Given a main task of interest and an auxiliary task which is not of direct interest, how do we weight the auxiliary loss? The typical multi-task approach uses:

\[ \text{arg min}_{\theta, \phi_{\text{main}}, \phi_{\text{aux}}} L_{\text{main}}(\theta, \phi_{\text{main}}) + \lambda L_{\text{aux}}(\theta, \phi_{\text{aux}}) \]

However, this could be sub-optimal since we care only about performance on the main task. E.g., the auxiliary loss might help initially but hurt later. We want to solve the following problem:

\[ \text{arg min}_{\theta, \phi_{\text{main}}, \phi_{\text{aux}}} \left( \theta(\theta^{(i)}) - \alpha \nabla_{\theta} L_{\text{main}}(\theta(\theta^{(i)}) + \lambda \nabla_{\phi_{\text{aux}}} L_{\text{aux}}(\theta(\theta^{(i)})) \right) \]

**Question:** How to automatically adapt the auxiliary loss so that it does not hurt the main loss?

### Motivating example

Given two tasks with shared parameters \( \theta \) & task-specific parameters \( \phi_{\text{main}} \& \phi_{\text{aux}} \), the update:

\[ \theta^{(i+1)} := \theta^{(i)} - \alpha^{(i)} \left( \nabla_{\theta} L_{\text{main}}(\theta^{(i)}) + \nabla_{\theta} L_{\text{aux}}(\theta^{(i)}) \right) \max(0, \nabla_{\phi_{\text{aux}}} L_{\text{aux}}(\theta^{(i)})) \]

\[ \phi_{\text{main}}^{(i+1)} := \phi_{\text{main}}^{(i)} - \alpha^{(i)} \nabla_{\phi_{\text{main}}} L_{\text{main}}(\theta^{(i)}) \quad \text{and} \quad \phi_{\text{aux}}^{(i+1)} := \phi_{\text{aux}}^{(i)} - \alpha^{(i)} \nabla_{\phi_{\text{aux}}} L_{\text{aux}}(\theta^{(i)}) \]

leads to convergence to local minimum of \( L_{\text{main}} \) w.r.t. \( \theta, \phi_{\text{main}} \) given small enough \( \alpha^{(i)} \).

**Weighted version:** Weight the auxiliary loss by cosine similarity (as above).

**Unweighted version:** Use aux loss when \( \cos > \text{Threshold} \) and ignore otherwise.

### 3. Supervised Learning Using Pairs Of ImageNet Classes

**Ground truth of task similarity** use Least Common Ancestor (LCA) and Frechet Inception Distance (FID) between ImageNet classes.

**Near pair:** the most similar, such as Trimaran and Catamaran

**Far pair:** the least similar, such as Rock python and Traffic light

### 4. Reinforcement Learning On Imperfect-Teacher Distillation

**Single task on Breakout:** the main task is Breakout, the auxiliary task is a sub-optimal pre-trained Breakout teacher

- **Only KL:** solely following the teacher leads to sub-optimal solutions
- **RL (Baseline):** single task learning without the teacher
- **RL + KL (Baseline):** the teacher only helps initially

**Our Method:** uses the teacher's knowledge when it helps initially and ignores when it hurts later on

**Multi-task on Breakout and Ms. PacMan:** the main task is multi-task Breakout + Ms. PacMan, the auxiliary task is a sub-optimal pre-trained Breakout teacher

- **Multi-task:** learns Ms. PacMan at the expense of Breakout
- **Multi-task RL + Distillation:** the teacher helps Breakout but hurts Ms. PacMan

**Our Method:** Ms. PacMan ignores the teacher when it hurts; both Breakout and Ms. Pacman learn well

### 5. Summary

- Proposed gradient cosine similarity as a simple yet effective way to automatically adapt the auxiliary task to help (& not hurt) the main task.

- Experiments on ImageNet and Atari show empirical success; paper contains additional experiments on cross-domain distillation tasks.

- Paper shows theoretical guarantees on the convergence to local optimum of the main task.

**Potential issues and Future directions**

- Guarantees convergence to local optimum of the main task but not faster convergence.
- Extend theory to optimizers that rely on statistics of the gradients or second order information (e.g., Adam or RMSprop).
- Apply our method to settings where the auxiliary task hurts initially but helps later.