	0.215 ± 0.104	0.213 ± 0.103
Supervised	0.832 ± 0.018	0.832 ± 0.018	0.832 ± 0.018	0.718 ± 0.017	0.707 ± 0.021	0.706 ± 0.018

cannot provide meaningful batch normalization statistics. With batch size 1, the batch mean equals the sample value and variance is undefined, making BatchNorm updates pure noise. SAR and EATA exhibit similar failures, with SAR degrading to 64.6% at 500 samples. These methods implicitly assume batch sizes large enough for stable gradient estimation, requiring O(K log K) samples where K is the number of classes.

AdaBN succeeds through simplicity. By directly replacing stored statistics with target batch statistics (momentum=1.0), AdaBN achieves optimal performance (83.3% on MNIST). This validates that distribution shift primarily manifests in normalization statistics for brightness corruption. However, this approach requires direct access to BatchNorm layers, violating deployment constraints in certified or encrypted systems.

Pseudo-labeling methods suffer from feedback loops. SHOT catastrophically fails (57.1% on MNIST with 1 sample), while basic pseudo-labeling shows surprising resilience (80.2%). The difference stems from SHOT's diversity regularization term, which assumes multiple samples per class. With single samples, the diversity maximization objective dominates, preventing convergence. RPL's conservative confidence threshold (0.9) prevents any updates when initial predictions are uncertain, explaining its baseline-equivalent performance.

Architecture-specific methods collapse. T3A's prototype-based adaptation assumes sufficient samples to estimate class centers, failing with 1 sample per class (56.6%). CoTTA's continual adaptation framework expects temporal correlation between samples, but random sampling violates this assumption, leading to exponential error accumulation (10.1% at 100 samples). MEMO's augmentation strategy cannot overcome the fundamental lack of calibration data, showing modest degradation.

ITEM achieves best performance without model access. Our method matches AdaBN's accuracy while respecting all deployment constraints shown in Table 1. The 50× sample efficiency gap (optimal performance with 10 samples versus others needing 500+) represents a fundamental algorithmic advantage. By operating in the constrained input space rather than high-dimensional parameter space, minimization achieves more efficient adaptation.

The results on Fashion-MNIST amplify these patterns. Methods requiring statistical estimation fail more severely due to the increased complexity of fashion items versus digits. ITEM maintains robust performance (73.0% with 1 sample), while gradient methods show negligible improvement even with 500 samples.

4.4 Analysis of Learned Transformations

We analyze the transformation parameters learned through entropy minimization and compare them to those required for perfect input reconstruction.

Table 5 reveals a systematic pattern: models prefer task-optimal transformations over pixel-perfect reconstruction.

For MNIST with $\alpha = 0.2$ brightness corruption, perfect reconstruction requires $c^* = 5.0$ to exactly invert the corruption. However, entropy minimization consistently learns $c = 3.80$, a 24% undercorrection. This bias increases with corruption severity: at $\alpha = 0.1$, models

Table 5. Learned transformation values across corruption levels. Models consistently prefer task-optimal transformations (undercorrection) over perfect reconstruction. ”1% Range” shows the c-value range achieving 99% of optimal performance.

Dataset	α	Best c	1% Range	Width	Theoretical c^*
MNIST	0.1	7.58±0.32	[3.84, 12.0]	8.16	10.0
	0.2	3.80±0.24	[1.95, 10.0]	8.05	5.0
	0.5	1.45±0.13	[0.81, 4.0]	3.19	2.0
	2.0	0.36±0.04	[0.19, 1.0]	0.81	0.5
Fashion	0.1	10.72±0.28	[8.0, 12.0]	4.0	10.0
	0.2	6.84±0.35	[4.05, 10.0]	5.95	5.0
	0.5	2.74±0.19	[1.62, 4.0]	2.38	2.0
	2.0	0.54±0.03	[0.40, 1.0]	0.60	0.5

learn $c = 7.58$ instead of 10.0 (24% undercorrection), while at $\alpha = 2.0$, they learn $c = 0.36$ instead of 0.5 (28% undercorrection).

Fashion-MNIST shows different behavior, learning slight overcorrections. For $\alpha = 0.2$, the optimal $c = 6.84$ exceeds the theoretical 5.0 by 37%. This dataset-specific difference reflects Fashion-MNIST's reliance on texture information, where slight contrast enhancement aids classification even at the cost of pixel fidelity.

The wide performance plateaus shown in the ”1% Range” column indicate robust adaptation. For MNIST with $\alpha = 0.2$, any value in [1.95, 10.0] achieves 99% of optimal accuracy. This 8-point range suggests entropy landscapes have broad minima where exact transformation values matter less than being in the correct region.

These findings reveal that models optimize for decision boundary compatibility rather than input reconstruction. Perfect restoration might push samples into regions where the model's confidence calibration breaks down, while task-optimal transformations maintain inputs within learned feature distributions. This principle, where adaptation prioritizes functional performance over reconstruction fidelity, represents a key insight for designing future test-time adaptation methods.

5 Conclusion

We present ITEM, the first preparation-agnostic sample adaptation method for test-time adaptation. By reformulating adaptation from parameter space to input space, ITEM enables performance recovery for arbitrary pretrained models while respecting deployment constraints that prohibit model modification.

Our key contributions challenge fundamental assumptions in test-time adaptation. First, we demonstrated that effective adaptation does not require parameter updates, architectural knowledge, or training preparation. Transforming inputs to minimize prediction entropy suffices for significant performance recovery. Second, we revealed extreme sample efficiency emerges naturally from input-space optimization, with single calibration samples providing sufficient signal when transformation complexity is appropriately constrained. Third, we discovered that task-optimal transformations systematically differ from reconstruction-optimal ones, indicating models prioritize maintaining inputs within learned decision boundaries over pixel-perfect restoration.

Broader implications. ITEM's success suggests the test-time adaptation community should reconsider the necessity of model modification. While parameter updates offer flexibility, our results demonstrate that careful input transformation can achieve comparable adaptation with far stricter constraints. The extreme sample efficiency, achieving with 10 samples what gradient methods require 500+ for, questions whether existing approaches over-engineer solutions to fundamentally simple problems. The systematic failure of established methods under sample scarcity reveals how architectural dependencies create unnecessary brittleness during deployment.

Limitations and Future Work. Our evaluation focuses on uniform brightness corruption across two benchmark datasets. While this controlled setting enables careful analysis, broader evaluation is needed. Testing across diverse corruption types (blur, noise, weather), additional datasets (CIFAR, ImageNet), and varied architectures (ResNets, Vision Transformers) would strengthen generalizability claims. The scalar transformation, while effective for global corruptions, represents the simplest instantiation of input-space adaptation. Extending to learnable spatial transformations under the constraints highlighted in Tab. 1 presents an important challenge. The framework naturally extends to richer parameterizations: neural network transformations could address spatially-varying corruptions while maintaining gradient-only operation.

Sample selection strategies offer another research direction. While random selection suffices, entropy-based selection could improve efficiency. Initial experiments suggest boundary samples with highest initial entropy provide marginally better adaptation signals. Combined with few-shot learning techniques, this approach could push sample requirements even lower.

ITEM opens a new research direction at the intersection of test-time adaptation and deployment-aware machine learning. By demonstrating that preparation-agnostic sample adaptation is not only possible but effective, we hope to inspire methods that bridge the gap between academic innovation and real-world constraints. As models increasingly become immutable services rather than modifiable artifacts, adaptation techniques must evolve accordingly. Our work shows that respecting deployment constraints need not compromise adaptation effectiveness.

Code: `https://github.com/LeTeutz/tta-item`

References

[1] Amazon Web Services. Amazon Rekognition Developer Guide. `https://docs.aws.amazon.com/rekognition/`, 2023. Accessed: June 2025.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010.

[3] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017.

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[5] Google Cloud. Google Cloud Vision API. `https://cloud.google.com/vision`, 2023. Accessed: June 2025.

[6] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, 2004.

[7] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13 (1):723–773, 2012.

[8] Hao Guan and Mingxia Liu. Domain adaptation for medical image analysis: A survey. *IEEE Transactions on Biomedical Engineering*, 69(3):1173–1185, 2022.

[9] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, pages 2427–2440, 2021.

[10] Youngeun Kim, Junho Yim, Juseung Yun, and Junmo Kim. Domain adaptation without source data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (11):7641–7653, 2022.

[11] Sandeep Kumar, Rajesh Singh, Wei Zhang, and Nisha Patel. Object detection in adverse weather for autonomous driving through data merging and YOLOv8. *Sensors*, 23 (20):8471, 2023. DOI: 10.3390/s23208471.

[12] Jogendra Nath Kundu, Akshay Kulkarni, Amit Singh, Varun Jampani, and R Venkatesh Babu. Generalize then adapt: Source-free domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7046–7056, 2021.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, page 896, 2013.

[15] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

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

[17] Jian Liang, Dapeng Hu, Yunbo Wang, Ran He, and Jiashi Feng. Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8602–8617, 2021.

[18] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2024.

[19] Microsoft Azure. Azure Computer Vision Documentation. `https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/`, 2023. Accessed: June 2025.

[20] M Jehanzeb Mirza, Pol Jané Soneira, Wei Lin, Mateusz Kozinski, Horst Possegger, Hilde Kuehne, and Horst Bischof. Norm: Test-time adaptation for out-of-distribution generalization via normalized prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1290–1299, 2023.

[21] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, pages 16888–16905. PMLR, 2022.

[22] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

[23] Junhyeong Park, Yeongmin Jin, Jeongyeol Jang, Sangho Bae, and Seungjun Kim. Test-time style shifting: Handling arbitrary styles in domain generalization. In *International Conference on Machine Learning*, pages 27133–27154. PMLR, 2023.

[24] Evgenia Rusak, Steffen Schneider, George Pachitariu, Luisa Eck, Peter V Gehler, Matthias Bethge, and Wieland Brendel. If your data distribution shifts, use self-learning. *Transactions on Machine Learning Research*, 2022.

[25] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *Advances in Neural Information Processing Systems*, pages 24340–24356, 2021.

[26] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5):985–1005, 2007.

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

[28] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[29] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

[30] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[31] Zehao Xiao and Cees G. M. Snoek. Beyond model adaptation at test time: A survey, 2024.

[32] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *Advances in Neural Information Processing Systems*, 34: 29393–29405, 2021.

[33] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, pages 38629–38642, 2022.

3 Background

This chapter establishes the theoretical foundations necessary for understanding test-time adaptation. We begin with deep learning architectures that power modern computer vision systems, examine how batch normalization creates both opportunities and vulnerabilities for adaptation, analyze distribution shift phenomena that degrade model performance, and explore entropy-based methods for uncertainty-aware adaptation. These concepts provide the foundation for understanding how models can adapt to new data distributions during deployment without access to their internal parameters.

3.1. Deep Neural Networks

3.1.1. Fundamentals of Neural Networks

Neural networks transform input data through sequences of parameterized operations, learning complex mappings from data rather than explicit programming. At its core, each layer computes:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ represents the input, $\mathbf{W} \in \mathbb{R}^{m \times n}$ the weight matrix, $\mathbf{b} \in \mathbb{R}^m$ the bias vector, and f a non-linear activation function. The non-linearity proves essential, since without it, any deep network collapses to a single linear transformation, severely limiting representational capacity. An example of such a neural network can be seen in Figure 3.1.

The Universal Approximation Theorem [5–7] provides theoretical justification for neural networks' effectiveness. This fundamental result proves that networks with sufficient capacity can approximate any continuous function on compact sets to arbitrary precision. The theorem guarantees existence of such approximations but does not specify the required architecture size or guarantee that gradient-based optimization will find these approximations. This gap between theoretical possibility and practical optimization remains a central challenge in deep learning.

3.1.2. Convolution Operation

The convolution operation forms the computational foundation of convolutional neural networks (CNNs). Unlike matrix multiplication in fully connected layers, convolution exploits the spatial structure of images through local connectivity and parameter sharing. Figure 3.2 illustrates how a kernel slides across an input to produce feature maps.

Mathematically, for 2D input I and kernel K:

$$(I * K)(i,j) = \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} I(i+m, j+n) \cdot K(m,n) \tag{3.2}$$

This operation exhibits three key properties. Local connectivity ensures each output depends only on a spatially localized input region, reducing parameters from $O(n^4)$ to $O(k^2n^2)$ for $n \times n$ images and $k \times k$ kernels. Parameter sharing applies the same kernel across all spatial locations, enforcing translation

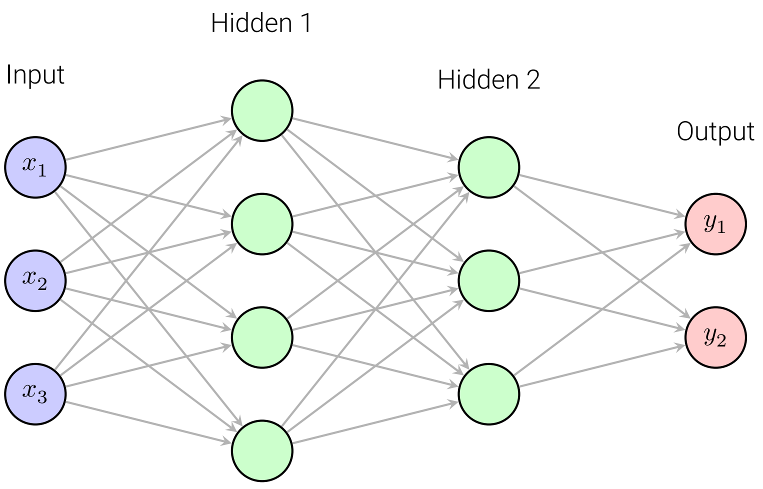


Figure 3.1: Feedforward neural network architecture. Information flows from input through hidden layers to output, with full connectivity between adjacent layers. Each neuron computes a weighted sum followed by non-linear activation.

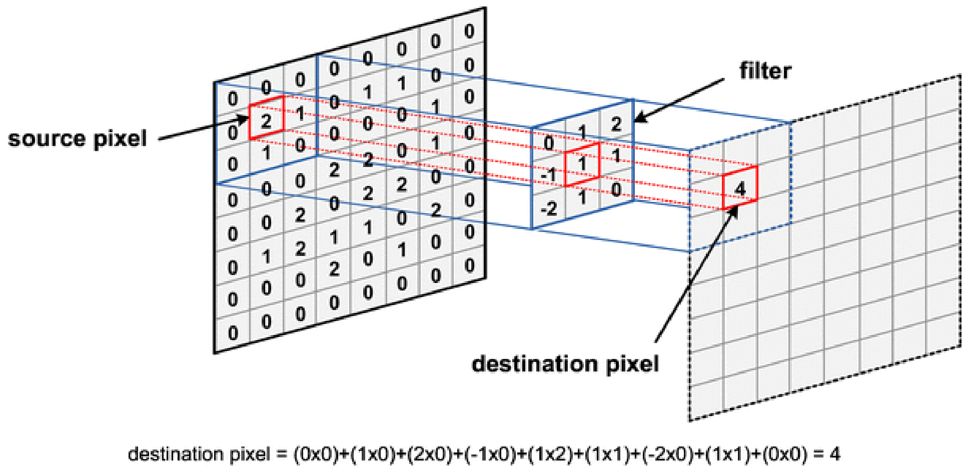

source pixel
filter
destination pixel
destination pixel = (0x0)+(1x0)+(2x0)+(-1x0)+(1x2)+(1x1)+(-2x0)+(1x1)+(0x0) = 4

Figure 3.2: Convolution operation. A 3×3 filter slides across the input, computing element-wise products at each position. The red box shows the current receptive field being processed. The output value represents the sum of element-wise products between the kernel and the highlighted input region. Figure taken from [8].

equivariance: a shifted input produces a correspondingly shifted output. These properties enable CNNs to learn visual features while maintaining computational efficiency.

3.1.3. Convolutional Neural Networks

Convolutional Neural Networks [9, 10] leverage the convolution operation within a hierarchical architecture designed for visual recognition. The key insight lies in combining local feature extraction with spatial pooling to build increasingly abstract representations. Figure 3.3 shows a typical CNN architecture.

A typical CNN architecture consists of several components working together. Convolutional layers apply banks of learnable filters to extract features at multiple scales and orientations. Each filter learns to detect specific patterns: edges, textures, or more complex structures in deeper layers. Pooling layers downsample spatial dimensions through max or average operations, providing translation invariance and computational efficiency. Activation functions, typically ReLU [11], introduce non-linearity: ReLU$(x) = \max(0, x)$. Finally, fully connected layers aggregate spatial features for classification.

3.1.4. Image Classification as a Computer Vision Task

Image classification represents a fundamental challenge in computer vision: assigning semantic labels to images based on their visual content. This task requires models to extract meaningful features from raw pixel intensities and map these representations to discrete categories. The complexity arises from

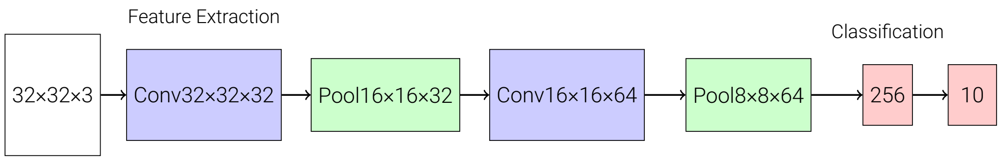


Figure 3.3: CNN architecture showing the progression from spatial feature maps through pooling to final classification. Spatial dimensions decrease while channel depth increases. The architecture extracts hierarchical features through alternating convolution and pooling operations before final classification.

intra-class variation such as dogs of different breeds, viewpoint changes, illumination differences, occlusions, and background clutter.

Classical approaches relied on hand-crafted features like SIFT [12], HOG [13], and bag-of-visual-words [14]. These methods required domain expertise and often failed to generalize across diverse datasets. The introduction of convolutional neural networks revolutionized this field by learning hierarchical feature representations directly from data.

For evaluation, classification accuracy measures the fraction of correct predictions, while top-5 error considers a prediction correct if the true label appears among the five highest confidence predictions. The MNIST dataset [15] of handwritten digits containing 28×28 grayscale images across 10 classes serves as a canonical benchmark for method development, though its simplicity limits its utility for evaluating modern architectures. Figure 3.4 shows sample digits from this dataset.

Figure 3.4: Example digits from the MNIST dataset showing handwritten numerals 0-9. Each image contains 784 pixels (28×28) representing grayscale intensity values. The dataset contains 60,000 training and 10,000 test images.

3.1.5. Training Deep Networks

Network training optimizes millions of parameters through gradient-based methods. Given loss function L, parameters θ update according to:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L \tag{3.3}$$

where α denotes the learning rate and $\nabla_\theta L$ represents the gradient of the loss with respect to parameters. Backpropagation [16] efficiently computes gradients through recursive application of the chain rule. Modern optimizers like SGD with momentum [17], Adam [18], and AdamW [19] incorporate adaptive learning rates and momentum terms for improved convergence.

The choice of learning rate α critically affects training dynamics. Too large values cause divergence; too small values slow convergence. Learning rate schedules, including step decay, cosine annealing [20], and warm restarts, help navigate this trade-off.

3.2. Batch Normalization

3.2.1. Motivation and Mechanism

Batch Normalization (BN) [21] addresses internal covariate shift: the changing distribution of layer inputs during training. As parameters in early layers update, subsequent layers must continuously adapt to shifting input distributions, necessitating lower learning rates and careful initialization.

For mini-batch $\mathcal{B} = \{x_1, ..., x_m\}$, BN normalizes activations through:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.4}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B})^2} \tag{3.5}$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}+\epsilon}}} \tag{3.6}$$

$$y_i = \gamma\hat{x}_i + \beta \tag{3.7}$$

where ϵ ensures numerical stability, and learnable parameters γ, β allow the network to recover the original distribution if beneficial. Figure 3.5 illustrates the batch normalization pipeline.

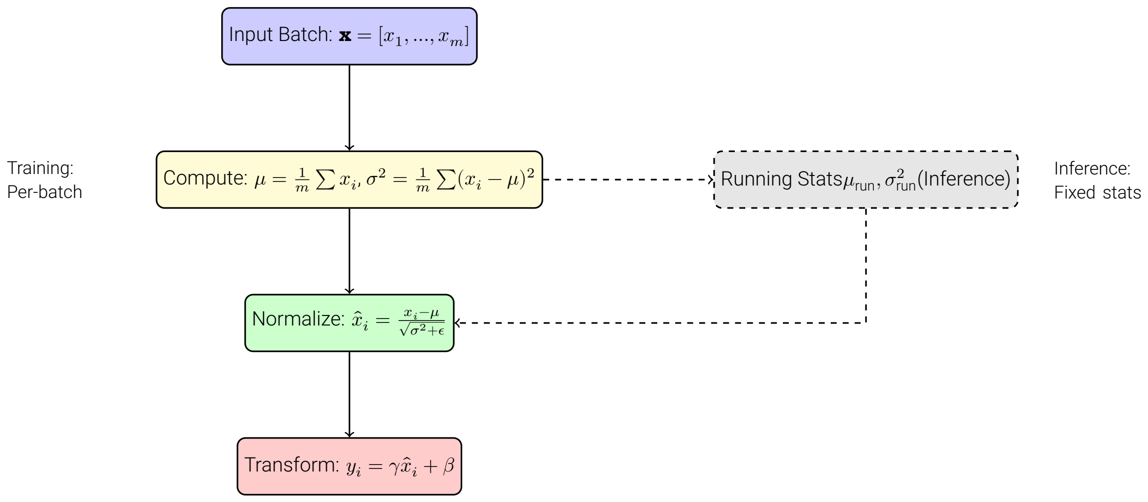

Input Batch: $\mathbf{x} = [x_1, ..., x_m]$
Training: Per-batch
Compute: $\mu = \frac{1}{m}\sum x_i, \sigma^2 = \frac{1}{m}\sum(x_i - \mu)^2$
Running Stats$\mu_{\text{run}}, \sigma^2_{\text{run}}$(Inference)
Inference: Fixed stats
Normalize: $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$
Transform: $y_i = \gamma\hat{x}_i + \beta$

Figure 3.5: Batch Normalization pipeline. During training, statistics are computed per batch. During inference, fixed running statistics ensure deterministic predictions. The dashed path indicates the use of accumulated statistics during inference mode.

3.2.2. Benefits and Vulnerabilities

BN accelerates training by enabling higher learning rates and reducing sensitivity to initialization [22, 23]. The normalization effect acts as implicit regularization, often eliminating the need for dropout [24].

However, BN introduces vulnerabilities during deployment. The fixed statistics assumption breaks under distribution shift: when test data differs from training data, pre-computed statistics become inappropriate. This mismatch particularly affects applications with varying deployment conditions, motivating test-time adaptation methods that adjust normalization statistics online.

3.3. Distribution Shift and Domain Adaptation

3.3.1. Types of Distribution Shift

Distribution shift represents a fundamental challenge in deploying machine learning systems. We categorize shifts based on which aspects of the joint distribution $P(X, Y)$ change between training and test time. Understanding these categories helps identify appropriate adaptation strategies. Thus, the shifts are categorized in the following way:

1. Covariate Shift occurs when $P_{\text{train}}(X) \neq P_{\text{test}}(X)$ while $P_{\text{train}}(Y|X) = P_{\text{test}}(Y|X)$. The marginal input distribution changes but the conditional label distribution remains stable. This shift commonly

appears in computer vision when lighting conditions, camera sensors, or image quality differ between training and deployment environments;

2. Concept Drift manifests when $P_{\text{train}}(Y|X) \neq P_{\text{test}}(Y|X)$. The fundamental relationship between inputs and outputs evolves over time. Fashion classification systems face this challenge as styles change seasonally, and malware detection systems encounter it as attack patterns evolve;
3. Prior Shift occurs when $P_{\text{train}}(Y) \neq P_{\text{test}}(Y)$ while $P_{\text{train}}(X|Y) = P_{\text{test}}(X|Y)$. Class frequencies change between training and deployment, which frequently happens in medical diagnosis where disease prevalence varies across populations and geographic regions.

3.3.2. Impact on Neural Networks

Distribution shift degrades model accuracy through cascading effects. Early layers encounter features outside their trained activation ranges, causing representation collapse. Batch normalization statistics calibrated on training data produce incorrect normalizations. Decision boundaries optimized for training distributions no longer separate classes effectively. Uncertainty estimates become miscalibrated, producing overconfident errors on out-of-distribution samples [25, 26].

Consider medical imaging systems trained on high-quality research data but deployed with commodity hardware. Lower resolution, different acquisition protocols, and patient population shifts combine to create substantial accuracy degradation. Without adaptation mechanisms, such systems fail to maintain clinical utility.

3.3.3. Domain Adaptation Approaches

The machine learning community has developed various strategies to address distribution shift, each with distinct assumptions and requirements. These approaches differ primarily in their access to source data and timing of adaptation. We highlight three such strategies:

1. Source-Dependent Methods assume continued access to source data during adaptation. Domain Adaptation (DA) [27, 28] learns invariant representations by jointly optimizing on source and target data. Transfer Learning [29] follows a two-stage approach: pre-training on large source datasets followed by fine-tuning on limited target data. These methods achieve strong results but require storing and accessing potentially sensitive training data;
2. Source-Free Methods operate without source data access, addressing privacy and storage constraints. Source-Free Domain Adaptation (SFDA) [30, 31] adapts pre-trained models using unlabeled target data collected before deployment. Test-Time Adaptation (TTA) pushes this constraint further, performing adaptation online during inference without any data collection phase;
3. Temporal Adaptation Strategies differ in when adaptation occurs relative to deployment. Offline methods collect target data for batch adaptation before deployment, allowing multiple optimization passes. Online methods adapt during deployment as samples arrive, requiring single-pass algorithms. Continual methods [32] handle non-stationary distributions over extended periods, incorporating forgetting prevention mechanisms.

3.4. Test-Time Adaptation

Test-time adaptation represents the most challenging setting: adapting models during deployment using only unlabeled test data. Given model f_θ trained on source distribution $\mathcal{D}_s$, TTA methods must improve accuracy on shifted distribution $\mathcal{D}_t$ without labels or source data access.

3.4.1. Categories of TTA Methods

TTA approaches fundamentally differ in which model component they modify during adaptation. Each category offers distinct trade-offs between adaptation effectiveness, computational requirements, and deployment constraints. This taxonomy, introduced by Liang et al. [33], provides a systematic framework for understanding the landscape of test-time adaptation methods. Figure 3.6 illustrates all five adaptation paradigms:

1. Model Adaptation directly updates neural network parameters during test time. These methods optimize unsupervised objectives like entropy minimization to align the model with test data characteristics. TENT [2] updates batch normalization affine parameters, while MEMO [34] performs full model adaptation using augmentation-based consistency. While effective, parameter updates require gradient computation and careful hyperparameter tuning to prevent catastrophic adaptation on outliers;

2. Sample Adaptation transforms test inputs to better match the training distribution rather than modifying the model. This approach learns input-space transformations that reduce distribution mismatch. DUA [35] applies learnable augmentations, while TTT [36] uses self-supervised rotation prediction to adapt representations. The model remains frozen, eliminating risks of parameter corruption while maintaining computational efficiency;

3. Normalization Adaptation updates only batch normalization statistics to match test data. This lightweight approach modifies normalization parameters while keeping all other model weights frozen. AdaBN [3] replaces source statistics with target statistics, while OTTA [37] uses optimal transport for statistics alignment. The method exploits BN's sensitivity to distribution shift, achieving significant improvements with minimal computational overhead;

4. Inference Adaptation adjusts predictions without modifying model parameters or inputs. These methods use techniques like self-ensembling, test-time augmentation, or prediction refinement. BACS [38] performs Bayesian pseudo-label correction, while T3A [39] uses a lightweight classifier adjustment module. The approach maintains model integrity while improving predictions through post-processing strategies;

5. Prompt Adaptation learns input-space prompts that guide pre-trained models toward better test-time predictions. This paradigm gained prominence with vision transformers and foundation models. TPT [4] optimizes visual prompts using confidence selection, while VPT [40] learns task-specific prompt tokens. Prompts act as task-specific conditioning signals that adapt model behavior without parameter updates.

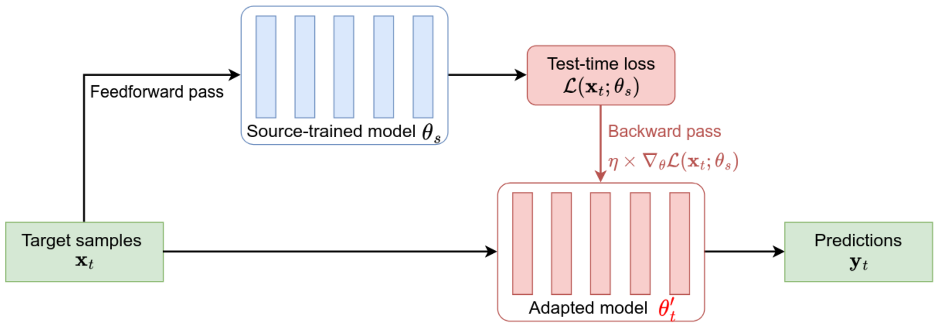
3.4.2. Preparation Requirements

TTA methods impose different requirements on model training and data preparation. These constraints determine which existing models can benefit from each adaptation approach. The following categorization, systematically introduced by Liang et al. [33], captures the spectrum of preparation complexity across TTA methods:

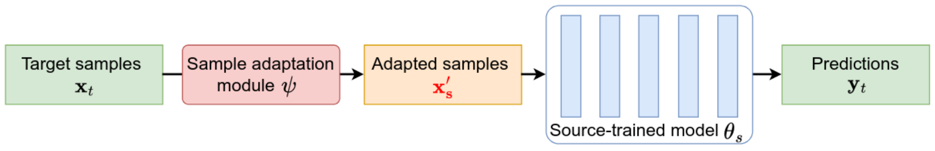
1. Preparation-Agnostic TTA works with any pre-trained model without special training procedures. These methods adapt standard models at test time using only the test data. TENT [2] exemplifies this approach by updating only batch normalization parameters, while BN adaptation methods like AdaBN [3] simply replace normalization statistics. This flexibility enables retrofitting existing deployed models with adaptation capabilities;

2. Training Preparation TTA requires specific training procedures but uses standard training data. TTT [36] trains with auxiliary rotation prediction tasks, while SAR [42] incorporates sharpness-aware regularization during training. These methods embed adaptation-friendly properties into models during training, limiting applicability to new training pipelines but achieving superior adaptation without data augmentation;

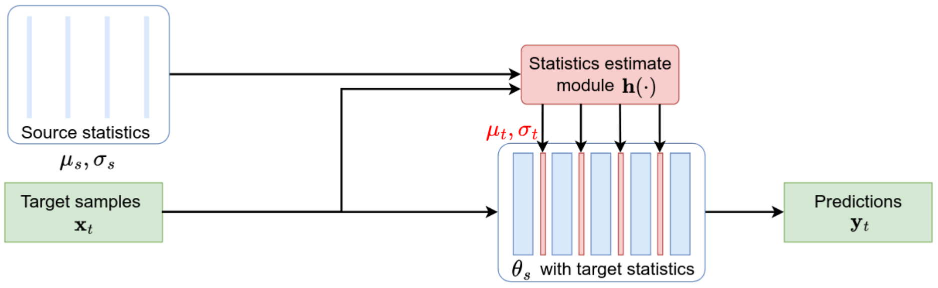
3. Training and Data Preparation TTA requires both specialized training procedures and augmented training data. MEMO [34] trains with consistency regularization across augmentations, while MetaTTA [43] employs meta-learning with simulated distribution shifts. These methods prepare models explicitly for test-time adaptation scenarios through adversarial training or meta-learning approaches. While achieving superior adaptation, the extensive preparation requirements limit practical deployment.


Feedforward pass
Source-trained model θ_s
Test-time loss $\mathcal{L}(\mathbf{x}_t;\theta_s)$
Backward pass $\eta \times \nabla_\theta \mathcal{L}(\mathbf{x}_t;\theta_s)$
Target samples $\mathbf{x}_t$
Adapted model θ'_t
Predictions $\mathbf{y}_t$

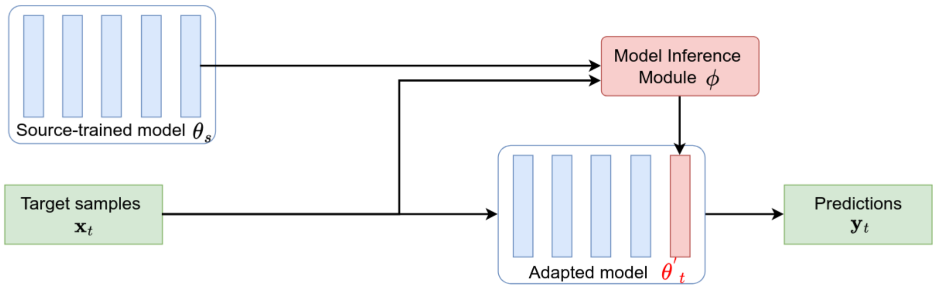
Model adaptation [41].


Target samples $\mathbf{x}_t$
Sample adaptation module ψ
Adapted samples $\mathbf{x}'_s$
Source-trained model θ_s
Predictions $\mathbf{y}_t$

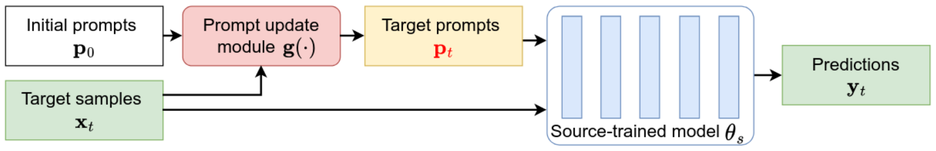
Sample adaptation [41].


Source statistics
μ_s, σ_s
Statistics estimate module $\mathbf{h}(\cdot)$
μ_t, σ_t
Target samples $\mathbf{x}_t$
θ_s with target statistics
Predictions $\mathbf{y}_t$

Normalization adaptation [41].


Source-trained model θ_s
Model Inference Module ϕ
Target samples $\mathbf{x}_t$
Adapted model θ'_t
Predictions $\mathbf{y}_t$

Inference adaptation [41].


Initial prompts $\mathbf{p}_0$
Prompt update module $\mathbf{g}(\cdot)$
Target prompts $\mathbf{p}_t$
Target samples $\mathbf{x}_t$
Source-trained model θ_s
Predictions $\mathbf{y}_t$

Prompt adaptation [41].

Figure 3.6: Five categories of test-time adaptation methods [33]. Each approach modifies different components: (a) updates model parameters, (b) transforms inputs, (c) adjusts normalization statistics, (d) refines predictions, and (e) learns adaptive prompts. Figures taken from [41].

3.4.3. Online versus Offline Adaptation

The temporal characteristics of test data arrival fundamentally shape adaptation strategies. Each setting presents unique challenges for maintaining model stability while achieving effective adaptation. The two situations mentioned above are:

1. Offline TTA assumes access to the entire test set before making predictions. This setting allows multiple passes through the data and computation of global statistics. Batch normalization adaptation methods often operate in this setting, computing stable statistics from the complete test distribution. The approach suits scenarios like adapting to a new deployment environment with representative data available;

2. Online TTA adapts to each sample or mini-batch as it arrives, making predictions immediately. This setting reflects real-world deployment scenarios where data arrives sequentially. Online methods must balance adaptation speed with stability, often using momentum-based updates or sample selection strategies. The challenge lies in preventing catastrophic updates from outliers while maintaining responsiveness to distribution changes.

3.4.4. Entropy Minimization for Test-Time Adaptation

TENT [2] exemplifies modern TTA approaches by minimizing prediction entropy through selective parameter updates:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_t} H(f_\theta(\mathbf{x})) \tag{3.8}$$

where $H(\cdot)$ denotes entropy. For details on entropy, see Section 3.5. This optimization encourages confident predictions, leveraging the cluster assumption that decision boundaries should lie in low-density regions. The method updates only the affine parameters of batch normalization layers, maintaining computational efficiency while achieving significant adaptation benefits.

3.4.5. Common Requirements and Limitations

TTA methods face several operational constraints that limit their deployment scenarios. Understanding these limitations helps select appropriate methods for specific applications. Such operational constraints are:

1. Gradient Access Requirements: Most TTA methods require computing gradients for optimization, limiting deployment to white-box settings. Black-box scenarios where only predictions are available require alternative approaches like input transformation or output calibration;

2. Batch Size Dependencies: Batch statistics estimation needs sufficient samples for reliability. Small batch sizes lead to noisy estimates, while large batches may contain multiple distribution modes. Methods must balance statistical stability with computational constraints;

3. Architectural Assumptions: Some approaches require specific architectural components like batch normalization layers. Models without these components cannot benefit from normalization-based adaptation. Architecture-agnostic methods trade effectiveness for broader applicability;

4. Distribution Shift Assumptions: Accuracy guarantees typically assume specific shift types like covariate shift. Methods optimized for one shift type may fail catastrophically under different shifts. Practical deployments must consider the expected shift characteristics [44].

3.5. Entropy and Uncertainty

3.5.1. Information-Theoretic Foundations

Shannon entropy [45] quantifies uncertainty in probability distributions. For discrete distribution p over outcomes $\mathcal{X}$:

$$H(p) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{3.9}$$

Entropy achieves its maximum $\log|\mathcal{X}|$ for uniform distributions representing maximum uncertainty and minimum 0 for deterministic outcomes. This measure provides a principled approach to quantifying prediction confidence.

Consider a 3-class classification problem with predictions $p = [0.8, 0.15, 0.05]$. The entropy equals:

$$H(p) = -(0.8 \log 0.8 + 0.15 \log 0.15 + 0.05 \log 0.05) \approx 0.64 \tag{3.10}$$

Compare this to a uniform distribution $p = [0.33, 0.33, 0.34]$ with entropy:

$$H(p) \approx 1.10 \approx \log 3 \tag{3.11}$$

The confident prediction yields lower entropy than the uncertain uniform distribution.

3.5.2. Entropy in Neural Network Predictions

Classification networks output probability distributions over classes through the softmax function:

$$p_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{3.12}$$

where z_i represents the logit for class i. The entropy of this predictive distribution:

$$H(p) = -\sum_{i=1}^{C} p_i \log p_i \tag{3.13}$$

serves as an uncertainty measure. Low entropy indicates confident predictions with probability mass concentrated on few classes. High entropy suggests uncertainty with probability spread across multiple classes.

3.5.3. The Cluster Assumption and Entropy Minimization

The cluster assumption [46] posits that decision boundaries should traverse low-density regions of the data distribution. This principle underlies semi-supervised learning and manifests in test-time adaptation through entropy minimization. Figure 3.7 visualizes this concept.

For well-separated clusters, samples near cluster centers should yield confident predictions with low entropy, while samples near boundaries produce uncertain predictions with high entropy. Entropy minimization during test time implicitly pulls samples toward nearby clusters, adapting the decision boundary to the test distribution.

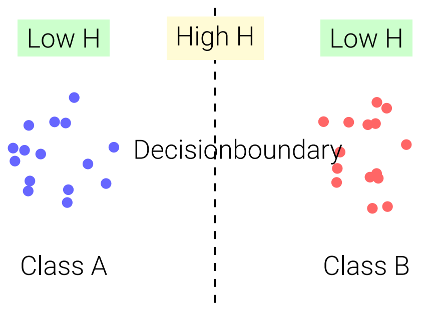


Figure 3.7: The cluster assumption illustrated. Decision boundaries lie in low-density regions between clusters. Samples near cluster centers have low entropy (confident predictions), while samples near boundaries have high entropy (uncertain predictions).

This approach assumes test samples lie near meaningful clusters in feature space, an assumption that may fail for severe distribution shifts or truly out-of-distribution samples. Recent work [42] explores entropy minimization techniques that identify and handle unreliable samples through sample selection and weighted updates.

Bibliography

[1] Shai Ben-David et al. “A theory of learning from different domains”. In: *Machine learning* 79.1 (2010), pp. 151–175.

[2] Dequan Wang et al. “Tent: Fully test-time adaptation by entropy minimization”. In: *International Conference on Learning Representations*. 2021.

[3] Yanghao Li et al. “Revisiting batch normalization for practical domain adaptation”. In: *arXiv preprint arXiv:1603.04779* (2016).

[4] Manli Shu et al. “Test-time prompt tuning for zero-shot generalization in vision-language models”. In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 14274–14289.

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[6] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[7] Ken-Ichi Funahashi. “On the approximate realization of continuous mappings by neural networks”. In: *Neural networks* 2.3 (1989), pp. 183–192.

[8] Anan Banharnsakun. “Towards improving the convolutional neural networks for deep learning using the distributed artificial bee colony method”. In: *International Journal of Machine Learning and Cybernetics* 10 (June 2019). DOI: `10.1007/s13042-018-0811-z`.

[9] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[11] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[12] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[13] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[14] Gabriella Csurka et al. “Visual categorization with bags of keypoints”. In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. 2004, pp. 1–2.

[15] Yann LeCun, Corinna Cortes, and Christopher JC Burges. “The MNIST database of handwritten digits”. In: *http://yann.lecun.com/exdb/mnist/* (1998).

[16] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[17] Boris T Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[18] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[19] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *International Conference on Learning Representations*. 2018.

[20] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic gradient descent with warm restarts”. In: *International Conference on Learning Representations*. 2017.

[21] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[22] Shibani Santurkar et al. “How does batch normalization help optimization?” In: *Advances in neural information processing systems* 31 (2018).

[23] Nils Björck et al. “Understanding batch normalization”. In: *Advances in neural information processing systems*. 2018, pp. 7694–7705.

[24] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[25] Chuan Guo et al. “On calibration of modern neural networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1321–1330.

[26] Yaniv Ovadia et al. “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift”. In: *Advances in neural information processing systems* 32 (2019).

[27] Yaroslav Ganin et al. “Domain-adversarial training of neural networks”. In: vol. 17. 1. JMLR. org, 2016, pp. 2096–2030.

[28] Eric Tzeng et al. “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[29] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems* 27 (2014).

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

[31] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. “Universal source-free domain adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4544–4553.

[32] Qin Wang et al. “Continual test-time domain adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7201–7211.

[33] Jian Liang, Ran He, and Tieniu Tan. “A Comprehensive Survey on Test-Time Adaptation Under Distribution Shifts”. In: *International Journal of Computer Vision* 133.1 (2025), pp. 31–64.

[34] Marvin Zhang, Sergey Levine, and Chelsea Finn. “Memo: Test time robustness via adaptation and augmentation”. In: *Advances in neural information processing systems* 35 (2022), pp. 38629–38642.

[35] M Jehanzeb Mirza et al. “Norm: Test-time adaptation for out-of-distribution generalization via normalized prediction”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 1290–1299.

[36] Yu Sun et al. “Test-time training with self-supervision for generalization under distribution shifts”. In: *International conference on machine learning*. PMLR. 2020, pp. 9229–9248.

[37] Jungsoo Jang et al. “Test-time adaptation via self-training with nearest neighbor information”. In: *International Conference on Learning Representations*. 2023.

[38] Jonghyun Park et al. “Toward better test-time adaptation via test-time transformation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 11909–11918.

[39] Yusuke Iwasawa and Yutaka Matsuo. “Test-time classifier adjustment module for model-agnostic domain generalization”. In: vol. 34. 2021, pp. 2427–2440.

[40] Menglin Jia et al. “Visual prompt tuning”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 709–727.

[41] Zehao Xiao and Cees GM Snoek. “Beyond model adaptation at test time: A survey”. In: *arXiv preprint arXiv:2411.03687* (2024).

[42] Shuaicheng Niu et al. “Towards stable test-time adaptation in dynamic wild world”. In: *International Conference on Learning Representations*. 2023.

[43] Alexander Bartler et al. “Mt3: Meta test-time training for self-supervised test-time adaption”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 3080–3090.

[44] Shuaicheng Niu et al. “Efficient test-time model adaptation without forgetting”. In: *International conference on machine learning*. PMLR. 2022, pp. 16888–16905.

[45] Claude E Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[46] Olivier Chapelle and Alexander Zien. “Semi-supervised classification by low density separation”. In: *International workshop on artificial intelligence and statistics*. PMLR. 2005, pp. 57–64.