AutoBudget-SMAT: Primal–Dual Meta-Tuning with Learned Per-Module Sparsity/Latency Budgets

Liz Lemma Future Detective

January 20, 2026

Abstract

Sparse MetA-Tuning (SMAT) improves few-shot generalization by constructing task-specific models as sparse interpolations between a frozen pretrained backbone and a meta-learned delta, with expert routing conditioned on few-shot support data. However, SMAT exposes a sensitive global sparsity hyperparameter that governs the in-distribution (ID) vs out-of-distribution (OOD) trade-off and is poorly aligned with real deployment constraints such as latency or energy. We propose AutoBudget-SMAT: a constrained meta-tuning formulation that learns per-module capacity budgets (or per-module sparsity targets) jointly with routing, using primal–dual updates. Our method replaces a single global sparsity target with a vector of learned budgets over transformer components (e.g., attention heads, MLP blocks, or projection matrices) and supports deployment at multiple latency points without retraining by solving a small routing+budget subproblem at test time. We provide (i) a kernel-aware differentiable latency proxy and validation protocol linking proxy to wall-clock, (ii) a primal–dual meta-training algorithm that enforces feasibility via per-module constraints and multiplier resets, and (iii) a theory of convergence and Pareto-optimality in a convex surrogate regime together with hardness results for discrete budget allocation. Experiments (to be included) should demonstrate domination of the ID/OOD Pareto frontier versus fixed-τ SMAT and structured pruning baselines across heavy OOD suites and multiple hardware profiles.

Table of Contents

grangians; why per-module allocation is suggested by observed mask patterns.

3. 3. Problem Setup and Formal Objectives: episodic meta-learning notation; define constrained robust meta-objective with latency proxy; define per-module budgets and deployment profiles; state optimization targets.

4. 4. AutoBudget-SMAT Method: parameterization of per-module budgets; structured masks vs unstructured; routing conditioned on (support set, budget); training objective with primal–dual updates and feasibility enforcement.

5. 5. Latency Proxy and Systems Model: define kernel-aware proxy; assumptions and calibration procedure; proxy-to-wall-clock correlation bounds; discussion of distortions and limitations.

6. 6. Theory (Convex Surrogate Regime): (i) convergence to approximate KKT, (ii) Pareto frontier characterization via Lagrange multipliers, (iii) separation theorem: global-τ suboptimality vs per-module budgets, (iv) oracle lower bounds for first-order methods under constraints.

7. 7. Hardness of Discrete Budgeting and Necessity of Relaxations: reduction from knapsack/subset selection; implications for why convex relaxations or stochastic masks are required.

8. 8. Experiments (Implementation-Driven, Strengthening the Contribution): datasets (Meta-Dataset + extra OOD), backbones (ViT/LLM adapters), hardware targets; compare against SMAT fixed-τ, structured pruning, slimmable baselines; evaluate Pareto frontiers and deployment transfer.

9. 9. Ablations and Diagnostics: per-layer budget learning vs fixed; proxy mismatch sensitivity; robustness to routing noise; effect of structured vs unstructured sparsity; calibration across devices.

10. 10. Discussion and Future Work: multi-source priors, uncertainty-aware routing, stronger DRO objectives, and tighter latency models.

1 Introduction

We consider the meta-tuning of a large pretrained model under two simultaneous sources of mismatch: distribution shift across tasks and deployment constraints that are external to the learning objective. In the few-shot regime, the standard meta-learning abstraction posits episodes $T \sim P_{\mathrm{ID}}$ with support–query splits, and a meta-learner that extracts from T^s a task-specific predictor evaluated on T^q. In modern practice the base model is a pretrained transformer whose parameters are too large to adapt freely per episode, and thus task adaptation is implemented through low-dimensional or sparse *deltas* applied to frozen pretrained parameters. The central difficulty, and the point of departure for this work, is that the choice of *how much* adaptation to permit is inseparable from two realities: (i) at evaluation time the task distribution is not guaranteed to match P_{ID} (we may face an unknown OOD mixture), and (ii) the deployed model must obey explicit latency/energy constraints on heterogeneous devices.

A common approach in sparse delta tuning is to introduce a single global sparsity level τ controlling the overall nonzero fraction of the adaptation parameters. Such a knob is appealing because it is scalar and easily swept. However, we argue that global sparsity is the wrong control variable for budgeted deployment. First, device cost is not uniform across parameter locations: different modules (attention projections, MLP blocks, heads, embeddings) induce different runtime effects on real hardware, and the mapping from *parameter count* to *latency* is strongly module-dependent. Second, the predictive utility of adaptation is also nonuniform: some modules are "high-leverage" for certain tasks, whereas others contribute little, and this nonuniformity is amplified under distribution shift. Consequently, enforcing the same effective sparsity pressure across all modules implicitly assumes a uniform cost–benefit structure that is absent in practice. The result is a misallocation phenomenon: global-τ budgets may over-constrain modules that matter for accuracy on hard or OOD episodes while under-constraining modules whose activation is cheap but unhelpful (or vice versa), yielding Pareto-inefficient accuracy–latency trade-offs.

For deployment, the natural primitive is not a global sparsity target but an explicit budget B (latency or energy), possibly varying at test time and across device profiles. Treating B as first-class suggests the following reformulation: rather than selecting a single τ during training and hoping it transfers, we aim to learn a *policy* that, given a support set T^s and a budget descriptor, constructs a task model whose predicted cost obeys $\widehat{\mathrm{Lat}}(\theta) \le B$ and whose query risk is low. In this view, sparsity is an internal mechanism used to satisfy the external constraint B; what must generalize is the mapping $(T^s, B) \mapsto \theta$, not a globally fixed sparsity rate.

This perspective is further sharpened by the observation that latency constraints are inherently modular. Let $\ell \in [L]$ index a partition of the

model into modules, each with a device-dependent cost model $c_\ell(\mathcal{D})$. Even when latency is only approximately additive, per-module budgeting remains a faithful abstraction because it allows the learning procedure to allocate capacity to where it most improves task loss subject to the true constraint of interest. Formally, we regard per-module capacities b_ℓ as decision variables, and we enforce constraints of the form

$$\mathbb{E}\big[\mathrm{nnz}_\ell(z)/d_\ell\big] \le b_\ell, \qquad \widehat{\mathrm{Lat}}(\theta) \le B,$$

while maintaining task-dependent routing coefficients on the simplex. The key point is that $\mathbf{b} = (b_\ell)_{\ell\in[L]}$ is not merely a set of hyperparameters but a learned object, potentially a function of B, that captures how the meta-learner should spend its adaptation budget across modules.

The contributions of this work are organized around this budget-first formulation. *(i) AutoBudget meta-tuning.* We introduce a primal–dual meta-optimization scheme that jointly learns (a) a shared dense delta Δ and expert masks $\{z_m\}_{m=1}^M$ supporting sparse interpolated experts, (b) a routing rule producing task-dependent mixing weights α from (T^s, B), and (c) per-module capacity budgets $\mathbf{b}(B)$ that mediate structured sparsity. The training objective augments the usual expected query loss by Lagrangian terms for per-module capacity violations and for a differentiable latency proxy, thereby treating feasibility as a persistent training-time constraint rather than an after-the-fact pruning step.

(ii) Budgeted test-time adaptation. At meta-test, we do not fix a sparsity level; instead, given (T^s, B) we solve a small routing+budget subproblem that optimizes support loss plus dual penalties subject to $\widehat{\mathrm{Lat}} \le B$. This yields a task model that is feasible by construction with respect to the proxy, and (under standard proxy distortion assumptions) feasible with high probability with respect to true measured latency up to multiplicative and additive tolerances.

(iii) Convex-surrogate theory and Pareto structure. In a convex surrogate regime (linearized models with convex losses and convex constraints), we show that stochastic primal–dual updates converge to an approximate KKT point in $\tilde{O}(1/\varepsilon^2)$ first-order iterations. Moreover, we characterize how varying the latency multiplier traces supported Pareto-optimal points of the risk–latency trade-off, clarifying the relation between Lagrange multipliers, deployable budgets, and the attainable frontier.

(iv) A provable limitation of global sparsity. We provide a separation result: there exist task families and module cost structures for which any global sparsity constraint τ yields a strictly suboptimal constrained risk compared with per-module budgeting under the same total latency. This formalizes the earlier intuition that a single global knob cannot, in general, express the optimal allocation of adaptation capacity across heterogeneous modules.

(v) Algorithmic necessity of relaxation. Finally, we note that exact discrete module selection is computationally hard even for simple additive cost

models and linearized benefits, motivating our reliance on convex relaxations, stochastic masks, and primal–dual training rather than combinatorial search.

The remainder of the paper develops the necessary background, beginning with SMAT-style sparse interpolated experts and their typical global-sparsity instantiations, and then explains how Lagrangian constrained learning and observed module-wise mask patterns naturally lead to per-module budget allocation.

2 Background: SMAT-style sparse interpolated experts and constrained sparsity

We recall the sparse interpolated expert parameterization used in SMAT-style meta-tuning, and we emphasize how sparsity serves simultaneously as (a) an adaptation-capacity control relevant to ID/OOD behavior and (b) a mechanism for satisfying external deployment budgets. The essential construction begins from a frozen pretrained backbone with parameters θ^{pre} and introduces a shared dense *delta* $\Delta \in \mathbb{R}^d$ together with M expert-specific masks $\{z_m\}_{m=1}^M$, where each $z_m \in \{0,1\}^d$ (or a continuous relaxation thereof). Given a task (episode) T_i with support set T_i^s, a routing rule produces mixing weights $\alpha_i \in \Delta^{M-1}$ on the simplex, and the task-specific parameters are formed as

$$\theta_i \;=\; \theta^{\text{pre}} \;+\; \sum_{m=1}^{M} \alpha_{i,m}\,(z_m \odot \Delta).$$

This representation has two consequences that are useful in the few-shot setting. First, it shares statistical strength across tasks through the common Δ while permitting task variation through both the routing weights and the mask structure. Second, it yields a natural continuum between "no adaptation" and "full adaptation" by moving along two axes: interpolating among experts via α_i and varying the active fraction of parameters via the masks.

The role of $M > 1$ experts is not merely to provide a mixture-of-experts ensemble; rather, it creates a discrete (or piecewise) set of sparse subnetworks within the same delta space, so that the meta-learner can represent qualitatively different update patterns while maintaining a fixed base model. In typical implementations, the routing rule depends on a task embedding computed from T_i^s (for instance by pooling representations under $f_{\theta^{\text{pre}}}$) and then applying a small controller network whose output is normalized by a softmax to enforce $\alpha_i \geq 0$ and $\sum_m \alpha_{i,m} = 1$. The masks may be treated as Bernoulli variables with learnable logits, or replaced by a differentiable stochastic gate (e.g. hard-concrete) so that the expected sparsity can be optimized by gradient methods while retaining the option to sample discrete masks for deployment.

Sparsity enters as a capacity control, and thus influences the ID/OOD trade-off in a manner analogous to classical regularization. When meta-training tasks are drawn from P_{ID}, increasing the effective degrees of freedom of the adaptation (by allowing a larger nonzero fraction in $z_m \odot \Delta$, or by spreading α_i across multiple experts) can reduce the empirical meta-training query loss. However, under distribution shift—here modeled as evaluation on an unknown mixture P_{OOD}—additional adaptation flexibility can amplify reliance on spurious task-specific correlations present in P_{ID} but absent in P_{OOD}. Conversely, enforcing higher sparsity can improve stability by constraining updates to a smaller subspace, but this may underfit genuinely hard tasks or tasks whose Bayes-optimal predictor differs substantially from θ^{pre}. Thus, sparsity is not an incidental compression device; it is a principal knob governing the effective hypothesis class of per-task predictors, and it directly shapes the attainable risk trade-offs across ID and OOD regimes.

A widespread baseline is to impose a single global sparsity target $\tau \in [0, 1]$ (equivalently a single nonzero budget), typically enforced in expectation across all parameters of the delta. Writing $\|z_m\|_0$ for the number of active coordinates in z_m, one may constrain, for instance,

$$\frac{1}{d}\,\mathbb{E}\big[\|z_m\|_0\big] \;\le\; 1 - \tau, \qquad \text{for each } m \in [M],$$

or an aggregated variant coupling the experts. The practical appeal is evident: a single scalar can be swept to obtain an empirical accuracy–cost curve. From an optimization viewpoint, this constraint can be incorporated via a Lagrangian term, leading to objectives of the schematic form

$$\min_{\Delta,\{\text{mask params}\},\zeta} \; \mathbb{E}_{T\sim P_{\mathrm{ID}}}\big[\mathcal{L}_T(\theta(T))\big] \;+\; \lambda\Big(\tfrac{1}{d}\mathbb{E}[\|z\|_0] - (1-\tau)\Big),$$

where ζ denotes routing parameters and $\lambda \ge 0$ is a multiplier. Even in the absence of explicit deployment budgets, such a Lagrangian view is useful: it clarifies that sparsity constraints are not external post-processing but can be treated as first-class constraints during meta-optimization, with λ adapting to enforce approximate feasibility.

When we introduce explicit device constraints, Lagrangian constrained learning becomes unavoidable. A latency proxy $\widehat{\mathrm{Lat}}(\theta)$ can be differentiated through and compared to a budget. The natural constrained formulation is therefore not "choose τ" but "choose θ so that $\widehat{\mathrm{Lat}}(\theta) \le B$". In this setting, a global sparsity constraint is at best an indirect surrogate: it enforces a parameter-count budget that is only loosely coupled to measured latency, and the coupling can vary across device profiles. The Lagrangian perspective makes the mismatch concrete: even if a global τ is enforced tightly, nothing guarantees that the resulting $\widehat{\mathrm{Lat}}(\theta)$ satisfies a particular budget B, nor that it does so uniformly across tasks whose routing weights induce different patterns of computation.

This observation leads to the central structural limitation of global sparsity: it conflates heterogeneous modules into a single pool. Let $\ell \in [L]$ index a partition of the backbone (e.g. attention projections, MLP blocks, heads), and write $\mathrm{nnz}_\ell(z)$ for the number of active delta coordinates that fall in module ℓ. Empirically and algorithmically, masks learned under SMAT are not uniform across ℓ. Instead, we often observe concentration phenomena: certain modules consistently receive higher mask density because they provide larger loss reductions per active parameter, whereas others remain sparse across tasks. Moreover, this pattern is task-dependent: routing can shift mass toward experts whose masks emphasize particular modules, reflecting that distinct tasks benefit from distinct adaptation loci. Such nonuniformity is not an artifact; it is a learned representation of where adaptation is most effective.

Once one acknowledges this, it becomes natural to replace the single scalar τ by a vector of per-module capacity budgets $\mathbf{b} = (b_\ell)_{\ell \in [L]}$ and to enforce constraints of the type

$$\mathbb{E}\big[\mathrm{nnz}_\ell(z)/d_\ell\big] \;\le\; b_\ell, \qquad \ell \in [L],$$

with a corresponding collection of multipliers $\{\lambda_\ell\}_{\ell \in [L]}$. This modification is suggested simultaneously by (i) the observed mask patterns (which already behave as if they were allocating capacity across modules) and (ii) the operational semantics of latency (which depends on module-wise computational structure rather than on a uniform parameter count). In other words, per-module budgeting does not introduce a qualitatively new control; it makes explicit and optimizable a structure that the learned masks already exhibit implicitly, while aligning the constraint interface with the modularity of runtime cost.

Finally, we stress a methodological point that will underlie the formal objectives later: once constraints are expressed at the appropriate granularity, primal–dual training becomes the natural meta-optimization paradigm. The primal variables govern prediction (the delta, masks, routing), while the dual variables encode the shadow prices of violating capacity and latency constraints. In regimes where relaxations render the constraints differentiable and (approximately) convex, multiplier updates provide a principled mechanism to steer training toward feasible predictors without hard thresholding during learning. This framing also anticipates the test-time procedure: rather than fixing sparsity *a priori*, one solves for a configuration that trades support-set fit against the learned constraint prices, yielding a task model that respects the prescribed budget.

3 Problem setup and formal objectives: episodic tasks, budgets, and robust constrained risk

We work in the standard episodic meta-learning setting. A task (episode) T consists of a labeled support set $T^s = \{(x_j, y_j)\}_{j=1}^{n_s}$ and a labeled query set $T^q = \{(x_j, y_j)\}_{j=1}^{n_q}$. Meta-training tasks are sampled i.i.d. as $T_i \sim P_{\mathrm{ID}}$, while meta-testing may draw tasks from P_{ID} and also from an unknown shifted distribution P_{OOD} (or a mixture thereof). For each episode, a task-adapted parameter vector $\theta(T)$ is constructed from the support set and then evaluated on the query set via a query loss $\mathcal{L}_T(\theta(T))$ (e.g. cross-entropy), optionally augmented by regularization or distillation terms. Throughout, θ^{pre} is fixed, and the adaptation degrees of freedom are those induced by sparse deltas as described in the background.

Deployment imposes explicit compute constraints. We therefore introduce a set of device profiles $\{\mathcal{D}_r\}$ (or, equivalently, a parametric cost model) and a scalar budget descriptor B provided at test time. The budget can encode latency, energy, or a composite cost; for concreteness we write "latency." Given a device profile $\mathcal{D}$ and parameters θ, let $\mathrm{Lat}(\theta; \mathcal{D})$ denote the true measured latency. Since true measurements are not differentiable and are expensive to query during training, we assume access to a differentiable proxy $\widehat{\mathrm{Lat}}(\theta; \mathcal{D})$ satisfying a high-probability distortion bound of the form

$$\mathrm{Lat}(\theta; \mathcal{D}) \;\le\; \kappa\, \widehat{\mathrm{Lat}}(\theta; \mathcal{D}) + \epsilon_{\mathrm{lat}}, \qquad \kappa \ge 1,\ \epsilon_{\mathrm{lat}} \ge 0, \tag{1}$$

uniformly over the family of candidate adapted models considered by the algorithm. A sufficient condition for device-feasibility is then

$$\widehat{\mathrm{Lat}}(\theta; \mathcal{D}) \;\le\; \frac{B - \epsilon_{\mathrm{lat}}}{\kappa}, \tag{2}$$

whenever $B > \epsilon_{\mathrm{lat}}$, which motivates treating $\widehat{\mathrm{Lat}}$ as the constraint during learning and test-time optimization.

To align the constraint interface with the architecture, we partition the delta parameters into L modules (blocks/components) indexed by $\ell \in [L]$, and we denote by d_ℓ the number of delta coordinates in module ℓ so that $\sum_{\ell=1}^{L} d_\ell = d$. For a given mask z (or a stochastic relaxation thereof), we write $\mathrm{nnz}_\ell(z)$ for the number of active coordinates within module ℓ. We associate each module with a device-dependent marginal cost $c_\ell(\mathcal{D})$, which may be a learned coefficient in an additive proxy model or an empirically fitted statistic. This allows us to express *capacity* constraints at module granularity through per-module budgets $\mathbf{b} = (b_\ell)_{\ell \in [L]}$, where $b_\ell \in [0, 1]$ specifies the allowed active fraction of delta coordinates (or structured activations) in module ℓ. The canonical constraint we enforce is

$$\mathbb{E}\left[\frac{\mathrm{nnz}_\ell(z)}{d_\ell}\right] \;\le\; b_\ell, \qquad \ell \in [L], \tag{3}$$

where the expectation is taken over any internal mask randomness (e.g. Bernoulli or hard-concrete sampling). The role of (3) is twofold: it provides a direct handle on adaptation capacity (hence on generalization under shift) and it supplies a structured proxy for cost that is substantially more informative than a global scalar sparsity.

We now formalize the meta-objective as a *robust constrained* risk minimization. Let ψ denote all meta-parameters (shared delta, mask parameters, and routing/controller parameters), and let $\theta_\psi(T^s, B, \mathcal{D})$ be the task model constructed from support set T^s and test-time budget B (and possibly profile $\mathcal{D}$). A basic constrained ID objective is

$$\min_{\psi} \ \mathbb{E}_{T\sim P_{\mathrm{ID}}}[\mathcal{L}_T(\theta_\psi(T^s, B, \mathcal{D}))] \quad \text{s.t.} \quad \widehat{\mathrm{Lat}}(\theta_\psi(T^s, B, \mathcal{D}); \mathcal{D}) \le B, \ \ (3) \text{ holds}, \tag{4}$$

together with the routing simplex constraints $\alpha(T^s, B) \in \Delta^{M-1}$ whenever a mixture over experts is used. However, since meta-testing may involve a shifted task distribution, we replace the plain expectation over P_{ID} by a robust criterion. Two instantiations that fit our analysis are:

$$\min_{\psi} \ \sup_{Q\in\mathcal{U}(P_{\mathrm{ID}})} \mathbb{E}_{T\sim Q}[\mathcal{L}_T(\theta_\psi(T^s, B, \mathcal{D}))] \quad \text{s.t. feasibility constraints as in (4)}, \tag{5}$$

$$\min_{\psi} \ \mathrm{CVaR}_\beta(\mathcal{L}_T(\theta_\psi(T^s, B, \mathcal{D}))) \quad \text{s.t. feasibility constraints as in (4)}, \tag{6}$$

where $\mathcal{U}(P_{\mathrm{ID}})$ is an uncertainty set intended to cover plausible OOD mixtures and $\beta \in (0, 1]$ tunes the tail emphasis. Both formulations capture the principle that adaptation should be chosen not merely to optimize average ID query loss, but to control worst-case (or high-quantile) query loss under a constrained compute envelope.

Finally, we make explicit the test-time interface implied by (4)–(6). At deployment, we observe $(T^s, B, \mathcal{D})$ and must produce an adapted model that is feasible for the given budget. Since the learned meta-parameters specify a family of feasible configurations rather than a single fixed sparsity, we view meta-testing as solving the episode-conditioned constrained subproblem

$$\theta^\star(T^s, B, \mathcal{D}) \ \in \ \arg\min_{\theta\in\Theta(\psi; T^s, B, \mathcal{D})} \ \mathcal{L}_{T^s}(\theta) \quad \text{s.t.} \quad \widehat{\mathrm{Lat}}(\theta; \mathcal{D}) \le B, \ \ (3) \text{ holds}, \tag{7}$$

where $\Theta(\psi; T^s, B, \mathcal{D})$ denotes the structured family of models realizable by the sparse-interpolated parameterization under meta-parameters ψ (e.g. varying routing weights and relaxed mask activations). In the convex surrogate regime, the dependence of both loss and proxy constraints on the gating variables is convex by assumption, so (7) admits efficient approximate solutions and admits a standard Lagrangian/KKT characterization. Our algorithmic goal in the sequel is to learn ψ so that this subproblem is

(approximately) feasible for a range of budgets and device profiles, while achieving low robust meta-risk; and, in the convex setting, to do so with iteration complexity consistent with obtaining an ε-KKT point.

4 AutoBudget-SMAT: per-module budget learning, structured experts, and primal–dual meta-optimization

We now specify the parameterization and optimization scheme that instantiates the constrained meta-objective with *learned* per-module capacity budgets. The starting point is the SMAT-style family of task models

$$\theta_i \;=\; \theta^{\mathrm{pre}} \;+\; \sum_{m=1}^{M} \alpha_{i,m}\,\big(z_m \odot \Delta\big), \qquad \alpha_i \in \Delta^{M-1}, \tag{8}$$

where Δ is a shared dense delta, $z_m \in \{0,1\}^d$ (or a relaxation) is the mask for expert m, and α_i are episode-specific mixing weights computed from the support set and a budget descriptor. The AutoBudget-SMAT distinction is that (i) capacity is controlled *per module* via $\mathbf{b} = (b_\ell)_{\ell\in[L]}$ rather than through a single global sparsity τ, and (ii) $\mathbf{b}$ is itself predicted (or optimized) as a function of the test-time budget B.

Per-module budget map. Rather than treating $\mathbf{b}$ as a fixed hyperparameter vector, we parameterize a *budget policy* $B \mapsto \mathbf{b}(B;\omega)$ with parameters ω. Conceptually, $\mathbf{b}(B;\omega)$ allocates the available compute among modules, and must satisfy

$$0 \le b_\ell(B;\omega) \le 1, \qquad \ell \in [L], \tag{9}$$

together with a qualitative monotonicity requirement: larger B should not (systematically) reduce allowed capacity. A convenient implementation is a coordinate-wise monotone map, e.g. $b_\ell(B;\omega) = \sigma(g_\ell(B;\omega))$ with g_ℓ nondecreasing in B (enforced by positive weights or isotonic calibration). This produces a differentiable interface through which gradients can flow from query loss and constraint penalties to ω, allowing the allocation policy to be learned from data rather than manually set.

Structured masks versus unstructured masks. The constraints are expressed at module granularity via expected activation fractions, and the choice of mask family governs both trainability and whether sparsity translates into realized device savings. In the most permissive (but systems-weak) form, z_m is unstructured and factors across coordinates, yielding

$$z_{m,j} \sim \mathrm{Bernoulli}(\pi_{m,j}), \qquad \pi_{m,j} \in (0,1), \tag{10}$$

with $\pi_{m,j}$ produced by learned logits (optionally per-module). While such masks can satisfy $\mathbb{E}[\mathrm{nnz}_\ell(z_m)/d_\ell] \le b_\ell$, they rarely produce proportional wall-clock improvements without specialized kernels.

Accordingly, we primarily target *structured* mask families that are compatible with standard acceleration (block sparsity, head pruning, MLP neuron groups, low-rank adapters with rank gating, etc.). Abstractly, we let each module ℓ be partitioned into K_ℓ groups, and we gate at the group level:

$$z_{m,\ell,k} \in \{0,1\}, \qquad z_m \odot \Delta \;\equiv\; \bigoplus_{\ell=1}^{L} \bigoplus_{k=1}^{K_\ell} z_{m,\ell,k}\, \Delta_{\ell,k}, \tag{11}$$

where $\Delta_{\ell,k}$ denotes the delta parameters belonging to group k in module ℓ and $\oplus$ denotes concatenation in parameter space. The module budget is then enforced in terms of expected active groups (or expected active parameters induced by groups), which provides a closer correspondence between the constraint and real compute reductions.

For differentiable optimization we relax the binary gates via a continuous distribution (e.g. hard-concrete), writing $\tilde{z}_{m,\ell,k} \in [0,1]$ with reparameterizable samples. The per-module capacity constraint becomes

$$\mathbb{E}\left[\frac{\mathrm{nnz}_\ell(\tilde{z}_m)}{d_\ell}\right] \;\le\; b_\ell(B;\omega), \qquad \ell \in [L], \tag{12}$$

where $\mathrm{nnz}_\ell(\tilde{z}_m)$ is interpreted as an expected active mass (or expected number of active groups mapped to an equivalent parameter count). This retains the modular accounting needed for budget allocation, while remaining compatible with gradient-based learning.

Budget-conditioned routing. Each episode T_i produces mixing weights α_i through a routing/controller network

$$\alpha_i \;=\; h_\zeta(T_i^s, B), \qquad \alpha_i \ge 0,\; \sum_{m=1}^{M} \alpha_{i,m} = 1, \tag{13}$$

with parameters ζ. The input T_i^s can be summarized by any permutation-invariant representation of the support set (e.g. pooling of frozen features $f_{\theta^{\mathrm{pre}}}(x)$ and labels), and the scalar B can be embedded and concatenated to this representation. Conditioning on B is essential: even if the task identity is fixed, different budgets should select different mixtures of experts and different effective adaptation capacities. In practice we interpret h_ζ as providing a *soft* selection among experts, with the masks z_m controlling which parts of Δ each expert can express.

Primal–dual training objective. We optimize a Lagrangian relaxation of the constrained robust meta-objective. Let $\psi = (\Delta, \phi, \zeta, \omega)$ collect primal parameters, where ϕ parameterizes the mask distributions (e.g. hard-concrete logits per expert and group). For a sampled task T_i and a training-time budget B_{train}, we form θ_i as in (8) using $\alpha_i = h_\zeta(T_i^s, B_{\text{train}})$ and masks drawn (or deterministically set) according to ϕ, subject to the current budget policy $\mathbf{b}(B_{\text{train}}; \omega)$. We then define constraint residuals

$$v_{i,\ell} \;=\; \mathbb{E}\Big[\frac{\mathrm{nnz}_\ell(\tilde{z})}{d_\ell}\Big] - b_\ell(B_{\text{train}}; \omega), \qquad u_i \;=\; \widehat{\mathrm{Lat}}(\theta_i) - B_{\text{train}}, \tag{14}$$

and the per-task Lagrangian contribution

$$\mathcal{J}_i(\psi, \lambda, \mu) \;=\; \mathcal{L}_{T_i}\big(\theta_i\big) \;+\; \sum_{\ell=1}^{L} \lambda_\ell \, v_{i,\ell} \;+\; \mu \, u_i, \qquad \lambda_\ell, \mu \geq 0. \tag{15}$$

The multipliers $\lambda = (\lambda_\ell)_{\ell \in [L]}$ enforce module capacities, while μ enforces the latency proxy budget. This separation is operationally important: λ redistributes capacity across modules as dictated by the data, whereas μ regulates the overall aggressiveness of adaptation as budgets change.

Stochastic primal–dual updates and feasibility enforcement. Meta-training alternates between (stochastic) descent on primal variables and projected ascent on dual variables. For a batch of tasks $\mathcal{B}$ we perform

$$\psi \leftarrow \psi - \eta \, \nabla_\psi \Big(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{J}_i(\psi, \lambda, \mu) \Big), \tag{16}$$

$$\lambda_\ell \leftarrow \Big[\lambda_\ell + \eta_\lambda \Big(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} v_{i,\ell} \Big) \Big]_+, \qquad \mu \leftarrow \Big[\mu + \eta_\mu \Big(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} u_i \Big) \Big]_+, \tag{17}$$

where $[\cdot]_+$ denotes projection to $\mathbb{R}_+$. To stabilize training in the nonconvex regime, it is often beneficial to include a simple feasibility heuristic: when empirical violations remain negative over a window, we damp or reset the corresponding multipliers (preventing over-penalization once constraints are satisfied), and we clamp $\mathbf{b}(B; \omega)$ within (9) at all times.

At meta-test time, given (T^s, B), we solve a small constrained routing+budget subproblem over $(\alpha, \mathbf{b})$ (and optionally relaxed mask parameters) using a few projected gradient or dual ascent steps, holding (Δ, ϕ) fixed. The resulting configuration defines $\theta = \theta^{\text{pre}} + \sum_m \alpha_m (z_m \odot \Delta)$, which is then used for prediction on the query set. This decomposition—heavy learning of shared structure in ψ and light budget-conditioned optimization at deployment—is the mechanism by which AutoBudget-SMAT achieves explicit feasibility while retaining task adaptivity.

Latency proxy and systems model. The constraint $\widehat{\mathrm{Lat}}(\theta_i) \le B$ is only useful insofar as $\widehat{\mathrm{Lat}}$ reflects what a deployment system will actually execute. We therefore treat latency estimation as a *kernel-aware* modeling problem tied to a device profile $\mathcal{D}$ (hardware, compiler/runtime, sequence length regime, batch size, precision, and sparsity kernel availability). Concretely, our module partition $\{\ell \in [L]\}$ is chosen to coincide with operator blocks whose execution time is separately measurable (e.g. attention projections, attention output, MLP up/down projections, layer norms, and task heads). For each such module we maintain an empirical cost model $c_\ell(\mathcal{D})$ that maps a *structured activation level* to a predicted incremental latency.

A kernel-aware proxy from structured gates. Because we enforce sparsity through structured groups (cf. (11)), the relevant control variable is not a coordinate-wise ℓ_0 count but a group-level activation statistic. For a task instantiation θ_i and module ℓ, we define an (expected) activation level

$$s_{i,\ell} \;:=\; \mathbb{E}\left[\frac{1}{K_\ell}\sum_{k=1}^{K_\ell} \tilde{z}_{\ell,k}\right] \;\in\; [0,1], \tag{18}$$

where for notational simplicity we suppress the expert index and interpret $\tilde{z}_{\ell,k}$ as the effective gate mass after combining experts (e.g. via an α-weighted mixture of per-expert gates, or via a deterministic selection rule when only one expert is activated). The latency proxy is then modeled as

$$\widehat{\mathrm{Lat}}(\theta_i;\mathcal{D}) \;=\; \mathrm{Lat}\big(\theta^{\mathrm{pre}};\mathcal{D}\big) \;+\; \sum_{\ell=1}^{L} \widehat{c}_\ell\big(s_{i,\ell};\mathcal{D}\big), \tag{19}$$

where $\mathrm{Lat}(\theta^{\mathrm{pre}};\mathcal{D})$ is a measured constant and $\widehat{c}_\ell(\cdot;\mathcal{D})$ is a differentiable function capturing the cost (or savings) induced by sparsity in module ℓ. The key point is that $\widehat{c}_\ell$ is *kernel-aware*: it is not assumed linear in $s_{i,\ell}$, since real kernels often exhibit quantization effects (e.g. block sizes), non-negligible launch overhead, and discrete algorithm switches. A simple and effective choice is a monotone piecewise-linear or piecewise-quadratic function, implemented in a differentiable manner (e.g. via softplus-smoothed hinges), and constrained to be convex in $s_{i,\ell}$ to preserve tractability in the surrogate analysis.

Calibration by measurement and conservative fitting. We calibrate $\widehat{c}_\ell(\cdot;\mathcal{D})$ by direct timing measurements on $\mathcal{D}$ (or on a faithful simulator). For each module ℓ we generate a grid of activation levels $\{s^{(t)}\}_{t=1}^{T}$ and instantiate corresponding structured masks (e.g. enabling a specified number of groups), then measure the realized end-to-end latency of the full model under a fixed inference setting (sequence length, batch size, precision). To isolate module contributions, we either (i) microbenchmark each module in isolation with

representative tensor shapes, or (ii) fit the additive model (19) from end-to-end measurements by regressing the observed latency residual against the vector $(s_{i,\ell})_{\ell\in[L]}$. In both cases we prefer a *conservative* fit: rather than least squares, we fit an upper envelope (e.g. a high-quantile regression or a least-squares model with a positive slack margin) so that $\widehat{\mathrm{Lat}}$ is biased upward. This choice aligns with feasibility at deployment: it is preferable to reject some configurations that would in fact run under budget than to accept configurations that violate the budget.

Proxy-to-wall-clock distortion model. Even with kernel-aware calibration, latency exhibits unavoidable variation from runtime jitter, memory effects, and interactions across modules (operator fusion, cache reuse, and parallelism). We therefore formalize the proxy error by distortion parameters $(\kappa, \epsilon_{\mathrm{lat}})$ as in (H3). For each profile $\mathcal{D}$ we postulate that, with probability at least $1-\delta$ over the measurement noise (and any randomized masking used at inference),

$$\mathrm{Lat}(\theta;\mathcal{D}) \;\le\; \kappa\,\widehat{\mathrm{Lat}}(\theta;\mathcal{D}) + \epsilon_{\mathrm{lat}}, \qquad \text{for all } \theta \text{ in the considered model family.} \tag{20}$$

This inequality should be read as a *high-probability one-sided bound*: we design $\widehat{\mathrm{Lat}}$ to be an approximate upper bound, but allow a multiplicative looseness $\kappa \ge 1$ and an additive slack $\epsilon_{\mathrm{lat}} \ge 0$ to accommodate systematic bias and fixed overheads (kernel launches, runtime framework overhead, and I/O). While (20) is an assumption, it can be empirically validated by holding out configurations not used in calibration and reporting coverage at level $1-\delta$.

Implications for feasibility under a true budget. If a deployment budget is specified in terms of true latency, i.e. $\mathrm{Lat}(\theta;\mathcal{D}) \le B$, then (20) implies a sufficient (proxy) condition

$$\widehat{\mathrm{Lat}}(\theta;\mathcal{D}) \;\le\; \frac{B-\epsilon_{\mathrm{lat}}}{\kappa} \quad\Longrightarrow\quad \mathrm{Lat}(\theta;\mathcal{D}) \le B \qquad \text{(on the event in (20)).} \tag{21}$$

Accordingly, when we enforce $\widehat{\mathrm{Lat}}(\theta_i) \le B$ inside the optimization, we may optionally interpret B as a *shrunk* effective budget $(B_{\mathrm{true}} - \epsilon_{\mathrm{lat}})/\kappa$ so that proxy-feasibility implies wall-clock feasibility with the desired confidence. This is also the point at which device-specific overhead enters: if ϵ_{lat} is dominated by constant runtime costs, then very small budgets are necessarily conservative, whereas for large models the multiplicative term κ typically dominates.

Multiple device profiles and budget descriptors. When deployment may occur on a set of profiles $\{\mathcal{D}_r\}$, we maintain either a separate proxy

(19) for each $\mathcal{D}_r$, or a single proxy parameterized by a learned embedding of $\mathcal{D}$. The budget descriptor B provided to the routing/budget policy can then be interpreted as profile-conditional (e.g. milliseconds on $\mathcal{D}_r$) or normalized (e.g. as a fraction of $\mathrm{Lat}(\theta^{\mathrm{pre}}; \mathcal{D}_r)$). In either case, the policy must learn to allocate capacity across modules in a manner compatible with the profile-dependent costs $c_\ell(\mathcal{D}_r)$: a module that is relatively expensive on one device (e.g. memory-bound MLPs) may be relatively cheap on another.

Limitations and failure modes. The proxy is only as good as the correspondence between *mask structure* and *kernel behavior*. Unstructured sparsity (coordinate-wise Bernoulli masks) typically fails to yield proportional savings, and even structured sparsity can underperform if kernels do not exploit the structure (e.g. missing support for certain block sizes or ranks). Moreover, end-to-end latency is not strictly additive across modules; (19) is an approximation whose error can grow when operator fusion or reordering changes with sparsity patterns. Finally, latency depends on inference-time details not represented in θ alone (sequence length distribution, KV-cache reuse, and compilation caching). For these reasons we treat $\widehat{\mathrm{Lat}}$ as a calibrated, differentiable *control signal* rather than a perfect predictor, we explicitly account for distortion via $(\kappa, \epsilon_{\mathrm{lat}})$, and we expect periodic recalibration when the deployment stack changes.

Convex surrogate regime and decision variables. For the purpose of analysis we consider a surrogate in which the feature extractor induced by θ^{pre} is frozen and the task loss depends on the adapted parameters only through a convex prediction layer (or, more generally, through a linearization of the network around θ^{pre}). In this regime the effective decision variables for a single episode T can be taken as the routing weights $\alpha \in \Delta^{M-1}$ together with continuous (relaxed) structured gates, which we summarize by the per-module activation levels $s = (s_\ell)_{\ell \in [L]} \in [0,1]^L$ as in (18). We write the expected query loss as $\ell_T(\alpha, s)$ and assume ℓ_T is convex and G-Lipschitz in (α, s), uniformly over $T \sim P_{\mathrm{ID}}$. Likewise, we assume the proxy cost term $\widehat{\mathrm{Lat}}(\alpha, s) := \widehat{\mathrm{Lat}}(\theta^{\mathrm{pre}}) + \sum_{\ell=1}^{L} \widehat{c}_\ell(s_\ell)$ is convex and H-Lipschitz in s. The per-module capacity constraints take the convex form

$$g_\ell(\alpha, s; \mathbf{b}) \;:=\; s_\ell - b_\ell \;\le\; 0, \qquad \ell \in [L], \tag{22}$$

and the (proxy) deployment constraint is

$$h(\alpha, s; B) \;:=\; \widehat{\mathrm{Lat}}(\alpha, s) - B \;\le\; 0. \tag{23}$$

We emphasize that $\mathbf{b}$ may itself be a learned quantity (or the output of a learned map $B \mapsto \mathbf{b}(B)$), but in the convergence statement below we treat $\mathbf{b}$ and B as fixed, since the standard primal–dual argument applies conditionally on the constraint specification.

Lagrangian formulation and ε-KKT. Define the (episode-wise) Lagrangian

$$\mathcal{L}_T(\alpha, s; \lambda, \mu) \;:=\; \ell_T(\alpha, s) \;+\; \sum_{\ell=1}^{L} \lambda_\ell \, g_\ell(\alpha, s; \mathbf{b}) \;+\; \mu \, h(\alpha, s; B), \tag{24}$$

with dual variables $\lambda \in \mathbb{R}_+^L$ and $\mu \in \mathbb{R}_+$. An ε-KKT point for the constrained problem consists of $(\alpha^\star, s^\star, \lambda^\star, \mu^\star)$ such that (i) primal feasibility holds up to ε, namely $g_\ell(\alpha^\star, s^\star; \mathbf{b}) \le \varepsilon$ and $h(\alpha^\star, s^\star; B) \le \varepsilon$ for all ℓ; (ii) dual feasibility $\lambda^\star, \mu^\star \ge 0$ holds exactly; (iii) complementary slackness holds up to ε, i.e. $\lambda_\ell^\star \, g_\ell(\alpha^\star, s^\star; \mathbf{b}) \le \varepsilon$ and $\mu^\star h(\alpha^\star, s^\star; B) \le \varepsilon$; and (iv) stationarity holds up to ε in the sense that the norm of the projected gradient of $(\alpha, s) \mapsto \mathbb{E}_T[\mathcal{L}_T(\alpha, s; \lambda^\star, \mu^\star)]$ at $(\alpha^\star, s^\star)$ is at most ε.

Stochastic primal–dual convergence. Consider stochastic projected primal–dual mirror descent on (24), where at iteration t we sample $T_t \sim P_{\mathrm{ID}}$, take a (sub)gradient step in (α, s) with step size η_t, and update (λ, μ) by projected ascent:

$$\lambda_{\ell,t+1} = \big[\lambda_{\ell,t} + \eta_t \, g_\ell(\alpha_t, s_t; \mathbf{b})\big]_+, \qquad \mu_{t+1} = \big[\mu_t + \eta_t \, h(\alpha_t, s_t; B)\big]_+. \tag{25}$$

Under boundedness of the feasible set and the Lipschitz assumptions above, standard saddle-point arguments yield that with $\eta_t = \Theta(1/\sqrt{T})$ the iterate average $(\bar{\alpha}_T, \bar{s}_T)$ satisfies both expected suboptimality and expected constraint violation of order $O(1/\sqrt{T})$. Equivalently, there exists an iterate (or an average) that is ε-KKT after $\tilde{O}(1/\varepsilon^2)$ stochastic first-order calls. The salient point for our setting is that the per-module constraints (22) do not alter the qualitative rate: they only increase the dimension of the dual by L, while dual feasibility is maintained by the projections in (25). In the full nonconvex transformer instantiation, we use the same updates as a heuristic; the surrogate theorem should be read as an explanation of why the mechanism (dual ascent on constraint violations coupled to primal descent) is a principled way to enforce budgets.

Pareto frontier via Lagrange multipliers. In the convex surrogate, the constrained problem induces a natural bi-objective trade-off between expected query loss and expected proxy latency. Consider the supported Pareto set of pairs

$$\Big(\mathbb{E}_T[\ell_T(\alpha, s)],\; \mathbb{E}_T[\widehat{\mathrm{Lat}}(\alpha, s)]\Big)$$

over feasible (α, s). By convex analysis, every supported Pareto-optimal point is a minimizer of a weighted sum $\mathbb{E}_T[\ell_T(\alpha, s)] + \mu \, \mathbb{E}_T[\widehat{\mathrm{Lat}}(\alpha, s)]$ for some $\mu \ge 0$, and conversely every μ yields a supported point (possibly nonunique). The multipliers (λ, μ) therefore play two roles: μ parameterizes

the loss–latency trade-off, while λ encodes the *shadow prices* of per-module capacity. When module costs $c_\ell(\mathcal{D})$ are heterogeneous, the ability to tune λ_ℓ separately amounts to reallocating capacity toward modules with higher marginal utility per unit cost, rather than enforcing a single global sparsity target τ that implicitly sets the same shadow price across all modules.

Separation between global-τ and per-module budgets. The preceding observation can be made formal by exhibiting instances in which any global sparsity constraint is strictly suboptimal. In a two-module surrogate ($L = 2$), let the feasible decisions be $(s_1, s_2) \in [0, 1]^2$, let the latency be additive with unequal costs, say $\widehat{\mathrm{Lat}}(s) = c_1 s_1 + c_2 s_2$ with $c_1 \gg c_2$, and let the expected loss be convex and separable with sharply different curvature, e.g.

$$\mathbb{E}_T[\ell_T(s)] = a_1(1 - s_1)^2 + a_2(1 - s_2)^2, \qquad a_1 \gg a_2 > 0. \tag{26}$$

Under a fixed latency budget $c_1 s_1 + c_2 s_2 \leq B$, the optimal solution allocates most capacity to module 1 (large a_1) until its marginal benefit matches that of module 2 scaled by costs. In contrast, a global-τ constraint forces s_1 and s_2 to follow a single sparsity level (or an equivalent shared budget), which prevents this cost-adjusted equalization of marginal gains. One may verify directly (by solving the convex programs in closed form) that there exists $\Delta > 0$ such that the minimum of (26) under the global-τ feasible set exceeds the minimum under the per-module (or latency) feasible set by at least Δ, even when the two solutions match in total proxy latency. This demonstrates that the improvement from learned per-module budgets is not merely empirical but can reflect a genuine increase in the expressive power of the constraint set.

First-order oracle lower bounds. Finally, the $\tilde{O}(1/\varepsilon^2)$ iteration complexity in the convex surrogate is essentially unimprovable for stochastic first-order methods. Indeed, even without constraints, stochastic convex optimization with Lipschitz gradients admits an $\Omega(1/\varepsilon^2)$ lower bound on the number of oracle calls needed to reach expected suboptimality ε. Constrained problems inherit this hardness by restricting attention to instances where the optimum lies in the interior (so constraints are inactive) or by a standard reduction that embeds an unconstrained hard instance into a feasible set with a simple projection operator. Thus, while better constants and adaptivity are possible, one should not expect asymptotically faster rates from any method that accesses the meta-objective only through noisy gradients of episodic query losses and convex proxy constraints.

Hardness of discrete per-module budgeting. We now isolate a basic combinatorial subproblem that already captures the difficulty of enforcing

deployment budgets with *discrete* architectural decisions. Consider a simplified instantiation in which routing is fixed (e.g. $M = 1$ and $\alpha \equiv 1$) and each module $\ell \in [L]$ is either fully active or fully inactive. Let $x_\ell \in \{0, 1\}$ denote whether the delta for module ℓ is enabled (equivalently, whether the structured mask for that module is nonzero). Assume an additive proxy latency model of the form

$$\widehat{\mathrm{Lat}}(x) \;=\; \widehat{\mathrm{Lat}}(\theta^{\mathrm{pre}}) \;+\; \sum_{\ell=1}^{L} c_\ell \, x_\ell, \qquad c_\ell > 0, \tag{27}$$

and a per-episode loss that (after linearization or in a regime where marginal improvements decouple) depends on the selected modules through a known linear benefit,

$$\ell_T(x) \;=\; \ell_T(0) \;-\; \sum_{\ell=1}^{L} v_\ell(T) \, x_\ell, \qquad v_\ell(T) \geq 0. \tag{28}$$

Given a budget B, the per-episode discrete budgeting problem is

$$\min_{x \in \{0,1\}^L} \; \ell_T(x) \quad \text{s.t.} \quad \widehat{\mathrm{Lat}}(x) \leq B, \tag{29}$$

or, equivalently, maximize $\sum_\ell v_\ell(T) x_\ell$ subject to $\sum_\ell c_\ell x_\ell \leq B - \widehat{\mathrm{Lat}}(\theta^{\mathrm{pre}})$.

Reduction from knapsack. We claim that (29) is NP-hard, even under the benign assumptions (27)–(28). Indeed, consider an instance of 0–1 knapsack with items $\{1, \ldots, L\}$, weights $w_\ell \in \mathbb{N}$, values $u_\ell \in \mathbb{N}$, and capacity $W \in \mathbb{N}$. We map item ℓ to module ℓ by setting

$$c_\ell := w_\ell, \qquad v_\ell(T) := u_\ell, \qquad B := \widehat{\mathrm{Lat}}(\theta^{\mathrm{pre}}) + W.$$

Then any feasible selection x corresponds to a feasible knapsack subset, and the objective in (29) differs from the knapsack value $\sum_\ell u_\ell x_\ell$ only by the constant $\ell_T(0)$. Therefore an optimizer of (29) yields an optimizer of knapsack. Since 0–1 knapsack is NP-hard, so is (29). The same construction yields NP-hardness for exact-fit variants (subset sum) by taking $v_\ell(T)$ to encode a feasibility decision and setting a target value threshold.

Hardness persists under additional structure. The preceding reduction uses only one binary decision per module. The full SMAT-style adaptation space is strictly richer: we may allow binary masks at finer granularity (within modules), multiple experts $\{z_m\}_{m=1}^M$, and task-dependent routing α_i. Restricting these richer models to the special case above recovers (29), hence any exact test-time solver that optimizes discrete masks under a budget would solve knapsack in the worst case. In particular, even if we replace (28) by a more faithful convex surrogate loss, the *discrete* feasibility set $\{0, 1\}^L$ already introduces combinatorial complexity independent of the curvature of the loss.

Why pseudo-polynomial algorithms do not resolve the meta-learning setting. One might object that knapsack admits pseudo-polynomial dynamic programming and an FPTAS. However, these guarantees are not directly actionable here for three reasons. First, the effective number of decisions in structured masking is typically much larger than L (e.g. block-wise gates across attention projections and MLPs), making pseudo-polynomial dependence on the budget magnitude prohibitive. Second, our objective is not a fixed, known value vector v_ℓ but an episode-dependent loss $\ell_T(\cdot)$ accessed only through stochastic gradients on (T^s, T^q); thus, even evaluating the benefit of a candidate subset is costly and noisy. Third, we require *fast* test-time adaptation (ideally a handful of forward passes and cheap projected steps), whereas exact or approximation schemes for combinatorial selection generally entail nontrivial per-instance computation and non-differentiable choices that hinder end-to-end learning.

Necessity of relaxations. The hardness result should be read as a structural justification for replacing discrete gates by continuous surrogates. Concretely, we introduce relaxed activation variables $s_\ell \in [0, 1]$ and optimize over (α, s) in a convex (or approximately convex) domain, subject to proxy constraints such as

$$s_\ell \le b_\ell, \qquad \widehat{\mathrm{Lat}}(\alpha, s) \le B.$$

This relaxation eliminates the combinatorial nature of (29) by permitting fractional allocations. When $\ell_T(\alpha, s)$ and $\widehat{\mathrm{Lat}}(\alpha, s)$ are convex in (α, s), the resulting constrained program can be solved to an ε-KKT point by standard first-order primal–dual methods, and the test-time subproblem becomes a small convex optimization rather than a subset-selection search.

Stochastic masks as differentiable approximations to discrete selection. In the nonconvex transformer regime, we do not truly solve a convex program, but the same relaxation principle applies: we parameterize masks via continuous distributions (e.g. hard-concrete or Gumbel-softmax relaxations) so that $\mathbb{E}[z]$ behaves like s and gradients propagate through the mask parameters. The per-module budgets b_ℓ then constrain either the expected nonzero fraction or a structured activation statistic. From the perspective of (29), we have replaced the intractable binary decision $x_\ell \in \{0, 1\}$ with a differentiable proxy $s_\ell \in [0, 1]$ that can be optimized jointly with routing α and the shared delta Δ.

Rounding, feasibility, and integrality effects. Relaxation inevitably raises the question of how fractional solutions translate to executable sparse models. In practice, structured execution (e.g. block sparsity) permits deterministic rounding by thresholding gates or sampling a binary mask according to learned probabilities. While worst-case integrality gaps can be large for

knapsack-type relaxations, our goal is not worst-case optimality but reliable budget satisfaction and good empirical risk under P_{ID} with robustness to P_{OOD}. The dual variables (shadow prices) learned during meta-training provide a mechanism to bias solutions away from near-violations, improving the probability that rounding respects $\widehat{\mathrm{Lat}} \leq B$ and, via the proxy distortion assumption, that true latency is controlled on device profiles.

Implication for the design of the test-time solver. The reduction above implies that, absent relaxations, a test-time procedure that exactly selects discrete modules under a budget is unlikely to be simultaneously (i) general, (ii) fast, and (iii) optimal. This motivates our design choice: we restrict the test-time solver to optimizing a continuous routing+budget parameterization (and, if needed, a low-step dual update) so that deployment feasibility can be enforced by projection or Lagrangian penalties. In short, the relaxation is not an aesthetic preference but a computational necessity dictated by the combinatorial hardness of discrete budgeting under additive resource constraints.

5 Experiments

Experimental questions. We design experiments to validate three claims suggested by the preceding optimization and hardness discussion: (i) budget-conditioned *per-module* capacity learning yields a strictly better accuracy–latency trade-off than any fixed global sparsity target τ (including SMAT-style fixed-τ baselines); (ii) the learned primal–dual policy produces models that satisfy deployment budgets under measured device latency, not merely under a proxy; and (iii) the learned budget policy transfers across deployment profiles, in the sense that a policy trained with a particular set of profiles $\{\mathcal{D}_r\}$ continues to trace a competitive Pareto frontier on unseen profiles.

Few-shot benchmarks and OOD protocol. Our primary meta-learning benchmark is Meta-Dataset, using the standard episodic protocol with support/query splits and dataset-balanced sampling during meta-training. We treat the canonical Meta-Dataset training datasets as inducing P_{ID} and evaluate on both (a) held-out episodes from the same datasets (ID generalization across episodes) and (b) datasets excluded from meta-training (OOD tasks). To stress distributional shift beyond the Meta-Dataset suite, we further introduce an *extra-OOD* pool constructed from datasets with different image statistics and label spaces (e.g. fine-grained natural domains and sketch/clipart-like domains). Episodes are generated by sampling N-way K-shot classification tasks with a fixed query size per class; we vary (N, K) to examine whether the learned budget policy is sensitive to task difficulty. When reporting robustness, we summarize performance not only by mean

query accuracy but also by tail-aware criteria (e.g. CVaR_β over tasks for a fixed β) to approximate evaluation under an unknown mixture P_{OOD}.

Backbones and module partitions. We instantiate $f_{\theta^{\mathrm{pre}}}$ as (i) a vision transformer backbone (ViT) for episodic image classification and (ii) an instruction-tuned language model for few-shot text classification and lightweight generation tasks, implemented through parameter-efficient adaptation. For ViT, we partition parameters into L modules aligned with transformer blocks and subcomponents (attention projections, attention output, MLP up/down projections, and the classification head). For language models, we focus on adapter-style modules (e.g. low-rank or MLP adapters injected into attention/MLP blocks) and treat each insertion site as a module. In both cases we meta-learn a shared dense delta Δ together with M expert masks $\{z_m\}$ and a routing network producing $\alpha = h_\zeta(T^s, B)$, so that test-time adaptation amounts to constructing $\theta(T) = \theta^{\mathrm{pre}} + \sum_m \alpha_m (z_m \odot \Delta)$ and solving a small routing+budget subproblem. We implement structured sparsity at the module level and (when supported by kernels) at a block level within each module; the module budgets b_ℓ constrain either the expected nonzero fraction or an activation-statistic proxy consistent with the deployed kernels.

Hardware targets, latency proxy, and true measurement. To make the budget constraints concrete, we consider multiple deployment profiles $\mathcal{D}$ spanning GPU and edge settings. For each profile, we construct a proxy $\widehat{\mathrm{Lat}}(\theta; \mathcal{D})$ by summing per-module costs $c_\ell(\mathcal{D})$ calibrated by microbenchmarks and applying a sparsity-aware correction for structured execution. We then measure true end-to-end latency $\mathrm{Lat}(\theta; \mathcal{D})$ by repeated timed forward passes with warm-up and fixed batch sizes matching the episodic query setting. Budgets B are specified in milliseconds (or equivalently in normalized units after subtracting the frozen backbone baseline), and during training we sample B from a range that spans tight to loose regimes, so that the learned mapping $B \mapsto \mathbf{b}(B)$ is identifiable. We report (a) the average and tail query loss/accuracy as a function of B and (b) the empirical feasibility rate $\mathbb{P}[\mathrm{Lat}(\theta(T); \mathcal{D}) \le B]$ over tasks, which directly tests the effect of proxy distortion (cf. the $\kappa, \epsilon_{\mathrm{lat}}$ model) on deployment reliability.

Baselines. We compare against four classes of methods, all using the same frozen pretrained backbone for fairness. First, *SMAT with fixed global sparsity* chooses a single τ (or a fixed global mask budget) and learns masks/routing under that constraint, without per-module budgets and without budget conditioning at test time. Second, *structured pruning* baselines allocate a fixed per-layer sparsity pattern learned offline (global or layerwise magnitude pruning, movement pruning, or group-Lasso variants) and then perform episodic

adaptation by fine-tuning only the remaining parameters, again without a budget-conditioned policy. Third, *slimmable/width-scaling* baselines vary the effective width or number of active blocks by a hand-designed schedule indexed by B, with a controller that does not solve a constrained optimization and does not learn module-wise shadow prices. Fourth, we include a *continuous relaxation ablation* that optimizes a global continuous sparsity variable (a single gate shared across modules) to isolate the contribution of *per-module* flexibility from the contribution of continuous optimization itself.

Training and test-time procedures. All methods are trained episodically with matched compute budgets and the same number of meta-iterations. For our method, we implement the primal–dual updates described earlier: primal variables include (Δ, ζ) and mask parameters (e.g. hard-concrete logits), and dual variables include $\{\lambda_\ell\}$ for per-module constraints and μ for the global latency budget. The budget policy $\mathbf{b}(B; \omega)$ is parameterized to be monotone in B (implemented by a constrained network or by monotone spline parameterization) to avoid pathological non-monotone allocations. At meta-test time, given (T^s, B), we run a low-step projected optimization (or dual ascent on μ with a small number of iterations) over $(\alpha, \mathbf{b})$ to enforce $\widehat{\mathrm{Lat}} \leq B$, and then instantiate a sparse model for prediction on T^q. We emphasize that this test-time computation is forward-pass dominated and does not require backpropagating through the backbone.

Pareto frontier and deployment transfer evaluation. For each device profile $\mathcal{D}$ and each method, we sweep budgets B and obtain a set of points

$$\mathcal{S}(\mathcal{D}) = \{(\mathrm{Lat}(\theta(T; B); \mathcal{D}),\ \mathrm{Acc}(T; B))\},$$

aggregated over tasks T (reporting mean and CVaR_β). We then compute the empirical Pareto frontier (non-dominated set) and summarize it by (i) domination counts (how often one method dominates another at matched latency), (ii) area-under-frontier style summaries under a fixed latency interval, and (iii) feasibility-adjusted accuracy, defined as accuracy on the subset of tasks satisfying $\mathrm{Lat} \leq B$ (penalizing systematic budget violations). To test deployment transfer, we train the proxy and policy on a subset of device profiles and then evaluate on unseen $\mathcal{D}'$ without re-training, using only the measured $c_\ell(\mathcal{D}')$ (or a small calibration set) to instantiate $\widehat{\mathrm{Lat}}(\cdot; \mathcal{D}')$. This directly probes whether per-module budgeting captures device-invariant structure (modules that are consistently expensive or cheap) and whether the learned dual variables induce conservative behavior under proxy mismatch.

Findings summarized. Across backbones and benchmarks, the budget-conditioned per-module policy traces a frontier that is consistently non-inferior to fixed-τ SMAT and to static structured pruning, with the largest

gains in the tight-budget regime where misallocation across modules is most costly. Feasibility rates under true measured latency improve relative to baselines that enforce only a global sparsity target, indicating that per-module constraints plus a latency-aware test-time solver better align with heterogeneous device costs. Under extra-OOD episodes, the learned policy tends to allocate capacity to modules that most affect representation adaptation, improving tail metrics (e.g. CVaR_β) at fixed latency. Finally, the deployment-transfer evaluation shows that the learned budget map and routing remain competitive on unseen profiles, supporting the intended interpretation of $\widehat{\mathrm{Lat}}$ as a portable proxy up to controlled distortion and motivating the diagnostic studies reported next.

5.1 Ablations and Diagnostics

Overview. We perform a targeted ablation suite to isolate which components are responsible for (i) improved accuracy–latency trade-offs, (ii) high feasibility under *measured* latency, and (iii) robustness under task shift and device mismatch. Throughout, we keep the pretrained backbone, episodic sampling, and training compute fixed, and we report (a) mean and CVaR_β query accuracy at a given budget B, (b) feasibility rate under true latency $\mathbb{P}[\mathrm{Lat}(\theta(T;B);\mathcal{D}) \le B]$, and (c) diagnostic statistics of the constrained optimization (constraint violations and dual variables).

Per-module budget learning versus fixed budgets. Our principal ablation replaces the learned budget policy $\mathbf{b}(B;\omega)$ with fixed budgets $\mathbf{b}^{\mathrm{fix}}$ chosen by heuristics. We consider: (i) *uniform allocation* $b_\ell^{\mathrm{fix}} \equiv \bar{b}$ tuned so that the proxy constraint is met in expectation; (ii) *cost-proportional allocation* $b_\ell^{\mathrm{fix}} \propto 1/c_\ell(\mathcal{D})$, which preferentially allocates to cheaper modules; and (iii) *loss-sensitivity allocation*, where b_ℓ^{fix} is proportional to a frozen-feature sensitivity score computed at initialization (e.g. $\|\nabla_{\theta_\ell}\mathcal{L}\|$ aggregated over episodes), then held constant during meta-training. In all cases we retain the same routing network $h_\zeta(T^s, B)$ and the same mask parameterization, so differences reflect budget learning rather than routing capacity. Empirically, fixed allocations tend to underperform in the tight-budget regime: cost-proportional rules overspend on cheap but low-utility modules, whereas sensitivity rules overfit to ID statistics and degrade OOD tail performance. The learned $\mathbf{b}(B)$ improves both mean and CVaR_β at matched latency, consistent with the separation intuition that module utility and cost are nonuniform and must be co-adapted.

Isolating budget conditioning from per-module flexibility. To distinguish the benefit of *conditioning on B* from the benefit of *having L degrees of freedom*, we include two intermediate variants: (i) per-module budgets

learned but *not* conditioned on B (a single vector $\mathbf{b}$ shared across all budgets), and (ii) a single scalar gate $\gamma(B)$ that scales a *fixed* per-module template $\bar{\mathbf{b}}$ via $b_\ell(B) = \gamma(B)\bar{b}_\ell$. The first tests whether a single sparse configuration suffices; the second tests whether one-dimensional control can approximate the Pareto set. We find that a budget-invariant $\mathbf{b}$ typically matches mid-range budgets but breaks at extremes, while the one-dimensional scaling fails to reallocate capacity across modules as B tightens, producing dominated points on the frontier.

Proxy mismatch sensitivity via controlled distortions. We stress-test feasibility under the proxy distortion model in (H3) by injecting controlled perturbations into the proxy at test time:

$$\widehat{\mathrm{Lat}}'(\theta;\mathcal{D}) \;=\; \kappa'\,\widehat{\mathrm{Lat}}(\theta;\mathcal{D}) + \epsilon'_{\mathrm{lat}},$$

with $(\kappa', \epsilon'_{\mathrm{lat}})$ spanning optimistic to pessimistic regimes. For each perturbation we rerun the test-time routing+budget solver and measure true feasibility. This diagnostic separates two failure modes: (i) solver-induced violations (inadequate optimization steps) and (ii) model-induced violations (proxy systematically misranking configurations). We observe that feasibility degrades gracefully with multiplicative inflation (as expected) but can be brittle to additive offsets when B is close to the backbone baseline; this motivates reporting budgets relative to the frozen baseline and motivates the calibration study below.

Leave-one-profile-out mismatch and cross-profile generalization. We further evaluate proxy mismatch by training the proxy calibration and the budget/routing policy on a subset of profiles and evaluating on an unseen $\mathcal{D}'$. We instantiate $\widehat{\mathrm{Lat}}(\cdot;\mathcal{D}')$ using only per-module microbenchmarks $c_\ell(\mathcal{D}')$ (no policy finetuning). We report the feasibility gap

$$\Delta_{\mathrm{feas}}(B) \;=\; \mathbb{P}[\widehat{\mathrm{Lat}}(\theta) \le B] - \mathbb{P}[\mathrm{Lat}(\theta) \le B],$$

and correlate it with per-module cost rank shifts between $\mathcal{D}$ and $\mathcal{D}'$. We find that most violations on unseen profiles are attributable to a small set of modules whose relative costs change drastically; this suggests that a small amount of per-profile calibration can suffice if it corrects those modules.

Calibration across devices: simple corrections and conservative margins. We study two calibration mechanisms. First, a lightweight affine correction $\mathrm{Lat} \approx a\,\widehat{\mathrm{Lat}} + b$ fit on a small set of sparse configurations, yielding a corrected proxy used only at test time. Second, a conservative margin policy that replaces B by $B - \delta$ in the solver, with δ selected to achieve a target feasibility rate. Both reduce systematic violations; the affine correction is more sample-efficient when the proxy is approximately linear, while

the margin policy is robust when the proxy misranks configurations but still provides an upper envelope. We additionally report the trade-off between feasibility and accuracy induced by increasing δ, making explicit the cost of conservatism.

Robustness to routing noise and limited support. Since routing $\alpha = h_\zeta(T^s, B)$ is computed from few-shot support sets, we inject controlled noise into the routing logits at test time and measure sensitivity of both accuracy and constraint satisfaction. Concretely, if $\tilde{\alpha} = \mathrm{softmax}(\log \alpha + \sigma\xi)$ with $\xi \sim \mathcal{N}(0, I)$, we sweep σ and observe whether the solver can compensate via budget adjustments. We also ablate support size (smaller K) to induce estimation noise endogenously. In both cases, feasibility is relatively stable (since the solver can reduce budgets when uncertainty increases), whereas accuracy degrades smoothly; this indicates that constraint satisfaction is primarily governed by the dual-weighted objective rather than by fragile routing decisions.

Structured versus unstructured sparsity. We compare our default structured sparsity (module- and block-structured masks aligned with kernels) against an unstructured elementwise mask with the same expected nonzero fraction per module. We evaluate (i) proxy fidelity, by measuring the correlation between $\widehat{\mathrm{Lat}}$ and Lat across sampled configurations, and (ii) effective speedups at fixed nominal sparsity. Unstructured masks can match or slightly improve accuracy at a given proxy budget, but they often underdeliver on true latency due to poor hardware utilization; structured masks yield a substantially tighter proxy–latency relationship, which in turn improves feasibility after test-time solving. This experiment supports the design choice that the constraint should be stated in the same granularity at which execution is efficient.

Optimization diagnostics: KKT residuals and dual behavior. Finally, we report diagnostics directly tied to the constrained formulation. For each meta-iteration we track (i) average constraint violations v_ℓ and u (per-module and latency), (ii) the distribution of dual variables λ_ℓ and μ, and (iii) an empirical ε-KKT residual combining stationarity (gradient norm of the Lagrangian), primal feasibility (violations), and complementarity (products $\lambda_\ell v_\ell$ and μu). We observe that the learned policy reaches a stable regime where violations are near zero in expectation and dual variables concentrate on a small set of consistently binding modules, providing an interpretable "shadow price" view of which components dominate the budget in practice.

5.2 Discussion and Future Work

Our development has treated the pretrained parameters θ^{pre} as a fixed prior and has focused optimization effort on a shared delta Δ, sparse expert masks $\{z_m\}_{m=1}^{M}$, and budget-conditioned routing. This framing isolates the constrained adaptation mechanism, but it is natural to ask how the picture changes when the *prior itself* is uncertain or multi-modal. In many realistic deployments, one has access to several pretrained backbones (or checkpoints at different stages), e.g. $\{\theta^{\text{pre},s}\}_{s=1}^{S}$ trained on different corpora or with different inductive biases. A direct extension is a multi-source prior mixture in which each task selects a convex combination of priors and sparse deltas:

$$\theta_i \;=\; \sum_{s=1}^{S} \pi_{i,s}\, \theta^{\text{pre},s} \;+\; \sum_{m=1}^{M} \alpha_{i,m}\, (z_m \odot \Delta), \qquad \pi_i \in \Delta^{S-1},$$

with π_i produced by a support-conditioned controller. The principal technical issue is that device costs and feasibility then depend on the chosen backbone as well as the delta. This suggests a two-level budgeting view: (i) coarse selection among priors (which changes the baseline latency and representational capacity), and (ii) fine per-module allocations for the delta. From a constrained optimization standpoint, this multi-source formulation may reduce OOD fragility by providing a larger feasible set of representational "starting points," at the price of a harder routing problem and potentially increased calibration burden across devices.

A second direction concerns the fact that routing $\alpha = h_\zeta(T^s, B)$ is computed from few-shot evidence and is therefore intrinsically uncertain. Our current implementation is point-estimate routing coupled to a (relaxed) constrained solver at test time. A more principled alternative is uncertainty-aware routing, in which we explicitly maintain a distribution over α (and possibly over $\mathbf{b}$) and optimize a risk-sensitive objective. One simple instantiation is to penalize entropy collapse and to bias toward routes whose performance is stable under perturbations:

$$\min\; \mathbb{E}\big[\mathcal{L}(\theta(\alpha,\mathbf{b}))\big] \;+\; \rho\,\mathrm{Var}\big(\mathcal{L}(\theta(\alpha,\mathbf{b}))\big) \quad \text{s.t.} \quad \widehat{\mathrm{Lat}}(\theta(\alpha,\mathbf{b})) \le B,$$

where the expectation is taken over a variational posterior for $(\alpha, \mathbf{b})$ induced by the support set. A complementary approach is to convert feasibility into a chance constraint $\mathbb{P}[\mathrm{Lat}(\theta;\mathcal{D}) \le B] \ge 1-\delta$ by introducing probabilistic latency models or conservative upper confidence bounds. Either viewpoint turns the test-time subproblem into a small stochastic program; the promise is improved robustness when support sets are ambiguous (e.g. mixed domains) or when budgets are tight and small routing errors can trigger infeasibility.

Third, while we have already emphasized tail-aware metrics (e.g. CVaR_β) as diagnostics, a stronger integration of distributional robustness into training is warranted. The episodic setting provides a natural adversarial lever:

rather than minimizing $\mathbb{E}_{T\sim P_{\mathrm{ID}}}[\mathcal{L}]$, we may optimize a robust objective of the form

$$\min \sup_{Q\in\mathcal{U}(P_{\mathrm{ID}})} \mathbb{E}_{T\sim Q}[\mathcal{L}(\theta(T))] \quad \text{s.t. feasibility},$$

where $\mathcal{U}(P_{\mathrm{ID}})$ can encode, for example, an f-divergence ball, a Wasserstein ball over task descriptors, or a mixture model capturing plausible OOD shifts. In our constrained setting this interacts nontrivially with the dual variables: robustifying the loss may shift which modules become binding, thereby changing the learned "shadow prices" $\{\lambda_\ell\}$ and the induced allocation pattern. Algorithmically, one may view this as a min–max saddle-point problem in which the adversary chooses task weights (or perturbs task sampling) while the learner updates $(\Delta, \zeta, \phi, \omega)$ and multipliers (λ, μ). A concrete open question is to characterize when the robustified objective yields genuinely improved OOD performance *at fixed budget* as opposed to simply producing more conservative (and hence less accurate) solutions; answering this likely requires combining stability arguments with explicit modeling of proxy distortion $(\kappa, \epsilon_{\mathrm{lat}})$.

Fourth, the fidelity of the latency proxy $\widehat{\mathrm{Lat}}$ is a central practical bottleneck. Our assumptions permit bounded distortion, but in realistic accelerators latency depends on memory bandwidth, kernel fusion, and shape-dependent dispatch rules; additive decompositions across modules can fail when modules interact through scheduling. A promising path is to learn tighter device-conditioned proxies that are both differentiable and conservative. One approach is hybrid modeling: retain per-module microbenchmarks $c_\ell(\mathcal{D})$ for interpretability, but correct them with a small learned residual model that ingests structured sparsity patterns (e.g. block sizes and layer shapes) and predicts non-additive overheads. Another approach is to fit quantile regressors or conformal predictors so that the proxy returns an upper confidence bound $\widehat{\mathrm{Lat}}^{\mathrm{UCB}}$ satisfying $\mathrm{Lat}(\theta;\mathcal{D}) \le \widehat{\mathrm{Lat}}^{\mathrm{UCB}}(\theta;\mathcal{D})$ with prescribed probability. In either case, the proxy is no longer merely a training convenience: it becomes part of the deployment contract, and its uncertainty should be propagated into the solver (e.g. replacing $\widehat{\mathrm{Lat}} \le B$ by $\widehat{\mathrm{Lat}}^{\mathrm{UCB}} \le B$).

Finally, we emphasize that the convex surrogate analysis serves as a guide rather than a complete explanation for transformer-scale behavior. The discrete nature of masks, the nonconvexity of deep networks, and the possibility of proxy misranking imply that classical KKT convergence is, at best, approximate. This suggests two complementary research programs: (i) develop better relaxations and rounding procedures with explicit bounds on feasibility loss (relative to the proxy and, when possible, relative to measured latency), and (ii) co-design sparsity structure with hardware execution so that the feasible set induced by per-module budgets more faithfully reflects true device constraints. Progress on these fronts would tighten the loop be-

tween theory, solver behavior, and deployment, and would move constrained meta-adaptation from a heuristic to a more reliable engineering primitive.