arXiv:2109.04617v1 [cs.LG] 10 Sep 2021

Efficiently Identifying Task Groupings for Multi-Task Learning

Christopher Fifty[1], **Ehsan Amid**[2], **Zhe Zhao**[1], **Tianhe Yu**[1,3],
Rohan Anil[1], **Chelsea Finn**[1,3]
Google Brain[1], Google Research[2], Stanford University[3]
cfifty@google.com

Abstract

Multi-task learning can leverage information learned by one task to benefit the training of other tasks. Despite this capacity, naïvely training all tasks together in one model often degrades performance, and exhaustively searching through combinations of task groupings can be prohibitively expensive. As a result, efficiently identifying the tasks that would benefit from co-training remains a challenging design question without a clear solution. In this paper, we suggest an approach to select which tasks should train together in multi-task learning models. Our method determines task groupings in a single training run by co-training all tasks together and quantifying the effect to which one task's gradient would affect another task's loss. On the large-scale Taskonomy computer vision dataset, we find this method can decrease test loss by 10.0% compared to simply training all tasks together while operating 11.6 times faster than a state-of-the-art task grouping method.

1 Introduction

Many of the forefront challenges in applied machine learning demand that a single model performs well on multiple tasks, or optimizes multiple objectives while simultaneously adhering to unmovable inference-time constraints. For instance, autonomous vehicles necessitate low inference time latency to make multiple predictions on a real-time video feed to precipitate a driving action [27]. Robotic arms are asked to concurrently learn how to pick, place, cover, align, and rearrange various objects to improve learning efficiency [25], and online movie recommendation systems model multiple engagement metrics to facilitate low-latency personalized recommendations [13]. Each of the above applications depends on multi-task learning, and advances which improve multi-task learning performance have the potential to make an out-sized impact on these and many other domains.

Multi-task learning can improve modeling performance by introducing an inductive bias to prefer hypothesis classes which explain multiple objectives and by focusing attention on relevant features [43]. However, it may also lead to severely degraded performance when tasks compete for model capacity or are unable to build a shared representation that can generalize to all objectives. Accordingly, finding groups of tasks that derive benefit from the positives of co-training while mitigating the negatives often improves the modeling performance of multi-task learning systems.

While recent work has developed new multi-task learning optimization schemes [28, 10, 45, 53, 11, 50], the problem of deciding which tasks should be trained together in the first place is an understudied and complex issue that is often left to human experts [56]. However, a human's understanding of similarity is motivated by their intuition and experience rather than a prescient knowledge of the underlying structures learned by a neural network. To further complicate matters, the benefit or detriment induced from co-training relies on many non-trivial decisions including, but not limited to, dataset characteristics, model architecture, hyperparameters, capacity, and convergence [51, 49, 46,

Preprint. Under review.

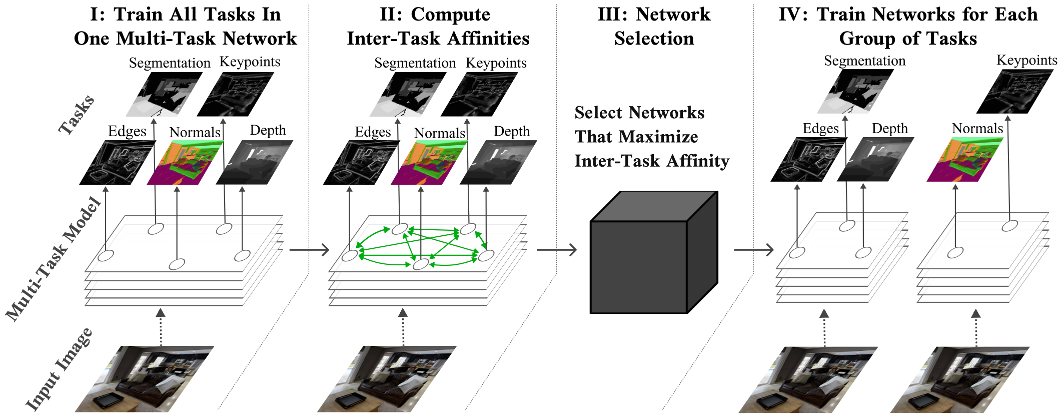


Figure 1: Overview of our suggested approach to efficiently determine task groupings. (I): Train all tasks together in a multi-task learning model. (II): Compute inter-task affinity scores during training. (III): Select multi-task networks that maximize the inter-task affinity score onto each serving-time task. (IV): Train the resulting networks and deploy to inference.

47]. As a result, a systematic technique to determine which tasks should train together in a multi-task neural network would be valuable to practitioners and researchers alike [5, 6].

One approach to select task groupings is to exhaustively search over the $2^{|\mathcal{T}|} - 1$ multi-task networks for a set of tasks $\mathcal{T}$. However, the cost associated with this search can be prohibitive, especially when there is a large number of tasks or for models under active development. Moreover, as model scale and complexity continues to increase, even approximate task grouping algorithms which evaluate only a subset of combinations may become prohibitively costly and time-consuming to evaluate.

In this paper, we aim to develop an efficient framework to select task groupings without sacrificing performance. We propose to measure inter-task affinity by training all tasks together in a single multi-task network and quantifying the effect to which one task's gradient update would affect another task's loss. This per-step quantity is averaged across training, and tasks are then grouped together to maximize the affinity onto each task. A visual depiction of the method is shown in Figure 1. Our suggested approach makes no assumptions regarding model architecture, and is applicable to any paradigm in which shared parameters are updated with respect to multiple losses.

In summary, our primary contribution is to suggest a measure of inter-task affinity that can be used to systematically and efficiently determine task groupings for multi-task learning. Our theoretical analysis shows that grouping tasks by maximizing inter-task affinity will outperform any other task grouping in the convex setting under mild conditions. Further on two challenging multi-task image benchmarks, our empirical analysis finds this approach outperforms training all tasks independently, training all tasks together (with and without training augmentations), and is competitive with a state-of-the-art task grouping method while decreasing runtime by more than an order of magnitude.

2 Related Work

Task Groupings. Prevailing wisdom suggests tasks which are similar or share a similar underlying structure may benefit from co-training in a multi-task system [9, 8, 4]. Early work in this domain pertaining to the convex setting assume all tasks share a common latent feature representation, and find that model performance can be significantly improved by clustering tasks based on the basis vectors they share in this latent space [26, 30]. However, early convex methods to determine task groupings often make prohibitive assumptions that do not scale to deep neural networks.

Deciding which tasks should train together in multi-task neural networks has traditionally been addressed with costly cross-validation techniques or high variance human intuition. An altogether different approach may leverage recent advances in transfer learning focused on understanding task relationships [54, 3, 15, 58, 2]; however, [46] show transfer learning algorithms which determine task similarity do not carry over to the multi-task learning domain and instead propose a multi-task specific framework which trains between $\binom{|\mathcal{T}|}{2} + |\mathcal{T}|$ and $2^{|\mathcal{T}|} - 1$ models to approximate exhaustive search performance. Our approach differs from [46] in that it computes task groupings from only a single training run.

Architectures and Training Dynamics. A plethora of multi-task methods addressing what parameters to share among tasks in a model have been developed, such as Neural Architecture Search [19, 47, 49, 35, 22, 39], Soft-Parameter Sharing [40, 14, 52], and asymmetric information transfer [31, 44, 32] to improve multi-task performance. Although this direction is promising, we direct our focus towards *when to share* tasks in a multi-task network rather than architecture modifications to maximize the benefits of co-training. Nevertheless, both approaches are complementary, and architecture augmentations seem to perform best when trained with related tasks [43].

Significant effort has also been invested to improve the optimization dynamics of MTL systems. In particular, dynamic loss reweighing has achieved performance superior to using fixed loss weights found with extensive hyperparameter search [28, 18, 37, 10, 45, 33]. Another set of methods seek to mitigate inter-task conflict by manipulating the direction of task gradients rather than simply their magnitude [48, 57, 53, 11, 50, 36]. In our experiments, we compare against Uncertainty Weights [28], GradNorm [10], and PCGrad [53] to contextualize the relative change in performance from splitting tasks into groups. We find that task grouping methods outperform all three training augmentations; nonetheless, they are naturally complementary. In Section 5, our results indicate enhancing the networks found by our method with PCGrad can lead to additional improvements in performance.

Looking into the Future. "Lookahead" methods in deep learning can often be characterized by saving the current state of the model, applying one or more gradient updates to a subset of the parameters, reloading the saved state, and then leveraging the information learned from the future state to modify the current set of parameters. This approach has been used extensively in the meta-learning [16, 42, 7, 17, 29], optimization [41, 21, 55, 23, 24], and recently auxiliary task learning domains [34]. Unlike the above mentioned methods which look into the future to modify optimization processes, our work adapts this central concept to the multi-task learning domain to characterize task interactions and assign tasks to groups of networks.

3 Task Grouping Problem Definition

We draw a distinction between inference-time latency constraints and inference-time budget. The former characterizes the speed at which predictions can be computed, with similarly sized models running in parallel having latency roughly equivalent to a single model running by itself. The latter relates to the number of parameters used by all models during inference, with a n-times parameter model having similar budget to an n-group of normal sized models. We configure our analysis to span both dimensions, but also provide analysis into only the latter in the Appendix.

Given a set of tasks $\mathcal{T}$, a fixed inference-time budget b, and a fixed inference-time latency constraint c, our aim is to assign tasks to networks such that combined task performance is maximized. Additionally, each network must have parameter count less than c; the networks must span our set of tasks $\mathcal{T}$; and the total number of networks is less than or equal to our inference time budget b. Moreover, tasks can be trained in a model where they are not served to assist in the training of other tasks. Formulating our task grouping framework in this manner aligns our work with industry trends where inference-time budget is limited and inference-time latency unmovable.

More formally, for a set of n tasks $\mathcal{T} = \{\tau_1, \tau_2, .., \tau_n\}$, we would like to construct a group of k multi-task neural networks $M = \{m_1, m_2, ..., m_k\}$ such that $\forall \tau_i \in \mathcal{T}$, $\exists$ **exactly one** multi-task network $m_j \in M$, parameter count of $m_j < c$, such that m_j makes an inference-time prediction for t_i subject to $k \leq b$ where b is our fixed inference-time budget. m_j is the j^{th} multi-task network which takes an input X, and concurrently trains a set of tasks $\{\tau_a, \tau_c, ..., \tau_f\}$, but only serves a subset of those tasks at inference. For a given performance measure $\mathcal{P}$, we can then define the aggregate performance of our task grouping as $\sum_{i=1}^{n} \mathcal{P}(\tau_i|M)$ where $\mathcal{P}(\tau_i|M)$ computes the performance of task τ_i from the set of models M using the model m_j which the task grouping algorithm predicts the performance of τ_i will be highest.

4 Grouping Tasks by Measuring Inter-Task Affinity

We propose a method to group tasks by examining the effect to which one task's gradient would increase or decrease another task's loss. We formally define the method in Section 4.1, describe a systematic procedure to go from inter-task affinity scores to a grouping of tasks in Section 4.2, and provide theoretical analysis in Section 4.3.

4.1 Inter-Task Affinity

Within the context of a hard-parameter sharing paradigm, tasks collaborate to build a shared feature representation which is then specialized by individual task-specific heads to output a prediction. Specifically, through the process of successive gradient updates to the shared parameters, tasks implicitly transfer information to each other. As a consequence, we propose to view the extent to which a task's successive gradient updates on the shared parameters affect the objective of other tasks in the network as a proxy measurement of inter-task affinity.

Consider a multitask loss function parameterized by $\{\theta_s\} \cup \{\theta_i | \, i \in \mathcal{T}\}$ where θ_s represents the shared parameters and θ_i represents the task $i \in \mathcal{T}$ specific parameters. Given a batch of examples $\mathcal{X}$, let

$$L_{\text{total}}(\mathcal{X}, \theta_s, \{\theta_i\}) = \sum_{i \in \mathcal{T}} L_i(\mathcal{X}, \theta_s, \theta_i)\,,$$

denote the total loss where L_i represents the non-negative loss of task i. For simplicity of notation, we set the loss weight of each task to be equal to 1, though our construction generalizes to arbitrary weightings.

For a given training batch $\mathcal{X}^t$ at time-step t, define the quantity $\theta_{s|i}^{t+1}$ to represent the updated shared parameters after a gradient step with respect to the task i. Assuming stochastic gradient descent for simplicity, we have

$$\theta_{s|i}^{t+1} := \theta_s^t - \eta \nabla_{\theta_s^t} L_i(\mathcal{X}^t, \theta_s^t, \theta_i^t)\,.$$

We can now calculate a *lookahead* loss for each task by using the using the updated shared parameters while keeping the task-specific parameters as well as the input batch unchanged. That is, in order to assess the effect of the gradient update of task i on a given task j, we can compare the loss of task j before and after applying the gradient update from task i onto the shared parameters. To eliminate the scale discrepancy among different task losses, we consider the ratio of a task's loss before and after the gradient step on the shared parameters as a scale invariant measure of relative progress. We can then define an asymmetric measure for calculating the *affinity* of task i at a given time-step t on task j as

$$\mathcal{Z}_{i \rightarrow j}^t = 1 - \frac{L_j(\mathcal{X}^t, \theta_{s|i}^{t+1}, \theta_j^t)}{L_j(\mathcal{X}^t, \theta_s^t, \theta_j^t)}\,. \tag{1}$$

Notice that a positive value of $\mathcal{Z}_{i \rightarrow j}^t$ indicates that the update on the shared parameters results in a lower loss on task j than the original parameter values, while a negative value of $\mathcal{Z}_{i \rightarrow j}^t$ indicates that the shared parameter update is antagonistic for this task's performance. Our suggested measure of inter-task affinity is computed at a per-step level of granularity, but can be averaged across all steps, every n steps, or a contiguous subset of steps to derive a "training-level" score:

$$\hat{\mathcal{Z}}_{i \rightarrow j} = \frac{1}{T} \sum_{t=1}^{T} \mathcal{Z}_{i \rightarrow j}^t$$

In Section 5, we find $\hat{\mathcal{Z}}_{i \rightarrow j}$ is empirically effective in selecting high performance task groupings and provide an ablation study along this dimension in Section 5.2.

4.2 Network Selection Algorithm

At a high level, our proposed approach trains all tasks together in one model, measures the pairwise task affinities throughout training, identifies task groupings that maximize total inter-task affinity, and then trains the resulting groupings for evaluation on the test set. We denote this framework as Task Affinity Grouping (TAG) and now describe the algorithm to convert inter-task affinity scores into a set of multi-task networks. More formally, given $\binom{|\mathcal{T}|}{2}$ values representing the pairwise inter-task affinity scores collected during a single training run, our network selection algorithm should produce k multi-task networks, $k \leq b$ where b is the inference-time budget, with the added constraint that every task must be served from exactly one network at inference.

For a group composed of a pair of tasks $\{a, b\}$, the affinity score onto task a would simply be $\hat{\mathcal{Z}}_{b \rightarrow a}$ and the affinity score onto task b would be $\hat{\mathcal{Z}}_{a \rightarrow b}$. For task groupings consisting of three or more tasks, we approximate the inter-task affinity onto a given task by averaging the pairwise affinities onto this given task. Consider the group consisting of tasks $\{a, b, c\}$. We can compute the total inter-task affinity onto task a by averaging the pair-wise affinities from tasks b and c onto a: $(\hat{\mathcal{Z}}_{b \to a} + \hat{\mathcal{Z}}_{c \to a})/2$.

After approximating higher order affinity scores for each network consisting of three or more tasks, we select a set of k multi-task networks such that the total affinity score onto each serving-time task is maximized. Informally, each task which is being served at inference should train with the tasks which most decrease its loss throughout training. This problem is NP-hard (reduction from Set-Cover), but can be solved efficiently with a branch-and-bound-like algorithm as detailed in [46] or with a binary integer programming solver as done by [54].

4.3 Theoretical Analysis

We now offer theoretical analysis of our measure of inter-task affinity. Specifically, our goal is to provide an answer to the following question: given that task b induces higher inter-task affinity than task c on task a, does training $\{a, b\}$ together result in a lower loss on task a than training $\{a, c\}$? Intuitively, we expect the answer to this question to always be positive. However, it is easy to construct counter examples for simple quadratic loss functions where training $\{a, b\}$ actually induce a higher loss than training $\{a, c\}$ (see Appendix). Nonetheless, we show that in a convex setting and under some mild assumptions, the grouping suggested by our measure of inter-task affinity is guaranteed to induce a lower loss value on task a.

For simplicity of notation, we ignore the task specific parameters and use $L_a(\theta)$ to denote the loss of task a evaluated at shared parameters θ. Additionally, we denote the gradient of tasks a, b, and c at θ as g_a, g_b, and g_c, respectively.

Lemma 1. *Let L_a be a α-strongly convex and β-strongly smooth loss function. Given that task b induces higher inter-task affinity than task c on task a, the following inequality holds:*

$$g_a \cdot g_c - \frac{\beta\,\eta}{2}\,\|g_c\|^2 + \frac{\alpha\,\eta}{2}\,\|g_b\|^2 \le g_a \cdot g_b \tag{2}$$

The proof is given in the Appendix.

Proposition 1. *Let L_a be a α-strongly convex and β-strongly smooth loss function. Let $\eta \le \frac{1}{\beta}$ be the learning rate. Suppose that in a given step, task b has higher inter-task affinity than task c on task a. Moreover, suppose that the gradients have equal norm, i.e. $\|g_a\| = \|g_b\| = \|g_c\|$. Then, taking a gradient step on the parameters using the combined gradient of a and b reduces L_a more so than taking a gradient step on the parameters using the combined gradient of a and c, given that $\cos(g_a, g_c) \le \frac{\eta}{4}\frac{\alpha\,\beta}{\beta - \alpha} - 1$ where $\cos(u, v) := \frac{u \cdot v}{\|u\|\,\|v\|}$ is the cosine similarity between u and v.*

Proof. Applying the strong smoothness upper-bound on the updated loss using the combined gradient $g_a + g_b$, we have

$$\begin{aligned} L_a(\theta - \eta\,(g_a + g_b)) &\le L_a(\theta) - \eta\, g_a \cdot (g_a + g_b) + \frac{\eta^2\,\beta}{2}\,\|g_a + g_b\|^2 \\ &= L_a(\theta) + (\eta^2\,\beta - \eta)\, g_a \cdot g_b + (\frac{\eta^2\,\beta}{2} - \eta)\,\|g_a\|^2 + \frac{\eta^2\,\beta}{2}\,\|g_b\|^2 \end{aligned}$$

We would like to show that the last line is less than or equal to a lower-bound on the loss obtained using the combined gradient $g_a + g_c$

$$L_a(a) - \eta\, g_a\,(a_a + g_c) + \frac{\eta^2\,\alpha}{2}\,\|g_a + g_c\|^2 \le L_a(\theta - \eta\,(g_a + g_c))$$

Eliminating the common terms, we would like to show the following inequality holds

$$(\eta\beta - 1)\, g_a \cdot g_b + \frac{\eta\,\beta}{2}(\|g_a\|^2 + \|g_b\|^2) \le (\eta\alpha - 1)\, g_a \cdot g_c + \frac{\eta\alpha}{2}(\|g_a\|^2 + \|g_c\|^2)\,.$$

Using $\eta \le \frac{1}{\beta}$, the first term becomes negative. Replacing for $g_a \cdot g_b$ using Eq. (2), it suffices to show the following inequality

$$\begin{aligned} (\eta\beta - 1)\,(g_a \cdot g_c + \frac{\eta\alpha}{2}(\|g_b\|^2) - \frac{\eta\beta}{2}(\|g_c\|^2)) + \frac{\eta\beta}{2}(\|g_a\|^2 + \|g_b\|^2) \\ \le (\eta\alpha - 1)\, g_a \cdot g_c + \frac{\eta\alpha}{2}(\|g_a\|^2 + \|g_c\|^2)\,. \end{aligned}$$

Rearranging the terms, it suffices to show

$$0 \leq \|g_a\|^2(\frac{\eta}{2}(\alpha-\beta)) + \|g_b\|^2(-\frac{\eta}{2}\alpha(\eta\beta-1)) + \|g_c\|^2(\frac{\eta}{2}(\alpha-\beta) + \frac{\eta^2}{2}\beta^2) + \eta(\alpha-\beta)\, g_a \cdot g_c \,.$$

Under the mild assumption that $\|g_a\| = \|g_b\| = \|g_c\|$, we can rearrange the terms to obtain $\cos(g_a, g_c) \leq \frac{\eta}{4}\frac{\beta/\alpha}{\beta/\alpha - 1} - 1$, where $\cos(u, v) := \frac{u \cdot v}{\|u\|\,\|v\|}$ is the cosine similarity between u and v. □

Proposition 1 intuitively implies the grouping chosen by maximizing per-task inter-task affinity is guaranteed to make more progress than any other group. Moreover, the assumptions for this condition to hold in the convex setting are fairly mild. Specifically, our first assumption relies on the gradients to have equal norms; but our proof can be generalized to when the gradient norms are approximately equal. This is often the case during training when tasks use similar loss functions and/or compute related quantities. The second assumption relies on the cosine similarity between the lower-affinity task c and the primary task a being smaller than a constant. This is a mild assumption since the ratio $\frac{\beta/\alpha}{\beta/\alpha - 1}$ becomes sufficiently large for the assumption to hold trivially when the Hessian of the loss has a sufficiently small condition number.

5 Experiments

We evaluate the capacity of TAG to select task groupings on CelebA, a large-scale face attributes dataset [38] and Taskonomy, a massive computer vision dataset of indoor scenes [54]. Following this analysis, we direct our focus to answering the following questions with ablation experiments on CelebA:

- Does our measure of inter-task affinity align with identifying which tasks should train together?
- Should inter-task affinity be measured at every step of training to determine task groupings?
- Is measuring the change in train loss comparable with the change in validation loss when computing inter-task affinity?
- Do changes in a model's hyperparamters change which tasks should be trained together?

As described in Section 3, we constrain all networks to have the same number of parameters to adhere to a fixed inference-time latency constraint. We provide experimental results removing this constraint, as well as additional experimental results and detail relating to experimental design, in the Appendix.

5.1 Supervised Task Grouping Evaluation

For our task grouping evaluation, we compare two classes of approaches: approaches that determine task groupings, and approaches that train on all tasks together but alter the optimization. In the first class, we consider simply training all tasks together in the same network (MTL), training every task by itself (STL), the expected value from randomly selecting task groupings (RG), grouping tasks by maximizing inter-task cosine similarity between pairs of gradients (CS), our method (TAG), and HOA [46] which approximates higher-order task groupings from pair-wise task performance. For the latter class, we consider Uncertainty Weights (UW) [28], GradNorm (GN) [10], and PCGrad [53]. In principle, these two classes of approaches are complementary and can be combined.

Our empirical findings are summarized in Figure 2. For task grouping methods, we report the time to determine task groupings, not determine task groupings and train the resultant multi-task networks. This is to facilitate comparison between the efficiency of different task grouping methods and provide a high-level overview of how long it takes to select task groupings compared to popular multi-task learning benchmarks.

CelebA. We select a subset of 9 attributes {a1, a2, a3, a4, a5, a6, a7, a8, a9} from the 40 possible attributes in CelebA and optimize the baseline MTL model by tuning architecture, batch size, and learning rate to maximize the performance of training all tasks together on the validation set. We do not tune other methods with the exception of GradNorm, for which we search over {0.1, 0.5, 1.0, 1.5, 2.0, 3.0, 5.0} for alpha. For task grouping algorithms, we evaluate the set of {2-splits, 3-splits, 4-splits} inference-time budgets. Our findings are summarized in Figure 2 (left).

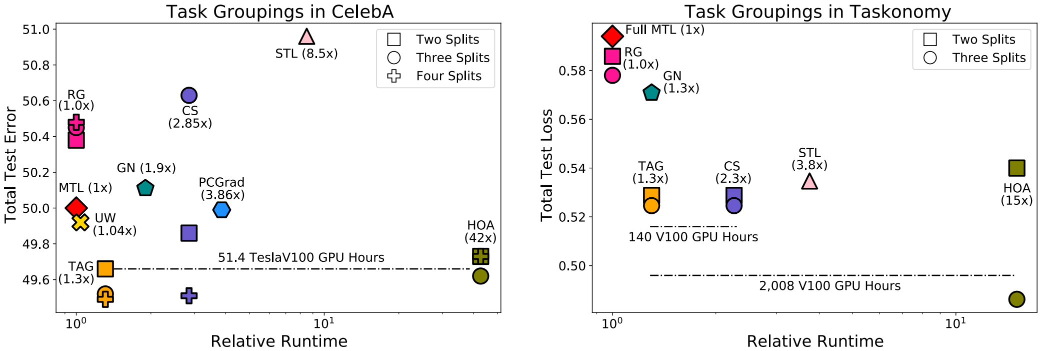


Figure 2: (Left) average classification error for 2, 3, and 4-split task groupings for the subset of 9 tasks in CelebA. (Right) total test loss for 2 and 3-split task groupings for the subset of 5 tasks in Taskonomy. All models were run on a TeslaV100 instance with the time to train the full MTL model being approximately 83 minutes in CelebA and 146 hours in Taskonomy. Note the x-axis is in log scale, and the relative runtime for each task grouping method only considers the time to find groups.

We find the performance of TAG surpasses that of the HOA, RG, and CS task grouping methods, while operating 22 times faster than HOA. We also find UW, GN, and PCGrad to perform worse than the groups found by either HOA or TAG, suggesting the improvement from identifying tasks which train well together cannot be replaced with current multi-task training augmentations. Nevertheless, we find multi-task training augmentations can be complementary with task grouping methods. For example, augmenting the groups found by TAG with PCGrad improves 2-splits performance by 0.85%, 3-splits performance by 0.18%, but changes 4-splits performance by -0.08%.

Similar to the results from [46], we find GradNorm [10] can sometimes perform worse than training all tasks together. We reason this difference is due to our common experimental design that loads the weights from the epoch with lowest validation loss to reduce overfitting, and differs from prior work which uses model weights after training for 100 epochs [45]. Reformulating our design to mirror [45], we find training the model to 100 epochs without early stopping results in worse performance on MTL, GradNorm, UW, and PCGrad; however with this change, each training augmentation method now significantly outperforms the baseline MTL method.

Taskonomy. Following the experimental setup of [46], we evaluate the capacity of TAG to select task groupings on the "Segmentic Segmentation", "Depth Estimation", "Keypoint Detection", "Edge Detection", and "Surface Normal Prediction" objectives in Taskonomy. Unlike [46], our evaluation uses an augmented version of the medium Taskonomy split (2.4 TB) as opposed to the Full+ version (12 TB) to reduce computational overhead and increase reproducibility.

Our results are summarized in Figure 2 (right). Similar to our findings on CelebA, TAG continues to outperform MTL by 10.0%, GN by 7.7%, STL by 1.5%, and RG by 9.5%. Comparing TAG to HOA, we find the 2-split task grouping found by TAG to surpass the performance of that found by HOA by 2.5%, but HOA's 3-split task grouping performance is superior to that of TAG. In terms of compute, TAG is significantly more efficient than HOA, with HOA demanding an additional 2,008 TeslaV100 GPU hours to find task groupings. To put this cost into perspective, even on an 8-GPU, on-demand p3.16xlarge AWS instance, the difference in monetary expenditure between TAG and HOA would be $6,144.48. On a similar note, the performance of TAG and CS on Taskonomy are equivalent, but TAG is more efficient, requiring 140 fewer TeslaV100 GPU hours to compute task groupings.

Tasks	Improvement in Test Accuracy Relative to Random Grouping		
	optimal	(ours)	worst
a1	2.60%	2.6%	-3.03%
a2	1.53%	1.29%	-1.95%
a3	2.37%	1.72%	-3.04%
a4	2.67%	0.72%	-4.14%
a5	2.75%	2.29%	-3.87%
a6	2.04%	0.00%	-2.08%
a7	2.40%	0.74%	-2.43%
a8	1.61%	-1.59%	-1.59%
a9	8.38%	8.38%	-6.21%

Table 1: Performance of each task when trained with it's partner task *relative* to the expected performance from random groupings.

5.2 Multi-Task Ablation Studies

Does our measure of inter-task affinity correlate with optimal task groupings? To further evaluate if TAG can be used to select which tasks should train together in multi-task learning models, we evaluate its capacity to select the best co-training partner for a given task. We compare with the optimal (best) and worst auxiliary task computed from the test set and normalize scores with respect to the expected performance of selecting an auxiliary task at random.

Our findings are summarized in Table 1 and indicates the performance of auxiliary tasks found by TAG correlates with performance of the optimal auxiliary task (Pearson's Correlation of 0.93%). However, a notable exception occurs with attribute a8 where TAG actually selects the worst partner for co-training. In this instance, and unlike every other objective in our dataset, no task manifests especially high or low inter-task affinity onto a8. The difference in normalized inter-task affinity for a8 between the best and worst partner predicted by inter-task affinity is 0.04, while the next smallest difference is 4x larger at 0.16 for a3. A table showing these differences is included in the Appendix. This case represents a limitation of TAG, and it will struggle to find the best auxiliary task for objectives like a8.

However, this weakness does not seem to significantly affect the capacity of TAG to select strong task groupings. As no other task exhibits especially high or low inter-task affinity onto a8, it is often slotted into a larger task group rather than being one of the tasks with a large difference in inter-task affinity which precipitates the formation of a new group.

Should Inter-Task Affinity Be Computed at Every Step?

It is likely that the inter-task affinity of consecutive steps are similar, and it could be the case that inter-task affinity signals at the beginning, middle, or end of training are sufficient for selecting which tasks should train together. In particular, if task relationships crystallize early in training, computing inter-task affinities during the initial stages of training may be sufficient. We evaluate both hypotheses on CelebA and our results are summarized in Table 2.

Method	Relative Performance	Relative Speedup
Every 1 Step	5.13%	1.0x
Every 5 Steps	5.13%	2.56x
Every 10 Steps	5.13%	3.19x
Every 25 Steps	4.84%	3.73x
Every 50 Steps	3.73%	3.96x
Every 100 Steps	2.06%	4.08x
First 25%	3.67%	4.00x
Middle 25%	4.31%	2.95x
Final 25%	4.02%	2.34x

Table 2: Change in total test accuracy across inference-time budget of {2-groups, 3-groups, 4-groups} relative to the expected performance from random groupings. Speedup is relative to computing inter-task affinity in each step.

Our findings indicate significant redundancy can be eliminated without degrading performance by computing inter-task affinities every 10 steps rather than every step. This modification increases training-time efficiency by 319% and is used in Section 5.1 to compute task groupings. After this threshold, signal strength decreases and leads to higher task grouping error. We also find that computing inter-task affinities in the first 25%, middle 25%, or final 25% of training degrades task-grouping performance. This result suggest the relationships among tasks change throughout training as measured by inter-task affinity. As a result, we choose to average inter-task affinity scores throughout the entirety of training to determine which tasks should train together.

Is Change in Train Loss Comparable to Change in Validation Loss? Given multi-task learning's capacity to improve generalization, computing the change in validation loss after a gradient step may capture a more informative signal as to which tasks should train together. On the other hand, certain datasets may not contain a validation split and loading a batch from the validation set every 10 steps of training would decrease efficiency.

To our surprise, the inter-task affinity scores computed on the validation set are very similar to the inter-task affinity scores computed on the training set (Pearson's Coefficient: 0.9804). For context, this similarity with computing inter-task affinities every step is greater than any other ablation in Table 2 with the exception of "Every 5 Steps" as measured by Pearson's Coefficient. Moreover, the performance of groupings found by both methods are similar: 49.574 average total error across our inference time budget of {2-groups, 3-groups, 4-groups} compared with 49.576 for groupings found on the validation set.

More formally, let X_{tr} and X_{val} be independent and identically distributed random variables for training and validation, respectively. Given the updated shared parameter $\theta^{t+1}_{s|i}$ using the gradient of task i calculated on X_{tr}, the loss of task j (s.t. $j \neq i$) yields similar expectations with respect to X_{tr} and X_{val}. That is,

$$\mathbb{E}_{\text{val}}[\mathbb{E}_{\text{tr}}[\mathcal{L}_j(X^t_{\text{val}}, \theta^{t+1}_{s|i}, \theta^t_j)]] \approx \mathbb{E}_{\text{val}}[\mathcal{L}_j(X^t_{\text{val}}, \mathbb{E}_{\text{tr}}[\theta^{t+1}_{s|i}], \theta^t_j)] = \mathbb{E}_{\text{tr}}[\mathcal{L}_j(X^t_{\text{tr}}, \mathbb{E}_{\text{tr}}[\theta^{t+1}_{s|i}], \theta^t_j)] \,.$$

The leftmost term is a joint expectation w.r.t. both X_{tr} and X_{val}, in which X_{tr} is only used to calculate the updated shared parameters $\theta^{t+1}_{s|i}$. Assuming both tasks have distinct loss functions that do not directly depend on each other during training, we can move the second expectation inside the loss function to obtain the middle term. By using the fact that X_{tr} and X_{val} are identically distributed, thus yielding the same expectation, we can move from the middle term to the rightmost term. Hence, the inter-task affinity computed on the training dataset would approximately equal the inter-task affinity computed on the validation set.

Do Changes in Hyperparameters Affect Task Groupings? We define three settings: (i) typical, the base setting, (ii) b=0.5x, or halving the batch size, and (iii) lr=2x, or increasing the learning rate by a factor of 2. To determine the ground-truth task groupings, we train all 511 combinations of task groupings from our 9-task subset of CelebA and select the groups in each setting with lowest total test error. Our aim is to assess the extent to which task groupings chosen in setting b=0.5x or lr=2x generalize to the typical setting.

Our results are summarized in Table 3. They indicate that simply changing the batch size or learning rate of a model changes which tasks should be trained together, with the groupings found by lr=2x exhibiting worse generalization than those found by b=0.5x. This result suggests that how tasks should be trained together does not simply depend on the relationships among tasks, but also on detailed aspects of the model and training. It is notably difficult to build intuition for the latter, illustrating the need to develop automated methods that can take into account these nuances.

Budget	Improvement in Test Accuracy Relative to Optimal Groupings	
	b=128x	lr=2x
2-groups	-0.61%	-1.22%
3-groups	-0.25%	-2.90%
4-groups	-1.87%	-3.21%

Table 3: Accuracy of task groupings found by b=128x and lr=2x relative to the typical setting.

6 Conclusion

In this work, we present an approach to quantify inter-task affinity in a single training run and show how this quantity can be used to systematically determine which tasks should train together in multi-task networks. Our empirical findings indicate our approach is highly competitive. It outperforms multi-task training augmentations like Uncertainty Weights, GradNorm, and PCGrad, and performs competitively with state-of-the-art task grouping methods like HOA, while improving computational efficiency by over an order of magnitude. Further, our findings are supported by extensive analysis that suggests inter-task affinity scores can find close to optimal auxiliary tasks, and in fact, implicitly measure generalization capability among tasks.

A plethora of research has been undertaken to design better multi-task learning architectures, or improve the optimization dynamics within multi-task learning systems, with relatively little work addressing the question of **which** tasks should train together in the first place. It is our hope this work renews interest in this domain, and given the sensitivity of task groupings to even small changes in hyperparameters, encourages the development of efficient and automatic methods to identify which tasks should train together in multi-task learning networks.

7 Broader Impact

Efficiently identifying task groupings in multi-task learning has the potential to save significant time and computational resources in both academic and industry environments. Despite this benefit, there are several risks associated with this work. In particular, inter-task affinities can be mistakenly interpreted as “task similarity”, and incorrectly create an association and/or causation relationship among tasks with high mutual inter-task affinity scores. This association would be especially problematic for datasets involving sensitive prediction quantities related to race, gender, religion, age, status, physical traits, etc., where inter-task affinities could be mistakenly used to support an unfounded conclusion that attempts to posit similarity among tasks. That said, we believe acknowledging these risks mitigates their potential for abuse, and the benefit from this work — most notably decreasing computational resources by over an order of magnitude compared with a state-of-the-art task grouping method while performing competitively in terms of accuracy — merits its dissemination.

Acknowledgement

We thank Vince Gatto, Bing-Rong Lin, Nick Bridle, Li Wei, Shawn Andrews, and Yuyan Wang for their thoughtful discussion related to a precursor of this work. We also thank James Chen and Linda Wang for providing feedback on an earlier draft of this paper. Lastly, we would like to recognize Zirui Wang for technical support with running experiments.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019.

[3] Alessandro Achille, Giovanni Paolini, Glen Mbeng, and Stefano Soatto. The information complexity of learning tasks, their structure and their distance. *arXiv preprint arXiv:1904.03292*, 2019.

[4] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine learning*, 73(3):243–272, 2008.

[5] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[6] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer, 2003.

[7] Lukas Brinkmeyer, Rafael Rego Drumond, Randolf Scholz, Josif Grabocka, and Lars Schmidt-Thieme. Chameleon: Learning model initializations across tasks with different schemas. *arXiv preprint arXiv:1909.13576*, 2019.

[8] Rich Caruana. Multitask learning. *Machine Learning, 28*, pages 41–75, 1997.

[9] Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

[10] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017.

[11] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33, 2020.

[12] François Chollet et al. Keras: The python deep learning library. *ascl*, pages ascl–1806, 2018.

[13] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[14] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, 2015.

[15] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12387–12396, 2019.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[17] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes, 2018.

[18] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–287, 2018.

[19] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. *arXiv preprint arXiv:2006.01895*, 2020.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186, 1987.

[22] Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, and Alexander Hauptmann. GNAS: A greedy neural architecture search method for multi-attribute learning. *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*, 2018.

[23] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[24] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[25] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[26] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

[27] Andrej Karpathy. Multi-task learning in the wilderness. ICML, 2019. URL `https://slideslive.com/38917690/multitask-learning-in-the-wilderness`.

[28] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[29] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *arXiv preprint arXiv:1806.03836*, 2018.

[30] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

[31] Giwoong Lee, Eunho Yang, and Sung Hwang. Asymmetric multi-task learning based on task relatedness and loss. In *International conference on machine learning*, pages 230–238. PMLR, 2016.

[32] Hae Beom Lee, Eunho Yang, and Sung Ju Hwang. Deep asymmetric multi-task feature learning. In *International Conference on Machine Learning*, pages 2956–2964. PMLR, 2018.

[33] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, pages 12037–12047, 2019.

[34] Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[35] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[36] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=IMPnRXEWpvr`.

[37] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019.

[38] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[39] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[40] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[41] Yu Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. In *Sov. Math. Dokl*, volume 27, pages 372–376, 1983.

[42] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[43] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[44] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[45] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

[46] Trevor Standley, Amir R Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? *arXiv preprint arXiv:1905.07553*, 2019.

[47] Ximeng Sun, Rameswar Panda, and Rogerio Feris. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

[48] Mihai Suteu and Yike Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019.

[49] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.

[50] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*, 2020.

[51] Sen Wu, Hongyang R Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. *arXiv preprint arXiv:2005.00944*, 2020.

[52] Yongxin Yang and Timothy M. Hospedales. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*, 2016.

[53] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[54] Amir R. Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. *2018 IEEE/CVF Conference*, Jun 2018.

[55] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9597–9608, 2019.

[56] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[57] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–416, 2018.

[58] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020.

Appendix

A Proofs of Theoretical Results

A.1 Proof of Lemma 1

Lemma 1. *Let L_a be a α-strongly convex and β-strongly smooth loss function. Given that task b induces higher inter-task affinity than task c on task a, the following inequality holds:*

$$g_a \cdot g_c - \frac{\beta\,\eta}{2}\,\|g_c\|^2 + \frac{\alpha\,\eta}{2}\,\|g_b\|^2 \le g_a \cdot g_b \tag{2}$$

Proof. Let θ be the initial parameters. Let $\theta_b^+ := \theta - \eta\, g_b$ and $\theta_c^+ := \theta - \eta\, g_c$ denote the updated shared parameters using gradient of task b and c, respectively. From inter-task affinity, we have

$$\mathcal{Z}_{b \rightsquigarrow a} = 1 - \frac{L_a(\theta_b^+)}{L_a(\theta)} \ge 1 - \frac{L_a(\theta_c^+)}{L_a(\theta)} = \mathcal{Z}_{c \rightsquigarrow a}$$

Thus, we have

$$L_a(\theta_b^+) \le L_a(\theta_c^+)$$

From the strong convexity and strong smoothness assumptions, we can respectively lower-bound and upper-bound the first and second terms. Thus, we have

$$L_a(\theta) - \eta\, g_a \cdot g_b + \frac{\alpha\,\eta^2}{2}\,\|g_b\|^2 \le L_a(\theta) - \eta\, g_a \cdot g_c + \frac{L\,\eta^2}{2}\,\|g_c\|^2$$

Rearranging the terms yields the result. □

A.2 Quadratic Counterexample

We provide a counterexample in a multi-task setup using a quadratic loss function. The loss function for the task a is defined as $\mathcal{L}_a(x_1, x_2) = {}^1\!/_2\,(x_1^2 + 10\,x_2^2)$. For this loss function, $\alpha = 1$ and $\beta = 10$. Also, the global minimum of the loss is at $(x_1, x_2) = (0, 0)$. We set the initial point to $(x_1, x_2) = (-2, -1)$ with a loss value of 7. Figure 3(a) shows the level sets of the loss function $\mathcal{L}_a$, along with the negative task gradient $-g_a$. We also plot two additional negative gradients, namely $-g_b$ and $-g_c$, belonging to the tasks b and c, respectively. The auxiliary task gradients g_b and g_c are chosen to have the same gradient norm as g_a, but pointed along the vectors $[8, -2]$ and $[-12, 2]$, respectively. Using a learning rate of $\eta = 0.09 < {}^1\!/_\beta$, the gradient of task b at this point reduces the value of the loss $\mathcal{L}_a$ more so than the gradient of task c. Specifically, the value of the loss after a gradient step using g_b amounts to 6.96 whereas using g_c, we obtain a loss value of 6.98. However, this ordering does not hold when combining the gradients, i.e. using the combined gradient $g_a + g_b$ results in a loss value 6.09 of whereas the combined gradient $g_a + g_c$ yields a loss of 6.07 (Figure 3(b)).

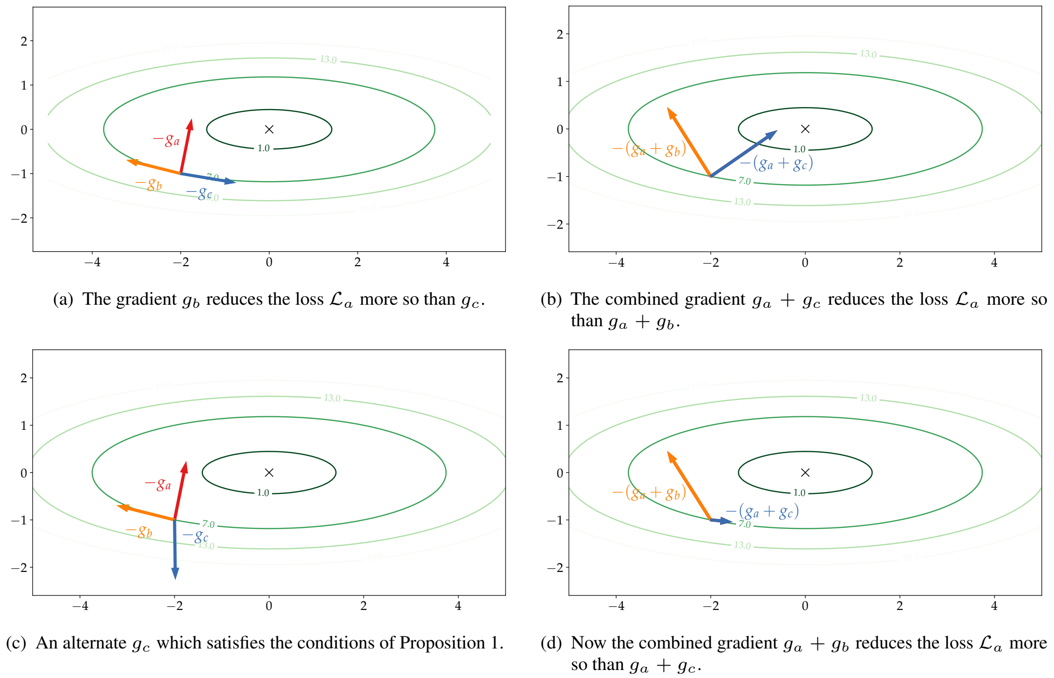
Alternatively, in Figure 3(c) we consider a gradient g_c along $[-0.2, 15]$ which also satisfies the second condition of Proposition 1, namely $\cos(a, c) \le \frac{\eta}{4}\frac{\beta/\alpha}{\beta/\alpha - 1}$ (while $\|g_a\| = \|g_c\|$). For this choice of g_c and as a result of Proposition 1, the fact that g_b reduces the loss $\mathcal{L}_a$ more so than g_c (6.96 vs 7.96) implies $g_a + g_b$ also reduces the loss more so than $g_a + g_c$ (6.09 vs 6.98, see Figure 3(d)).

B Additional Experimental Results

We provide additional experimental results to supplement our empirical analysis of TAG in Section 5. In particular, we evaluate task groupings without a fixed inference-time latency constraint, supply extra information related to the CelebA and Taskonomy analyses, and offer additional discussion on the ablation studies presented in Section 5.2.

B.1 Task Grouping Evaluation Without Latency Constraint

We analyze the effect of removing the inference-time latency constraint from our problem definition. Without this limitation, we can scale up the size of our multi-task learning baselines to equal the total



(a) The gradient g_b reduces the loss $\mathcal{L}_a$ more so than g_c.

(b) The combined gradient $g_a + g_c$ reduces the loss $\mathcal{L}_a$ more so than $g_a + g_b$.

(c) An alternate g_c which satisfies the conditions of Proposition 1.

(d) Now the combined gradient $g_a + g_b$ reduces the loss $\mathcal{L}_a$ more so than $g_a + g_c$.

Figure 3: A counterexample on a quadratic loss function where the task grouping based on inter-task similarity results in an inferior performance.

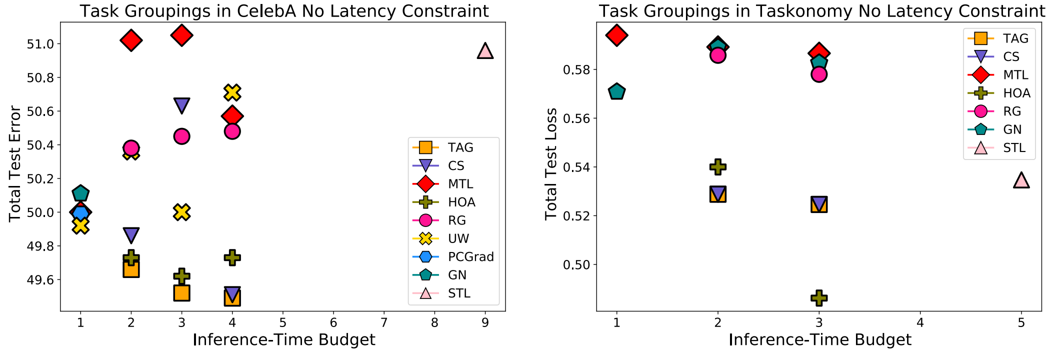


Figure 4: (Left) average classification error for 2, 3, and 4-split task groupings for the subset of 9 tasks in CelebA. (Right) total test loss for 2 and 3-split task groupings for the subset of 5 tasks in Taskonomy. The x-axis is the inference-time budget relative to the number of parameters in the baseline MTL model from Section 5.

number of parameters used in each task grouping. Similar to [46], we choose to scale capacity by increasing the number of channels in each convolutional layer. A 2-splits task grouping would then correspond with a multi-task model with 2 times the number of channels in each conv layer. On the CelebA dataset, running PCGrad with double the number of channels resulted in an out of memory (OOM) error on a 16 GB TeslaV100 GPU. When implemented with distributed training, we received a runtime error. As a result, we do not include PCGrad results in this analysis, and this particular method would also likely surpass typical computational budget constraints due to its high memory usage.

Our results are summarized in Figure 4. Similar to the findings presented in Figure 2 of Section 5, the task grouping approaches continue to outperform simply training all tasks together, as well as optimization augmentations like Uncertainty Weights and GradNorm. For CelebA, increasing the number of channels in the layers of our ResNet model actually reduces performance, indicating our

CelebA Baseline Methods

Method	a1	a2	a3	a4	a5	a6	a7	a8	a9	Total Error
MTL	6.54 ± 0.026	11.09 ± 0.009	4.19 ± 0.017	12.59 ± 0.085	2.60 ± 0.0.003	2.73 ± 0.128	4.81 ± 0.010	4.74 ± 0.012	0.70 ± 0.007	50.00
STL	6.56 ± 0.003	11.37 ± 0.009	4.19 ± 0.025	12.58 ± 0.102	2.69 ± 0.017	3.06 ± 0.010	4.97 ± 0.006	4.83 ± 0.010	0.71 ± 0.007	49.99
UW [28]	6.51 ± 0.038	11.43 ± 0.034	4.18 ± 0.015	11.91 ± 0.132	2.50 ± 0.028	2.95 ± 0.010	4.81 ± 0.026	4.89 ± 0.028	0.74 ± 0.007	49.92
GradNorm [10]	6.44 ± 0.033	11.09 ± 0.0.021	4.01 ± 0.051	12.38 ± 0.097	2.65 ± 0.022	2.96 ± 0.017	4.89 ± 0.015	4.87 ± 0.003	0.81 ± 0.009	50.11
PCGrad [53]	6.57 ± 0.015	10.95 ± 0.020	4.04 ± 0.015	12.73 ± 0.033	2.67 ± 0.013	2.87 ± 0.021	4.76 ± 0.006	4.76 ± 0.015	0.64 ± 0.010	49.99

Table 4: Mean and standard error for benchmark methods run on CelebA.

Inference Time Budget = 2 Splits Task Groupings

Method	Splits	a1	a2	a3	a4	a5	a6	a7	a8	a9	Total Error
CS	group 1	—	—	—	—	**2.60 ± 0.010**	**3.05 ± 0.007**	—	—	—	49.86
	group 2	**6.55 ± 0.009**	**11.19 ± 0.020**	**4.10 ± 0.012**	**12.02 ± 0.029**	2.57 ± 0.007	—	**4.78 ± 0.015**	**4.85 ± 0.006**	**0.73 ± 0.010**	
HOA	group 1	—	—	—	—	**2.59 ± 0.006**	—	**4.71 ± 0.000**	—	—	49.73
	group 2	**6.49 ± 0.037**	**11.34 ± 0.022**	**4.25 ± 0.052**	**11.76 ± 0.090**	—	**3.00 ± 0.022**	—	**4.91 ± 0.059**	**0.69 ± 0.009**	
TAG	group 1	**6.39 ± 0.006**	—	—	—	—	—	**4.79 ± 0.006**	—	—	49.66
	group 2	—	**11.10 ± 0.065**	**4.16 ± 0.003**	**12.29 ± 0.202**	**2.55 ± 0.025**	**2.94 ± 0.015**	—	**4.69 ± 0.026**	**0.74 ± 0.013**	
Optimal	group 1	**6.60 ± 0.009**	11.21 ± 0.017	4.40 ± 0.007	11.91 ± 0.051	**2.60 ± 0.009**	**2.87 ± 0.003**	**4.81 ± 0.015**	**4.58 ± 0.009**	—	49.37
	group 2	—	**11.14 ± 0.044**	**4.03 ± 0.012**	**11.90 ± 0.020**	—	—	—	—	**0.75 ± 0.018**	

Table 5: Two-split task groupings in CelebA. We report mean and standard error.

Inference Time Budget = 3 Splits Task Groupings

Method	Splits	a1	a2	a3	a4	a5	a6	a7	a8	a9	Total Error
CS	group 1	—	—	—	—	**2.60 ± 0.010**	**3.05 ± 0.007**	—	—	—	50.63
	group 2	**6.39 ± 0.006**	—	—	—	—	—	**4.79 ± 0.006**	—	—	
	group 3	—	**11.20 ± 0.031**	**4.09 ± 0.010**	**12.97 ± 0.015**	—	—	—	**4.82 ± 0.018**	**0.73 ± 0.003**	
HOA	group 1	—	—	—	—	**2.59 ± 0.006**	—	**4.71 ± 0.000**	—	—	49.73
	group 2	—	**11.08 ± 0.017**	—	**12.20 ± 0.067**	—	—	—	—	—	
	group 3	**6.55 ± 0.012**	—	**4.15 ± 0.013**	—	2.70 ± 0.012	**2.84 ± 0.003**	4.90 ± 0.015	**4.76 ± 0.015**	**0.75 ± 0.013**	
TAG	group 1	**6.39 ± 0.006**	—	—	—	—	—	**4.79 ± 0.006**	—	—	49.52
	group 2	—	**11.08 ± 0.017**	—	**12.20 ± 0.067**	—	—	—	—	—	
	group 3	—	11.11 ± 0.127	**4.08 ± 0.006**	—	**2.52 ± 0.025**	**2.96 ± 0.050**	4.98 ± 0.058	**4.73 ± 0.015**	**0.78 ± 0.009**	
Optimal	group 1	6.34 ± 0.067	—	—	—	**2.57 ± 0.012**	**2.91 ± 0.019**	4.74 ± 0.031	**4.70 ± 0.017**	0.78 ± 0.012	48.92
	group 2	—	**11.14 ± 0.044**	**4.03 ± 0.012**	**11.90 ± 0.020**	—	—	—	—	**0.75 ± 0.018**	
	group 3	**6.25 ± 0.045**	—	—	—	—	—	**4.67 ± 0.009**	—	0.71 ± 0.006	

Table 6: Three-split task groupings in CelebA. We report mean and standard error.

Inference Time Budget = 4 Splits Task Groupings

Method	Splits	a1	a2	a3	a4	a5	a6	a7	a8	a9	Total Error
CS	group 1	—	—	—	—	**2.60 ± 0.010**	**3.05 ± 0.007**	—	—	—	49.51
	group 2	**6.39 ± 0.006**	—	—	—	—	—	**4.79 ± 0.006**	—	—	
	group 3	—	**11.08 ± 0.017**	—	**12.20 ± 0.067**	—	—	—	—	—	
	group 4	—	11.00 ± 0.013	**4.07 ± 0.012**	12.40 ± 0.045	—	—	4.96 ± 0.031	**4.62 ± 0.009**	**0.72 ± 0.007**	
HOA	group 1	—	—	—	—	**2.59 ± 0.006**	—	**4.71 ± 0.000**	—	—	49.73
	group 2	—	**11.08 ± 0.017**	—	**12.20 ± 0.067**	—	—	—	—	—	
	group 3	**6.63 ± 0.024**	—	—	—	**2.70 ± 0.006**	—	—	—	—	
	group 4	—	11.25 ± 0.049	**4.15 ± 0.024**	—	2.58 ± 0.023	**2.88 ± 0.045**	**4.90 ± 0.065**	**4.74 ± 0.026**	**0.76 ± 0.012**	
TAG	group 1	**6.39 ± 0.006**	—	—	—	—	—	**4.79 ± 0.006**	—	—	49.49
	group 2	—	**11.08 ± 0.017**	—	**12.20 ± 0.067**	—	—	—	—	—	
	group 3	—	—	—	—	**2.63 ± 0.003**	**2.99 ± 0.015**	4.75 ± 0.012	—	—	
	group 4	—	11.00 ± 0.013	**4.07 ± 0.012**	12.40 ± 0.045	—	—	4.96 ± 0.031	**4.62 ± 0.009**	**0.72 ± 0.007**	
Optimal	group 1	—	11.20 ± 0.049	4.09 ± 0.022	**11.77 ± 0.197**	—	—	—	—	—	48.57
	group 2	—	—	**4.00 ± 0.023**	—	—	**2.90 ± 0.020**	4.85 ± 0.041	—	**0.75 ± 0.010**	
	group 3	**6.25 ± 0.045**	—	—	—	—	—	**4.67 ± 0.009**	—	0.71 ± 0.006	
	group 4	—	**10.96 ± 0.045**	—	12.74 ± 0.136	**2.56 ± 0.031**	—	—	**4.75 ± 0.072**	0.77 ± 0.022	

Table 7: Four-split task groupings in CelebA. We report mean and standard error.

model is already at near optimal capacity for this dataset. This is reasonable given our tuning of the CelebA model architecture and hyperparameters to maximize MTL performance. For Taskonomy, and similar to the results presented in [46], we find scaling multi-task model capacity does not meaningfully improve MTL or GradNorm performance.

B.2 Additional CelebA Task Grouping Results

In this section, we provide further detail into our experimental results for the CelebA dataset. We present the raw values used to create Figure 2 (left) and Figure 4 (left) as well as underpin the Section 5.2 ablation studies in Table 4, Table 5, Table 6, Table 7, and Table 8. All quantities are averaged across three independent runs, and we report mean and standard error. A task within a group is highlighted in bold when this task is chosen to “serve” from this assigned task grouping. Duplicate tasks that are not bolded are only used to assist in the training other tasks.

CelebA High Capacity Performance											
Method	Capacity	a1	a2	a3	a4	a5	a6	a7	a8	a9	Total Error
MTL	2x	6.42 ± 0.007	10.76 ± 0.070	4.35 ± 0.010	13.01 ± 0.152	2.66 ± 0.0.015	3.05 ± 0.009	4.76 ± 0.038	5.20 ± 0.047	0.80 ± 0.007	51.02
	3x	6.43 ± 0.032	11.65 ± 0.128	4.21 ± 0.060	12.69 ± 0.918	2.61 ± 0.029	2.94 ± 0.009	4.87 ± 0.067	4.86 ± 0.064	0.80 ± 0.034	51.05
	4x	6.91 ± 0.055	11.23 ± 0.060	4.31 ± 0.032	12.51 ± 0.104	2.57 ± 0.021	3.01 ± 0.009	4.59 ± 0.036	4.81 ± 0.018	0.64 ± 0.012	50.57
UW	2x	6.40 ± 0.006	11.01 ± 0.023	4.36 ± 0.015	12.54 ± 0.022	2.63 ± 0.0.013	3.03 ± 0.003	4.68 ± 0.007	5.00 ± 0.015	0.71 ± 0.012	50.36
	3x	6.36 ± 0.050	11.32 ± 0.063	4.27 ± 0.020	12.12 ± 0.019	2.51 ± 0.012	2.94 ± 0.037	4.93 ± 0.045	4.76 ± 0.037	0.79 ± 0.006	50.00
	4x	6.70 ± 0.146	11.49 ± 0.107	4.37 ± 0.009	12.35 ± 0.120	2.63 ± 0.021	3.07 ± 0.057	4.61 ± 0.023	4.83 ± 0.013	0.64 ± 0.018	50.71
GN	2x	6.38 ± 0.038	10.98 ± 0.031	4.13 ± 0.029	12.57 ± 0.095	2.71 ± 0.0.000	3.00 ± 0.043	4.73 ± 0.009	5.21 ± 0.146	0.86 ± 0.019	50.57
	3x	6.38 ± 0.021	11.67 ± 0.217	4.35 ± 0.075	12.35 ± 0.248	2.61 ± 0.020	3.00 ± 0.089	4.93 ± 0.056	4.86 ± 0.006	0.85 ± 0.037	51.00
	4x	6.56 ± 0.063	11.49 ± 0.261	4.33 ± 0.030	13.29 ± 0.646	2.57 ± 0.052	3.07 ± 0.067	4.82 ± 0.087	4.82 ± 0.057	0.78 ± 0.003	51.74

Table 8: Mean and standard error for CelebA high capacity experiments.

Baseline Methods on Taskonomy						
Method	s	d	n	t	k	Total Test Loss
MTL	0.0586	0.2879	0.1076	0.0428	0.1079	0.5940
STL	0.0509	0.2616	0.0975	0.0337	0.0910	0.5347
GN	0.0542	0.2818	0.1011	0.0305	0.1032	0.5708

Table 9: Baseline methods in Taskonomy.

Inference Time Budget = 2 Splits Task Groupings							
Method	Splits	s	d	n	t	k	Total Test Loss
CS	group 1	**0.532**	**0.2527**	**0.1064**	—	—	0.5288
	group 2	—	—	—	**0.0232**	**0.0933**	
HOA	group 1	**0.0603**	**0.2725**	**0.1075**	**0.0429**	—	0.5400
	group 2	—	—	0.1110	—	**0.0568**	
TAG	group 1	**0.532**	**0.2527**	**0.1064**	—	—	0.5288
	group 2	—	—	—	**0.0232**	**0.0933**	
Optimal	group 1	**0.0532**	**0.2527**	**0.1064**	—	—	
	group 2	—	—	0.1096	**0.0271**	**0.0750**	0.5176

Table 10: Two-split task groupings in Taskonomy.

Inference Time Budget = 3 Splits Task Groupings							
Method	Splits	s	d	n	t	k	Total Test Loss
CS	group 1	**0.528**	0.2636	—	—	—	0.5246
	group 2	—	—	—	**0.0232**	**0.0933**	
	group 3	—	**0.2551**	0.1002	—	—	
HOA	group 1	**0.0532**	**0.2527**	**0.1064**	—	—	0.4923
	group 2	—	—	0.1110	—	**0.0568**	
	group 3	—	—	—	**0.0232**	0.0933	
TAG	group 1	**0.528**	0.2636	—	—	—	0.5246
	group 2	—	—	—	**0.0232**	**0.0933**	
	group 3	—	**0.2551**	0.1002	—	—	
Optimal	group 1	0.0532	**0.2527**	0.1064	—	—	0.4862
	group 2	**0.0500**	—	**0.1025**	**0.0242**	—	
	group 3	—	—	0.1110	—	**0.0568**	

Table 11: Three-split task groupings in Taskonomy.

B.3 Additional Taskonomy Task Grouping Results

We also report the raw scores used in our empirical analysis of Taskonomy to create Figure 2 (right) and Figure 4 (right) in Table 9, Table 10, Table 11, and Table 12. Given the size of the Taskonomy dataset, and computational cost associated with running a single model (approximately 146 Tesla V100 GPU hours for the MTL baseline), we evaluate only a single run.

B.4 Supplementary Information on Ablation Studies

Does our measure of inter-task affinity correlate with optimal task groupings? Expanding on our analysis in Section 5.2, we present the difference in normalized inter-task affinity values for all attributes in our CelebA analysis in Table 13. Note the affinity onto a8 is significantly less than the affinity onto any other task and 4 times smaller than the next smallest difference. We posit the

Taskonomy High Capacity Performance						
Method	s	d	n	t	k	Total Test Loss
MTL (2x)	0.00536	0.2664	0.1078	0.0441	0.11.73	0.5892
MTL (3x)	0.0538	0.2891	0.1090	0.0363	0.0984	0.5866
GN (2x)	0.0552	0.2977	0.1013	0.0289	0.1060	0.5891
GN (3x)	0.0570	0.2806	0.1044	0.0401	0.1006	0.5827

Table 12: Performance of high capacity models on Taskonomy.

reason TAG fails to select a positive group for a8 is due to the fact that no other task significantly decreases (or increases) the loss of a8 throughout the course of training. Tasks which exhibit similar characteristics are also likely to be difficult cases for TAG when finding a task's best co-training partner.

Is Change in Train Loss Comparable to Change in Validation Loss?
From Section 5.2, we show $\mathcal{Z}^t_{i \rightsquigarrow j}$ computed on the training set is approximately equal to $\mathcal{Z}^t_{i \rightsquigarrow j}$ computed on the validation set when $i \neq j$:

$$\begin{aligned}\mathbb{E}_{\text{val}}[\mathbb{E}_{\text{tr}}[\mathcal{L}_j(X^t_{\text{val}}, \theta^{t+1}_{s|i}, \theta^t_j)]] &\approx \mathbb{E}_{\text{val}}[\mathcal{L}_j(X^t_{\text{val}}, \mathbb{E}_{\text{tr}}[\theta^{t+1}_{s|i}], \theta^t_j)] \\ &= \mathbb{E}_{\text{tr}}[\mathcal{L}_j(X^t_{\text{tr}}, \mathbb{E}_{\text{tr}}[\theta^{t+1}_{s|i}], \theta^t_j)]\,.\end{aligned}$$

A corollary to this result is the above approximation does not hold when $j = i$: comparing task i's capacity to decrease it's own train loss is not comparable with task j's capacity to decrease the train loss of task i. As a result, TAG acting on the training dataset will never group a task by itself. While we find a single-task group is never optimal in any of our experiments, one can tradeoff a small decrease in efficiency from loading the validation dataset during training with the capacity of TAG to select single-task groupings.

Task	max - min
a1	0.35
a2	0.39
a3	0.16
a4	0.31
a5	0.47
a6	0.31
a7	0.43
a8	**0.04**
a9	0.17

Table 13: Difference in normalized inter-task affinity onto each task. Notice the affinity onto $a8$ is **significantly** less than the affinity onto any other task.

B.5 Experimental Design

In this section, we aim to accurately and precisely describe our experimental design to facilitate reproducibility. We also release our code in the supplementary material to supplement this written explanation.

B.6 CelebA

We accessed the CelebA dataset publicly available on TensorFlow datasets https://tensorflow.org/datasets/catalog/celeb_a under an Apache 2.0 license and filtered the 40 annotated attributes down to a set of 9 attributes for our analysis. Our experiments were run on a combination of Keras [12] and TensorFlow [1].

The encoder architecture is based loosely on ResNet 18 [20] with task-specific decoders being composed of a single projection layer. A coarse architecture search revealed adding additional layers to the encoder and decoder did not meaningfully improve model performance. A learning rate of 0.0005 is used for 100 epochs, with the learning rate being halved every 15 epochs. The learning rate was tuned on the validation split of the CelebA dataset over the set of $\{0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01\}$. GradNorm [10] alpha was determined by searching over the set of {0.01, 0.1, 0.5, 1.0, 1.5, 2.0, 3.0, 5.0}, and choosing the alpha with the highest total accuracy on the validation set.

We train until the validation increases for 10 consecutive epochs, load the parameters from the best validation checkpoint, and evaluate on the test set. We use the splits default to TensorFlow datasets of (162,770, 19,867, 19,962) for (Train, Valid, Test). As each model trains for a varying number of epochs, we report the worst-case runtime in Figure 2(left) which approximates the time required to train each method when the model is trained for the full 100 epochs. This is similar to the method in [46] which trains to completion, loads the best weights, and then evaluates on the test set. We choose 100 epochs as our setup mirrors that of [45] on CelebA with the exception of adding an early stopping condition.

B.7 Taskonomy

Our experiments mirror the settings and hyperparameters of "Setting 2" in [46] by directly implementing TAG and its approximation in the framework provided by the author's official code release (https://github.com/tstandley/taskgrouping at hash dc6c89c269021597d222860406fa0fb81b02a231). The encoder is a modified Xception Network and each task-specific decoder consists of four transposed convolutional layers and four convolutional layers. Further information regarding network specifications and training details can be found in [46].

To mitigate computational requirements and increase accessibility, we replace the 12 TB full+ Taskonomy split used by [46] with an augmented version of the medium Tasknomy split by filtering out buildings with corrupted images and adding additional buildings to replace the corrupted ones. We download the Taskonomy dataset from the official repository (https://github.com/StanfordVL/taskonomy) created by [54] which is released under an MIT license. The final size of our medium+ taskonomy split is approximately 2.4 TB. The list of buildings used in our analysis is encapsulated within our released code. We reuse the implementation of GradNorm in [46] and follow their settings of $\alpha = 1.5$.

While [46] load the parameters with lowest validation loss into the model before evaluating on the test set, their early-stopping window size is equal to the total number of epochs. Therefore, each model trains for a full 100 epochs, irrespective of whether the lowest validation loss occurred early or late into training. Accordingly, the runtimes in Figure 2 (right) is the time taken by each cloud instance to completely train the model to 100 epochs.