

Multi-Task Learning & Transfer Learning Basics

CS 330

Logistics

Optional homework 0 due **Monday 9/27.**

PyTorch review session **tomorrow at 6:00 pm PT.**

Project guidelines posted.

Office hours start today

Plan for Today

Multi-Task Learning

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

Transfer Learning

- Pre-training & fine-tuning

Goals for by the end of lecture:

- Know the **key design decisions** when building multi-task learning systems
- Understand the **difference** between multi-task learning and **transfer learning**
- Understand the **basics** of **transfer learning**

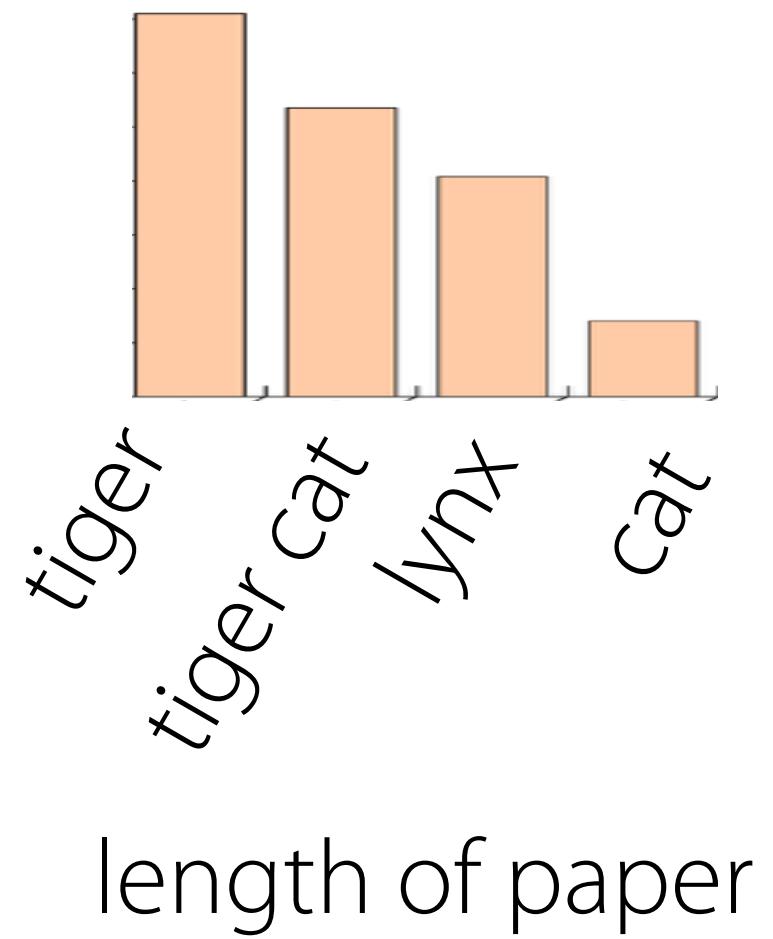
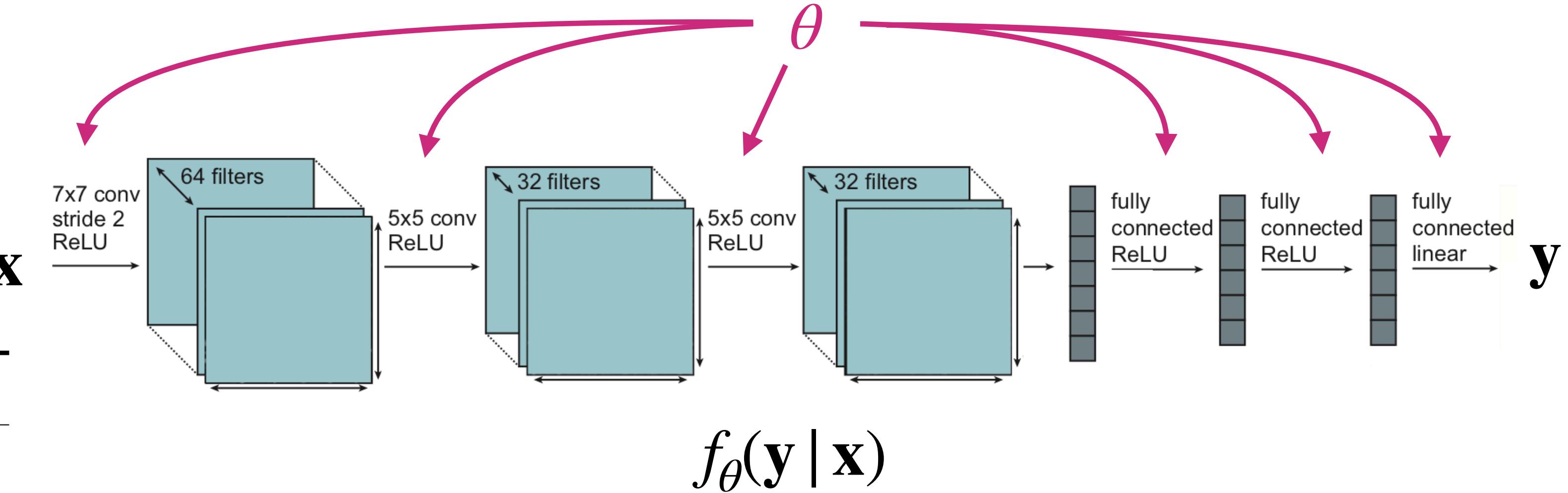
Multi-Task Learning

Some notation



\mathbf{x}

ImageNet Classification with Deep Convolutional Neural Networks



Single-task learning: $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})_k\}$
[supervised]

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{D})$$

Typical loss: negative log likelihood

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

What is a task? (more formally this time)

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

data generating distributions

Corresponding datasets: \mathcal{D}_i^{tr} \mathcal{D}_i^{test}
will use \mathcal{D}_i as shorthand for \mathcal{D}_i^{tr} :

Examples of Tasks

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

data generating distributions

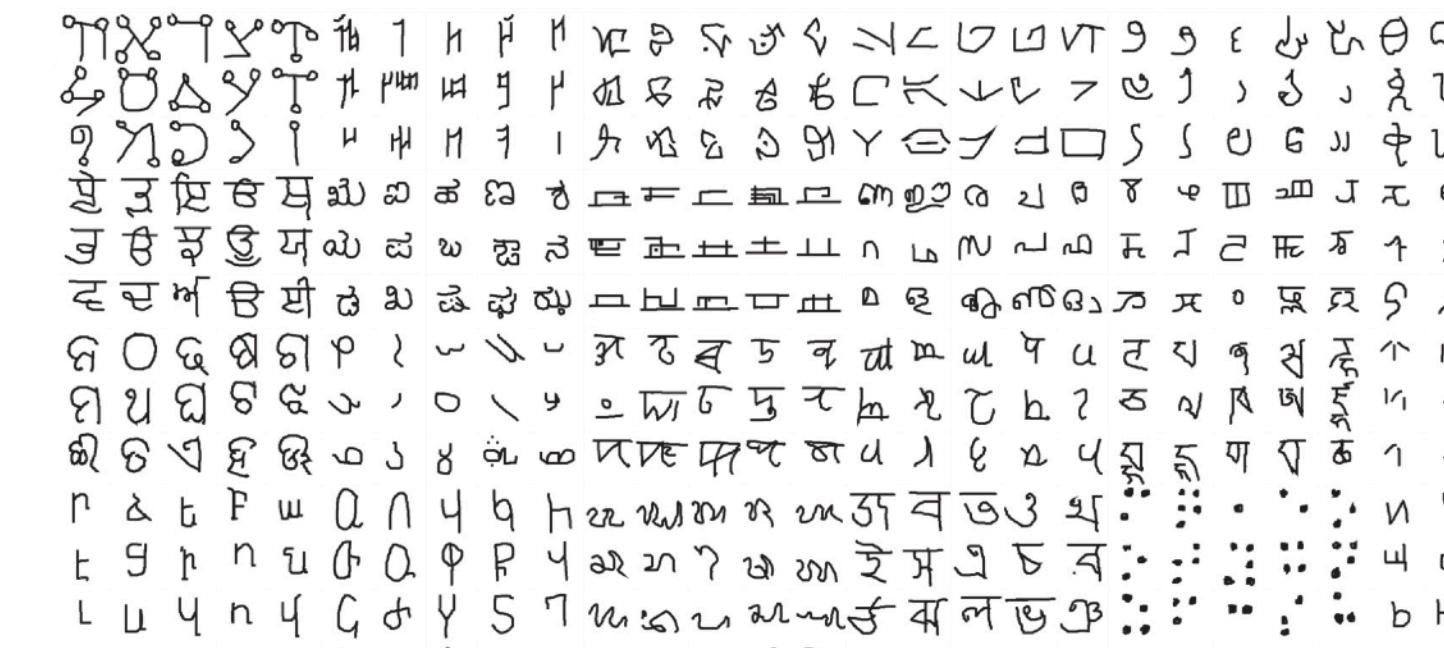
Corresponding datasets: \mathcal{D}_i^{tr} \mathcal{D}_i^{test}

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

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

e.g. per-language
handwriting recognition

e.g. personalized
spam filter



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

e.g. CelebA attribute recognition

e.g. scene understanding

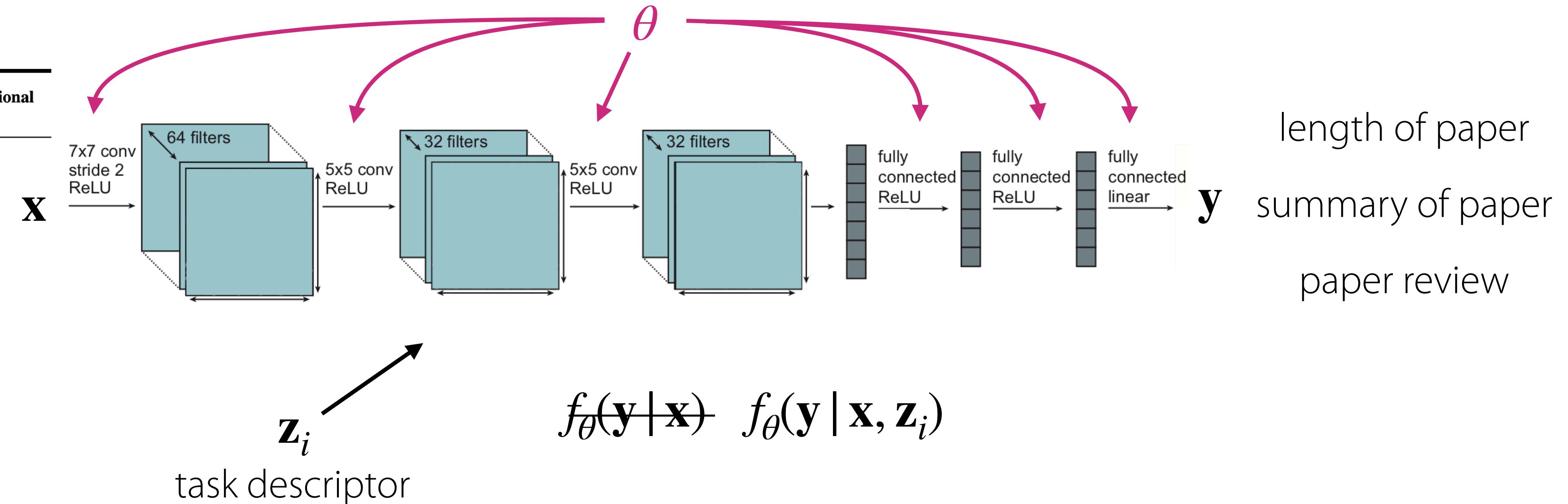


$$L_{\text{tot}} = w_{\text{depth}} L_{\text{depth}} + w_{\text{kpt}} L_{\text{kpt}} + w_{\text{normals}} L_{\text{normals}}$$

When might \mathcal{L}_i vary across tasks?

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

ImageNet Classification with Deep Convolutional Neural Networks



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

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

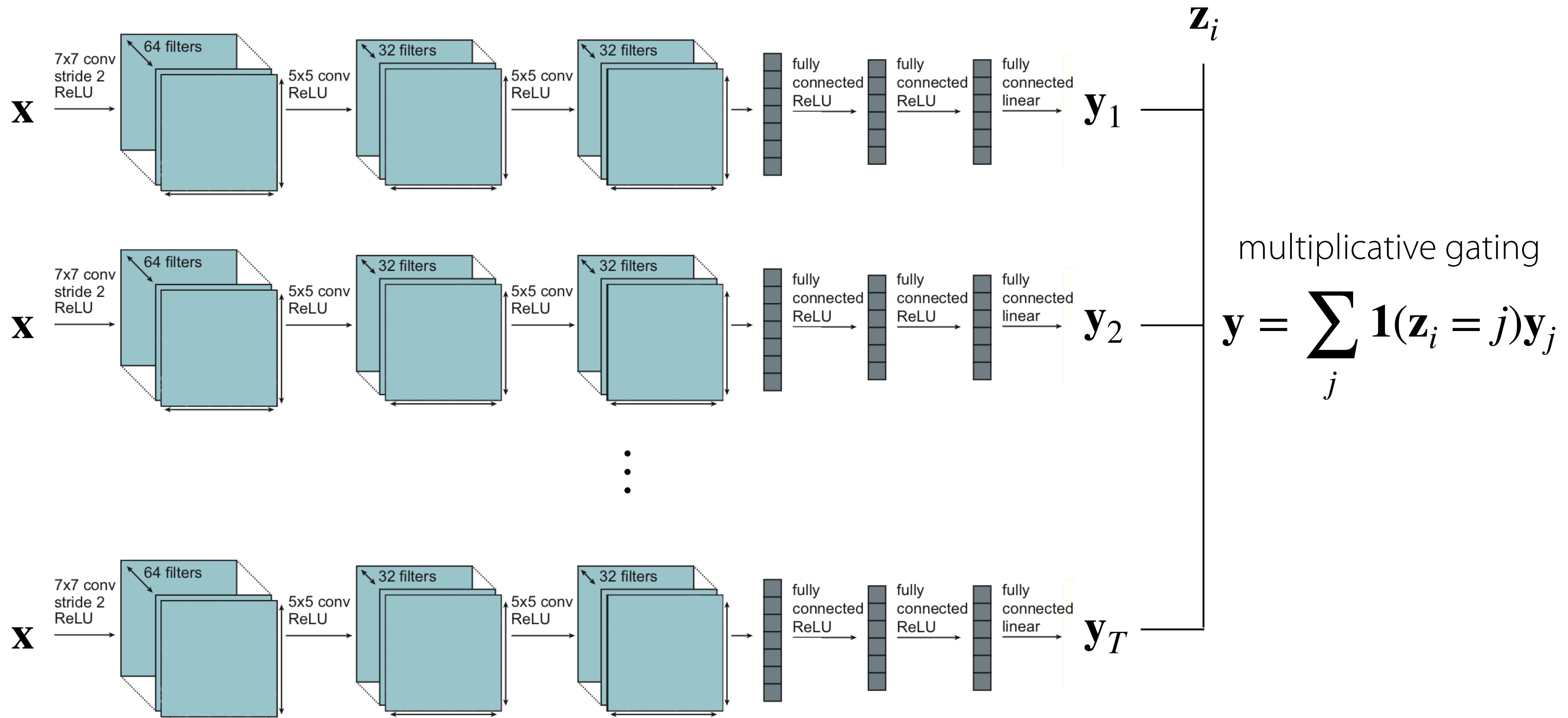
How should the objective be optimized?

Conditioning on the task

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

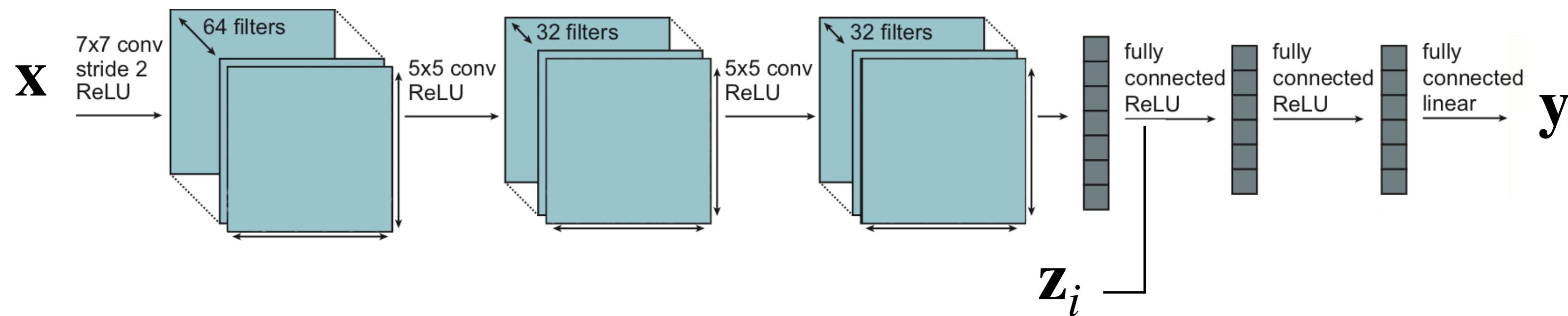
Question: How should you condition on the task in order to share as little as possible?

Conditioning on the task



→ 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:

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

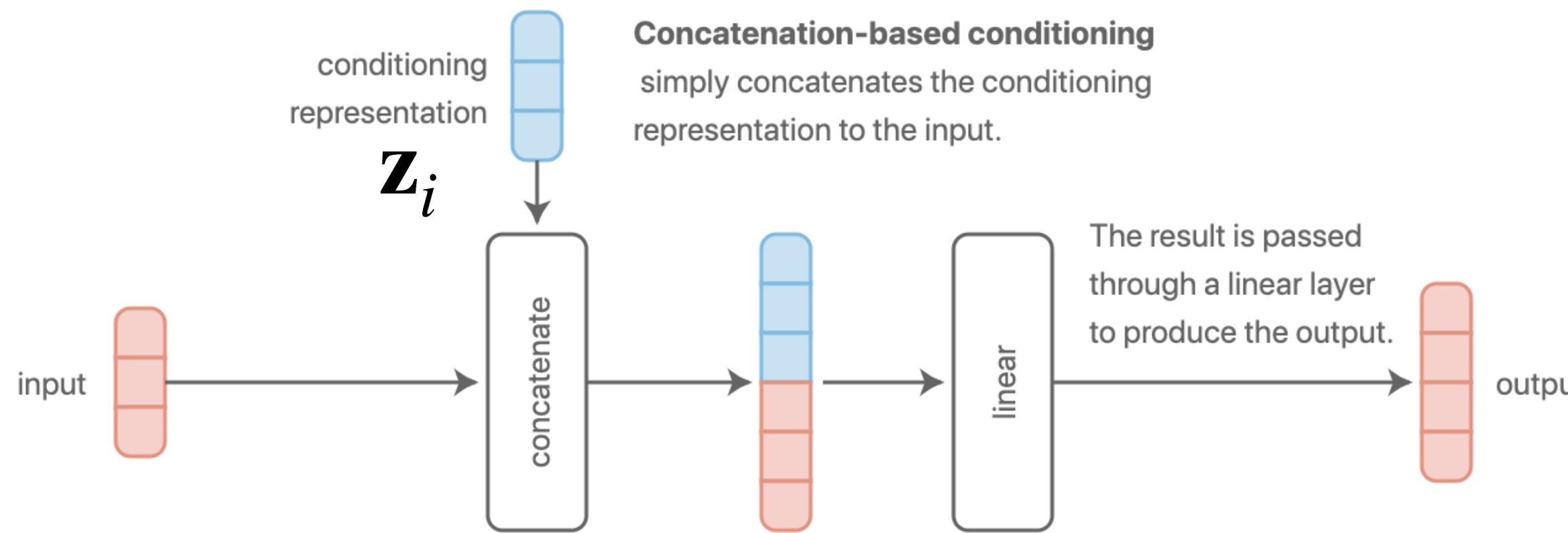
Choosing how to condition on \mathbf{z}_i

equivalent to

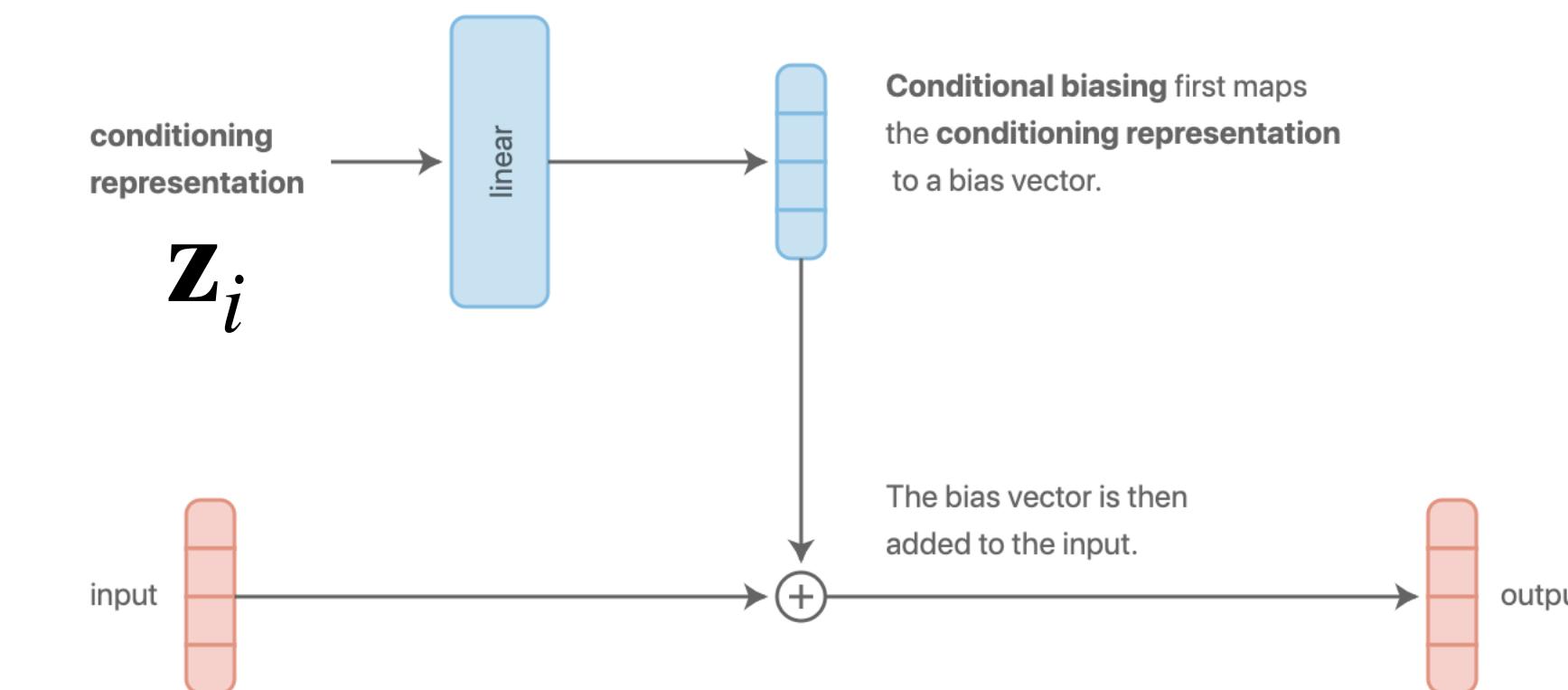
Choosing how & where to share parameters

Conditioning: Some Common Choices

1. Concatenation-based conditioning



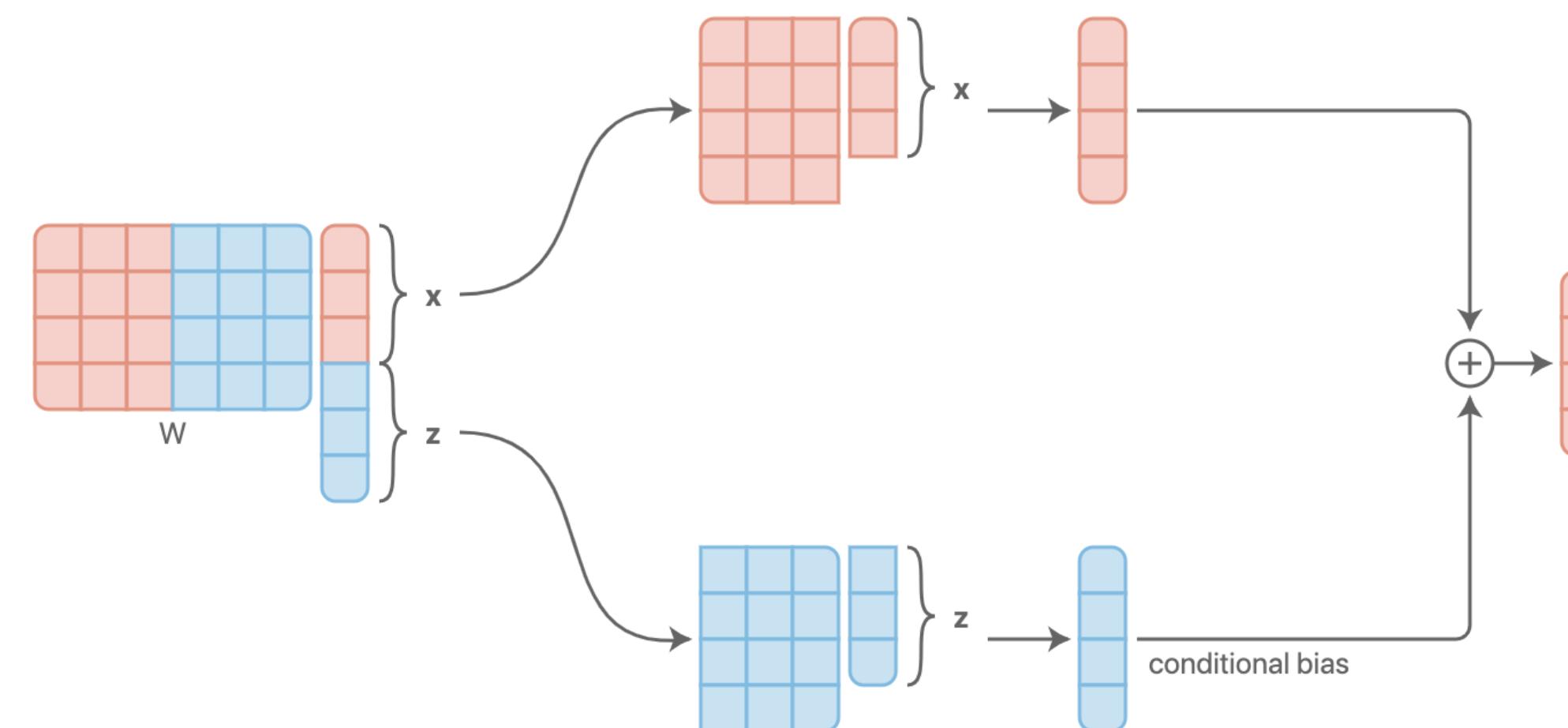
2. Additive conditioning



These are actually equivalent!

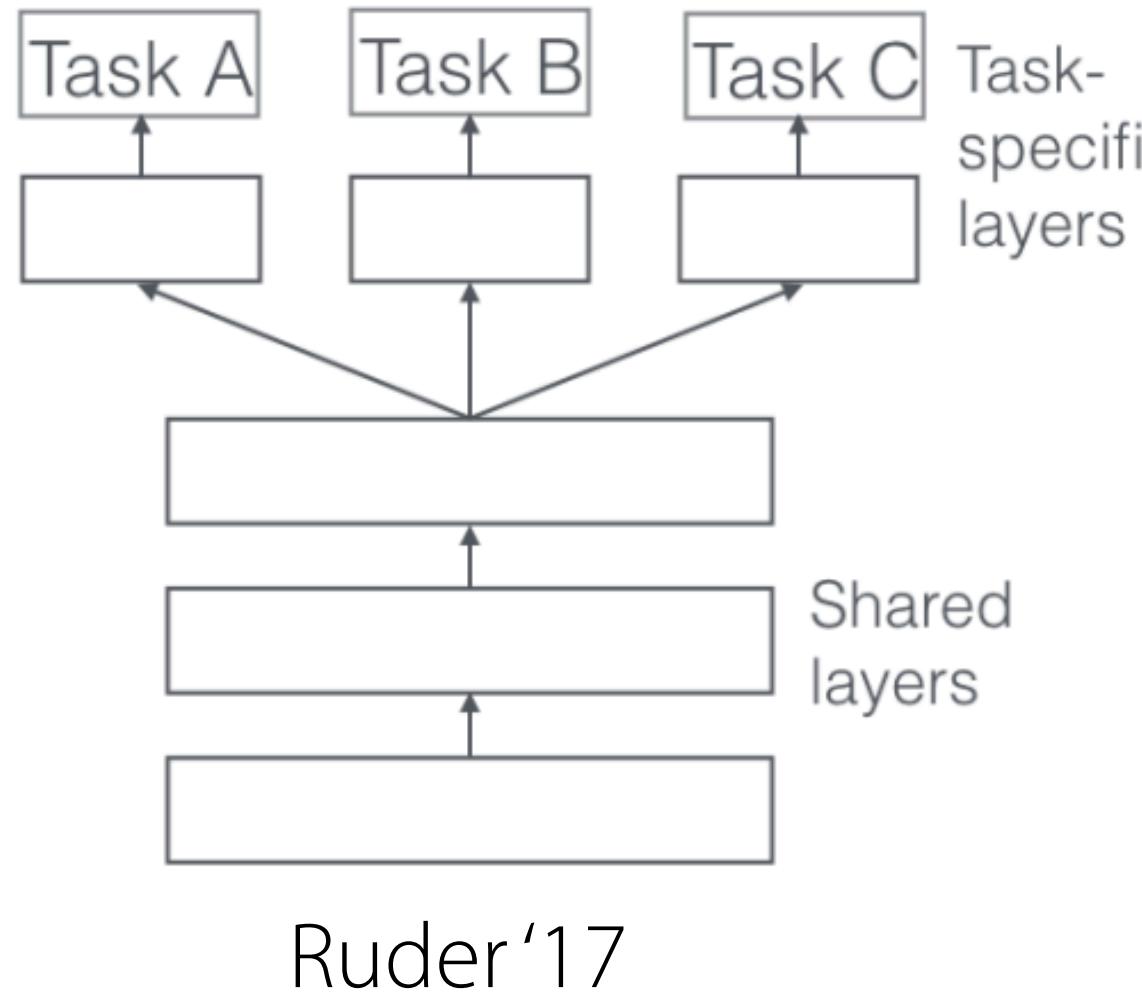
Question: why are they the same thing? (raise your hand)

Concat followed by a fully-connected layer:

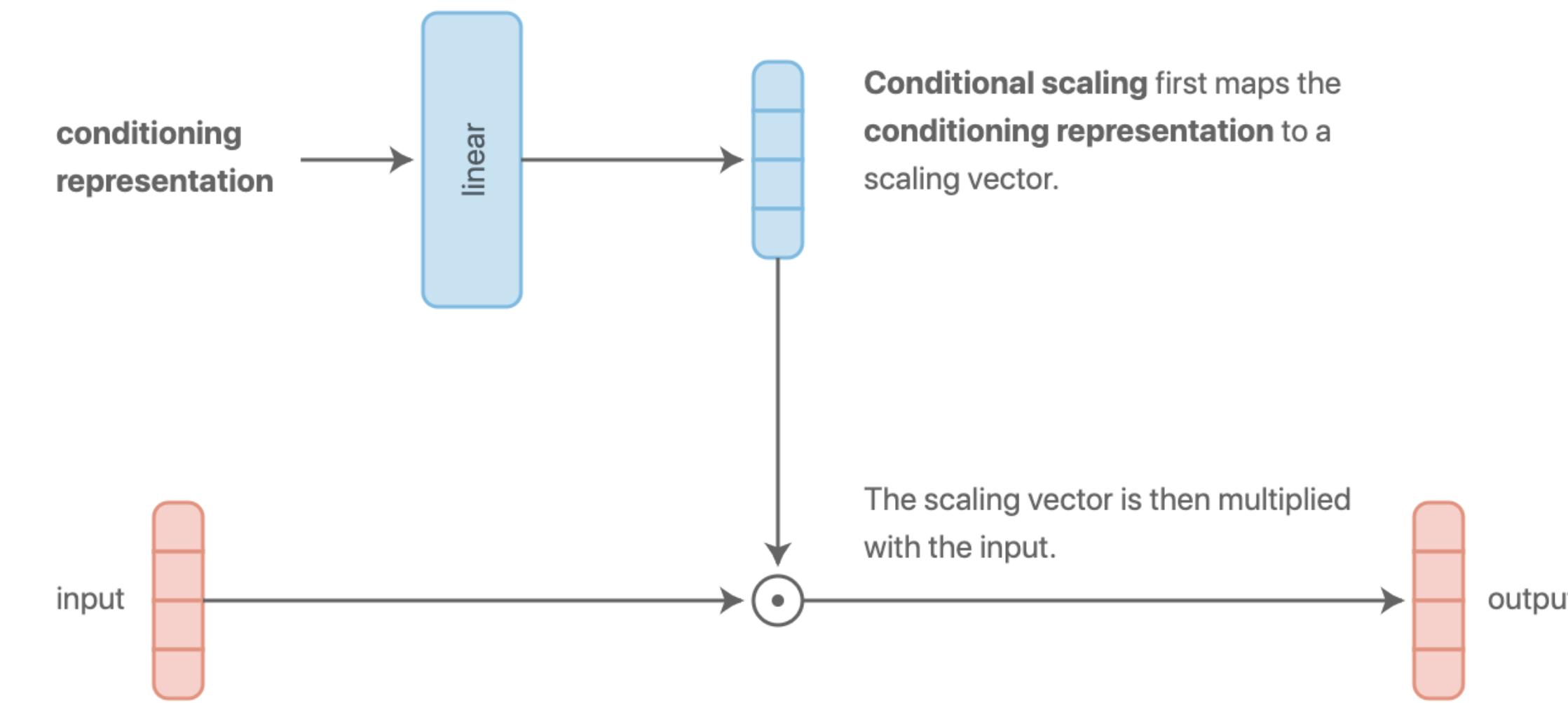


Conditioning: Some Common Choices

3. Multi-head architecture

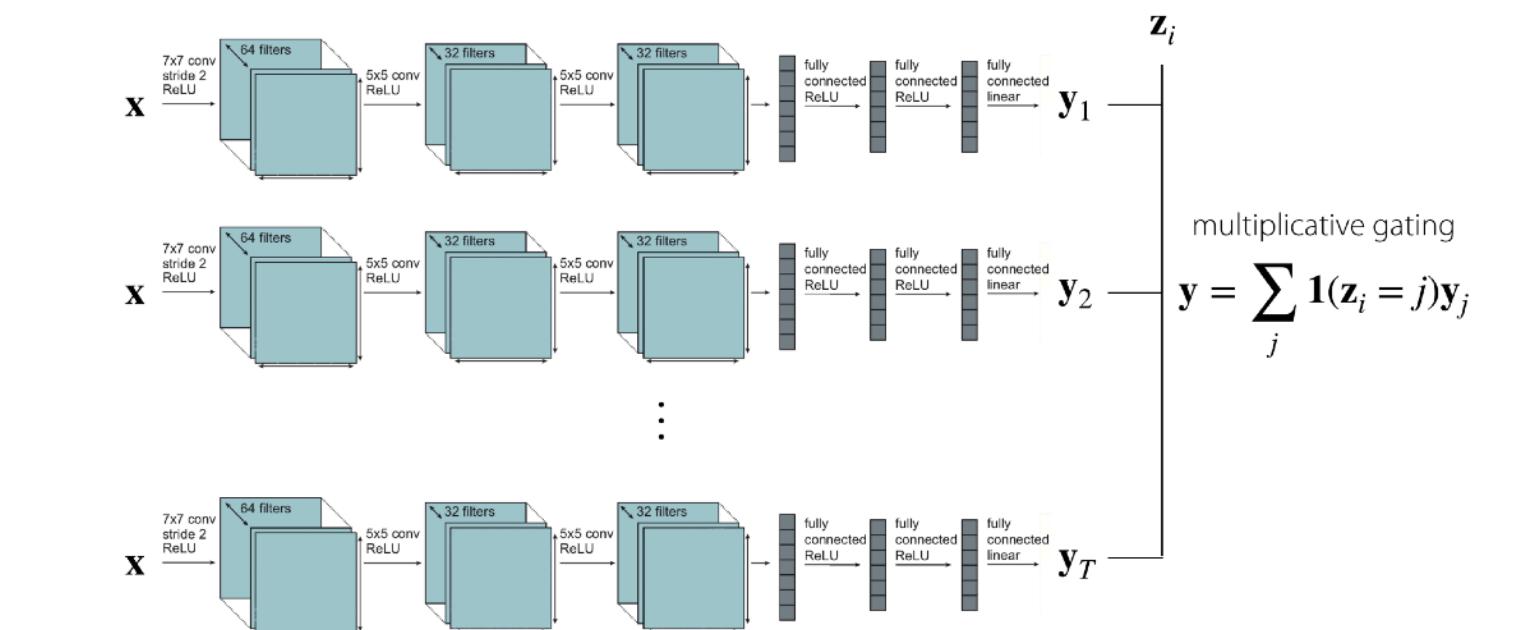


4. Multiplicative conditioning



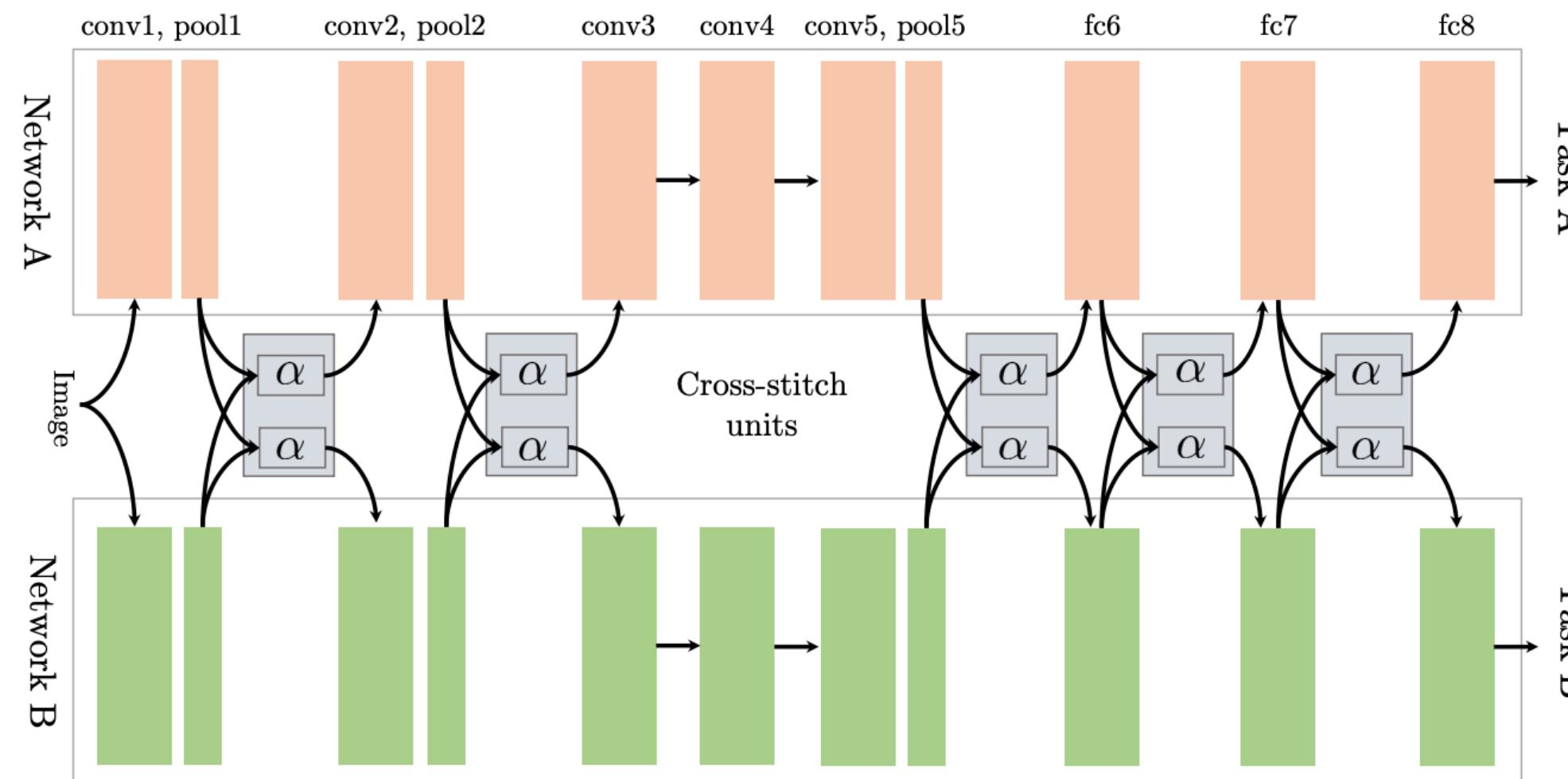
Why might multiplicative conditioning be a good idea?

- more expressive per layer
- recall: multiplicative gating

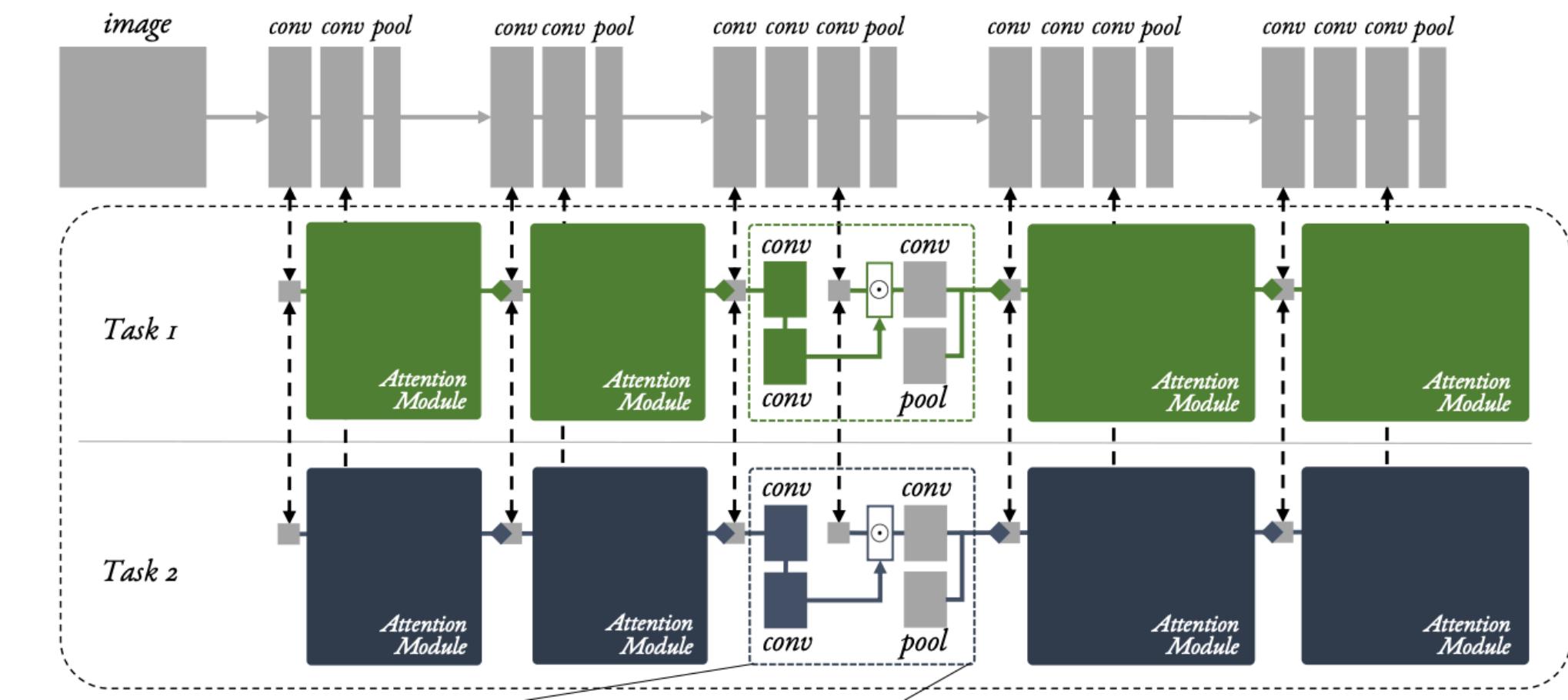


Multiplicative conditioning **generalizes** independent networks and independent heads.

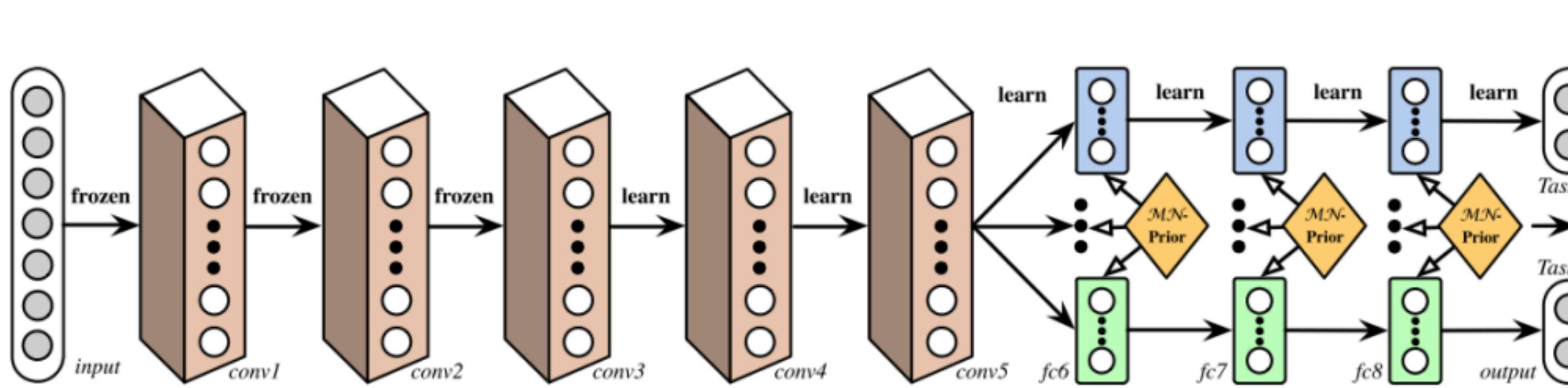
Conditioning: More Complex Choices



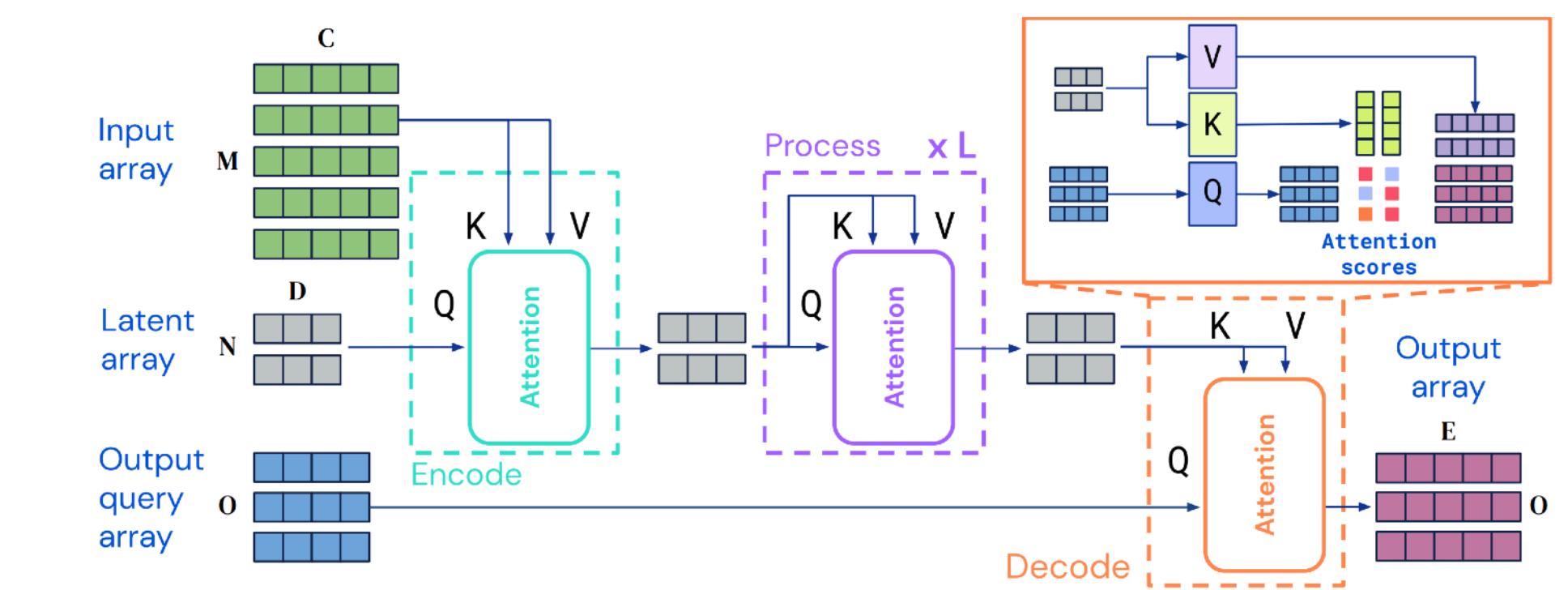
Cross-Stitch Networks. Misra, Shrivastava, Gupta, Hebert '16



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

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

How to choose w_i ?

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

Often want to weight tasks differently:

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

a. various heuristics

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

b. optimize for the worst-case task loss

$$\min_{\theta} \max_i \mathcal{L}_i(\theta, \mathcal{D}_i)$$

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

Basic Version:

1. Sample mini-batch of tasks $\mathcal{B} \sim \{\mathcal{T}_i\}$
2. Sample mini-batch datapoints for each task $\mathcal{D}_i^b \sim \mathcal{D}_i$
3. Compute loss on the mini-batch: $\hat{\mathcal{L}}(\theta, \mathcal{B}) = \sum_{\mathcal{T}_k \in \mathcal{B}} \mathcal{L}_k(\theta, \mathcal{D}_k^b)$
4. Backpropagate loss to compute gradient $\nabla_{\theta} \hat{\mathcal{L}}$
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	
task specific, all-fc (Rosenbaum et al., 2018)	49	}
cross stitch, all-fc (Misra et al., 2016b)	53	}
independent	67.7	}

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

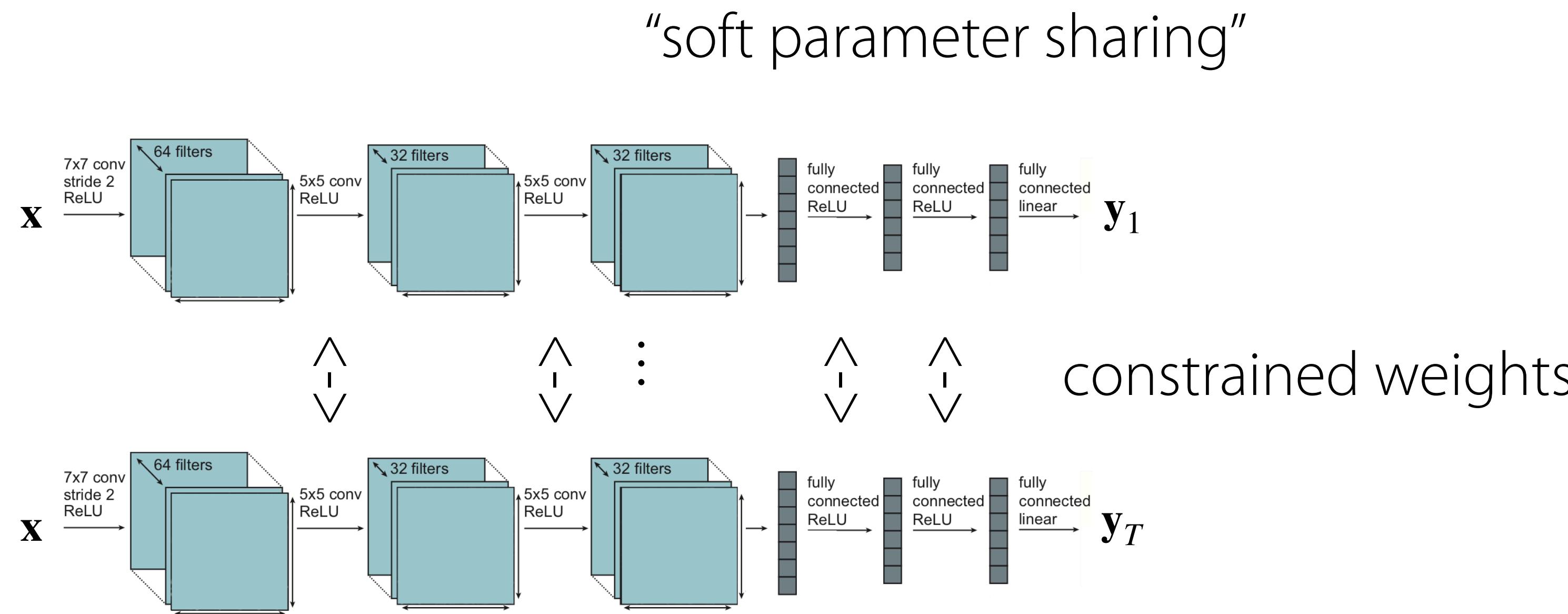
Why?

- **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!

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



- + allows for more fluid degrees of parameter sharing
- yet another set of design decisions / hyperparameters

Challenge #2: Overfitting

You may not be sharing enough!

Multi-task learning <-> a form of regularization

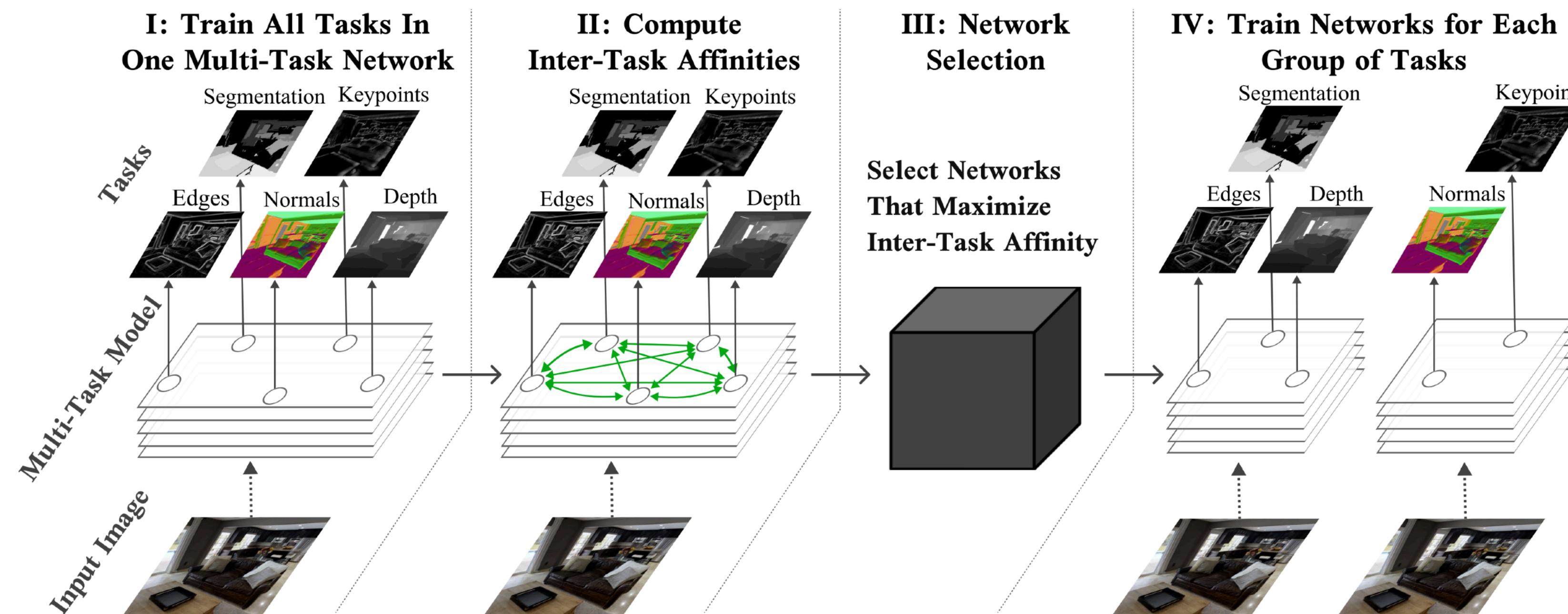
Solution: Share more.

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*. 2021

Plan for Today

Multi-Task Learning

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

Transfer Learning

- Pre-training & fine-tuning

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

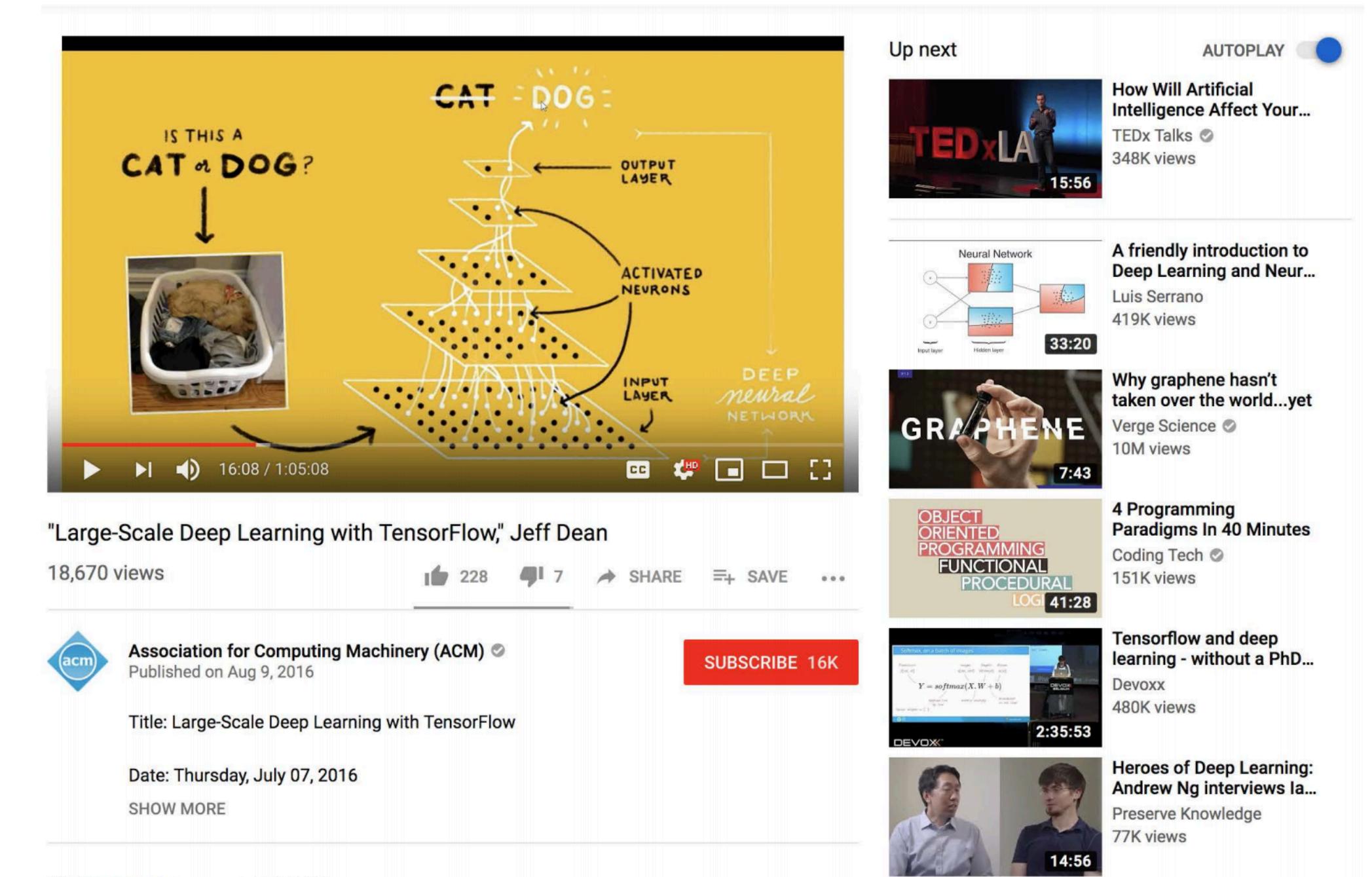


Figure 4: Recommending what to watch next on YouTube.

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

- videos that users will rate highly

Conflicting objectives:

- videos that users they will share
- videos that user will watch

implicit bias caused by *feedback*:

user may have watched it because it was recommended!

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

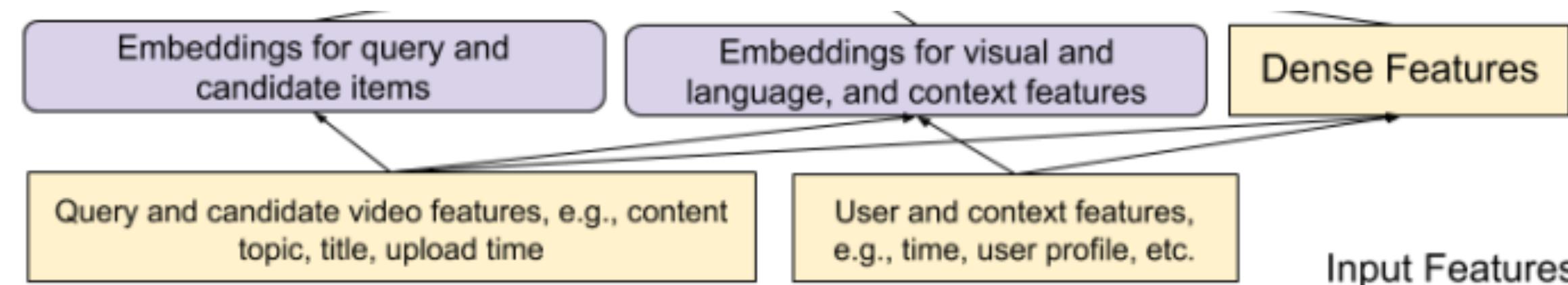
Candidate videos: pool videos from multiple candidate generation algorithms

- 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 for tasks related to **time spent**

Satisfaction:

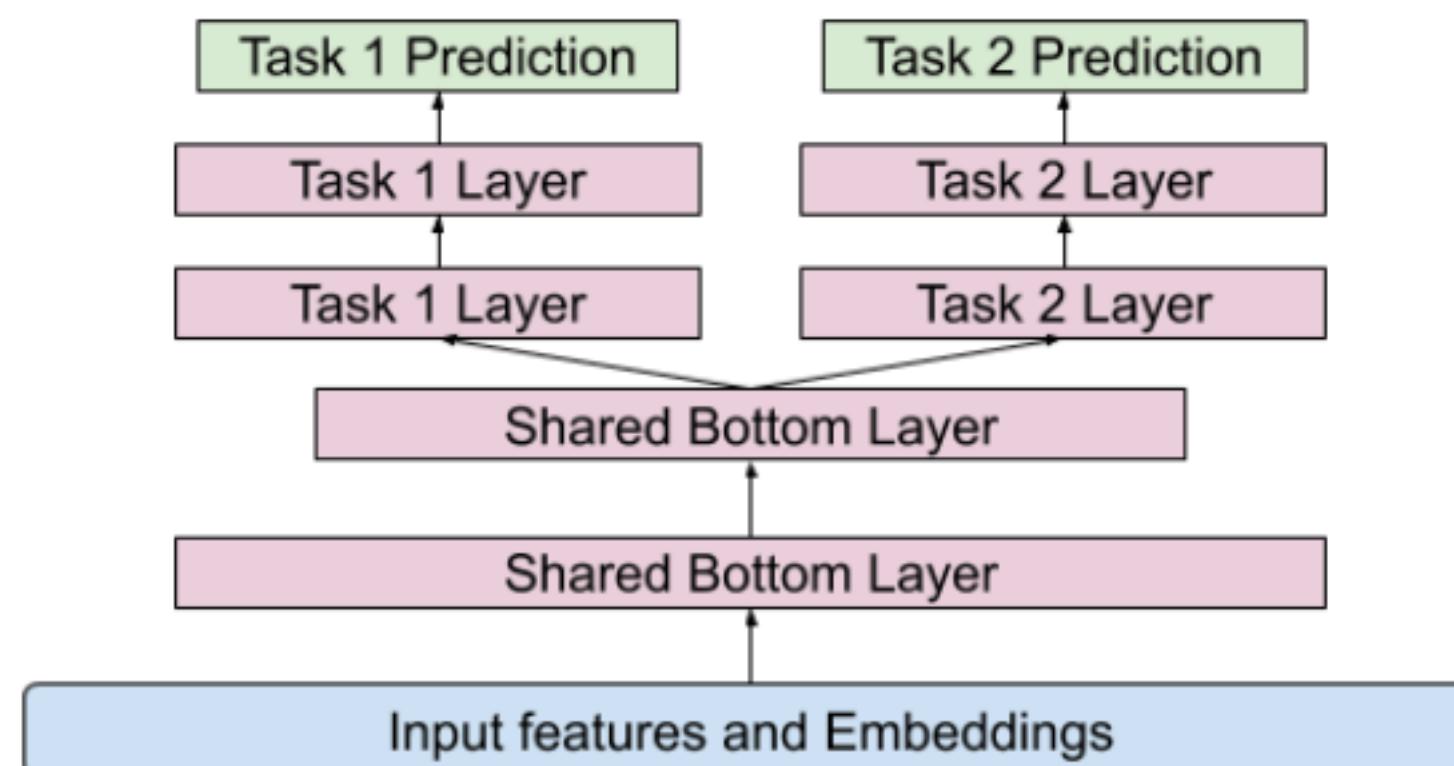
- binary classification tasks like **clicking “like”**
- regression tasks for tasks such as **rating**

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.

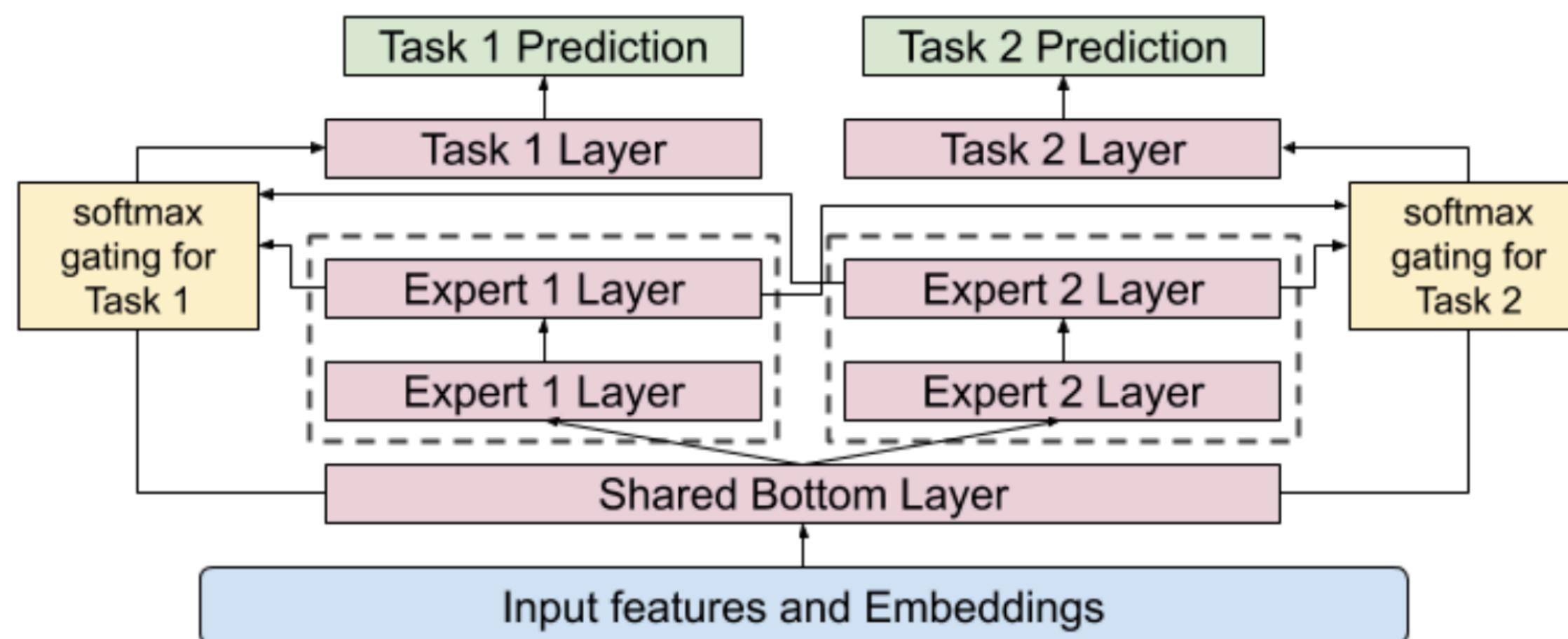
-> harm learning when correlation between tasks is low

The Architecture

Instead: use a form of soft-parameter sharing

"Multi-gate Mixture-of-Experts (MMoE)"

Allow different parts of the network to "specialize" expert neural networks $f_i(x)$



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

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

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

Compute features from selected expert:

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

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

Experiments

Set-Up

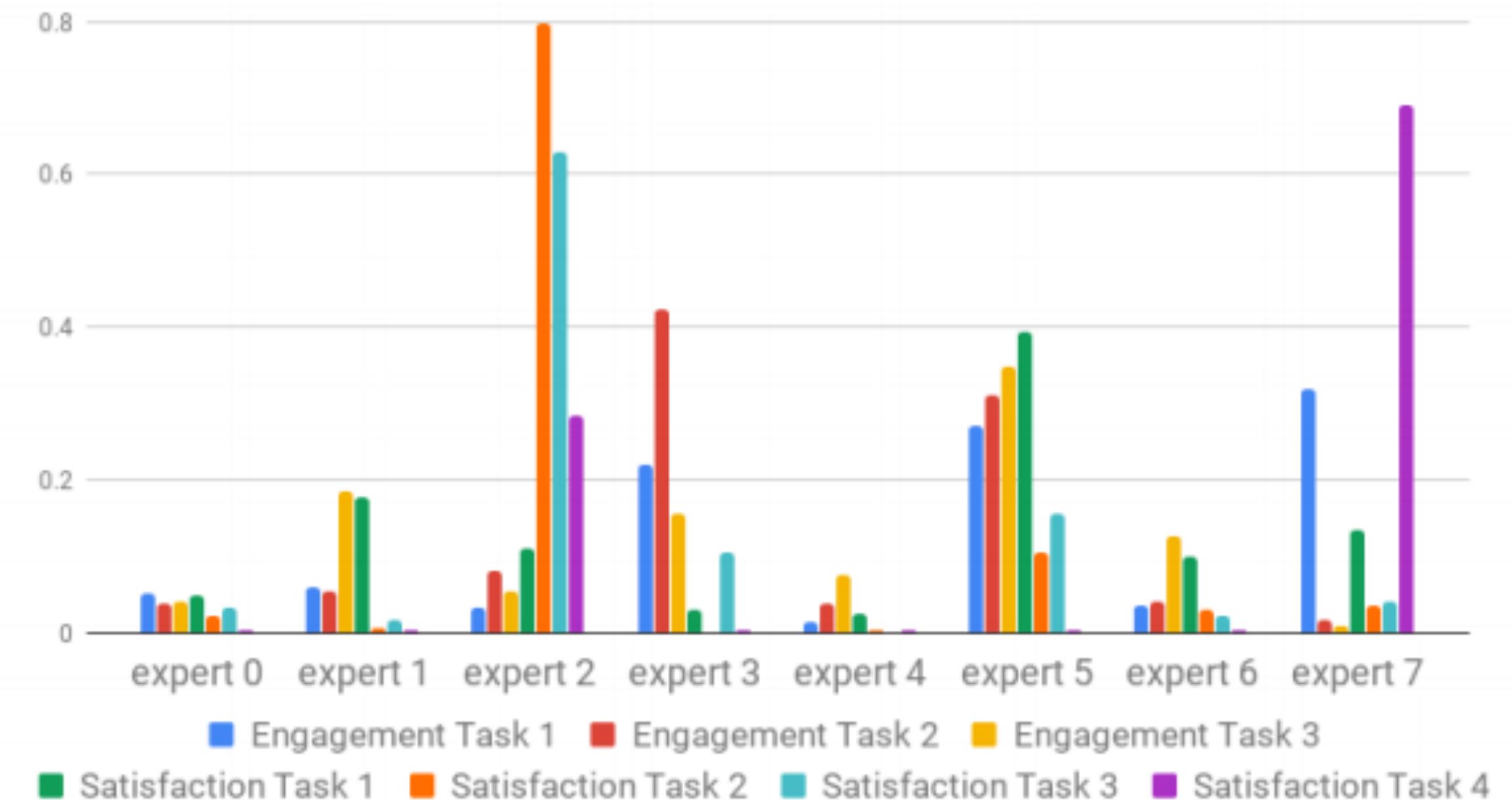
- Implementation in TensorFlow, TPUs
- Train in *temporal order*, running training continuously to consume newly arriving data
- **Offline** AUC & squared error metrics
- **Online A/B testing** in comparison to production system
 - live metrics based on time spent, survey responses, rate of dismissals
- 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.

Expert Utilization for Multiple Tasks



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

Plan for Today

Multi-Task Learning

- Problem statement
- Models & training
- Challenges
- Case study of real-world multi-task learning

Transfer Learning

- Pre-training & fine-tuning

Multi-Task Learning vs. Transfer Learning

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \dots, \mathcal{T}_T$ at once.

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

Transfer learning is a valid solution to multi-task learning.
(but not vice versa)

Transfer Learning

Solve target task \mathcal{T}_b after solving source task \mathcal{T}_a
by *transferring* knowledge learned from \mathcal{T}_a

Key assumption: Cannot access data \mathcal{D}_a during transfer.

Side note: \mathcal{T}_a may include
multiple tasks itself.

Question: In what settings might transfer learning make sense?
(answer in chat or raise hand)

Transfer learning via fine-tuning

$$\phi \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

(typically for many gradient steps)

pre-trained parameters

training data
for new task

Pre-trained Dataset	PASCAL	SUN
Original	58.3	52.2
Random	41.3 [21]	35.7 [2]

What makes ImageNet good for transfer learning? Huh, Agrawal, Efros. '16

Where do you get the pre-trained parameters?

- ImageNet classification
- Models trained on large language corpora (BERT, LMs)
- Other unsupervised learning techniques
- Whatever large, diverse dataset you might have

Pre-trained models often available online.

Some common practices

- Fine-tune with a smaller learning rate
- Smaller learning rate for earlier layers
- Freeze earlier layers, gradually unfreeze
- Reinitialize last layer
- Search over hyperparameters via cross-val
- Architecture choices matter (e.g. ResNets)

Universal Language Model Fine-Tuning for Text Classification. Howard, Ruder. '18

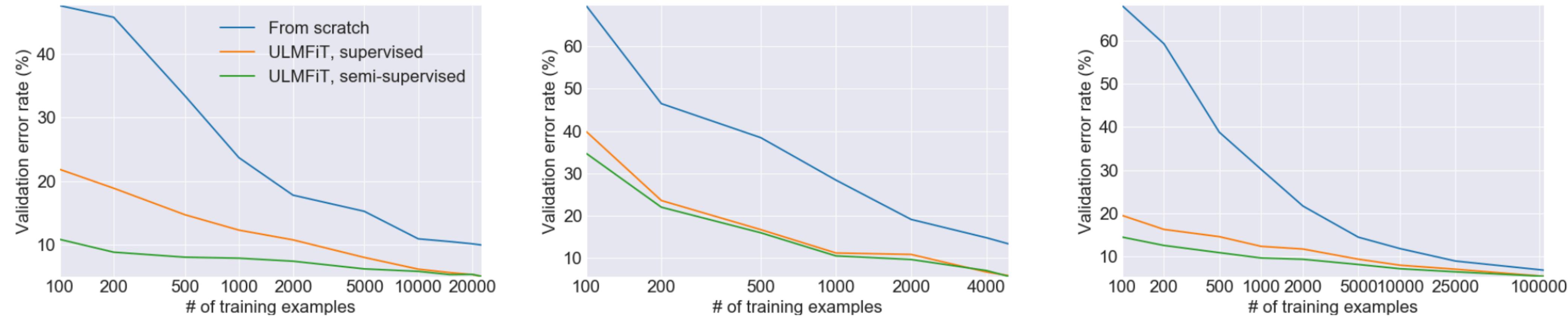


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Fine-tuning doesn't work well with small target task datasets

Upcoming lectures: few-shot learning via meta-learning

Plan for Today

Multi-Task Learning

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

Transfer Learning

- Pre-training & fine-tuning

Goals for by the end of lecture:

- Know the **key design decisions** when building multi-task learning systems
- Understand the **difference** between multi-task learning and **transfer learning**
- Understand the **basics** of **transfer learning**

Reminders

Next time: Meta-learning problem statement, Black-box meta-learning, GPT-3