ADAPTING AUXILIARY LOSSES USING GRADIENT SIMILARITY

Anonymous authors
Paper under double-blind review

ABSTRACT

One approach to deal with the statistical inefficiency of neural networks is to rely on auxiliary losses that help building useful representations. However is not always trivial to know if an auxiliary task will be helpful for the main task and when it could start hurting. We explore using gradient cosine similarity as an adaptive weight for the auxiliary loss, and demonstrate the usefulness of the proposed algorithm in a few domains, including multi-task supervised learning using subsets of ImageNet, and reinforcement learning using Atari games. Additionally, we show that our approach is guaranteed to converge to critical points of the main task. This is not guaranteed otherwise, and in principle adding a mis-matched auxiliary loss can lead to divergence on the main task.

1 INTRODUCTION

Neural networks are extremely powerful function approximators that have excelled on a wide range of tasks (Simonyan and Zisserman, 2015; Mnih et al., 2015; He et al., 2016a; Silver et al., 2016; Vaswani et al., 2017). Despite state of the art results across domains, they remain data-inefficient and expensive to train. In image classification, large deep learning (DL) benchmarks like ImageNet (Russakovsky et al., 2015), with a million or more examples are needed. In reinforcement learning (RL), agents typically consume millions of frames of experience before learning to act in complex environments (Silver et al., 2016; Espeholt et al., 2018).

Collecting enough data can be expensive, sometimes prohibitively so. For reinforcement learning, this translates into consuming more environment frames, which puts pressure on compute power, the speed of the environment simulator and, in the worst case (e.g. robotics), could make the approach impractical. For supervised learning, it has the additional implication of requiring human intervention to label a larger dataset which can be expensive.

Different techniques have been studied for improving data efficiency, from data augmentation (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Hauberg et al., 2016) to transfer learning (Taylor and Stone, 2009; Pan et al., 2010). In this work we focus on a particular setup for transfer learning. We assume that besides a main task, one has access to multiple auxiliary tasks that share some unknown structure with the main task. In order to improve data efficiency, these additional tasks can be used as auxiliary losses. However, while the model is trained simultaneously on all these tasks, only the performance on the main task is of interest. Any improvement on the auxiliary losses is useful only to the extent that it helps learning features or behaviours for the main task.

Auxiliary tasks have been to shown work well in practice (e.g Zhang et al., 2016; Jaderberg et al., 2017; Mirowski et al., 2017; Papoudakis et al., 2018). However, their success depends on how well aligned the auxiliary losses are with the main task. Knowing this apriori is typically non-trivial, and the usefulness of an auxiliary task can change through the course of training. In this work, we explore a simple yet effective approach for measuring similarity between an auxiliary task and the task of interest given the value of the parameters, and show that this measure can be used to decide which auxiliary losses are helpful and for how long.

1.1 NOTATION AND PROBLEM DESCRIPTION

Assume we have a main task $\mathcal{T}_{main}$ and auxiliary task $\mathcal{T}_{aux}$ that induce two losses $\mathcal{L}_{main}$ and $\mathcal{L}_{aux}$, where we care about only about maximizing performance on $\mathcal{T}_{main}$ (main task), while $\mathcal{T}_{aux}$

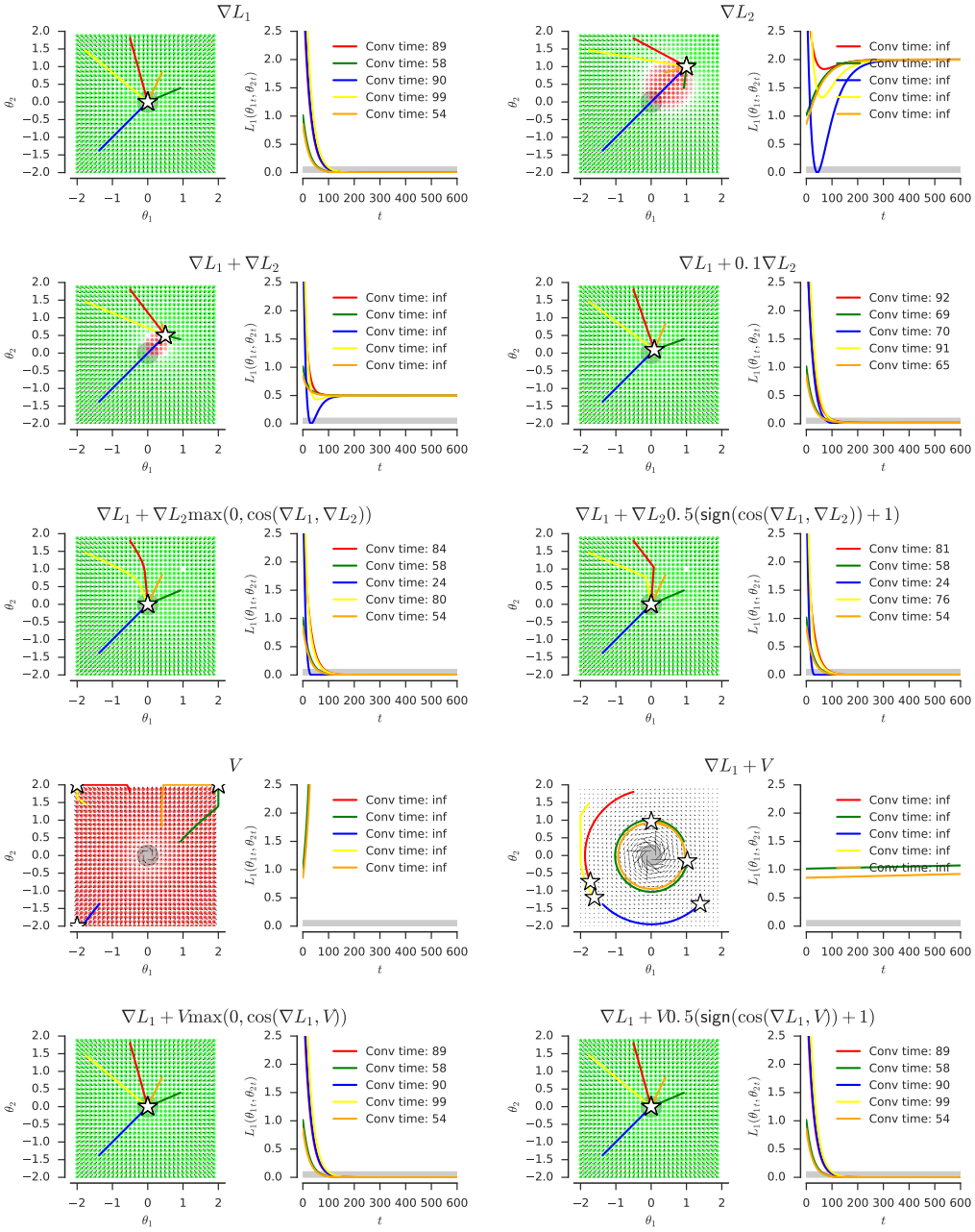


Figure 1: Example optimization for $L_1(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$, $L_2(\theta_1, \theta_2) = (\theta_1 - 1)^2 + (\theta_2 - 1)^2$ and $V(\theta_1, \theta_2) = [-\frac{\theta_2}{\theta_1^2+\theta_2^2} - 2\theta_1, \frac{\theta_1}{\theta_1^2+\theta_2^2} - 2\theta_2]$. Each colored trajectory represents one optimization run with random initial position. Star represents the convergence point. All experiments run with constant step size of 0.01 and 600 iterations of steepest descent method. Convergence time is defined as number of steps needed to get below 0.1 loss of L_1 (gray region). Color of each point represents its alignment with ∇L_1 (green – positive alignment, red – negative alignment, white – directions are perpendicular). Note that in this example L_2 is helpful for L_1 as it reinforces good descent directions in most of the space. However, simple mixing is actually slowing optimization down (or makes it fail completely), while the proposed methods converge, and usually do so faster. When using non-conservative vector field V one obtains lack of convergence (cyclic behaviour), while the proposed merging still works well.

is an auxiliary task which is not of direct interest. Note that this is different from multi-objective optimization in which both tasks are of interest. We propose to parametrize the solution for $\mathcal{T}_{main}$ and $\mathcal{T}_{aux}$ by two neural networks $f(\cdot, \boldsymbol{\theta}, \boldsymbol{\phi}_{main})$ and $g(\cdot, \boldsymbol{\theta}, \boldsymbol{\phi}_{aux})$ such that they share a subset of parameters denoted here by $\boldsymbol{\theta}$. Generally, the auxiliary loss literature proposes to minimize

$$\underset{\boldsymbol{\theta}, \boldsymbol{\phi}_{main}, \boldsymbol{\phi}_{aux}}{\arg\min} \quad \mathcal{L}_{main}(\boldsymbol{\theta}, \boldsymbol{\phi}_{main}) + \lambda \mathcal{L}_{aux}(\boldsymbol{\theta}, \boldsymbol{\phi}_{aux}) \tag{1}$$

under the intuition that modifying $\boldsymbol{\theta}$ to minimize $\mathcal{L}_{aux}$ will improve $\mathcal{L}_{main}$ if the two tasks are sufficiently related. We propose to modulate the weight λ at each learning iteration t by how useful $\mathcal{T}_{aux}$ is for $\mathcal{T}_{main}$ given $\boldsymbol{\theta}^{(t)}, \boldsymbol{\phi}^{(t)}_{main}, \boldsymbol{\phi}^{(t)}_{aux}$. That is, at each optimization iteration, we want to efficiently approximate the solution to

$$\underset{\lambda^{(t)}}{\arg\min}\, \mathcal{L}_{main}\left(\boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}}(\mathcal{L}_{main} + \lambda^{(t)}\mathcal{L}_{aux}), \boldsymbol{\phi}^{(t)}_{main} - \alpha \nabla_{\boldsymbol{\phi}_{main}} \mathcal{L}_{main}\right). \tag{2}$$

Note that the input space of $\mathcal{T}_{main}$ and $\mathcal{T}_{aux}$ do not have to match, and in particular $\mathcal{T}_{aux}$ does not need to be defined for an input of $\mathcal{T}_{main}$ or the other way around.[1] Solving equation 2 is expensive. Instead, we are looking for a cheap heuristic to approximate $\lambda^{(t)}$ which is better than keeping $\lambda^{(t)}$ constant and does not require hyper-tuning.

2 Cosine Similarity Between Gradients of Tasks

We propose to use gradient cosine similarity as a measure of task similarity and hence for approximating $\lambda^{(t)}$. Consider an example where the target function that we wish to minimize is $\mathcal{L}_{main} = (x - 10)^2$, and the auxiliary function is $\mathcal{L}_{aux} = x^2$. Assume we initialize x at $x = -20$; at this point, the gradients of the target and auxiliary function point in the same direction and the cosine similarity is 1, indicating that minimizing the auxiliary loss is beneficial for minimizing the target loss. However, at a different point $x = 5$, the gradients of the auxiliary function and the target function point in different directions and the cosine similarity is -1; hence minimizing the auxiliary loss would hinder minimizing the target loss. See Figure 7 in Appendix B for an illustration.

This example suggests a natural strategy for approximating $\lambda^{(t)}$: *minimize the auxiliary loss as long as the auxiliary gradient has non-negative cosine similarity with the target gradient, and ignore the auxiliary loss when the auxiliary gradient has negative cosine similarity with the target gradient*. This follows the well known intuition that if a vector is in the same half-space as the gradient then it is a decent direction for f, which reduces our strategy to asking if the gradient of the auxiliary loss is a *descent direction* for the loss of interest.

Proposition 1. *Given any gradient vector field $G(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ and any vector field $V(\boldsymbol{\theta})$ (such as gradient of another loss function, but could be arbitrary set of updates). Given update rule of form*

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \alpha^{(t)}(G(\boldsymbol{\theta}^{(t)}) + V(\boldsymbol{\theta}^{(t)}) \max(0, \cos(G(\boldsymbol{\theta}^{(t)}), V(\boldsymbol{\theta}^{(t)})))$$

converges to the local minimum of $\mathcal{L}$ given small enough $\alpha^{(t)}$.

Proof is provided in Appendix A.1.

Note, that the above statement does not guarantee any improvement of convergence, but only guarantees lack of divergence. In particular, cosine similarity is not a silver bullet that guarantees transfer, it just guarantees dropping the "worst case scenarios". One can, in principle, create toy examples where convergence is affected in an arbitrary way (both positively, see figure 1, and negatively see Appendix D). Nevertheless, as the proposition shows, convergence on the target task is guaranteed.

Note, that simple adding of an arbitrary vector field does not have this property, for example choosing $V(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) + \left[-\frac{\theta_2}{\theta_1^2+\theta_2^2}, \frac{\theta_1}{\theta_1^2+\theta_2^2}\right]^T$ for two-dimensional case, leads to using an update of form $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \left[-\frac{\theta_2}{\theta_1^2+\theta_2^2}, \frac{\theta_1}{\theta_1^2+\theta_2^2}\right]^T$ which is a non-conservative vector field, causing optimizer

[1] In the supervised learning case, when the input features are shared, this setting resembles the *multi-task learning without label correspondences* setting (Quadrianto et al., 2010).

to follow concentric circles around the origin (see Figure 1). This is particularly important, since there are realistic scenarios, such as in RL, where one does not always deal with gradient fields (e.g. updates of the Q-learning algorithms do not form a gradient field).

Figure 1 provides a few illustrative examples on quadratic functions of the proposed approach, which helps intuitively understand the kind of scenarios for which the approach could help.

The above proposition refers to losses with the same set of parameters $\boldsymbol{\theta}$, while equation 2 refers to the scenario when each loss has task specific parameters (e.g. $\boldsymbol{\phi}_{main}$ and $\boldsymbol{\phi}_{aux}$). The following proposition extends to this scenario:

Proposition 2. *Given two losses parametrized with $\boldsymbol{\Theta}$ (some of which are shared $\boldsymbol{\theta}$ and some unique to each loss $\boldsymbol{\phi}_{main}$ and $\boldsymbol{\phi}_{aux}$), learning rule:*

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \alpha^{(t)}(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}(\boldsymbol{\theta}^{(t)}) + \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}(\boldsymbol{\theta}^{(t)}) \max(0, \cos(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}(\boldsymbol{\theta}^{(t)}), \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}(\boldsymbol{\theta}^{(t)})))$$

$$\boldsymbol{\phi}_{main}^{(t+1)} := \boldsymbol{\phi}_{main}^{(t)} - \alpha^{(t)}\nabla_{\boldsymbol{\phi}_{main}}\mathcal{L}_{main}(\boldsymbol{\Theta}^{(t)}) \quad \textit{and} \quad \boldsymbol{\phi}_{aux}^{(t+1)} := \boldsymbol{\phi}_{aux}^{(t)} - \alpha^{(t)}\nabla_{\boldsymbol{\phi}_{aux}}\mathcal{L}_{aux}(\boldsymbol{\Theta}^{(t)})$$

leads to convergence to local minimum of $\mathcal{L}_{main}$ w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi}_{main})$ given small enough $\alpha^{(t)}$.

Proof. Comes directly from the previous proposition for $G = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}$ and $V = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}$ and the fact that for any vector fields A, B, C we have that $\langle A, B\rangle \geq 0$ and $\langle C, B\rangle \geq 0$ implies $\langle A + C, B\rangle \geq 0$. □

Analogous guarantees hold for the *unweighted version* of this algorithm, where instead of weighting by $\cos(G, V)$ we use $(\text{sign}(\cos(G, V)) + 1)/2$ which is equivalent to using V iff $\cos(G, V) > 0$. Additionally, note that there is no guarantee that $\mathcal{L}_{aux}$ is optimized, e.g. if $\mathcal{L}_{aux} = -\mathcal{L}_{main}$ then $\mathcal{L}_{aux}$ is ignored. Figure 1 visualizes this scenario. When training with mini-batches, accurately estimating $\cos(G, V)$ can be difficult due to mini-batch noise; however, the unweighted version only requires $(\text{sign}(\cos(G, V))$ which can be estimated more robustly. Hence, we use this variant in our experiments (unless otherwise specified).

Despite its simplicity, the update rule proposed can give rise to really interesting phenomena. In particular, we can show that the emerging vector field could be non-conservative and thus it means that there does not exist a loss function, for which it is a gradient. While this might seem problematic, it only describes the global structure, while typically used optimizers are local in nature (they do local, linear or quadratic approximations of the function). Consequently, in practice, one should not expect any negative effects from this phenomena. It simply shows that proposed technique is actually changing qualitatively the nature of updates used for training.

Proposition 3. *In general, update rule proposed does not have to create a conservative vector field.*

Proof is provided in Appendix A.2.

3 Applications of Gradient Cosine Similarity

In this section, we use the gradient cosine similarity to decide when to train on the auxiliary task. All experiments follow the training procedure summarized in Algorithm 1 and 2 in Appendix C.

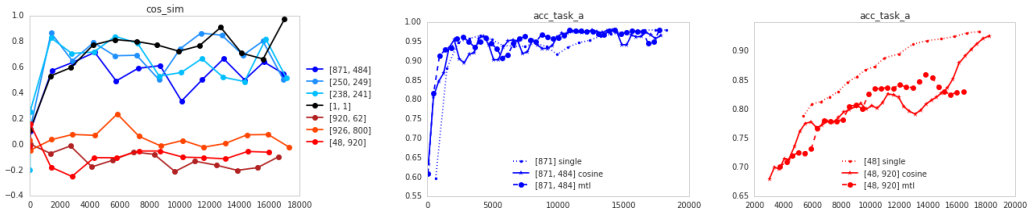
3.1 Experiments on image classification tasks

First, we consider a classification problem on ImageNet (Russakovsky et al., 2015) and design a simple multi-task binary classification task to experiment our hypothesis that *transferable tasks should have high cosine similarity (and vice versa)*. We take a pair of classes from ImageNet, refer to these as class A and class B; all the other 998 classes in ImageNet (except A and B) are referred to as the $background$. Our tasks $\mathcal{T}_{main}$ and $\mathcal{T}_{aux}$ are then formed as a binary classification of *if an image is class A (otherwise $background$)* and *if an image is class B (otherwise $background$)* respectively.

Ideally, we want to pick groups of class pairs that reflect *near* or *far* distance, for the purpose of providing a baseline of *transferability*. Therefore, we used two distance measures, *lowest common ancestor (LCA)* and *Frechet Inception Distance (FID)* (Heusel et al., 2017), to serve as a ground truth of class similarity for selecting class pair A and B. Based on these measures, we picked three pair of classes for *near*, class 871 vs 484, 250 vs 249 and 238 vs 241 and *far*, class 920 vs 62, 926 vs 800,

48 vs 920. From each pair we build $\mathcal{L}_{main}$ and $\mathcal{L}_{aux}$ by solving the binary classification problem. Details about the class pair selection are described in Appendix E.

We use a modified version of ResNetV2-18 (He et al., 2016b) as the training model for this experiment. All parameters in the convolutional layers are shared (denote as $\boldsymbol{\theta}$), followed by task specific branches for $\mathcal{T}_{main}$ and $\mathcal{T}_{aux}$. First, we use a multi-task learning setup and minimize $\mathcal{L}_{main} + \mathcal{L}_{aux}$, and measure cosine similarity on shared parameters through the course of training. Figure 2(a) shows that cosine similarity is higher for close pairs (blue lines) and lower for far pairs (red lines). Next, we compare single-task training, multi-task training and our proposed variant. Figure 2(b) shows that on a near pair, all variants perform similarly. However, as Figure 2(c) shows, multi-task learning leads to poorer performance than single-task learning on target task for a *far* pair due to potential negative transfer whereas our method which uses gradient cosine similarity leads to improved performance.



(a) Cosine similarities on near pairs (blue) and far pairs (red).

(b) Near pair: 871 vs 484

(c) Far pair: 48 vs 920

Figure 2: Multi-task learning setup on ImageNet class pairs. (a): gradient cosine similarity is higher for near pairs and lower for far pairs. (b) and (c): testing accuracy on single task (dotted), naive multi-task (dashed) and our method (solid).

3.2 Experiments on reinforcement learning tasks

Let us now consider a typical RL problem where one aims at finding a policy π that maximizes sum of future discounted rewards, $\mathbb{E}_\pi[\sum_{t'=1}^{N} \gamma^{t'-1} r_{t'}]$, in some partially observable Markov decision process (POMDP). There have been many techniques proposed to solve this optimization problem, from classical policy gradient (Williams, 1992) or Q-Learning (Watkins, 1989) to more modern Proximal Policy Optimization (Schulman et al., 2017) or V-Trace (Espeholt et al., 2018). Inherently, these techniques are usually very data inefficient due to the complexity of the problem being solved. One way to address this issue is to use transfer learning, for example from pre-trained policies (Rusu et al., 2015). However, in the scenario of which a teacher policy is not available for the target task, one can train policies in other tasks that share enough similarities to hope for positive transfer. One way of exploiting this extra information is to use behavioural cloning, or distillation (Hinton et al., 2015; Rusu et al., 2015), to guide the main RL algorithm in its initial learning phase (Schmitt et al., 2018). In practice, it might be difficult to find a suitable strategy that combines the two losses and/or smoothly transition between them. Typically, the teacher policy can be treated as an auxiliary loss (Schmitt et al., 2018) or a prior (Teh et al., 2017) with a fixed mixing coefficient. However, these techniques become unsound if the teacher policy is only helpful in specific states, while in others it leads to a negative transfer.

We propose a simple RL experiment to show that our method is capable of finding the strategy of combining auxiliary loss and the target loss. We define a distribution over 15×15 mazes, where an agent observes its surrounding (up to 4 pixels away) and can move in 4 directions.We randomly place two types of positive rewards, $+5$ and $+10$ points, both terminating an episode. In order to guarantee (expected) finite length of episodes we add fixed probability of 0.01 of transitioning to a non-rewarding terminal state. Full details on the experiment settings are provided in Appendix F. We train a Q-learning agent on such a maze, giving us a *teacher* policy π^{Q}. To create a target task to which there is a possible positive) knowledge transfer, we create a target environment which keeps the same maze layout, but remove the $+10$ rewards (and corresponding states are no longer terminating). Consequently we have two tasks, $\mathcal{T}_{aux}$, where we have strong policy, and $\mathcal{T}_{main}$, the target task. We sample 1,000 such environment pairs and report expected returns obtained (100 evaluation episodes at each evaluation point) using various training regimes.

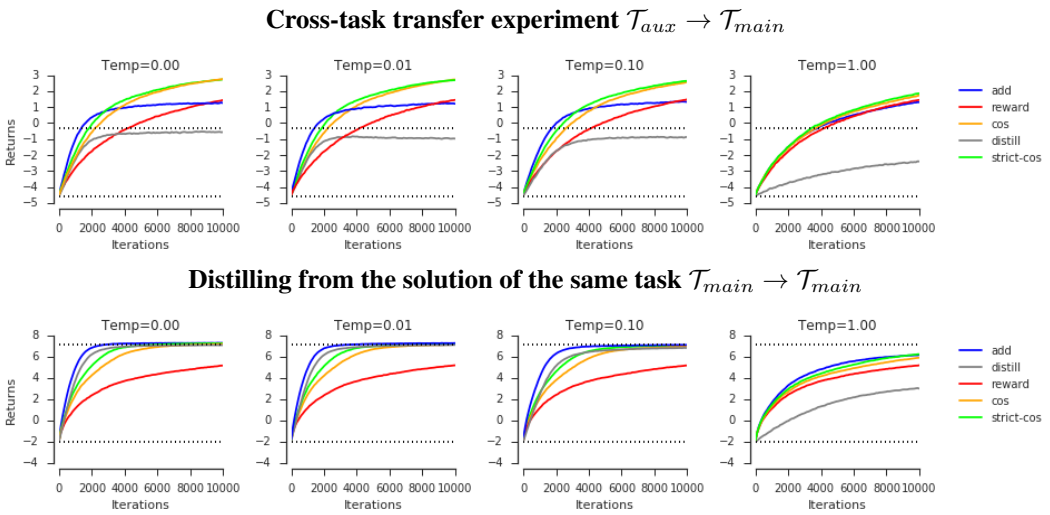


Figure 3: Expected learning curves for cross-environment distillation experiments, averaged across 1,000 partially observable grid worlds. Teacher's policy is based on Q-Learning. Its performance on a new environment (with modified positive rewards) is represented by the top dotted line (bottom one represents random policy score). Each column represents different temperature applied to teacher policy. 0 temperature refer to the original deterministic greedy policy given by Q-Learning. We report five methods: **reward** using just policy gradient in the new task, **distill** using just distillation cost towards teacher, **add** adding the two above, **cos** using soft cosine mixing (Algorithm 2), **strict cos** using hard cosine mixing (Algorithm 1). Bottom row shows what happens when teacher is perfect, so the optimal thing is to trust it everywhere.

One can use any RL method to solve the target task and learn π, here we use episode-level policy gradient (Williams, 1992) with value function as a baseline method. Policies are parametrized as logits $\boldsymbol{\theta}$ of $\pi(a|s) = \frac{\exp(\theta_{s,a})}{\sum_b \exp(\theta_{s,b})} \in [0, 1]$. This baseline gives a score of slightly above **1** point after 10,000 steps (see Figure 3 top row).

To leverage expert policies for $\mathcal{T}_{aux}$, we define the auxiliary loss to be a distillation loss, which is a per-state cross-entropy between teacher's and student's distributions over actions. First, we tested on using solely distillation loss while sampling trajectories from the student. We recover the subset of teacher's behaviours and end up with **0** point—an expected negative transfer as teacher is guiding us to states that are no longer rewarding.

Then, we test simply adding gradients estimated by policy gradient and distillation. The resulting policy learns quickly but saturates at a return of **1** point, showing very limited positive transfer. However, if we use the proposed gradient cosine similarity as the measure of transferability, we get a significant performance boost; learned policies reach baseline performance after just one third of steps and on average obtain **3** points after 10,000 steps[2] (see Appendix F for visualization of tasks and an example solution).

This experiment shows that gradient cosine similarity allows one to use knowledge from other related tasks in an automatic fashion. Agent is simply ignoring teacher signal when it disagrees with policy gradient estimator. If they do agree in terms of which actions to reinforce – teacher logits are used for better replication of useful policies. In particular, in Figure 3 bottom row we present an experiment of transfer between the same task $\mathcal{T}_{main}$. We see that the cosine similarity experiments underperformed that of simply adding the two losses. This is expected as the noise in the gradients make it hard to measure if the two tasks are a good fit or not.

[2]Note that we compute cosine similarity between a distillation gradient and a single sample of the policy gradient estimator, meaning that we are using a high variance estimate of the similarity. For larger experiments one would need to compute running means for reliable statistics.

3.3 EXPERIMENTS ON ATARI

Finally, we consider a similar RL setup on the Atari domain (Bellemare et al., 2013). Analogous to the previous experiment, we first look at training an agent to play a target task (here, Breakout) given a *sub-optimal* solution to the task. In this example, one could again leverage information about the task by distilling (with a Kullback-Leibler (KL) loss) the teacher's behaviour, but solely rely on distillation leads to sub-optimal performance. We use the same convolutional architecture as in previous works (Mnih et al., 2015; 2016; Espeholt et al., 2018; Hessel et al., 2018) and train using the batched actor-critic with V-trace algorithm (Espeholt et al., 2018).

Figure 4 shows that purely distilling from the sub-optimal teacher agent leads to lower performance. Training with both distillation and RL losses leads to slightly better, but also sub-optimal performance. While both approaches learn very quickly, they plateau much lower than the pure RL approach. In our method, the KL penalty is scaled at every time-step by the cosine similarity between the policy gradient and distillation losses; once this falls below a fixed threshold, the loss is 'turned off'. We see that our approach is able to learn quickly at the start but continue fine-tuning with pure RL once the distillation loss is zeroed out. Details on the experiment is provided in Appendix G.

Lastly, we consider the task of training an agent to play two Atari games, Breakout and Ms PacMan, in a multi-task training setup. Ms PacMan is the target task $\mathcal{T}_{main}$ and Breakout is the auxiliary task $\mathcal{T}_{aux}$. Similar to previous experiments, we have access to a teacher agent trained on just Breakout, from which we distill a policy. We consider a distillation loss as was done previously by adding the auxiliary KL loss $\mathcal{L}_{aux}$ between the teacher and student policies, to the RL multi-task loss $\mathcal{L}_{main}$ at every time step. Intuitively, doing this would lead to the agent only able to solve one of the tasks—the one the teacher knows about, as the gradients from distillation loss would interfere with the policy gradient.

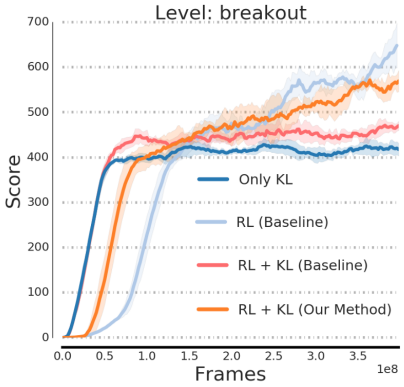


Figure 4: Results on Breakout. We look at the effects of distilling a suboptimal policy as an auxiliary task.

We then evaluate our method on this setting and show results in Figure 5 (averaged over 3 seeds). We see that both the multi-task and the distillation baseline approaches learn one task at the expense of the other. Our method is able to compensate for this by learning Breakout from the teacher and then turning off the distillation; it learns Ms PacMan without forgetting Breakout. We can further see that the evolution of the cosine similarity provides a meaningful cue for the usefulness of $\mathcal{L}_{aux}$.

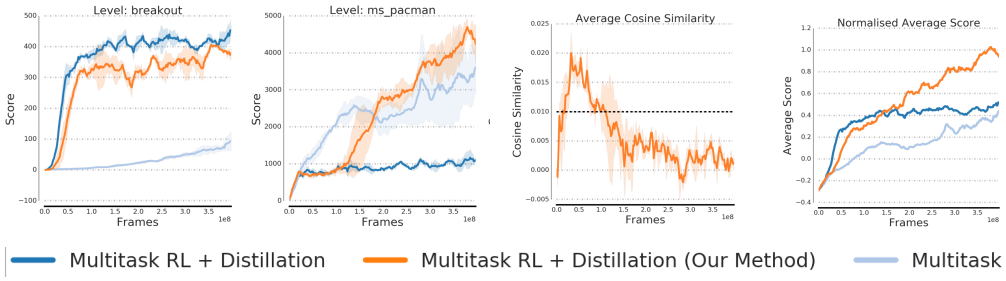


Figure 5: Results on Breakout and Ms Pacman. This experiment aims to jointly learn a model to solve the two distinct tasks. As an auxiliary loss $\mathcal{L}_{aux}$, we distill from a good policy of Breakout. The two plots to the left show performance on Breakout and Ms Pacman respectively. The third plot to the left shows the how cosine similarity between the two tasks changes during training. The last plot shows the multi-task performance, each game is normalized independently based on the best score achieved across all experiments.

4 RELATED WORK

Our work is related to the work on identifying task similarity in transfer learning. It is generally believed that *positive transfer* can be achieved when *source* task(s) and *target* task(s) are *related*. However, it is usually assumed that this relatedness mapping is provided by human experts (Taylor and Stone, 2009; Pan et al., 2010); few works have addressed the problem of finding a general measure of similarity to predict transferability between tasks. In the RL domain, methods have been proposed to use Markov decision process (MDP) similarity as a measure of task relatedness (Carroll and Seppi, 2005; Ammar et al., 2014). Similarly, Yosinski et al. (2014) defined image similarity in ImageNet by manually splitting classes into man-made versus natural objects. While they in some degree capture task similarity, these measures are often domain-specific and not generalizable. More importantly, none of these work have used learned similarity metrics to improve performance. In this work, we propose to use cosine similarity of gradients as a generalizable measure across domains and show it can be directly leveraged to improve the performance of target task(s). One important aspect of task similarity for transfer is that it is highly dependent on the parametrization of the model and current value of the parameters. We exploit this property by providing a heuristic similarity measure for the current parameters, resulting in an approach that relies on an adaptive weight over the updates of the model.

Auxiliary tasks have shown to be beneficial in facilitating learning across domains. In image classification, Zhang et al. (2016) used unsupervised reconstruction tasks. In RL, the UNREAL framework (Jaderberg et al., 2017) incorporates unsupervised control tasks in addition to reward prediction learning as auxiliary tasks. Mirowski et al. (2017) studied auxiliary tasks in the context of navigation. Papoudakis et al. (2018) also explored auxiliary loses for VizDoom. However, these works rely on empirical results and do not address how auxiliary tasks were selected in the first instance. We aim to propose a simple yet effective way of explicitly selecting auxiliary task by using cosine similarity of task gradients.

5 DISCUSSION

In this work we have explored a simple yet efficient technique to ensure that an auxiliary loss does not hurt progress on the main task. The approach reduces to applying the gradients of the auxiliary task only if they are also a descent direction of the main task.

The approach suffers from a few shortcomings. First, estimating the cosine similarity between the gradients of the two tasks might be expensive or noisy (e.g. due to relying Monte Carlo estimates of the gradients). Though this can probably be addressed by smoothing and by relying on a potentially hyper-tuned threshold for the cosine similarity. Note that intuitively one would assume that the high-dimensionality can also work against the approach as random vectors tend to be orthogonal in high-dimensional spaces, that cosine will be naturally driven to 0. Note that the intuition of our measurement being affected by noisy gradients is not exactly correct. In fact, the opposite happens. If two gradients are meant to be co-linear, the noise components cancel each other. Figure 11 in Appendix H explores this empirically.

Second, the new loss surface might be less smooth. This can be problematic when using optimizers that rely on statistics of the gradients or second order information (e.g. Adam or RMSprop). In these cases, the transition from just the gradient of the main task to the sum of the gradients can affect the statistics of the optimizer in unwanted ways.

Lastly, theoretically one can only ensure that the proposed update rule still leads to convergence on the main task, while positive transfer can not be guaranteed. While removing worst case scenarios is important, one might care more for data efficiency when using auxiliary losses. Particularly in Appendix D we construct a counter-example where the proposed update rule results in slowing down learning compared to optimizing the main task alone. Empirically, however, the approach works quite well on complex and noisy tasks, like Atari games, using RMSprop.

We have mostly considered techniques where auxiliary task helps initially, but hurts later. It would be interesting to apply gradient cosine similarity in settings where the auxiliary task hurts initially, but helps in the end. Some examples of this scenario are annealing β in β-VAE (Higgins et al., 2017), and annealing the confidence penalty in (Pereyra et al., 2017).

REFERENCES

H. B. Ammar, E. Eaton, M. E. Taylor, D. C. Mocanu, K. Driessens, G. Weiss, and K. Tuyls. An automated measure of MDP similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

J. L. Carroll and K. Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 803–808. IEEE, 2005.

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

A. Goldstein. Cauchy's method of minimization. *Numerische Mathematik*, 4(1):146–150, 1962.

S. Hauberg, O. Freifeld, A. B. L. Larsen, J. Fisher, and L. Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pages 342–350, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt. Multi-task deep reinforcement learning with popart. *arXiv preprint arXiv:1809.04474*, 2018.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a Nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. In *International Conference on Learning Representations*, 2017.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

S. J. Pan, Q. Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

G. Papoudakis, K. C. Chatzidimitriou, and P. A. Mitkas. Deep reinforcement learning for doom using unsupervised auxiliary tasks. *CoRR*, abs/1807.01960, 2018.

G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

N. Quadrianto, J. Petterson, T. S. Caetano, A. J. Smola, and S. Vishwanathan. Multitask learning without label correspondences. In *Advances in Neural Information Processing Systems*, pages 1957–1965, 2010.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

S. Schmitt, J. J. Hudson, A. Zidek, S. Osindero, C. Doersch, W. M. Czarnecki, J. Z. Leibo, H. Kuttler, A. Zisserman, K. Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

Y. Zhang, K. Lee, and H. Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *International Conference on Machine Learning*, pages 612–621, 2016.

A PROOFS

A.1 PROOF FOR PROPOSITION 1

Given any gradient vector field $G(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ and any vector field $V(\boldsymbol{\theta})$ (such as gradient of another loss function, but could be arbitrary set of updates). Given update rule of form

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \alpha^{(t)}(G(\boldsymbol{\theta}^{(t)}) + V(\boldsymbol{\theta}^{(t)})\max(0, \cos(G(\boldsymbol{\theta}^{(t)}), V(\boldsymbol{\theta}^{(t)})))$$

converges to the local minimum of $\mathcal{L}$ given small enough $\alpha^{(t)}$.

Proof. Let us denote

$$G^{(t)} := G(\boldsymbol{\theta}^{(t)}) \qquad V^{(t)} := V(\boldsymbol{\theta}^{(t)}) \qquad \nabla\mathcal{L}^{(t)} := \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)})$$

$$\Delta\boldsymbol{\theta}^{(t)} := G^{(t)} + V^{(t)}\max(0, \cos(G^{(t)}, V^{(t)})).$$

Our update rule is simply $\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \alpha^{(t)}\Delta\boldsymbol{\theta}^{(t)}$ and we have

$$\langle\Delta\boldsymbol{\theta}^{(t)}, \nabla\mathcal{L}^{(t)}\rangle = \langle G^{(t)} + V^{(t)}\max(0, \cos(G^{(t)}, V^{(t)})), \nabla\mathcal{L}^{(t)}\rangle \tag{3}$$

$$= \langle G^{(t)}, \nabla\mathcal{L}^{(t)}\rangle + \langle V^{(t)}\max(0, \cos(G^{(t)}, V^{(t)})), \nabla\mathcal{L}^{(t)}\rangle \tag{4}$$

$$= \|\nabla\mathcal{L}^{(t)}\|^2 + \frac{1}{\|V^{(t)}\|\|\nabla\mathcal{L}^{(t)}\|}\max(0, \langle\nabla\mathcal{L}^{(t)}, V^{(t)}\rangle)\langle V^{(t)}, \nabla\mathcal{L}^{(t)}\rangle \geq 0. \tag{5}$$

And it can be 0 if and only if $\|\nabla\mathcal{L}^{(t)}\| = 0$ (since sum of two non-negative terms is zero iff both are zero, and step from (4) to (5) is only possible if this is not true), thus it is 0 only when we are at the critical point of $\mathcal{L}$. Thus method converges due to convergence of steepest descent methods, see "Cauchy's method of minimization" (Goldstein, 1962). □

A.2 PROOF FOR PROPOSITION 3

In general, update rule proposed does not have to create a conservative vector field.

Proof. Proof comes from a counterexample, let us define in 2D space:

$$\mathcal{L}_{targ}(\theta_1, \theta_2) = a\theta_1$$

$$\mathcal{L}_{aux}(\theta_1, \theta_2) = \begin{cases} a\theta_1 & \text{if } \theta_1 \in [1, 2] \wedge \theta_2 \in [0, 1] \\ 0 & \text{therwise} \end{cases}$$

for some fixed $\alpha \neq 0$. Let us now define two paths (parametrized by s) between points $(0, 0)$ and $(2, 2)$, A which is a concatenation of a line from $(0, 0)$ to $(0, 2)$ (we call it U, since it goes up) and line from $(0, 2)$ to $(2, 2)$ (which we call R as it goes right), and path B which first goes right and then up. Let us denote by V_{cos} the update rule we follow, then:

$$\int_A V_{\text{cos}} ds = \int_A \nabla\mathcal{L}_{\text{targ}} ds = \int_U \nabla\mathcal{L}_{\text{targ}} ds + \int_R \nabla\mathcal{L}_{\text{targ}} ds = \int_R \nabla\mathcal{L}_{\text{targ}} ds = 2a$$

At the same time, since gradient of $\mathcal{L}_{targ}$ is conservative by definition:

$$\int_B V_{\text{cos}} ds = \int_B \nabla\mathcal{L}_{\text{targ}} ds + \int_C \nabla\mathcal{L}_{\text{aux}} ds = \int_A \nabla\mathcal{L}_{\text{targ}} ds + \int_C \nabla\mathcal{L}_{\text{aux}} ds = 2a + \int_C \nabla\mathcal{L}_{\text{aux}} ds = 3a$$

where C is a part of B that goes through $[1, 2] \times [0, 1]$. We conclude that $\int_A V_{\text{cos}} ds \neq \int_B V_{\text{cos}} ds$, so our vector field is not path invariant, thus by the Green's Theorem it is not conservative, which concludes the proof. See Figure 6 for visualisation. □

B ONE-DIMENSIONAL TOY EXAMPLE

Figure 7 shows the surfaces along with gradients for the one-dimensional motivating example described in Section 2.

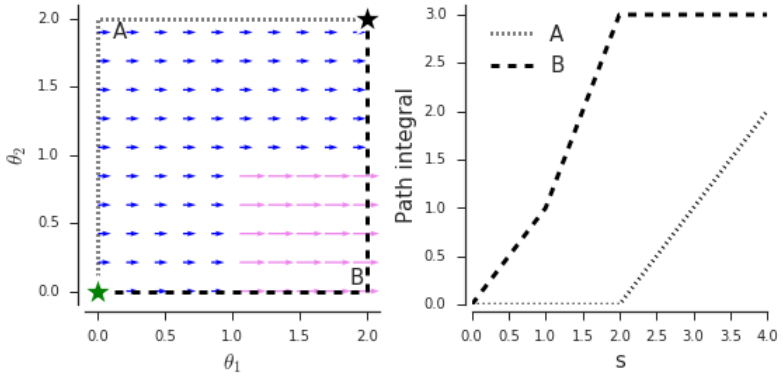


Figure 6: Visualisation of the counterexample from Proposition 3, stars denote starting (green) and end (black) points. Dotted and dashed lines correspond to paths A and B respectively. Blue arrows represent gradient vector field of the main loss, while the violet ones the merged vector field.

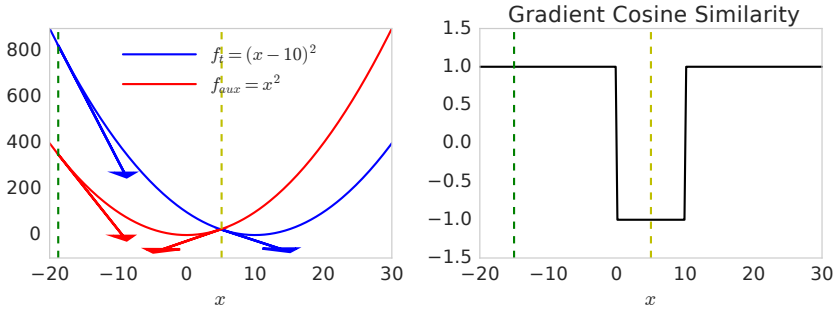


Figure 7: Illustration of cosine similarity between gradients on synthetic loss surfaces.

C PSEUDOCODE

Algorithms 1 and 2 describe the pseudocode of our method.

D TOY EXAMPLE SHOWING SLOW-DOWN

We discuss here a few potential issues of using cosine similarity of gradients to measure task similarity. First, the method depends on being able to compute cosine between gradients. However, in DL we rarely are able to compute exact gradients in practice, nut instead depend on their high variance estimators (mini batches in supervised learning, or Monte Carlo estimators in RL). Consequently, estimating cosine might require additional tricks such as keeping moving averages of estimates. Second, adding additional task gradient in selected subset of iterates can lead to very bumpy surface from the perspective of optimizer, causing methods which keep track of gradient statistics/estimate higher order derivatives, can be less efficient. Finally, one can construct specific functions, where despite still minimizing the loss, one significantly slows down optimization process. Figure 8 provides one such function as an example.

E IDENTIFYING NEAR AND FAR CLASSES IN IMAGENET

As a ground truth for class similarity, we identify pairs of ImageNet classes to be near or far using, *lowest common ancestor (LCA)* and *Frechet Inception Distance (FID)* (Heusel et al., 2017).

ImageNet follows a tree hierarchy where each class is a leaf node. We define the distance between a pair of classes as at which tree level their LCA is found. In particular, there are 19 levels in the class tree, each leaf node (i.e. class) is considered to be level 0 while the root node is considered to be level 19. We perform bottom-up search for one pair of random sampled classes and find their LCA

Algorithm 1 Pseudocode of our method (hard cosine mixing).

Initialize shared parameters $\boldsymbol{\theta}$ and task specific parameters $\boldsymbol{\phi}_{main}, \boldsymbol{\phi}_{aux}$. randomly.
for iter $= 1 :$ max_iter **do**
  Compute $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}, \nabla_{\boldsymbol{\phi}_{main}}\mathcal{L}_{main}, \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}, \nabla_{\boldsymbol{\phi}_{aux}}\mathcal{L}_{aux}$.
  Update $\boldsymbol{\phi}_{main}$ and $\boldsymbol{\phi}_{aux}$ using corresponding gradients
  if $\cos(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}, \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}) \geq 0$ **then**
    Update $\boldsymbol{\theta}$ using $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main} + \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}$
  else
    Update $\boldsymbol{\theta}$ using $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}$

Algorithm 2 Pseudocode of smooth version of our method (soft cosine mixing).

Initialize shared parameters $\boldsymbol{\theta}$ and task specific parameters $\boldsymbol{\phi}_{main}, \boldsymbol{\phi}_{aux}$. randomly.
for iter $= 1 :$ max_iter **do**
  Compute $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}, \nabla_{\boldsymbol{\phi}_{main}}\mathcal{L}_{main}, \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}, \nabla_{\boldsymbol{\phi}_{aux}}\mathcal{L}_{aux}$.
  Update $\boldsymbol{\phi}_{main}$ and $\boldsymbol{\phi}_{aux}$ using corresponding gradients
  Update $\boldsymbol{\theta}$ using $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main} + \max(0, \cos(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{main}, \nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}))\nabla_{\boldsymbol{\theta}}\mathcal{L}_{aux}$

node—the class distance is then defined as the level number of this node. For example, class 871 ("trimaran") and class 484 ("catamaran") has class distance 1 because their LCA is one level up.

FID is used as a second measure of similarity. We obtain the image embedding of a pair of classes using the penultimate layer of a pre-trained ResNetV2-50 model (He et al., 2016b) and then compute the embedding distance using FID, defined in Heusel et al. (2017) as:

$$\text{FID} = d^2\big((m_1, C_1), (m_2, C_2)\big) = \|m_1 - m_2\|_2^2 + \text{Tr}\big(C_1 + C_2 - 2(C_1C_2)^{1/2}\big). \tag{6}$$

where m_k, C_k denote the mean and covariance of the embeddings from class k.

We randomly sampled 50 pairs of classes for each level of $LCA = \{1, 2, 3, 4, 16, 17, 18, 19\}$ (400 pair of classes in total) and compute their FID. Figure 9 shows a plot of LCA (x-axis) verses FID (y-axis) over our sampled class pairs. It can be seen that LCA and FID are (loosely) correlated and that they reflect human intuition of task similarity for some pairs. For example, trimaran and catamaran (bottom-left) are similar both visually and conceptually, whereas rock python and traffic light (top-right) are dissimilar both visually and conceptually. However, there are contrary examples where LCA disagrees with FID; monkey pinscher and doberman pinscher (top-left) are visually dissimilar but conceptually similar, whereas bubble and sundial (bottom-right) are visually similar but conceptually dissimilar.

Per the observations, in subsequent experiments we pick class pairs that are {*Low LCA, Low FID*} as *near* pairs (e.g., trimaran and catamaran), and class pairs that are {*high LCA, high FID*} as *far* pairs (e.g., rock python and traffic light).

F GRID WORLD EXPERIMENTS

We define a distribution over 15×15 mazes, where an agent observes its surrounding (up to 4 pixels away) and can move in 4 directions (with 10% transition noise). We randomly place walls (blocking movement) as well as two types of positive rewards: $+5$ and $+10$ points, both terminating an episode. There are also some negative rewards (both terminating and non-terminating) to make problem harder. In order to guarantee (expected) finite length of episodes we add fixed probability of 0.01 of transitioning to a non-rewarding terminal state.

For the sake of simplicity we use episode-level policy gradient (Williams, 1992) with value function baseline, with policies parametrized as logits $\boldsymbol{\theta}$ of $\pi(a|s) = \frac{\exp(\theta_{s,a})}{\sum_b \exp(\theta_{s,b})} \in [0, 1]$, baselines as $B_s \in \mathbb{R}$, with fixed learning rate of $\alpha = 0.01$, discount factor $\gamma = 0.95$ and 10,000 training steps (states visited).

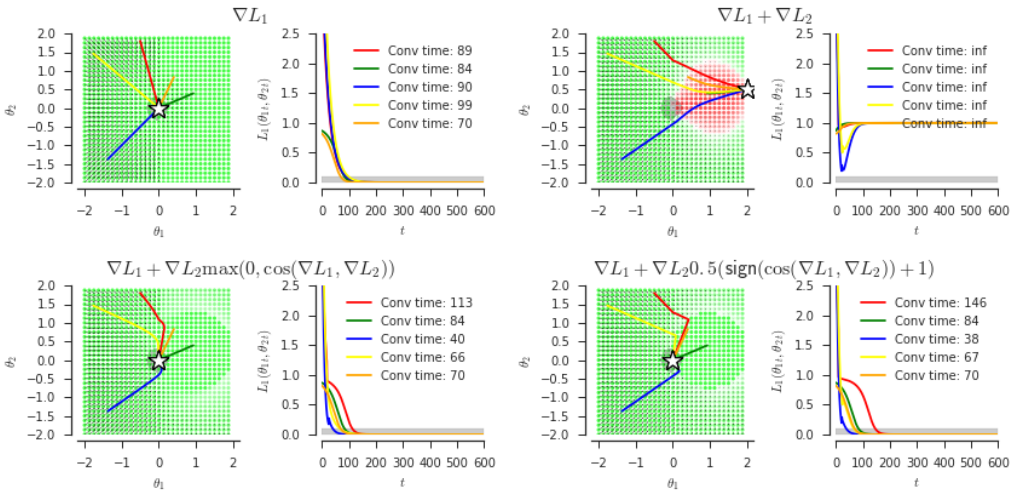


Figure 8: Example 2D function where the proposed method can lead to slow-down (compare red runs). $L_1(\theta) = (\theta_1 < 0)(\theta_1^2 + \theta_2^2) + (\theta_1 > 0)\Big(1 - \exp\big(-2(\theta_1^2 + \theta_2^2)\big)\Big)$ and $L_2(\theta) = (\theta_1 - 2)^2 + (\theta_2 - 0.5)^2$. For simplicity we are using L_1 which is non differentiable/smooth for $x = 0$ but one can create smooth function with analogous properties, this path simply gave us an equation which is easier to present. The core idea is to have a flat region of the loss surface, to which the auxiliary task is gonna push the iterates. Even though this move is still decreasing the loss – it is going to slow down the whole optimization process.

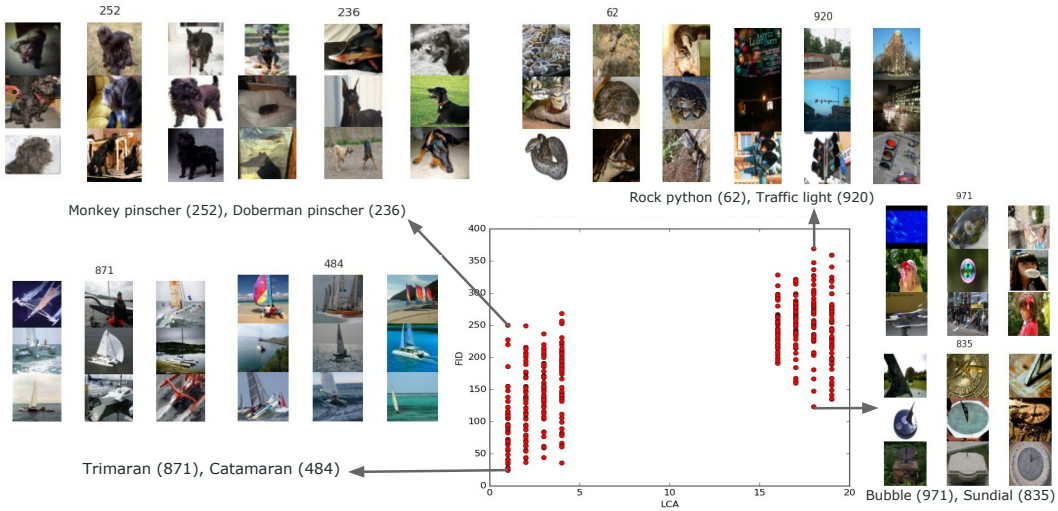


Figure 9: LCA (x-axis) versus FID (y-axis) as a ground truth for class similarity. The measurements reflect human intuition of class similarity; trimaran and catamaran (bottom-left) are similar both visually and conceptually, whereas rock python and traffic light (top-right) are dissimilar both visually and conceptually.

For this setup, the update rule for each sequence $\tau = \big((s_1, a_1, r_1), \ldots (s_N, a_N, r_N)\big)$ is thus given by

$$\Delta \boldsymbol{\theta} = \alpha \nabla_{\boldsymbol{\theta}} \log \pi(a_{t'}|s_{t'}) \left[\sum_{i=0}^{N-t'} r_{t'+i} - B_{s_{t'}} \right] = \alpha G^{(t)} \qquad \Delta B_{s_{t'}} = -\alpha \nabla_{B_{s_{t'}}} (B_{s_{t'}} - \sum_{i=0}^{N-t'} r_{t'+i})^2.$$

In order to make use of expert policies for $\mathcal{T}_{aux}$ we define auxiliary loss as a distillation loss, which is just a per-state cross-entropy between teacher's and student's distributions over actions. If we just

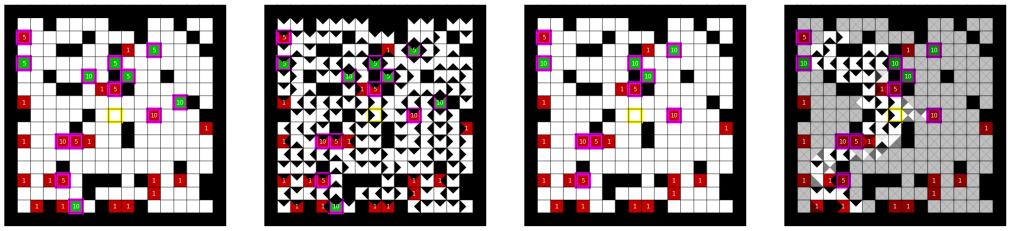


Figure 10: **Left most**: Initial task $\mathcal{T}_{main}$, yellow border denotes starting point and violet ones terminating states. Red states are penalizing with the value in the box while the green ones provide positive reward. **Middle Left**: Solution found by a single run of Q-learning with uniform exploration policy. **Middle Right**: Transformed task $\mathcal{T}_{aux}$. **Right most**: Solution found by gradient cosine similarity driven distillation with policy gradient.

add gradients estimated by policy gradient, and the ones given by distillation, the update is given by

$$\Delta\boldsymbol{\theta} = \alpha\left[G^{(t)} - \nabla_{\boldsymbol{\theta}}\mathrm{H}^{\times}(\pi^{\mathrm{Q}}(\cdot|s_{t'})\|\pi(\cdot|s_{t'}))\right] = \alpha[G^{(t)} + \sum_{a}\pi^{\mathrm{Q}}(a|s_{t'})\nabla_{\boldsymbol{\theta}}\log\pi(a|s_{t'})],$$

where $V^{(t)} = \sum_a \pi^{\mathrm{Q}}(a|s_{t'})\nabla_{\boldsymbol{\theta}}\log\pi(a|s_{t'})$ and $\mathrm{H}^{\times}(p,q) = -\sum_k p_k \log q_k$ is the cross entropy.

However, if we use the proposed gradient cosine similarity, we get the following update

$$\Delta\boldsymbol{\theta} = \alpha\left[G^{(t)} + V^{(t)}\big(2\cdot\mathrm{sign}(\cos(G^{(t)}, V^{(t)})) - 1\big)\right]$$

. This get a significant boost to performance, and policies that score on average **3** points after 10,000 steps and obtain baseline performance after just one third of steps. Figure 10 shows a depiction of the task and an example solution.

G ATARI EXPERIMENTS

For these experiments, we use a convolutional architecture as in previous work (Espeholt et al., 2018; Hessel et al., 2018; Mnih et al., 2015; 2016), trained with batched actor-critic with the V-trace algorithm (Espeholt et al., 2018). We use a learning rate of 0.0006 and an entropy cost of 0.01 for all experiments, with a batch size of 32 and 200 parallel actors.

For the single game experiment, Breakout, , we use 0.02 for the threshold on the cosine similarity and, for technical reasons we ended up computing the cosine distance on a per-layer basis and then averaged. We additionally need to do a moving average of the cosine over time ($0.999c^{(t-1)} + 0.001c^{(t)}$) to ensure there are no sudden spikes in the weighting due to noisy gradients. Same setting is used for the multi-task experiment, just that the threshold is set to 0.01.

H COSINE SIMILARITY IN HIGH DIMENSIONS

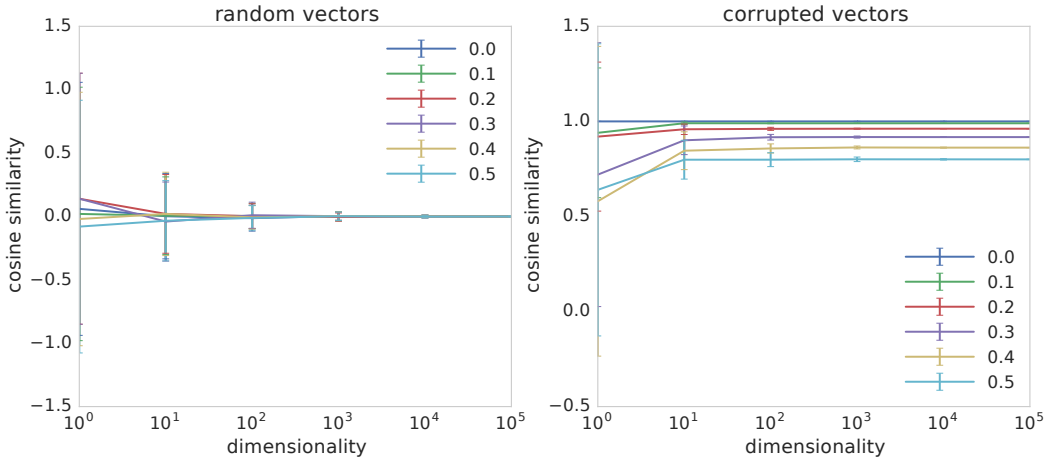


Figure 11: Cosine similarity as a function of dimensionality. On the left, we generate two random vectors from a Gaussian distribution and as expected, the cosine similarity drops to zero very quickly as the dimensionality increases. On the right, we mimic a scenario where the true gradients of the main and auxiliary are aligned, however we observe only *corrupted* noisy gradients which are noisy copies of the same underlying vector; in this case, cosine similarity is larger in higher dimensions (as the inner product of the corruption noise goes to zero).