Cascaded Transfer Learning for Multiple Tasks

Eloi Campagne [1 2] **Yvenn Amara-Ouali** [3 2] **Yannig Goude** [3 2] **Mathilde Mougeot** [1 4] **Argyris Kalogeratos** [1]

Abstract

Many-Task Learning refers to the setting where a large number of related tasks need to be learned, the exact relationships between tasks are not known, and budget constraints are in place. We introduce the Cascaded Transfer Learning (CTL), a novel many-task transfer learning paradigm where information (e.g. model parameters) cascades hierarchically though tasks that are learned by individual models of the same class, while respecting given budget constraints. The cascade hierarchy is incrementally built by finding ordered task triplets (source, intermediate, target), where the intermediate task is chosen to facilitate the source-target transfer. We give the conditions under which a transfer passing through an intermediate task improves over a direct source-target transfer. Based on these insights, we design a cascaded transfer mechanism that is deployed over a tree structure connecting the tasks, and allocates the available training budget along its branches. Experiments on synthetic and real many-task settings show that the resulting method enables more accurate and cost-effective adaptation across large task collections compared to alternative approaches.

1. Introduction

Modern learning systems increasingly operate in settings where a large number of related tasks must be handled under computational or training-budget constraints. This arises in domains involving personalized, localized, or fine-grained problems, where each task comes with limited data and solving all tasks independently becomes inefficient. Exploiting relationships across tasks is essential for achieving accurate models while maintaining feasible computational cost in such a context.

Multi-task learning (MTL) has been specifically developed for leveraging shared information across tasks, seeking to improve performance by joint training of multiple related tasks. The main MTL method families include feature-sharing architectures, low-rank parameterizations, task clustering and other methods that learn task relationships (Zhang & Yang, 2018; 2021; Ruiz et al., 2024). Although effective, they typically require synchronized and globally coordinated optimization for training across all tasks, which lead to substantial memory or communication overhead. Moreover, relying on accurate task relations is sensitive point for MTL, hence there are methods that try to infer those relations, but in the literature this is most usually required to be apriori known.

As the number of tasks grows, these limitations become more pronounced, giving rise to the *Many-task Learning* (MaTL) setting. Here, the focus shifts to handling hundreds or thousands of related tasks, whose relatedness is typically unknown or only partially observable, and often under resource constraints that make joint optimization impractical. The MaTL regime appears naturally in personalized modeling, distributed sensing, or large-scale prediction environments, where one seeks scalable mechanisms for exploiting task relatedness. Energy networks, transportation, retail, where thousands of related tasks arise in parallel (He et al., 2019), are among the typical application sectors that fall within the MaTL regime. Prior work has shown that scaling to many tasks requires explicit mechanisms to control parameter sharing and interference within a *single* model, such as task-specific routing (Strezoski et al., 2019), hierarchical many-task architectures (Hashimoto et al., 2017; Liu et al., 2021), or transferability-based task selection that guides efficient knowledge reuse across large task collections (Tan et al., 2024). Studies of extreme multi-task pre-training with over one hundred tasks further highlight the benefits and the complexity of this regime (Aribandi et al., 2022). In parallel, foundation models have recently been explored for graphs, with the goal of capturing generalities across heterogeneous graph tasks. A particularly illustrative instance of these pressures is found in modern time-series forecasting. Transformer-based architectures such as PatchTST (Nie et al., 2022) and universal forecasters like Moirai (Woo

[1]Centre Borelli, Université Paris-Saclay, CNRS, Ecole Normale Supérieure Paris-Saclay, France [2]EDF R&D, Palaiseau, France [3]Laboratoire de Mathématiques d'Orsay (LMO), Université Paris-Saclay, CNRS, Faculté des Sciences d'Orsay, France [4]ENSIIE, Évry-Courcouronnes, France. Correspondence to: Eloi Campagne and Argyris Kalogeratos <name.surname@ens-paris-saclay.fr>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

et al., 2024) demonstrate that shared representations can be remarkably effective across a wide variety of series and domains. Yet these approaches typically rely on large-scale centralized pretraining and assume that inference or fine-tuning can be performed without stringent per-task budget constraints. In applications involving thousands of forecasting tasks (each requiring lightweight, task-specific adaptation) global training becomes costly or infeasible, and the problem shifts toward designing scalable mechanisms for transferring information across many related tasks.

Thinking about sharing information from one task to assist another without requiring global joint training, *Transfer Learning* (TL) comes naturally into the discussion. TL methods are categorized to instance-based, feature-based, parameter-based and relational transfer (Pan & Yang, 2009; Weiss et al., 2016; Zhuang et al., 2020). In practice, TL can be straightforward to deploy in modern neural settings and scales well to new tasks, but it treats each source-target pair independently. This pairwise structure limits its applicability into the MaTL regime, as it provides only simple deployment schemes such as star-shaped transfers from a single source model. Beyond such configurations, classical TL offers little guidance on how to organize or coordinate transfers across a large collection of tasks, which makes it inadequate as the primary mechanism for structuring information flow in MTL, not to mention the MaTL regime.

Contributions. In this work, we propose a new approach that occupies the conceptual space between these two paradigms. Rather than adapting a source model directly to each target task, we introduce *Cascaded Transfer Learning*, a mechanism in which knowledge is propagated through a sequence of tasks (see Figure 1). The key idea is that coordinated transfer across a graph structure connecting the tasks can realize gains that are not accessible through isolated, direct adaptations. This perspective extends the idea of sequential transfer and provides a scalable alternative to global multi-task optimization. To the best of our knowledge, our work is the first to provide a theoretical and algorithmic framework for budget-constrained cascaded transfer over large task collections, rather than designing a single many-task model.

Our contributions can be summarized as follows. First, we establish a general theoretical rationale for cascaded transfer, identifying conditions that guarantee improvements over direct adaptation. Second, we design a scalable graph-based algorithm that constructs transfer cascades and distributes a limited training budget over a large collection of tasks. Third, we demonstrate empirically on synthetic and real multisite forecasting datasets that our method improves predictive performance while reducing computational cost relative to standard fine-tuning approaches. Finally, the CTL paradigm is generic as it applies to any setting in which tasks share a common model class and convex loss function, allowing parameters to be transferred from a source to a target task and refined for the latter one. This work therefore positions cascaded transfer as a general and efficient paradigm for many-task learning under budget constraints, offering a practical alternative to classical transfer learning and multi-task learning in large-scale, resource-limited environments.

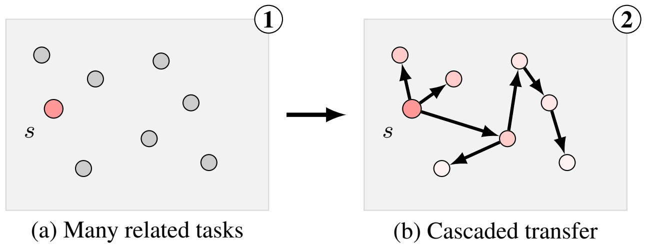


Figure 1. **The Cascaded Transfer Learning (CTL) paradigm.** (a) A collection of related tasks visualized over a landscape in which proximity represents task relatedness. A seed task (red) is selected among all tasks. (b) Cascaded transfer starting from the seed and refining each task in turn along a tree. Darker red color shades indicate nodes refined earlier in the process.

Notations and preliminary concepts. In a multi-task learning context, there are tasks that need to be learned by models of the same class. Each task v has input data $\boldsymbol{X}_v \in \mathbb{R}^{n_v \times d}$ in a d dimensional feature space, it is learned by a model with a parameter vector $\boldsymbol{\theta}_v \in \mathbb{R}^d$. We denote by G_v^k the refinement operator that applies k steps of gradient on its loss. Provided the tasks have information (e.g. parameters or data) belonging to a common space $\mathcal{I}$, more complicated *multisource fusion-transfer* scenarios are possible, i.e. $\{S_1, S_2, ...\} \to T$, as long as a map $\mathrm{Agg} : \mathcal{I}^{|\{S_1,S_2,...\}|} \mapsto \mathcal{I}$ aggregates the information from the source tasks and pass it to the target. A k-budgeted *direct transfer* from a source task S to a target task T, expressed as $S \to T$, consists in initializing the target with the information received from the source (or an aggregated variant thereof), and then refining it through a k-budget refinement operator such as G_T^k.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a structure formed by edges in the set $\mathcal{E}$ connecting pairs of nodes from the set $\mathcal{V}$. A Directed Acyclic Graph (DAG) is a connected directed graph with no directed cycles. A DAG defines a topological linear order in which, for each edge $(i, j) \in \mathcal{E}$, i appears earlier than j in that order. That order can be decomposed in paths connecting the *source* node, which has zero in-degree, to the *sink* nodes that have zero out-degree. A rooted tree $\mathcal{T}$ is a DAG where a node's in-degree is either zero or one; here, the source and sinks are called *root* and *leaves*, respectively.

Algorithm 1 Abstract Cascaded Transfer Learning

```
1: Input: set of tasks $\mathcal{V} = \{v_1, v_2, \dots\}$, common information space $\mathcal{I}$, total budget $B$
2: Output: refined information $\{\widetilde{I}_v\}_{v \in \mathcal{V}}$
3: Seed selection: choose the starting task $s$ of the cascade
4: Graph construction: specify a rooted DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ indicating the information flow through tasks
5: Cascaded transfer: starting from task $s$, propagate information along $\mathcal{G}$; for each task $v$:
   – If $\mathrm{Pa}(v) = \emptyset$, set $I_v^{(0)} = \mathrm{Agg}_v(\cdot)$
   – If $\mathrm{Pa}(v) \neq \emptyset$, set $I_v^{(0)} = \mathrm{Agg}_v\big(\{\widetilde{I}_u : u \in \mathrm{Pa}(v)\}\big)$
   Refine: $\widetilde{I}_v = G_v^{b_v}\big(I_v^{(0)}\big)$
6: Return: refined information $\{\widetilde{I}_v : v \in \mathcal{V}\}$
```

2. Cascaded Transfer Learning

The proposed framework admits a clean formulation as an abstraction over a DAG, allowing the overall methodology to remain conceptually simple.

Definition 2.1 (Cascaded Transfer Learning (CTL)). Let $\mathcal{V}$ be a set of tasks, with information belonging to a common space $\mathcal{I}$, each to be learned by a different model. *Cascaded Transfer Learning* is a learning process that transfers information and refines models in the topological order induced by a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ connecting the tasks.

According to the above definition, in the topological order of $\mathcal{G}$, each task $v \in \mathcal{V}$ receives information from its parents $\mathrm{Pa}(v)$, aggregates it through a map $\mathrm{Agg}_v : \mathcal{I}^{|\mathrm{Pa}(v)|} \mapsto \mathcal{I}$, refines the result using local adaptation, and passes information to each of its children tasks. To simplify the presentation, we henceforth restrict ourselves to cascades over tree graphs, that can be performed by typical source-target direct transfers. Also, all CTL instances aim to learn all the tasks as good as possible, minimizing a global criterion.

The cascaded transfer process is summarized by the high-level Algorithm 1 whose steps are detailed next. This abstract formulation highlights the three central design choices required for any CTL method:

(i) *Seed selection.* The seed s serves as the root of the cascade and provides the initial parameter vector.

(ii) *Graph construction.* A cascade requires a DAG to decompose the global training process into a sequences of local budget-limited transfers. As said, we restrict the process to relying on a rooted tree graph.

(iii) *Budget allocation and refinement.* Given the total training budget B, the refinement budgets must be assigned to tasks and executed along the DAG. Allocation strategies may favor early nodes, deeper nodes, or follow other heuristics.

Figure 1 provides an overview of the CTL framework, illustrating how, starting from a chosen seed, information is propagated from task to task along the cascade.

2.1. From Distance-based to Alignment-based Trees

Designing a suitable transfer hierarchy is a central question in CTL. In an idealized setting where the functional relationships between tasks were known (for instance, through distances between their optimal predictors), one could construct an optimal cascade by computing a minimum-weight rooted tree, such as a minimum spanning tree (MST). This structure connects each task to its most similar neighbor and is optimal for recovering nearest-neighbor chains when distances are exact.

However, in practical settings the available surrogates for functional similarity are inherently noisy. Distances computed from estimated parameters, empirical features, or local statistics may deviate significantly from the true functional distances, especially in low-data regimes. As a result, an MST built over such noisy surrogates can lead to transfer-wise suboptimal cascade trees. In particular, it may miss beneficial two-step chains of the form $S \to I \to T$ that improve transfer relative to the direct connection $S \to T$.

This observation motivates the introduction of alignment-based criteria. Instead of relying solely on pairwise distances, alignment-based methods evaluate how well a candidate intermediate task I aligns with the transfer direction from S to T. Theoretical analysis gives conditions on when the intermediate I is beneficial to T with respect to S. Alignment-based tree construction therefore incorporates a two-step lookahead mechanism: when choosing an edge $u \to v$, the algorithm evaluates not only the immediate surrogate distance between u and v but also how future transfers from v to other tasks are expected to behave. This additional layer of evaluation mitigates the effect of noise in distance surrogates and helps recover transfer structures that are closer to those induced by the true functional geometry.

From this perspective, MST-based trees are the solution to an edge-additive optimization problem when the available pairwise distances are accurate proxies for the underlying functional geometry, whereas noisy distance measurements call for adapting the tree-construction objective and naturally motivate lookahead mechanisms such as alignment-based construction. The former relies on local distance comparisons, while the latter incorporates directional information derived from potential two-step cascades. Both are compatible with the general CTL formulation, and their empirical performance can differ significantly depending on the noise level in the surrogate features.

Extensions. The abstract CTL framework naturally admits several generalizations. First, while we restrict the exposition to a single rooted tree for clarity, the same principles apply to a *cascade forest* in which multiple seeds initiate independent cascades over disjoint subsets of tasks. Selecting such seeds, or deciding when additional seeds are beneficial, is itself a structural design problem that can be informed by similarity measures, clustering criteria, or transferability estimates. Second, CTL does not rely on supervised losses: any setting in which tasks admit a common information space $\mathcal{I}$ and a local adaptation operator can support cascaded transfer, including unsupervised, self-supervised, or representation-learning scenarios. Finally, the DAG formulation of Definition 2.1 suggests a broader family of cascades in which a task may fuse information from multiple parents before refinement, although investigating these richer transfer structures is beyond the scope of the present work.

3. Analysis of Alignment Conditions

Having introduced the CTL framework and its tree-based instantiation, we now analyze the conditions under which an intermediate task can improve transfer performance under a fixed refinement budget. For presentation clarity, we conduct our analysis in a linear regression setting, although the underlying principles extend more broadly. We progressively move from the model parameter space to the feature space, and finally to the noisy setting, where we derive probabilistic guarantees. These developments formalize directional alignment and justify its use for constructing alignment-based cascade trees.

3.1. Analysis in the Parameter Space

The loss associated with a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ at node v is given by $\mathcal{L}_v(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_v^\star\|^2$, with gradient $\nabla\mathcal{L}_v(\boldsymbol{\theta}) = \boldsymbol{\theta} - \boldsymbol{\theta}_v^\star$.

Proposition 3.1 (Benefit of an Intermediate Node in the Parameter Space)**.** *Let $\boldsymbol{\theta}_S$, $\boldsymbol{\theta}_I^\star$, and $\boldsymbol{\theta}_T^\star$ be the respective global minima of S, I, and T. Denote by $\boldsymbol{\theta}_T^{d(2)}$ the parameter obtained after two direct steps from S to T, and by $\boldsymbol{\theta}_T^{c(2)}$ the parameter obtained by a two-step cascade $S \to I \to T$. Then, for a fixed learning rate $\eta \in (0,1)$,*

$$\|\boldsymbol{\theta}_T^{c(2)} - \boldsymbol{\theta}_T^\star\|^2 \le \|\boldsymbol{\theta}_T^{d(2)} - \boldsymbol{\theta}_T^\star\|^2 \iff \langle \boldsymbol{\theta}_S - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\rangle \le -\frac{\eta}{2(1-\eta)}\|\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\|^2.$$

Proof. The result follows by expanding both gradient descent updates and comparing the squared distances to $\boldsymbol{\theta}_T^\star$ after simplification. □

Directional alignment. The inequality of Proposition 4.1 is governed by an inner product involving the relative positions of the (S, I, T) tasks in the parameter space. We define this quantity formally.

Definition 3.2 (Directional Alignment)**.** For three tasks (S, I, T) with optima $(\boldsymbol{\theta}_S^\star, \boldsymbol{\theta}_I^\star, \boldsymbol{\theta}_T^\star)$, the *directional alignment of I with respect to* (S,T) is expressed as:

$$\mathcal{A}(S,I,T) = \langle \boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\rangle.$$

We say that I is *favorably aligned* for transfer from S to T when $\langle \boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\rangle \le -\frac{\eta}{2(1-\eta)}\|\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\|^2$.

This alignment quantity measures the extent to which using the direction of the update $S \to I$ as an intermediate target can serve the direction of the transfer $S \to T$. A negative value of $\mathcal{A}(S,I,T)$ indicates that the displacement vector $(\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star)$ points in a direction that is compatible with moving from S toward T, but Proposition 4.1 shows that direction alone is not sufficient. For fixed $\boldsymbol{\theta}_S^\star$ and $\boldsymbol{\theta}_T^\star$, the cascade improvement condition $\langle \boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\rangle \le -\frac{\eta}{2(1-\eta)}\|\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\|^2$ characterizes the set of admissible intermediates $\boldsymbol{\theta}_I^\star$. In two dimensions, this set is a disk $\{\boldsymbol{\theta}_I^\star : \|\boldsymbol{\theta}_I^\star - \mathbf{c}(\eta)\| \le r(\eta)\}$, centered at $\mathbf{c}(\eta) = \boldsymbol{\theta}_T^\star - \frac{1-\eta}{\eta}(\boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star)$ with radius $r(\eta) = \frac{1-\eta}{\eta}\|\boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star\|$. Figure 2 illustrates this geometry.

3.2. Analysis in the Feature Space

We now consider linear models defined by parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$. Each node v is associated with a target $\boldsymbol{y}_v \in \mathbb{R}^n$ and loss $\mathcal{L}_v(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}_v\|^2$, with gradient $\nabla\mathcal{L}_v(\boldsymbol{\theta}) = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}_v)$. A gradient update takes the form

$$G_v(\boldsymbol{\theta}) = \mathbf{M}\boldsymbol{\theta} + \eta\boldsymbol{X}^\top\boldsymbol{y}_v, \qquad \mathbf{M} = \mathbf{I}_d - \eta\boldsymbol{X}^\top\boldsymbol{X},$$

where $\mathbf{I}_d$ denotes the identity matrix of size d. We first assume $\boldsymbol{y}_v = \boldsymbol{X}\boldsymbol{\theta}_v^*$ for some optimal parameters $\boldsymbol{\theta}_v^*$.

Proposition 3.3. *For any node v and $k \ge 1$, the k-th iterate of the gradient update writes*

$$G_v^k(\boldsymbol{\theta}) = \mathbf{M}^k\boldsymbol{\theta} + \eta\sum_{i=0}^{k-1}\mathbf{M}^i\boldsymbol{X}^\top\boldsymbol{y}_v.$$

Proof. The proof can be done by induction on k. □

Proposition 3.4 (General k-step Transfer)**.** *For $k = k_I + k_T$, define $\mathbf{Q}_{k_I,k_T} = \sum_{i=0}^{k_I-1}\mathbf{M}^{i+k_T}$. Then*

$$\|\boldsymbol{\theta}_T^{c(k)} - \boldsymbol{\theta}_T^\star\|^2 \le \|\boldsymbol{\theta}_T^{d(k)} - \boldsymbol{\theta}_T^\star\|^2 \iff \langle \mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}\rangle \le -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k_T}\mathbf{v}\|^2.$$

with $\boldsymbol{\Delta} = \boldsymbol{\theta}_S - \boldsymbol{\theta}_T^\star$ and $\mathbf{v} = \boldsymbol{X}^\top(\boldsymbol{y}_I - \boldsymbol{y}_T)$.

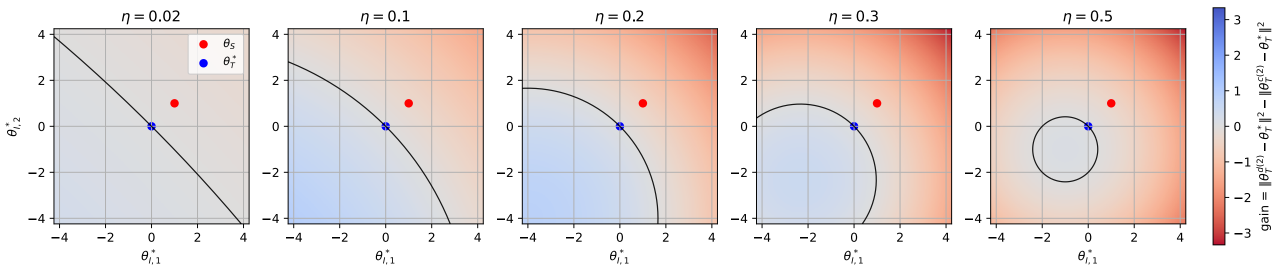


Figure 2. Feasible region in the parameter space for the intermediate optimum. The circle defines the boundary (equality case), while the interior corresponds to the area where I should be put so the cascade $S \to I \to T$ outperforms direct transfer $S \to T$.

Proposition 4.4 generalizes the directional alignment condition of the parameter space to the feature space. The cascade is beneficial when the intermediate residual $\mathbf{Q}_{k_I,k_T}\mathbf{v}$ is well-aligned with the gradient-propagated error $\mathbf{M}^k\boldsymbol{\Delta}$.

Proof. See proof in Appendix A. □

Relaxation of the Alignment Condition. The condition for alignment in the linear feature space assumes that we have knowledge of $\boldsymbol{\Delta}$, but in practice we do not have access to the parameter vector that generates the target variable. One way to relax the condition of Proposition 4.4 is to exploit the definition of the gradient of the cost function $\mathcal{L}_T$, noting that $\nabla\mathcal{L}_T(\boldsymbol{\theta}_S) = \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\Delta}$. Therefore,

$$\boldsymbol{\Delta} \approx (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\nabla\mathcal{L}_T(\boldsymbol{\theta}_S) = (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta}_S - \boldsymbol{y}_T),$$

where $\dagger$ denotes the pseudo-inverse (another way could be to use Tikhonov regularization). Remark that if $\boldsymbol{X}^\top\boldsymbol{X}$ is definite positive, then the previous approximation becomes an equality. Hence we can set $\widehat{\boldsymbol{\Delta}} = (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta}_S - \boldsymbol{y}_T)$, yielding a practical condition by substituting $\widehat{\boldsymbol{\Delta}}$ for $\boldsymbol{\Delta}$ in Proposition 4.4.

3.3. Noisy Target Setting

We now consider the noisy setting where $\boldsymbol{y}_v = \boldsymbol{X}\boldsymbol{\theta}_v^\star + \boldsymbol{\varepsilon}_v$, and $\boldsymbol{\varepsilon}_v$ denotes a random vector in $\mathbb{R}^n$ defined on some probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Each noise vector $\boldsymbol{\varepsilon}_v$ has independent mean-zero sub-Gaussian coordinates with proxy variance σ_v^2, hence $\boldsymbol{\varepsilon}_I - \boldsymbol{\varepsilon}_T$ is sub-Gaussian with proxy variance $\tau^2 = \sigma_I^2 + \sigma_T^2$.

Proposition 3.5 (General Stochastic Case)**.** *For an intermediate node and total $k = k_I + k_T$ steps, with probability at least $1 - \delta$,*

$$\langle \mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k+k_T}\mathbf{v}\rangle \leq -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k+k_T}\mathbf{v}^*\|^2 + \tau\,\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|\sqrt{2\log(1/\delta)}.$$

We write $\mathbf{v} = \mathbf{v}^\star + \delta\mathbf{v}$, where $\mathbf{v}^\star = \boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star)$ and $\delta\mathbf{v} = \boldsymbol{X}^\top(\boldsymbol{\varepsilon}_I - \boldsymbol{\varepsilon}_T)$.

Proof. See Proof in Appendix B. □

Observe that for a fixed η relatively small, $\mathbf{M}^k \approx \mathbf{I}_d$ and $\mathbf{Q}_{k_I,k+k_T} \propto \mathbf{I}_d$, hence we can re-write Proposition 4.4 as $-\langle\boldsymbol{\Delta} \mid \mathbf{v}\rangle/(\alpha\|\mathbf{v}^\star\|^2 + \beta) \geq 0$, where $\alpha, \beta > 0$ are constants derived from the approximations.

From local intermediate effects to global cascades. Section 4 analyzes when a single learning step can facilitate a subsequent one, by comparing direct transfers to short cascades involving an intermediate task. While these results are derived from local three-task configurations, they suggest a more general principle: if learning a task produces a parameter state that is closer to the solution of neighboring tasks, then chaining such learning steps can reduce the overall cost of adapting a collection of tasks. We now formalize this idea at the level of cascaded transfer over rooted trees, and derive sufficient conditions under which tree-structured CTL improves upon independent training and pairwise transfer schemes.

4. On the Benefit of Intermediate Tasks in Cascaded Transfer

Having introduced the CTL framework and its tree-based instantiation, we now analyze the conditions under which an intermediate task can improve transfer performance under a fixed refinement budget. For presentation clarity, we conduct our analysis in a linear regression setting, although the underlying principles extend more broadly. We progressively move from the model parameter space to the feature space, and finally to the noisy setting, where we derive probabilistic guarantees. These results provide theoretical insight into why structured transfer graphs such as rooted trees can be effective in CTL, while leaving open the choice of the specific cascade construction strategy.

4.1. Analysis in the Parameter Space

The loss associated with a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ at node v is given by $\mathcal{L}_v(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_v^\star\|^2$, with gradient $\nabla\mathcal{L}_v(\boldsymbol{\theta}) = \boldsymbol{\theta} - \boldsymbol{\theta}_v^\star$.

Proposition 4.1 (Benefit of an Intermediate Node in the Parameter Space)**.** *Let $\boldsymbol{\theta}_S$, $\boldsymbol{\theta}_I^\star$, and $\boldsymbol{\theta}_T^\star$ be the respective global minima of S, I, and T. Denote by $\boldsymbol{\theta}_T^{d(2)}$ the parameter obtained after two direct steps from S to T, and by $\boldsymbol{\theta}_T^{c(2)}$ the parameter obtained by a two-step cascade $S \to I \to T$. Then, for a fixed learning rate $\eta \in (0,1)$,*

$$\|\boldsymbol{\theta}_T^{c(2)} - \boldsymbol{\theta}_T^\star\|^2 \leq \|\boldsymbol{\theta}_T^{d(2)} - \boldsymbol{\theta}_T^\star\|^2 \iff \langle \boldsymbol{\theta}_S - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star \rangle \leq -\frac{\eta}{2(1-\eta)}\|\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\|^2.$$

Proof. The result follows by expanding both gradient descent updates and comparing the squared distances to $\boldsymbol{\theta}_T^\star$ after simplification. □

Directional alignment. The inequality of Proposition 4.1 is governed by an inner product involving the relative positions of the (S, I, T) tasks in the parameter space. We define this quantity formally.

Definition 4.2 (Directional Alignment)**.** For three tasks (S, I, T) with optima $(\boldsymbol{\theta}_S^\star, \boldsymbol{\theta}_I^\star, \boldsymbol{\theta}_T^\star)$, the *directional alignment of I with respect to (S,T)* is expressed as:

$$\mathcal{A}(S, I, T) = \langle \boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star \rangle.$$

We say that I is *favorably aligned* for transfer from S to T when $\langle \boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star \rangle \leq -\frac{\eta}{2(1-\eta)}\|\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\|^2$.

This alignment quantity measures the extent to which using the direction of the update $S \to I$ as an intermediate target can serve the direction of the transfer $S \to T$. A negative value of $\mathcal{A}(S, I, T)$ indicates that the displacement vector $(\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star)$ points in a direction that is compatible with moving from S toward T, but Proposition 4.1 shows that direction alone is not sufficient. For fixed $\boldsymbol{\theta}_S^\star$ and $\boldsymbol{\theta}_T^\star$, the cascade improvement condition $\langle \boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star \mid \boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star \rangle \leq -\frac{\eta}{2(1-\eta)}\|\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star\|^2$ characterizes the set of admissible intermediates $\boldsymbol{\theta}_I^\star$. In two dimensions, this set is a disk $\{\boldsymbol{\theta}_I^\star : \|\boldsymbol{\theta}_I^\star - \mathbf{c}(\eta)\| \leq r(\eta)\}$, centered at $\mathbf{c}(\eta) = \boldsymbol{\theta}_T^\star - \frac{1-\eta}{\eta}(\boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star)$ with radius $r(\eta) = \frac{1-\eta}{\eta}\|\boldsymbol{\theta}_S^\star - \boldsymbol{\theta}_T^\star\|$. Figure 2 illustrates this geometry.

4.2. Analysis in the Feature Space

We now consider linear models defined by parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. Each node v is associated with a target $\boldsymbol{y}_v \in \mathbb{R}^n$ and loss $\mathcal{L}_v(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}_v\|^2$, with gradient $\nabla\mathcal{L}_v(\boldsymbol{\theta}) = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}_v)$. A gradient update takes the form

$$G_v(\boldsymbol{\theta}) = \mathbf{M}\boldsymbol{\theta} + \eta\boldsymbol{X}^\top\boldsymbol{y}_v, \qquad \mathbf{M} = \mathbf{I}_d - \eta\boldsymbol{X}^\top\boldsymbol{X},$$

where $\mathbf{I}_d$ denotes the identity matrix of size d. We first assume $\boldsymbol{y}_v = \boldsymbol{X}\boldsymbol{\theta}_v^*$ for some optimal parameters $\boldsymbol{\theta}_v^*$.

Proposition 4.3. *For any node v and $k \geq 1$, the k-th iterate of the gradient update writes*

$$G_v^k(\boldsymbol{\theta}) = \mathbf{M}^k\boldsymbol{\theta} + \eta\sum_{i=0}^{k-1}\mathbf{M}^i\boldsymbol{X}^\top\boldsymbol{y}_v.$$

Proof. The proof can be done by induction on k. □

Proposition 4.4 (General k-step Transfer)**.** *For $k = k_I + k_T$, define $\mathbf{Q}_{k_I,k_T} = \sum_{i=0}^{k_I-1}\mathbf{M}^{i+k_T}$. Then*

$$\|\boldsymbol{\theta}_T^{c(k)} - \boldsymbol{\theta}_T^\star\|^2 \leq \|\boldsymbol{\theta}_T^{d(k)} - \boldsymbol{\theta}_T^\star\|^2 \iff \langle \mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v} \rangle \leq -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k_T}\mathbf{v}\|^2.$$

with $\boldsymbol{\Delta} = \boldsymbol{\theta}_S - \boldsymbol{\theta}_T^\star$ and $\mathbf{v} = \boldsymbol{X}^\top(\boldsymbol{y}_I - \boldsymbol{y}_T)$.

Proposition 4.4 generalizes the directional alignment condition of the parameter space to the feature space. The cascade is beneficial when the intermediate residual $\mathbf{Q}_{k_I,k_T}\mathbf{v}$ is well-aligned with the gradient-propagated error $\mathbf{M}^k\boldsymbol{\Delta}$.

Proof. See proof in Appendix A. □

Relaxation of the Alignment Condition. The condition for alignment in the linear feature space assumes that we have knowledge of $\boldsymbol{\Delta}$, but in practice we do not have access to the parameter vector that generates the target variable. One way to relax the condition of Proposition 4.4 is to exploit the definition of the gradient of the cost function $\mathcal{L}_T$, noting that $\nabla\mathcal{L}_T(\boldsymbol{\theta}_S) = \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\Delta}$. Therefore,

$$\boldsymbol{\Delta} \approx (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\nabla\mathcal{L}_T(\boldsymbol{\theta}_S) = (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta}_S - \boldsymbol{y}_T),$$

where † denotes the pseudo-inverse (another way could be to use Tikhonov regularization). Remark that if $\boldsymbol{X}^\top\boldsymbol{X}$ is definite positive, then the previous approximation becomes an equality. Hence we can set $\widehat{\boldsymbol{\Delta}} = (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta}_S - \boldsymbol{y}_T)$, yielding a practical condition by substituting $\widehat{\boldsymbol{\Delta}}$ for $\boldsymbol{\Delta}$ in Proposition 4.4.

4.3. Noisy Target Setting

We now consider the noisy setting where $\boldsymbol{y}_v = \boldsymbol{X}\boldsymbol{\theta}_v^\star + \boldsymbol{\varepsilon}_v$, and $\boldsymbol{\varepsilon}_v$ denotes a random vector in $\mathbb{R}^n$ defined on some probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Each noise vector $\boldsymbol{\varepsilon}_v$ has independent mean-zero sub-Gaussian coordinates with proxy variance σ_v^2, hence $\boldsymbol{\varepsilon}_I - \boldsymbol{\varepsilon}_T$ is sub-Gaussian with proxy variance $\tau^2 = \sigma_I^2 + \sigma_T^2$.

Proposition 4.5 (General Stochastic Case)**.** *For an intermediate node and total $k = k_I + k_T$ steps, with probability at least $1 - \delta$,*

$$\langle \mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k+k_T}\mathbf{v} \rangle \leq -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k+k_T}\mathbf{v}^*\|^2 + \tau\,\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|\sqrt{2\log(1/\delta)}.$$

We write $\mathbf{v} = \mathbf{v}^\star + \delta\mathbf{v}$*, where* $\mathbf{v}^\star = \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star)$ *and* $\delta\mathbf{v} = \boldsymbol{X}^\top(\boldsymbol{\varepsilon}_I - \boldsymbol{\varepsilon}_T)$.

Proof. See Proof in Appendix B. □

Observe that for a fixed η relatively small, $\mathbf{M}^k \approx \mathbf{I}_d$ and $\mathbf{Q}_{k_I,k+k_T} \propto \mathbf{I}_d$, hence we can re-write Proposition 4.4 as $-\langle \boldsymbol{\Delta} \mid \mathbf{v}\rangle / (\alpha\|\mathbf{v}^\star\|^2 + \beta) \geq 0$, where $\alpha, \beta > 0$ are constants derived from the approximations.

4.4. Computational Complexity

The practical algorithm restricts candidate sources and intermediates to K-nearest-neighbor neighborhoods, so that each iteration (introducing one new target) evaluates at most $\mathcal{O}(K^2)$ triplets. With $|\mathcal{V}|$ iterations and observation dimension d, the overall complexity is $\mathcal{O}(|\mathcal{V}|K^2 d)$, since all triplet scores for a fixed target are computed in a single batched matrix multiplication. Tree updates (edge additions or local insertions) require only constant-time operations and do not affect the asymptotic cost. Compared with MST-based trees, which require $\mathcal{O}(|\mathcal{V}|^2 \log |\mathcal{V}|)$ time, the alignment-based construction scales linearly in $|\mathcal{V}|$ for fixed K, making it suitable for many-task regimes.

5. Experiments

We evaluate the proposed cascading algorithm on a controlled synthetic benchmark and compare it with a series of baseline transfer schemes. This section details the construction of the synthetic tasks, the set of competing methods, and the experimental protocol used for training and evaluation.

5.1. Datasets

Synthetic Dataset. We construct families of linear regression tasks of the form $\boldsymbol{y}_v = \boldsymbol{X}_v\boldsymbol{\theta}_v + \boldsymbol{\varepsilon}_v$, where $\boldsymbol{X}_v \in \mathbb{R}^{n\times d}$ has i.i.d. standard Gaussian rows and $\boldsymbol{\varepsilon}_v \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_n)$. To control task relatedness, we consider two parameter generative models. In the *homogeneous perturbation* model, a global vector $\boldsymbol{\theta}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ is sampled once and each task parameter is set to $\boldsymbol{\theta}_v = \boldsymbol{\theta}_0 + \boldsymbol{\xi}_v$ with $\boldsymbol{\xi}_v \sim \mathcal{N}(\mathbf{0}, \tau^2\mathbf{I}_d)$, so that τ determines the dispersion of tasks around a shared latent component, see Figure 3a. In the *clustered* model, we first draw a global center $\boldsymbol{\theta}_0$ and K cluster shifts $\delta_c \sim \mathcal{N}(\mathbf{0}, \tau^2_{\text{between}}\mathbf{I}_d)$, every task v is assigned a cluster $c(v)$ and its parameter is generated as $\boldsymbol{\theta}_v = \boldsymbol{\theta}_0 + \delta_{c(v)} + \boldsymbol{\zeta}_v$ with $\boldsymbol{\zeta}_v \sim \mathcal{N}(\mathbf{0}, \tau^2_{\text{within}}\mathbf{I}_d)$. Varying τ or $(\tau_{\text{within}}, \tau_{\text{between}})$ controls the degree and structure of heterogeneity, enabling systematic evaluation across regimes ranging from nearly homogeneous tasks to well-separated clusters, see Figure 3b.

Real Dataset. The UK dataset[1] provides aggregated half-hourly residential electricity demand collected from smart meters by two major UK Distribution Network Operators (SSEN and NGED). Together, they cover roughly 2 million meters, 120,000 low-voltage feeders, and 50,000 substations, with data available from early 2024 onward. For our experiments, we focus on a subset of 100 nodes in the Oxford urban area. We use calendar-based features (e.g., day of week, time of day), lagged load values, and the substation-level demand as the prediction target.

[1] https://weave.energy/

5.2. Experimental Settings

For regression experiments on both synthetic and real datasets, we divide the data into two sets: a training set and a test set. For the synthetic dataset, we use subsets with sizes of $n_{train} = 32$ and $n_{test} = 128$. For the real dataset, the training set ranges from February 14, 2024, at $12:00$ a.m. to February 24, 2024, at $11:30$ p.m., and the test set ranges from February 25, 2024, at $12:00$ a.m. to February 28, 2024, at $11:30$ p.m.

5.3. Baselines

We compare our approach with several tree-structured or non-transfer baselines. All methods use identical training routines $\mathcal{A}_0$ (seed training) and $\mathcal{A}_1$ (local refinement).

– *Individual models.* For each task, an individual model is trained independently for a fixed computational budget. This serves as a baseline that uses no multi-task or transfer learning.

– *Star tree.* A seed task s is selected, and then a star graph is constructed: $(s \to v)$, for all v. Each node receives the seed parameters and performs independent refinement.

– *Random tree.* A seed task s is selected, and a random spanning tree over all tasks is generated by sampling a Prüfer sequence uniformly, which yields a labeled tree drawn uniformly from the set of all spanning trees. The tree is then rooted at s by orienting all edges away from it. This baseline represents an uninformed hierarchical transfer structure that does not exploit any task-related information.

– *Minimum-Spanning Tree (MST).* An MST rooted at the seed task is built, using Prim's algorithm over mean feature vectors of each task. This baseline captures coarse geometric structure of the tasks.

– *Alignment Tree.* The practical cascade uses the data-driven scoring function introduced in Equation (**??**), where alignment between tasks is inferred from empirical discrepancies rather than true parameters. This version constructs the diffusion tree using only observable quantities such as $(\boldsymbol{X}_v, \boldsymbol{y}_v)$. It represents the fully implementable algorithm and is the main method evaluated in the experiments.

– *Alignment Tree Oracle.* This variant corresponds to an idealized setting in which the alignment score is computed using the true latent task parameters. Because the improvement condition can be evaluated without estimation noise, the resulting diffusion tree reflects the structure that would be obtained if perfect information about task relationships were available. This oracle construction is used to assess how closely the practical, data-driven cascade can approximate the behavior of an alignment-based method operating under full knowledge of the underlying task geometry.

5.4. Evaluation Protocol

For every configuration of $(\tau, T, n_{\mathrm{train}}, n_{\mathrm{test}}, B)$, we generate multiple independent datasets and evaluate each baseline on identical task families. All methods receive the same global refinement budget B and an initial seed budget b_{seed}. For a given method, the output of the cascade is a parameter estimate $\hat{\boldsymbol{\theta}}_v$ for each task v. Performance is measured through the per-task test mean squared error, and we report both its empirical distribution across tasks and its empirical cumulative distribution function (CDF). Repeating the entire procedure across 50 random seeds allows us to assess stability with respect to dataset realization and to initialization of the tree. This protocol provides a consistent and fine-grained comparison of: (i) no-transfer learning (INDIV), (ii) shallow structure (STAR), and (iii) CTL methods (CTL-RANDTREE, CTL-MST, CTL-ALIGNTREE) as well as its oracle counterpart (CTL-ALIGNTREEORACLE). All reported results share identical data, budget, and optimization conditions, isolating the effect of the transfer structure itself.

5.5. Results

Figure **??** summarizes the distribution of test MSE across tasks, together with the corresponding empirical CDFs, for the homogeneous setting at heterogeneity level $\tau = 1.0$. Individual training (INDIV) yields the highest and most variable errors, with wide dispersion and a substantial upper tail, illustrating the difficulty of learning each task independently under limited per-task data. Introducing transfer through a star topology (STAR) provides a clear improvement, but its error distribution remains noticeably broader than that of the tree-based methods.

All CTL variants achieve markedly lower MSEs: the interquartile ranges contract substantially, median errors decrease by more than a factor of two relative to STAR, and outliers become rare. The CDFs make this contrast explicit: for each CTL method, the curve shifts sharply to the left, indicating that a large majority of tasks benefit from the shared refinement process. Among these, alignment-based transfer (CTL-ALIGNTREE) yields the most favorable distribution, closely tracking the performance of the oracle construction (CTL-ALIGNTREEORACLE) while relying only on data-derived alignment signals. Geometry-based transfer (CTL-MST) and uninformed random hierarchies (CTL-RANDTREE) also improve substantially over STAR, though their CDFs exhibit slightly heavier tails.

Overall, the results show that tree-structured cascaded transfer provides robust gains over both no-transfer and shallow-transfer baselines, and that incorporating alignment information into the tree construction consistently leads to the strongest and most reliable improvements.

6. Conclusion and Perspectives

This work introduced a general framework for budget-constrained many-task learning based on CTL. CTL views knowledge sharing as a structured diffusion of information across tasks organized by a rooted acyclic graph, where each task aggregates information from its parents and performs a single refinement step under a local budget. Within this general abstraction, we analyzed the fundamental question of when a transfer through an intermediate task can outperform direct transfer. We provided precise alignment conditions in parameter space, feature space, and noisy settings under which such improvements provably arise.

Building on these insights, we instantiated CTL with an alignment-guided tree construction that selects informative intermediate tasks and yields a practical hierarchical diffusion mechanism. Experiments on synthetic and real data demonstrate that this instantiation leads to substantial accuracy and efficiency gains compared to standard fine-tuning and geometric baselines.

Several research directions follow from this study. First, extending the theoretical analysis beyond quadratic losses and linear models to nonlinear predictors and deep representations would deepen our understanding of CTL in modern architectures. Second, the general DAG formulation of CTL opens the door to richer transfer structures, where tasks may fuse information from multiple parents, developing principled methods for constructing or learning such graphs is an important challenge. Third, optimizing budget allocation over the cascade, potentially through adaptive or meta-learned strategies, may further enhance performance under stringent computational constraints. Finally, applying CTL at scale to complex task systems, such as large sensor networks, personalized models, or distributed forecasting infrastructures, offers a promising avenue for assessing robustness and scalability in real-world environments.

Overall, CTL provides a flexible and computationally efficient paradigm for structured knowledge transfer across large collections of tasks. We believe it establishes a principled foundation for future work at the intersection of many-task learning, transfer learning, and resource-constrained model adaptation.

References

Aribandi, V., Tay, Y., Schuster, T., Rao, J., Zheng, H. S., Mehta, S. V., Zhuang, H., Tran, V. Q., Bahri, D., Ni, J., et al. ExT5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*, 2022.

Hashimoto, K., Xiong, C., Tsuruoka, Y., and Socher, R. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1923–1933, 2017.

He, X., Alesiani, F., and Shaker, A. Efficient and scalable multi-task regression on massive number of tasks. In *AAAI Conference on Artificial Intelligence*, volume 33, pp. 3763–3770, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada, 2009.

Liu, J., Xia, Z., Lei, Y., Li, X., and Wang, X. Multi-faceted hierarchical multi-task learning for a large number of tasks with multi-dimensional relations. *Preprint arXiv:2110.13365*, 2021.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers, 2022.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22 (10):1345–1359, 2009.

Ruiz, C., Alaíz, C. M., and Dorronsoro, J. R. A survey on kernel-based multi-task learning. *Neurocomputing*, 577: 127–255, 2024.

Strezoski, G., Noord, N. v., and Worring, M. Many task learning with task routing. In *IEEE/CVF International Conference on Computer Vision*, pp. 1375–1384, 2019.

Tan, Y., Zhang, E., Li, Y., Huang, S.-L., and Zhang, X.-P. Transferability-guided cross-domain cross-task transfer learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers, 2024.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

Zhang, Y. and Yang, Q. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A. Proof of Proposition 4.4

For $k = k_I + k_T$, define $\mathbf{Q}_{k_I,k_T} = \sum_{i=0}^{k_I-1} \mathbf{M}^{i+k_T}$. Then

$$\|\boldsymbol{\theta}_T^{c(k)} - \boldsymbol{\theta}_T^\star\|^2 \leq \|\boldsymbol{\theta}_T^{d(k)} - \boldsymbol{\theta}_T^\star\|^2 \iff \langle \mathbf{M}^k \boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}\rangle \leq -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k_T}\mathbf{v}\|^2.$$

Proof. Starting from $\boldsymbol{\theta}_S$ and applying k steps on node T yields $\boldsymbol{\theta}_T^{d(k)} = G_T^k(\boldsymbol{\theta}_S)$. Using Proposition 4.3 and the fact that $G_T^k(\boldsymbol{\theta}_T^\star) = \boldsymbol{\theta}_T^\star$, we obtain

$$\boldsymbol{\theta}_T^{d(k)} - \boldsymbol{\theta}_T^\star = \mathbf{M}^k(\boldsymbol{\theta}_S - \boldsymbol{\theta}_T^\star) = \mathbf{M}^k \boldsymbol{\Delta}.$$

The intermediate update gives $G_I^{k_I}(\boldsymbol{\theta}_S) = \mathbf{M}^{k_I}\boldsymbol{\theta}_S + \eta\sum_{i=0}^{k_I-1}\mathbf{M}^i \boldsymbol{X}^\top \boldsymbol{y}_I$. Then, applying k_T steps on node T yields $\boldsymbol{\theta}_T^{c(k)} = G_T^{k_T}\big(G_I^{k_I}(\boldsymbol{\theta}_S)\big) = \mathbf{M}^k\boldsymbol{\theta}_S + \eta\sum_{i=0}^{k_I-1}\mathbf{M}^{i+k_T}\boldsymbol{X}^\top\boldsymbol{y}_I + \eta\sum_{i=0}^{k_T-1}\mathbf{M}^i\boldsymbol{X}^\top\boldsymbol{y}_T$. Subtracting $\boldsymbol{\theta}_T^\star$ and again using $G_T^k(\boldsymbol{\theta}_T^\star) = \boldsymbol{\theta}_T^\star$, we get

$$\boldsymbol{\theta}_T^{c(k)} - \boldsymbol{\theta}_T^\star = \mathbf{M}^k\boldsymbol{\Delta} + \eta\sum_{i=0}^{k_I-1}\mathbf{M}^{i+k_T}\boldsymbol{X}^\top(\boldsymbol{y}_I - \boldsymbol{y}_T).$$

Define $\mathbf{Q}_{k_I,k_T} = \sum_{i=0}^{k_I-1}\mathbf{M}^{i+k_T}$, and $\mathbf{v} = \boldsymbol{X}^\top(\boldsymbol{y}_I - \boldsymbol{y}_T)$, so that $\boldsymbol{\theta}_T^{c(k)} - \boldsymbol{\theta}_T^\star = \mathbf{M}^k\boldsymbol{\Delta} + \eta\,\mathbf{Q}_{k_I,k_T}\mathbf{v}$.

We compare squared distances:

$$\begin{aligned}\|\mathbf{M}^k\boldsymbol{\Delta} + \eta\mathbf{Q}_{k_I,k_T}\mathbf{v}\|^2 \leq \|\mathbf{M}^k\boldsymbol{\Delta}\|^2 &\iff 2\eta\langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}\rangle + \eta^2\|\mathbf{Q}_{k_I,k_T}\mathbf{v}\|^2 \leq 0\\ &\iff \langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}\rangle \leq -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k_T}\mathbf{v}\|^2.\end{aligned}$$

□

B. Proof of Proposition 4.5

For total $k = k_I + k_T$ steps, with probability at least $1-\delta$,

$$\langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}\rangle \leq -\frac{\eta}{2}\|\mathbf{Q}_{k_I,k_T}\mathbf{v}^\star\|^2 + \tau\,\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|\sqrt{2\log(1/\delta)}.$$

We write $\mathbf{v} = \mathbf{v}^\star + \delta\mathbf{v}$, where $\mathbf{v}^\star = \boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{\theta}_I^\star - \boldsymbol{\theta}_T^\star)$ and $\delta\mathbf{v} = \boldsymbol{X}^\top(\boldsymbol{\varepsilon}_I - \boldsymbol{\varepsilon}_T)$.

Proof. Let us first decompose the inner product as follows:

$$\langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}\rangle = \langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\mathbf{v}^\star\rangle + \langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\delta\mathbf{v}\rangle$$

With Proposition 4.4, we can upper-bound the first term by $-\frac{\eta}{2}\|\mathbf{Q}_{k_I,k_T}\mathbf{v}^\star\|^2$. The second term can be re-written as $\langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\delta\mathbf{v}\rangle = (\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta})^\top(\boldsymbol{\varepsilon}_I - \boldsymbol{\varepsilon}_T)$, which is a sub-Gaussian random variable of variance proxy $\tau^2\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|^2$. With a Chernoff bound, we get for $t \geq 0$:

$$\mathbb{P}\left(\langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\delta\mathbf{v}\rangle \geq t\right) \leq \exp\left(-\frac{t^2}{2\tau^2\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|^2}\right).$$

Then, taking $t = \tau\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|\sqrt{2\log 1/\delta}$ yields with probability greater than $1-\delta$,

$$\langle\mathbf{M}^k\boldsymbol{\Delta} \mid \mathbf{Q}_{k_I,k_T}\delta\mathbf{v}\rangle \leq \tau\|\boldsymbol{X}\mathbf{Q}_{k_I,k+k_T}\boldsymbol{\Delta}\|\sqrt{2\log 1/\delta}.$$

□

C. Construction of the Alignment-based Tree

We describe here the construction of the alignment-based CTL diffusion tree used throughout the experiments. The procedure incrementally builds a directed tree over tasks by repeatedly selecting the most favorable (S, I, T) triplet according to the alignment score derived in Section 4. The tree is initialized from a single seed task and expanded until all tasks are inserted.

At each iteration, the algorithm considers uncovered target tasks T and searches for a source task S already in the tree, together with a potential intermediate task I, such that cascading transfer from S to T through I yields the largest predicted improvement. To ensure scalability, the search is restricted to local neighborhoods defined by a K-nearest-neighbor (KNN) graph, and all computations are carried out using precomputed dot products and norms of proxy vectors rather than explicit displacement vectors.

The alignment score for a candidate triplet (S, I, T) is then given by $\gamma_{S,I,T} = \dfrac{-\langle \mathbf{D}_S \mid \mathbf{D}_I \rangle}{\alpha \|\mathbf{D}_I\|^2 + \beta}$, with $\alpha = \dfrac{\eta}{2(1-\eta)}$, which can be evaluated efficiently from precomputed dot products and norms.

Candidate restriction via KNN. For a given target T, candidate sources are first restricted to $\mathcal{S}_T = \mathcal{T} \cap \mathcal{N}(T)$, where $\mathcal{T}$ denotes the current set of tasks already in the tree and $\mathcal{N}(T)$ is the K-nearest-neighbor set of T. If $\mathcal{S}_T$ is empty, all tree nodes are considered as potential sources. Candidate intermediates are initially taken as $\mathcal{I}_{S,T} = \mathcal{N}(T) \setminus \{S, T\}$.

Additional pruning rules are then applied to $\mathcal{I}_{S,T}$ to reduce the search space.

Pruning strategies. We consider three pruning strategies, trading off theoretical faithfulness and practical robustness.

Hyperplane pruning. This baseline strategy enforces directional consistency using pairwise distances: $\mathcal{I}_{S,T} \leftarrow \big\{ I \in \mathcal{I}_{S,T} : d(I,S) > d(S,T) \ \wedge \ d(I,T) < d(S,T) \big\}$, where $d(u,v) = \|\boldsymbol{y}_u - \boldsymbol{y}_v\|$. This heuristic favors intermediates that lie "behind" T with respect to S in the features space.

Raw alignment ball pruning. From the theoretical analysis, the cascade improvement condition characterizes admissible intermediates through the inequality $\langle \mathbf{D}_S \mid \mathbf{D}_I \rangle \le -\alpha \|\mathbf{D}_I\|^2$, which corresponds geometrically to a ball in parameter space. When operating on proxy vectors, we relax this condition using a nonnegative relaxation parameter κ: $\mathcal{I}_{S,T} \leftarrow \Big\{ I \in \mathcal{I}_{S,T} : \langle \mathbf{D}_S \mid \mathbf{D}_I \rangle \le -(1-\kappa)\alpha \|\mathbf{D}_I\|^2 \Big\}$. For $\kappa = 0$, this recovers the exact theoretical condition, $\kappa > 0$ enlarges the admissible set to account for proxy mismatch. When applied to noisy feature-space proxies, the hard alignment ball constraint becomes overly permissive and fails to reduce the candidate set. This motivates soft, ranking-based approximations of the alignment geometry.

Soft alignment ball pruning. To obtain a more robust alternative, we introduce a soft version of the alignment ball constraint. Instead of enforcing feasibility, we rank candidates by their *alignment margin* $m_{I,S,T} = \langle \mathbf{D}_S \mid \mathbf{D}_I \rangle + \alpha \|\mathbf{D}_I\|^2$, which measures the degree of violation of the alignment condition. For each (S, T) pair, we retain only a fixed fraction $q \in (0, 1]$ of candidates in $\mathcal{I}_{S,T}$ with the smallest margins, and discard the rest. This soft pruning favors intermediates that are closest to satisfying the theoretical alignment condition, while avoiding degenerate pruning when proxies are noisy or high-dimensional.

Among all evaluated triplets, the one with the highest alignment score is selected. If the target task is not yet in the tree, it is added either directly or via the selected intermediate; if the target is already present, a local insertion may be performed. This process is repeated until all tasks are covered.

Algorithm 2 summarizes the full procedure.

Algorithm 2 Alignment–based CTL Diffusion Tree

```
1: Input: task set $\mathcal{V}$, target vectors $\{\boldsymbol{y}_v\}_{v\in\mathcal{V}}$, learning rate $\eta$, KNN neighborhoods $\{\mathcal{N}(v)\}_{v\in\mathcal{V}}$, pruning method with mode
   mode $\in \{\textsc{Hyperplane}, \textsc{BallRaw}, \textsc{BallSoft}\}$, relaxation parameter $\kappa \geq 0$.
2: Precompute:
   – dot-product matrix $\mathrm{dot}[u,v] = \langle \boldsymbol{y}_u \mid \boldsymbol{y}_v \rangle$,
   – squared norms $\mathrm{norm}[v] = \mathrm{dot}[v,v]$,
   – distances $d(u,v) = \|\boldsymbol{y}_u - \boldsymbol{y}_v\|$,
   – alignment constant $\alpha = \eta/(2(1-\eta))$.
3: Seed selection: $s \leftarrow \arg\min_{v\in\mathcal{V}} \sum_{w\in\mathcal{V}} \|\boldsymbol{y}_v - \boldsymbol{y}_w\|$
4: $\mathcal{T} \leftarrow \{s\}, \quad \mathcal{U} \leftarrow \mathcal{V} \setminus \{s\}, \quad E \leftarrow \emptyset$
5: while $\mathcal{U} \neq \emptyset$ do
6:   $(S^\star, I^\star, T^\star, \gamma^\star) \leftarrow (\emptyset, \emptyset, \emptyset, -\infty)$
7:   for $T \in \mathcal{U}$ do
8:     Candidate sources: $\mathcal{S}_T \leftarrow \mathcal{T} \cap \mathcal{N}(T)$
9:     if $\mathcal{S}_T = \emptyset$ then
10:      $\mathcal{S}_T \leftarrow \mathcal{T}$
11:    end if
12:    Candidate intermediates: $\mathcal{I}_{S,T} \leftarrow \mathcal{N}(T) \setminus \{T, S\}$
13:    Define proxy differences $\boldsymbol{d}_v^{(T)} := \boldsymbol{y}_v - \boldsymbol{y}_T$.
14:    Pruning: $\mathcal{I}_{S,T} \leftarrow$ pruning($\mathcal{I}_{S,T}$, mode, $d$, $\boldsymbol{d}^{(T)}$)
15:    Evaluate all alignment scores $\gamma_{S,I,T}$ for $(S,I) \in \mathcal{S}_T \times \mathcal{I}_{S,T}$:
           $$\gamma_{S,I,T} = \frac{-\big(\mathrm{dot}[I,S] - \mathrm{dot}[I,T] - \mathrm{dot}[S,T] + \mathrm{norm}[T]\big)}{\alpha\big(\mathrm{norm}[I] - 2\mathrm{dot}[I,T] + \mathrm{norm}[T]\big) + \varepsilon}$$
16:    Let $(S_T, I_T, \gamma_T)$ be the maximizer for target $T$
17:    if $\gamma_T > \gamma^\star$ then
18:      $(S^\star, I^\star, T^\star, \gamma^\star) \leftarrow (S_T, I_T, T, \gamma_T)$
19:    end if
20:  end for
21:  if $T^\star \in \mathcal{T}$ then
22:    ■ Case 1: insertion
23:    if $I^\star \notin \mathcal{T}$ and $\mathrm{parent}[T^\star] = S^\star$ then
24:      Remove edge $(S^\star, T^\star)$
25:      Add edges $(S^\star, I^\star)$ and $(I^\star, T^\star)$
26:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{I^\star\}$
27:    end if
28:  else
29:    ■ Case 2: extension
30:    if $I^\star \notin \mathcal{T}$ then
31:      Add edges $(S^\star, I^\star)$ and $(I^\star, T^\star)$
32:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{I^\star, T^\star\}$
33:    else
34:      Add edge $(I^\star, T^\star)$
35:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{T^\star\}$
36:    end if
37:  end if
38:  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{T^\star\}$
39: end while
40: return $(\mathcal{T}, E)$.
```

D. Cascaded Adaptation Algorithm

Algorithm 3 implements a hierarchical adaptation scheme under a global optimization budget. Tasks are optionally partitioned into clusters to prevent negative transfer across distant regimes, after which an alignment-based diffusion tree is constructed independently within each cluster. Within a cluster, a single seed task is trained from scratch, and all remaining tasks are adapted exactly once from their parent in the tree.

The adaptation proceeds in a top, down breadth-first manner, ensuring that parent models are always adapted before their children. This enforces a strictly acyclic diffusion of information and avoids backward or joint updates, which is consistent with the cascade improvement analysis developed in earlier sections. The training operators $\mathcal{A}_0$ and $\mathcal{A}_1$ are intentionally abstracted to allow instantiations with linear or nonlinear models, provided that adaptation steps remain local.

A fixed adaptation budget B is allocated across tasks in the cluster using normalized tree-based weights. While the precise choice of weighting scheme is flexible, it reflects the intuition that tasks deeper in the tree or further from the seed may require additional adaptation capacity. In practice, rounding effects may introduce a small budget slack, which is negligible relative to B and does not affect the qualitative behavior of the algorithm.

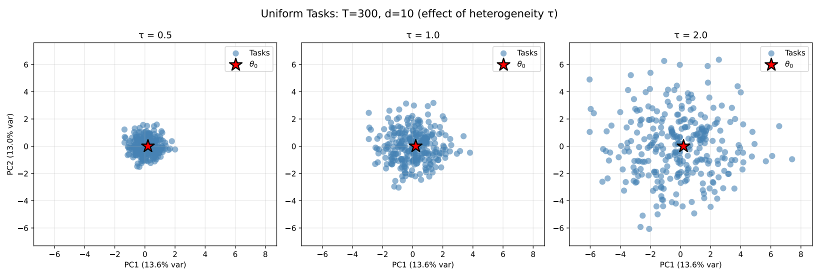
Overall, the procedure emphasizes budget-awareness, stability, and modularity, and serves as a practical realization of the alignment-based cascade transfer framework analyzed in this work.

Algorithm 3 CTL Adaptation

```
1: Input: task observations $\{\boldsymbol{y}_v\}_{v\in\mathcal{V}}$, optional cluster count $n_{clust}$, clustering routine findClusters(), seed selection routine
   findSeed(), seed-training budget $b$, per-cluster adaptation budget $B$, training operators $\mathcal{A}_0$ (seed training) and $\mathcal{A}_1$ (task adaptation).
2: Output: adapted models $\{f_{\widetilde{\boldsymbol{\theta}}_v}\}_{v\in\mathcal{V}}$
3: ■ (Optional) Cluster the tasks.
4: if $K$ is specified then
5:     $\{C_k\}_{k=1}^{n_{clust}} \leftarrow$ findClusters($\{\boldsymbol{y}_v\}_{v\in\mathcal{V}}, n_{clust}$)
6: else
7:     $K \leftarrow 1;\quad C_1 \leftarrow \mathcal{V}$
8: end if
9: ■ Process each cluster independently
10: for each cluster $C$ do
11:     $s \leftarrow$ findSeed($C$)                                        ▷ Select cluster seed
12:     $(\mathcal{T}, E) \leftarrow$ computeTree($C, s$)                    ▷ Alignment-guided tree over $C$
13:     ■ Allocate adaptation budgets along the tree
14:     $\mathbf{w} \leftarrow$ extractTreeWeights($\mathcal{T}, E$)
15:     $\mathbf{w}' \leftarrow$ normalize($\mathbf{w}$)
16:     for $v \in C$ do
17:         $b_v \leftarrow \lceil w'_v \cdot B \rceil$
18:     end for
19:     ■ Train the seed model
20:     $f_{\widetilde{\boldsymbol{\theta}}_s} \leftarrow \mathcal{A}_0(f_{\boldsymbol{\theta}}, \boldsymbol{y}_s, b)$
21:     ■ Hierarchical model adaptation
22:     Initialize queue $\mathcal{Q} \leftarrow [\,s\,]$
23:     while $\neg$ isEmpty($\mathcal{Q}$) do
24:         $u \leftarrow$ dequeue($\mathcal{Q}$)
25:         for each child $v$ of $u$ in $(\mathcal{T}, E)$ do
26:             $\mathcal{Q} \leftarrow$ enqueue($\mathcal{Q}, v$)
27:             $f_{\widetilde{\boldsymbol{\theta}}_v} \leftarrow \mathcal{A}_1(f_{\widetilde{\boldsymbol{\theta}}_u}, \boldsymbol{y}_v, b_v)$      ▷ Adapt task $v$ from its parent's model
28:         end for
29:     end while
30: end for
31: return $\{f_{\widetilde{\boldsymbol{\theta}}_v}\}_{v\in\mathcal{V}}$.
```

E. Extended Results on the Synthetic Dataset

Tables 1 and 2 report the test MSE obtained across a systematic sweep of heterogeneity levels τ, number of tasks T, and global refinement budgets B, evaluated under both homogeneous and clustered synthetic regimes (see Figure 4 for cluster results). Each row corresponds to an identical configuration applied to the two generative settings, enabling direct comparison of transfer behaviors as structure varies. Among the six baselines, the individual model (INDIV) provides a no-transfer reference, whereas STAR, CTL-RANDTREE, and CTL-MST impose shallow or geometry-driven hierarchies. The alignment-based variants (CTL-ALIGNTREE and the oracle CTL-ALIGNTREEORACLE) consistently achieve the lowest errors. Overall, CTL provides substantial gains over both no-transfer and uninformed baselines, and these gains amplify in the clustered setting or under moderate-to-high heterogeneity, where task relationships become more structured.



figures/clustered_tasks_2d_plot.pdf

(a) Homogeneous tasks.

(b) Clustered tasks.

Figure 3. Two-dimensional PCA projection of task parameters under the homogeneous and clustered generative models. The homogeneous setting (left) illustrates the effect of increasing heterogeneity τ, while the clustered setting (right) reveals structured groups induced by shared latent class shifts.

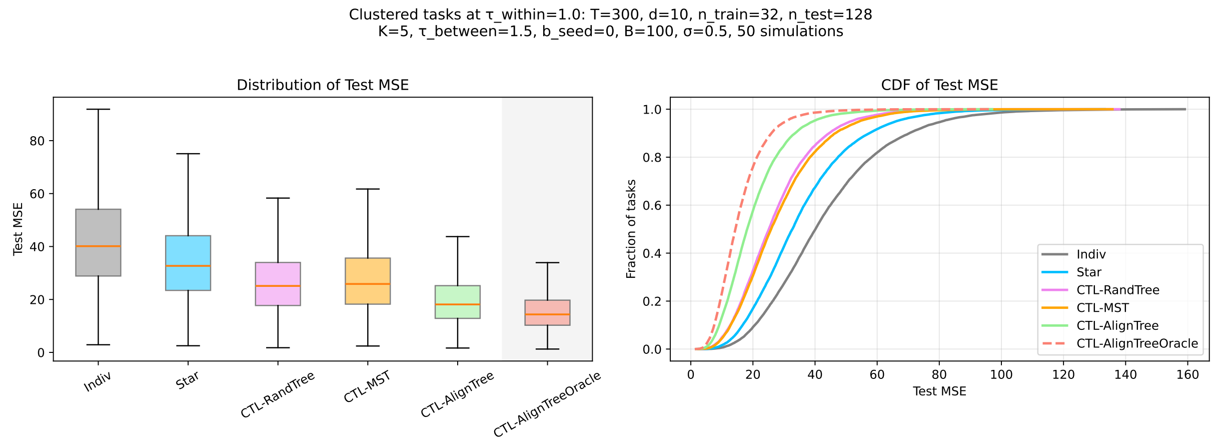


Figure 4. Performance of transfer schemes on clustered synthetic tasks at within-cluster heterogeneity $\tau_{\text{within}} = 1.0$. *Left:* distribution of per-task test MSE over $T = 300$ tasks and 50 simulations. *Right:* corresponding empirical CDFs. Cluster structure makes shallow transfer (STAR) and uninformed hierarchies (CTL-RANDTREE) substantially less effective, while all CTL variants improve markedly over the no-transfer baseline (INDIV). Alignment-based constructions (CTL-ALIGNTREE and CTL-ALIGNTREEORACLE) achieve the lowest median errors and the most left-shifted CDFs, indicating more reliable transfer when tasks form latent clusters. Lower MSE and CDFs closer to the origin correspond to better predictive performance.

F. Extended Results on the UK Dataset

Figure 5 provides additional context on the UK electricity benchmark. Figure 5a displays the spatial distribution of the 100 sampled smart meters across Oxford, showing broad geographic coverage without clear spatial clustering of high- or low-consumption households. Figure 5b reports the empirical distribution of electricity usage. The histogram is notably asymmetric: most households fall between roughly $2 - 7$ MWh per year, with a median near 5 MWh, while a long right tail extends beyond 15 MWh. This skewness reflects substantial heterogeneity across tasks, with a minority of high-consumption users acting as outliers.

Such variability is challenging for transfer learning: independent training underfits low-data tasks, whereas naive sharing can be distorted by the heavy-tailed distribution. The results in Table 4 illustrate this effect. Across all budgets, alignment-based CTL achieves the lowest or second-lowest RMSE, indicating that directional alignment is more robust to heterogeneous task magnitudes than distance-based or star-shaped baselines. In contrast, MST-based CTL improves over independent and

Table 1. Test MSE for **uniform tasks** across heterogeneity levels τ, number of tasks T, and global budgets B. Each configuration is evaluated under six strategies across 50 simulations. **Bold** and $\underline{\text{underlined}}$ entries denote best and second-best performance (excluding oracle performance).

τ	T	B	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree	*CTL-AlignTreeOracle*
1.0	100	100	160.6 ± 40.8	$\underline{158.2 \pm 34.9}$	158.9 ± 32.8	163.6 ± 36.1	$\mathbf{152.6 \pm 35.5}$	$\mathit{163.4 \pm 37.3}$
1.0	100	500	155.8 ± 37.9	162.2 ± 38.0	$\mathbf{130.9 \pm 25.7}$	$\underline{135.2 \pm 27.2}$	139.9 ± 31.5	$\mathit{140.7 \pm 29.1}$
1.0	100	1000	151.1 ± 34.4	155.0 ± 37.6	124.4 ± 23.5	$\underline{123.5 \pm 21.3}$	$\mathbf{123.4 \pm 22.5}$	$\mathit{127.7 \pm 25.8}$
1.0	200	100	159.6 ± 37.0	164.3 ± 39.5	$\underline{146.9 \pm 31.9}$	$\mathbf{144.6 \pm 29.6}$	154.4 ± 35.5	$\mathit{151.2 \pm 32.1}$
1.0	200	500	160.6 ± 40.6	161.0 ± 42.7	$\mathbf{135.1 \pm 26.7}$	$\underline{146.3 \pm 33.6}$	156.9 ± 39.4	$\mathit{151.7 \pm 34.1}$
1.0	200	1000	159.3 ± 40.7	145.0 ± 31.4	138.8 ± 28.0	$\mathbf{129.5 \pm 23.5}$	$\underline{136.5 \pm 32.0}$	$\mathit{140.8 \pm 29.0}$
5.0	100	100	333.1 ± 171.9	324.1 ± 156.5	338.0 ± 169.0	$\underline{298.3 \pm 147.0}$	$\mathbf{294.8 \pm 137.3}$	$\mathit{348.1 \pm 182.0}$
5.0	100	500	333.5 ± 176.6	322.0 ± 175.8	$\underline{286.8 \pm 139.5}$	$\mathbf{281.4 \pm 135.1}$	298.2 ± 150.4	$\mathit{306.1 \pm 159.3}$
5.0	100	1000	307.7 ± 150.0	308.9 ± 148.0	260.4 ± 118.6	$\mathbf{254.9 \pm 106.8}$	$\underline{256.0 \pm 114.1}$	$\mathit{268.7 \pm 118.5}$
5.0	200	100	339.4 ± 183.8	338.6 ± 178.9	$\underline{318.3 \pm 162.5}$	322.7 ± 161.9	$\mathbf{313.8 \pm 161.4}$	$\mathit{321.5 \pm 166.5}$
5.0	200	500	338.9 ± 175.2	323.8 ± 165.4	304.8 ± 155.8	$\underline{294.5 \pm 142.3}$	$\mathbf{290.6 \pm 135.5}$	$\mathit{315.2 \pm 164.1}$
5.0	200	1000	325.9 ± 167.4	323.5 ± 176.1	282.3 ± 133.9	$\mathbf{272.0 \pm 131.6}$	$\underline{278.8 \pm 131.3}$	$\mathit{302.4 \pm 153.5}$
10.0	100	100	900.4 ± 598.5	885.5 ± 593.9	$\underline{782.5 \pm 480.5}$	889.9 ± 633.5	$\mathbf{776.9 \pm 512.8}$	$\mathit{824.1 \pm 555.3}$
10.0	100	500	903.9 ± 653.5	841.9 ± 596.7	$\underline{767.2 \pm 518.2}$	$\mathbf{741.0 \pm 475.8}$	775.9 ± 517.6	$\mathit{776.0 \pm 528.1}$
10.0	100	1000	808.5 ± 518.0	754.5 ± 508.1	673.2 ± 402.4	$\mathbf{649.5 \pm 362.8}$	$\underline{660.5 \pm 391.8}$	$\mathit{690.7 \pm 430.1}$
10.0	200	100	880.4 ± 623.5	$\underline{873.5 \pm 631.0}$	873.8 ± 616.7	$\mathbf{831.9 \pm 566.9}$	890.0 ± 635.0	$\mathit{808.6 \pm 550.4}$
10.0	200	500	882.2 ± 611.3	869.9 ± 597.0	$\underline{784.5 \pm 499.4}$	852.1 ± 584.0	$\mathbf{774.8 \pm 489.2}$	$\mathit{833.2 \pm 545.9}$
10.0	200	1000	862.6 ± 611.9	869.6 ± 605.5	741.3 ± 481.9	$\mathbf{725.6 \pm 460.9}$	$\underline{729.1 \pm 460.3}$	$\mathit{768.0 \pm 522.2}$

random trees but is less stable when faced with outlier tasks in the heavy tail.

We extended previous experiments on the UK dataset to nonlinear models. The model considered is a simple single-layer MLP, and the results are shown in Table 5.These results once again place CTL methods ahead of individual models and the STAR transfer method, although the ALIGNTREE method performs less well than MST or RANDTREE.

G. Ablation Studies

We ablate key design choices of CTL-AlignTree to assess robustness and to isolate which components contribute most to performance. Unless stated otherwise, all ablations keep the same dataset split, learning rate η, total adaptation budget B, seed budget fraction b_{seed}, and KNN restriction parameter k. We report performance as average RMSE across tasks (lower is better), and boxplots summarize variability over repeated runs.

G.1. Seed Selection

CTL-ALIGNTREE constructs a rooted diffusion tree and performs cascade adaptation from the root. Since information flow is directed, the choice of root can affect both (i) the tree topology obtained by greedy construction and (ii) error propagation through the cascade. We therefore evaluate the sensitivity of CTL-ALIGNTREE to the seed selection strategy.

Protocol. We compare four seed selection rules: (i) **Medoid**: task minimizing the sum of pairwise distances in the task–observation space (the default of the tree builder), (ii) **Random**: uniformly sampled root task, (iii) **KNN-central**: node with maximum in-degree in the K-nearest-neighbor graph (most central), (iv) **Farthest**: node maximizing mean distance to all others (adversarial baseline). Each strategy is evaluated on identical task sets, with all other parameters held constant.

Results. Figure 6a reports the distribution of average RMSE across tasks over 100 repetitions for each seed strategy with a large global budget ($B = 5000$). In this regime, the differences between central-root strategies are small: **Random** attains the lowest mean RMSE (mean 2424.25, std 3.58; median 2423.90), closely followed by **Medoid** (mean 2424.32, std 3.38; median 2424.05). **KNN-central** is slightly worse on average (mean 2425.35, std 4.06; median 2425.55). The **Farthest** root remains the worst performer (mean 2426.63, std 5.03; median 2426.19), corresponding to an increase of about 2.38 relative to Random. Variability also increases for Farthest compared to the other strategies. For comparison, under a much tighter budget (small B), we observed larger absolute RMSE values and a clearer separation: farthest-root configurations degrade performance more noticeably, while medoid/KNN-central-type choices tend to be more stable. Taken together,

Table 2. Test MSE for **clustered tasks** across heterogeneity levels τ, number of tasks T, and global budgets B. Each configuration is evaluated under six strategies across 50 simulations. **Bold** and <u>underlined</u> entries denote best and second-best performance (excluding oracle performance).

τ	T	B	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree	*CTL-AlignTreeOracle*
1.0	100	100	940.3 ± 748.8	884.1 ± 667.0	$\mathbf{843.7 \pm 569.9}$	907.0 ± 681.9	$\underline{869.4 \pm 663.1}$	$\mathit{837.9 \pm 784.2}$
1.0	100	500	875.6 ± 639.0	908.3 ± 711.1	$\mathbf{711.4 \pm 493.8}$	754.5 ± 514.5	$\underline{720.4 \pm 443.1}$	$\mathit{743.8 \pm 523.9}$
1.0	100	1000	898.4 ± 541.3	851.9 ± 487.8	$\underline{757.6 \pm 392.6}$	$\mathbf{727.8 \pm 413.8}$	764.1 ± 423.5	$\mathit{788.5 \pm 424.2}$
1.0	200	100	881.9 ± 584.7	$\mathbf{836.6 \pm 576.6}$	865.9 ± 659.8	$\underline{851.1 \pm 590.2}$	868.5 ± 570.1	$\mathit{764.3 \pm 461.7}$
1.0	200	500	802.6 ± 573.4	744.7 ± 430.4	$\underline{714.4 \pm 458.4}$	$\mathbf{674.7 \pm 445.0}$	724.7 ± 487.0	$\mathit{728.5 \pm 468.7}$
1.0	200	1000	706.8 ± 529.8	693.6 ± 570.8	593.5 ± 431.5	$\mathbf{512.4 \pm 350.8}$	$\underline{579.1 \pm 436.1}$	$\mathit{633.6 \pm 384.0}$
5.0	100	100	1116.3 ± 851.9	1194.6 ± 870.0	$\underline{1044.4 \pm 755.1}$	$\mathbf{985.8 \pm 760.1}$	1047.6 ± 767.4	$\mathit{1098.0 \pm 871.1}$
5.0	100	500	1064.0 ± 793.5	1095.9 ± 868.8	943.4 ± 711.7	$\underline{933.5 \pm 720.7}$	$\mathbf{928.4 \pm 752.5}$	$\mathit{985.4 \pm 754.3}$
5.0	100	1000	1097.4 ± 754.8	961.8 ± 554.8	$\underline{889.9 \pm 536.8}$	956.1 ± 647.5	$\mathbf{877.1 \pm 498.4}$	$\mathit{885.0 \pm 576.4}$
5.0	200	100	1065.9 ± 770.4	1172.1 ± 834.5	$\underline{1038.5 \pm 712.5}$	$\mathbf{994.2 \pm 737.1}$	1058.8 ± 772.6	$\mathit{1057.0 \pm 766.7}$
5.0	200	500	965.4 ± 700.0	1029.5 ± 767.8	927.3 ± 602.7	$\underline{820.7 \pm 570.6}$	$\mathbf{806.3 \pm 522.8}$	$\mathit{874.7 \pm 657.2}$
5.0	200	1000	874.5 ± 702.6	851.7 ± 730.3	762.7 ± 500.0	$\underline{760.1 \pm 494.3}$	$\mathbf{747.9 \pm 507.7}$	$\mathit{778.8 \pm 682.9}$
10.0	100	100	1650.9 ± 1221.4	1579.6 ± 1401.9	$\underline{1553.1 \pm 1193.6}$	1659.8 ± 1352.8	$\mathbf{1513.5 \pm 1080.1}$	$\mathit{1551.3 \pm 1161.6}$
10.0	100	500	1632.4 ± 1250.2	1729.3 ± 1337.8	$\mathbf{1308.1 \pm 953.0}$	$\underline{1310.5 \pm 933.1}$	1467.2 ± 1175.5	$\mathit{1417.8 \pm 968.3}$
10.0	100	1000	1578.1 ± 1142.9	1588.9 ± 1143.1	1334.2 ± 927.2	$\underline{1302.6 \pm 820.4}$	$\mathbf{1252.4 \pm 791.5}$	$\mathit{1373.7 \pm 867.5}$
10.0	200	100	1607.3 ± 1212.0	1796.8 ± 1413.8	$\mathbf{1445.7 \pm 1032.9}$	1608.7 ± 1191.1	$\underline{1504.9 \pm 1089.7}$	$\mathit{1569.9 \pm 1097.9}$
10.0	200	500	1519.0 ± 1153.1	1566.3 ± 1217.1	$\underline{1418.3 \pm 1044.4}$	1430.9 ± 1029.9	$\mathbf{1398.5 \pm 985.5}$	$\mathit{1388.6 \pm 1019.1}$
10.0	200	1000	1425.9 ± 1102.6	1313.7 ± 1009.2	1225.4 ± 1031.7	$\underline{1219.9 \pm 916.6}$	$\mathbf{1097.6 \pm 722.5}$	$\mathit{1183.4 \pm 838.9}$

Table 3. Test RMSE (kWh) across global budgets B and seed budget b_{seed} on the UK electricity dataset, with a **linear** model. Each configuration is evaluated under five strategies across 30 simulations, **bold** and <u>underlined</u> entries denote best and second-best performance. Alignment-based CTL variants provide the most robust improvements across all settings.

B	K	η	b_{seed}	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree
100	5	2.0e-03	0.800	2829.892 ± 10.968	2845.335 ± 80.716	$\underline{2457.077 \pm 47.857}$	2607.956 ± 58.798	$\mathbf{2451.384 \pm 71.113}$
100	10	2.0e-03	0.800	2848.996 ± 12.660	2844.774 ± 84.874	$\underline{2373.057 \pm 62.239}$	2629.563 ± 75.194	$\mathbf{2333.388 \pm 45.825}$
100	20	2.0e-03	0.800	2853.266 ± 11.763	2763.712 ± 101.988	$\mathbf{2466.044 \pm 51.142}$	2520.385 ± 61.140	$\underline{2495.603 \pm 55.850}$
150	5	2.0e-03	0.800	2844.853 ± 10.055	2758.207 ± 64.822	2567.082 ± 68.141	$\underline{2464.587 \pm 52.672}$	$\mathbf{2444.110 \pm 32.640}$
150	10	2.0e-03	0.800	2826.969 ± 11.868	2656.323 ± 75.382	$\underline{2470.939 \pm 59.883}$	2510.837 ± 55.270	$\mathbf{2375.472 \pm 51.502}$
150	20	2.0e-03	0.800	2854.365 ± 11.767	2753.055 ± 67.057	$\underline{2515.887 \pm 45.781}$	2524.619 ± 69.341	$\mathbf{2499.562 \pm 47.826}$
200	5	2.0e-03	0.800	2753.549 ± 12.031	2833.626 ± 96.197	$\underline{2229.231 \pm 44.225}$	2294.464 ± 48.342	$\mathbf{2146.277 \pm 32.518}$
200	10	2.0e-03	0.800	2747.167 ± 11.096	2613.106 ± 86.356	$\underline{2282.548 \pm 39.112}$	2423.935 ± 58.984	$\mathbf{2171.550 \pm 43.689}$
200	20	2.0e-03	0.800	2744.488 ± 10.088	2662.037 ± 77.895	$\underline{2289.662 \pm 47.521}$	2385.993 ± 62.458	$\mathbf{2211.068 \pm 51.167}$

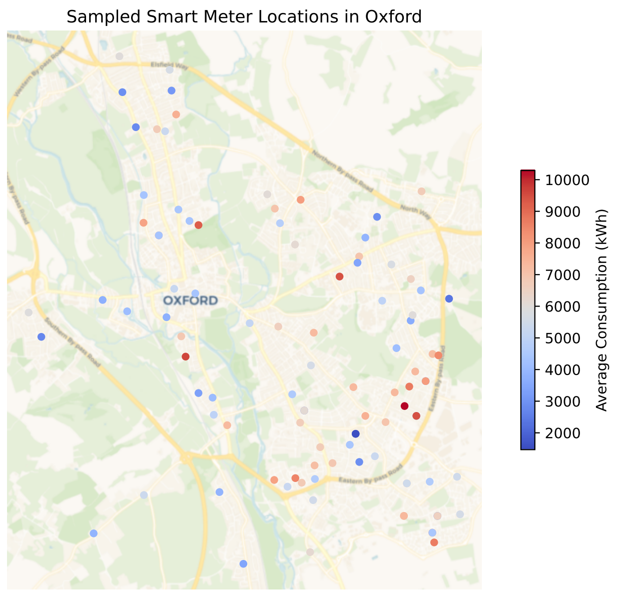
these results suggest that CTL-AlignTree is broadly robust to the root choice, and that careful seed selection matters most in the low-budget regime where each node receives only limited adaptation.

G.2. Budget Allocation

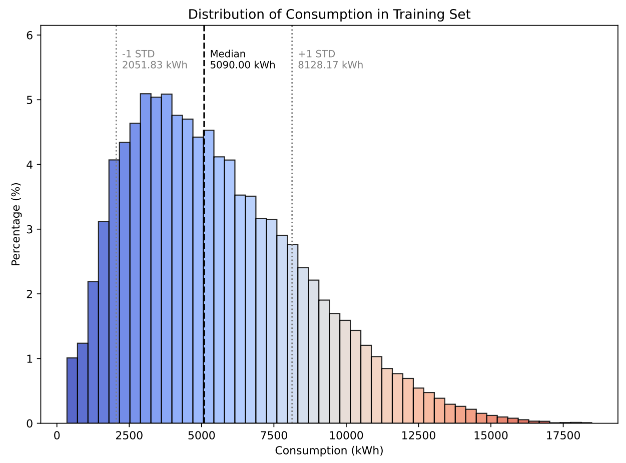
CTL-AlignTree allocates a finite adaptation budget across tasks. While our default implementation distributes budget uniformly over non-root nodes (after training a seed at the root), it is natural to consider alternative allocations that prioritize early nodes (which influence many descendants) or structurally important nodes. We therefore study how performance varies with different budget allocation schemes.

Protocol. We fix the alignment tree construction, the root selection (medoid), and all optimization settings. We only vary the allocation of per-node fine-tuning steps b_v under a fixed total budget B and prototype fraction b_{proto}. Let $\text{depth}(v)$ denote the distance (number of edges) from the root to node v. Let $\text{subtree}(v)$ denote the set of descendants of v in the rooted diffusion tree (including v itself), and $|\text{subtree}(v)|$ its size. We compare: (i) **Uniform**: constant budget for every non-root node, (ii) **Depth**: $b_v \propto \exp(-\lambda\,\text{depth}(v))$, (iii) **Subtree**: $b_v \propto |\text{subtree}(v)|^\gamma$, (iv) **Depth×Subtree**: $b_v \propto |\text{subtree}(v)|^\gamma \exp(-\lambda\,\text{depth}(v))$. Budgets are normalized to sum to B (up to integer rounding) with a minimum of one step per node. In all reported runs, we fix $\gamma = 1$ and $\lambda = 0.5$.

Results. Figure 6b shows average RMSE across tasks over 100 repetitions for four budget allocation schemes with a large global budget ($B = 5000$). In this regime, differences remain small in absolute value but are consistently ordered: **Subtree**


Sampled Smart Meter Locations in Oxford
OXFORD
Average Consumption (kWh)
10000
9000
8000
7000
6000
5000
4000
3000
2000

(a) Map of the 100 sampled smart meter locations.



(b) Distribution of yearly consumption across the sampled meters.

Figure 5. Overview of spatial positions and consumption distribution of the sampled smart meters in Oxford's urban area.

Table 4. Test RMSE (kWh) across global budgets B and seed budget b_{seed} on the UK electricity dataset. Each configuration is evaluated under five strategies across 50 simulations, **bold** and underlined entries denote best and second-best performance. Alignment-based CTL variants provide the most robust improvements across all settings.

B	k	η	b_{seed}	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree
500	20	1.5e-03	0.010	2881.870 ± 11.212	2954.772 ± 68.945	2649.688 ± 56.599	$\mathbf{2378.970 \pm 47.672}$	$\underline{2633.650 \pm 66.031}$
1000	20	1.5e-03	0.010	2818.954 ± 9.876	2706.877 ± 62.042	2474.350 ± 54.871	$\mathbf{2154.532 \pm 48.215}$	$\underline{2370.213 \pm 63.938}$
2000	20	1.5e-03	0.010	2736.551 ± 12.149	2553.488 ± 57.982	$\underline{2070.322 \pm 39.794}$	$\mathbf{1738.657 \pm 30.625}$	2204.660 ± 47.701

and **Depth**$\times$**Subtree** achieve the lowest mean RMSE (means 2422.05 and 2422.10, std 1.79 and 1.52; medians 2422.03 and 2422.01), followed by **Depth** (mean 2423.23, std 3.02), while **Uniform** is worst on average (mean 2424.30, std 3.44; median 2424.80). The mean gap between the best and worst schemes is about 2.25, indicating that budget allocation has only a mild effect when the overall budget is sufficiently large. In contrast, in the low-budget regime (e.g., the setting used in our initial ablation with small B), all schemes yield very similar performance, suggesting that when each node receives only a handful of gradient steps, the allocation rule has limited capacity to alter the outcome. Overall, CTL-ALIGNTREE appears robust to the budget allocation choice across regimes, structure-aware allocations provide small, repeatable gains when sufficient compute is available, but differences largely vanish under very tight budgets.

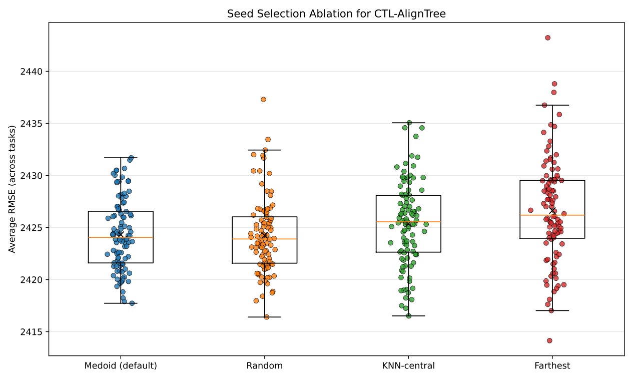
H. CTL for Image Classification

To assess the applicability of cascaded transfer learning beyond linear regression, we evaluate CTL on real, high-dimensional image classification tasks using the Fashion-MNIST and CIFAR-10 datasets (Xiao et al., 2017; Krizhevsky et al., 2009). Both datasets contain 10 classes, allowing us to construct binary classification tasks by selecting pairs of classes. With $\binom{10}{2} = 45$ possible class pairs per dataset, we sample T tasks by drawing class pairs **with replacement**. This means multiple tasks may share the same class pair but are trained on **different random subsets** of the data. This setup reflects realistic scenarios such as federated learning, where multiple clients solve similar classification problems with different local datasets. We consider a *clustered* regime in which tasks are partitioned into $K = 5$ groups, each group sharing the same underlying class pair, thereby introducing strong within-cluster relatedness.

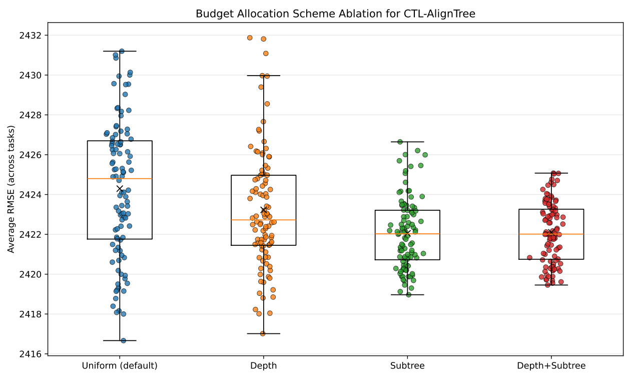
For Fashion-MNIST (28×28 grayscale images, 784 features), we apply PCA to reduce dimensionality to $d = 100$ components, preserving approximately 87% of the variance. For CIFAR-10 (32×32 RGB images, 3072 features), the

Table 5. Test RMSE (kWh) across global budgets B and seed budget b_{seed} on the UK electricity dataset, with a **non-linear** model. Each configuration is evaluated under five strategies across 30 simulations, **bold** and $\underline{\text{underlined}}$ entries denote best and second-best performance. Alignment-based CTL variants provide the most robust improvements across all settings.

B	k	η	b_{seed}	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree
100	5	1.0e-03	0.140	2618.312 ± 10.844	2689.839 ± 70.291	$\underline{2538.926 \pm 106.290}$	$\mathbf{2531.036 \pm 70.542}$	2554.687 ± 72.791
100	10	1.0e-03	0.140	$\underline{2616.487 \pm 6.629}$	2620.501 ± 68.239	2756.109 ± 60.017	2650.239 ± 46.971	$\mathbf{2557.733 \pm 69.272}$
100	20	1.0e-03	0.140	2596.840 ± 10.762	2555.750 ± 42.886	2763.908 ± 51.748	$\underline{2552.141 \pm 31.296}$	$\mathbf{2526.076 \pm 91.139}$
100	30	1.0e-03	0.140	$\underline{2599.460 \pm 18.559}$	$\mathbf{2581.927 \pm 44.590}$	2661.225 ± 70.452	2622.910 ± 66.562	2605.141 ± 57.290
150	5	1.0e-03	0.140	$\underline{2612.102 \pm 15.611}$	2804.600 ± 62.319	2653.122 ± 69.023	2696.127 ± 67.847	$\mathbf{2518.120 \pm 68.357}$
150	10	1.0e-03	0.140	2632.350 ± 13.371	$\underline{2557.400 \pm 55.668}$	2567.129 ± 47.092	2701.479 ± 62.125	$\mathbf{2481.770 \pm 100.584}$
150	20	1.0e-03	0.140	2622.093 ± 9.891	$\underline{2534.871 \pm 69.803}$	2575.999 ± 79.423	$\mathbf{2477.170 \pm 49.529}$	2599.402 ± 142.141
150	30	1.0e-03	0.140	2589.497 ± 12.852	2648.683 ± 59.126	$\underline{2491.868 \pm 42.244}$	$\mathbf{2418.865 \pm 20.030}$	2593.545 ± 83.780
200	5	1.0e-03	0.140	2607.507 ± 14.629	$\mathbf{2451.972 \pm 60.704}$	2783.387 ± 34.579	$\underline{2581.784 \pm 85.146}$	2683.777 ± 71.872
200	10	1.0e-03	0.140	2605.168 ± 17.913	2695.644 ± 65.504	$\underline{2561.368 \pm 92.097}$	$\mathbf{2544.781 \pm 101.739}$	2691.378 ± 71.093
200	20	1.0e-03	0.140	2627.906 ± 11.246	$\underline{2606.665 \pm 37.325}$	$\mathbf{2588.619 \pm 74.146}$	2730.747 ± 76.473	2638.743 ± 73.718
200	30	1.0e-03	0.140	$\underline{2599.201 \pm 7.824}$	2641.124 ± 87.155	2629.621 ± 121.504	2616.583 ± 43.735	$\mathbf{2560.698 \pm 66.951}$



(a) Seed selection.



(b) Budget allocation.

Figure 6. Ablation studies for CTL-AlignTree. Distributions of average RMSE across tasks over 100 repetitions. Left: varying root selection. Right: varying budget allocation.

same PCA reduction to $d = 100$ components preserves approximately 90% of the variance. Each task trains a logistic regression classifier with ℓ_2-regularization ($\lambda = 10^{-4}$) using gradient descent. The baselines coincide with those used in the synthetic experiments: independent training (Indiv), a star-shaped transfer hierarchy (Star), cascades built on random trees (CTL-RandTree), cascades built from feature-mean distances (CTL-MST), and cascades built from alignment distances estimated from low-budget classifiers (CTL-AlignTree). For every configuration (T, B), all methods receive the same total refinement budget, and results are averaged over 50 independent repetitions.

The transfer structures in Figure 7 exhibit differences in how each method organizes the task population.

Table 6 reports mean classification accuracy and standard deviation across tasks. Across most configurations, alignment-based CTL (and to a slightly lesser extent CTL-MST) achieves the highest accuracy, indicating that even coarse alignment signals extracted from low-budget classifiers suffice to construct effective transfer structures in this real-data setting. Gains of $3 - 5\%$ over independent training are common, and the best CTL variants consistently outperform both Star and RandTree by significant margins. These experiments demonstrate that CTL extends to realistic classification problems, reinforcing the central observation that transfer governed by principled, data-derived task alignments yields reliable benefits even when true task parameters are unknown.

Table 7 reports mean classification accuracy and standard deviation on CIFAR-10 binary tasks. The results corroborate and extend the findings from Fashion-MNIST. CTL-AlignTree consistently achieves the highest or near-highest accuracy across all configurations, with CTL-MST as a close second. Both alignment-based methods yield substantial improvements of $4 - 5\%$ over independent training, reaching accuracies above 73% in most settings. Notably, CTL-MST and CTL-AlignTree perform nearly identically in the Clustered regime, suggesting that when task structure is strong, even simpler feature-based distances can recover effective transfer topologies. The relatively modest absolute accuracies ($67 - 74\%$)

Table 6. Classification Accuracy (%) on Fashion-MNIST binary tasks. **Bold** indicates best performance, underline indicates second best.

Params		Methods				
T	B	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree
50	200	$\underline{94.8 \pm 7.6}$	93.4 ± 8.6	93.5 ± 8.0	$\mathbf{95.3 \pm 6.6}$	94.4 ± 8.0
50	400	$\underline{95.3 \pm 6.6}$	94.7 ± 7.1	94.6 ± 6.8	$\mathbf{95.7 \pm 5.7}$	95.1 ± 6.6
50	500	$\underline{95.6 \pm 5.9}$	95.0 ± 6.5	94.9 ± 6.1	$\mathbf{95.9 \pm 5.5}$	95.4 ± 6.0
100	200	93.2 ± 9.6	90.5 ± 12.2	91.6 ± 10.7	$\mathbf{94.8 \pm 7.7}$	$\underline{93.3 \pm 9.8}$
100	400	95.1 ± 7.2	93.8 ± 8.1	93.8 ± 7.7	$\mathbf{96.0 \pm 6.0}$	$\underline{95.2 \pm 6.9}$
100	500	$\underline{94.7 \pm 7.5}$	93.4 ± 8.7	93.6 ± 7.7	$\mathbf{95.4 \pm 6.4}$	94.6 ± 7.3
200	200	92.6 ± 9.6	89.6 ± 11.2	90.3 ± 11.0	$\mathbf{94.9 \pm 7.0}$	$\underline{93.9 \pm 8.0}$
200	400	93.8 ± 8.8	91.5 ± 10.5	92.2 ± 9.2	$\mathbf{95.6 \pm 6.6}$	$\underline{95.1 \pm 7.0}$
200	500	92.2 ± 10.5	89.1 ± 13.4	90.3 ± 11.4	$\mathbf{94.3 \pm 7.8}$	$\underline{93.7 \pm 8.5}$

Table 7. Classification Accuracy (%) on CIFAR-10 binary tasks. **Bold** indicates best performance, underline indicates second best.

Params		Methods				
T	B	Indiv	Star	CTL-RandTree	CTL-MST	CTL-AlignTree
50	200	$\underline{67.2 \pm 10.6}$	65.9 ± 10.7	66.1 ± 10.5	$\mathbf{67.9 \pm 10.5}$	66.6 ± 10.6
50	400	$\underline{68.9 \pm 9.6}$	67.7 ± 10.0	67.3 ± 9.8	$\mathbf{69.6 \pm 9.8}$	67.7 ± 9.7
50	500	$\underline{69.5 \pm 10.7}$	68.2 ± 11.0	68.0 ± 10.2	$\mathbf{69.8 \pm 10.7}$	68.3 ± 10.6
100	200	$\underline{66.6 \pm 10.2}$	63.8 ± 11.5	65.5 ± 10.1	$\mathbf{67.8 \pm 10.3}$	66.2 ± 10.6
100	400	67.0 ± 10.3	65.5 ± 10.8	66.0 ± 10.2	$\mathbf{68.4 \pm 10.5}$	$\underline{67.1 \pm 10.5}$
100	500	67.4 ± 10.6	66.7 ± 10.5	66.6 ± 10.1	$\mathbf{69.0 \pm 10.4}$	$\underline{67.5 \pm 10.5}$
200	200	62.7 ± 12.7	62.7 ± 11.0	64.9 ± 10.2	$\mathbf{67.4 \pm 10.4}$	$\underline{66.4 \pm 10.3}$
200	400	66.3 ± 10.8	64.7 ± 11.2	65.2 ± 11.0	$\mathbf{68.5 \pm 11.1}$	$\underline{66.8 \pm 11.3}$
200	500	67.0 ± 10.6	65.0 ± 10.9	65.8 ± 10.9	$\mathbf{68.8 \pm 10.9}$	$\underline{67.4 \pm 11.2}$

reflect the inherent difficulty of CIFAR-10 compared to Fashion-MNIST, yet the relative ranking of methods remains stable, confirming that principled transfer topology construction provides robust benefits even on more challenging, higher-complexity image data.

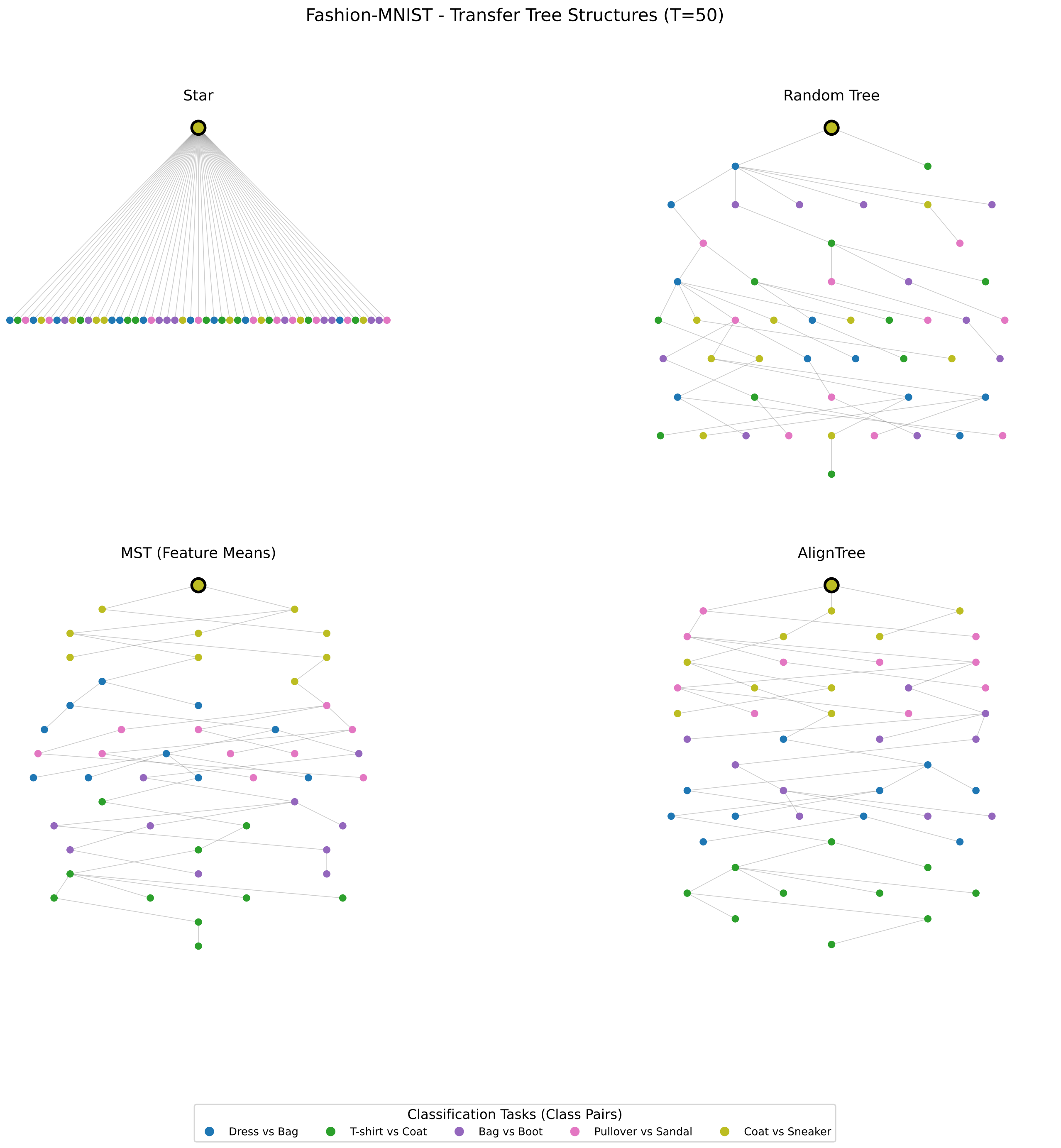


Figure 7. Transfer hierarchies produced by different methods on Fashion-MNIST binary classification tasks. The Star baseline places all tasks at depth one, RandomTree yields an unstructured topology, MST groups tasks according to feature-level similarity, and AlignTree recovers a more coherent and semantically aligned hierarchy based on data-driven task alignment.