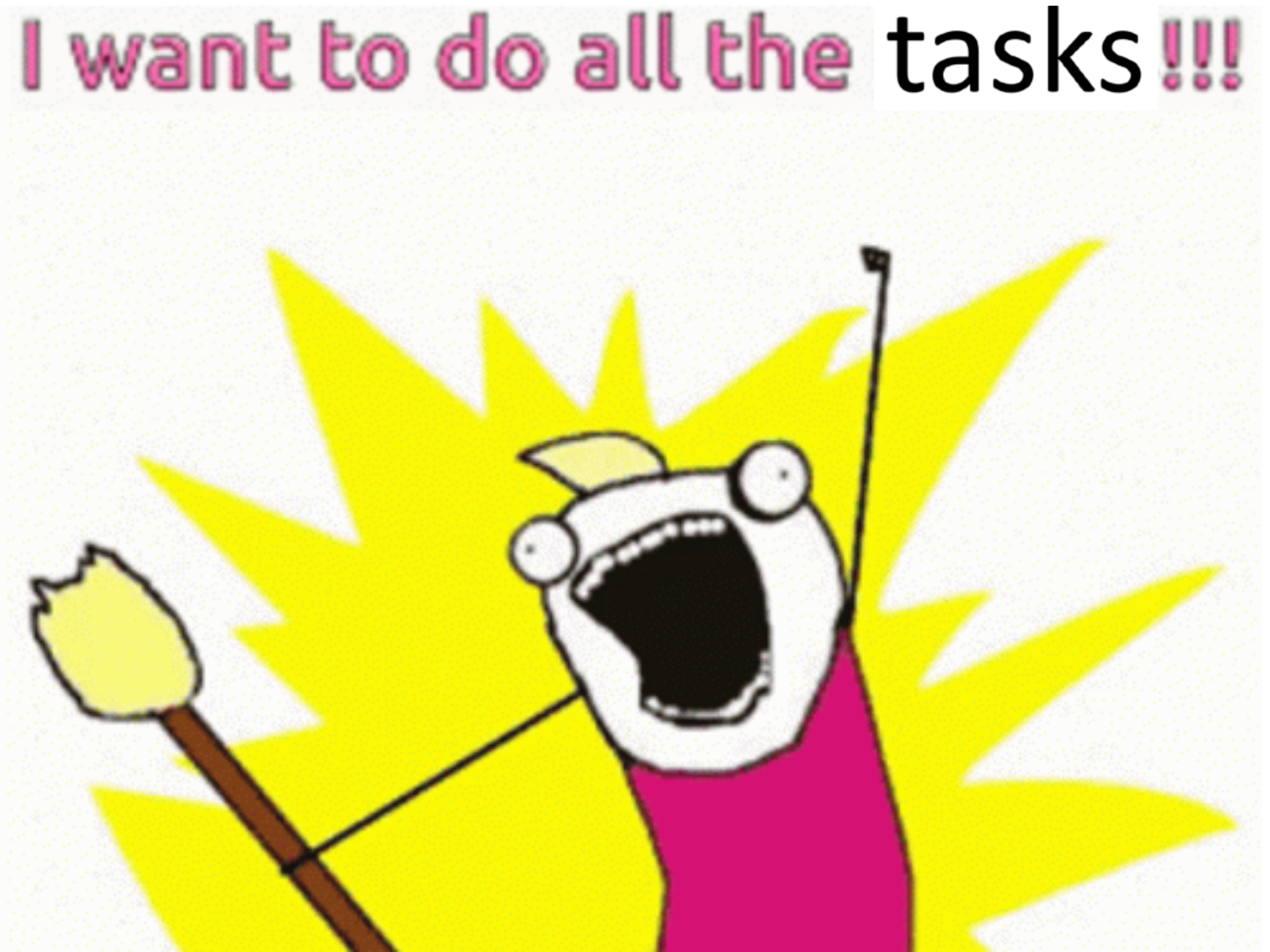
Multi-Task Learning Basics

CS 330



Logistics

Homework 0 due **Monday 10/3** at **11:59 pm PT**.

PyTorch review session **tomorrow at <u>4:30 pm</u> PT.**

Office hours start today

Plan for Today

Multi-Task Learning

- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning

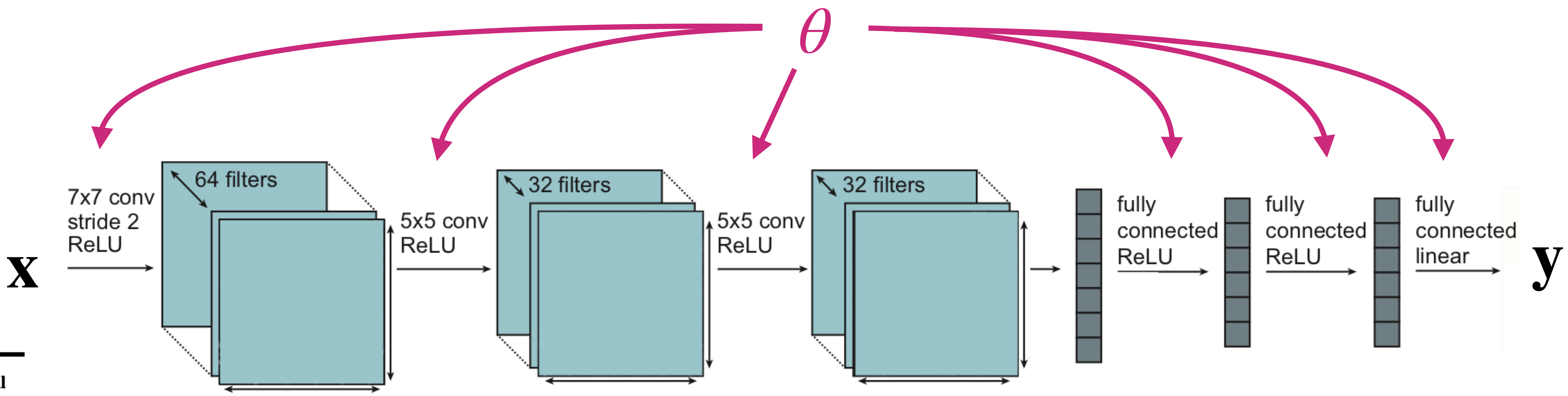
Goals for by the end of lecture:

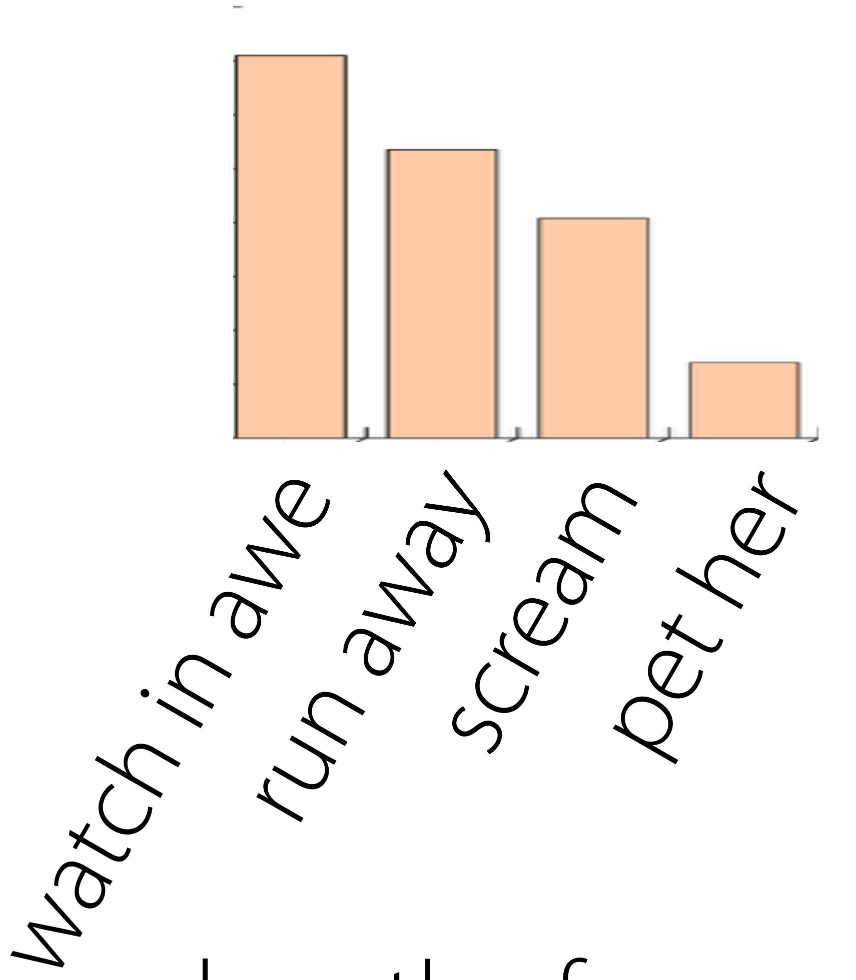
- Understand the key design decisions when building multi-task learning systems

Multi-Task Learning

Some notation

ImageNet Classification with Deep Convolutional Neural Networks





length of paper

$$f_\theta(\mathbf{y} \mid \mathbf{x})$$

Single-task learning: $\mathscr{D} = \{(\mathbf{x}, \mathbf{y})_k\}$

[supervised] $\min_\theta \mathscr{L}(\theta, \mathscr{D})$

Typical loss: negative log likelihood

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

What is a task? (more formally this time)

A task: $\mathscr{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathscr{L}_i\}$

data generating distributions

Corresponding datasets: $\mathscr{D}_i^{tr}$ $\mathscr{D}_i^{test}$

will use $\mathscr{D}_i$ as shorthand for $\mathscr{D}_i^{tr}$:

Examples of Tasks

A task: $\mathscr{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathscr{L}_i\}$

data generating distributions

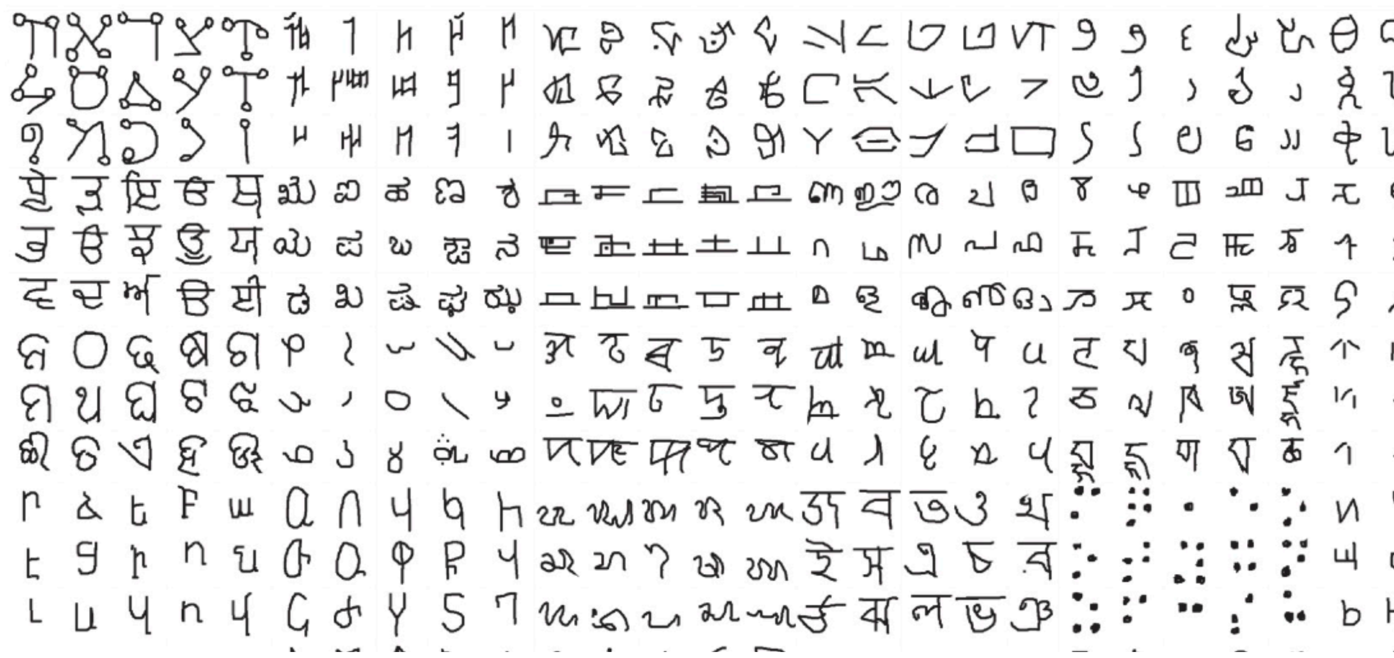
Corresponding datasets: $\mathscr{D}_i^{tr}$ $\mathscr{D}_i^{test}$

will use $\mathscr{D}_i$ as shorthand for $\mathscr{D}_i^{tr}$:

Multi-task classification: $\mathscr{L}_i$ same across all tasks

e.g. per-language handwriting recognition

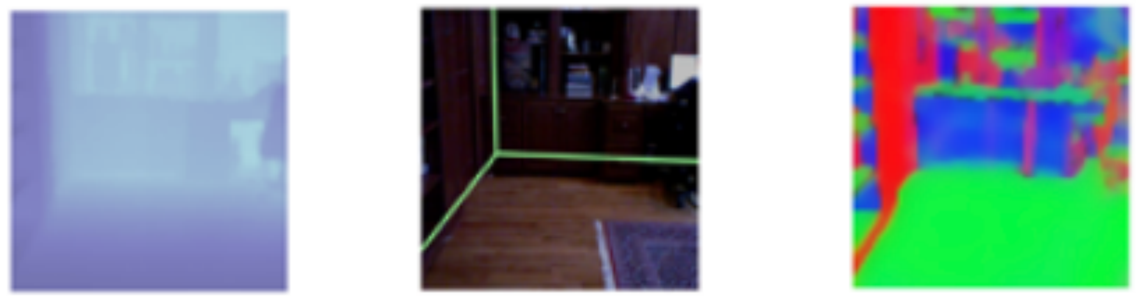
e.g. personalized spam filter



Multi-label learning: $\mathscr{L}_i$, $p_i(\mathbf{x})$ same across all tasks

e.g. face attribute recognition

e.g. scene understanding

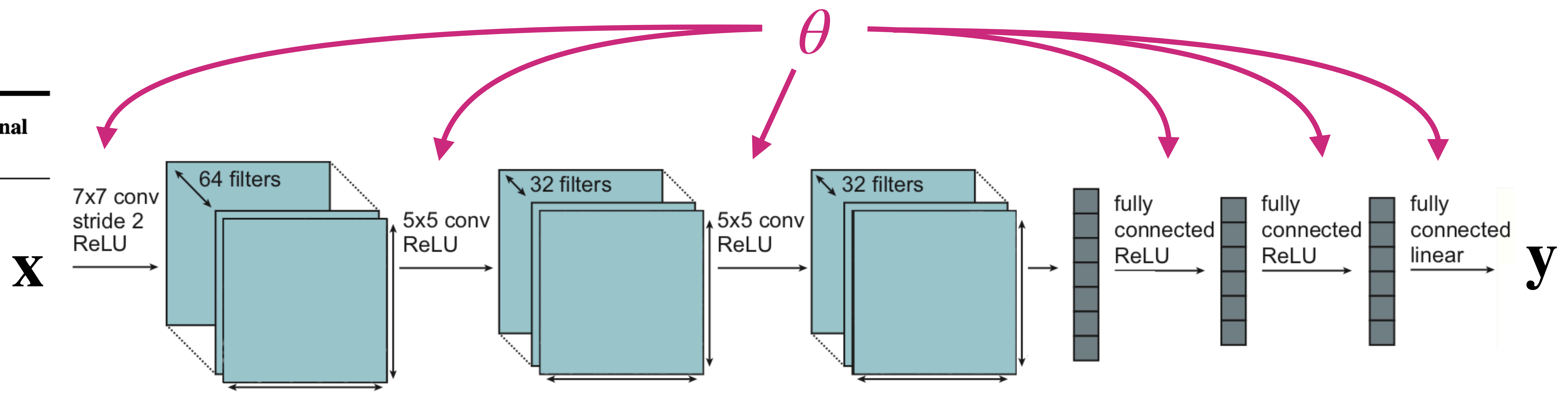


$$L_{\text{tot}} = w_{\text{depth}} L_{\text{depth}} + w_{\text{kpt}} L_{\text{kpt}} + w_{\text{normals}} L_{\text{normals}}$$

When might $\mathscr{L}_i$ vary across tasks?

- mixed discrete, continuous labels across tasks
- multiple metrics that you care about

ImageNet Classification with Deep Convolutional Neural Networks



length of paper

summary of paper

paper review

$\mathbf{z}_i$

task descriptor

$\cancel{f_\theta(\mathbf{y} \,|\, \mathbf{x})}$ $\quad f_\theta(\mathbf{y} \,|\, \mathbf{x}, \mathbf{z}_i)$

e.g. one-hot encoding of the task index

or, whatever meta-data you have

- personalization: user features/attributes
- language description of the task
- formal specifications of the task

Vanilla MTL Objective

$$\min_\theta \sum_{i=1}^{T} \mathscr{L}_i(\theta, \mathscr{D}_i)$$

Decisions on the model, the objective, and the optimization.

How should we condition on $\mathbf{z}_i$? What objective should we use?

How to optimize our objective?

Model

How should the model be conditioned on $\mathbf{z}_i$?

What parameters of the model should be shared?

Objective

How should the objective be formed?

Optimization

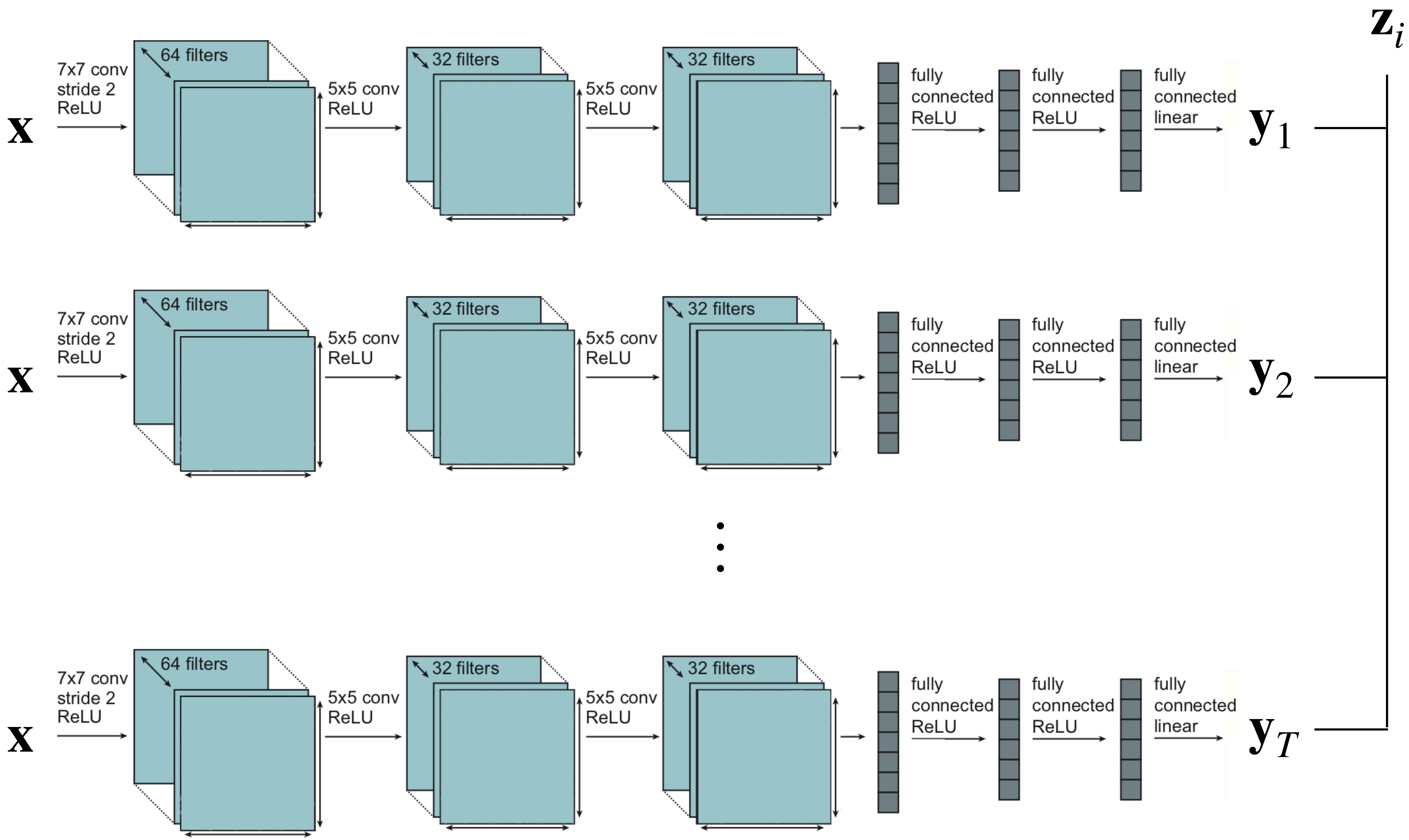
How should the objective be optimized?

Conditioning on the task

Let's assume $\mathbf{z}_i$ is the one-hot task index.

Question: How should you condition on the task in order to share as little as possible?

Conditioning on the task

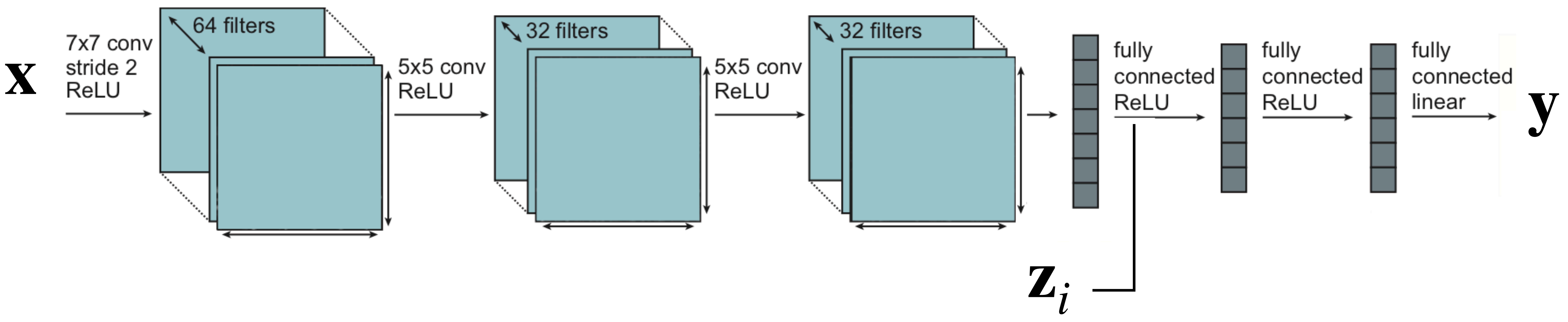


multiplicative gating

$$\mathbf{y} = \sum_j \mathbf{1}(\mathbf{z}_i = j)\mathbf{y}_j$$

—> independent training within a single network!
with no shared parameters

The other extreme



Concatenate $\mathbf{z}_i$ with input and/or activations

all parameters are shared
(except the parameters directly following $\mathbf{z}_i$, if $\mathbf{z}_i$ is one-hot)

An Alternative View on the Multi-Task Architecture

Split θ into shared parameters θ^{sh} and task-specific parameters θ^i

Then, our objective is: $\min_{\theta^{sh},\theta^1,\ldots,\theta^T} \sum_{i=1}^{T} \mathscr{L}_i(\{\theta^{sh}, \theta^i\}, \mathscr{D}_i)$

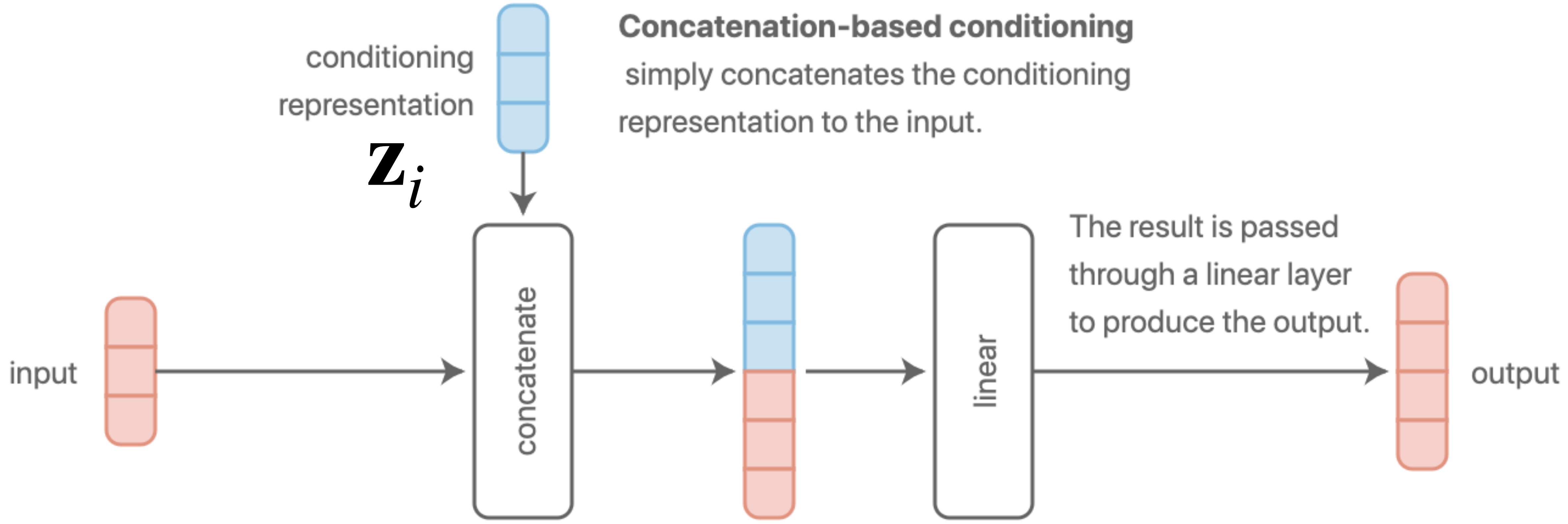
Choosing how to condition on $\mathbf{z}_i$

equivalent to

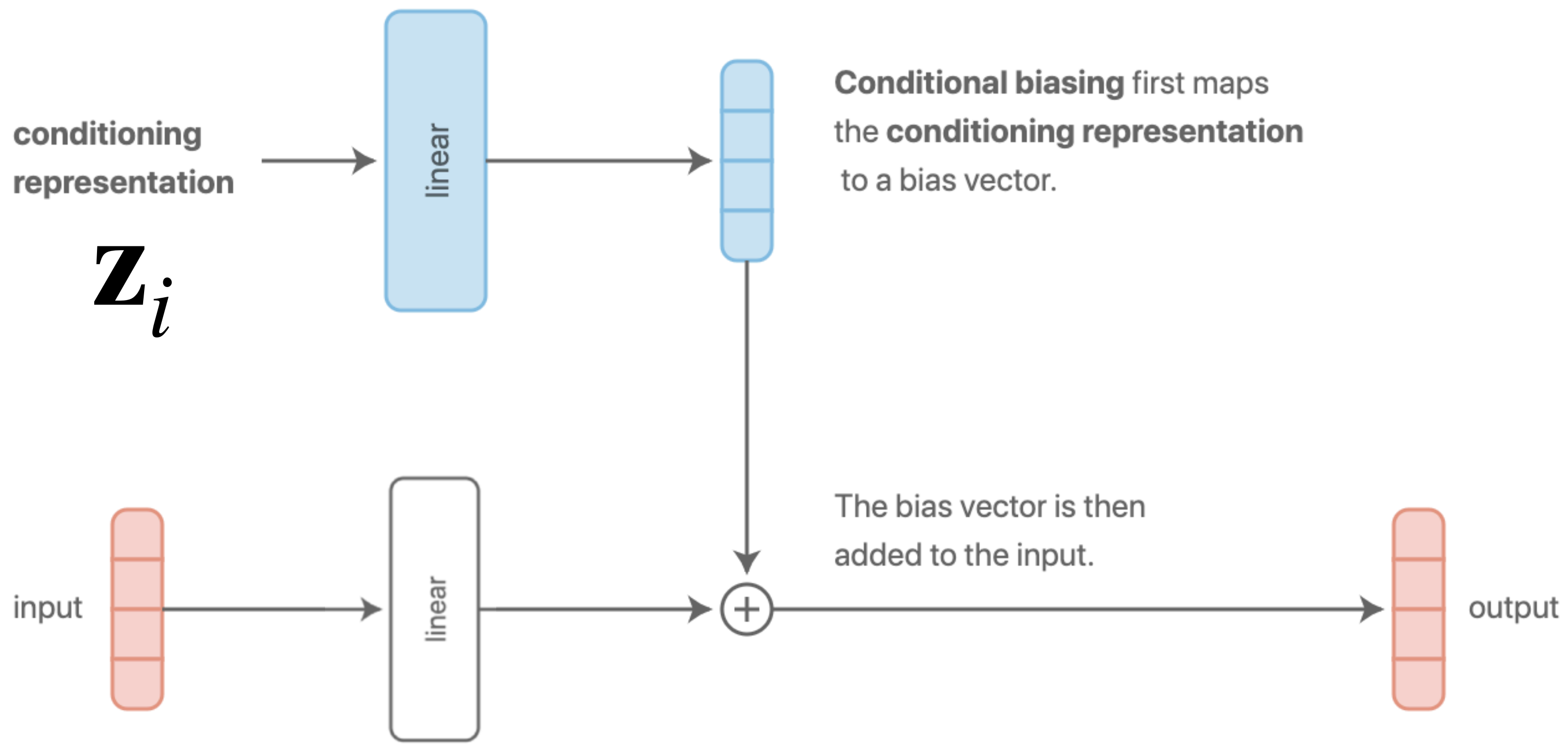
Choosing how & where to share parameters

Conditioning: Some Common Choices

1. **Concatenation-based** conditioning



2. **Additive** conditioning



These are actually equivalent!

Question: why are they the same thing? (raise your hand)

Concat followed by a fully-connected layer:

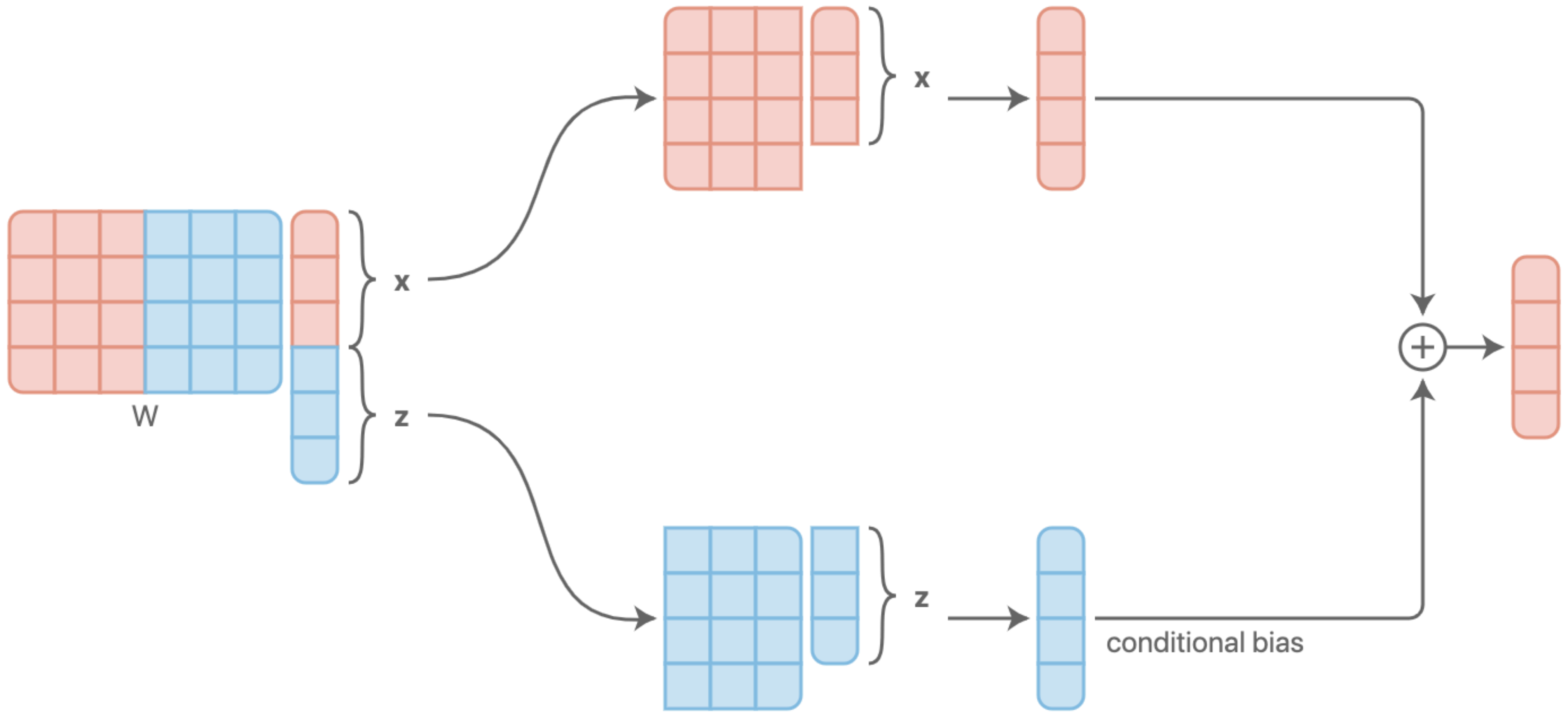
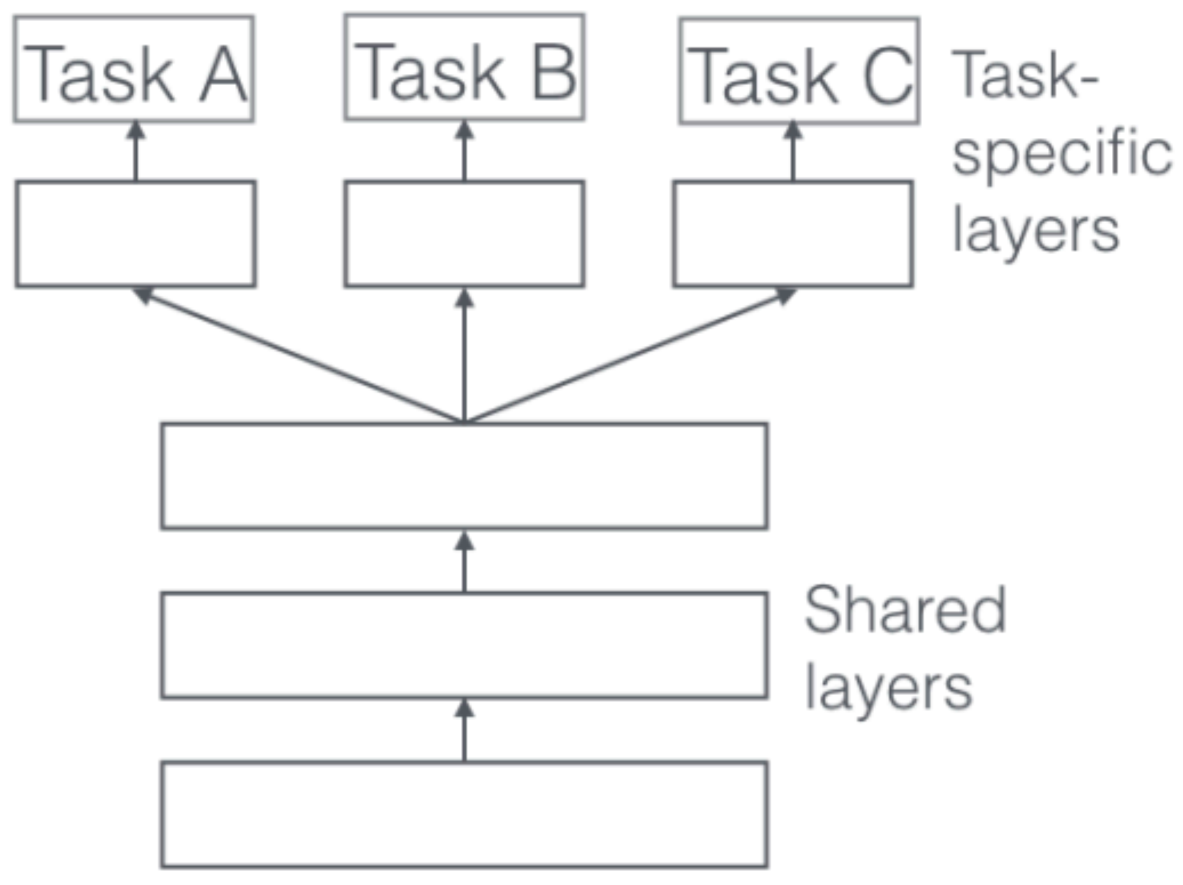


Diagram sources: distill.pub/2018/feature-wise-transformations/

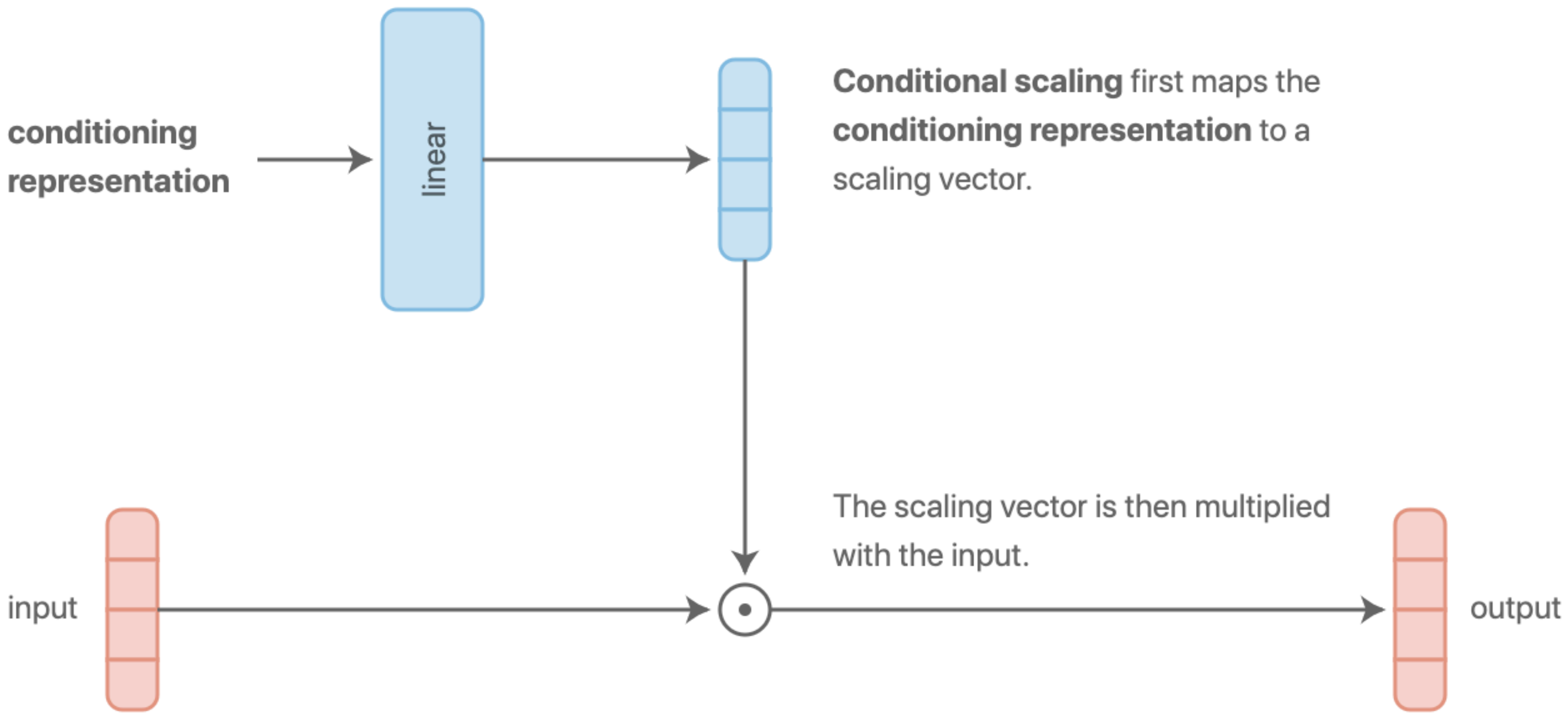
Conditioning: Some Common Choices

3. **Multi-head architecture**



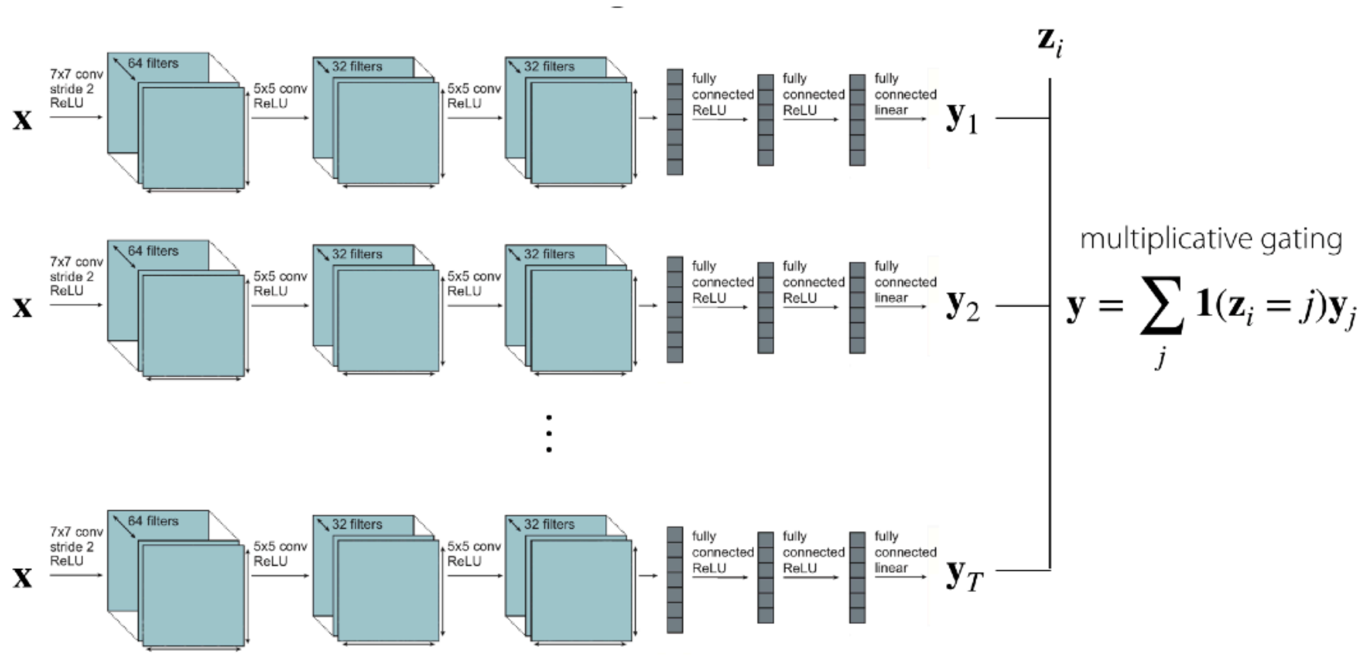
Ruder '17

4. **Multiplicative** conditioning



Why might multiplicative conditioning be a good idea?

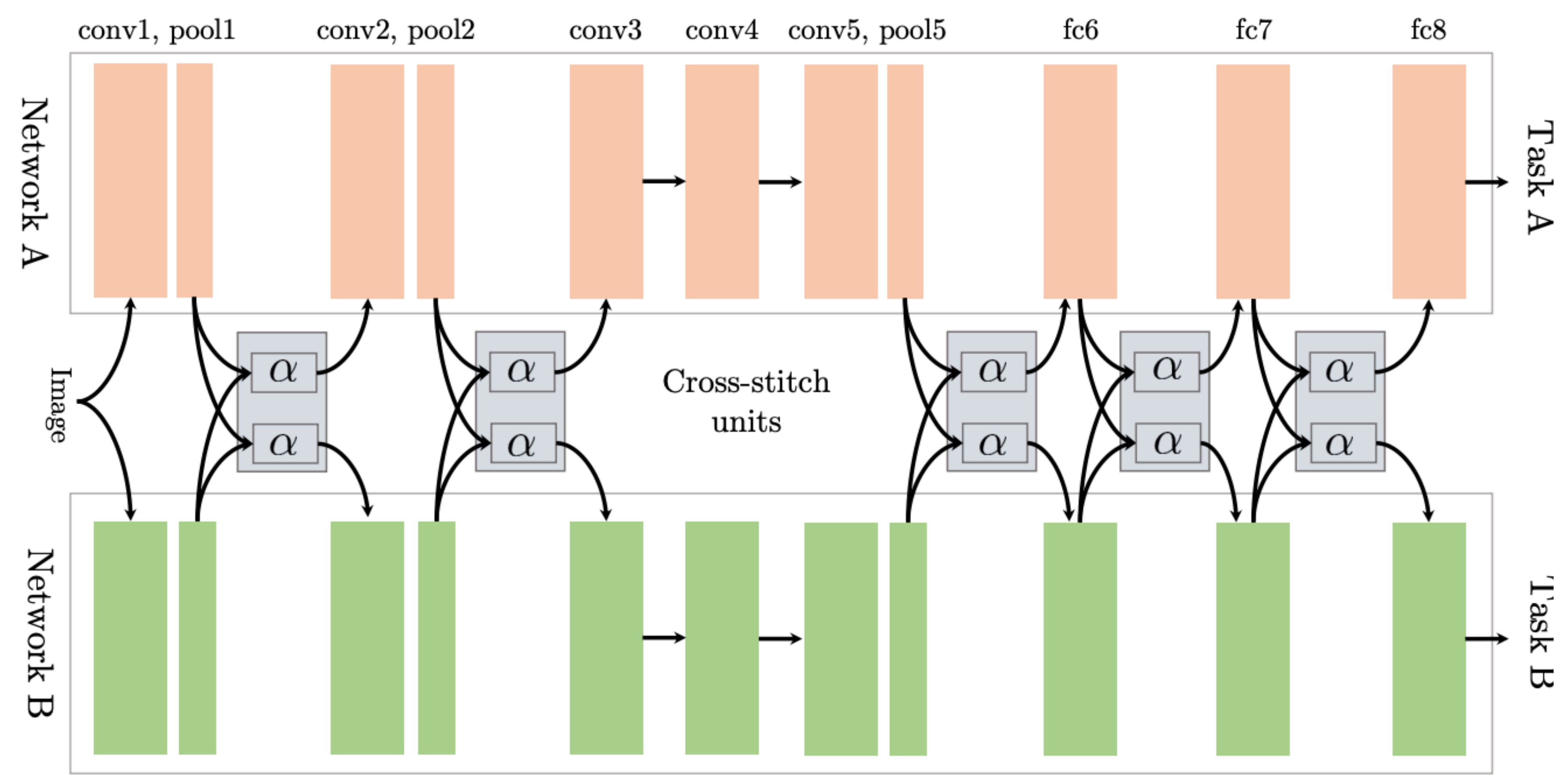
- more expressive per layer
- recall: multiplicative gating



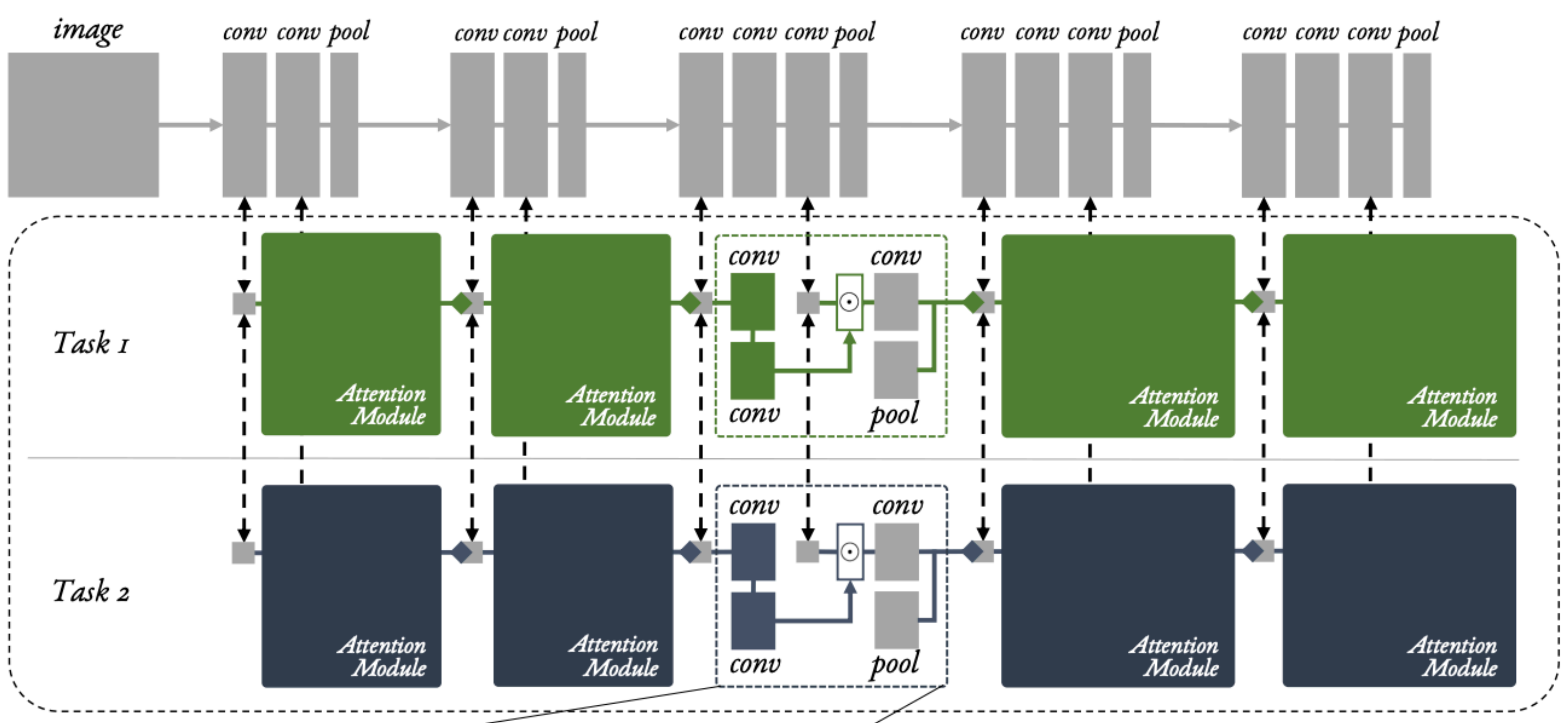
Multiplicative conditioning **generalizes** independent networks and independent heads.

Diagram sources: distill.pub/2018/feature-wise-transformations/

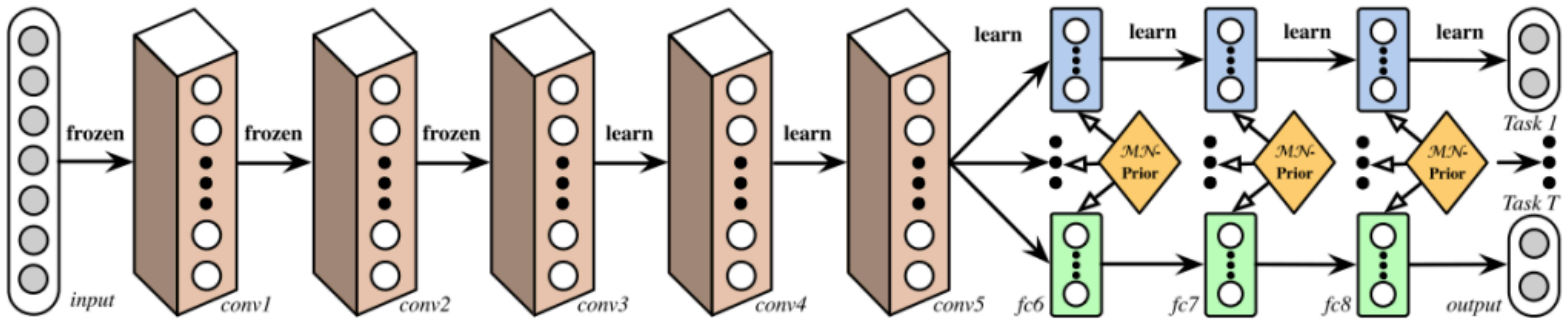
Conditioning: More Complex Choices



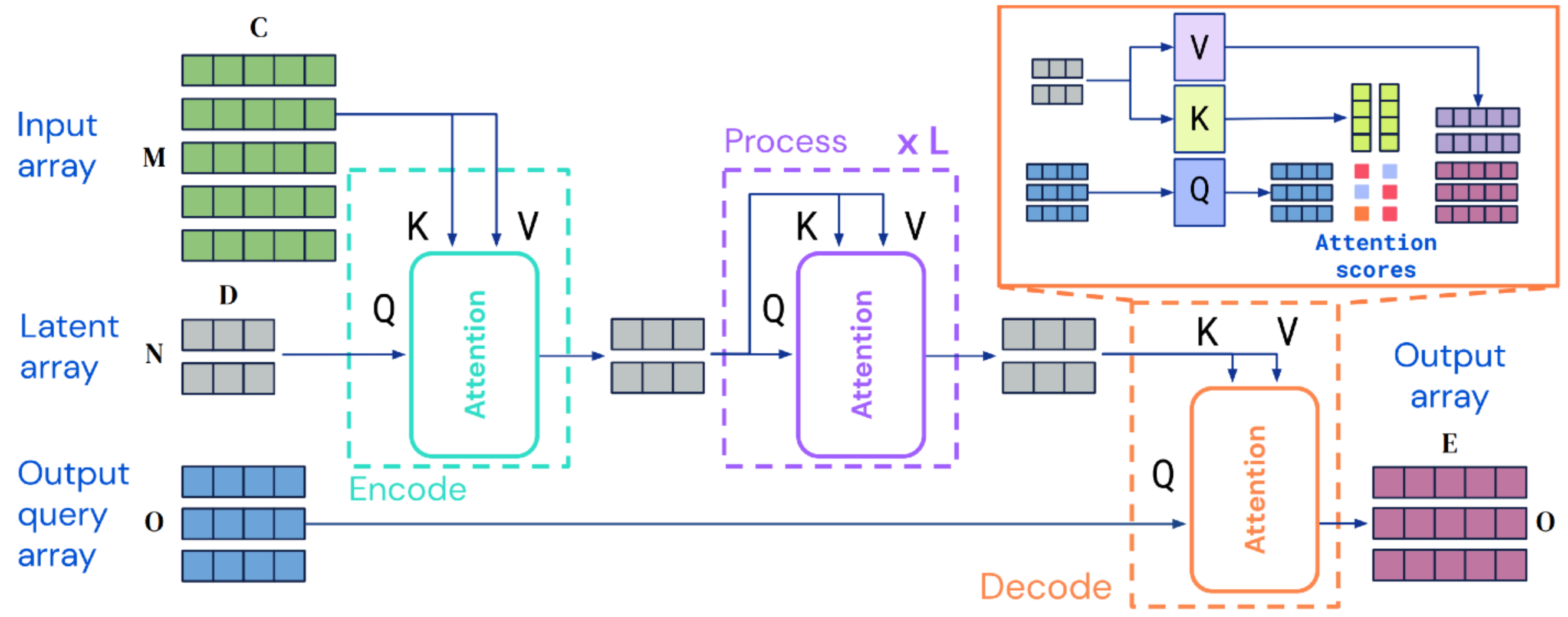
Cross-Stitch Networks. Misra, Shrivastava, Gupta, Hebert '16



Multi-Task Attention Network. Liu, Johns, Davison '18



Deep Relation Networks. Long, Wang '15



Perceiver IO. Jaegle et al. '21

Conditioning Choices

Unfortunately, these design decisions are like neural network architecture tuning:

- **problem dependent**
- largely guided by **intuition** or **knowledge** of the problem
- currently more of an **art** than a science

Model	How should the model be conditioned on $\mathbf{z}_i$?
	What parameters of the model should be shared?
Objective	How should the objective be formed?
Optimization	How should the objective be optimized?

Vanilla MTL objective:

$$\min_{\theta} \sum_{i=1}^{T} \mathscr{L}_i(\theta, \mathscr{D}_i)$$

Often want to weight tasks differently:

$$\min_{\theta} \sum_{i=1}^{T} w_i \mathscr{L}_i(\theta, \mathscr{D}_i)$$

How to choose w_i?

- manually based on importance or priority
- *dynamically* adjust throughout training

a. various heuristics

encourage gradients to have similar magnitudes
(Chen et al. GradNorm. ICML 2018)

b. optimize for the worst-case task loss

$$\min_{\theta} \max_{i} \mathscr{L}_i(\theta, \mathscr{D}_i)$$

(e.g. for task robustness, or for fairness)

Model	How should the model be conditioned on $\mathbf{z}_i$?
	What parameters of the model should be shared?
Objective	How should the objective be formed?
Optimization	**How should the objective be optimized?**

Optimizing the objective

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

Basic Version:

1. Sample mini-batch of tasks $\mathscr{B} \sim \{\mathscr{T}_i\}$
2. Sample mini-batch datapoints for each task $\mathscr{D}_i^b \sim \mathscr{D}_i$
3. Compute loss on the mini-batch: $\hat{\mathscr{L}}(\theta, \mathscr{B}) = \sum_{\mathscr{T}_k \in \mathscr{B}} \mathscr{L}_k(\theta, \mathscr{D}_k^b)$
4. Backpropagate loss to compute gradient $\nabla_\theta \hat{\mathscr{L}}$
5. Apply gradient with your favorite neural net optimizer (e.g. Adam)

Note: This ensures that tasks are sampled uniformly, regardless of data quantities.

Tip: For regression problems, make sure your task labels are on the same scale!

Challenges

Challenge #1: Negative transfer

Negative transfer: Sometimes independent networks work the best.

Multi-Task CIFAR-100

recent approaches

	% 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)

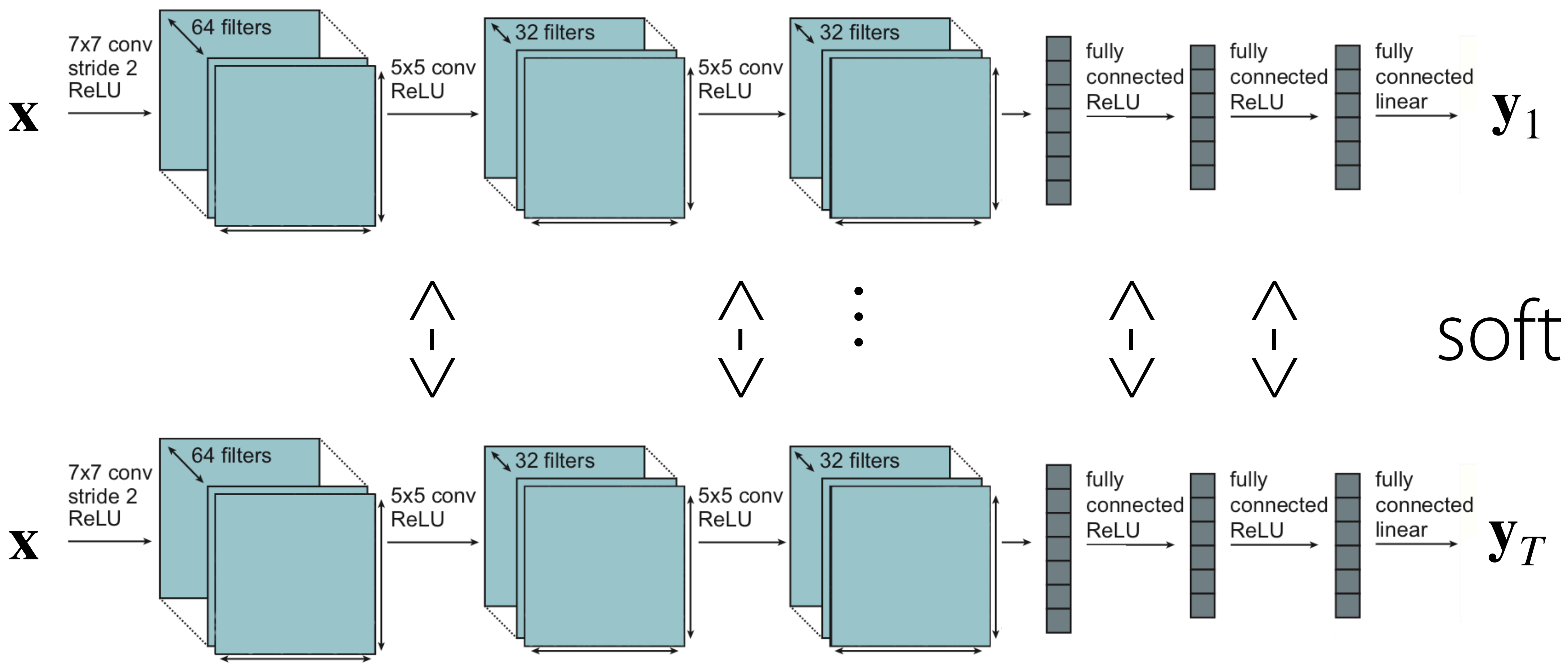
Why?

- **optimization challenges**
  - caused by cross-task interference
  - tasks may learn at different rates
- **limited representational capacity**
  - multi-task networks often need to be *much larger* than their single-task counterparts

If you have negative transfer, **share less** across tasks.

It's not just a binary decision!

$$\min_{\theta^{sh},\theta^1,\ldots,\theta^T} \sum_{i=1}^{T} \mathcal{L}_i(\{\theta^{sh},\theta^i\},\mathcal{D}_i) + \lambda \underbrace{\sum_{i'=1}^{T} \|\theta^i - \theta^{i'}\|}_{\text{“soft parameter sharing”}}$$



softly constrained weights

+ allows for more fluid degrees of parameter sharing

- yet another set of design decisions / hyperparameters

- more memory intensive

Challenge #2: Overfitting

You may not be sharing enough!

Multi-task learning <-> a form of regularization

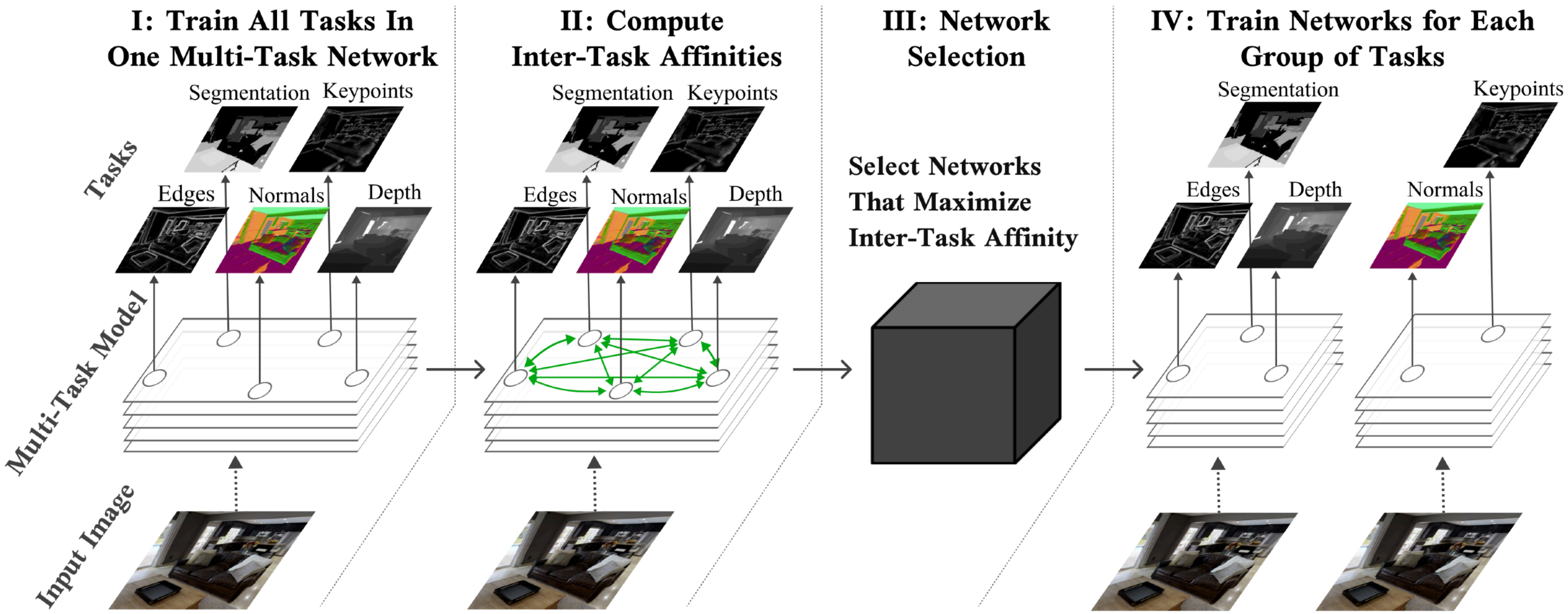
Solution: Share more.

Challenge #3: What if you have a lot of tasks?

Should you train all of them together? Which ones will be complementary?

The bad news: No closed-form solution for measuring task similarity.

The good news: There are ways to approximate it from one training run.



Fifty, Amid, Zhao, Yu, Anil, Finn. *Efficiently Identifying Task Groupings for Multi-Task Learning*. NeurIPS 2021

Multi-Task Learning Recap

A task: $\mathscr{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathscr{L}_i\}$

Corresponding datasets: $\mathscr{D}_i^{tr}$ $\mathscr{D}_i^{test}$

Model Architecture

- multiplicative vs. additive conditioning on $\mathbf{z}_i$
- share more vs. less depending on observed transfer

Objective & Optimization

$$\min_\theta \sum_{i=1}^{T} w_i \mathscr{L}_i(\theta, \mathscr{D}_i^{tr})$$

- choosing task weights
- stratified mini-batches

Plan for Today

Multi-Task Learning
- Problem statement
- Models, objectives, optimization
- Challenges
- **Case study of real-world multi-task learning**

Case study

Recommending What Video to Watch Next: A Multitask Ranking System

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi
Google, Inc.
{zhezhao,lichan,liwei,jilinc,aniruddhnath,shawnandrews,aditeek,nlogn,xinyang,edchi}@google.com

Goal: Make recommendations for YouTube

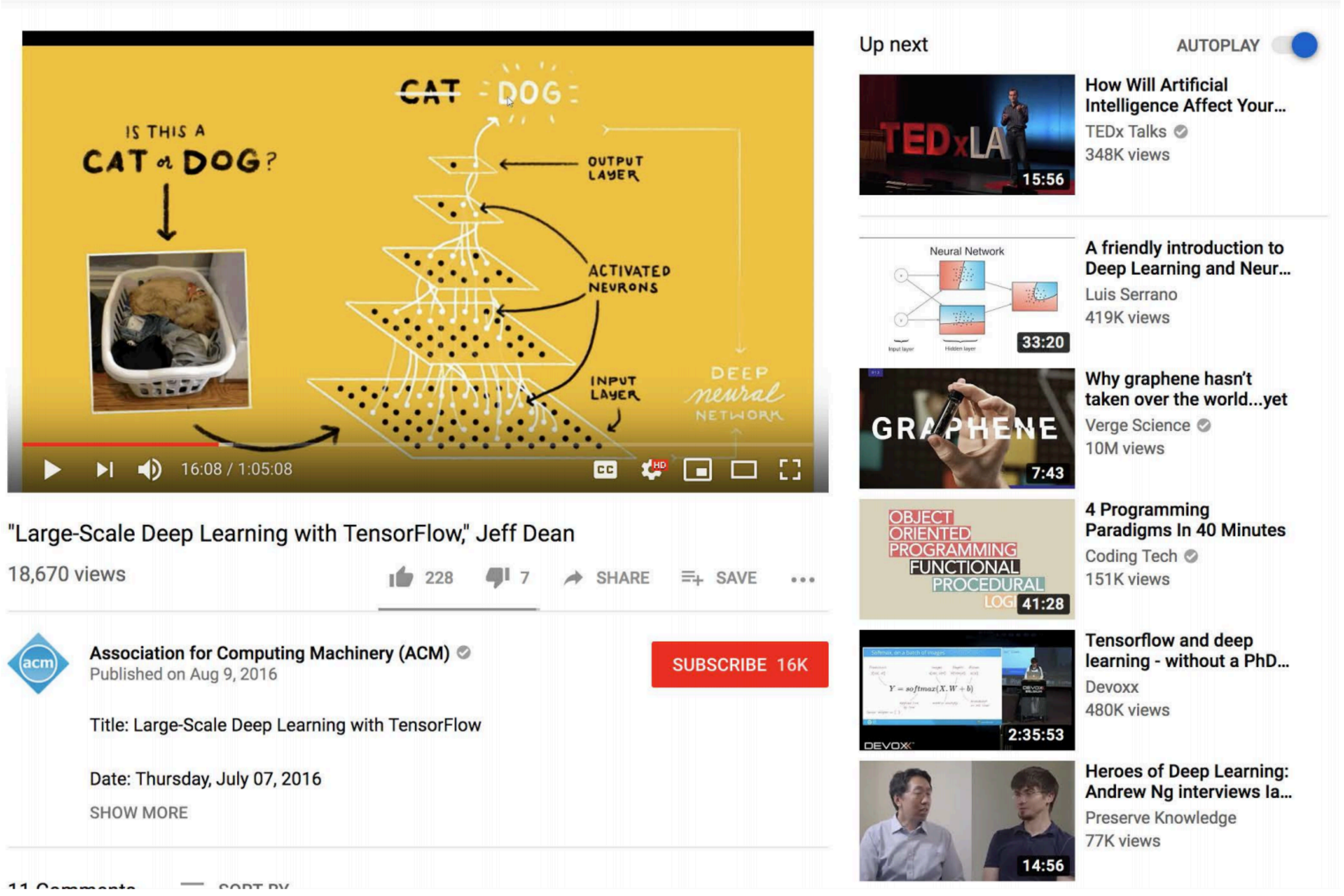


Figure 4: Recommending what to watch next on YouTube.

Framework Set-Up

Input: what the user is currently watching (query video) + user features

1. Generate a few hundred of candidate videos
2. Rank candidates
3. Serve top ranking videos to the user

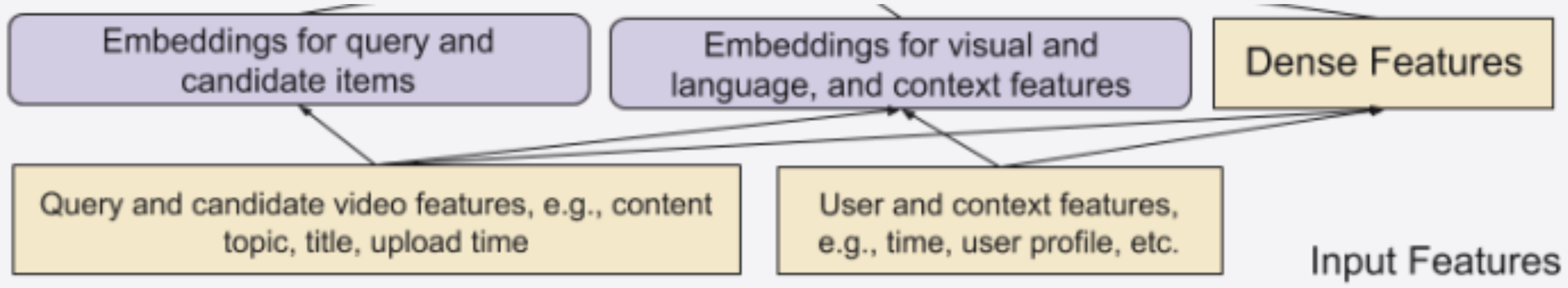
Candidate videos: pool videos from multiple candidate generation algorithms

- matching topics of query video
- videos most frequently watched with query video
- And others

Ranking: central topic of this paper

The Ranking Problem

Input: query video, candidate video, user & context features



Model output: engagement and satisfaction with candidate video

Engagement:
- binary classification tasks like **clicks**
- regression tasks related to **time spent**

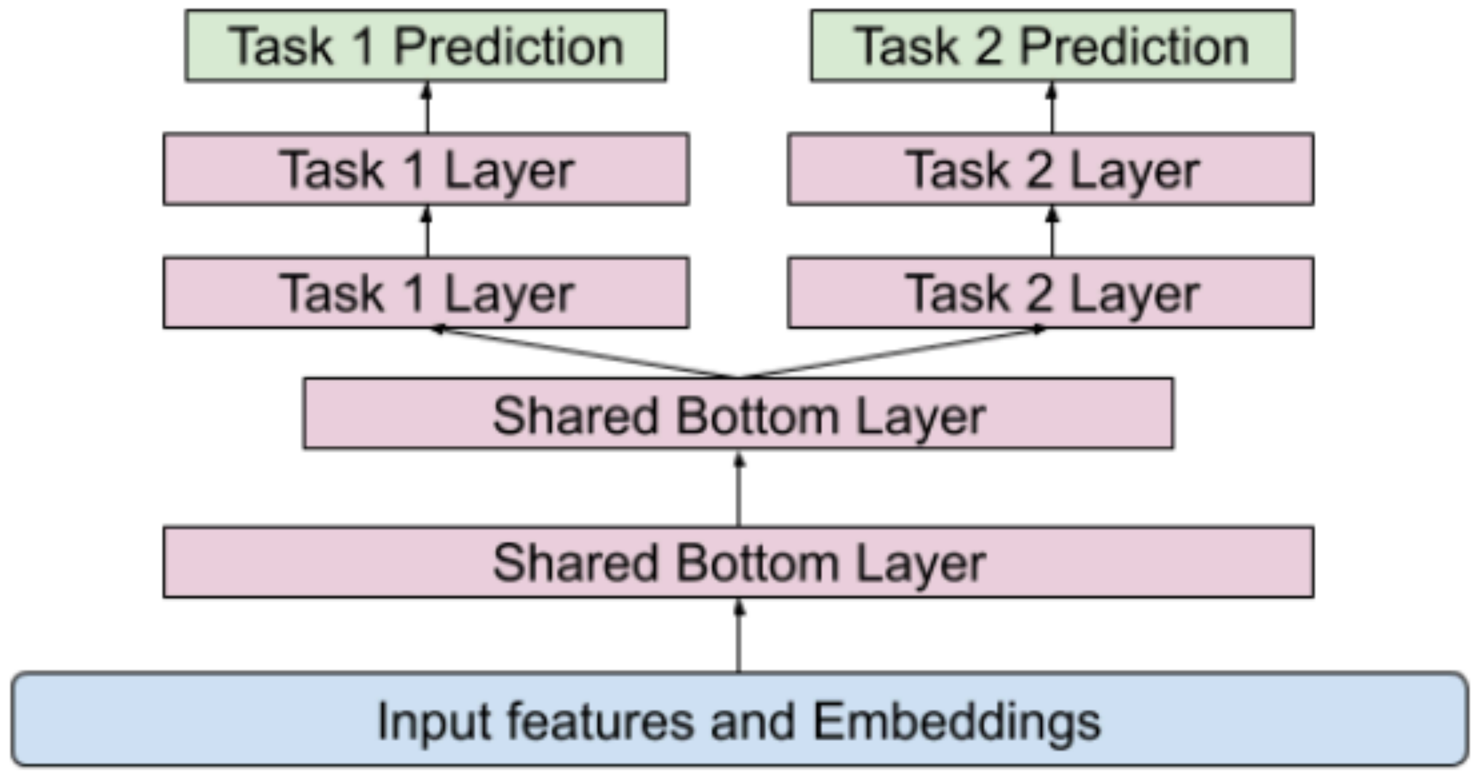
Satisfaction:
- binary classification tasks like **clicking "like"**
- regression tasks such as **rating**

Weighted combination of **engagement** & **satisfaction** predictions -> **ranking score**
score weights manually tuned

Question: Are these objectives reasonable? What are some of the issues that might come up?

The Architecture

Basic option: “Shared-Bottom Model"
(i.e. multi-head architecture)

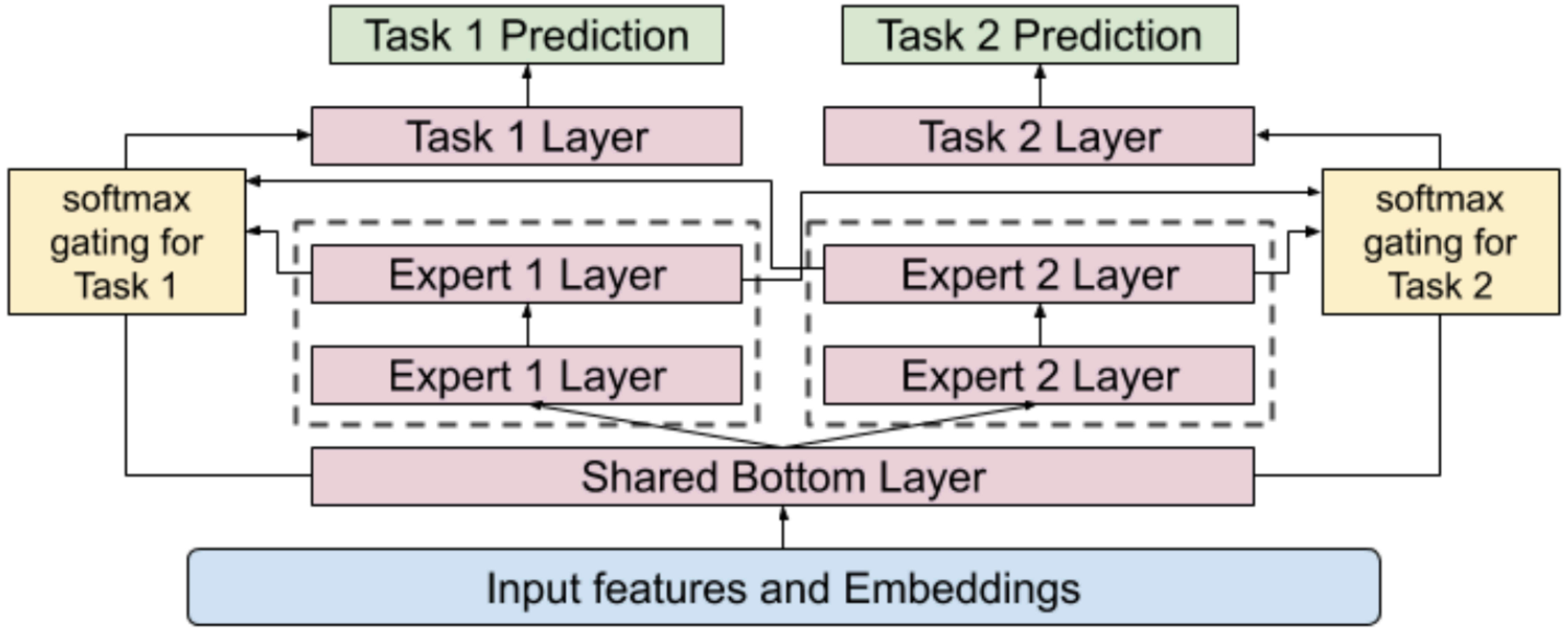


(a) Shared-Bottom Model with shared bottom hidden layers and separate towers for two tasks.

-> harms learning when correlation between tasks is low

The Architecture

Instead: use a form of soft-parameter sharing
"**Multi-gate Mixture-of-Experts (MMoE)**"



(b) Multi-gate Mixture-of-Expert Model with one shared bottom layer and separate hidden layers for two tasks.

Allow different parts of the network to "specialize"

expert neural networks $f_i(x)$

Decide which expert to use for input x, task k:

$$g^k(x) = \text{softmax}(W_{g^k} x)$$

Compute features from selected expert:

$$f^k(x) = \sum_{i=1}^{n} g^k_{(i)}(x) f_i(x)$$

Compute output: $y_k = h^k(f^k(x)),$

Experiments

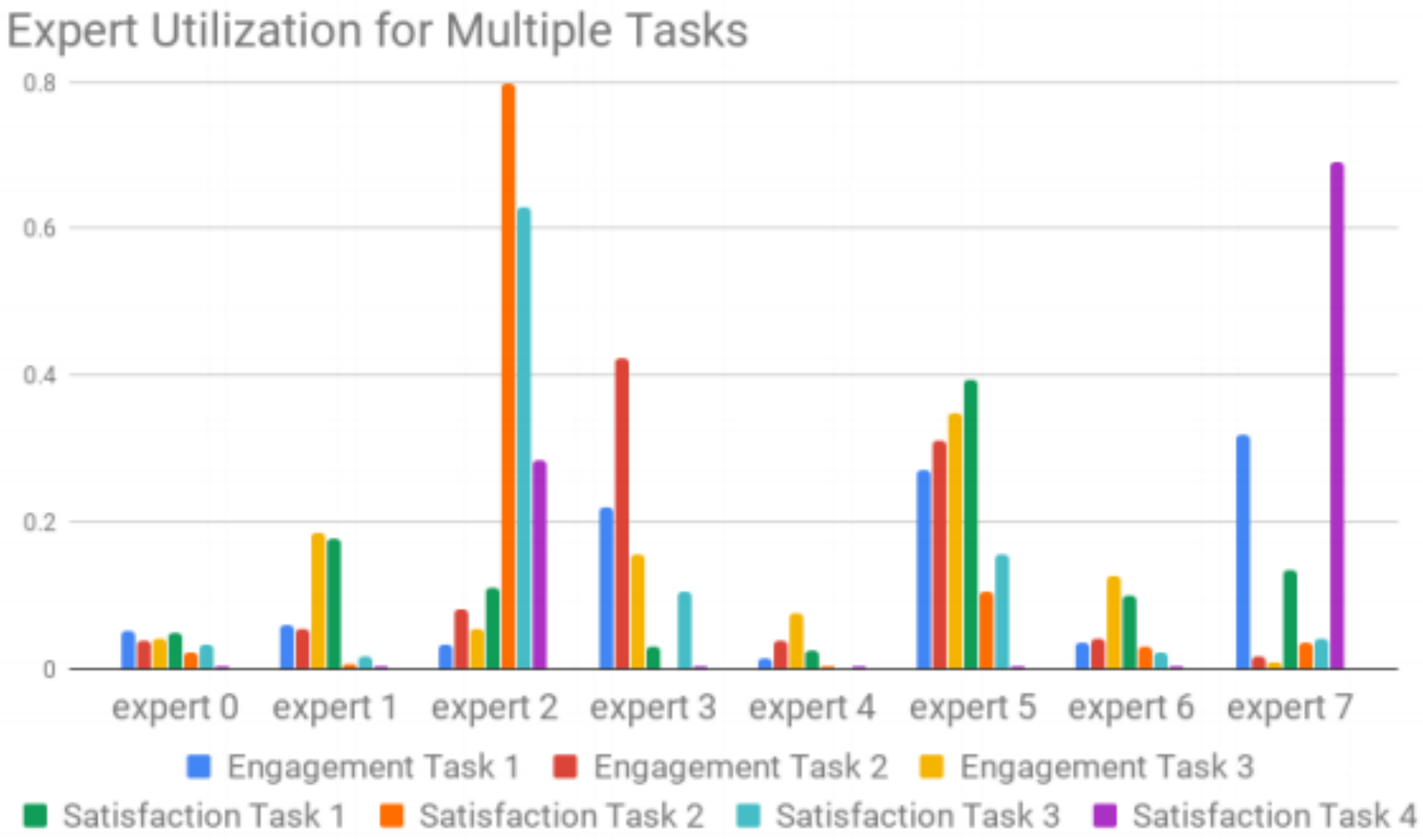
Set-Up

- Implementation in TensorFlow, TPUs
- Train in ***temporal order***, running training continuously to consume newly arriving data
- **Online A/B testing** in comparison to production system
    - live metrics based on time spent, survey responses, rate of dismissals
- Model **computational efficiency** matters

Results

Model Architecture	Number of Multiplications	Engagement Metric	Satisfaction Metric
Shared-Bottom	3.7M	/	/
Shared-Bottom	6.1M	+0.1%	+ 1.89%
MMoE (4 experts)	3.7M	+0.20%	+ 1.22%
MMoE (8 Experts)	6.1M	+0.45%	+ 3.07%

Table 1: YouTube live experiment results for MMoE.



Found 20% chance of gating polarization during distributed training -> use drop-out on experts

Lecture Recap

- Multi-task learning learns neural network conditioned on task descriptor $\mathbf{z}_i$
- Choice of task weighting w_i affects prioritization of tasks.
- Choice of how to condition on $\mathbf{z}_i$ affects how parameters are shared.
  - If you observe negative transfer, share less.

    If you observe overfitting, try sharing more.

Goals for by the end of lecture:

- Understand the key design decisions when building multi-task learning systems

Reminders

Homework 0 due **Monday 10/3** at **11:59 pm PT**.

PyTorch review session **tomorrow at 4:30 pm PT.**

Office hours start today

Next time: Transfer learning basics, meta-learning problem statement