# Knowledge Transfer and Adaptation

## Multi-Task Learning

Antonio Carta

antonio.carta@unipi.it

# Outline

- introduction to Multi-Task Learning
- problems and design choices
  - We will see more examples of methods in the CL lectures
- notebook on finetuning and MTL

# Multi-Task Learning - Motivations

**Example**: robot learning new skills.

- Can it generalize to slightly different tools or objects?
  - with minimal training (few-shot)
  - without training (robustness and generalization, zero-shot)
    - We will talk about this later

**Problems**:

- low-data or long tail problems. Examples: autonomous driving, medicine, low-resource languages
- few-shot learning. Quickly adapt to new domains/tasks using few instances

# Notation

- task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$,
  - True data-generating distribution $p_i(x, y)$
  - loss $\mathcal{L}_i \leftarrow$ this is the evaluation loss function, not necessarily the training loss function
- usually, we have to some samples $\mathcal{D}_\mathrm{i} = \{(\mathbf{x}, \mathbf{y})_k \sim p_i(x, y)\}$

# What is a Task?

what is a task? data $D$, loss $L \rightarrow$ model $f$

- different objects, people, objectives
- Different problems: classification, detection, segmentation, …
- examples: data from different objects or different backgrounds, domains etc…
- domain informs design choices

# MTL Problem types

**Classification**: $\mathcal{L}(\theta, \mathcal{D}) = -\mathbb{E}_{(x,y)\sim\mathcal{D}}[\log f_\theta(\mathbf{y} \mid \mathbf{x})]$

- split into training/validation/test data: $\mathcal{D}^{tr}, \mathcal{D}^{val}, \mathcal{D}^{ts}$

**Problem types**:

- *Multi-task classification*: $\mathcal{L}$ shared among tasks
- *Multi-label learning*: $\mathcal{L}\ , p\ (\mathbf{x})$ shared among tasks
- sometimes we have access to a task descriptor $z_i$. An integer identifier or a more complex representation (vector of task features).

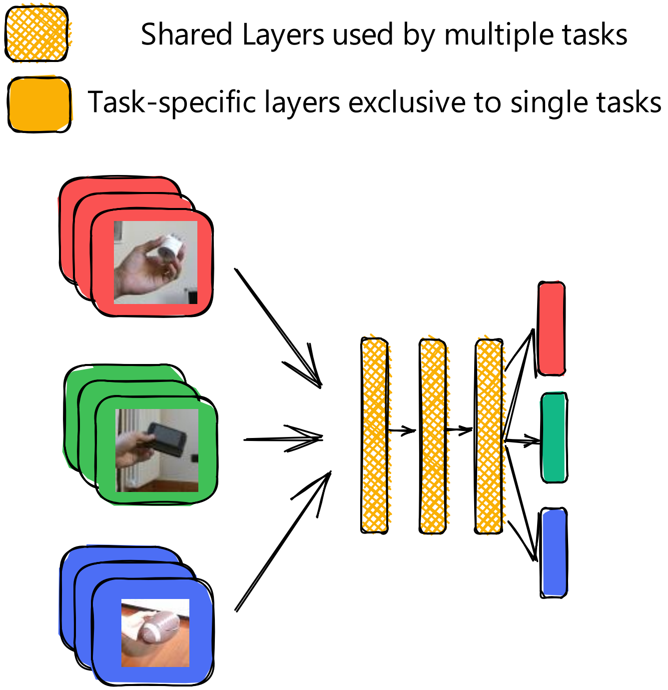
# Problem Statement

**MTL Objective**: $\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i\,(\theta, \mathcal{D}_i)$

- solve all the tasks concurrently
- share knowledge between tasks
- exploit tasks relationships to converge faster and generalize better

**Critical Assumption**:

- tasks share some common structure
  - helps learning multiple tasks jointly
  - it may also cause interference!



# Objectives

MTL is not just about learn “multiple” tasks together

- **fast adaptation**: learn all tasks more quickly
  - The MT model should converge faster than the single task models
- **forward transfer**: improve generalization
  - The MT model has a lower loss than than the single task models
  - May happen in low-data settings
- **meta learning**: given a MT pretrained model, learn new tasks more quickly
  - we will refine this definition later…
- **few-shot learning**: learn to generalize from a very limited set of samples
  - Extreme version of the low-data setting

# Design Choices

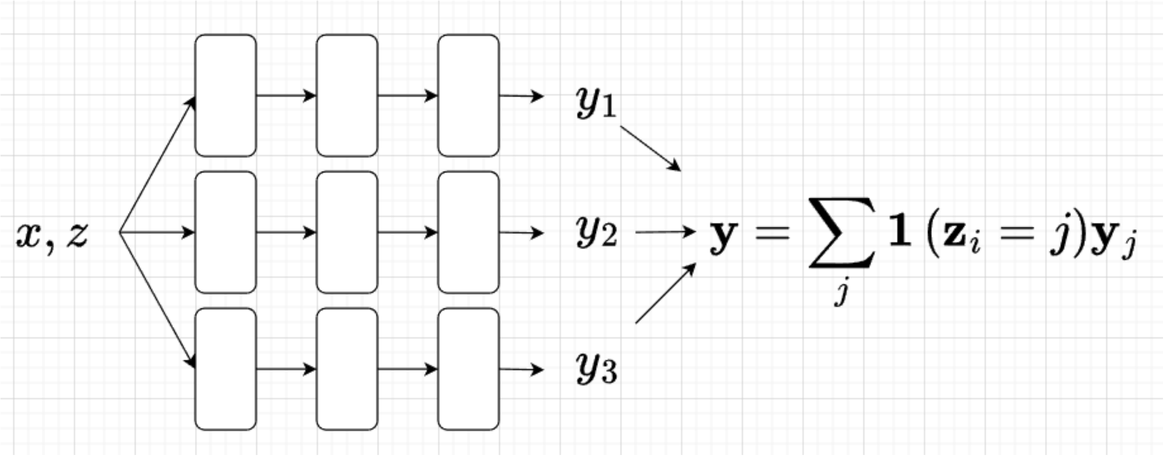
# Task Labels

- let $z$ be the **task label**, an integer that uniquely identifies each task
- we assume to always know the task label
- we can use $z$ to perform task-specific computations
- we can use $z$ to have task-specific parameters

**PROBLEM**: How do we choose which layer to share?

# Weight Sharing – No Thank You

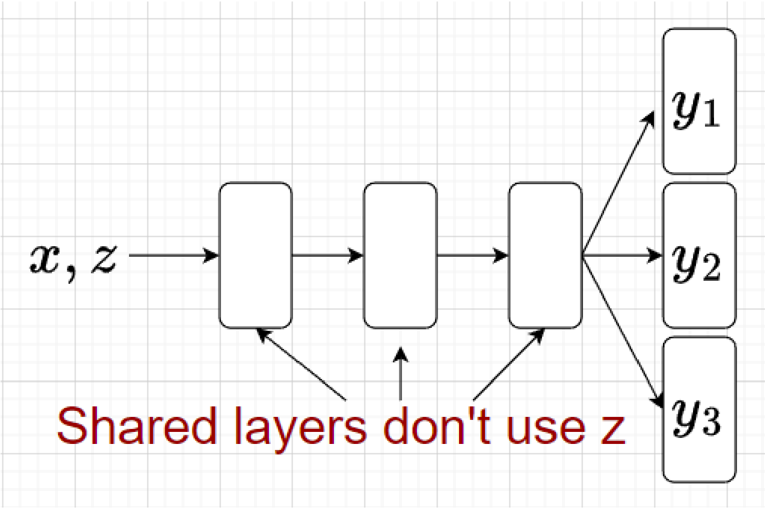
- independent networks (no sharing)
- multiplicative gating mechanism selects the correct model for each input
- no shared components



# Weight Sharing – Multi-Head

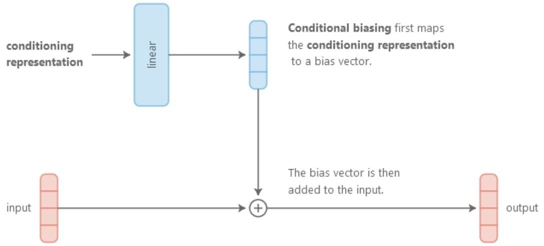
**Multi-Head** architectures have:

- a shared feature extractor
- a separate linear classifier (head) for each task
- the correct head is selected for each example via multiplicative gating

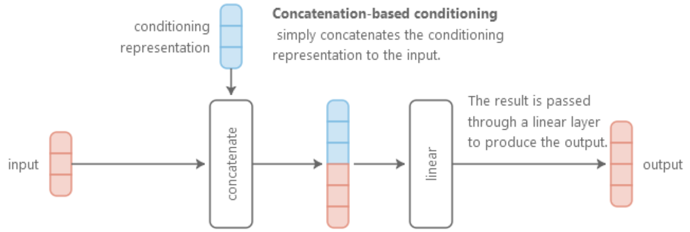


# Weight Sharing – Task Conditioning

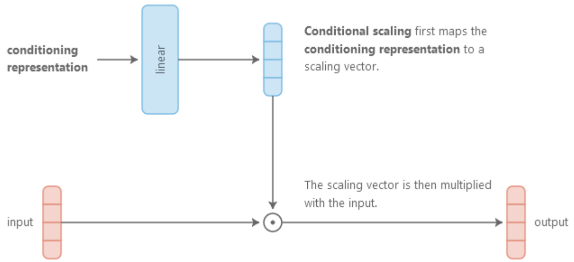
## Sum conditioning




## cat conditioning




## Multiplicative conditioning



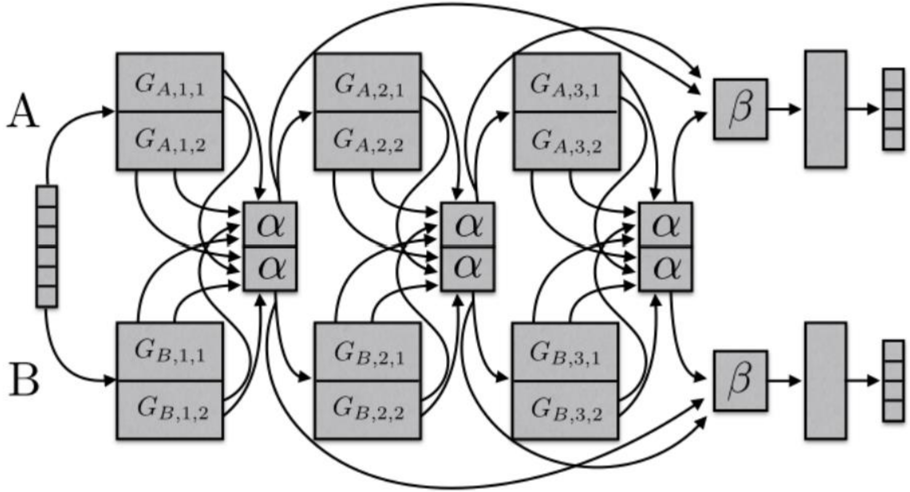

https://distill.pub/2018/feature-wise-transformations/

# Weight Sharing - Task Conditioning

- shared layer with task conditioning. Common examples: sum, concatenation, multiplication
- with task-specific and shared parameters we can decompose the parameterization in task-specific and task-agnostic parameters
  - $\min_{\theta^{sh},\theta^1,\ldots,\theta^T} \sum_{i=1}^{T} \mathcal{L}_i\left(\left\{\theta^{sh},\theta^i\right\},\mathcal{D}_i\right)$
- concatenate $z$ or add $z$ to condition.
- multiplicative conditioning $z$.
  - More expressive
  - Can be used to gate activations

# More Complex Conditioning

- things get complicated easily (e.g. sluice networks)
  - optimal choice is problem-dependent (how close are the domains?)




S, Ruder et al. "Latent multi-task architecture learning". AAAI 2019

# MTL Objective

- weighted objective $\min_\theta \sum_{i=1}^{T} w_i \mathcal{L}_i(\theta, \mathcal{D}_i)$
- how to choose the weights?
  - a predefined relative importance
  - balancing amount of data
  - heuristics
    - gradient of similar magnitudes (Chen et al. GradNorm. ICML 2018)
    - optimize for the worst task

# MTL Optimization

**Naive MT-SGD**

- sample tasks
- sample examples for each task
- SGD step: forward → backward → descent step

- NOTE: we implicitly balance over tasks instead of over samples
- NOTE: losses may have different magnitudes (e.g. in regression problems)

# Transfer and Interference

- **Positive Transfer**: training tasks jointly (i.e. sharing weights) improves the performance on the single tasks
  - if the tasks are small the joint solution is more robust and less prone to overfitting
- **Negative Transfer**:
  - Sometimes independent models are better
  - cross-task interference, different rates of learning
  - representational capacity, MT nets need to be bigger

| | % accuracy | |
|---|---|---|
| task specific, 1-fc (Rosenbaum et al., 2018) | 42 | multi-head architectures |
| task specific, all-fc (Rosenbaum et al., 2018) | 49 | |
| cross stitch, all-fc (Misra et al., 2016b) | 53 | cross-stitch architecture |
| independent | 67.7 | independent training |

Yu et al. Gradient Surgery for Multi-Task Learning. 2020

# Task-Similarity Measures

- Can we guess in advance whether multi-task training will results in positive/negative transfer?
- Not yet…

"Disentangling Task Transfer Learning", CVPR 2018

# Conclusion and Take-Home Messages

- Multi-Task Learning is the problem of jointly learning different tasks
- We assume sharing is beneficial because tasks are related
- Multiple objectives:
  - generalization and transfer between tasks via sharing
  - fast adaptation
  - few-shot learning
- We will see more methods in the Deep Continual Learning lectures

# References

- Slides + footnotes
- Stanford CS330 has some recorded lectures on the topic