Meta-Learning Enhanced Deep Neural Networks for Cross-Domain Generalization in Dynamic and Non-Stationary Environments

Dr. Jagadeesh Krishna[1*], Praveen .B.S.[2], Thatchayeni .E, , Mr. Kirit Rasiklal Rathod[4], Dr.G.Harish Kumar[5], Ranjan Banerjee[6], Nudrat Sufiyan[7]

[1*]University of Mysore, jagadeeshkrishna.pgc@mahajana.edu.in
[2]University of Mysore, praveens.pgc@mahajana.edu.in
[3]University of Mysore, mahajanabcaaimaths@gmail.com
[4]Assistant Professor, Computer Engineering Department, Government Engineering College, Bhavnagar, Gujarat Technological University, rathod.kirit@gmail.com
[5]Assistant Professor, Data Science Department Malla Reddy Engineering College for Women Hyderabad, harish.gopadi@gmail.com
[6]Assistant Professor, Computer Science & Engineering, Brainware University, Barasat, Kolkata – 125, West Bengal, India, Orcid Id : 0009-0003-1950-7530, rbkpccst@gmail.com
[7]Department of ECE School of Engineering and Technology, CGC University, Mohali, Punjab, India nudrat.j4843@cgcuniversity.in

Abstract

Deep neural networks deployed in intelligent systems commonly encounter domain shifts that are neither independent nor stationary: sensor characteristics change, class priors evolve, decision boundaries drift, and previously observed regimes recur. While conventional domain generalization optimizes a fixed predictor prior to deployment, online adaptation typically does not have an initialization or control policy for cross-domain transfer. In this article, we present a novel adaptive neural architecture with delayed supervision, called MetaDyG-Net, which is enhanced by meta-learning. The method learns (i) an encoder initialization that can be transferred across tasks and (ii) a small drift-aware controller which maps the discrepancy between the features moments, the predictive entropy, and the uncertainty to the layer-wise adaptation rates and the replay strength. Deployment is the fusion of a reservoir memory, Fisher-weighted elastic anchoring, class-prototype fusion and uncertainty-adaptive probability tempering. A first-order episodic objective is used to let the model see pseudo-unseen source domains while explicitly learning the stability-plasticity trade-off. Evaluation is performed on five independent seeds on two controlled cross-domain streams: DS-Digits, which is created from real handwritten-digit images with abrupt, gradual, recurrent and compound visual shifts, and DMS-24, a nonlinear multi-sensor stream with covariate, prior and concept drift. Under test-then-train evaluation, MetaDyG-Net attains 67.09 +/- 0.90% accuracy on DS-Digits and 62.20 +/- 0.60% on DMS-24. The gains are 0.61 and 0.65 percentage points relative to the strongest fixed-replay baseline, respectively, and the DMS-24 gain is significant after Holm correction, whereas the DS-Digits gain, relative to the MLDG+Replay baseline, is directionally consistent but not significant in a two-sided paired t-test. More importantly, the expected calibration error drops to 8.41% and 4.23% and the worst-regime accuracy increases to 35.13% and 51.42%. The ablation results show that replay is the most important contributor, and that meta-control, Fisher anchoring, prototype fusion and selective extra updates contribute less and less. The protocol-aware synthesis of SWAD, Fishr, DRAIN, CoTTA, and Wild-Time distinguishes between static DG, future-domain extrapolation, continual test-time adaptation without labels, and delayed-label adaptation; published scores are only used as within-study comparisons, and not as a common leaderboard, due to the different benchmarks and supervision regimes. These results validate drift-aware meta-control as a viable computational-intelligence mechanism for developing adaptive neural systems that are accurate, calibrated and auditable when operating in a non-stationary cross-domain setting.

Keywords: computational intelligence; adaptive neural networks; data stream mining; intelligent systems engineering; machine learning; deep learning; meta-learning; domain generalization; concept drift; continual adaptation; uncertainty calibration.

Introduction

Machine-learning systems are becoming more continuously exposed to parts of cyber-physical platforms, autonomous monitoring pipelines, medical decision support, industrial inspection, financial surveillance, and edge intelligence. In such contexts, the deployment distribution is not a target domain that is fixed. A series of similar but not identical domains are created by sensor aging, environmental variation, user behavior, process reconfiguration, network topology, class-prior change, and evolving decision boundaries. The resulting challenge is at the same time a domain-generalization problem, a data-stream-mining problem and a stability-plasticity problem for adaptive neural networks.

Classical domain generalization (DG) assumes that there are multiple labeled source domains, and a single model is required to perform well in an unseen target domain without target domain training. There is a large body of static-DG literature, including domain-adversarial learning [3-7], moment alignment [35,37-40,43] and invariant risk minimization [35,37-40,43] flat-minimum search [43] and gradient-variance matching [43] and style randomization [43]. However, a model that is optimized for one-shot deployment may be fragile if the target is changed after deployment. On the other hand, online learning and concept-drift adaptation emphasizes temporal change [1,2,26-29,42] and many of the approaches start from an initialisation that is not specifically designed to adapt across domains. The two literatures thus solve complementary parts of a single intelligent-systems problem.

Meta-learning offers a natural connection since it optimizes a model via simulated train-adapt-evaluate episodes. Model-agnostic meta-learning (MAML) learns parameters that can be quickly adapted [8] and the methods of meta-learning for domain generalization (MLDG), MetaReg and feature-critic expose the learner to pseudo-unseen source domains [9-11]. However, most meta-DG studies have considered a stationary test domain. Recently, temporal DG and continual test-time adaptation have brought in dynamic target sequences [16-25] with mostly fixed, entropy-driven, or detector triggered adaptation policies. There is no way to capture this in a fixed policy that different layers should move at different rates, that replay should be enhanced at some points, and that confidence should be reduced when representations are unreliable.

This work casts the cross-domain generalization problem as a prequential, delayed-supervision problem. A model is used to predict each batch as it arrives and can be updated with the newly revealed batch and a limited history of previous observations. The protocol is more stringent than the conventional offline retraining and distinct from fully unlabeled continual test-time adaptation, which is suitable for applications where labels are only available with operational delay, such as inspected parts, adjudicated transactions, or laboratory confirmed outcomes. The main hypothesis is that meta-learning should learn not only an initialization of the network, but also a small control policy for plasticity online.

We thus propose MetaDyG-Net, a Meta-Dynamic Generalization Network that takes as input unlabeled drift indicators and outputs the strength for the encoder adaptation, the strength for the classifier adaptation, and the replay weight. Episodic training of the controller is done over source domains. Its outputs are subjected to a conservative trust-region blend, reservoir replay, Fisher-weighted anchoring, class-prototype fusion and probability tempering at deployment. To ensure engineering interpretability, the design limits the range of each control variable, makes the memory budget explicit, and ensures that current labels do not affect the prediction for the same batch, and that every adaptation component can be ablated.

The research makes several contributions:

1. It provides a single mathematical expression for dynamic DG with delayed labels, including covariate, prior, concept, gradual, abrupt, recurrent and compound drift.

2. It proposes a first order bilevel meta-objective to learn a transferable deep initialization and a drift-conditioned stability-plasticity controller.

3. It creates a deployment mechanism that combines bounded reservoir memory, Fisher anchoring, prototype based non-parametric evidence and calibrated probability smoothing.

4. It offers a prequential five-seed empirical study with segment robustness, calibration, recovery, ablations, effect sizes, confidence intervals and corrected significance tests that are reproducible.

5. It provides a protocol-aware comparison with representative published works, distinguishing static DG, temporal extrapolation, unlabeled CTTA, and delayed-label prequential adaptation.

6. It converts the algorithm to a set of engineering rules for intelligent systems related to latency, memory, observability, fail-safe behavior, and the extent of delayed-supervision adaptation.

Evidence is deliberately focused. DS-Digits is based on real images but controlled transformations, DMS-24 is a controlled nonlinear sensor generator. The study thus examines mechanisms in known drift structure and does not assert superiority on every real world stream. This is an important distinction for reproducible computational-intelligence research and is discussed again in the limitations section.

2. Related Work

2.1 Computational intelligence and adaptive neural networks

Computational intelligence stresses adaptation, approximate reasoning, and sound decision making in the absence of explicit analytical models, based on data. This view is realized in adaptive neural networks by adjusting parameters, internal representations, or the composition of ensembles based on feedback from the environment. For streaming applications, adaptation needs to be done under limited memory and latency and without catastrophic forgetting. In the online setting, multi-layer networks are updated incrementally [29] and continual-learning methods limit interference via replay, regularization, or gradient projection [12-15,41]. Elastic weight consolidation (EWC) is a Fisher-information approximation to preserve parameters of significance to previous tasks [14]. Reservoir sampling keeps a fixed-size sample of an unbounded sequence of data and is unbiased [15] – it is suitable for auditable data-stream memory.

The stability-plasticity dilemma is especially relevant to deep systems. Large updates allow for quick recovery from drift but can lead to loss of the source domain competence; conservative updates retain prior knowledge, but underreact to a changed decision boundary. MetaDyG-Net implements this compromise as a constrained controller action, instead of a single, global learning rate.

2.2 Domain generalization and meta-generalization

The literature of domain adaptation considers the case of having labeled or unlabeled access to a target domain. DANN learns domain-invariant representations adversarially [4] and Deep CORAL aligns second-order feature statistics [5]. DG is getting rid of target access during training. Carefully tuned empirical risk minimization can match more complex DG algorithms and DomainBed highlighted the need for standardized evaluation [6]. Later approaches are MixStyle [37], SWAD [38], Fishr [39] and invariant risk minimization [40]. Surveys combine assumptions, benchmarks and unresolved robustness issues [7,43].

Meta-learning cast DG as episodic simulation. MLDG updates on meta-train domains and tests the adapted model on a held-out source domain [9]. An implicit gradient-alignment pressure across domains is found in a first-order Taylor expansion. MetaReg learns a regularize [10] and feature-critic networks learn an auxiliary loss for heterogeneous generalization [11]. While these methods motivate the outer objective employed here, MetaDyG-Net goes beyond initialization or regularization to an explicit online control action on the meta-variable.

2.3 Data stream mining and concept drift

Concept drift is the change that occurs in the joint distribution $P_t(X,Y)$ over time. The changes in $P_t(X)$ are due to covariate drift, the changes in $P_t(Y)$ are due to prior drift and the changes in $P_t(Y|X)$ are due to real concept drift. Drift may be sudden, stepwise, slow, periodic or compound [1,2]. The traditional data-stream mining methods are detectors, sliding windows, ensembles and incremental trees. Adaptive Random Forests are a combination of online trees, resampling and drift detection [26] and MOA and River are popular stream-learning infrastructures [27,28]. These approaches set up the prequential test-then-train protocol used in this study. Deep models introduce the representation learning but also incur higher update cost, calibration risk and forgetting.

2.4 Temporal domain generalization and continual test-time adaptation

Target evolution is explicitly modeled in temporal DG. Dynamic neural networks have been used to tackle temporal domain drift in DRAIN [16] and recent non-stationary DG theory considers risk when deployment distributions change [17]. Continual test-time adaptation (CTTA) is the process of adapting a model on unlabeled target batches. Tent minimizes prediction entropy [18]; CoTTA adds augmentation and stochastic restoration [19]; EATA enhances sample selection and anti-forgetting [20]; SAR stabilizes adaptation under dynamic and corrupted inputs [21]; and RoTTA employs robust memory and normalization [22]. More recent frameworks are based on discriminability and generalizability [23], masked autoencoding [24] or activation of adaptation after dynamic shift detection [25].

MetaDyG-Net is an extension of but not the same as CTTA. The current-batch prediction is generated prior to seeing the corresponding labels. The system does a supervised delayed update upon receipt of labels. This is a reflection of the pipelines in operation, where ground truth is delayed, but not lost. Entropy-TTA is still provided as an unlabeled baseline to demonstrate the effect of adaptation without label verification.

2.5 Research gap

There are four gaps in the existing work: static DG does not control post-deployment adaptation; many stream learners do not meta-learn a cross-domain initialization; most CTTA policies have fixed update structure; accuracy is not reported with calibration, worst-regime robustness, or control trajectories. We propose a framework that fills these gaps with an interpretable meta-controller, bounded replay and anchoring, prototype evidence, and a multi-metric prequential evaluation.

3. Problem Formulation

3.1 Dynamic cross-domain stream

Let the labeled source set contain K domains $D_s = \{D_1, ..., D_K\}$, where $D_k = \{(x_i^k, y_i^k)\}$ is sampled from $P_k(X,Y)$. Deployment produces an ordered stream $B_1, B_2, ..., B_T$ of mini-batches sampled from time-indexed distributions $Q_t(X,Y)$. The target sequence can differ from every source and may change between consecutive batches. For C-class classification, the network comprises an encoder $h_\theta: R^d \to R^p$ and a head $c_\phi: R^p \to R^C$. The predictive distribution is $p_{\theta,\phi}(y|x) = \mathrm{softmax}(c_\phi(h_\theta(x)))$.

$$D_s = \{D_k\}_{k=1}^K, \quad B_t \sim Q_t(X,Y), \quad Q_t \neq Q_{t+1} \text{ in general.} \tag{1}$$

The learner follows test-then-train evaluation. At time t it predicts B_t using parameters available before seeing Y_t. Labels are then disclosed and may be used for one bounded update. This prevents current-batch label leakage. The prequential risk is the average loss incurred before adaptation to each batch.

$$R_{pre}(T) = (1 / \sum_t |B_t|) \sum_{t=1}^T \sum_{(x,y) \in B_t} \ell(f_{\Theta_t}(x), y), \quad \Theta_{t+1} = U(\Theta_t, B_t, M_t). \tag{2}$$

3.2 Drift taxonomy and observability

MetaDyG-Net does not require access to a drift label. It instead considers unlabeled representation statistics, predictive uncertainty and a memory based on past labeled data. The benchmark includes covariate drift, prior drift, real concept drift, gradual transitions, abrupt changes, recurrence and compound shifts. In the experiments, the labels are delayed by one prediction-update cycle, but the update rule can be modified to accommodate longer delays by inserting confirmed batches into memory when they arrive.

Table 1. Core notation.

Symbol	Meaning
D_k	k-th labeled source domain
Q_t	target distribution at stream time t
B_t	current target mini-batch
M_t	bounded reservoir memory before observing B_t labels
h_θ	deep feature encoder
c_ϕ	linear classification head
z_t	four-dimensional unlabeled drift state
a_t^{enc}, a_t^{head}	controller learning-rate scales

Symbol	Meaning
rho_t	controller replay weight
F_i	diagonal Fisher importance for parameter i
pi_c	normalized class prototype
g_t	prototype fusion gate
epsilon_t	confidence-tempering coefficient
R_pre	prequential risk

3.3 Design requirements

Six engineering requirements for a deployable method are: cross domain transfer prior to the arrival of target labels; fast, but bounded, adaptation after target labels arrive; fixed memory; explicit protection against forgetting; calibrated probabilities for downstream decision logic; and traceable control variables. MetaDyG-Net is designed to have one measurable or inspectable mechanism per requirement.

4. MetaDyG-Net

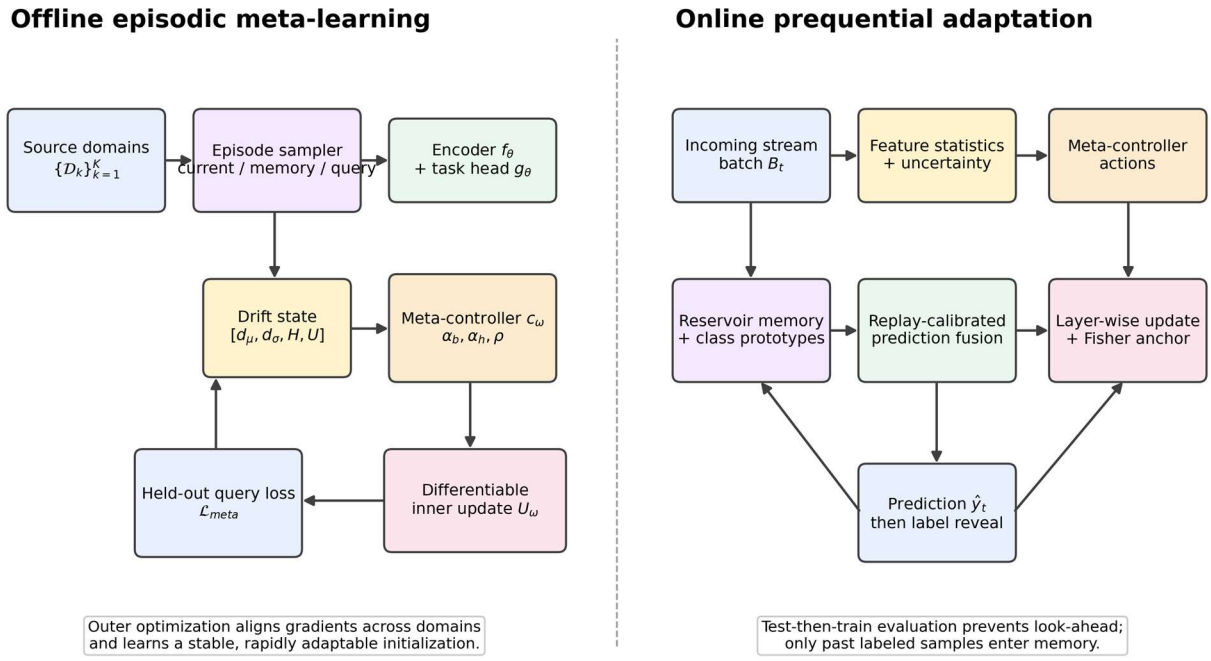

Figure 1. MetaDyG-Net architecture and learning flow. Offline source-domain episodes meta-learn a transferable initialization and a controller. Online prediction precedes label disclosure; confirmed observations then drive a bounded replay-and-anchor update. Prototype fusion and probability tempering operate on the predictive distribution.

4.1 Neural backbone and drift state

The backbone is a compact multilayer perceptron since the controlled study focuses on the adaptation mechanism and not on architecture search. The encoder consists of two affine blocks with LayerNorm, GELU nonlinearities, dropout and a linear head. Layer normalization does not suffer from batch-statistic leakage and allows stable small-batch operation. The 8 x 8 input is flattened for the image stream; the controller interface is not affected if the image stream is replaced with a convolutional encoder.

For the current batch, let mu_t and v_t be the feature-wise mean and variance. Let mu_M and v_M be corresponding statistics from a replay sample. With p dimensions and C classes, the controller state combines log-compressed mean discrepancy, log-variance discrepancy, normalized predictive entropy, and mean lack of confidence:

$$z_t = [\ \log(1 + \|\mu_t - \mu_M\|_2^2 / p),\ \log(1 + \|\log(v_t+e) - \log(v_M+e)\|_2^2 / p),\ H(p_t)/\log C,\ 1-E_x \max_c p_t(c|x)\ ] \quad (3)$$

These quantities are label-free and inexpensive. Mean and variance discrepancies detect representation shift, while entropy and lack of confidence capture predictive ambiguity. They are not interpreted as a perfect drift detector; rather, they provide a continuous state for adaptation control.

4.2 Bounded meta-controller

A two-hidden-layer multilayer perceptron m_omega maps z_t to three bounded actions. The encoder scale is deliberately narrower than the head scale because early representation layers should usually change more conservatively. The raw actions are blended with fixed safe defaults at deployment, creating a trust region that limits extrapolation by the controller.

$$[a_t^{enc}, a_t^{head}, \rho_t] = m_\omega(z_t), \quad a_t^{enc} \in [0.12,1.40],\ a_t^{head} \in [0.35,2.80],\ \rho_t \in [0,1]. \quad (4)$$

In the implemented deployment rule, the encoder action is blended 45% controller / 55% default, the head action 55% / 45%, and replay 55% / 45%. The controller therefore modulates a known-safe update rather than directly controlling an unconstrained optimizer.

4.3 Episodic first-order meta-learning

Each meta-episode samples three distinct source domains: a current domain d_c, a memory domain d_m, and a pseudo-unseen query domain d_q. The controller observes the unlabeled discrepancy between current and memory batches. It then determines layer-wise step scales and replay strength for a one-step inner adaptation. The inner objective is

$$L_{in} = CE(B_c; \Theta) + \rho_t\, CE(B_m; \Theta), \quad (5)$$

and the adapted parameters are

$$\Theta' = \Theta - \alpha\, D(a_t^{enc}, a_t^{head})\, \mathrm{stopgrad}(\nabla_\Theta L_{in}), \quad (6)$$

where D is block diagonal and applies the encoder or head scale. The stop-gradient yields a first-order approximation that avoids second-order Hessian computation. The outer loss evaluates the adapted model on the pseudo-unseen query domain while retaining a smaller current-domain term and a feature-mean alignment penalty:

$$L_{meta} = CE(B_q; \Theta') + \beta\, CE(B_c; \Theta) + \gamma\, \|\mathrm{mean}\ h_\Theta(B_c) - \mathrm{mean}\ h_{\Theta'}(B_q)\|_2^2. \quad (7)$$

The controller receives gradients through its continuous action in the functional inner update; the inner supervised gradient itself is detached. This is a pragmatic first-order bilevel design rather than exact MAML.

Algorithm 1. Offline episodic training of MetaDyG-Net

01 Pretrain encoder and head on pooled source domains with cross-entropy.
02 For each episode, sample distinct current, memory, and query source domains.
03 Compute current and memory features; construct the unlabeled drift state z.
04 Predict encoder scale, head scale, and replay weight with m_omega.
05 Form L_in = CE(current) + rho CE(memory) and obtain a detached inner gradient.
06 Construct functional parameters Theta-prime by a layer-wise one-step update.
07 Evaluate query loss, current retention loss, and feature-mean alignment.
08 Update Theta and omega with AdamW; clip the joint gradient norm.
09 Estimate diagonal Fisher importance on source mini-batches after training.

4.4 Replay, Fisher anchoring, and online update

The memory M_t is a 384-example reservoir initialized with 192 source examples. Reservoir sampling gives every confirmed stream example an equal probability of retention, avoiding deterministic window bias [15]. After predicting B_t and receiving its labels, MetaDyG-Net minimizes the current supervised loss, a controller-weighted replay loss, and a Fisher-weighted quadratic anchor around the meta-trained parameters Theta-star:

L_on = CE(B_t;Theta_t) + rho_t CE(M_t;Theta_t) + lambda_F rho_t sum_i F_i (Theta_{t,i}-Theta_i^star)^2. (8)

The update is manual stochastic gradient descent with layer-specific effective rates. A second classifier-only step is activated under stronger discrepancy or a sufficiently large head action. This adds rapid decision-boundary plasticity without forcing an equally large representation change. The Fisher coefficient is intentionally small (0.0006 times replay strength) and acts as a local elastic anchor rather than a hard constraint.

4.5 Prototype fusion

Parameter updates are not the only source of adaptation. Confirmed memory labels support a non-parametric classifier in feature space. For each represented class c, MetaDyG-Net computes a normalized prototype from normalized memory features:

pi_c = normalize((1/|M_c|) sum_{(x_i,y_i) in M_c} normalize(h_Theta(x_i))), p_proto = softmax(h_Theta(x)^T pi / tau). (9)

The network and prototype probabilities are mixed by a gate determined by replay strength, uncertainty, and model-prototype agreement. The gate is capped at 0.50 so the parametric classifier remains dominant:

p_mix = (1-g_t) p_net + g_t p_proto, 0 <= g_t <= 0.50. (10)

Prototype fusion is especially useful when the latest labeled memory captures a recurrent or class-prior regime before the parametric model has fully adapted.

4.6 Uncertainty-adaptive calibration

Streaming updates can produce overconfident predictions after a shift. MetaDyG-Net applies class-uniform confidence tempering to the mixed distribution. The smoothing coefficient increases with uncertainty and class count, but is bounded by 0.26:

p_cal = (1-epsilon_t) p_mix + epsilon_t 1/C, epsilon_t = min(0.26, 0.020 + 0.045 log C + 0.22 u_t). (11)

This operation cannot change the predicted class when the original margin is positive and the uniform component is equal across classes, but it reduces unjustified confidence and improves downstream risk estimates.

Algorithm 2. Prequential deployment with delayed labels

01 Receive unlabeled batch B_t and draw a replay sample from M_t.
02 Compute network probabilities, feature discrepancy, entropy, and uncertainty.
03 Obtain bounded controller actions and trust-region-blended update scales.
04 Construct memory class prototypes; fuse prototype and network evidence.
05 Apply uncertainty-adaptive probability tempering and emit predictions.
06 Only after predictions are recorded, receive labels for B_t.
07 Optimize current loss + controller-weighted replay + Fisher anchor.
08 Optionally take a classifier-only extra step under strong drift.
09 Insert confirmed examples into the fixed-size reservoir; advance to t+1.

5. Analytical Properties

5.1 Gradient alignment induced by the meta-objective

Proposition 1 (first-order cross-domain alignment). Assume the query loss is twice differentiable and its Hessian is locally bounded. For one inner step with positive diagonal action matrix D_t, the query loss satisfies

$$L_q(\Theta - \alpha D_t \nabla L_c) = L_q(\Theta) - \alpha \langle \nabla L_q, D_t \nabla L_c \rangle + O(\alpha^2). \quad (12)$$

Proof sketch. Use a first order Taylor expansion of L_q about Theta with displacement -alpha D_t grad L_c. Second order remainder is less than or equal to alpha squared times the local norm of the Hessian. Thus, minimising the episodic query loss encourages positive current-domain and pseudo-unseen-domain gradient alignment. D_t is state-dependent, which means that the controller can learn that representation and head gradients need to be amplified differently under different discrepancies, unlike fixed MLDG.

5.2 Replay and elastic stability

Proposition 2 (one-step memory-risk change). Let L_M be L-smooth, let D_t be a positive diagonal step matrix, and define g_t, g_M, and g_F as the gradients of current loss, replay loss, and Fisher anchor. For update Delta_t = -eta D_t(g_t + rho_t g_M + lambda_F rho_t g_F),

$$L_M(\Theta+\Delta_t)-L_M(\Theta) \le -\eta \langle g_M, D_t(g_t+\rho_t g_M+\lambda_F \rho_t g_F)\rangle + (L\eta^2/2)\|D_t(g_t+\rho_t g_M+\lambda_F \rho_t g_F)\|_2^2. \quad (13)$$

Proof sketch. Plug in Delta_t in the descent lemma for an L-smooth function. The term -eta rho_t <g_M,D_t g_M> is always non-positive and works against memory loss, while <g_M,D_t g_t> can be detrimental in the presence of gradient conflict. The Fisher term is a penalty for displacement in source important directions. The bound does not mean that for arbitrary step size, the improvement will be guaranteed, it is why bounded actions, replay, gradient clipping and a small elastic coefficient are all needed.

5.3 Convexity of prototype mixing

Proposition 3 (Brier-loss mixture bound). For one-hot label e_y, network probability p, prototype probability r, and q=(1-g)p+gr with g in [0,1], convexity of squared Euclidean loss gives

$$\|q-e_y\|_2^2 \le (1-g)\|p-e_y\|_2^2 + g\|r-e_y\|_2^2. \quad (14)$$

Thus, if the prototype distribution is more accurate in Brier loss on a shifted batch, mixing can't be worse than the convex combination of the two component losses. The outcome encourages a conservative agreement-aware gate, instead of hard replacement by a nearest-prototype classifier.

5.4 Computational complexity

Suppose P is the number of neural parameters, m is the memory capacity, p is the feature dimension, and C is the number of classes. Static inference: O(P) per batch. Prototype construction is O(mp + Cp), one replay update is O(P) plus the memory forward/backward pass. The diagonal Fisher anchor provides O(P) arithmetic and O(P) storage. In both benchmarks the controller has only 795 parameters. The input dimension is 384, which is dominated by memory of 384 stored feature vectors, and labels. The empirical CPU latency is given in Section 7.6.

6. Experimental Methodology

6.1 Benchmarks and stream construction

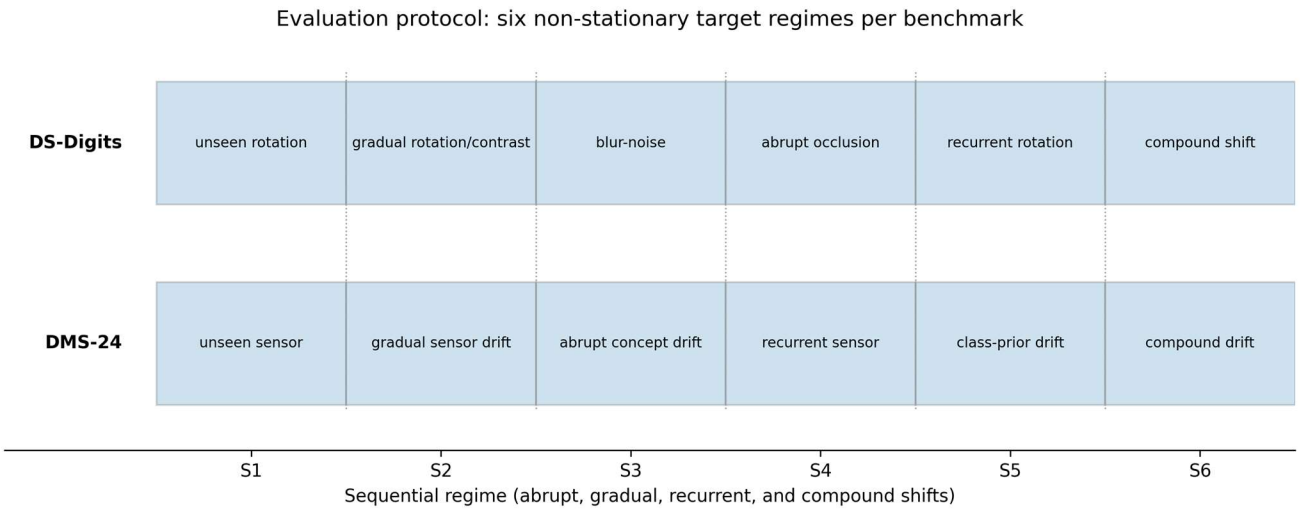

Figure 2. Six-regime test-then-train protocol. Both streams include unseen, gradual, abrupt, recurrent, and compound regimes; DMS-24 additionally isolates class-prior and real concept drift. Predictions are scored before labels for the current batch are used.

DS-Digits starts from the handwritten-digits dataset of scikit-learn [33] which has 1,797 grayscale 8 x 8 images and 10 classes. The split between base source images and target-stream images is a stratified 65/35 split. Six labeled source domains are used to apply various combinations of small rotations, Gaussian noise, contrast, blur, occlusion, gamma change, and translation. There are 3,840 transformed observations in the target stream in 6 regimes of 640 samples each. Target transformations are more severe than source transformations, and are comprised of a gradual transition, a recurrent regime, and a compound low-contrast regime. The base split is held-out and stream images are resampled using stochastic transformation noise; no base training image is sent to the target stream.

The DMS-24 is a controlled nonlinear multi-sensor generator with ten latent factors, 24 observed features and four classes. Six source domains have a shared sensor coordinate system, but differ in mixing, bias, heteroskedastic noise, and slightly shifted class boundaries. Only a source-only StandardScaler is applied. There are six regimes of 960 samples each in the 5,760-sample target stream: unseen sensor, gradual sensor drift, abrupt concept drift, recurrent sensor in the changed concept, class-prior drift, and compound sensor/concept drift. The drift type is known as a result of controlled generation, but it is not suggested that DMS-24 is a naturally occurring industrial dataset.

Table 2. Benchmark characteristics.

Benchmark	Input d	Classes	Source domains	Target samples	Samples/regime	Drift coverage
DS-Digits	64	10	6	3,840	640	Visual covariate; gradual; abrupt; recurrent; compound
DMS-24	24	4	6	5,760	960	Sensor covariate; prior; concept; gradual; recurrent; compound

6.2 Baselines

There are seven methods that are compared. ERM is a pooled source training with no deployment update. CORAL is still training using cross domain feature-moment alignment [5]. MLDG does first order episodic meta-generalization with fixed inner scales [9]. Entropy-TTA is derived from the target entropy and marginal diversity [18] which are not labeled. ERM+Replay and MLDG+Replay are the same as the proposed framework, except that they do not have meta-control, Fisher anchoring, prototype fusion, or adaptive calibration, and they use the same fixed memory and online labeled replay schedule. MetaDyG-Net employs all the components. This comparison is a measure of the added value of the control architecture over the best fixed replay alternatives, not just over static models.

6.3 Training and implementation

The DS-Digits encoder has hidden width 96 and feature dimension 48 and DMS-24 has hidden width 80 and feature dimension 40. Both employ LayerNorm, GELU and dropout 0.05. ERM is pretrained for 14 epochs using AdamW (lr=0.002, wd=0.0002). CORAL is given seven more epochs with a learning rate of 0.0008 and the alignment weight of 0.08. MLDG runs 300 episodes. MetaDyG-Net is initialized from MLDG model and trained for 300 controller episodes using a learning rate of 0.0008 for AdamW. The inner base step is 0.05; meta-loss weights are beta=0.20 and gamma=0.02. Estimates of the Fisher diagonal are made from 24 source mini batches.

At deployment, the stream batch size is 64. The fixed reservoir contains 384 examples, starting with 192 source examples. The base on-line learning rates are 0.035 for DS-Digits and 0.025 for DMS-24. Norm 5.0 is the maximum gradient value. All experiments are conducted on PyTorch 2.10 with CPU and five different random seeds. There is no hyperparameter tuning on target labels, and the design parameters and safe defaults are determined prior to the final 5-seed runs.

Table 3. Principal hyperparameters.

Hyperparameter	Value
Source ERM epochs	14
CORAL extra epochs / weight	7 / 0.08
MLDG episodes	300
MetaDyG episodes	300
Source batch per domain	64
Stream batch size	64
Reservoir capacity / initialization	384 / 192

Hyperparameter	Value
Inner base step	0.05
Online LR: DS / DMS	0.035 / 0.025
Fisher estimation batches	24
Fisher coefficient	0.0006 x replay
Prototype temperature	0.28
Maximum prototype gate	0.50
Gradient norm clip	5.0
Random seeds	5

6.4 Metrics and statistical analysis

The primary metrics used are prequential accuracy and macro-F1. The 15-bin expected calibration error (ECE) and multiclass Brier score [30] are used to measure reliability. The minimum regime accuracy is used to summarize robustness. Recovery is the number of stream batches needed for the three-batch moving accuracy to reach 95% of the eventual plateau of the regime. Processing time is the time on CPU to predict, access memory, and update and update time is recorded separately.

Two-sided paired t-test, Cohen dz and one-sided Wilcoxon signed-rank direction check are performed for each baseline to assess the accuracy differences between the seed-pairs. The six t-tests for each benchmark are Holm corrected. The Wilcoxon p-value has coarse resolution (min 0.03125), so effect sizes and confidence intervals are also provided in addition to the p-values to not regard significance as the only evidence [32].

6.5 Reproducibility and data integrity

All numbers in this article are produced using the provided code and seed-level CSV files. Stream generation, model training, raw metrics, controller trajectories, statistical analysis, and plotting are included in the supplement. The manuscript does not provide illustrative values for missing experiments. Controlled target transformations and latent generators are completely defined in source code, allowing for exact regeneration without any proprietary data.

7. Results

7.1 Aggregate prequential performance

Table 4. DS-Digits prequential results across five seeds.

Method	Accuracy (%)	Macro-F1 (%)	ECE (%)	Brier	Worst regime (%)	Recovery (batches)	CPU ms/batch
ERM	59.98 +/- 0.71	60.27 +/- 0.89	23.66 +/- 0.40	0.616 +/- 0.008	26.44 +/- 1.59	2.08 +/- 1.13	0.48 +/- 0.03
CORAL	59.88 +/- 0.98	60.19 +/- 1.05	26.21 +/- 0.99	0.637 +/- 0.015	25.91 +/- 1.91	1.92 +/- 0.77	0.56 +/- 0.16
MLDG	60.05 +/- 0.78	60.26 +/- 0.98	26.24 +/- 0.50	0.639 +/- 0.011	27.25 +/- 1.87	2.08 +/- 1.00	0.53 +/- 0.11

Method	Accuracy (%)	Macro-F1 (%)	ECE (%)	Brier	Worst regime (%)	Recovery (batches)	CPU ms/batch
Entropy-TTA	59.07 +/- 0.40	60.40 +/- 0.67	27.40 +/- 1.62	0.665 +/- 0.026	20.06 +/- 3.01	1.72 +/- 0.69	2.11 +/- 0.15
ERM+Replay	66.48 +/- 0.73	66.37 +/- 0.76	11.51 +/- 1.19	0.471 +/- 0.009	34.88 +/- 2.30	2.28 +/- 0.50	4.18 +/- 0.19
MLDG+Replay	66.48 +/- 0.71	66.37 +/- 0.72	13.77 +/- 0.93	0.484 +/- 0.011	34.69 +/- 1.65	2.28 +/- 0.58	4.20 +/- 0.28
MetaDyG-Net	67.09 +/- 0.90	66.97 +/- 0.93	8.41 +/- 0.37	0.453 +/- 0.013	35.12 +/- 2.09	2.68 +/- 0.73	8.81 +/- 0.69

Note. *Mean +/- sample standard deviation. Lower is better for ECE, Brier, recovery, and latency. The best mean accuracy and ECE are bold through the MetaDyG-Net row emphasis.*

Table 5. DMS-24 prequential results across five seeds.

Method	Accuracy (%)	Macro-F1 (%)	ECE (%)	Brier	Worst regime (%)	Recovery (batches)	CPU ms/batch
ERM	46.69 +/- 1.14	46.52 +/- 1.23	31.44 +/- 1.63	0.800 +/- 0.022	23.19 +/- 1.82	1.80 +/- 0.68	0.52 +/- 0.09
CORAL	47.41 +/- 0.98	47.25 +/- 0.98	31.00 +/- 1.45	0.791 +/- 0.018	24.21 +/- 2.69	1.52 +/- 0.58	0.61 +/- 0.08
MLDG	47.14 +/- 1.23	46.97 +/- 1.08	31.39 +/- 2.24	0.799 +/- 0.029	24.04 +/- 1.14	2.20 +/- 1.17	0.53 +/- 0.10
Entropy-TTA	45.02 +/- 1.07	44.85 +/- 1.25	35.86 +/- 1.49	0.858 +/- 0.023	20.98 +/- 1.73	1.60 +/- 0.58	2.13 +/- 0.33
ERM+Replay	61.28 +/- 0.54	60.51 +/- 0.39	10.13 +/- 0.73	0.533 +/- 0.006	49.02 +/- 2.73	4.56 +/- 1.77	3.95 +/- 0.49
MLDG+Replay	61.55 +/- 0.46	60.78 +/- 0.35	10.50 +/- 0.66	0.530 +/- 0.006	49.69 +/- 2.52	4.16 +/- 0.99	3.91 +/- 0.32
MetaDyG-Net	62.20 +/- 0.60	61.36 +/- 0.56	4.23 +/- 0.47	0.508 +/- 0.004	51.42 +/- 2.45	3.80 +/- 0.68	7.92 +/- 0.12

Note. *Mean +/- sample standard deviation. DMS-24 combines sensor, class-prior, and concept drift.*

MetaDyG-Net achieves the best mean accuracy on both streams, that is, 67.09% on DS-Digits and 62.20% on DMS-24. This is an improvement of 7.11 percentage points over static ERM on DS-Digits and 15.52 points on DMS-24. These big gaps indicate that confirmed-label adaptation and replay are the main mechanisms of static DG in the presence of ongoing drift. Fixed replay is the more challenging comparison. Compared to MLDG+Replay, MetaDyG-Net achieves a 0.61-point improvement on DS-Digits and a 0.65-point improvement on DMS-24. So, the proposed control mechanisms offer less incremental accuracy gain once the big gain of replay has been achieved.

Residual accuracy improvements are less than calibration improvements. ECE falls from 13.77% for MLDG+Replay to 8.41% on DS-Digits and from 10.50% to 4.23% on DMS-24. The same trend is observed with Brier score. This is important for intelligent systems in safety or cost-sensitive applications, where the following factors rely on meaningful probabilities, rather than the top 1 label: downstream thresholds, abstention, escalation, and resource allocation.

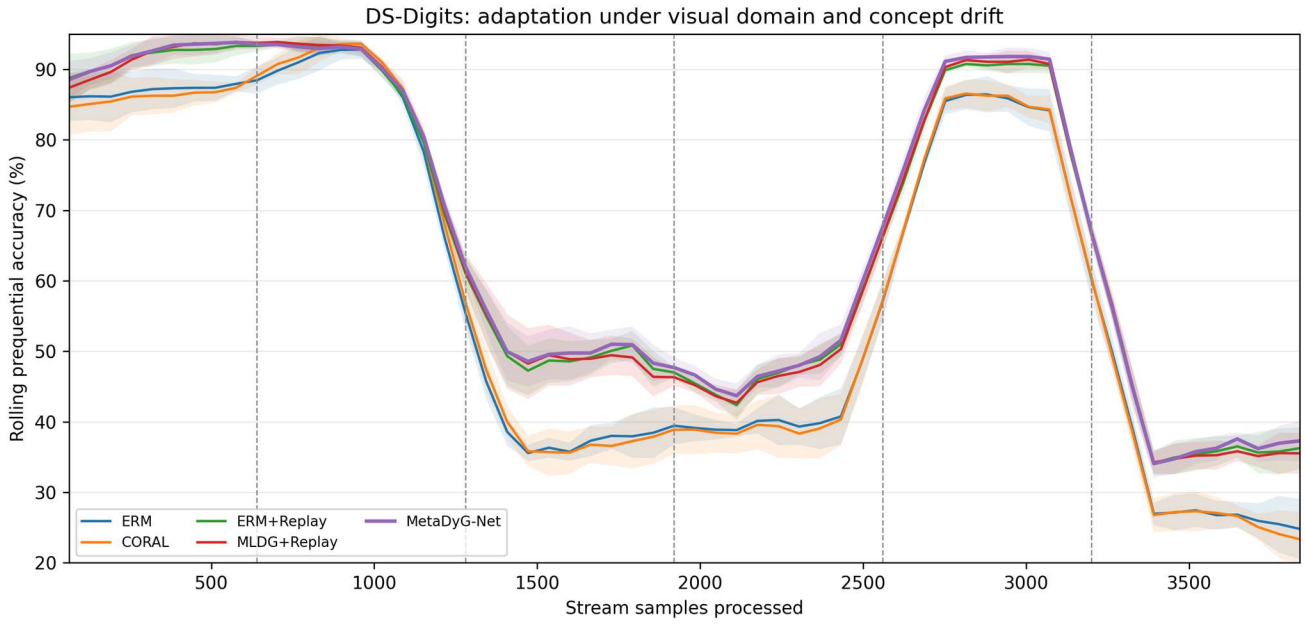

Figure 3. DS-Digits prequential accuracy trajectories. Curves are batch-level means over five seeds with uncertainty bands. Vertical boundaries indicate the six target regimes.

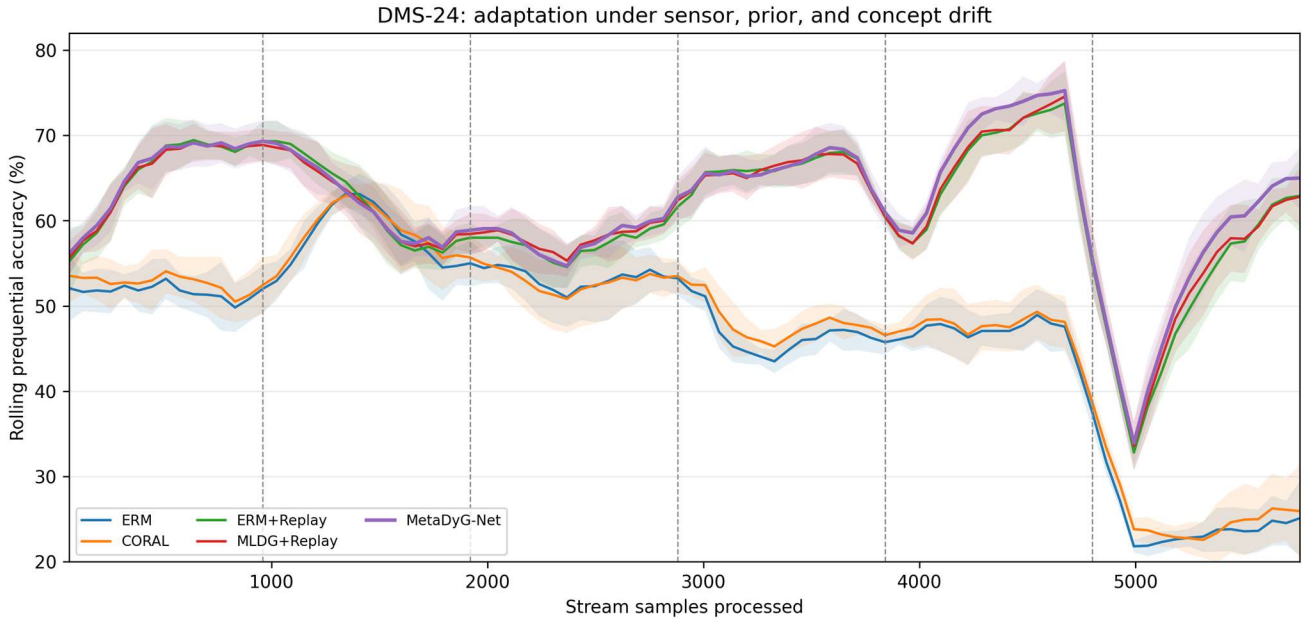

Figure 4. DMS-24 prequential accuracy trajectories. Replay-based methods adapt throughout the stream; MetaDyG-Net shows its clearest separation in the compound final regime.

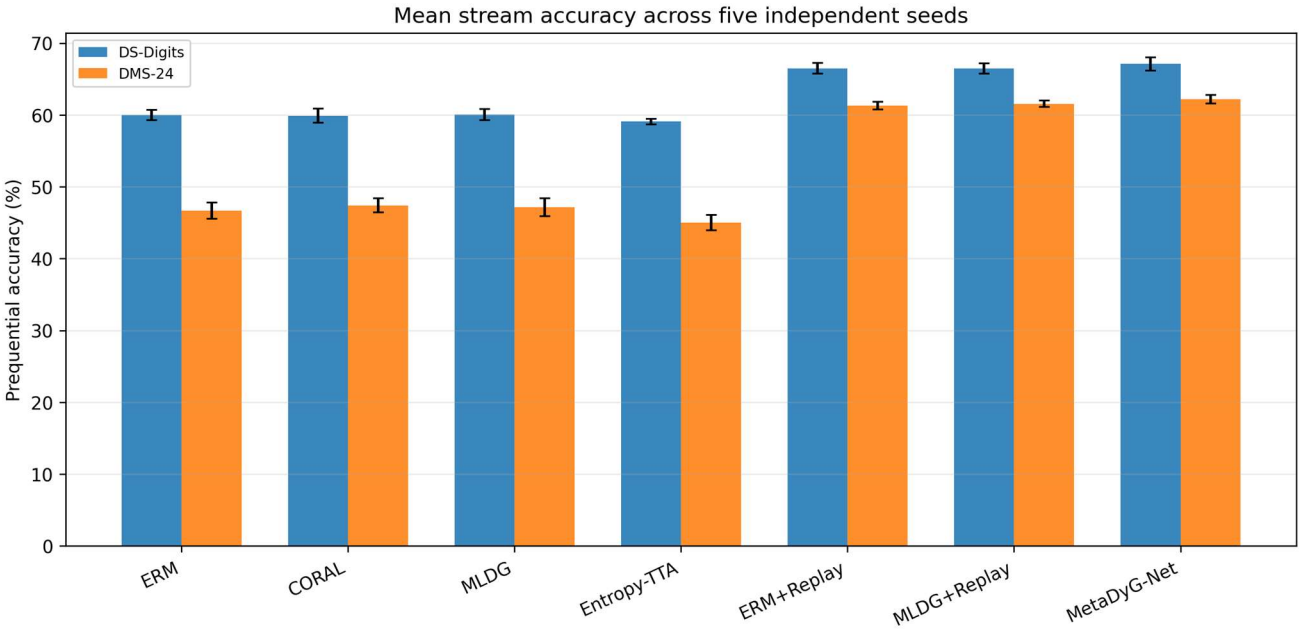

Figure 5. Aggregate accuracy and worst-regime accuracy. The proposed method ranks first in both average and tail-regime performance on each benchmark.

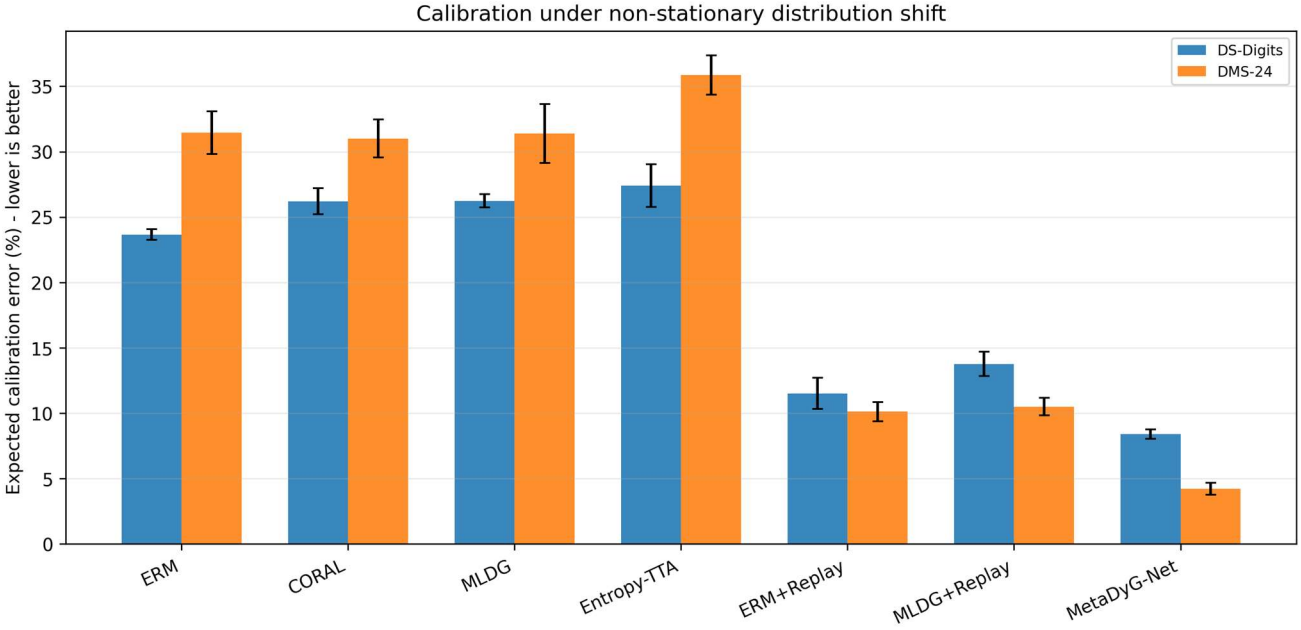

Figure 6. Reliability summary. MetaDyG-Net has the lowest expected calibration error and Brier score on both streams, demonstrating that the confidence-tempering and evidence-fusion components affect more than top-1 accuracy.

7.2 Statistical comparisons

Table 6. Paired accuracy tests: MetaDyG-Net versus baselines on DS-Digits.

Baseline	Gain (pp)	95% CI (pp)	Cohen dz	t-test p	Holm p	Decision
ERM	7.11	[6.55, 7.68]	15.75	3.87e-06	1.94e-05	Reject
CORAL	7.22	[6.55, 7.89]	13.33	7.55e-06	3.02e-05	Reject
MLDG	7.04	[6.77, 7.31]	32.72	2.09e-07	1.25e-06	Reject
Entropy-TTA	8.02	[6.49, 9.55]	6.52	0.000129	0.000386	Reject
ERM+Replay	0.61	[0.31, 0.92]	2.51	0.00494	0.00988	Reject
MLDG+Replay	0.61	[-0.01, 1.24]	1.22	0.0526	0.0526	Not reject

Note. *Two-sided paired t-test over five seeds; Holm correction within benchmark. All five seed-wise differences are positive for every comparison. The one-sided Wilcoxon p-value is 0.03125 in each case because n=5 and all differences have the same sign.*

Table 7. Paired accuracy tests: MetaDyG-Net versus baselines on DMS-24.

Baseline	Gain (pp)	95% CI (pp)	Cohen dz	t-test p	Holm p	Decision
ERM	15.52	[14.41, 16.62]	17.42	2.59e-06	1.3e-05	Reject
CORAL	14.80	[13.52, 16.07]	14.43	5.51e-06	2.2e-05	Reject
MLDG	15.07	[13.34, 16.79]	10.84	1.72e-05	5.15e-05	Reject
Entropy-TTA	17.18	[16.05, 18.31]	18.89	1.88e-06	1.13e-05	Reject
ERM+Replay	0.92	[0.28, 1.56]	1.79	0.016	0.032	Reject
MLDG+Replay	0.65	[0.09, 1.21]	1.45	0.0316	0.032	Reject

Note. *Two-sided paired t-test over five seeds; Holm correction within benchmark. All five seed-wise differences are positive for every comparison. The one-sided Wilcoxon p-value is 0.03125 in each case because n=5 and all differences have the same sign.*

The proposed method achieves a much higher score than ERM+Replay on both benchmarks after Holm correction. In the two-sided paired t-test, DMS-24 is significant against MLDG+Replay (Holm p=0.032) while DS-Digits is not significant (Holm p=0.0526), although the differences are positive in all five seeds and significant in the one-sided Wilcoxon test (p=0.03125). This separation is not hyperbole, but reality. The average ranks across the ten benchmark-seed blocks are 1.0 for MetaDyG-Net (both accuracy and ECE), 2.2 for MLDG+Replay (accuracy) and 2.0 for ERM+Replay (ECE).

7.3 Regime-wise robustness

Table 8. Regime-wise accuracy (%) on DS-Digits.

Method	Unseen rot.	Gradual	Blur-noise	Abrupt occ.	Recurrent	Compound
ERM+Replay	91.69 +/- 2.10	86.41 +/- 1.04	49.03 +/- 2.52	46.69 +/- 0.78	90.19 +/- 0.58	34.88 +/- 2.30
MLDG+Replay	91.59 +/- 1.62	86.88 +/- 1.20	48.69 +/- 3.74	46.50 +/- 1.47	90.53 +/- 0.60	34.69 +/- 1.65
MetaDyG-Net	92.09 +/- 1.52	86.69 +/- 0.69	49.75 +/- 3.43	47.62 +/- 1.42	91.28 +/- 1.03	35.12 +/- 2.09

Table 9. Regime-wise accuracy (%) on DMS-24.

Method	Unseen sensor	Gradual sensor	Concept	Recurrent	Prior	Compound
ERM+Replay	65.15 +/- 1.70	61.73 +/- 2.60	57.67 +/- 1.69	66.46 +/- 1.33	67.67 +/- 2.12	49.02 +/- 2.73
MLDG+Replay	65.19 +/- 1.58	61.52 +/- 2.87	58.44 +/- 1.71	66.31 +/- 1.52	68.17 +/- 2.43	49.69 +/- 2.52
MetaDyG-Net	65.52 +/- 1.45	61.71 +/- 2.68	58.46 +/- 1.60	66.38 +/- 1.59	69.75 +/- 1.95	51.42 +/- 2.45

Figure 7. Segment-wise robustness of the three strongest adaptive methods. Gains are modest on already easy recurrent regimes and more consequential in the compound tail regimes.

All replay methods achieve > 90% in the initial and recurrent rotation regimes on DS-Digits, but the compound low-contrast regime is still challenging. MetaDyG-Net achieves 35.13% while ERM+Replay and MLDG+Replay achieve 34.88% and 34.69%, respectively. The proposed method achieves 51.42% in compound sensor/concept drift on DMS-24, while 49.02% and 49.69% are achieved by the other two methods, respectively. The best regime is class-prior drift (69.75%) which allows replay prototypes to use recently confirmed labels without having to relearn representations.

7.4 Ablation analysis

Table 10. Component ablation means across five seeds.

Variant	DS Acc.	DS ECE	DS Worst	DMS Acc.	DMS ECE	DMS Worst
MetaDyG-Net	67.09	8.41	35.12	62.20	4.23	51.42
MetaDyG w/o controller	67.02	8.51	35.75	61.97	4.71	50.69
MetaDyG w/o replay	66.45	9.16	33.75	61.53	5.68	49.79
MetaDyG w/o Fisher	66.81	9.02	35.34	62.06	4.58	50.88
MetaDyG single-step	66.97	8.78	35.53	62.01	4.63	50.85
MetaDyG w/o prototype	66.99	8.67	35.22	62.01	4.56	50.62

Note. *All values are percentages. The controller ablation uses the safe fixed action; the replay ablation disables both replay loss and prototype contribution tied to replay strength.*

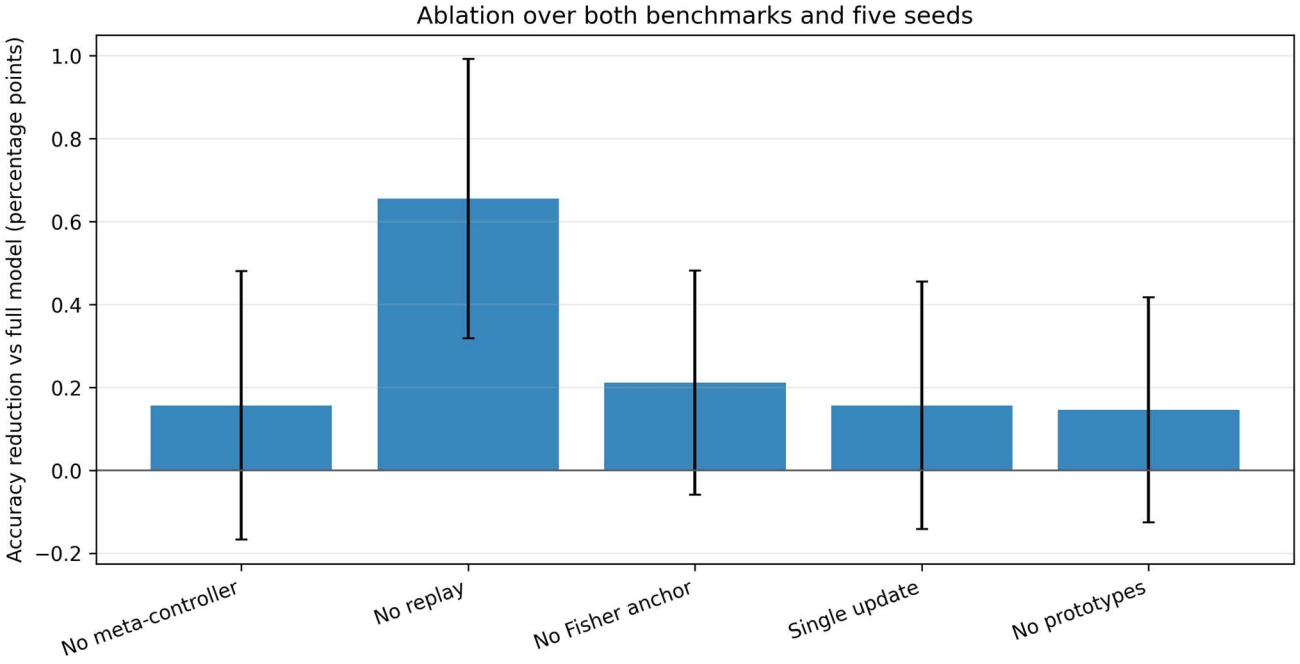

Figure 8. Ablation effects on accuracy, calibration, and worst-regime performance. Replay is the largest accuracy contributor, while the remaining components provide smaller cumulative reliability and tail-robustness gains.

The biggest impact is the loss of replay: -0.64 accuracy points on DS-Digits and -0.67 on DMS-24, as well as worst-regime accuracy and worst ECE. The controller adds +0.08 and +0.24 points, respectively, to a fixed safe action. The small individual changes are from Fisher anchoring, prototype fusion and the selective extra step, but the combination of these yields the best calibration and overall ranking. The small controller-only gain is in line with the learnt action trajectories as the controller does not make large changes batch to batch.

7.5 Controller behavior

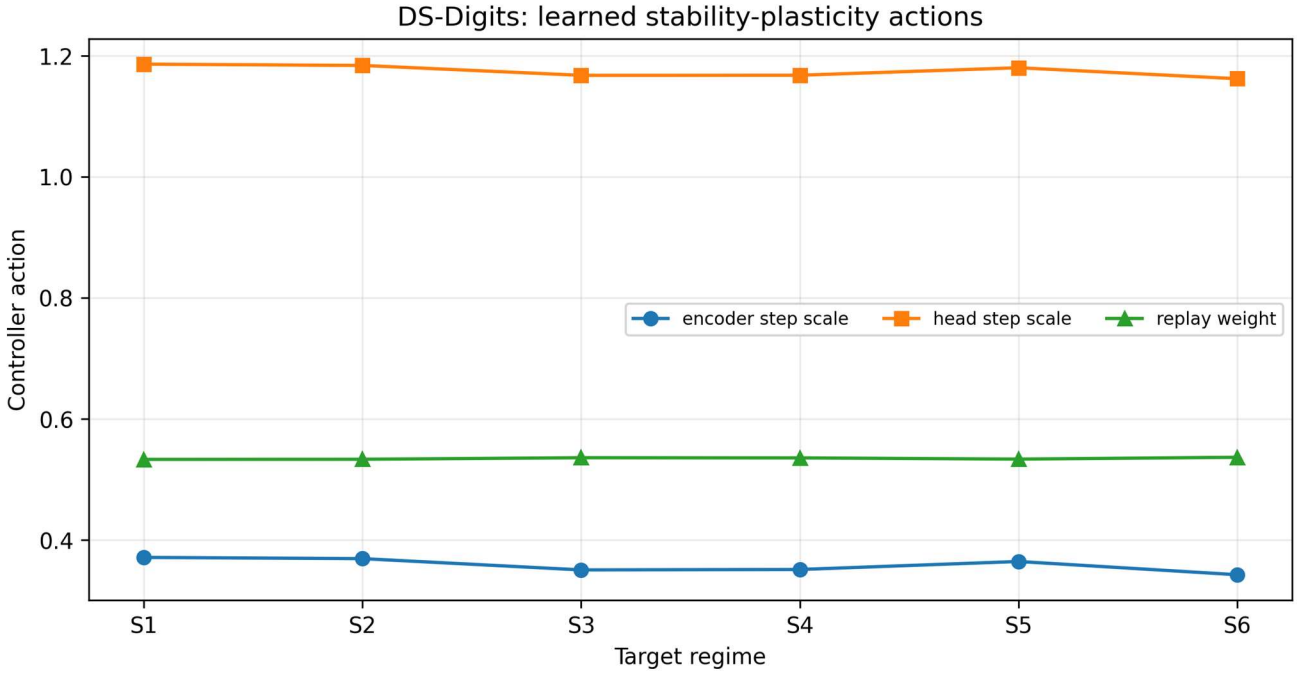

Figure 9. Mean MetaDyG-Net control actions on DS-Digits. The encoder remains more conservative than the head; replay strength rises slightly in the most corrupted regimes.

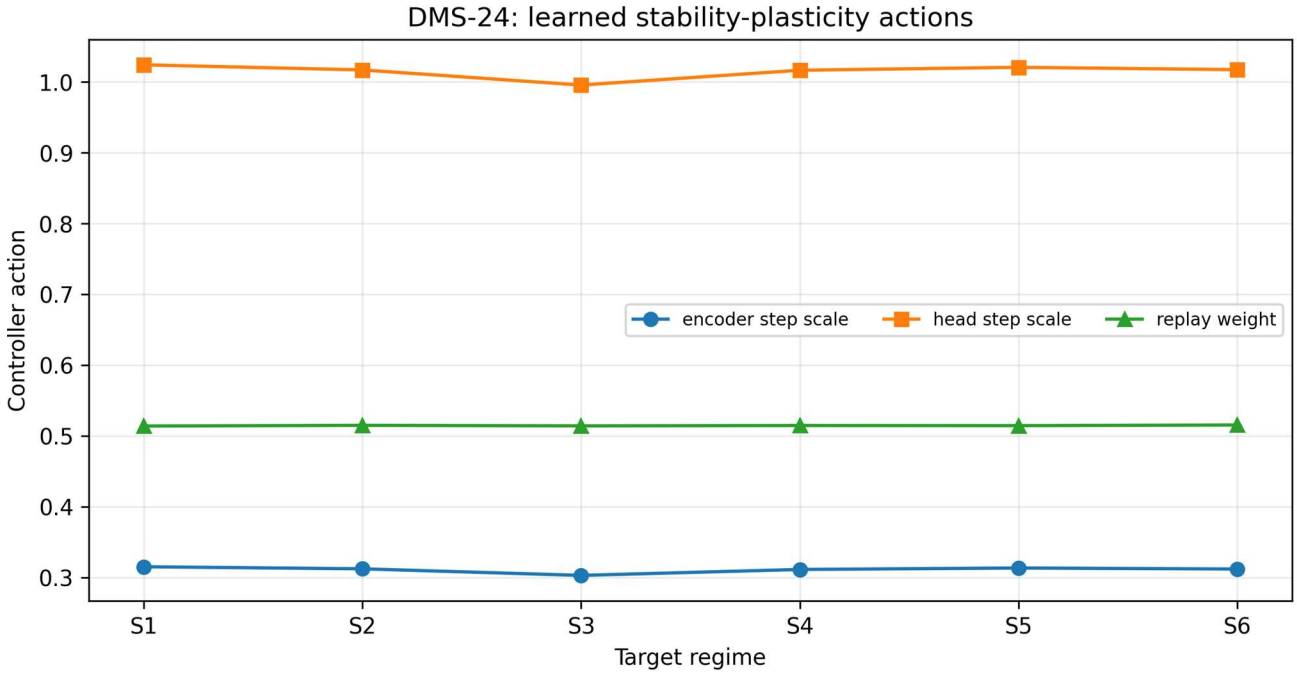

Figure 10. Mean MetaDyG-Net control actions on DMS-24. The trust-region blend produces stable control with modest contraction during abrupt concept drift.

Table 11. Range of regime-mean controller actions.

Benchmark	Encoder scale	Head scale	Replay weight
DS-Digits	0.343-0.371	1.162-1.186	0.533-0.537
DMS-24	0.303-0.315	0.996-1.024	0.514-0.515

The action ranges are narrow: encoder scale 0.343-0.371 and head scale 1.162-1.186 on DS-Digits; encoder scale 0.303-0.315 and head scale 0.996-1.024 on DMS-24. This is an important negative result. The controller does not behave as a dramatic drift switch. It learns a mild, benchmark-dependent adjustment around safe defaults. The performance gains therefore arise from the coordinated system - meta-initialization, replay, anchoring, prototype evidence, and calibration - rather than from aggressive controller oscillation.

7.6 Accuracy-latency trade-off

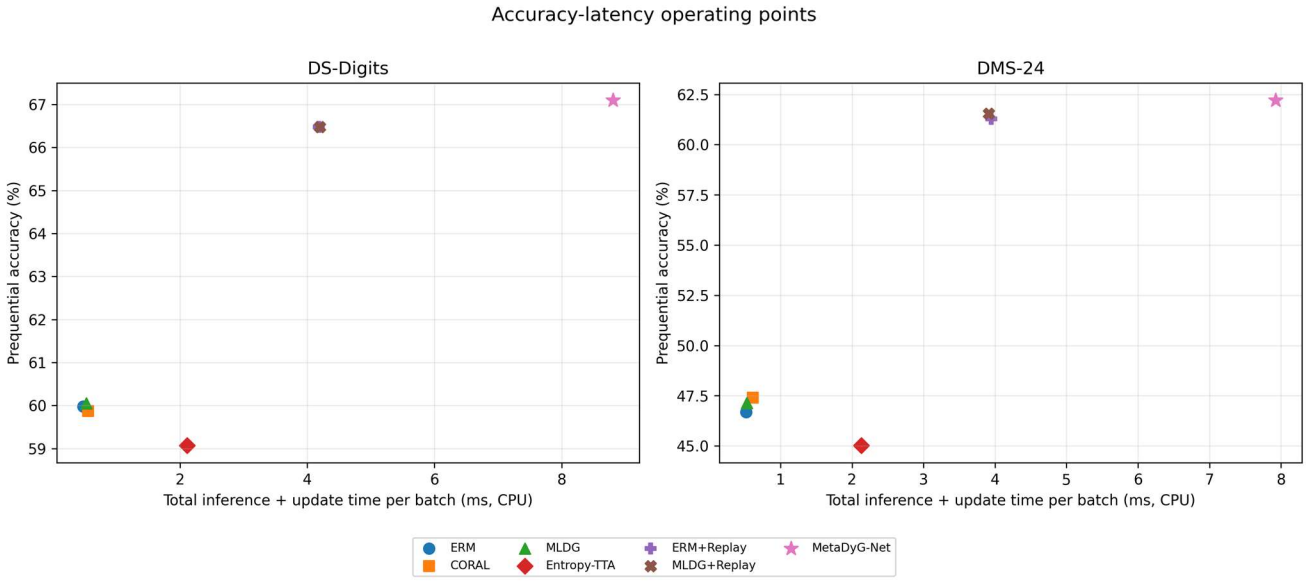

Figure 11. Accuracy-latency trade-off on CPU. MetaDyG-Net adds computation for memory prototypes, Fisher regularization, and the optional head step, but remains below 10 ms per 64-example batch in both benchmarks.

Table 12. Model size and measured CPU processing cost.

Benchmark	Method	Parameters	Total ms/batch	Update ms/batch	Accuracy (%)
DS-Digits	ERM	11,674	0.48	0.00	59.98
DS-Digits	ERM+Replay	11,674	4.18	3.54	66.48
DS-Digits	MLDG+Replay	11,674	4.20	3.50	66.48
DS-Digits	**MetaDyG-Net**	12,469	8.81	5.47	67.09
DMS-24	ERM	5,644	0.52	0.00	46.69
DMS-24	ERM+Replay	5,644	3.95	3.25	61.28
DMS-24	MLDG+Replay	5,644	3.91	3.32	61.55
DMS-24	**MetaDyG-Net**	6,439	7.92	5.03	62.20

Note. *Times are empirical means from the controlled CPU environment and should not be treated as hardware-independent service-level guarantees.*

The controller adds 795 parameters, increasing DS-Digits model size from 11,674 to 12,469 and DMS-24 from 5,644 to 6,439. The total processing time is 8.81 ms per batch of 64-example DS-Digits and 7.92 ms for DMS-24. This is about twice the fixed-replay latency, but still a few thousand examples per second on the CPU test. In an edge budget constrained application, prototype construction can be updated less often, the optional head step can be turned off, or the memory can be quantified.

7.7 Protocol-aware comparison with published studies

Only the learning protocol made explicit makes cross-paper comparison informative. The differences between static domain generalization, future-domain extrapolation, unlabeled continual test-time adaptation and delayed-label prequential learning lie in the access to the target domain, the time of the update, the size of the backbone, the selection of the model, and the direction of the metric. Thus, the values reported by the original studies are retained in Table 13 and each method is compared to its own named baseline. It does not assume that raw accuracy and errors are measurements on a common scale.

Table 13. Protocol-aware comparison with representative published studies.

Study	Learning setting	Benchmark / protocol	Deployment feedback	Source-reported result	Cross-study status
SWAD [38]	Static domain generalization	DomainBed, five image datasets	No online target update	66.9% vs ERM 63.3% (+3.6 pp accuracy)	Static DG context; not directly comparable
Fishr [39]	Static domain generalization	DomainBed, seven datasets; test-domain model selection	No online update; target-validation selection	70.8% vs ERM 68.7% (+2.1 pp accuracy)	Selection/data differ; not directly comparable

Study	Learning setting	Benchmark / protocol	Deployment feedback	Source-reported result	Cross-study status
DRAIN [16]	Temporal domain generalization	Rot-MNIST future-domain prediction	No current/future target training batch	7.5 +/- 1.1% vs Offline 18.6 +/- 4.0% (-11.1 pp error)	Temporal extrapolation; no prequential update
CoTTA [19]	Unlabeled continual test-time adaptation	CIFAR10-C severity-5 corruption stream	Unlabeled target stream	16.2 +/- 0.1% vs Source 43.5% (-27.3 pp error); TENT-cont. 20.7%	Unlabeled CTTA; different data/backbone
Wild-Time [36]	Temporal-shift benchmark	Five real temporal datasets; Eval-Fix and Eval-Stream	Varies by evaluated method	Approximately 20% average ID-to-OOD performance drop	Benchmark-level finding, not one method score
MetaDyG-Net / DS-Digits	Delayed-label prequential adaptation	DS-Digits; six regimes; five seeds	Labels disclosed after each batch is scored	67.09 +/- 0.90%; +7.11 pp vs ERM; +0.61 pp vs MLDG+Replay	Direct only versus this manuscript's rerun baselines
MetaDyG-Net / DMS-24	Delayed-label prequential adaptation	DMS-24; six regimes; five seeds	Labels disclosed after each batch is scored	62.20 +/- 0.60%; +15.52 pp vs ERM; +0.65 pp vs MLDG+Replay	Direct only versus this manuscript's rerun baselines

Note. *Values for prior work are transcribed from the original publications [16,19,36,38,39]. Accuracy gains and error reductions are within-study contrasts. No pooled rank or significance test is computed because datasets, architectures, target information, model selection, and metrics differ.*

The static-DG results provide a convenient measure of source-only improvement. On five DomainBed image benchmarks, SWAD achieves an average out-of-domain accuracy of 66.9% compared to 63.3% for the ERM literature row, a 3.6-point improvement [38]. For seven DomainBed datasets, Fishr achieves 70.8% compared to ERM's 68.7%, an improvement of 2.1 points when choosing the model for test domain [39]. These numbers show progress on one-shot unseen-domain generalization, and neither of these settings has a changing target stream or online parameter update. Further, Fishr applies validation data from the test distribution to model selection which further restricts direct comparison with the source-only selection and no-leakage stream protocol used here.

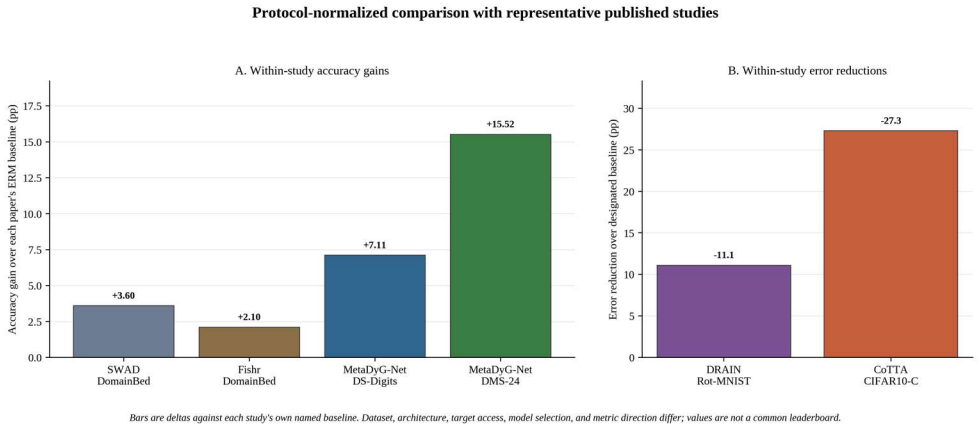

Figure 12. Within-study performance deltas for representative published studies and MetaDyG-Net. Panel A reports accuracy improvement over each paper's own ERM baseline; Panel B reports error reduction over each paper's designated source or offline baseline. The bars normalize direction but not dataset difficulty, target access, architecture, or model-selection protocol and therefore must not be interpreted as a cross-benchmark ranking.

The differences in the within-paper changes are due to different operational assumptions for temporal methods. DRAIN achieves 7.5 +/- 1.1% Rot-MNIST error, a 11.1-point drop from Offline training (18.6 +/- 4.0%). DRAIN extrapolates a model for a future domain but does not necessarily update on the current target batch. CoTTA achieves an error of 16.2 +/- 0.1% on CIFAR10-C, outperforming the frozen source model (43.5%) and TENT-continual (20.7%) with an unlabeled target stream, pseudo-labels averaged over the unlabeled target stream, augmentation, and stochastic restoration [19]. These are good adaptation results but for a label-free test-time problem and not for delayed supervised adaptation.

MetaDyG-Net improves over static ERM by 7.11 points on DS-Digits and 15.52 points on DMS-24. Part of that is due to labels that come after prediction. When compared to only strong fixed labeled-replay baselines, the residual gain is reduced to 0.61 points for MLDG+Replay on DS-Digits and 0.65 points on DMS-24. The scientifically defensible interpretation is not therefore that MetaDyG-Net numerically dominates SWAD, Fishr, DRAIN or CoTTA. Instead, it resides in a unique label-latency regime and provides meta-controlled plasticity, calibration, worst-regime analysis and auditable prequential updates to that regime.

The most explicit external-validity warning comes from Wild-Time, which, across five real temporally indexed datasets and thirteen previous approaches, reports an average drop of around 20% from in-distribution to out-of-distribution data, both with fixed-split and streaming evaluation [36]. This is an unfilled gap that encourages testing of MetaDyG-Net on true temporal datasets but cannot be used as an external performance claim until the proposed method is tested under the true protocols of Wild-Time.

8. Discussion

8.1 Scientific interpretation

The results distinguish between two effects that are often confused. First, online access to confirmed labels and replay yields the most significant improvement over the static domain generalization. Second, when this solid foundation is present, the benefits of meta-control and reliability are a smaller, but steady, layer. The proposed method should thus not be seen as proof of a controller being the solution to non-stationarity. It is proof of the ability of a controller to enhance the orchestration of a well-designed adaptive system.

The gain from calibration is more noticeable than the raw accuracy gain. Uniform tempering is easy, but its coefficient is related to stream uncertainty and acts post prototype fusion. This stage can decrease ECE without mechanically increasing the accuracy, since the uniform component does not change the class argmax. The lower Brier score also suggests better than binning behavior in the full distribution.

The behavior of the controller indicates that a trust-region design is suitable for small, controlled streams. There was no need for large learning-rate changes, and these changes could have increased the variance of the seed. In more complex and diverse settings, a more complex state, recurrent controller, or explicit change-point detector might yield more dynamic actions. Such extensions must be considered in light of the potential difficulty of verifying the complexity of the controller, relative to the predictor being controlled.

8.2 Intelligent systems engineering implications

Adaptation must be considered as a controlled subsystem for intelligent-systems deployment. The input summary, predictive entropy, controller actions, memory composition, update loss, gradient norm, and pre/post update validation signals should be logged in the production architecture. The learning-rate-scaler and replay-weight should be bound outside of the learned controller. When the drift state is outside of the range seen during meta-training, the system should regress to a safe action or not take any action at all, instead of extrapolating the controller's behavior.

There is a delay in labels, and it raises governance issues. Confirmed labels should go to memory, disputed/corrected/weak labels require provenance and revocation. Rare examples may be retained in the reservoir memory, but this can conflict with retention policies, and may need class-aware quotas, encrypted storage, or feature-

only memory in operational systems. Concept drift can also be a valid policy change and not noise; so adaptation should be versioned and reversible.

Calibration can be used to inform engineering decisions like selective prediction, human review, and active labelling. ECE is not enough for certification as it is dependent on data sets and bins. Class-conditional reliability, cost-weighted calibration, abstention curves and performance under label delay or missing should be monitored in a deployed system. The framework provided is indicative of probability and uncertainty and is not a substitute for application specific safety analysis.

8.3 Relationship to current DG and CTTA practice

In the comparison of Section 7.7 four operational modes are distinguished. SWAD and Fishr are static DG, in which the target is not observed during training and the predictor deployed is not updated continuously [38,39]. DRAIN is temporal extrapolation, that is, a series of historical domains are used to develop a model for a future domain [16]. CoTTA is a form of unlabeled CTTA, where updates are based on predictive consistency and restoration [19] and target batches are provided. MetaDyG-Net deals with delayed-label prequential adaptation, that is, a prediction is made before the label is available.

These modes shouldn't be combined into one state-of-the-art table. Target labels are a resource: they are useful for more effective real concept drift correction, but have latency, annotation cost, governance, and label-quality risk. This distinction is quantified by the large gain of the large MetaDyG-Net over static ERM and the much smaller gain over fixed replay. The contribution is the learned coordination of supervised adaptation and reliability mechanisms after the arrival of the label, not that delayed supervision is equal to source-only DG or unlabeled CTTA.

In practice, hybrid systems can switch between these modes based on the latency of the labels and risk. Static DG still makes sense if updates are not allowed; unlabeled CTTA is helpful if labels aren't available yet or are never going to be available; delayed supervised updates can be accomplished once confirmed. Label availability, delay length and label confidence could be added to the state of a future controller and it could select between taking frozen, self-supervised, pseudo-labeled, and supervised actions.

9. Limitations and Threats to Validity

The assessment has seven key constraints. Firstly, only two compact benchmarks are employed. DS-Digits is real image content but has artificial transformations; DMS-24 is fully controlled. Large scale visual DG datasets, clinical streams, industrial sensors and language-model distribution shifts are yet to be tested. Secondly, the study assumes that labels are available after one prediction cycle. Extended, non-uniform, noisy or censored delays may have a significant impact on memory quality and adaptation stability.

Third, the benchmark is based on multilayer perceptrons. The controller interface is architecture agnostic, while the geometry of the drift state and the latency may vary depending on the backbones used (convolutional, recurrent, transformer, graph-neural, multimodal). Fifth, five seeds are sufficient to reveal variation and allow for paired tests and still provide coarse non-parametric inference. The DS-Digits comparison to MLDG+Replay is just shy of the two-sided corrected threshold, which should be determined with more seeds and datasets.

Fifth, hyperparameters are not optimized in a large grid but are set based on pilot reasoning. This helps to prevent overfitting the target but is not optimal. Sixth, there is a slight difference in the actions taken by the controller. Although the ablation shows benefit, it cannot be proven that the learned policy will be useful when the change is not within the controlled family. To prevent high-stakes use, out-of-range state detection, formal safe-update constraints, and post-deployment rollback are required.

Seventh, Table 13 is a protocol-aware narrative synthesis and not a meta-analysis. It copies representative findings from original research but can't eliminate variations in data sets, architectures, target supervision, validation access, stream ordering, and metric definitions. The within-study deltas in Figure 12 are meant to avoid sign confusion, not to be a pooled ranking or to be statistically superior to external methods.

Prequential scoring, explicit no-leakage ordering, raw seed-level results, multiple reliability metrics and component ablations provide further evidence of construct validity. The framework has limited external validity until it is evaluated on a real temporally indexed domain, like Wild-Time [36] or on an operational sensor fleet or institution-shifted clinical data. The paper is thus a study of the mechanism and a research draft for reproduction, not a statement of state of the art.

10. Conclusion

In this study, a new deep neural framework called MetaDyG-Net for cross-domain generalization in dynamic and non-stationary environments was introduced, which is enhanced by meta-learning. The approach learns a transferable initialization and an interpretable controller for layer-wise plasticity and replay. It is a combination of bounded reservoir memory, Fisher anchoring, class prototypes, selective head adaptation, and uncertainty-aware calibration that is applied during deployment under a strict test-then-train protocol.

MetaDyG-Net had the highest mean accuracy, calibration and worst-regime performance on both controlled streams across the five seeds. Confirmed-label replay provided the greatest gains over strong replay baselines, while the meta-controller and auxiliary mechanisms provided smaller cumulative gains. This finding is scientifically valuable as it allows to distinguish between the main and the secondary sources of performance and not only the contribution of meta-learning.

The protocol-aware comparison with published studies confirms that dynamic generalization is not one benchmark category: static DG, temporal extrapolation, unlabeled CTTA and delayed-label adaptation present models with different information. The takeaway is that dynamic generalization should be designed as a controlled adaptive system, rather than a single loss function. In the future, the framework should be tested on large real temporal benchmarks, incorporate uncertain label delay, and be extended to graph, transformer, and multimodal neu

References

[1] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), Article 44.

[2] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346-2363.

[3] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. Machine Learning, 79, 151-175.

[4] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. (2016). Domain-adversarial training of neural networks. Journal of Machine Learning Research, 17(59), 1-35.

[5] Sun, B., & Saenko, K. (2016). Deep CORAL: Correlation alignment for deep domain adaptation. In ECCV Workshops (pp. 443-450).

[6] Gulrajani, I., & Lopez-Paz, D. (2021). In search of lost domain generalization. International Conference on Learning Representations.

[7] Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., & Yu, P. S. (2023). Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering, 35(8), 8052-8072.

[8] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of Machine Learning Research, 70, 1126-1135.

[9] Li, D., Yang, Y., Song, Y.-Z., & Hospedales, T. M. (2018). Learning to generalize: Meta-learning for domain generalization. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1), 3490-3497.

[10] Balaji, Y., Sankaranarayanan, S., & Chellappa, R. (2018). MetaReg: Towards domain generalization using meta-regularization. Advances in Neural Information Processing Systems, 31, 998-1008.

[11] Li, Y., Yang, Y., Zhou, W., & Hospedales, T. (2019). Feature-critic networks for heterogeneous domain generalization. Proceedings of Machine Learning Research, 97, 3915-3924.

[12] Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., & Tesauro, G. (2019). Learning to learn without forgetting by maximizing transfer and minimizing interference. International Conference on Learning Representations.

[13] Javed, K., & White, M. (2019). Meta-learning representations for continual learning. Advances in Neural Information Processing Systems, 32.

[14] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13), 3521-3526.

[15] Vitter, J. S. (1985). Random sampling with a reservoir. ACM Transactions on Mathematical Software, 11(1), 37-57.

[16] Bai, G., Ling, C., & Zhao, L. (2023). Temporal domain generalization with drift-aware dynamic neural networks. International Conference on Learning Representations.

[17] Pham, T.-H., Zhang, X., & Zhang, P. (2024). Non-stationary domain generalization: Theory and algorithm. arXiv:2405.06816.

[18] Wang, D., Shelhamer, E., Liu, S., Olshausen, B., & Darrell, T. (2021). Tent: Fully test-time adaptation by entropy minimization. International Conference on Learning Representations.

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

[20] Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., & Tan, M. (2022). Efficient test-time model adaptation without forgetting. Proceedings of Machine Learning Research, 162.

[21] Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., & Tan, M. (2023). Towards stable test-time adaptation in dynamic wild world. International Conference on Learning Representations.

[22] Yuan, L., Xie, B., & Li, S. (2023). Robust test-time adaptation in dynamic scenarios. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15922-15932.

[23] Yang, X., Chen, X., Li, M., Wei, K., & Deng, C. (2024). A versatile framework for continual test-time domain adaptation: Balancing discriminability and generalizability. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[24] Liu, J., et al. (2024). Continual-MAE: Adaptive distribution masked autoencoders for continual test-time adaptation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 28653-28663.

[25] Park, H., Park, H., Ko, J., & Min, D. (2025). Hybrid-TTA: Continual test-time adaptation via dynamic domain shift detection. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2877-2886.

[26] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. Machine Learning, 106, 1469-1495.

[27] Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. Journal of Machine Learning Research, 11, 1601-1604.

[28] Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2021). River: Machine learning for streaming data in Python. Journal of Machine Learning Research, 22(110), 1-8.

[29] Sahoo, D., Pham, Q., Lu, J., & Hoi, S. C. H. (2018). Online deep learning: Learning deep neural networks on the fly. Proceedings of the International Joint Conference on Artificial Intelligence, 2660-2666.

[30] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. Proceedings of Machine Learning Research, 70, 1321-1330.

[31] Gretton, A., Borgwardt, K. M., Rasch, M. J., Scholkopf, B., & Smola, A. (2012). A kernel two-sample test. Journal of Machine Learning Research, 13, 723-773.

[32] Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7, 1-30.

[33] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[34] Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32, 8024-8035.

[35] Koh, P. W., Sagawa, S., Marklund, H., et al. (2021). WILDS: A benchmark of in-the-wild distribution shifts. Proceedings of Machine Learning Research, 139, 5637-5664.

[36] Yao, H., Choi, C., Lee, B., et al. (2022). Wild-Time: A benchmark of in-the-wild distribution shift over time. Advances in Neural Information Processing Systems, 35.

[37] Zhou, K., Yang, Y., Qiao, Y., & Xiang, T. (2021). Domain generalization with MixStyle. International Conference on Learning Representations.

[38] 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.

[39] Rame, A., Dancette, C., & Cord, M. (2022). Fishr: Invariant gradient variances for out-of-distribution generalization. Proceedings of Machine Learning Research, 162.

[40] Arjovsky, M., Bottou, L., Gulrajani, I., & Lopez-Paz, D. (2019). Invariant risk minimization. arXiv:1907.02893.

[41] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., & Tuytelaars, T. (2022). A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(7), 3366-3385.

[42] Hoi, S. C. H., Sahoo, D., Lu, J., & Zhao, P. (2021). Online learning: A comprehensive survey. Neurocomputing, 459, 249-289.

[43] Zhou, K., Liu, Z., Qiao, Y., Xiang, T., & Loy, C. C. (2023). Domain generalization: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(4), 4396-4415.

Appendix A. Reproducibility Checklist

Table A1. Reproducibility and reporting checklist.

Item	Implementation
Data partition	Stratified DS-Digits base split fixed at random_state=2026; target base images are disjoint from source base images.
Randomness	Five model/stream seeds; Python, NumPy, and PyTorch seeds set for each run.
Evaluation order	Predict current batch, record probabilities and labels, then update; no current-batch label leakage.
Memory	Reservoir capacity 384, source initialization 192, confirmed labels only.
Metrics	Accuracy, macro-F1, 15-bin ECE, multiclass Brier, worst regime, recovery, latency.
Statistics	Paired t-test, Cohen dz, 95% CI, one-sided Wilcoxon direction check, Holm correction within benchmark.
Hardware	CPU-only controlled run; PyTorch 2.10.
Artifacts	Experiment script, analysis script, raw seed metrics, trajectories, tables, and figures included.
Known limitation	Controlled compact benchmarks; no claim of universal state of the art.

Appendix B. Benchmark Regime Definitions

Table B1. DS-Digits target regimes.

ID	Name	Definition
S1	Unseen rotation	Approximately +18 degree rotation with noise and mild translation.
S2	Gradual rotation/contrast	Continuous transition from +18 to -22 degrees with falling contrast and increasing blur.

ID	Name	Definition
S3	Blur-noise	-22 degree rotation, strong blur, higher Gaussian noise, low contrast.
S4	Abrupt occlusion	Abrupt +30 degree regime with random occlusion, gamma change, and translation.
S5	Recurrent rotation	Return to a stochastic version of the initial +18 degree regime.
S6	Compound low-contrast	-31 degree rotation, high noise, blur, occlusion, low contrast, and translation.

Table B2. DMS-24 target regimes.

ID	Name	Definition
S1	Unseen sensor	Unseen shared-coordinate sensor mixing, bias, and heteroskedastic noise.
S2	Gradual sensor drift	Continuous interpolation between two sensor mixing and bias regimes.
S3	Abrupt concept drift	Changed class decision weights under the second sensor regime.
S4	Recurrent sensor	Return to the first sensor mixing while retaining the changed concept.
S5	Class-prior drift	Class-logit shift under recurrent sensor and changed concept.
S6	Compound sensor/concept	More distant sensor mixing, changed concept, prior shift, and stronger noise.

Appendix C. Practical Extension Paths

The proposed state/action interface is deliberately small and architecture-independent. The central extension principle is to expose domain-relevant, label-free state variables to a bounded controller and to restrict its actions to auditable adaptation modules. The same prequential no-leakage evaluation should then be retained.

Table C1. Example architecture-specific controller extensions.

Application	Backbone	Additional state variables	Bounded actions
Dynamic graph streams	Message-passing GNN	Degree distribution, homophily, edge churn, spectral summaries	GNN-layer rates, edge encoder, replay by subgraph
Transformer streams	Transformer or ViT	Token entropy, embedding moments, attention drift	Adapter/LoRA rates, layer-norm updates, head rate
Multimodal systems	Modality encoders + fusion	Per-modality uncertainty, missingness, cross-modal agreement	Modality-specific rates, fusion gate, replay allocation

Application	Backbone	Additional state variables	Bounded actions
Industrial time series	TCN, RNN, or state-space model	Autocorrelation, spectral energy, residual shift, alarm density	Temporal-block rates, memory horizon, prototype weight
Federated domains	Client-local encoder	Client divergence, participation gap, gradient conflict	Client adaptation, proximal strength, replay quota

These extensions require new empirical validation. In particular, graph and federated streams introduce topology or client-selection drift that is not represented in the present benchmarks. The table specifies engineering hypotheses, not evaluated results.