ellis

European Laboratory for Learning and Intelligent Systems

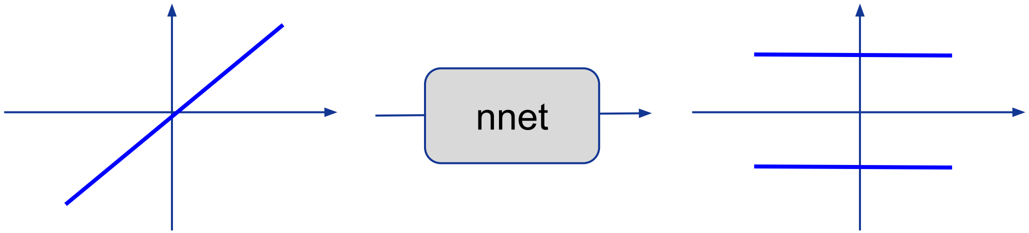
Large-Scale Continual Learning

Marc'Aurelio Ranzato
Google DeepMind
ranzato@google.com

ELLIS Summer school
Large-scale Artificial Intelligence
Modena, Italy 18-22 September 2023

Trivia Quiz

Can you design a neural net layer s.t.:



Google DeepMind

Large-Scale Continual Learning

Marc'Aurelio Ranzato
ranzato@google.com
https://ranzato.github.io/

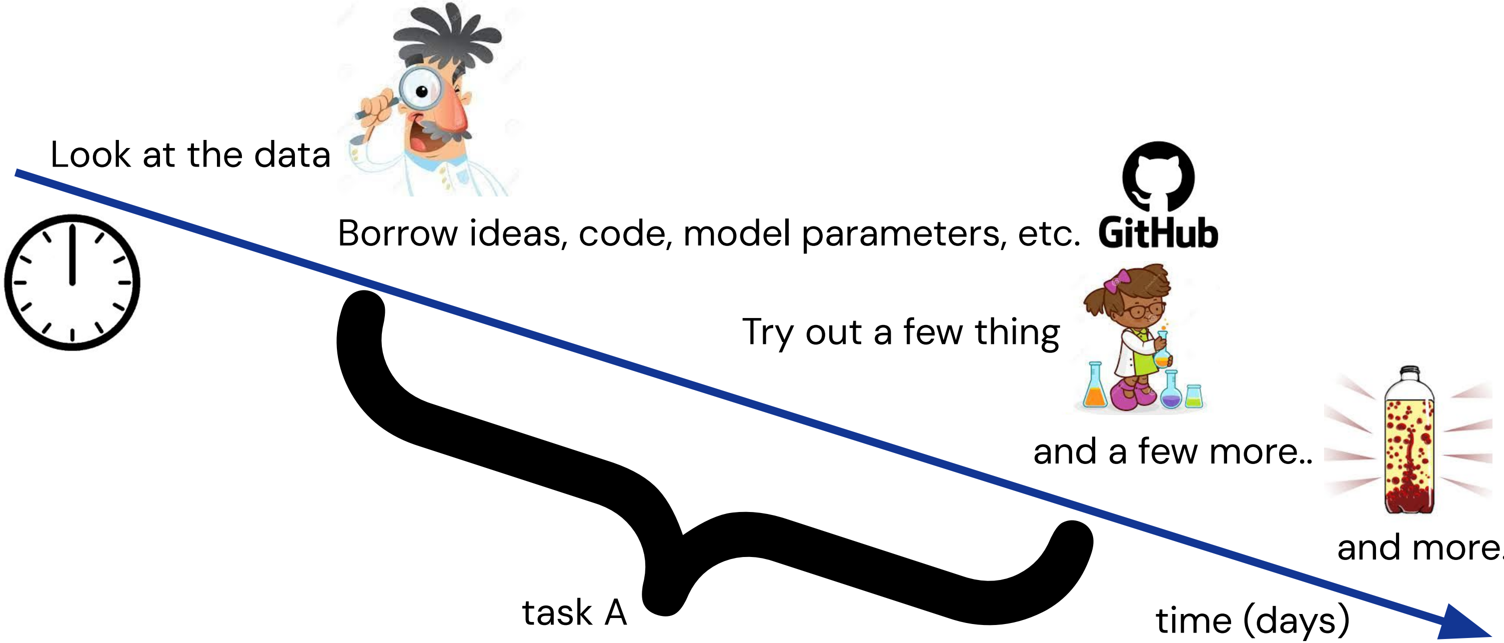
Summer School on Large-Scale AI, Modena, 22 Septmeber 2023

Agenda

- Motivation [15min]
    - Continual Learning
    - Modular Learning
- Modularity via Mixture Models [60min]
    - Detour
    - Flat Hard Mixtures [1, 2]
    - Composable Mixtures [3]
- Conclusions [15min]

References
[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023
[3] Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors ICLR 2021

The typical life cycle of a ML practitioner:



Learning is a continual process.
Learning is about striking good trade-offs.

Look at the data

Borrow ideas, code, model parameters, etc. GitHub

Try out a few things

and a few more..

and more..

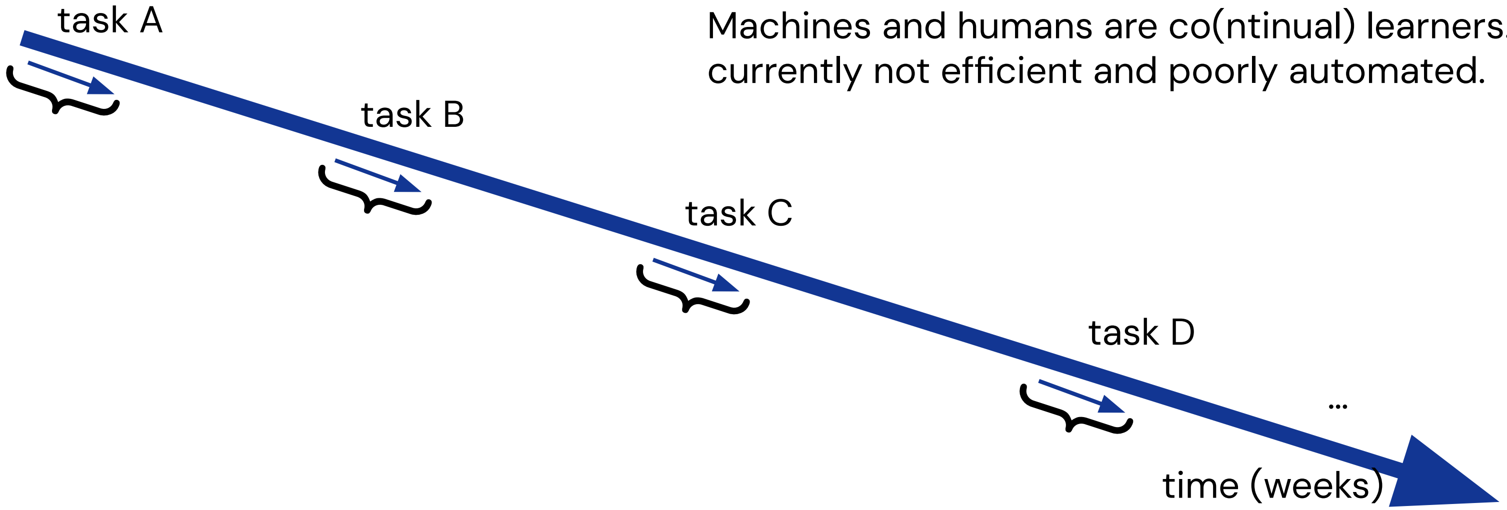
task A

time (days)

The typical life cycle of a ML practitioner:

There is a hierarchy of continual learning problems.

Machines and humans are co(ntinual) learners. The process is currently not efficient and poorly automated.

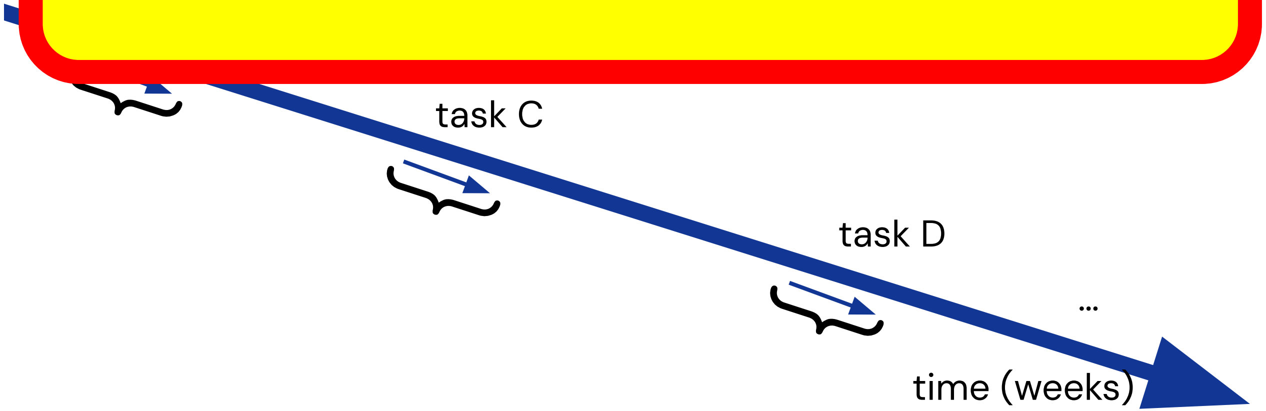


Learning is a continual process.
Learning is about striking good trade-offs.

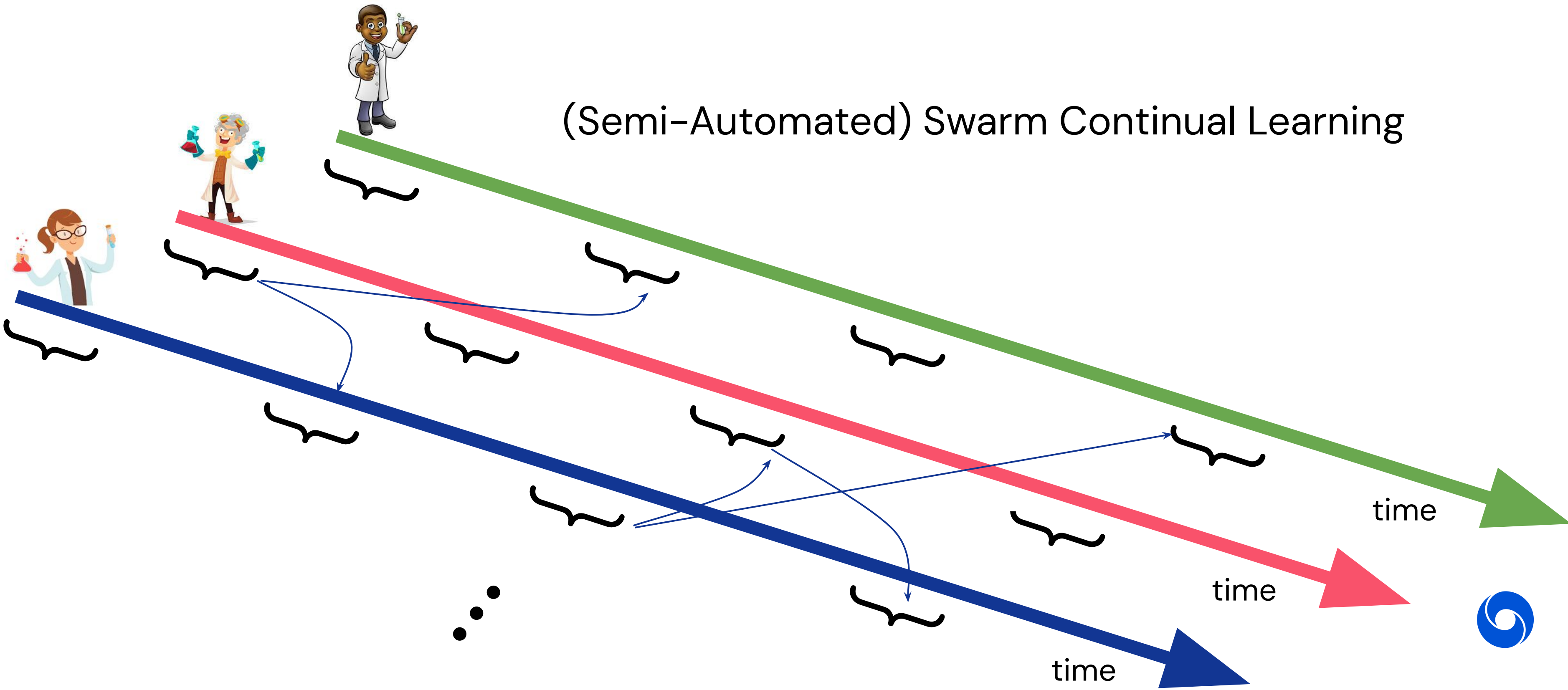
ems.

Continual learning enables efficient learning, via knowledge transfer.

The process is



(Semi-Automated) Swarm Continual Learning

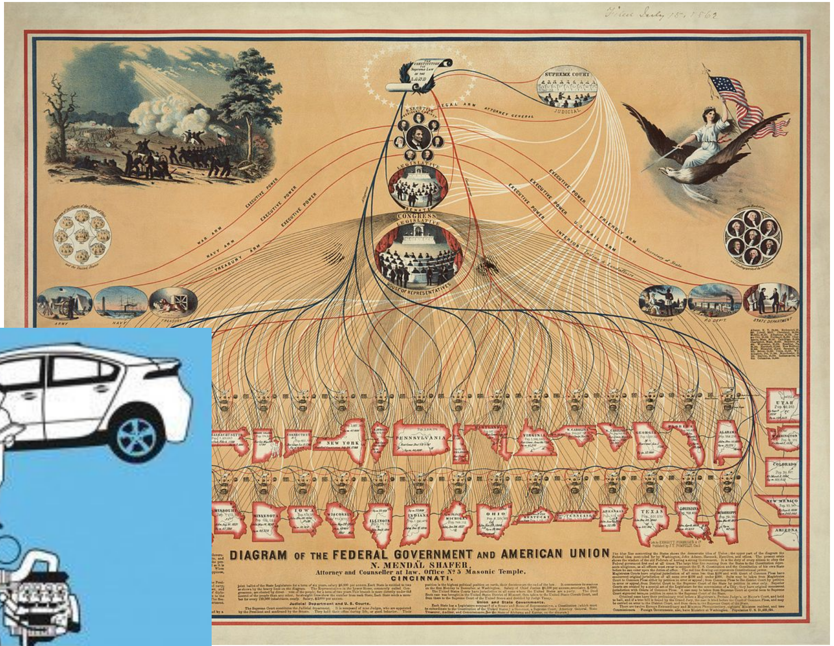


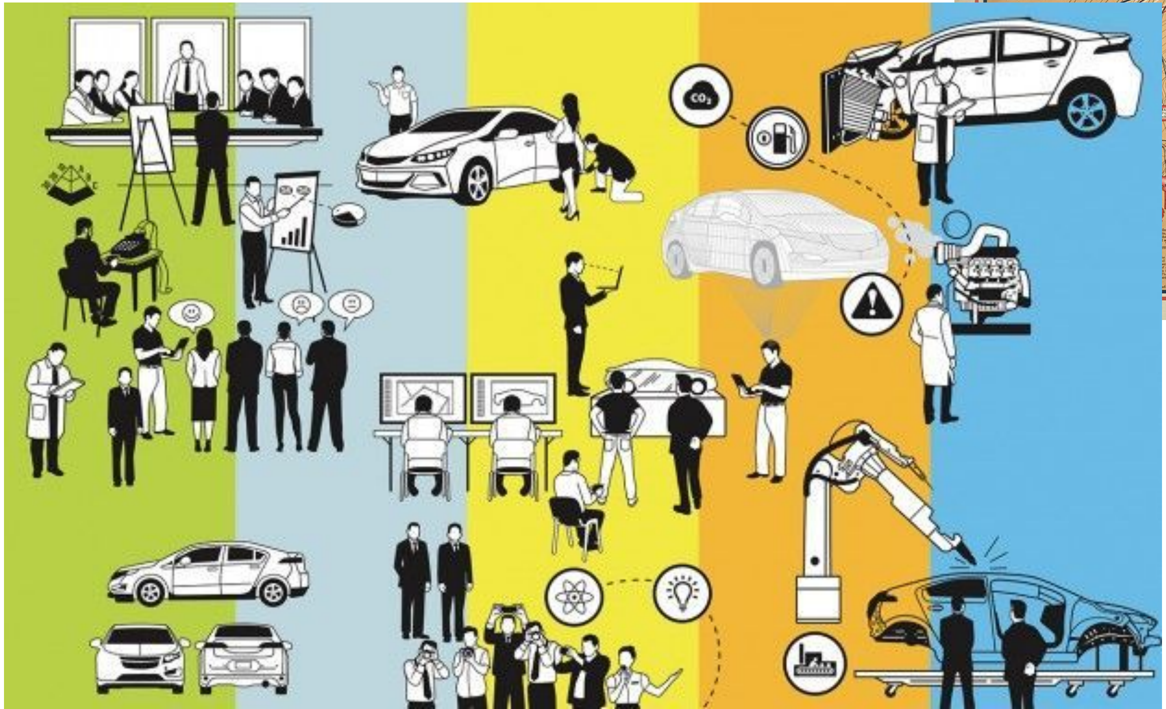
Learning is a continual process.
Learning is about striking good trade-offs.

Continual learning enables efficient learning, via knowledge transfer.

Continual learning is already modular!

Divide & Conquer





Dream: Distributed Never-Ending Learning System

Public

Dream: Distributed Never-Ending Learning System

Public

Only few parts are needed at any given time.

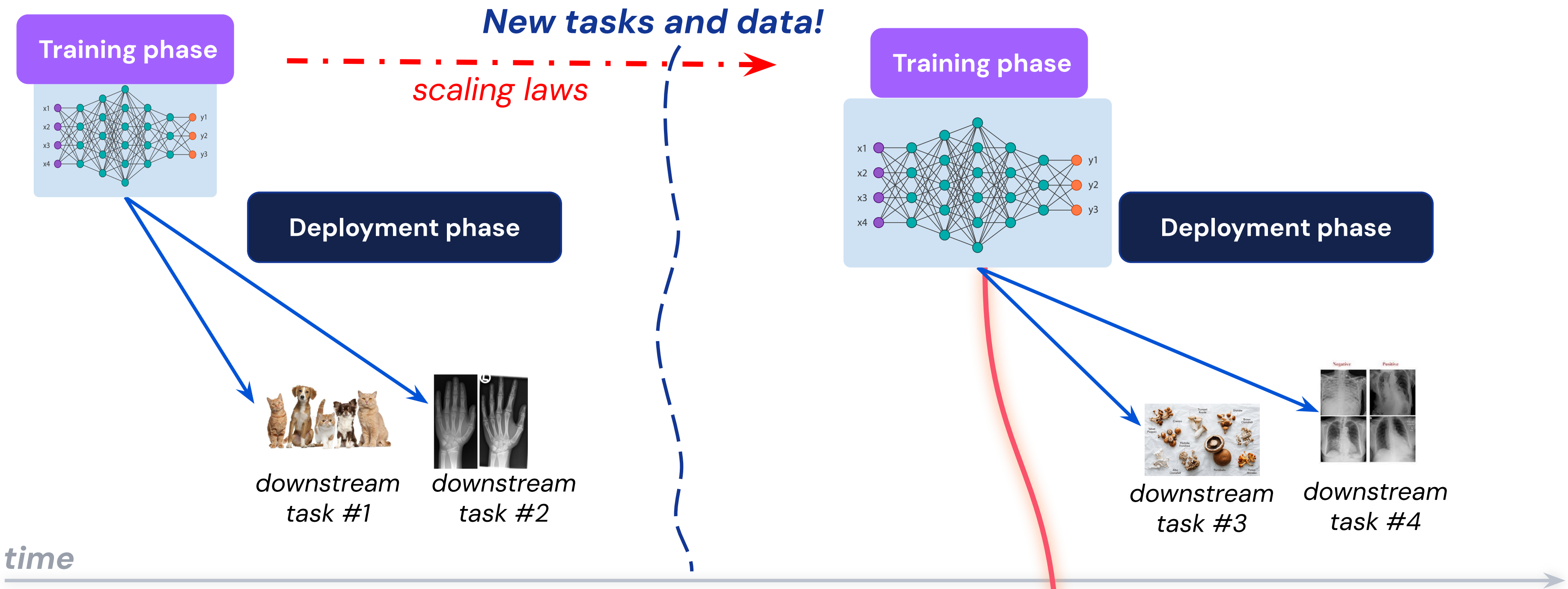
Dream: Distributed Never-Ending Learning System

Public

Only few parts are needed at any given time.
Parts can be added/removed/updated over time.

[Colin Raffel's blog post on github ML]

The current state of affairs in large-scale modeling



Current large-scale systems are accurate but costly and inefficient.
Monolithic & inefficient because of the weak transfer.

Hard Questions

- What abstraction to use for continual learning?
    - What does cross-validation mean in this context?
    - What data can be useful to study this problem in a controlled setting?
- How to characterize a modular model?
- How to measure performance?

It is often a good idea to start from a concrete application or problem, and derive from there abstractions.
Judgement is required to figure out the good level of coarseness of the abstraction.
In our case, we want to build a large-scale system that is effective but also efficient at both training and testing time.

Public

Summary

- Assumptions
  - No entity nor individual will ever have enough resources:
    - bigger is always better
    - there is always something new to learn
- Observations
  - ML is already continual
  - ML is already modular
- Goal
  - Build a single distributed never-ending learning system which is more efficient and scalable
- Hypotheses
  - ML needs to be co-designed with distributed hardware
  - Modularity as a way to cooperate
  - Modularity is key to retain/gain efficiency as we scale

Agenda

- Motivation [15min]
  - Continual Learning
  - Modular Learning
- Modularity via Mixture Models [60min]
  - Detour
  - Flat Hard Mixtures [1, 2]
  - Composable Mixtures [3]
- Conclusions [15min]

References
[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023
[3] Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors ICLR 2021

Public

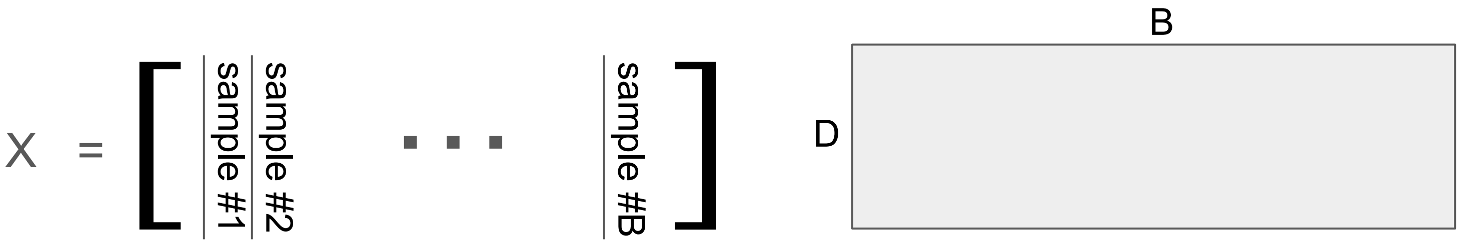
Setting

X data matrix of size DxB

Each input feature has dimensionality D.

There are B samples.

Assume: B >> D.



Matrix Factorization: SVD

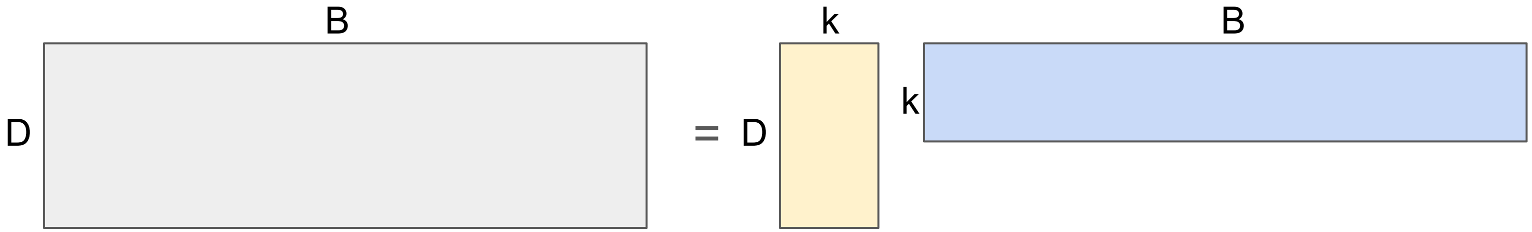
$X = U S V = (U\, \mathrm{sqrt}(S))\, ((\mathrm{sqrt}(S)\, V) = W Z$

$W = U\, \mathrm{sqrt}(S)$ matrix of size $D \times k$, $k < D$

$Z = \mathrm{sqrt}(S)\, V$ matrix of size $k \times B$, $k < B$

Columns of W can be interpreted as bases.

Columns of Z are interpreted as code/features/latent variables/representations of samples in X.



Matrix Factorization: NMF, FA, …

There are LOTS of ways to factorize a matrix, each imposing a different set of constraints.

Matrix Factorization: Sparse Coding

X = W Z

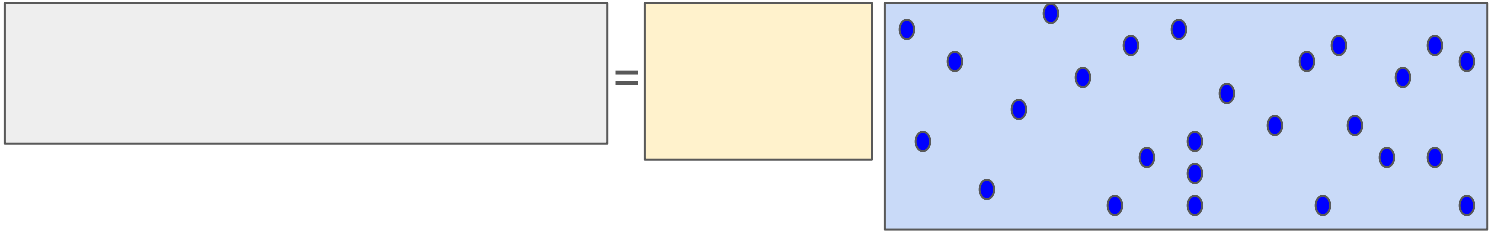
W matrix of size D x k

Z   matrix of size k x B

This time, k is not constrained to be less than D! Nor columns of W need to be orthogonal.

The constraint is:

||Z||_0 < threshold



Matrix Factorization: Sparse Coding

$X = W Z$

W matrix of size $D \times k$

Z matrix of size $k \times B$

How to find Z:

$L = ||X - WZ||^2 + \lambda ||Z||_0$

Matrix Factorization: Sparse Coding

X = W Z

W matrix of size D x k

Z   matrix of size k x B

How to find Z in practice? Minimize:

L = ||X - WZ||^2 + \lambda ||Z||_1

Matrix Factorization: Sparse Coding

$X = W Z$

W matrix of size $D \times k$

Z matrix of size $k \times B$

How to find Z in practice:

$L = ||X - WZ||^2 + \lambda ||Z||_1$

How to find W (for a given Z):

Example of EM (or coordinate descent)

$L = ||X - WZ||^2$

Matrix Factorization: Efficient Sparse Coding

$X = W Z$

W matrix of size $D \times k$

Z matrix of size $k \times B$

How to find Z in practice:

Learn a neural predictor $Z \sim g(X)$ to approximate the costly optimization process!

$L = ||X - WZ||^2 + \lambda ||Z||_1$

How to find W (for a given Z):

Example of EM (or coordinate descent)

$L = ||X - WZ||^2$

Kavukcuoglu et al. "[Fast inference in sparse coding..]" arXiv 2008 and my PhD thesis

Matrix Factorization: k-Means

X = W Z

W matrix of size D x k

Z matrix of size k x B

How to find Z in practice:

L = ||X - WZ||^2, s.t. Z being 1-of-k

How to find W (for a given Z):

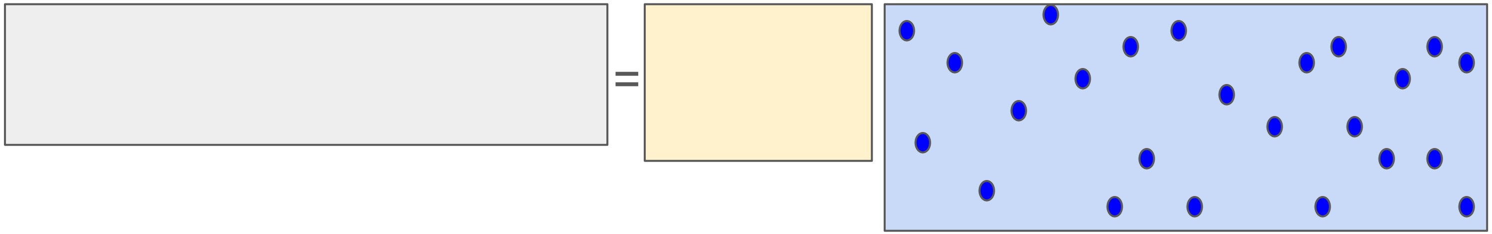
L = ||X - WZ||^2

Example of EM (or coordinate descent)


END
DETOUR

k-Means Recipe

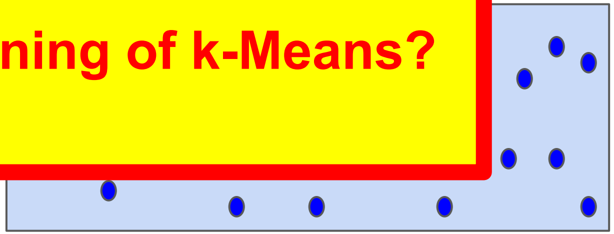
1) Assign each example to a cluster
2) Update prototypes using all samples assigned to them



k-Means Recipe

1) Assign each example to a cluster
2) Update prototypes using all samples assigned to them

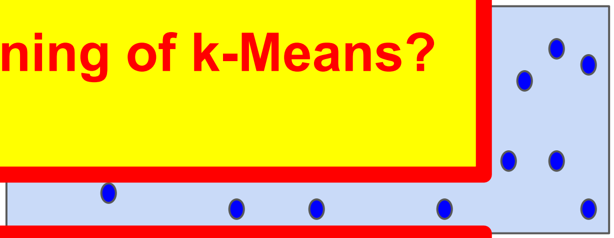
How could we distribute training of k-Means?



k-Means Recipe

1) Assign each example to a cluster
2) Update prototypes using all samples assigned to them

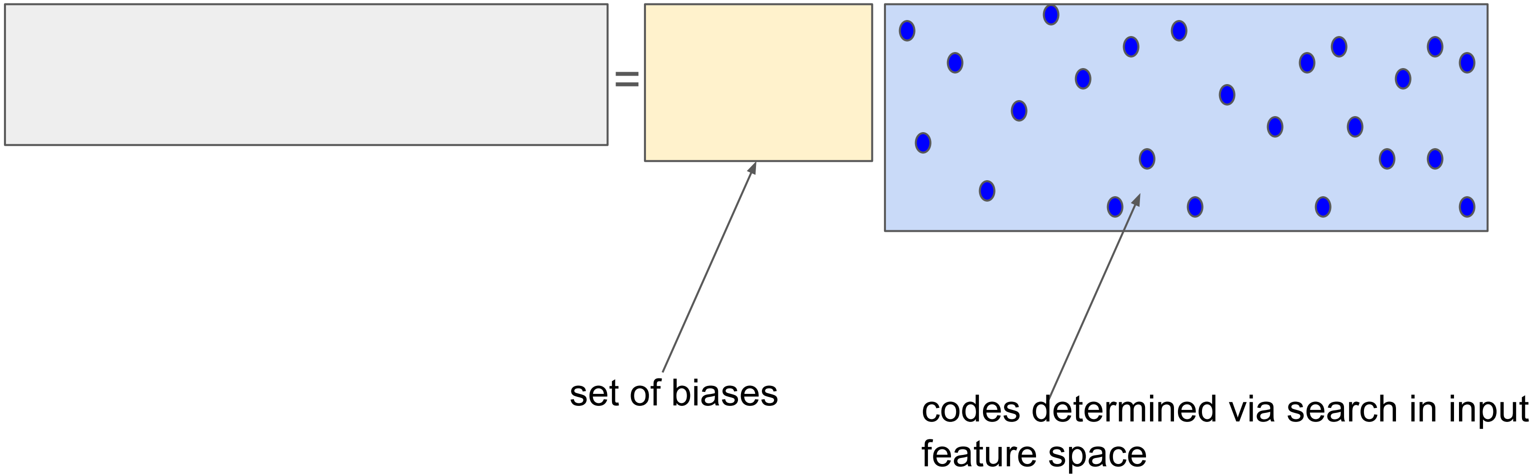
How could we distribute training of k-Means?



How could we make k-Means “deep”?

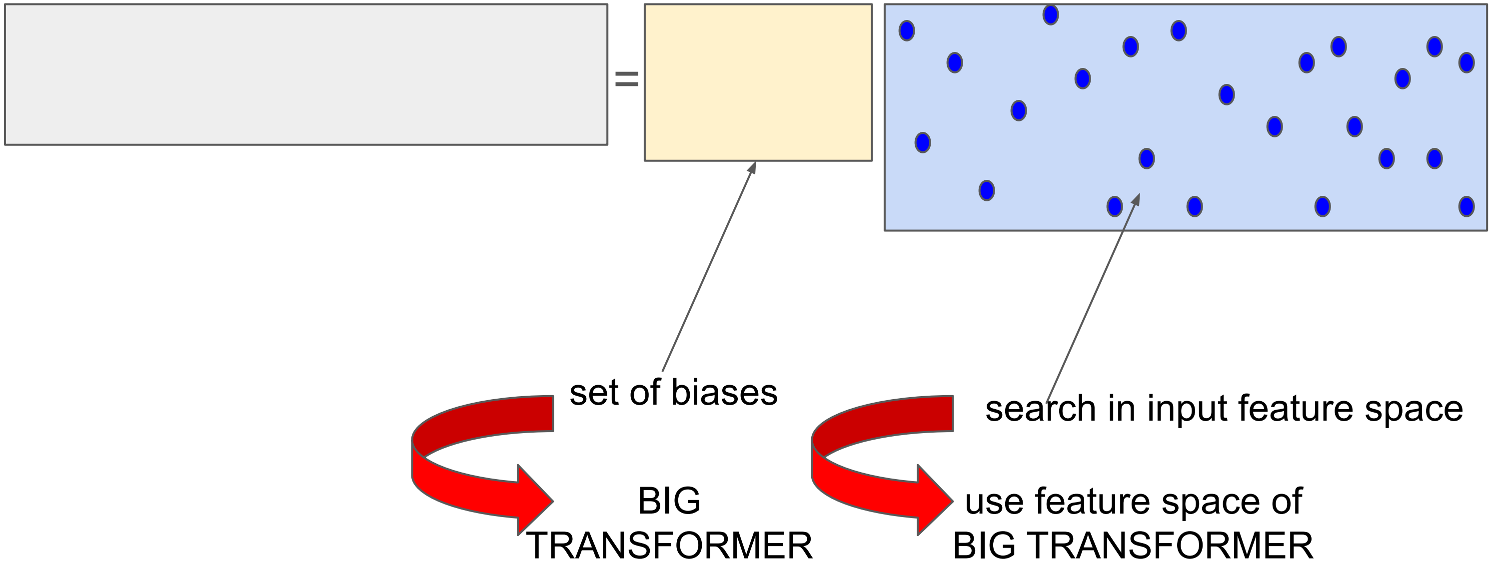
k-Means Recipe

1) Assign each example to a cluster
2) Update prototypes using all samples assigned to them



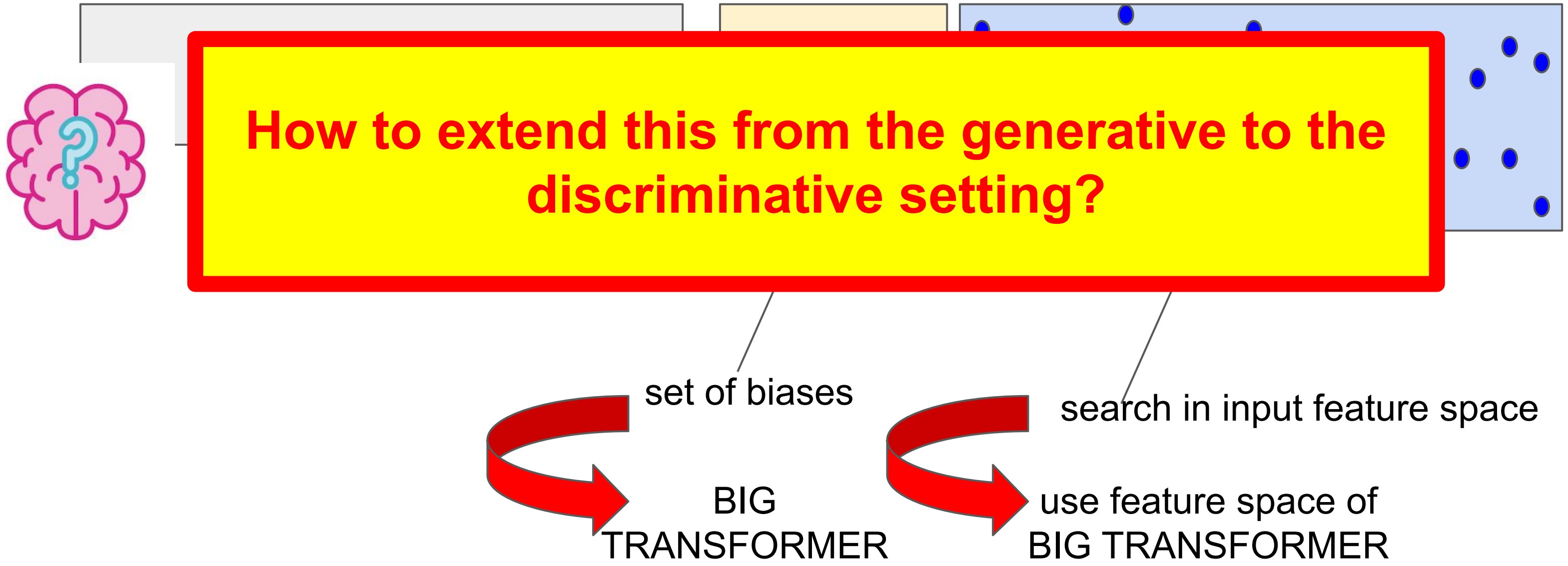
k-Means Recipe

1) Assign each example to a cluster
2) Update prototypes using all samples assigned to them



k-Means Recipe

1) Assign each example to a cluster
2) Update prototypes using all samples assigned to them



Public

Summary

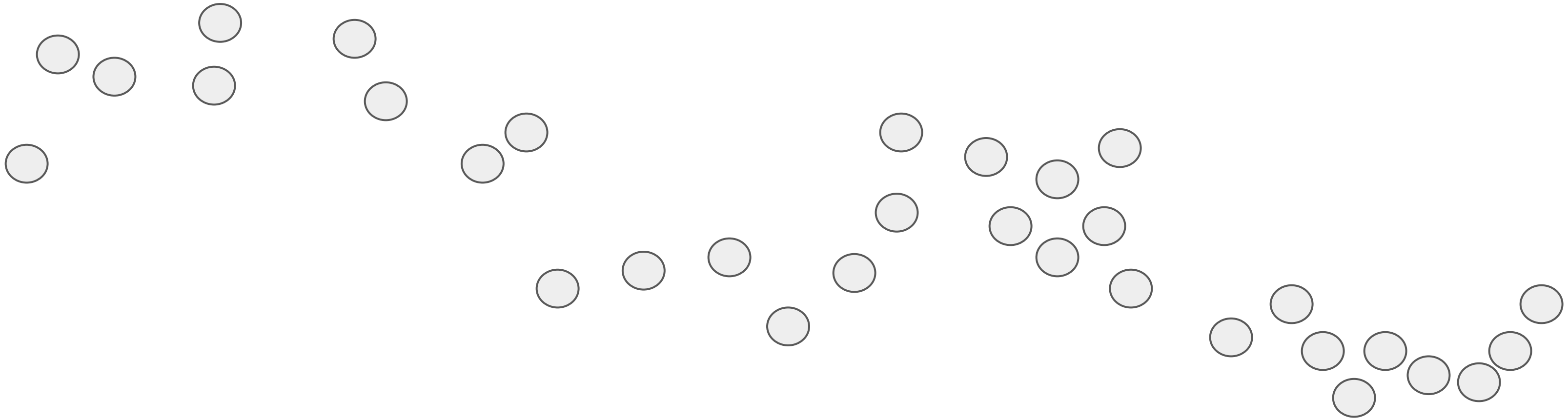
- Matrix Factorization = Dataset Compression = Model Learning
  - Core building block of ML
  - Sparse coding & k-Means are special case
- k-Means is easy to distribute/scale up
- Is there a deep version of k-Means?

Agenda

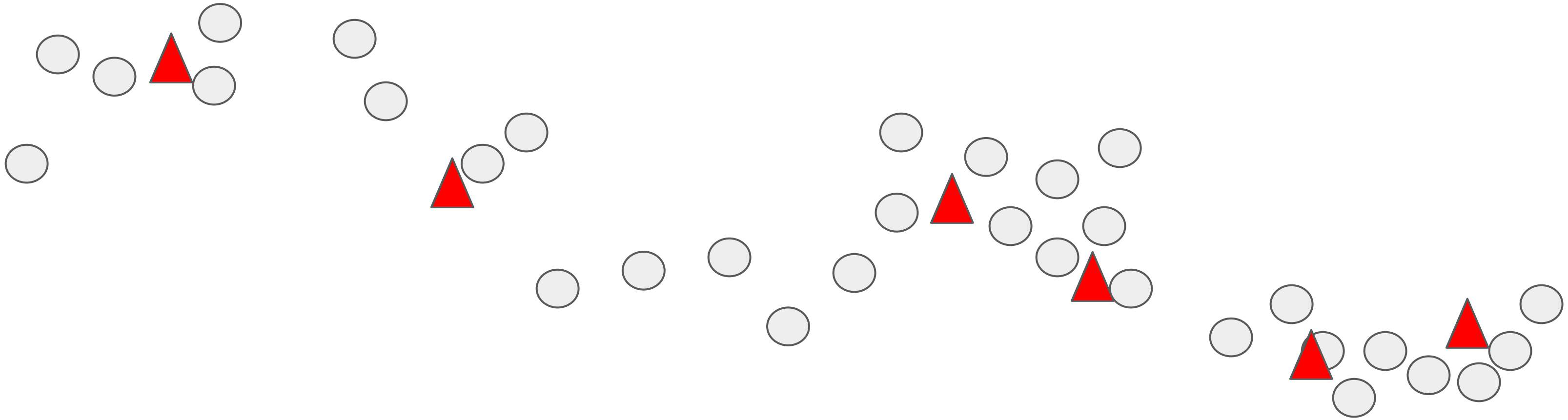
- Motivation [15min]
  - Continual Learning
  - Modular Learning
- Modularity via Mixture Models [60min]
  - Detour
  - Flat Hard Mixtures [1, 2]
  - Composable Mixtures [3]
- Conclusions [15min]

References
[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023
[3] Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors ICLR 2021

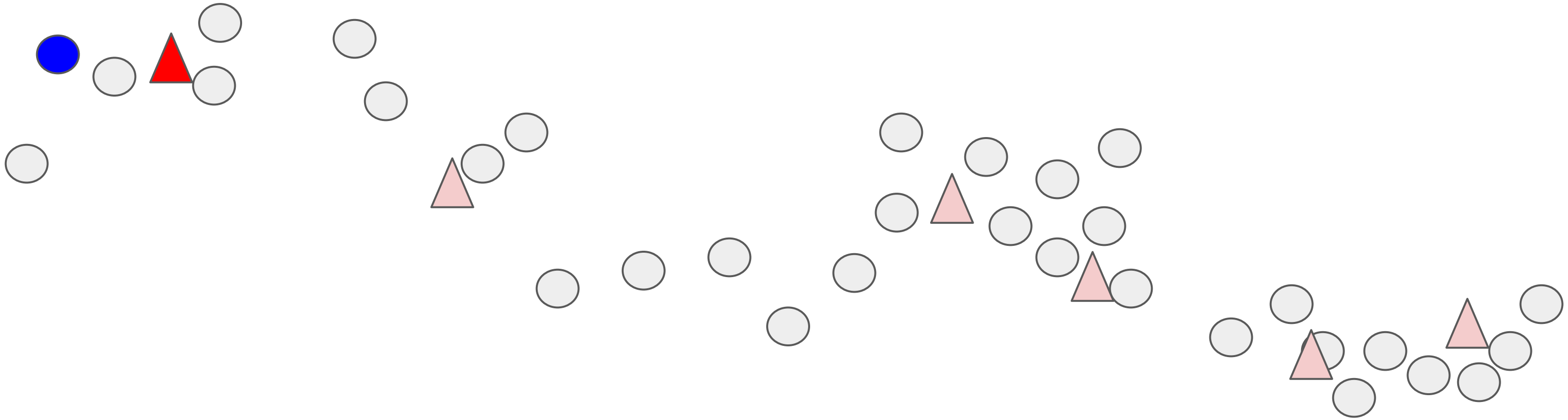
Data



k-Means



k-Means



Local Learning Algorithms

Léon Bottou, Vladimir Vapnik
AT&T Bell Laboratories, Holmdel, NJ 07733, USA

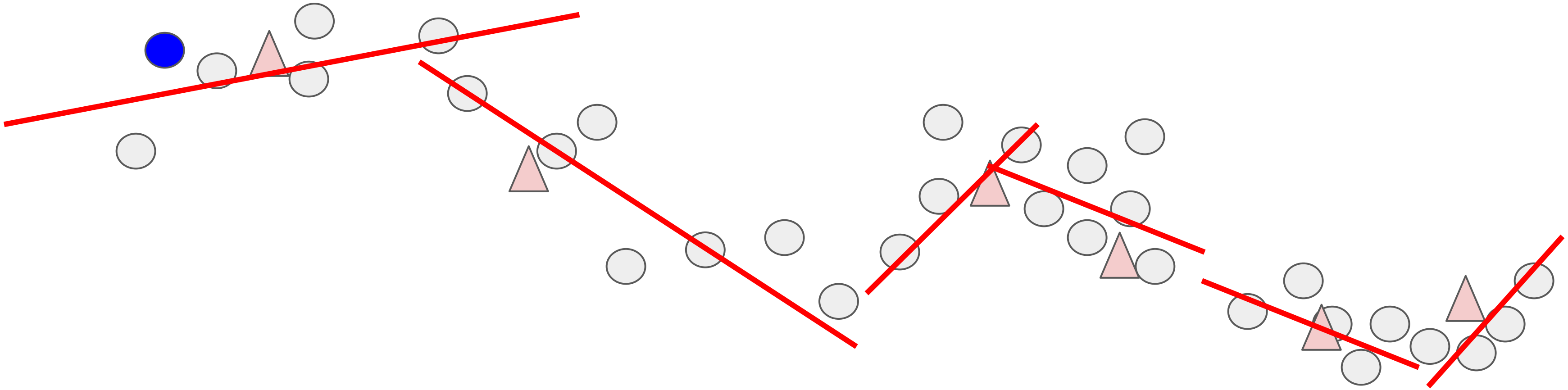
Abstract

Very rarely are training data evenly distributed in the input space. Local learning algorithms attempt to locally adjust the capacity of the training system to the properties of the training set in each area of the input space.

The family of local learning algorithms contains known methods, like the k-Nearest Neighbors method (kNN) or the Radial Basis Function networks (RBF), as well as new algorithms. A single analysis models some aspects of these algorithms. In particular, it suggests that neither kNN or RBF, nor non local classifiers, achieve the best compromise between locality and capacity.

A careful control of these parameters in a simple local learning algorithm has provided a performance breakthrough for an optical character recognition problem. Both the error rate and the rejection performance have been significantly improved.

Clustered / Locally Linear SVMs



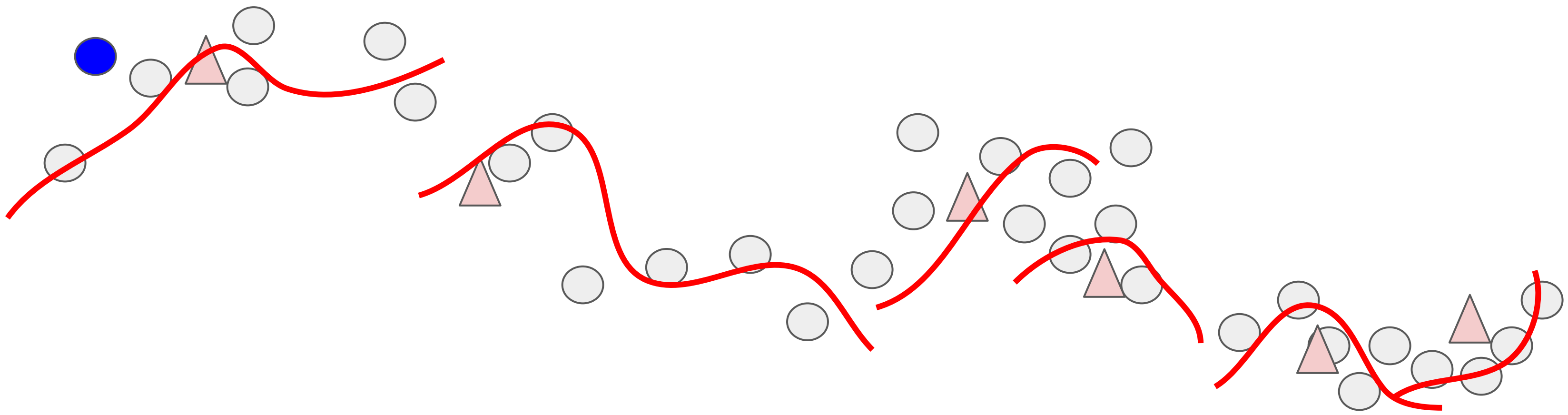
1996 **Clustered Support Vector Machines**

Quanquan Gu
Department of Computer Science
University of Illinois at Urbana-Champaign
qgu3@illinois.edu

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

2013

Flat Hard Mixture of Experts



Hard Mixtures of Experts for Large Scale Weakly Supervised Vision

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam

Facebook AI Research (FAIR)

Abstract

Training convolutional networks (CNN's) that fit on a single GPU with minibatch stochastic gradient descent has become effective in practice. However, there is still no effective method for training large CNN's that do not fit in the memory of a few GPU cards, or for parallelizing CNN training. In this work we show that a simple hard mixture of experts model can be efficiently trained to good effect on large scale hashtag (multilabel) prediction tasks. Mixture of experts models are not new [7, 3], but in the past, researchers have had to devise sophisticated

As the data gets bigger, we can expect to be able to scale up our models as well, and get better features; more data means more refined models with less overfitting. However, even today's state of the art convolutional models cannot keep up with the size of today's weakly supervised data. With our current optimization technology and hardware, more images are posted to photo sharing sites in a day than can be passed through the training pipeline of standard state of the art convolutional architectures. Furthermore, there is evidence [8, 6] and below in this work, that these architectures are already underfitting on datasets at the scale of hundreds of millions of images.

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Assumptions

- Data is plentiful
- Any single model operating on a single machine underfits
- There exist lots of machines
- Communication cost across machines (or engineering cost) is high
- Computation is cheap relative to communication

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Step 0: Train feature extractor

- Train small model on a single machine using standard training tools.
  - E.g.: Supervised learning of a small CNN
- Goal: Learn good representations
  - E.g.: Chop off top-most layer to extract features

$$h(x; \theta_h) : \mathcal{R}^D \rightarrow \mathcal{R}^H$$

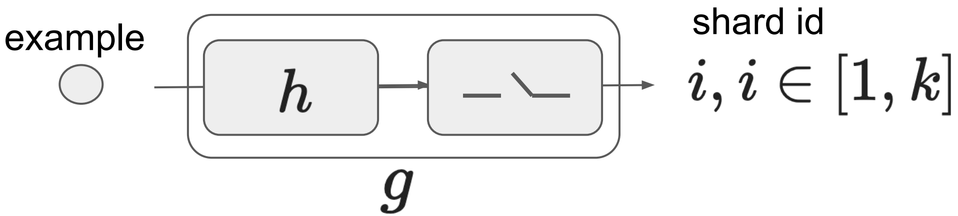
input (image) space

feature space

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Step 1: Train Gater

- Map each input example into feature space of h
- k-Means. Let's call g the composition of h with k-means assignment



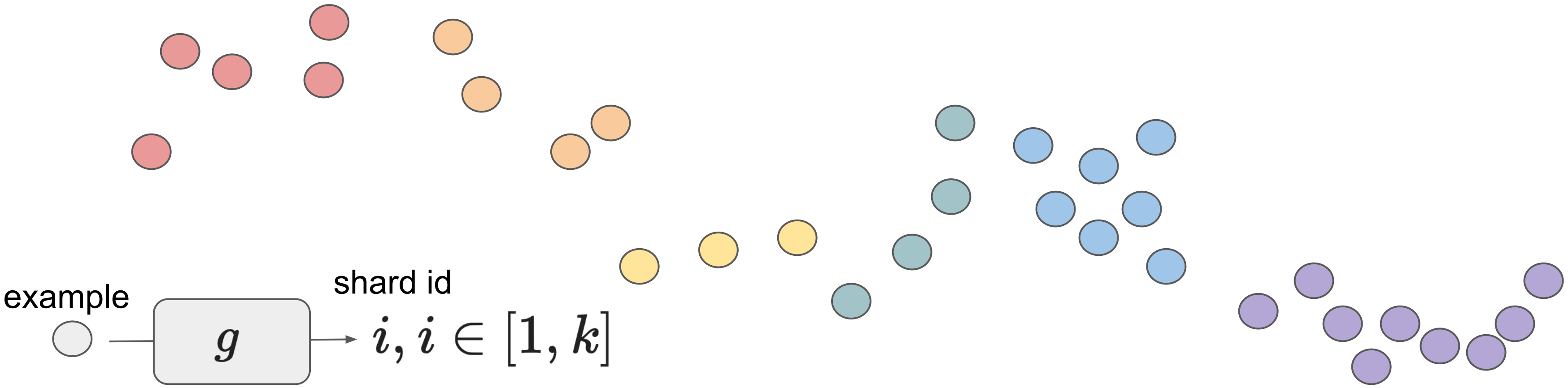
k-Means assignment: $\arg\min_{i \in [1,k]} ||h(x) - W_i||^2$

i-th prototype

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Step 1: Train Gater

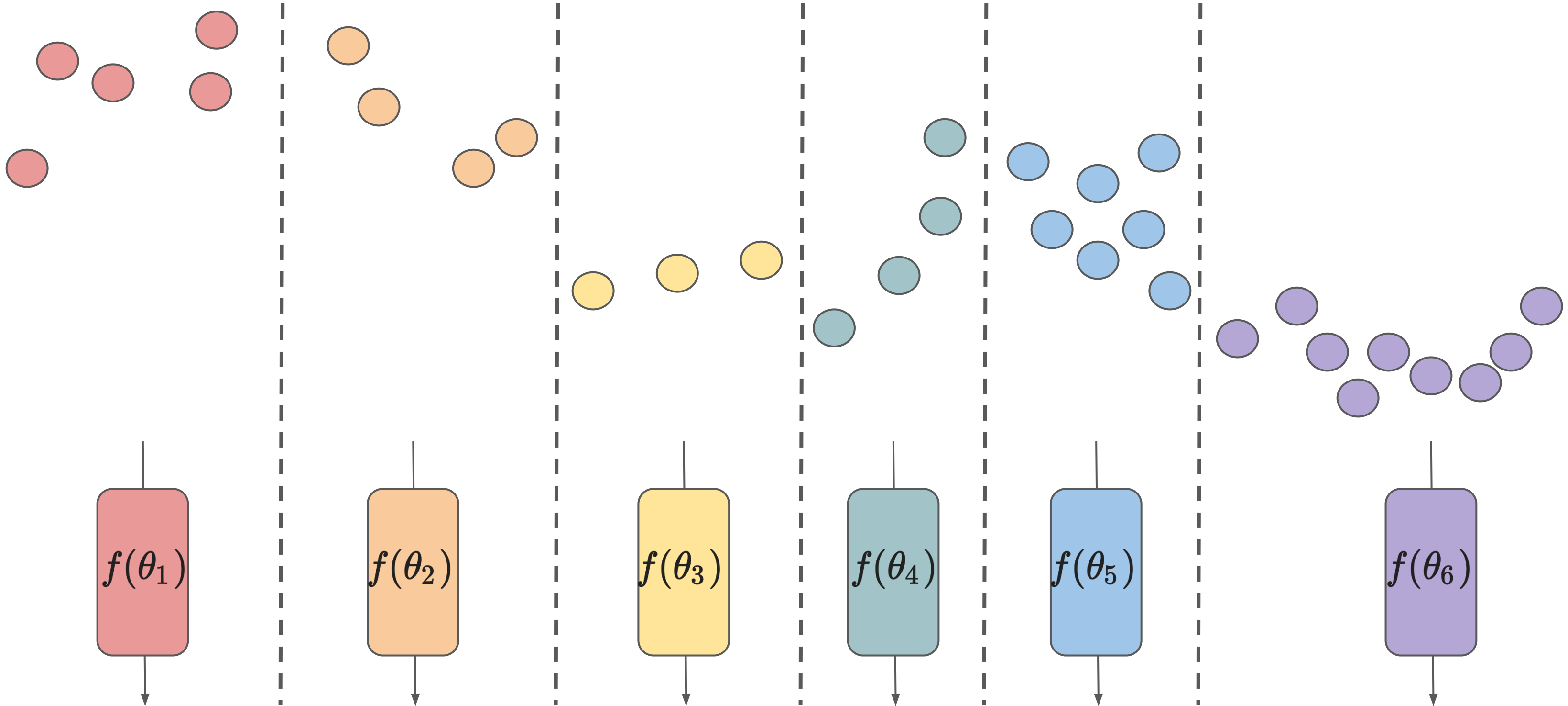
- Map each input example into feature space of h
- k-Means. Let's call g the composition of h with k-means assignment
- Divide original dataset into k shards using g



[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Step 2: Train Experts

- Train a network (expert) on each shard **independently**



Mixture of Experts

output prediction: $$\sum_{i=1}^{k} g(x;\theta_g)f(x;\theta_i)$$

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Mixture of Experts

output prediction: $$\sum_{i=1}^{k} \underbrace{\boxed{g(x;\theta_g)}}_{\text{gater}} f(x;\theta_i)$$

If the gater outputs a 1-of-K distribution over experts, then the mixture is “**hard**”.
In a hard mixture, the sum contains only a single term.

Benefits of hard mixtures: 1) easy to train, 2) cheap to deploy.

Hard EM for hard mixtures: Alternate minimization over gater assignments and experts parameters.
In this paper, we performed just one step!

[1] Gross et al. “Hard mixture of experts for large-scale weakly supervised vision” CVPR 2017

Mixture of Experts

output prediction: $$\sum_{i=1}^{k} \underbrace{\boxed{g(x;\theta_g)}}_{\text{gater}} f(x;\theta_i)$$

Ensembling would instead use a fixed uniform gating.

Hard EM for hard mixtures: Alternate minimization over gater assignments and experts parameters.
In this paper, we performed just one step!

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Mixture of Experts

output prediction: $$\sum_{i=1}^{k} \underbrace{\boxed{g(x;\theta_g)}}_{\text{gater}} f(x;\theta_i)$$

Ensembling would instead use a fixed uniform gating.

Boosting has a rule to set the (constant) gating function and trains experts sequentially…

[1] Gros… mixture of experts for large scale weakly supervised vision CVPR 2017

Mixture of Experts

output prediction: $$\sum_{i=1}^{k} \underbrace{\boxed{g(x;\theta_g)}}_{\text{gater}} \underbrace{\boxed{f(x;\theta_i)}}_{\text{experts}}$$

Experts can be heterogeneous, in terms of:

- architecture,
- hyper-parameters choice,
- training devices,
- etc.

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Results on YFCC100M

Model	q@1
ResNet-18	3.04%
ResNet-34	3.31%
ResNet-50	3.47%
ResNet-50 4×feature size	3.80%
ResNet-18 ensemble-50	3.37%
ResNet-18 MoE-25	5.35%
ResNet-18 MoE-50	6.12%
ResNet-18 MoE-75	6.65%
ResNet-18 MoE-100	6.87%
ResNet-34 MoE-50	6.77%

- Training is fully parallelizable
- For similar inference cost, MoE achieves much better accuracy
- Relative to ensembles, MoEs performs better and is cheaper at test time

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Results on YFCC100M

Model	q@1
ResNet-18	3.04%
ResNet-34	3.31%
ResNet-50	3.47%
ResNet-50 4×feature size	3.80%
ResNet-18 ensemble-50	3.37%
ResNet-18 MoE-25	5.35%
ResNet-18 MoE-50	6.12%
ResNet-18 MoE-75	6.65%
ResNet-18 MoE-100	6.87%
ResNet-34 MoE-50	6.77%

- Training is fully parallelizable
- For similar inference cost, MoE achieves much better accuracy

Results on ImageNet

Model	Top-1 error	Top-5 error
ResNet-18	30.64	10.69%
ResNet-18 MoE-50	30.43	11.7%

too much overfitting…

[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017

Summary

- Matrix Factorization
  - Core building block of ML
  - Sparse coding & k-Means are a special case
- Hard MoEs can be interpreted as generalization of k-Means
- Hard mixtures are amenable to distributed computation

Public

Summary

Hard mixtures

- pros
  - simple
  - trivially parallelizable (no synchronization)
  - cheap at test time
  - works when base model underfits
  - supports continual learning
- Cons
  - ML inefficient: requires more total parameters and more passes over data
  - computationally inefficient at training time

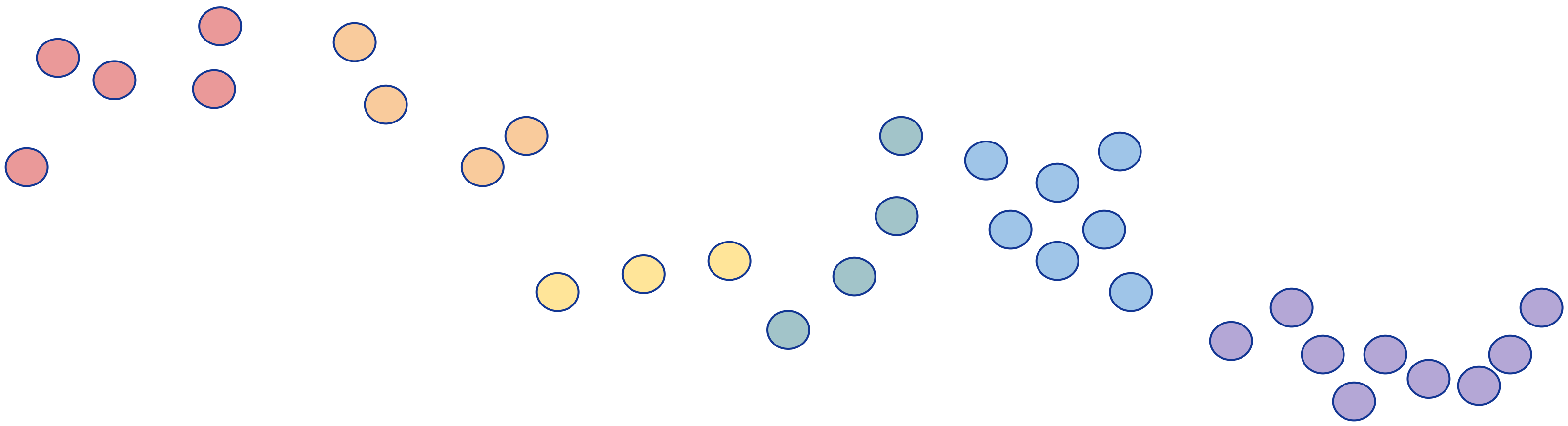
Public

Summary

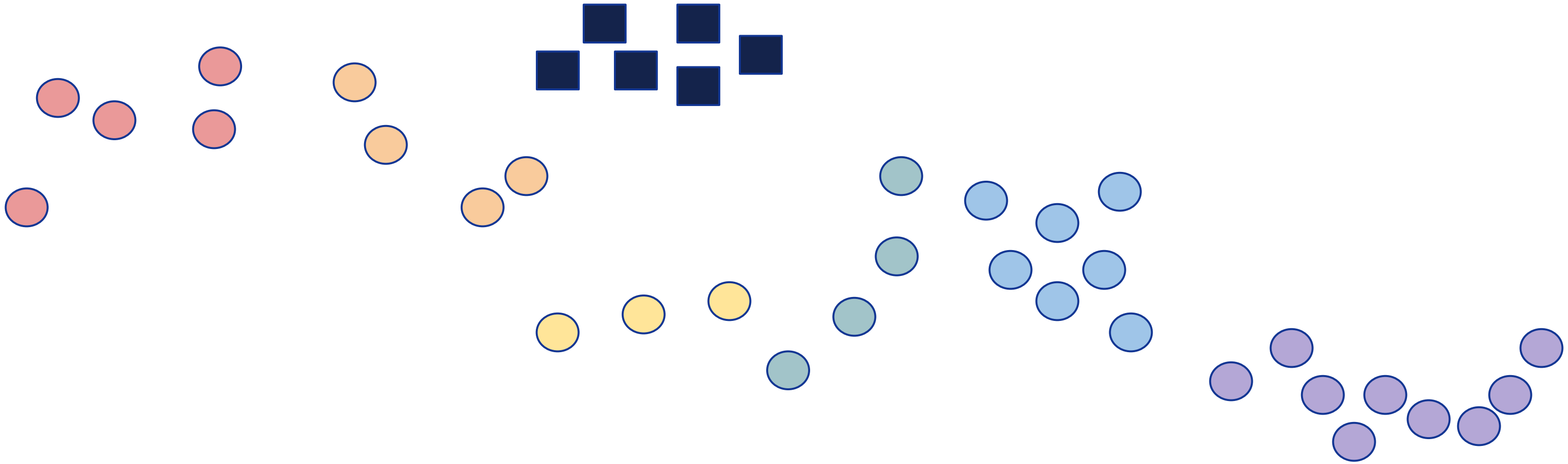
Hard mixtures

- pros
  - simple
  - trivially parallelizable (no communication)
  - cheap at test time
  - works when base model underfits
  - supports continual learning
- Cons
  - ML inefficient: requires more total parameters and more passes over data
  - computationally inefficient at training time

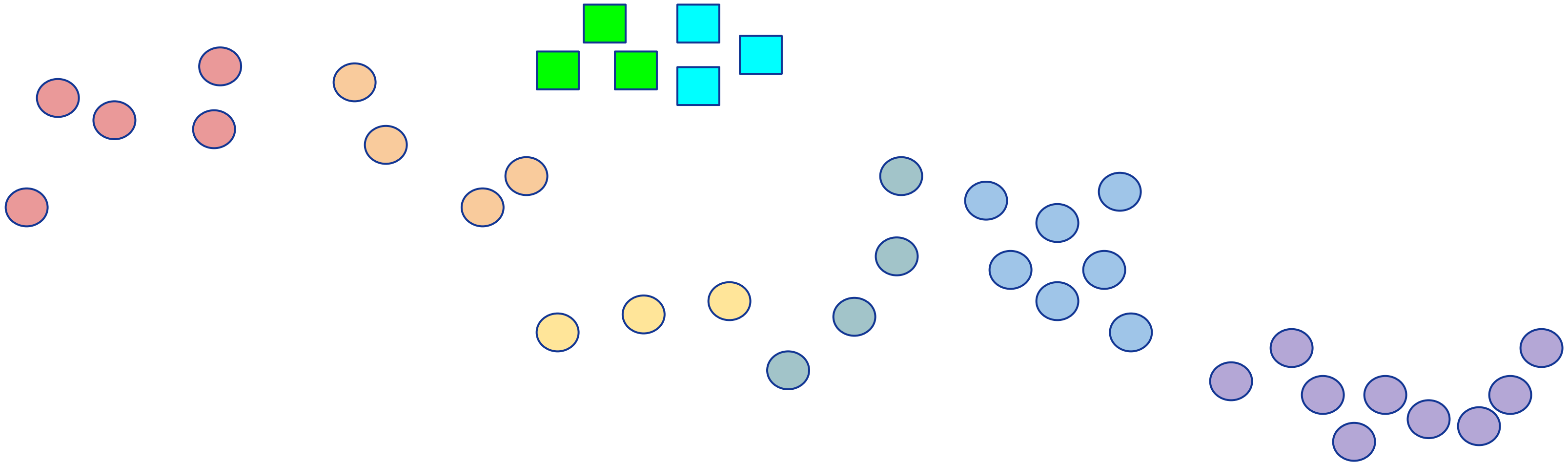
Original dataset



New data can be handled by adding a new cluster/expert



Performance can be improved by splitting an expert in two, for instance.



In general, experts can be updated, added, and removed on demand.

Public

Summary

Hard mixtures

- pros
  - sin
  - su

How to transfer knowledge across experts?

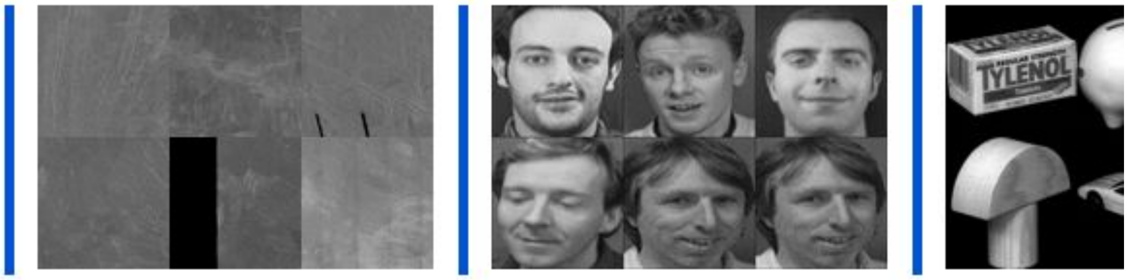
- Cons
  - ML inefficient: requires more total parameters and more passes over data
  - computationally inefficient at training time

NEVIS'22 Benchmark

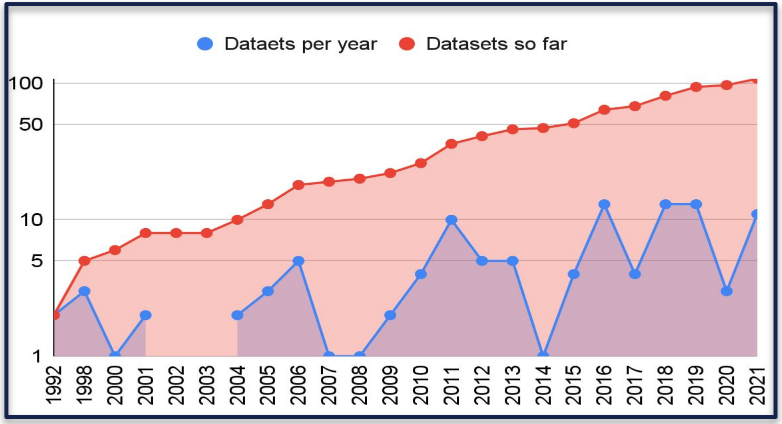
106 task in total

~8 million images

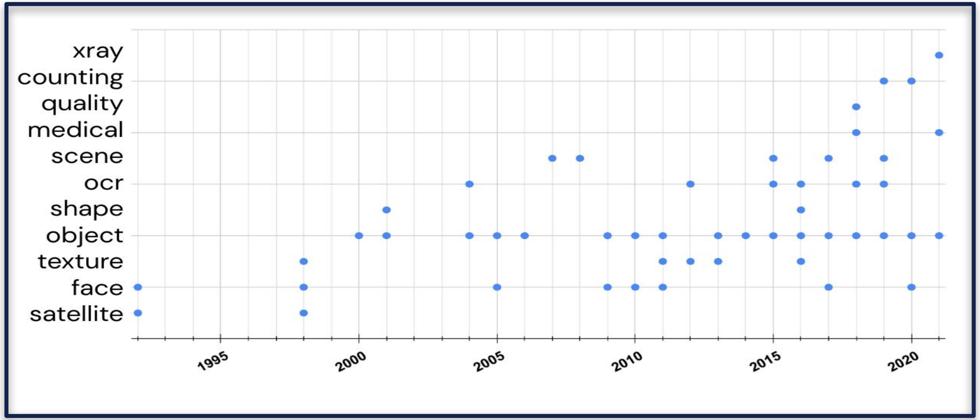
Many domains



Number of datasets



Data types per year



[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023

Assumptions

- Input is a stream of tasks
- Tasks relate to each other in unknown ways
- Learner can revisit old tasks, but cannot peek in the future
- Disk is cheap
- Compute is expensive

[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023

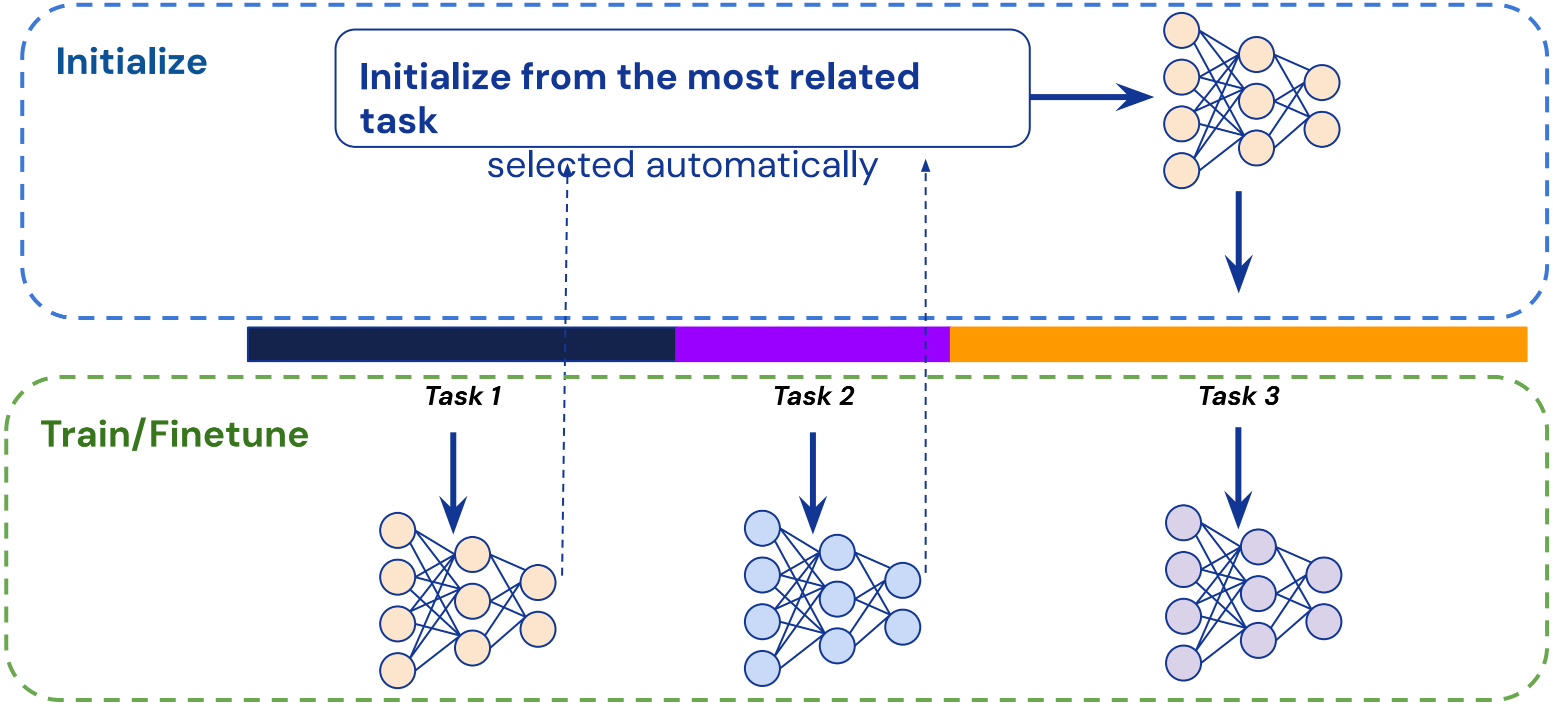
Assumptions

- Input is a stream of tasks
- Tasks relate to each other in unknown ways
- Learner can revisit old tasks, but cannot peek in the future
- Disk is cheap
- Comput

We can train a network (expert) per task.
How to transfer knowledge across experts?

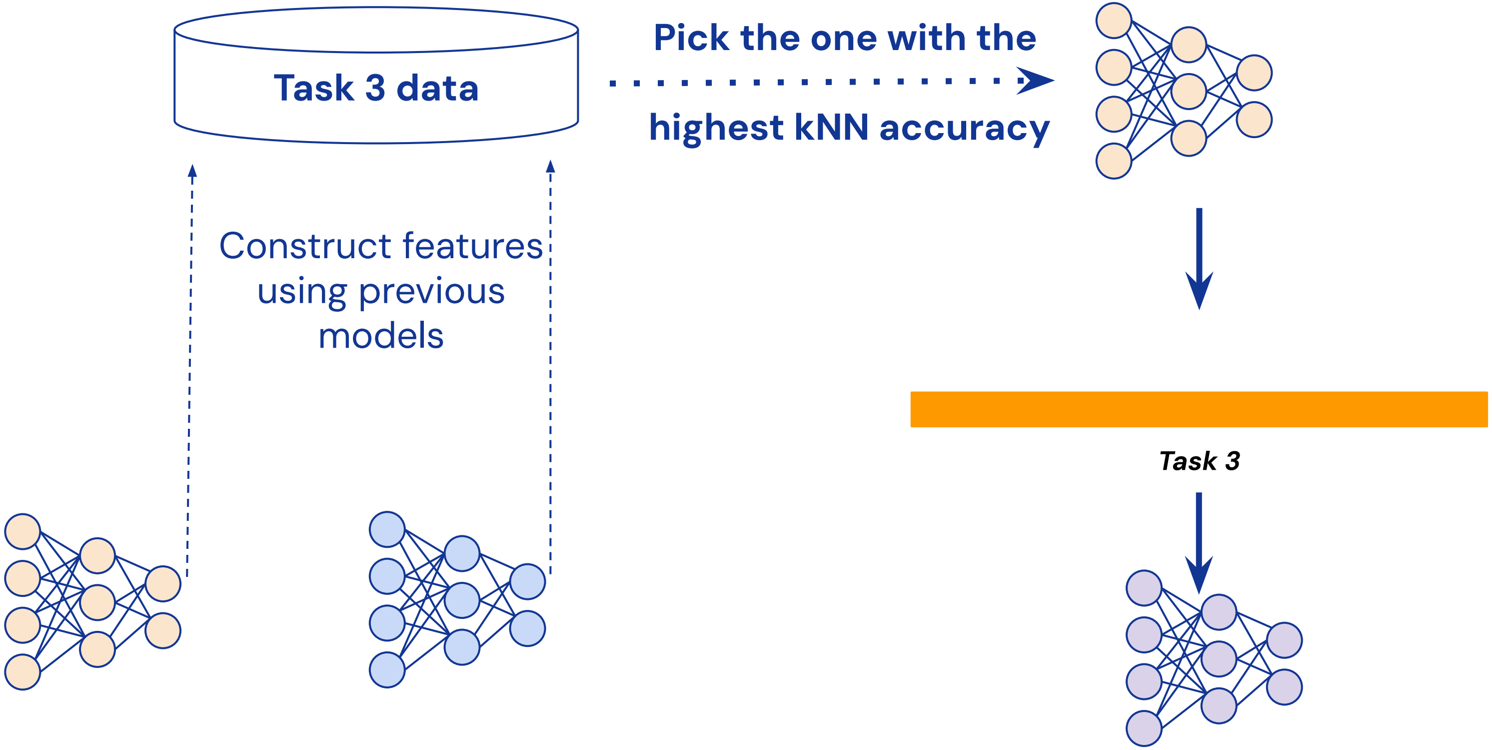
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023

Finetuning from best encountered model



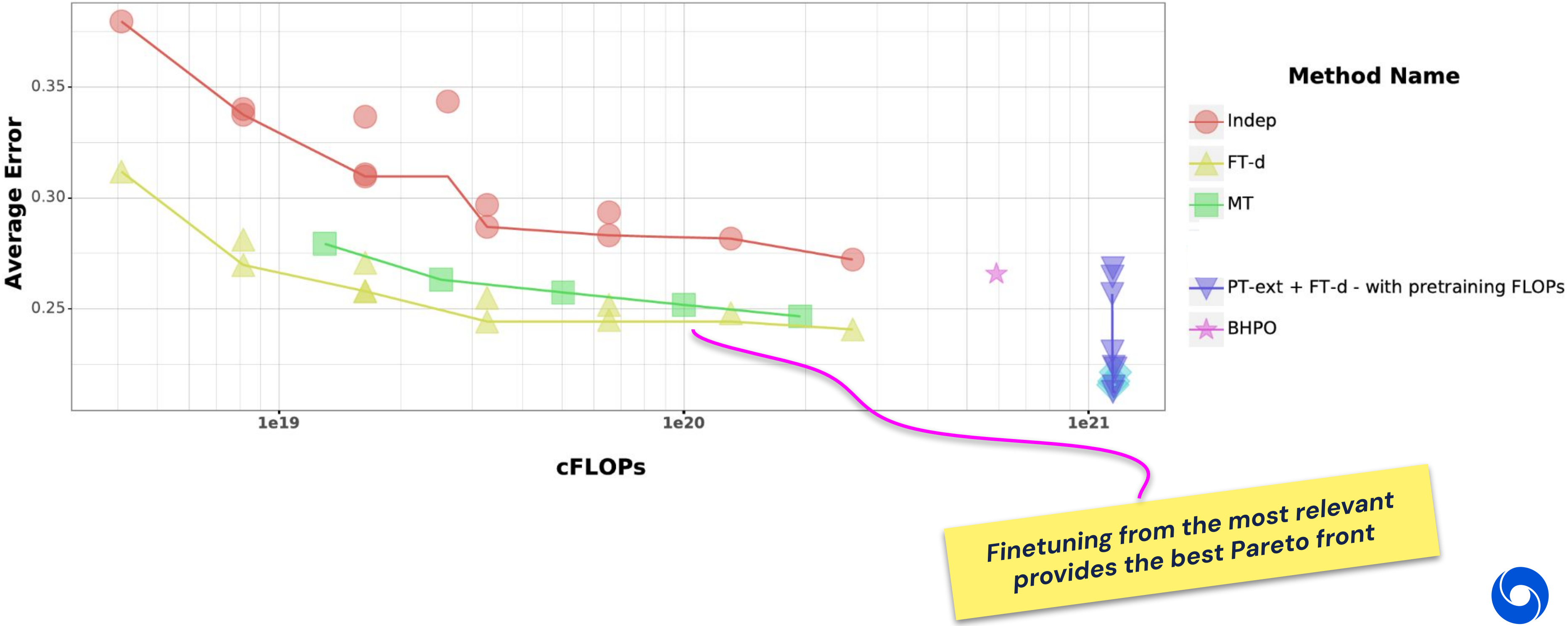
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023

Finetuning from best encountered model



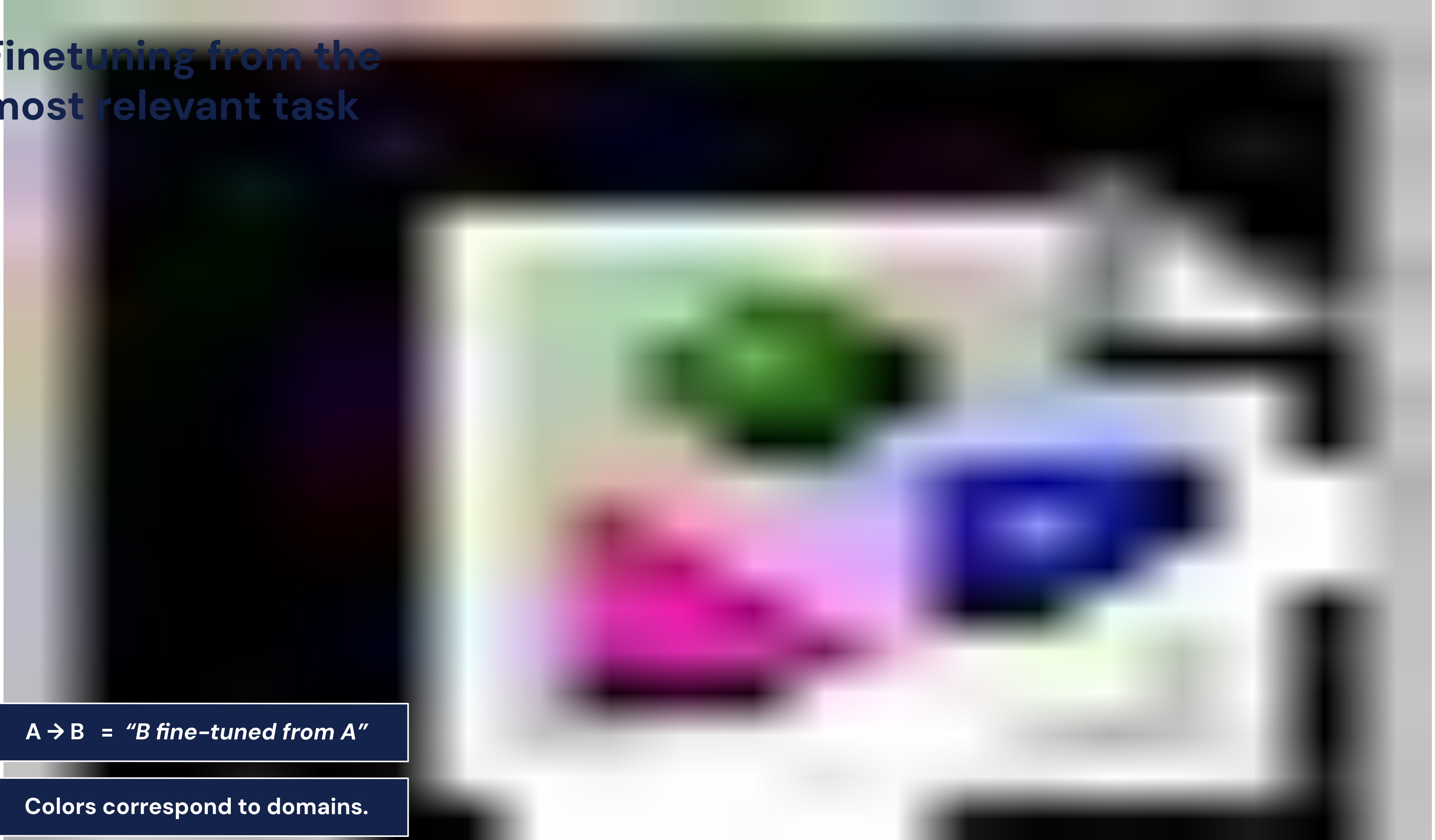
Veniat et al. “Efficient continual learning with modular networks and task-driven priors” ICLR 2021

Pareto Fronts



[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023

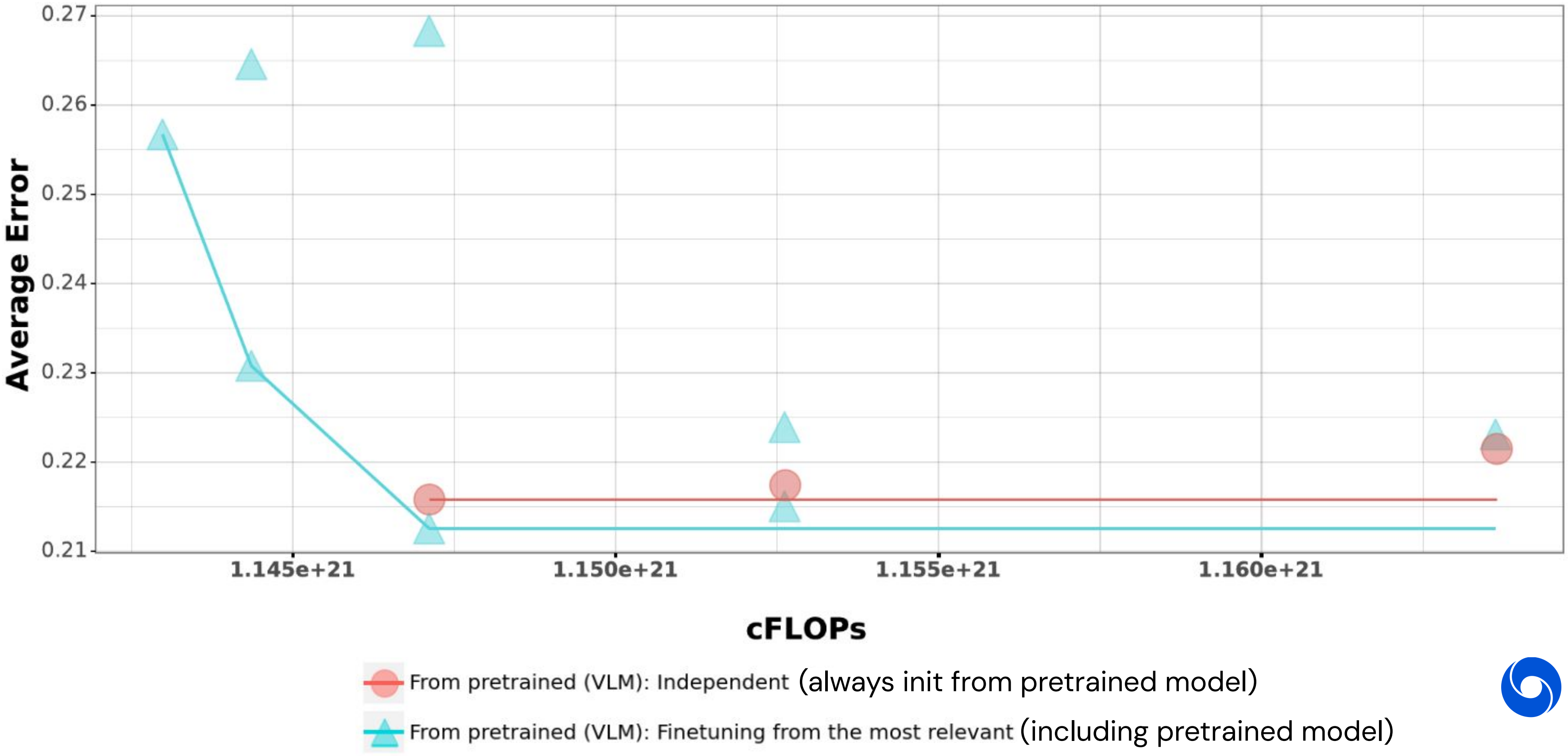
Finetuning from the most relevant task



A → B = *"B fine-tuned from A"*

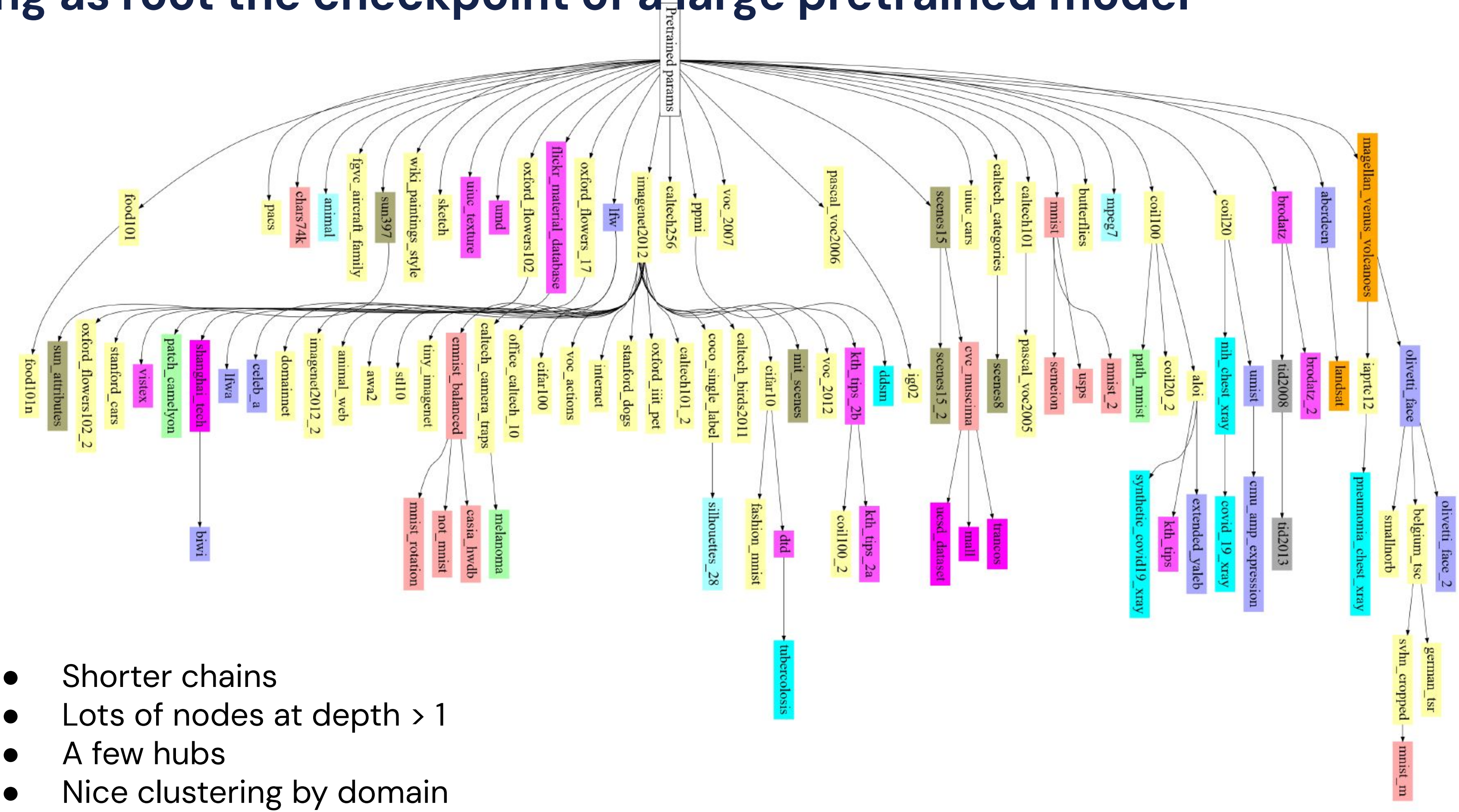
Colors correspond to domains.

Selective Finetuning Combined with Pre-Training



Using as root the checkpoint of a large pretrained model

Public



- Shorter chains
- Lots of nodes at depth > 1
- A few hubs
- Nice clustering by domain

Selective Finetuning

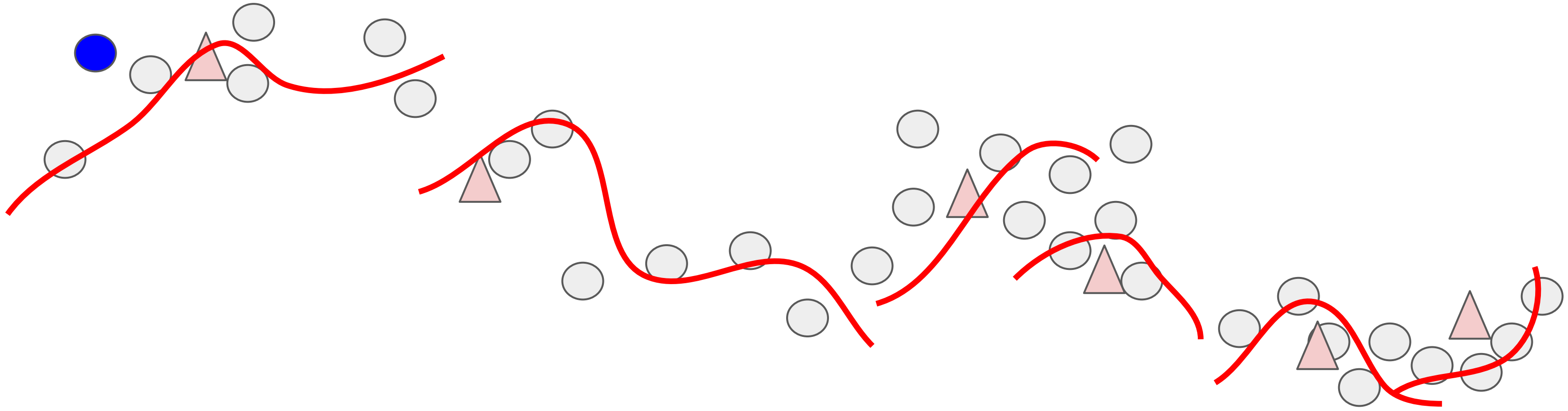
Pros
- Simple.
- Effective.

Cons
- Knowledge transfer only through initialization.
- Sometimes only a subset of parameters are useful.
- Better generalization if we could share parameters across tasks.

Selective finetuning yields a hard MoE which is incrementally built over time.
Gating is not learned, it is determined by the task id (provided at the input).

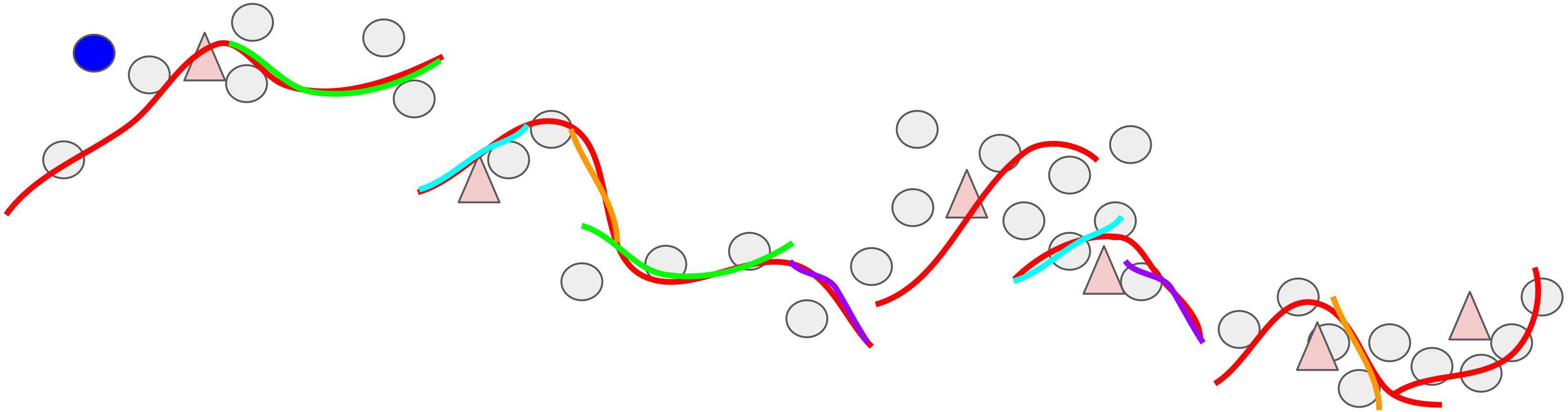
Similar findings in NLP domain:
Fisch et al. "Towards robust and efficient continual language learning" arXiv 2023

flat hard Mixture of Experts



Issue: As we increase the number of experts, eventually the model starts overfitting.

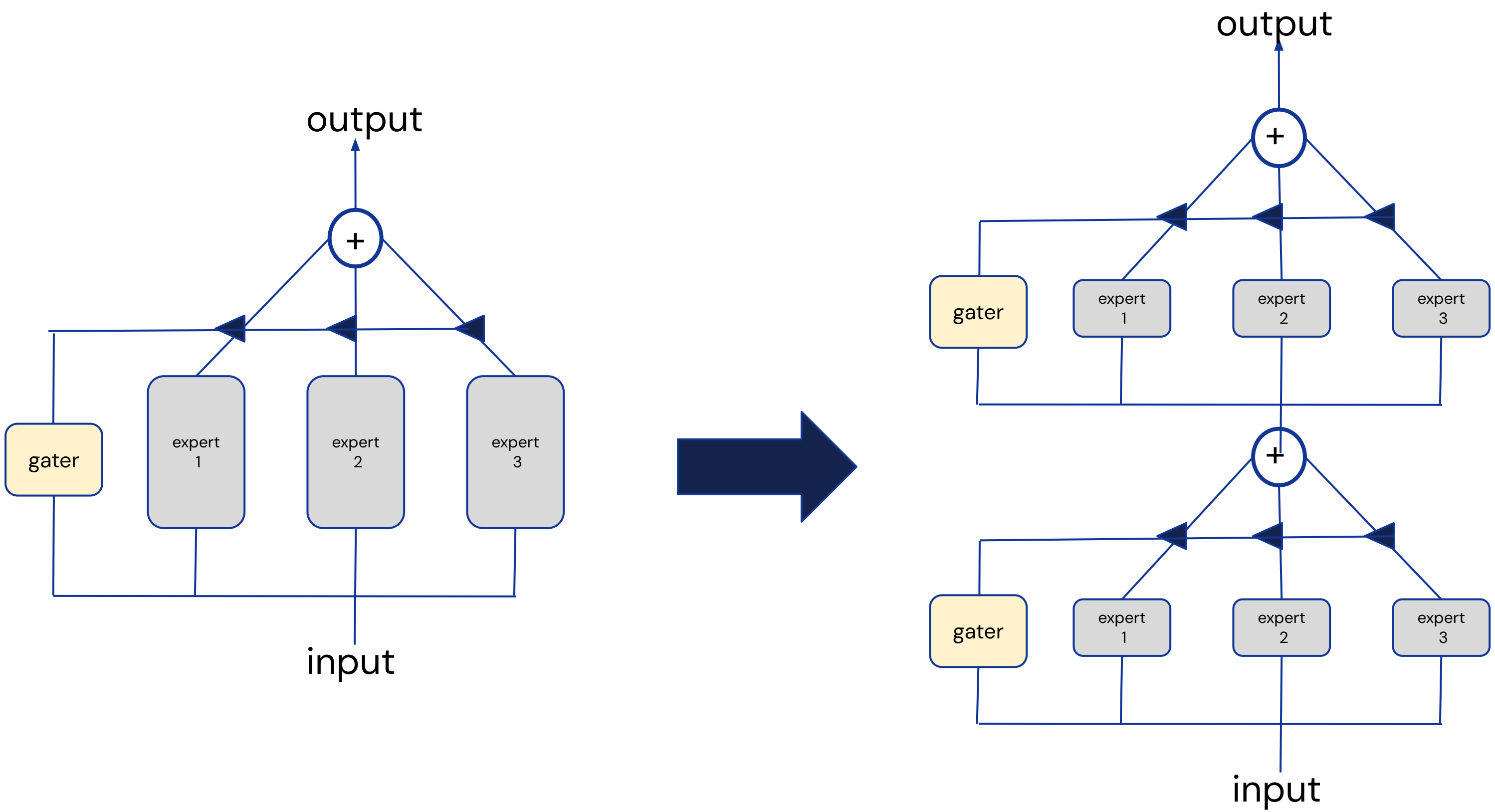
composable mixture of experts



Public

FLAT

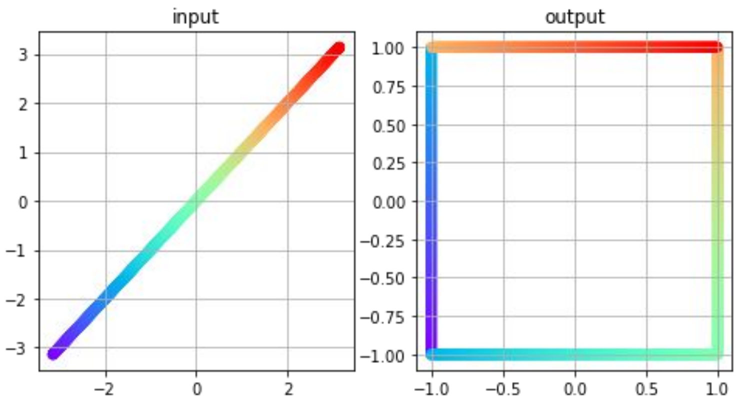
COMPOSABLE



Composable Mixtures

- pros
  - ML efficient as paths share parameters
  - compositional generalization
  - trivial continual learning extensions
- Cons
  - gater is hard to learn
  - engineering complexity
  - not easy to distribute

Colab Demo on composable MoEs



Agenda

- Motivation [15min]
  - Continual Learning
  - Modular Learning
- Modularity via Mixture Models [60min]
  - Detour
  - Flat Hard Mixtures [1, 2]
  - Composable Mixtures [3]
- Conclusions [15min]

References
[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023
[3] Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors ICLR 2021

Setting

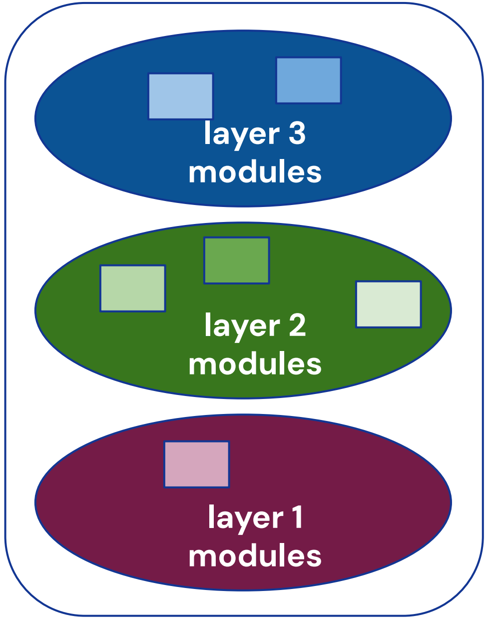
- Input is a stream of tasks
- Tasks relate to each other in unknown ways
- Learner cannot peek in the future
- Learner can load models of old tasks

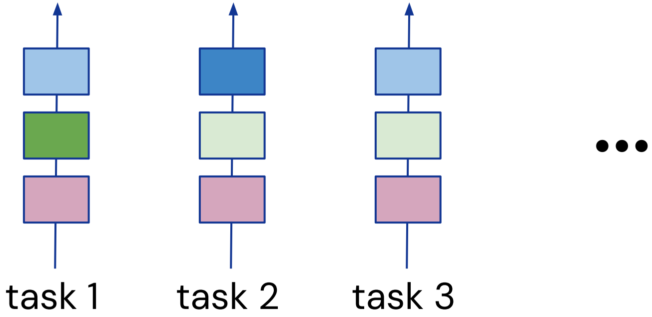
Goals:

- Achieve best average accuracy
- Learn quickly new tasks
- Overall model size does not grow linearly over time

Modular network at time t

Existing pool of modules



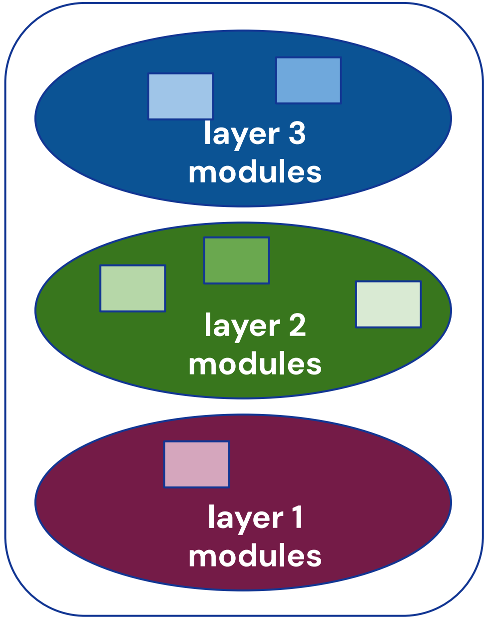


Mixture of experts with gating performed at the task level.

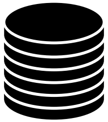
Veniat et al. [Efficient Continual Learning with Modular Networks and Task Driven Priors] (ICLR 2021)

Step 1: Receive new task

Existing pool of modules



Data of new task

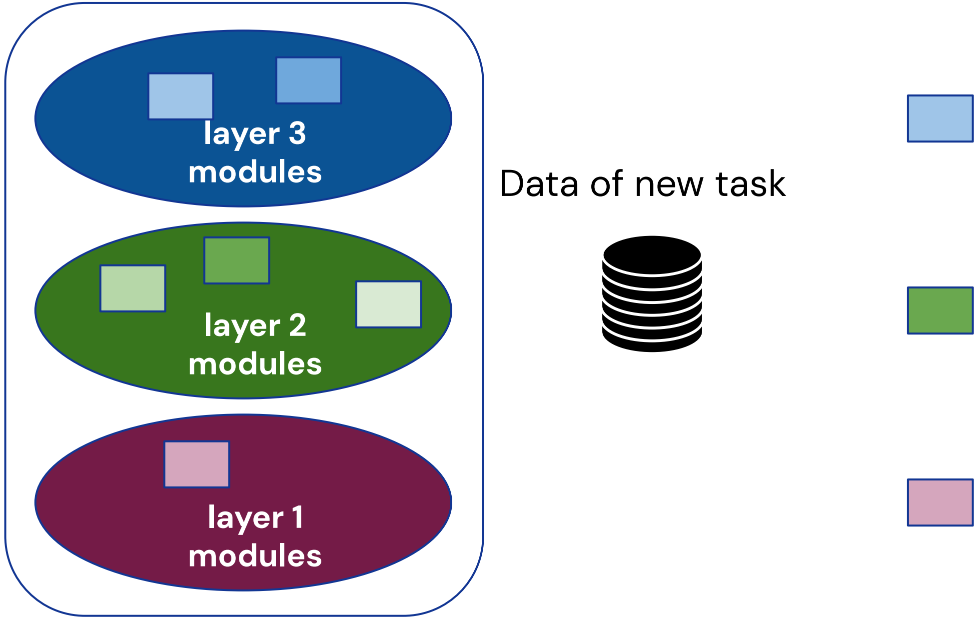


Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)

Step 2: Module retrieval

Existing pool of modules

Module retrieval

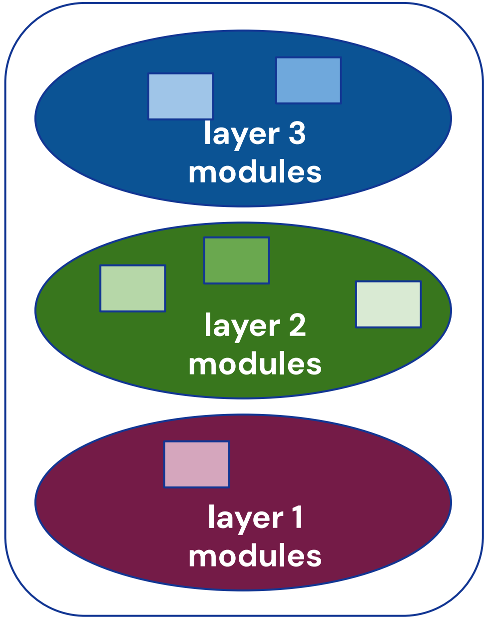


Retrieve most relevant modules at each layer.
E.g.: select modules of networks trained on most related past tasks (determined via kNN).
The retrieval set is a (data-driven) prior.

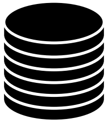
Veniat et al. [Efficient Continual Learning with Modular Networks and Task Driven Priors]() (ICLR 2021)

Step 3: Perturb & Search

Existing pool of modules

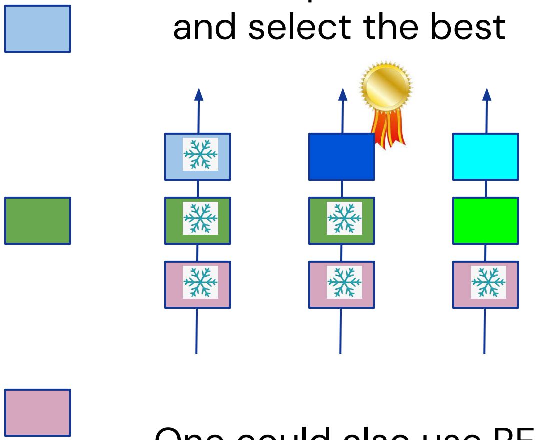


Data of new task



Module retrieval

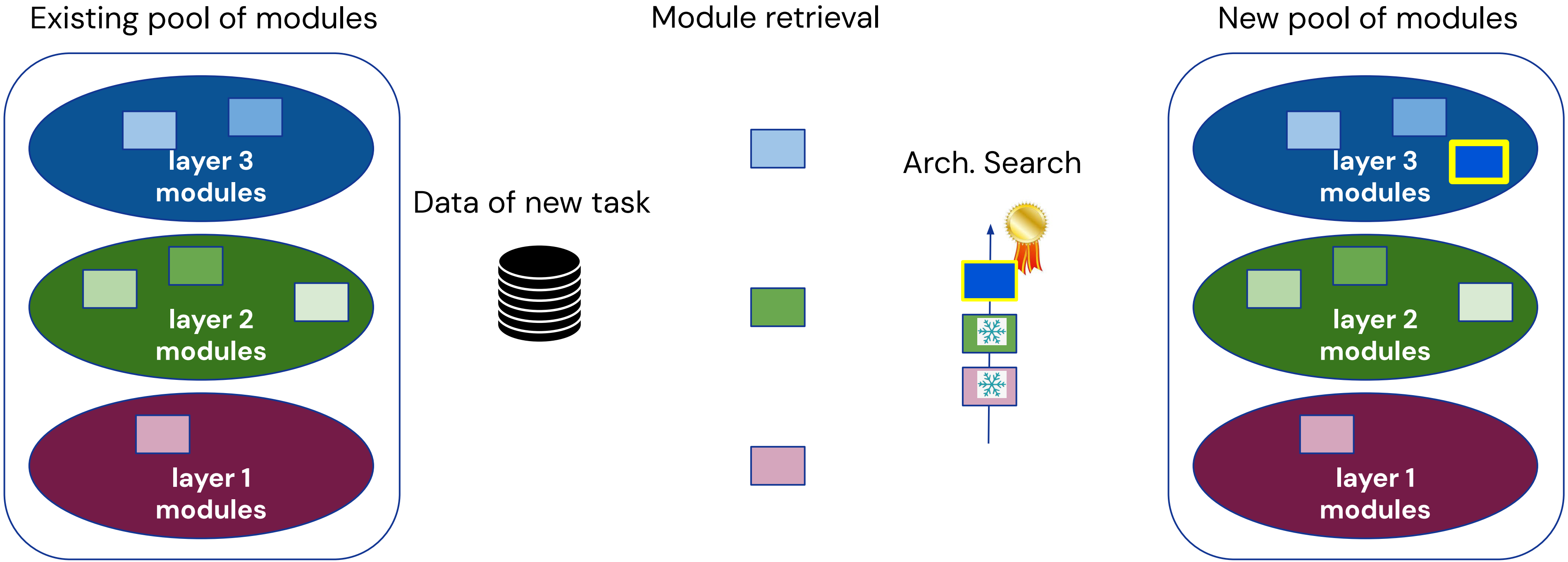
Train in parallel k variants
and select the best



One could also use REINFORCE to
train the k variants all at once.

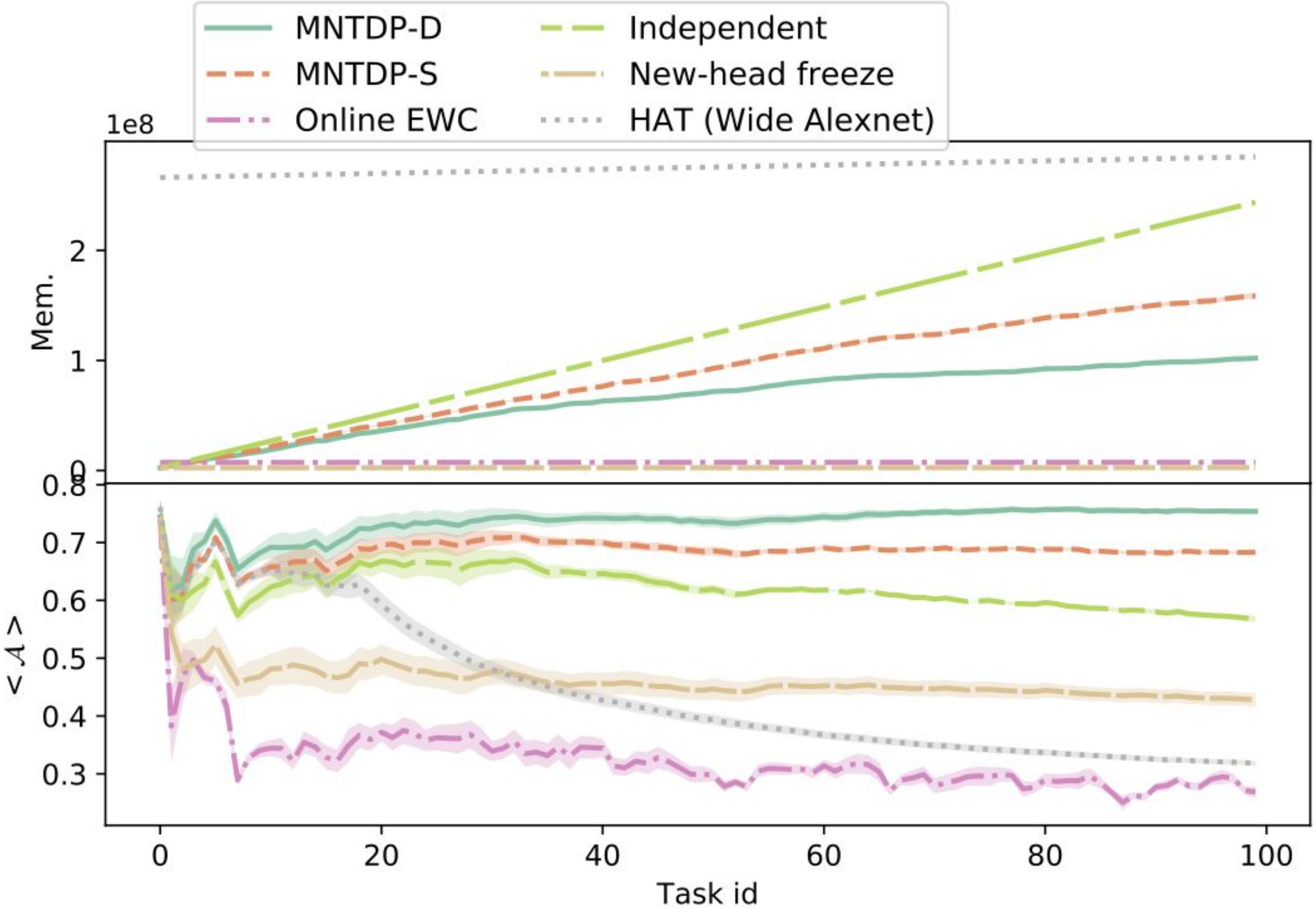
Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)

Step 4: Pool expansion



Veniat et al. [Efficient Continual Learning with Modular Networks and Task Driven Priors]() (ICLR 2021)

Results on $\mathcal{S}^{\text{long}}$ (toy stream with 100 tasks)



MNTDP achieves highest average accuracy while growing sub-linearly in memory.

Modular Networks with Task Driven Priors

- General idea:
  - Retrieve most similar modules
  - Perturb & learn
  - Expand existing pool with newly trained modules
- Size of search space defines efficiency/efficacy trade-off.
- Because of growth, model is not going to lose plasticity over time.

Similar to selective finetuning chains, but operating at the level of modules as opposed to entire networks.

Open questions

- Scaling up
- Efficient architecture search
- How to learn a good initial set of modules

Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors (ICLR 2021)

Agenda

- Motivation [15min]
  - Continual Learning
  - Modular Learning
- Modularity via Mixture Models [60min]
  - Detour
  - Flat Hard Mixtures [1, 2]
  - Composable Mixtures [3]
- Conclusions [15min]

References
[1] Gross et al. "Hard mixture of experts for large-scale weakly supervised vision" CVPR 2017
[2] Bornschein et al. NEVIS'22 Benchmark JMLR 2023
[3] Veniat et al. Efficient Continual Learning with Modular Networks and Task Driven Priors ICLR 2021

Public

Success

Advice: Co-design what's realizable in the next 5 years.

Examples:
- 90s-00s: SVMs & CPUs
- 2010-2020: CNNs & GPUs

Conjecture:
- Data will never be exhausted
- We need to scale further
- No single entity will have enough compute: we must cooperate
- My bet is that we need to co-design around a distributed decentralized system

Conclusions

- Learning is about dealing with trade-offs.
- Machine learning is continual.
- Continual learning aims at improving efficiency via knowledge transfer.
- (Most) ML is continual in a naïve and poorly automated way.
- As models get bigger, it is more important than ever to make them more efficient.
- Conjecture: Modularity is key to scaling, efficiency, and continual learning.

On Intelligence

Intelligence must arise when there are suitable constraints.

What are the constraints?

- number of examples?
- compute?
- memory?
- time?
- ?

Continual learning is an instance of multi-objective learning.
Don’t tell me which method is most accurate.. but which one strikes the best trade-off between efficiency and accuracy.

Legg et al. Universal Intelligence arXiv 2007

On Intelligence 

Intelligence must arise when there are suitable constraints.

What are

Time has come to go beyond Empirical Risk Minimization!

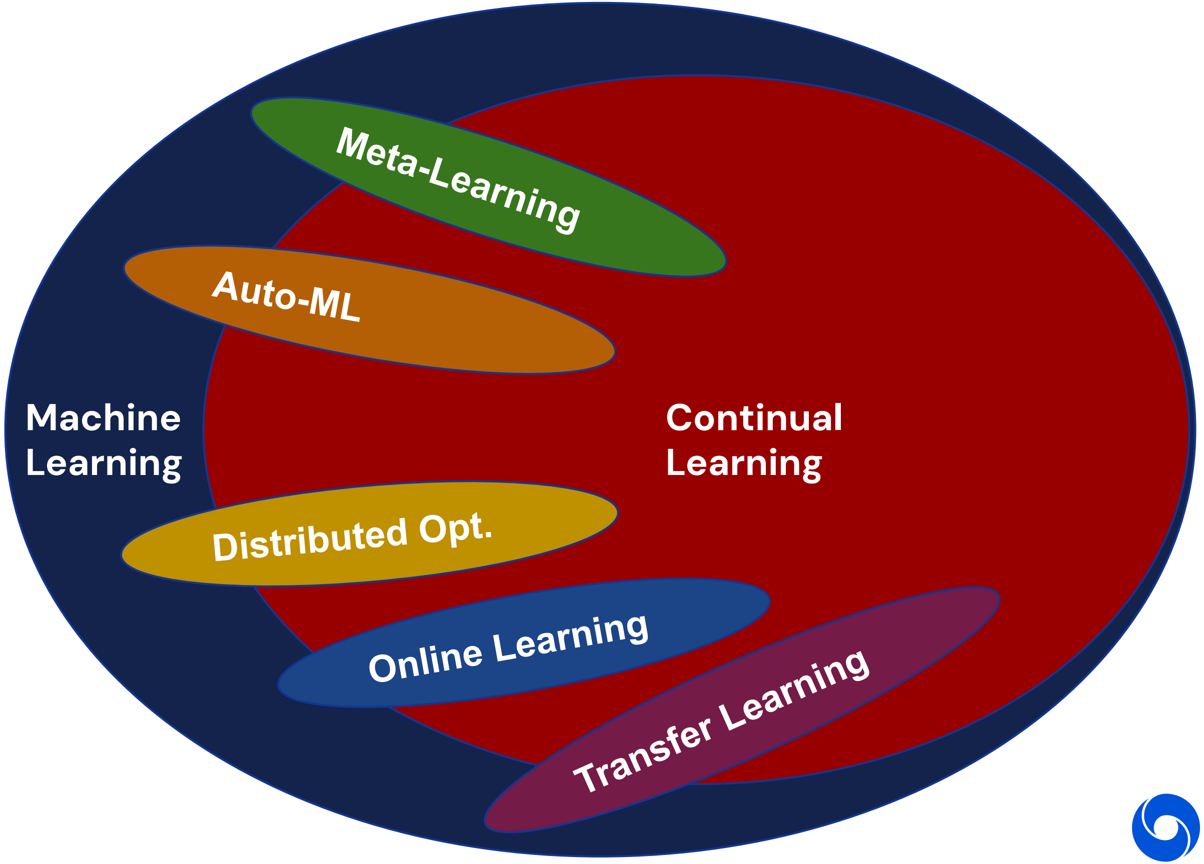
- ?

Continual learning is an instance of multi-objective learning.
Don’t tell me which method is most accurate.. but which one strikes the best trade-off between efficiency and accuracy.

Legg et al. [Universal Intelligence] arXiv 2007

Relation of CL to Other Fields

Most of ML is continual!
CL needs input from sub-fields like meta-learning and auto-ml.
Vice-versa CL can lift these subfields and make them more practical.



Some Open Research Questions

- How to contribute to the development of large-scale learning without access to huge computational resources?
- Learning is about striking trade-offs: How to formalize and derive practical algorithms or architectures?
- What constraints are meaningful in practice?
- How to retain efficiency as we scale up?
- How to modularize in a distributed way?
- How to grow from small to big?
- How to do efficient meta-learning?
- How do we cross-validate in a never-ending learning setting?
- How to add/update/remove knowledge?
- What’s the role of memory?

Questions?

ranzato@google.com

https://ranzato.github.io/