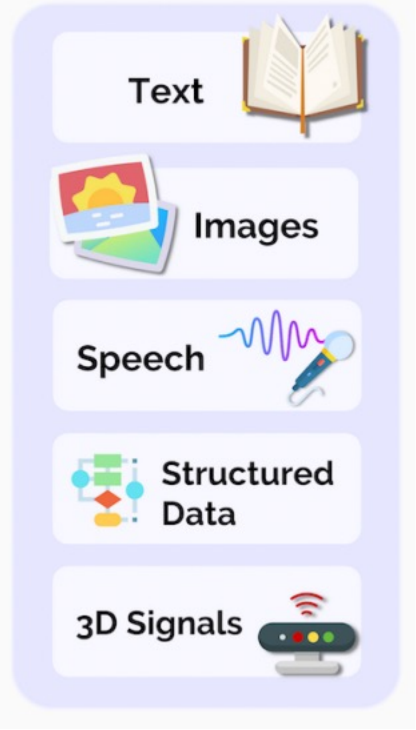
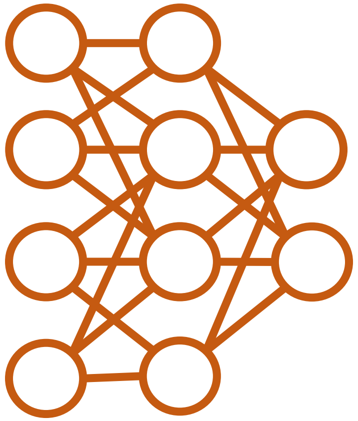
How to use pre-trained models?

Aditi Raghunathan

A new era in ML



Step one:
pretraining



Step two:
adaptation

Diverse (typically unlabeled) data

Pre-trained model

Specialize to narrow distribution

Bommasani et al. 2021

A new era in ML

- Spearheaded in NLP (with the widespread success of BERT)
- Now finding it’s way into other applications as well
- Heard of these?
    - CLIP [Radford et al. 2021]

Why pre-training?

Convenience of few-shot learning: Do not need to collect lot of training data for each new task

Improved downstream performance: Effectively incorporate useful information from a lot of data

Especially true for robustness when test distribution is different from training distribution you collected labeled data from

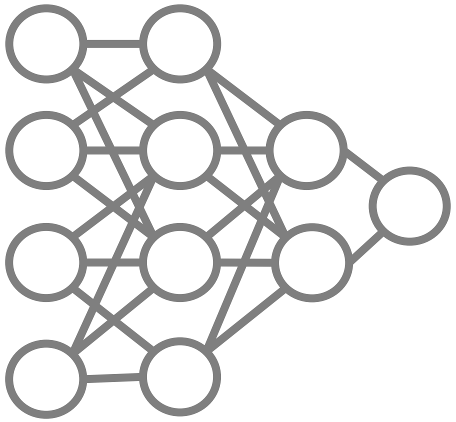
One of the most reliable methods to improve robustness across several natural shifts

Why pre-training?

Consider satellite remote sensing task

We have training data from **North America**, but very limited data from **Africa**

Standard supervised learning



"In-distribution" training data

North America

Performs poorly on **OOD test data** from Africa

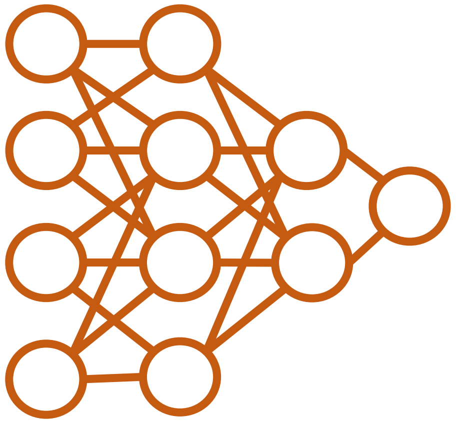
Why pre-training?

Consider satellite remote sensing task

We have training data from **North America**, but very limited data from **Africa**

Transfer learning setting

Pre-trained model



"In-distribution" training data

North America

Performs better on **OOD test data** from Africa

How to use pre-trained models?

Too many, difficult to select?

Promptless Fine-tuning

Fixed-prompt Tuning

Prompt+LM Fine-tuning

Tuning-free Prompting

Fixed-LM Prompt Tuning

If you have a huge pre-trained language model (e.g., GPT3)

If you have few training samples?

If you have lots of training samples?

Talk outline: part one

Too many, difficult to select?

Promptless Fine-tuning

Fixed-prompt Tuning

Prompt+LM Fine-tuning

Tuning-free Prompting

Fixed-LM Prompt Tuning

If you have a huge pre-trained language model (e.g., GPT3)

If you have few training samples?

If you have lots of training samples?

Talk outline: part two

Too many, difficult to select?

Promptless Fine-tuning

Fixed-prompt Tuning

Prompt+LM Fine-tuning

Tuning-free Prompting

Fixed-LM Prompt Tuning

If you have a huge pre-trained language model (e.g., GPT3)

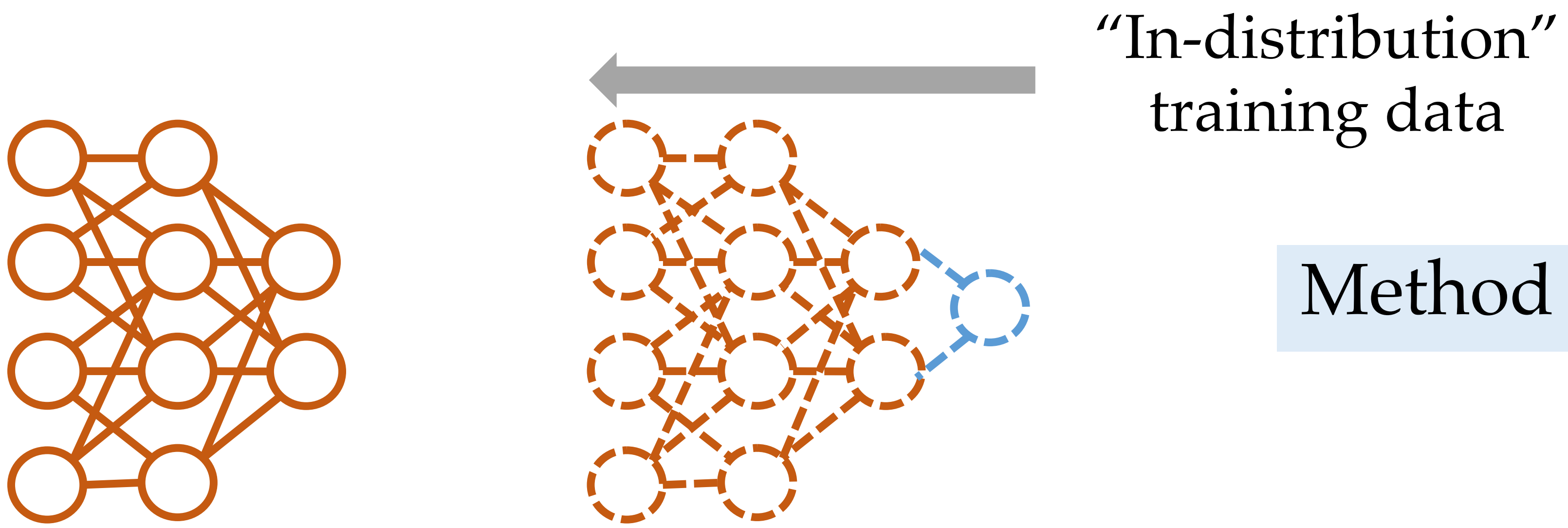
If you have few training samples?

If you have lots of training samples?

How to fine-tune pretrained models?

How to use pre-trained models?

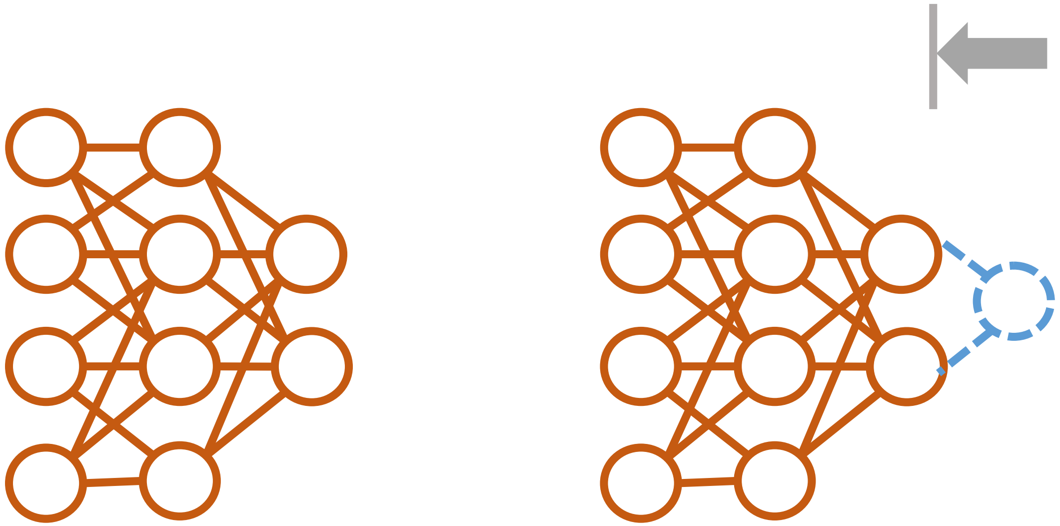
How to leverage the diverse information contained in pre-trained models?



Method one: Fine-tuning

How to use pre-trained models?

How to leverage the diverse information contained in pre-trained models?

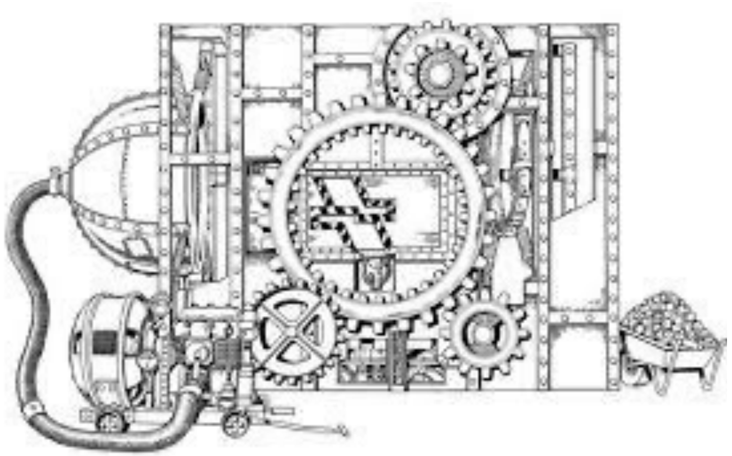


“In-distribution” training data

Method two: Linear probing

Understanding transfer learning

Several moving pieces



Model architecture

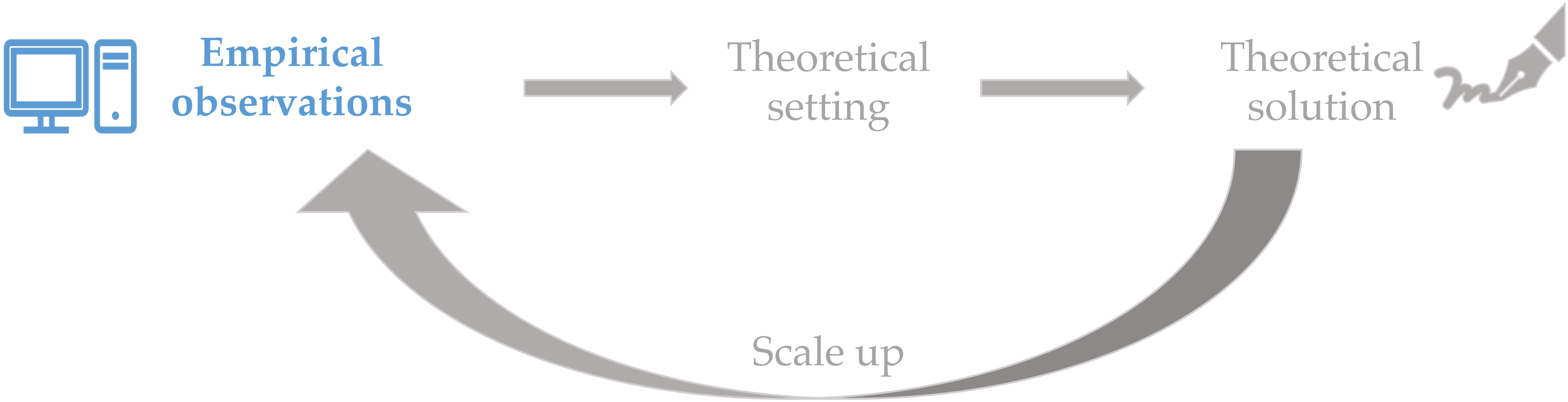
Pre-training distribution

Pre-training procedure

Adaptation distribution

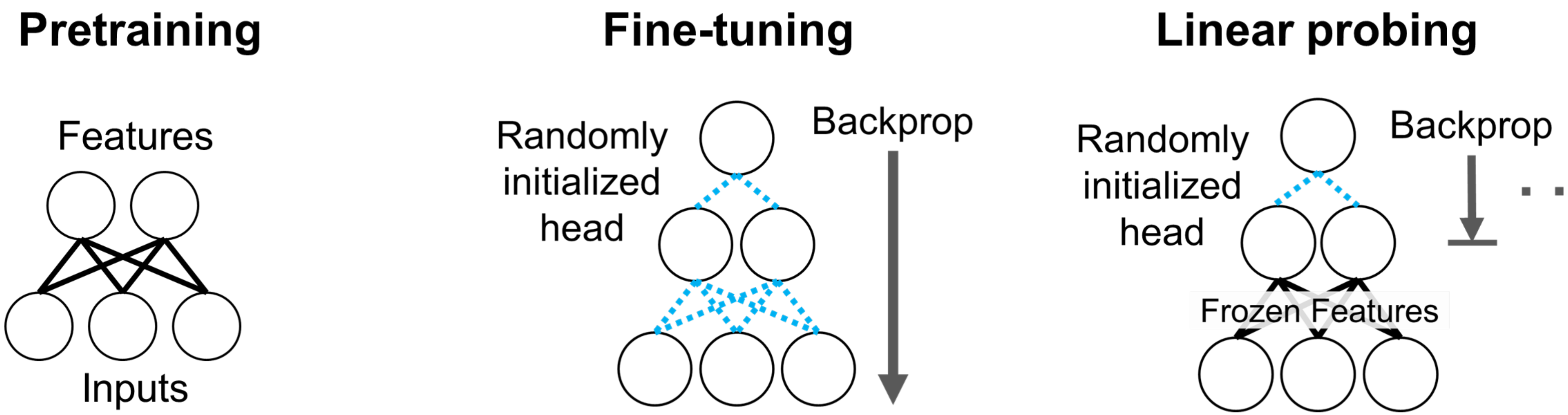
Adaptation procedure

Talk outline: format



Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Linear probing vs fine-tuning



Pop quiz!

Dataset: BREEDS Living-17

Task: classify into animal categories

Train distribution: one subset of ImageNet hierarchy tree with animal category as root

Test distribution: other subset of ImageNet hierarchy tree with animal category as root

Pretrained model: MoCo-V2, which has seen *unlabeled* ImageNet images (including various types of animals)

Train

Test

Santurkar et al. 2020

Pop quiz: living-17

Living-17	ID	OOD
Scratch	92.4%	58.2%
Linear probing	96.5%	?
Fine-tuning	97.1%	

Does linear probing do better than scratch OOD?

Pop quiz: living-17

Living-17	ID	OOD
Scratch	92.4%	58.2%
Linear probing	96.5%	82.2%
Fine-tuning	97.1%	

Does linear probing do better than scratch OOD?

Yes!

Pop quiz: living-17

Living-17	ID	OOD
Scratch	92.4%	58.2%
Linear probing	96.5%	82.2%
Fine-tuning	97.1%	?

Does fine-tuning do better than linear probing OOD?

Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Pop quiz: living-17

Living-17	ID	OOD
Scratch	92.4%	58.2%
Linear probing	96.5%	82.2%
Fine-tuning	97.1%	77.7%

Does linear probing do better than fine-tuning OOD?

No!

Dataset: CIFAR 10.1

Task: classify into CIFAR-10 categories

Train distribution: original CIFAR-10 dataset

Test distribution: recent near-replication of the pipeline

Pretrained model: MoCo-V2, which has seen *unlabeled* ImageNet images

Recht et al. 2019

Pop quiz: CIFAR10.1

Living-17	ID	OOD
Linear probing	91.8%	82.7
Fine-tuning	97.3%	?

Does linear probing do better than fine-tuning OOD?

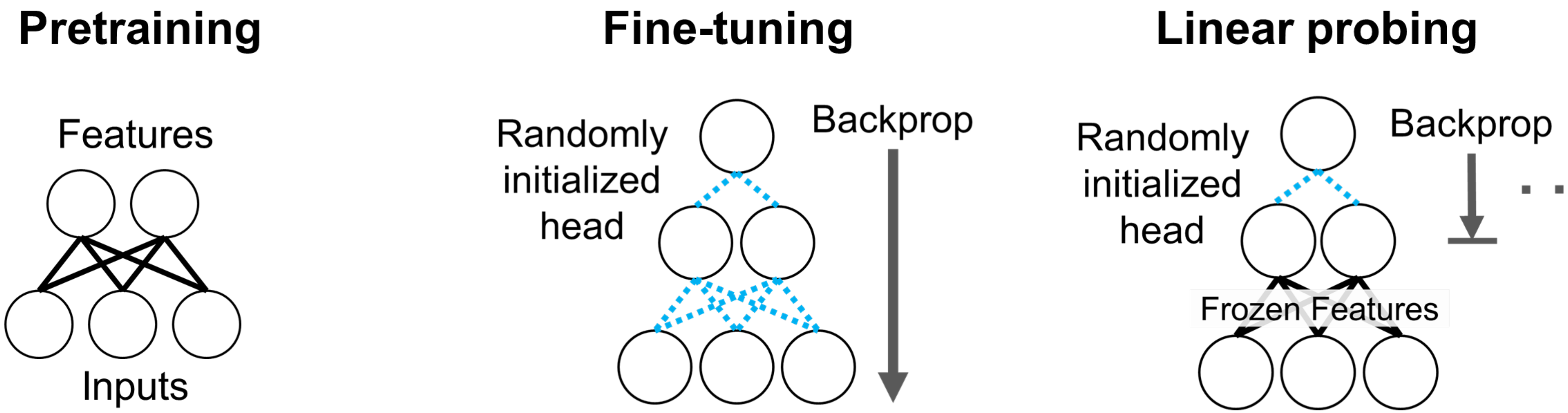
Pop quiz: CIFAR10.1

Living-17	ID	OOD
Linear probing	91.8%	82.7
Fine-tuning	97.3%	92.3%

Does linear probing do better than fine-tuning OOD?

No!

Linear probing vs fine-tuning summary



Which method does better?

Linear probing vs fine-tuning summary

	ID	OOD
Linear probing	82.9%	
Fine-tuning	85.1%	

Averaged over 10 datasets

Common wisdom is fine-tuning works better than linear probing

Linear probing vs fine-tuning summary

	ID	OOD
Linear probing	82.9%	66.2%
Fine-tuning	85.1%	59.3%

Averaged over 10 datasets

LP performs better than FT OOD on 8 out of 10 datasets

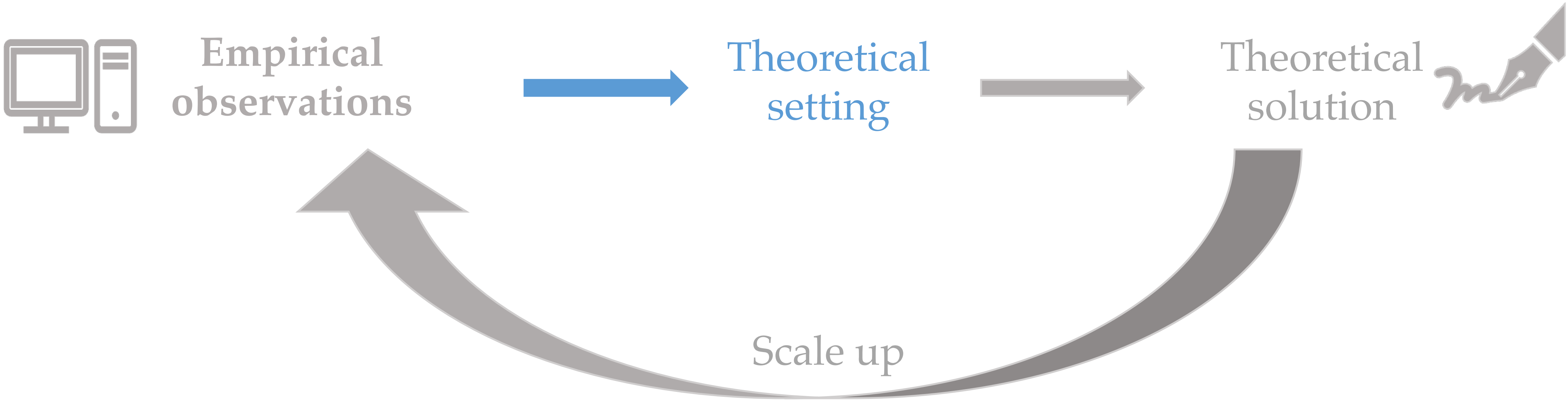
Linear probing vs fine-tuning summary

- Common wisdom is fine-tuning works better than linear probing
- Linear probing can often perform better out-of-distribution
  - Especially with **high quality** pre-trained features and **large** distribution shifts

*There is probably a lot we can do to improve **downstream** methods…*

Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Talk outline: format

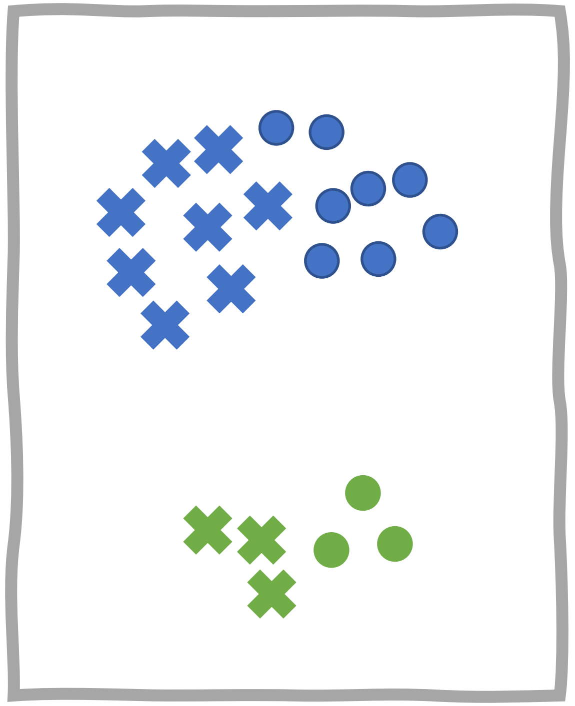


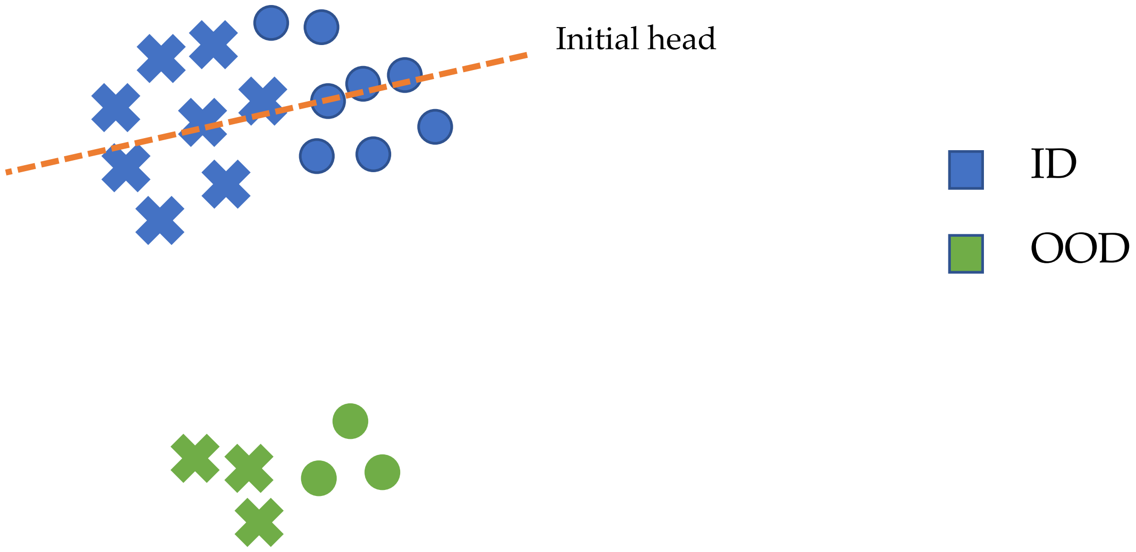
Theoretical analyses

- Prior transfer learning theory mostly looks at only linear probing which is convex (Wu et al. 2020, Tripuraneni et al. 2020, Du et al. 2020, Xie et al. 2020)
- We want to analyze the **non-convex** objective of fine-tuning
- Same objective as training from scratch but **different training dynamics** stemming from pre-trained initialization
- Cannot assume random initialization and associated simplifications

Intuition for theoretical result

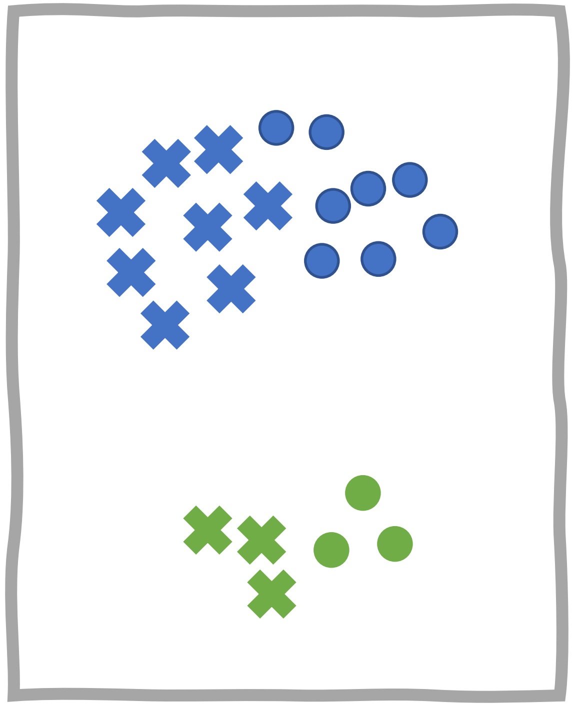
Pretrained
Features



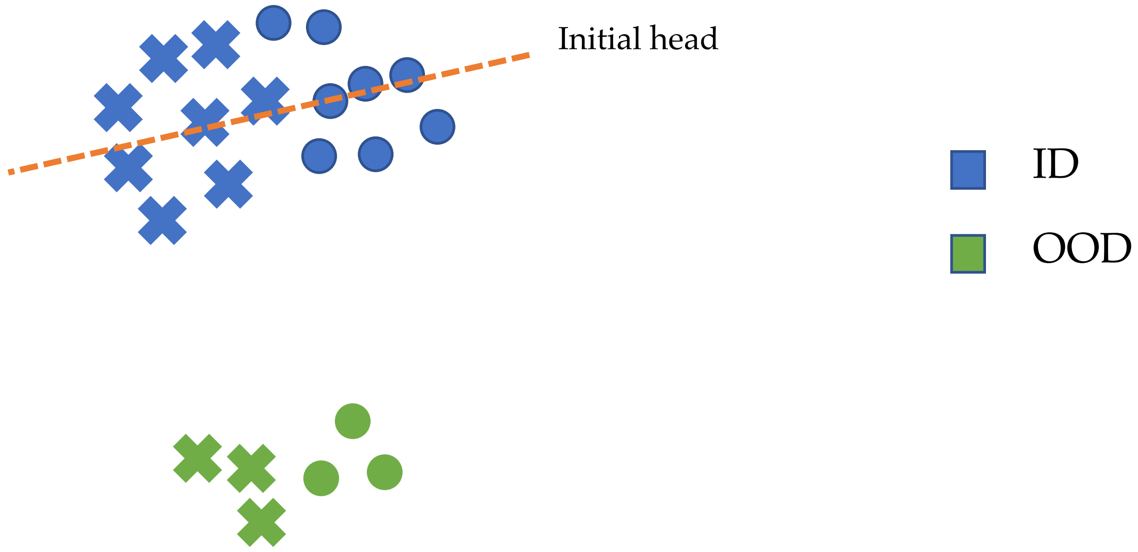


Intuition for theoretical result

Pretrained
Features



Fine-tuning: features for ID examples change in sync with the linear head

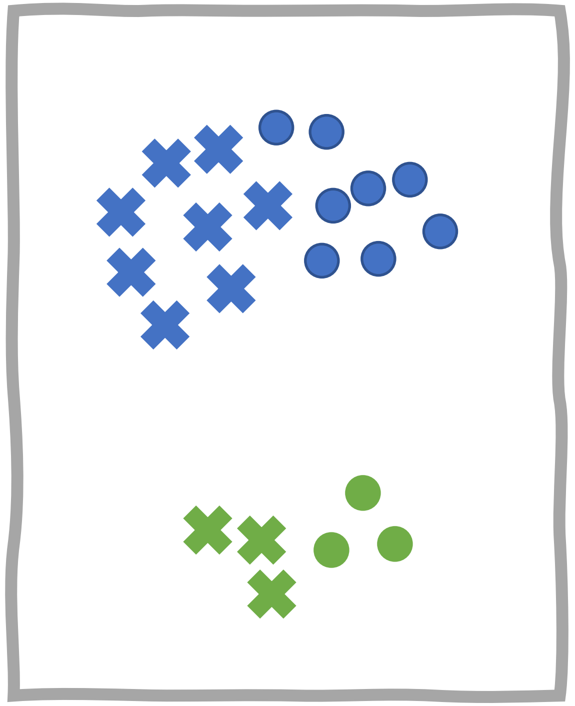


Features for OOD examples change less

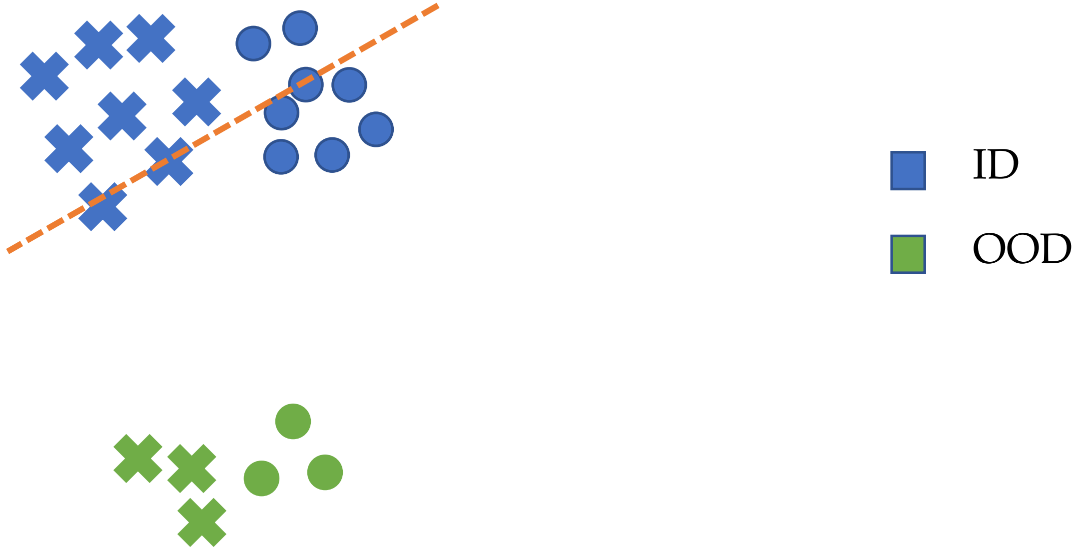
Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Intuition for theoretical result

Pretrained Features



Fine-tuning: features for ID examples change in sync with the linear head

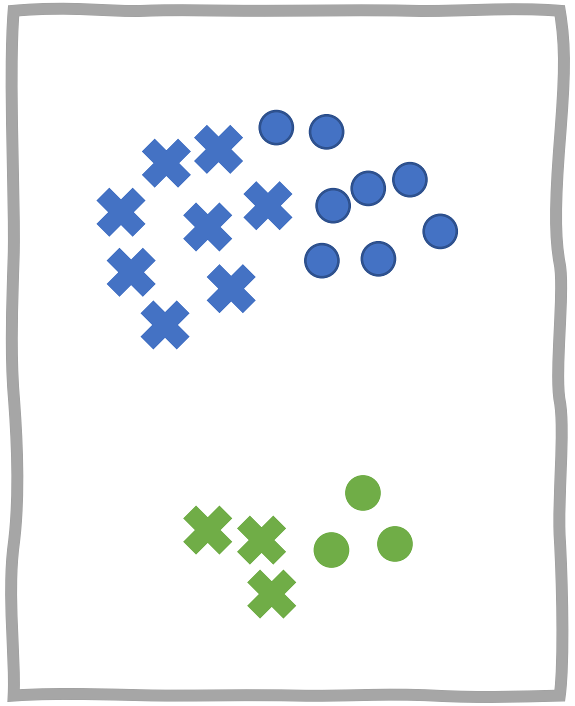


Features for OOD examples change less

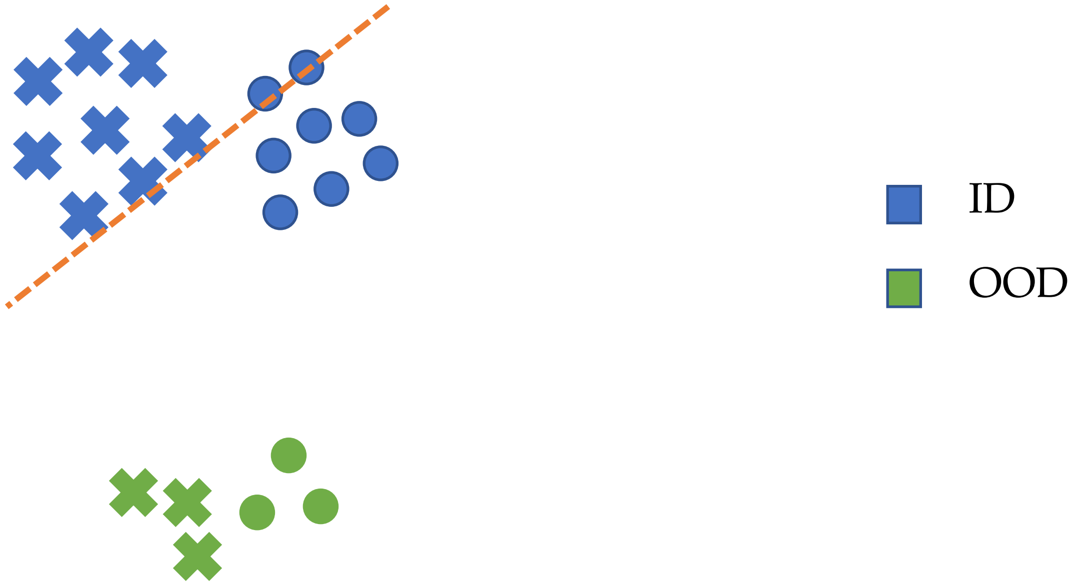
Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Intuition for theoretical result

Pretrained
Features



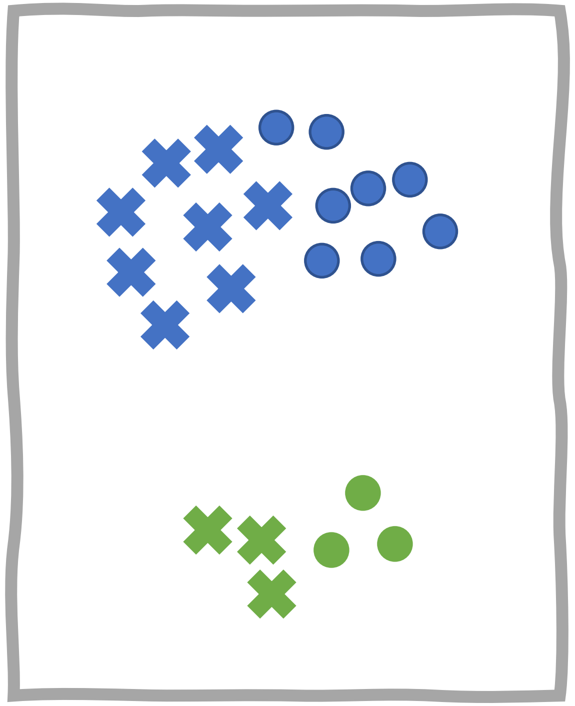
Fine-tuning: features for ID examples change in sync with the linear head



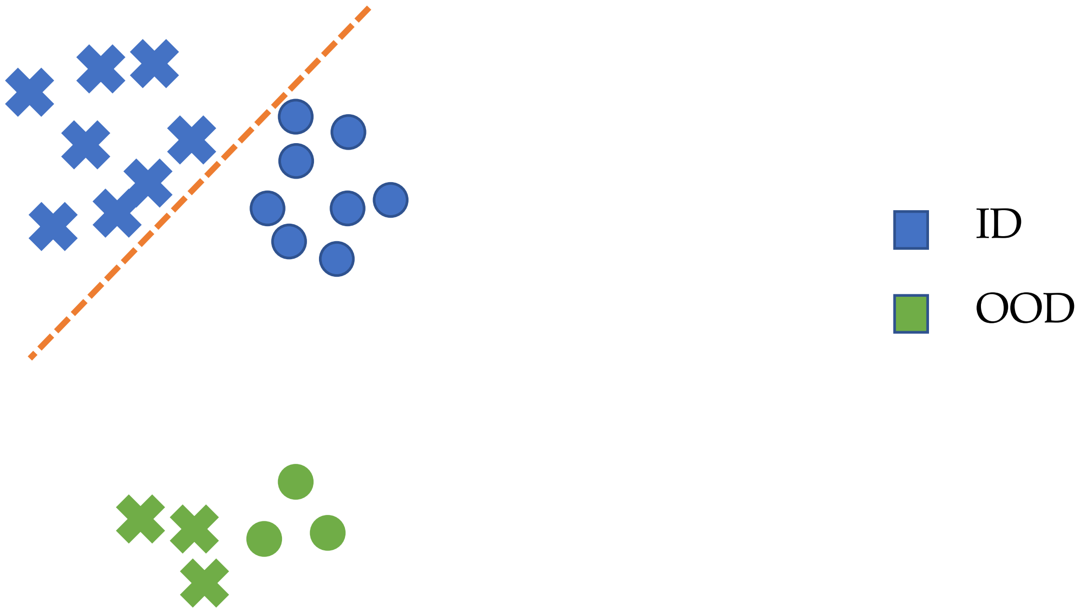
Features for OOD examples change less

Intuition for theoretical result

Pretrained Features



Fine-tuning: features for ID examples change in sync with the linear head

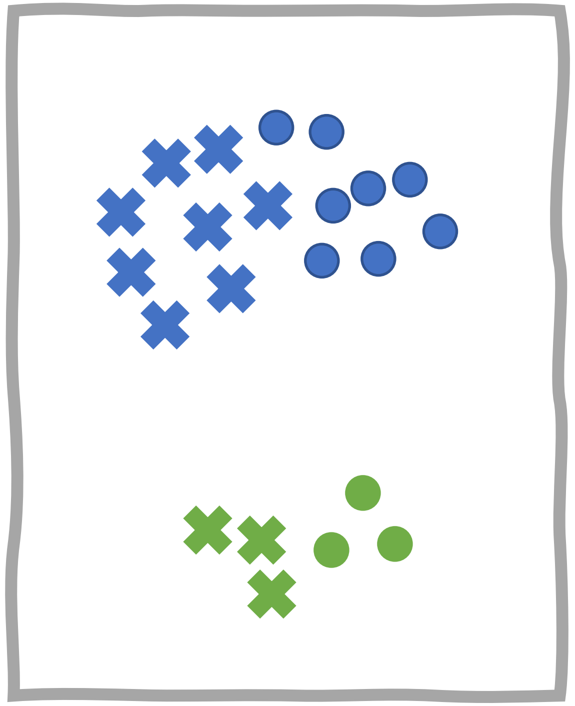


Features for OOD examples change less

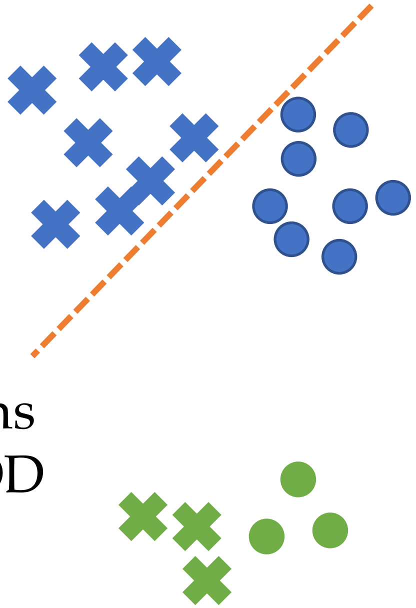
Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Intuition for theoretical result

Pretrained Features

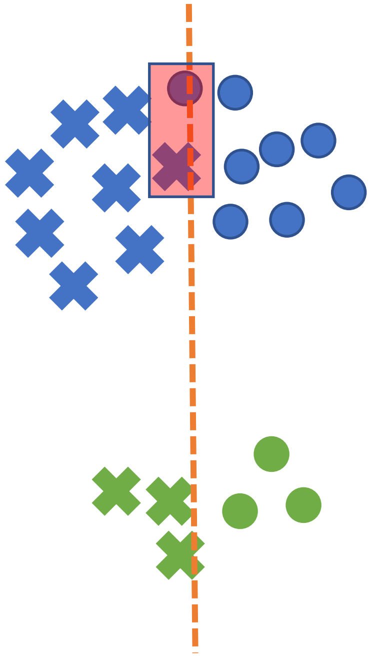


Fine-tuning: features for ID examples change in sync with the linear head



Head performs poorly on OOD examples

Linear probing: freezes pretrained features



Head is decent on OOD examples

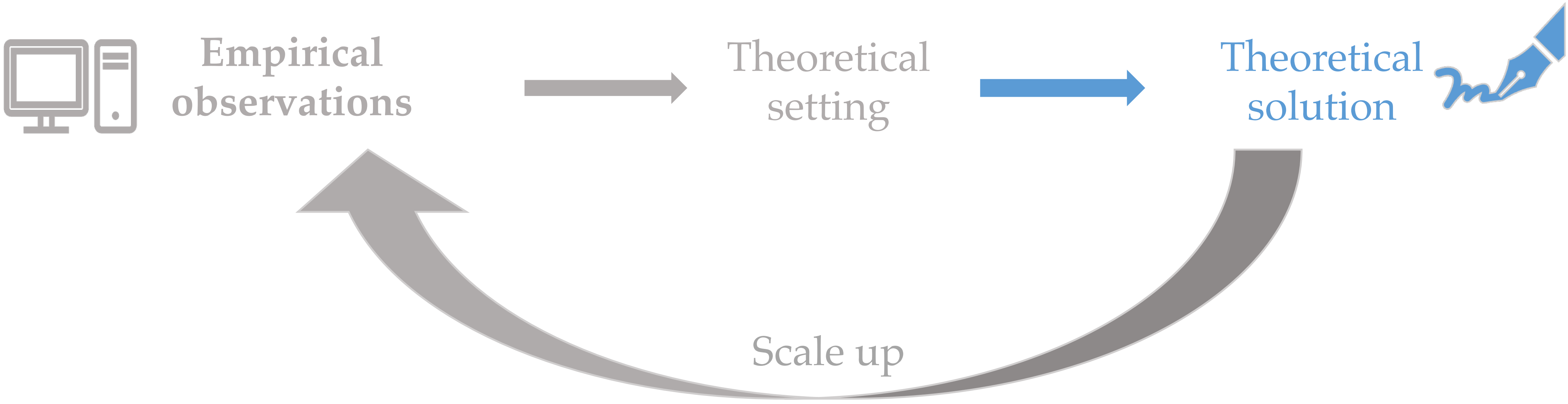
Fine-tuning can lead to feature distortion

Theorem (informal)

Under simplifying assumptions (two-layer linear networks, squared error, OOD data in orthogonal subspace to ID training data),

$$\forall t, \frac{L_{\text{ood}}\left(\theta_{\text{lp}}(t)\right)}{L_{\text{ood}}\left(\theta_{\text{ft}}(t)\right)} \xrightarrow{p} 0, \quad \text{as pretrained features} \rightarrow \text{optimal}$$

Talk outline: format



Best of both worlds

Why does FT do better ID?

Training data may not be linearly separable in the space of pre-trained features i.e. imperfect pre-trained features

Why does FT do worse OOD?

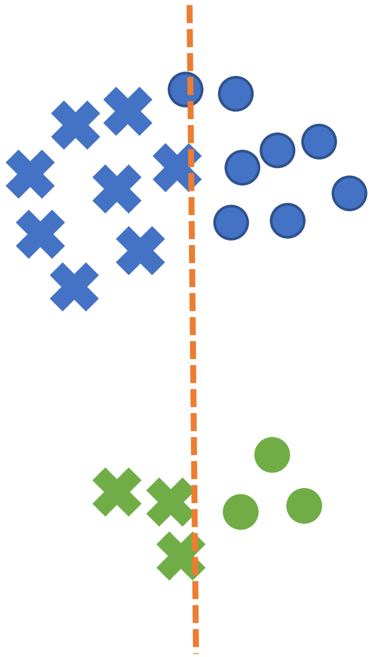
Features can change a lot to accommodate a randomly initialized head

Can we refine features without distorting them too much?

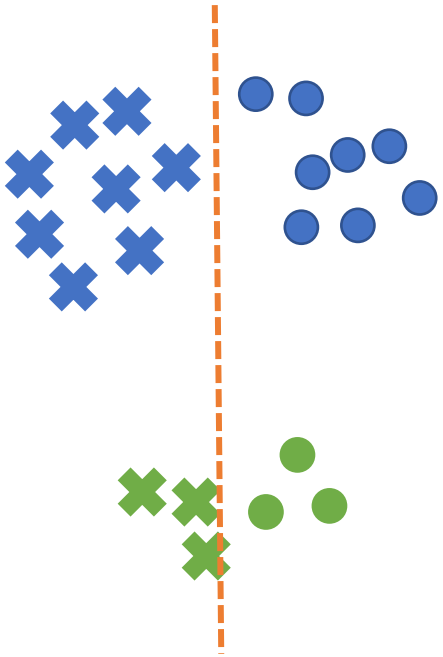
Method to achieve best of both worlds

Idea: modify pre-trained features **only as necessary**

Step 1: Linear probe



Step 2: Fine-tune



Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Method to achieve best of both worlds

Idea: modify pre-trained features **only as necessary**

Step 1: Linear probe

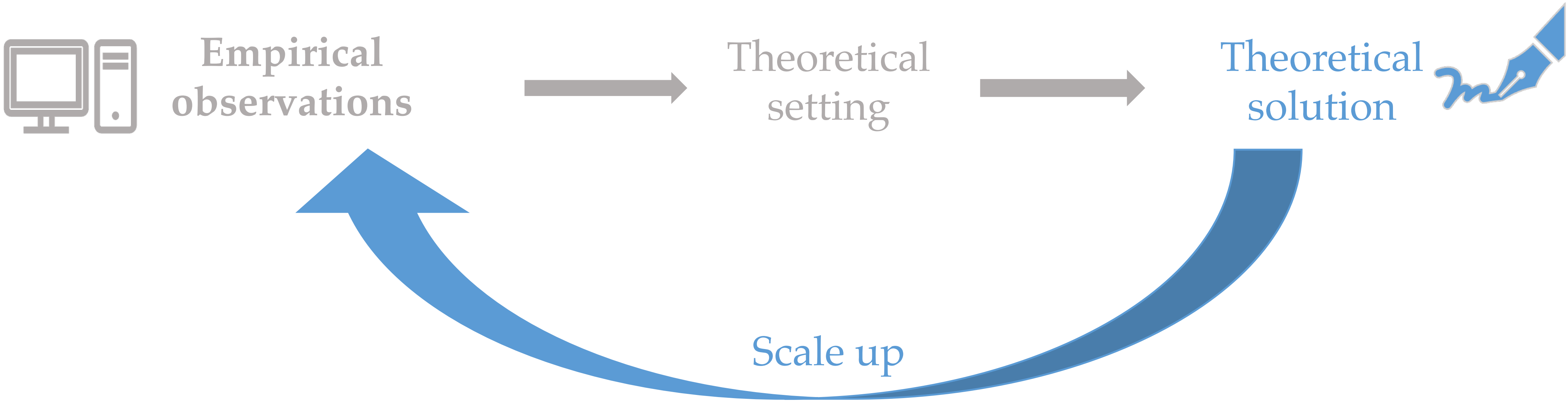
Step 2: Fine-tune

LP-FT method

Can prove that LP-FT dominates both LP and FT under the simple setting of perfect features

Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Talk outline: format



Improving fine-tuning

	ID	OOD
Linear probing	82.9%	66.2%
Fine-tuning	85.1%	59.3%
LP-FT	**85.7%**	**68.9%**

+10% over fine-tuning!

LP-FT obtains better than the best of both worlds

In-Distribution Accuracies

	CIFAR-10	Ent-30	Liv-17	DomainNet	FMoW	ImageNet	Average
FT	**97.3 (0.2)**	**93.6 (0.2)**	97.1 (0.2)	84.5 (0.6)	**56.5 (0.3)**	**81.7 (-)**	85.1
LP	91.8 (0.0)	90.6 (0.2)	96.5 (0.2)	89.4 (0.1)	49.1 (0.0)	79.7 (-)	82.9
LP-FT	**97.5 (0.1)**	**93.7 (0.1)**	**97.8 (0.2)**	**91.6 (0.0)**	51.8 (0.2)	**81.7 (-)**	**85.7**

Kumar, Raghunathan, Jones, Ma, and Liang. *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022.

Out-of-Distribution Accuracies

	STL	CIFAR-10.1	Ent-30	Liv-17	DomainNet	FMoW
FT	82.4 (0.4)	92.3 (0.4)	60.7 (0.2)	77.8 (0.7)	55.5 (2.2)	32.0 (3.5)
LP	85.1 (0.2)	82.7 (0.2)	**63.2 (1.3)**	82.2 (0.2)	79.7 (0.6)	**36.6 (0.0)**
LP-FT	**90.7 (0.3)**	**93.5 (0.1)**	**62.3 (0.9)**	**82.6 (0.3)**	**80.7 (0.9)**	**36.8 (1.3)**

	ImNetV2	ImNet-R	ImNet-Sk	ImNet-A	Average
FT	**71.5 (-)**	52.4 (-)	40.5 (-)	27.8 (-)	59.3
LP	69.7 (-)	70.6 (-)	46.4 (-)	45.7 (-)	66.2
LP-FT	**71.6 (-)**	**72.9 (-)**	**48.4 (-)**	**49.1 (-)**	**68.9**

Experimental investigation

- ID features change much more than OOD features (l_2 distance) when doing vanilla fine-tuning
- ID features change an order of magnitude less when doing LP-FT rather than vanilla fine-tuning (same training loss)

Discussion

- Pretrained models give large improvements in accuracy, but **how we fine-tune them is key**
- LP-FT is just a starting point and one example
- More broadly, light-weight fine-tuning (in NLP) improves robustness
  - Adapter modules [Houlsby et al. 2019], prefix tuning [Li and Liang, 2021]
  - See similar tradeoffs i.e. drop in in-distribution performance

Can we skip fine-tuning entirely?
(in-context learning)

Large language models (LMs)

- Large LMs are trained to predict the next token given previous tokens on internet-scale text datasets

Albert Einstein was a German theoretical $\xrightarrow{\text{predict}}$ physicist

If we can predict next token well, can we solve all tasks?

The need for “learning”

Albert Einstein was $\xrightarrow{\text{predict}}$ A physicist

$\xrightarrow{\text{predict}}$ German

$\xrightarrow{\text{predict}}$ A genius

Need to specify the task in the prompt

“Prompt engineering”

Can we use data to so the same?

“In-context” learning

- Just present the training data directly in the prompt
  - No parameters are optimized

Concatenate independent examples

Marie Curie was Polish \n Mahatma Gandhi was Indian \n Albert Einstein was → German

Gets SOTA on LAMBADA (completion), TriviaQA (question answering), etc.

[Brown et al. 2020]

Why is this possible?

Mismatch with pretraining

- LM is not explicitly trained to do learning
- Prompts not formatted like natural language (e.g., concatenate independent examples).

Pretraining documents

Albert Einstein was a German theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is best known for developing the theory of relativity, but he also ….

Mismatch

In-context learning prompt

Albert Einstein was German \n Mahatma Gandhi was Indian \n Marie Curie was

How does in-context learning work?

Hard to answer because

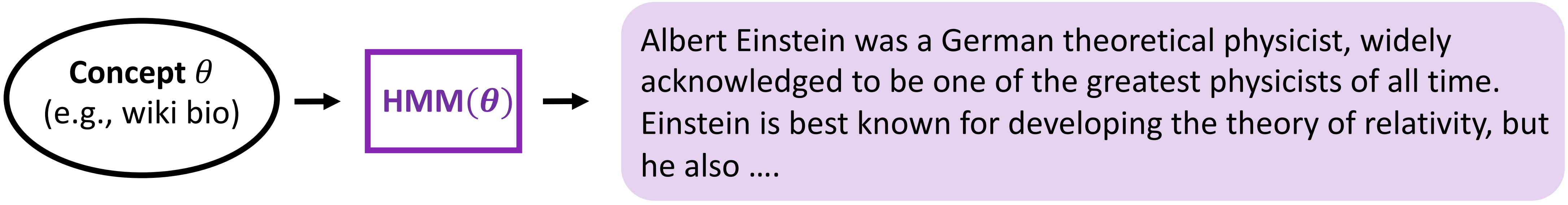
- Real pretraining data is messy and huge
- The models are huge (170B params)

Our goals

1. First step for understanding in-context learning with simple framework
2. Provide small-scale dataset as a testbed for in-context learning
3. Use insights to figure out how to better design prompts?

Mental model of pretraining distribution

- There is a latent concept θ
- Conditioned on θ, data is generated via a Hidden Markov Model
- Documents are generated as follows:
  - Sample θ
  - Sample text from HMM(θ)



Importance of latent concept

- All sentences in a document share a concept (long-term coherence)
- To predict coherent next words, LM must infer shared concept

Albert Einstein was a German theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is best known for developing the theory of relativity, but he also ....

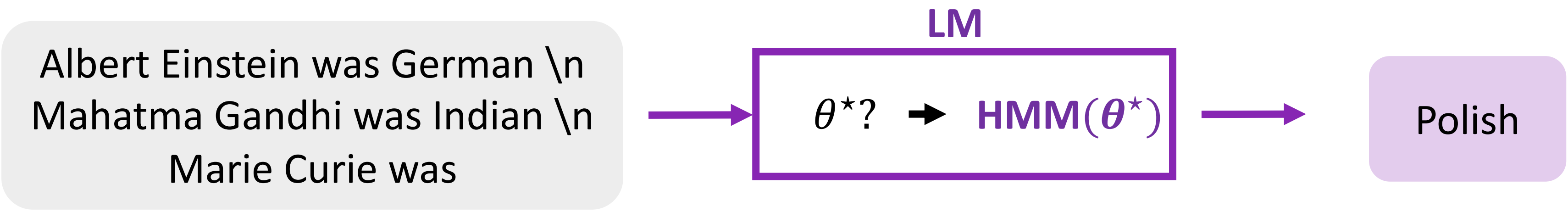
LM

$\theta? \rightarrow$ **HMM**$(\boldsymbol{\theta})$

made important contributions to the development of quantum mechanics.

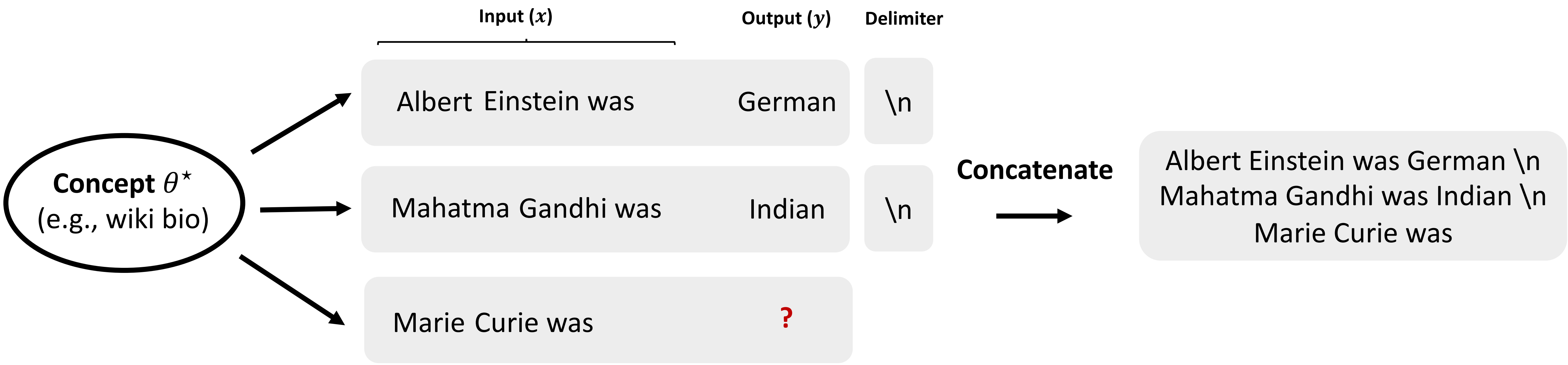
From pretraining to in-context learning

- If the LM also infers the prompt concept from examples (despite distributional mismatch) -> in-context learning emerges
- If the pretraining data is diverse, the LM can infer many different concepts



Prompt distribution

- **Prompt distribution $\boldsymbol{p}_{\mathbf{prompt}}$** with *prompt concept* $\theta^\star$
  - Generate independent examples from HMM($\theta^\star$) and concatenate with delimiters
  - $\boldsymbol{p}_{\mathbf{prompt}}$ can influence distribution of x (e.g., full names)
    - Allows $\boldsymbol{p}_{\mathbf{prompt}}$ to define the task



In-context learning as implicit Bayesian inference

- Assume pretrained LM fits pretraining distribution perfectly
  - Reduces problem to comparing pretrain vs prompt distributions
- Given prompt $\sim p_{prompt}$ (not pretraining distribution p)
- Posterior predictive distribution:

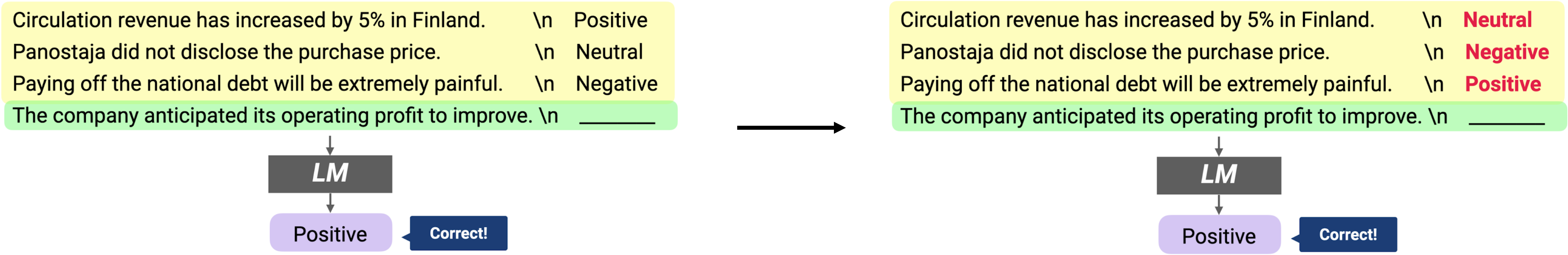
Weight on each concept

$$p(y \mid \text{prompt}) = \int_{\theta} p(y \mid \text{prompt}, \theta)\boxed{p(\theta \mid \text{prompt})}d\theta$$

- Ideal: $p(\theta \mid \text{prompt})$ concentrates on prompt concept $\theta^{\star}$ with more examples

Empirical evidence from NLP benchmarks

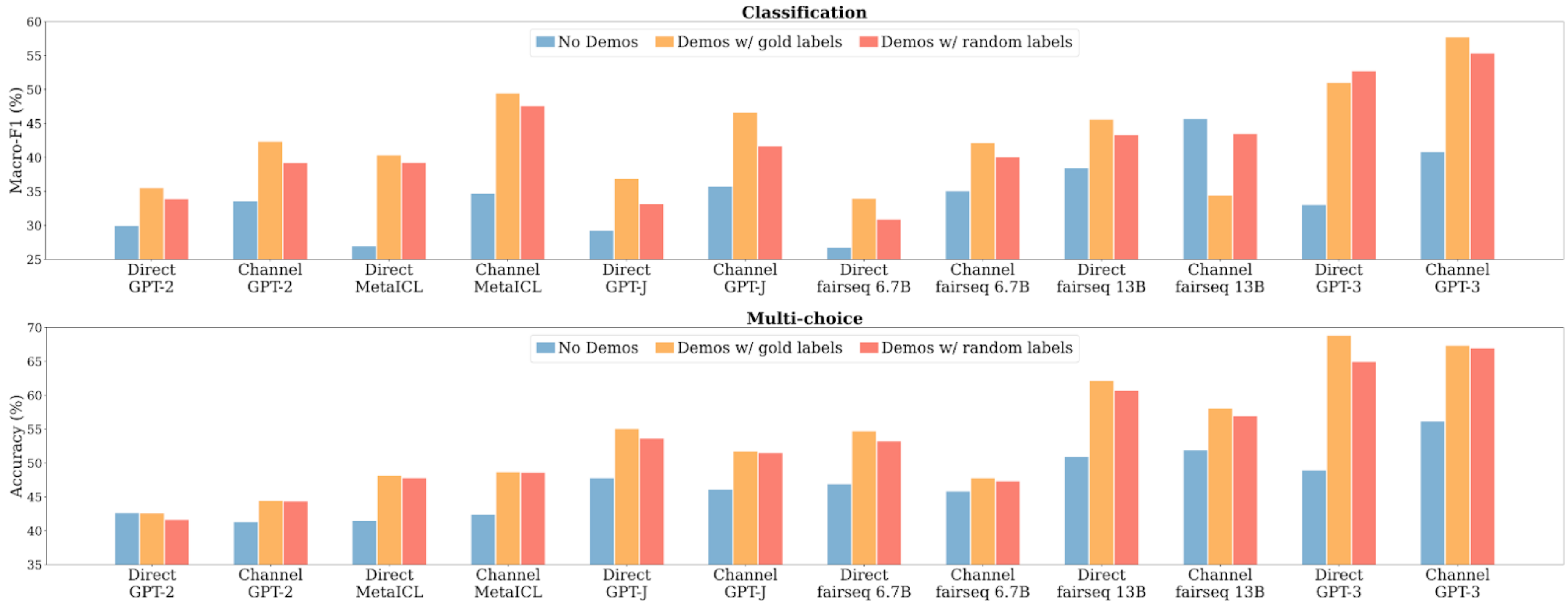
- Experiment (Min et al. 2022): randomize the labels in in-context training examples
- Traditional supervised learning would fail to generalize
- Via in-context learning, the model can still infer $\theta^\star$ as most likely



Figs from Min et al. 2022

Empirical evidence from NLP benchmarks

- Surprisingly, in-context accuracy doesn't drop much with random labels across 26 datasets



Figs from Min et al. 2022

Lessons for prompt design

Delimiters between examples should aid in inferring $\theta^\star$

- Such delimiters can be neutral (equally likely for all concepts)
- Or more likely to be generated by HMM($\theta^\star$) vs HMM($\tilde{\theta}$)

- **Neutral delimiter**: newlines, slashes
- **Bad delimiter**: Confuse the model towards another concept / task. Insert "Birthdate of" before each example
- **Good delimiter**: Insert "Nationality of" before each example

GINC: **g**enerative **in-c**ontext dataset

- We create GINC: a small-scale dataset for studying in-context learning
- Pretrain: 1000 documents, each doc is one long sequence from one HMM, given some θ
- Prompt: 2500 prompts per setting, concatenate independent examples

Pretraining document

```
f / h x ax o a k au ap /
a o u au ae f ao an / ah
u y as a k au j w ax l
aw r ae au g au ap / / u
aj ae d a h x af u aj i
r j w j as y x n i ap
```

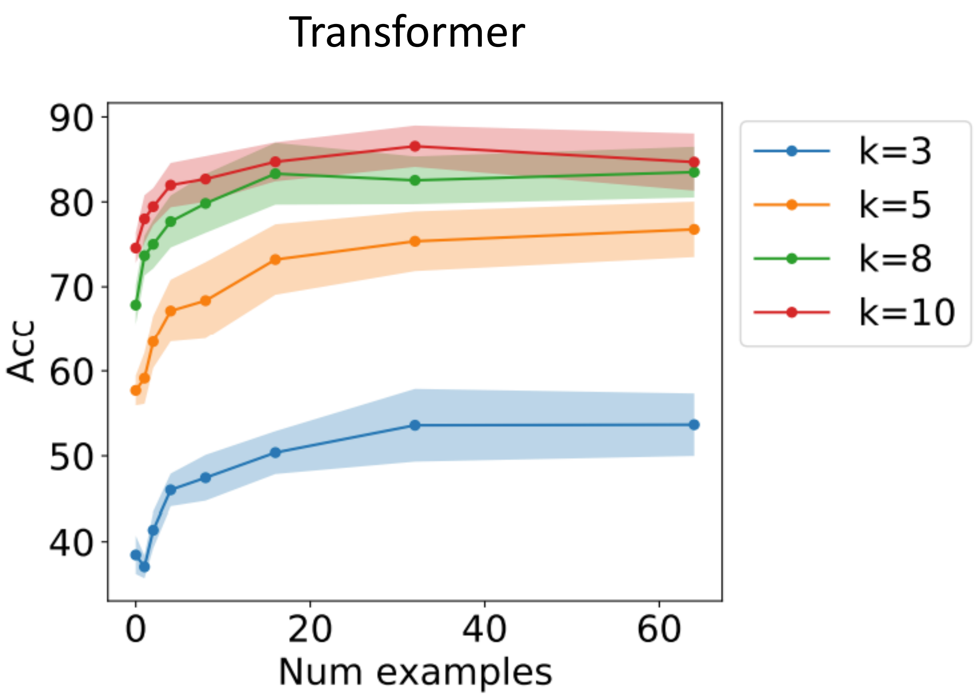
...

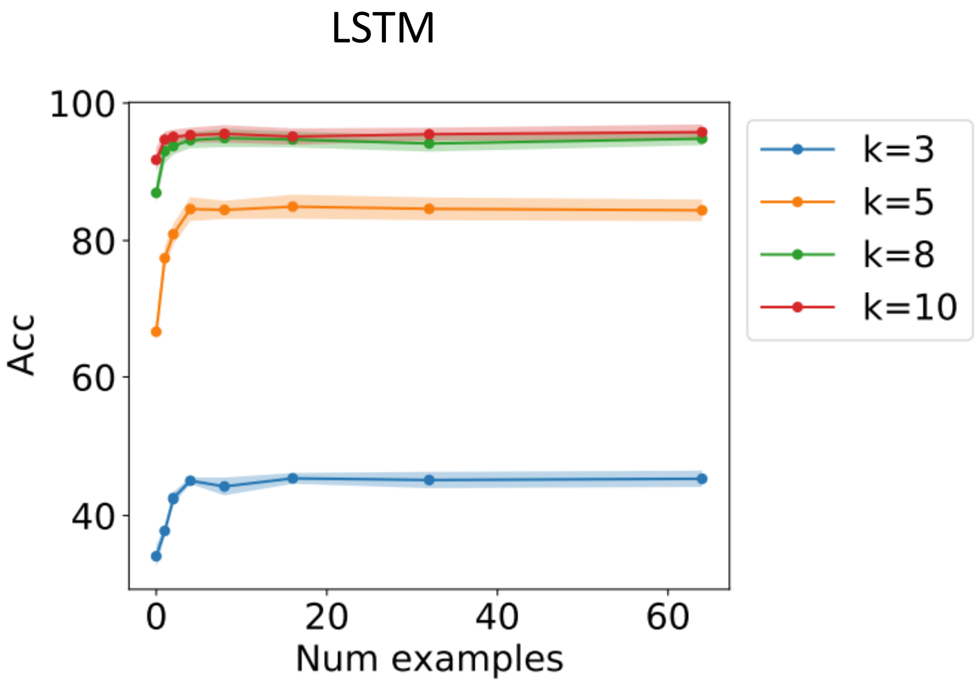
In-context Prompt

```
l aw ac / ax aj ae / ac j
```

In-context learning in GINC

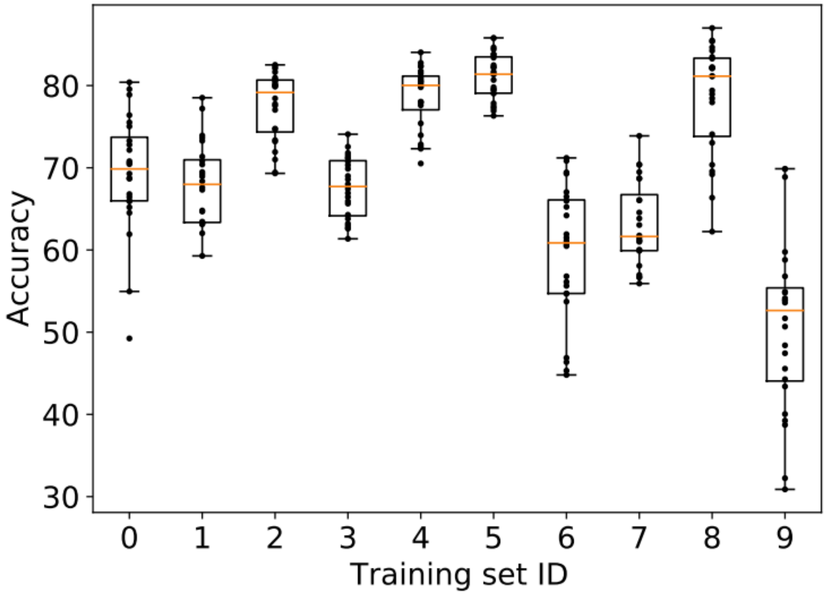
- In-context learning emerges for both Transformers and LSTM
  - Main effect comes from the pretraining distribution





Sensitivity to example ordering

- Zhao et al. 2021: GPT-3 accuracy varies from 50% to 90% depending on example order in prompt
- We mirror this in GINC
- More careful theoretical analysis could capture effect of order



Extrapolation to unseen tasks

GPT-3 seems to work on weird / unseen tasks (Rong et al 2021)

Training examples (truncated)

```
beet: sport
golf: animal
horse: plant/vegetable
corn: sport
football: animal
```

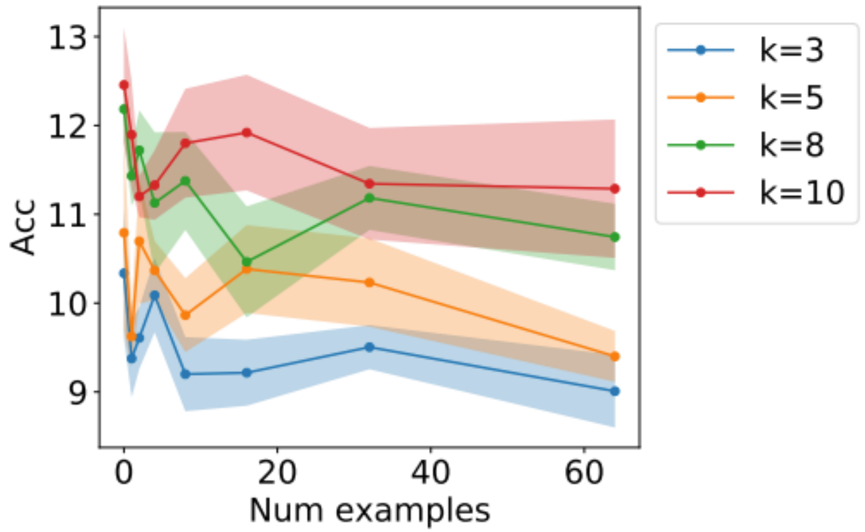
→

Test input and predictions

```
monkey: plant/vegetable ✓
panda: plant/vegetable ✓
cucumber: sport ✓
peas: sport ✓
baseball: animal ✓
tennis: animal ✓
```

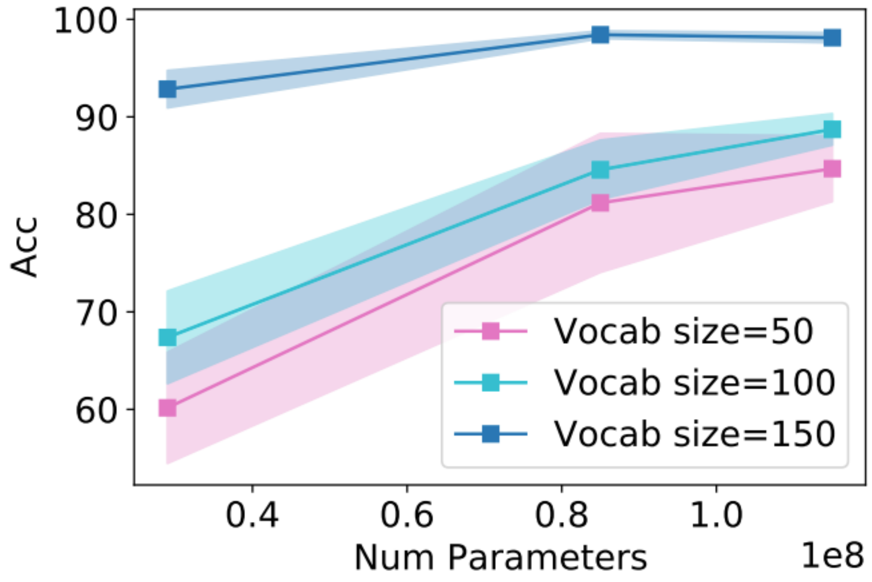
Extrapolation to unseen tasks

- Our theory is limited to concepts seen during pretraining
- In GINC, random unseen concepts can't be learned by in-context learning
- However, still possible for Bayesian inference to extrapolate: e.g., separate latent variables for semantics and syntax -> generalize to new combinations



Effect of model scaling

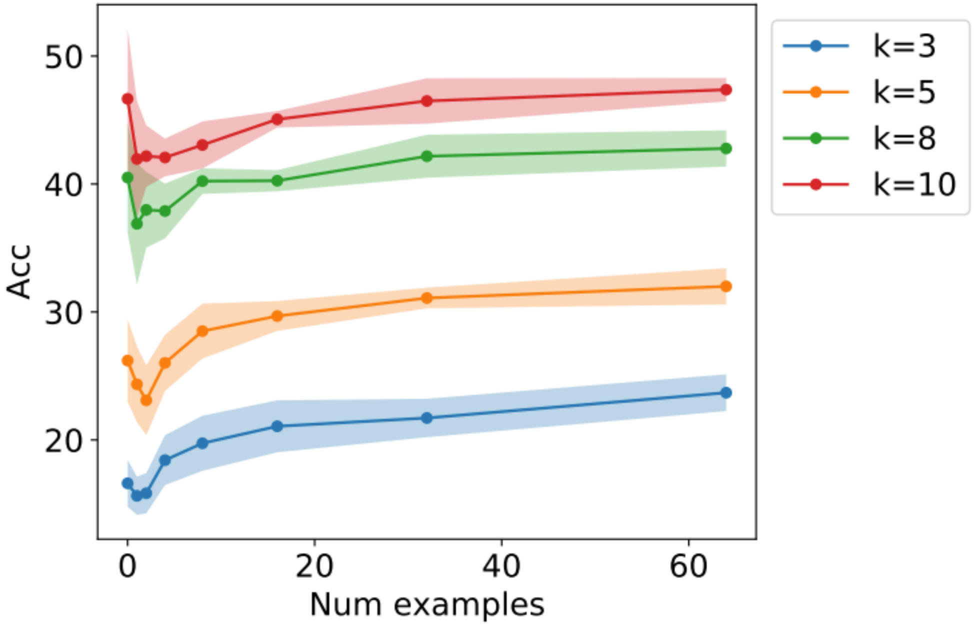
- In GINC: in-context accuracy improves with model size (as is common)
- Interestingly, improves **even if pretraining loss is the same**
- Inductive bias for in-context learning improves with model size?



Transformer # layers	GINC Vocab size	Pretrain Val loss	In-context Acc
12 layer	50	1.33	81.2
16 layer	50	1.33	**84.7**

Zero-shot is sometimes better than 1-shot

- Zero-shot in GPT-3 is better than 1-shot for some datasets (e.g., LAMBADA, HellaSwag, PhysicalQA, RACE-m)
- We also find instances in GINC where adding 1 training example (1 low-prob transition) hurts performance



Small-scale test bed

- Can quickly try out different prompting strategies
- Can test out different pretraining methods as well

Summary

- Pre-trained models need to be adapted in some way
  - Naïve adaptation can lead to larger change than necessary which can lead to “overfitting”
  - Simple changes can fix these problems (like LP-FT)

- Can the model automatically discover how to adapt?
  - “In-context learning” gets at that and is a surprising capability
  - We do not have a good understanding of where this ability comes from, how to best harness it, and how to pre-train to induce in-context learning

Thanks!

Ananya Kumar

Sang Michael Xie

Robbie Jones

Tengyu Ma

Percy Liang

Open
Philanthropy

About me

- I am new to CMU and happy to chat with you
- I work in machine learning, particularly interested in
  - Making ML models work when test distribution differs from train distribution
  - Uncovering and understanding surprising or unintended trends in models
- raditi@cmu.edu; GHC 7005

Appendix

Internals of GINC

- GINC outputs tokens from a memory matrix
  - Rows are “entities”
  - Columns are “properties” (e.g., name, nationality)

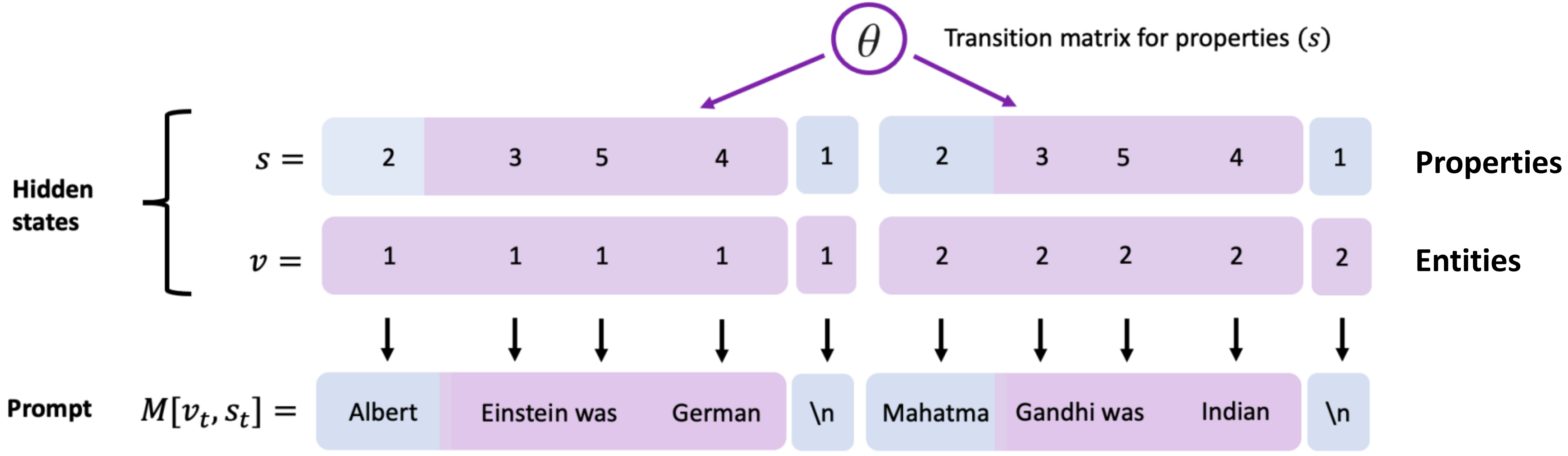
Memory matrix $M =$

Properties (s)

Entities (v)	Newline	First name	Last name	Nationality	Linking verb	etc.
	\n	Albert	Einstein	German	was	⋯
	\n	Mahatma	Gandhi	Indian	was	
			⋮			

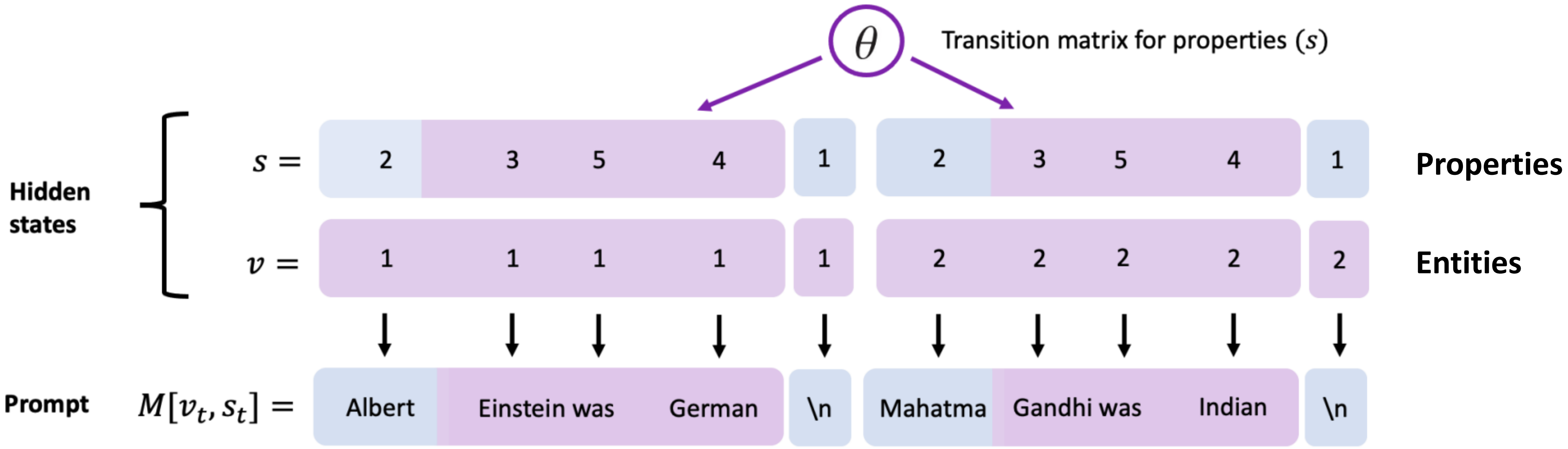
Internals of GINC

- GINC defines a mixture of HMMs
  - 2 independent hidden state chains (properties and entities)
  - Output by indexing into a memory matrix M



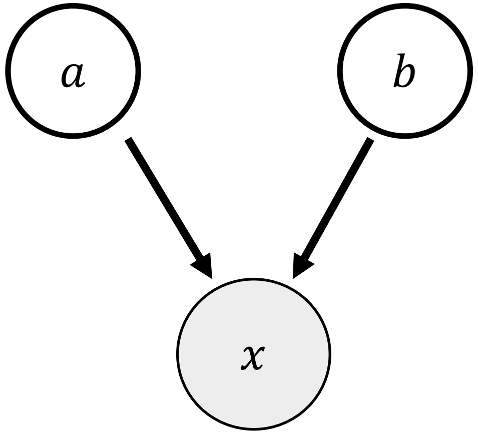
Internals of GINC

- Concept θ is the property transition matrix
  - Pattern of properties defines the “task” (name->nationality)
- Entity transition matrix is fixed and entities evolve slowly



Extrapolation to unseen concepts

- Is extrapolation possible?
  - Possible extension: pretraining distribution samples both entity and property transition matrices from a prior distribution
  - Extrapolate to new entity-property pairs
- Simple illustration
  - 2 latents a, b
  - Observed variable x
  - Perhaps not all pairs of a, b are present in training data, but extrapolation to new pairs may still be possible



- In general, possibly learn a family of “operations” on existing concepts

GPT-3 experiment on LAMBADA

- Does example length matter in GPT3?
- Define short examples (200-300 characters) and long examples (500-600 chars) in LAMBADA completion task
- Test on short examples only: long examples improve performance without adding explicit task-related information or examples

Prompt example length	Test Acc (200–300 chars)
5 examples	
Short (200–300 chars)	69.8
Long (500–600 chars)	70.7
10 examples	
Short, duplicated examples	$\boxed{69.6}$
Short, independent examples	71.4

Duplicating short examples to have same total prompt length doesn't help