Learning from Non-Stationary Datasets

Marc'Aurelio Ranzato

Facebook AI Research → DeepMind

ranzatomr@gmail.com

Princeton Deep Learning Theory Summer School — 27 July 2021

Goal: Efficient Learning

What

Statistical Efficiency

Computational Efficiency

How

Goal: Efficient Learning

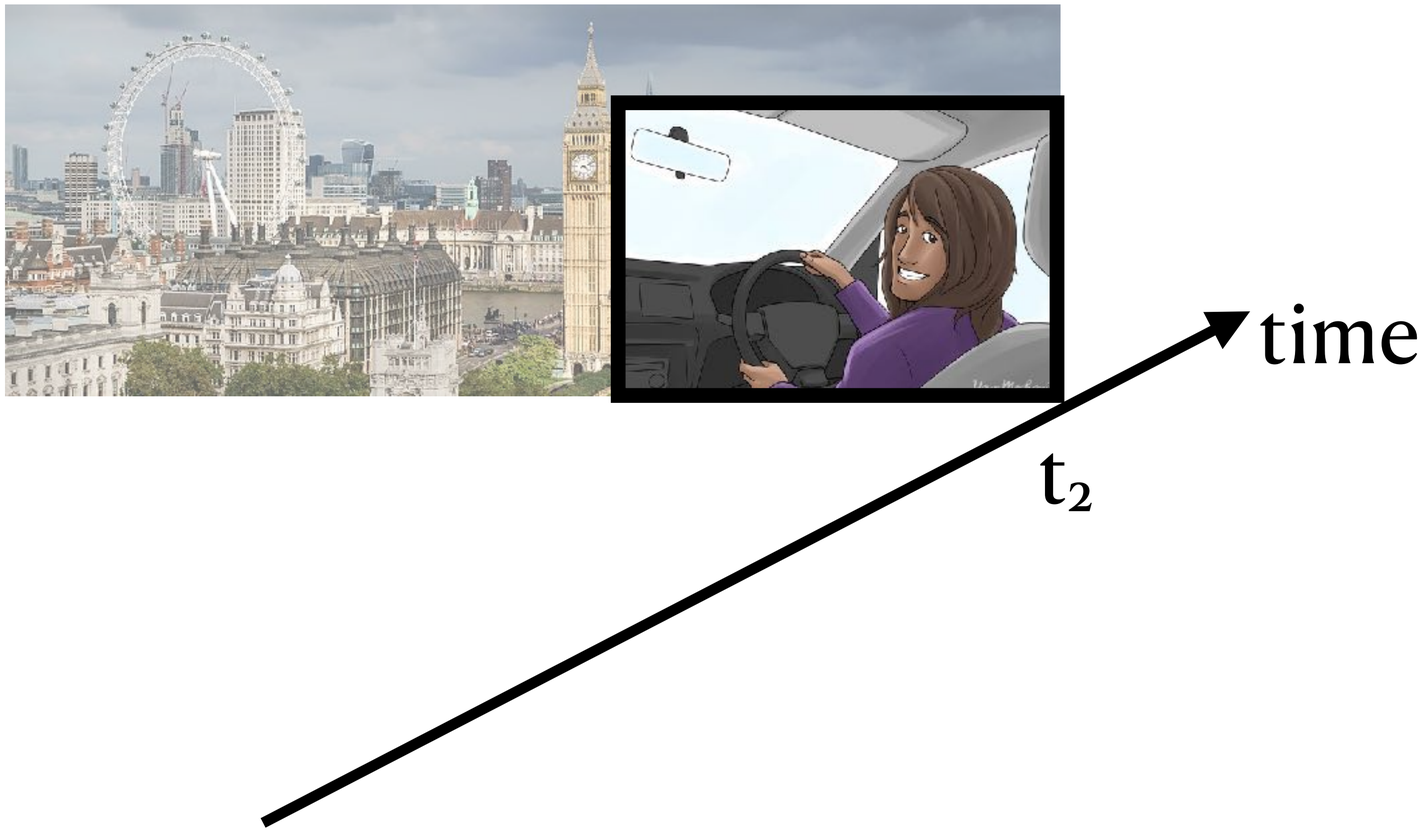
What

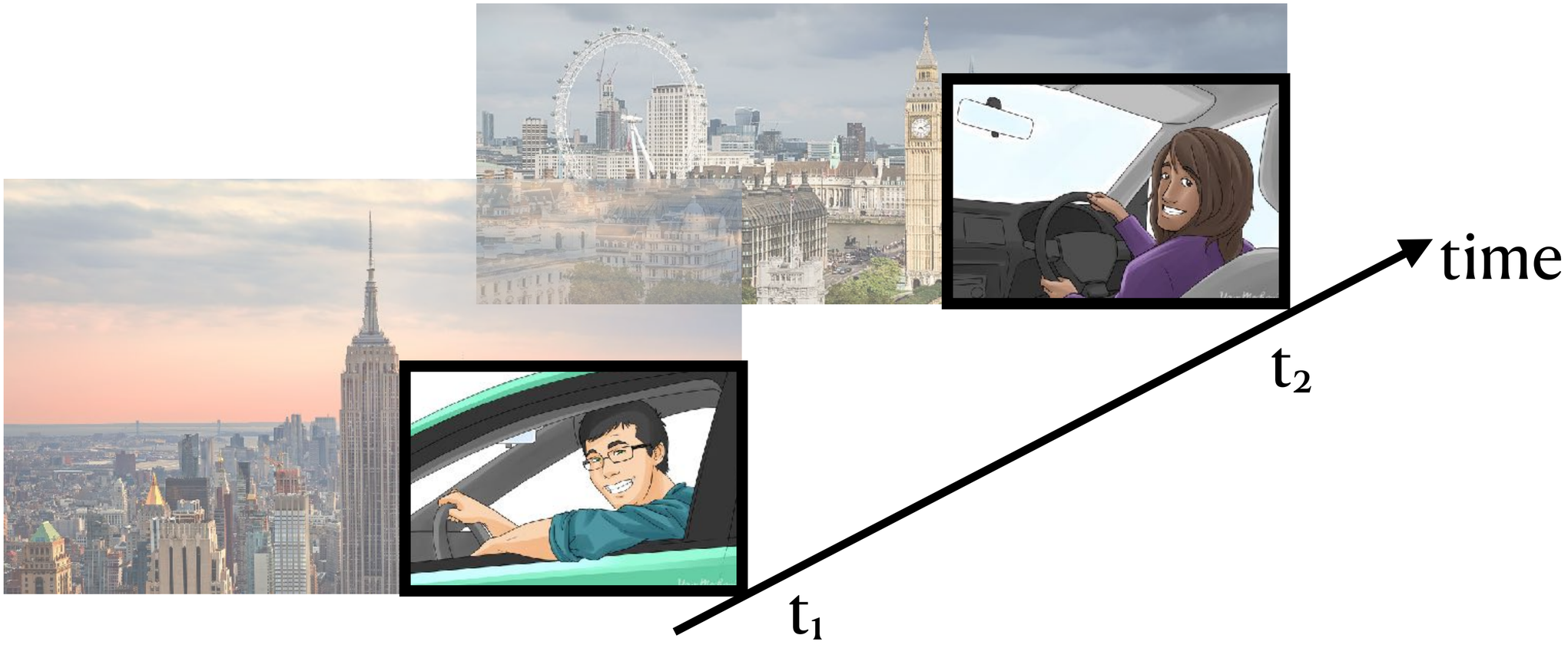
Statistical Efficiency

Computational Efficiency

How

Data Driven Priors


time
t_2



Leveraging prior experience to build a prior and transfer knowledge.

Open questions:

a) how to learn from a non-stationary stream of data?
b) How to retain plasticity through time?
c) How to avoid interference?
d) What is knowledge and how to transfer it?
e) How to best generalize and optimize in this setting?

Goal: Efficient Learning

What

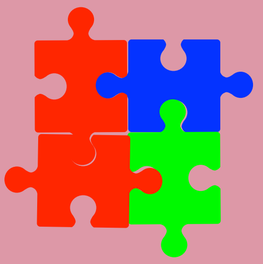
Statistical Efficiency

Computational Efficiency

How

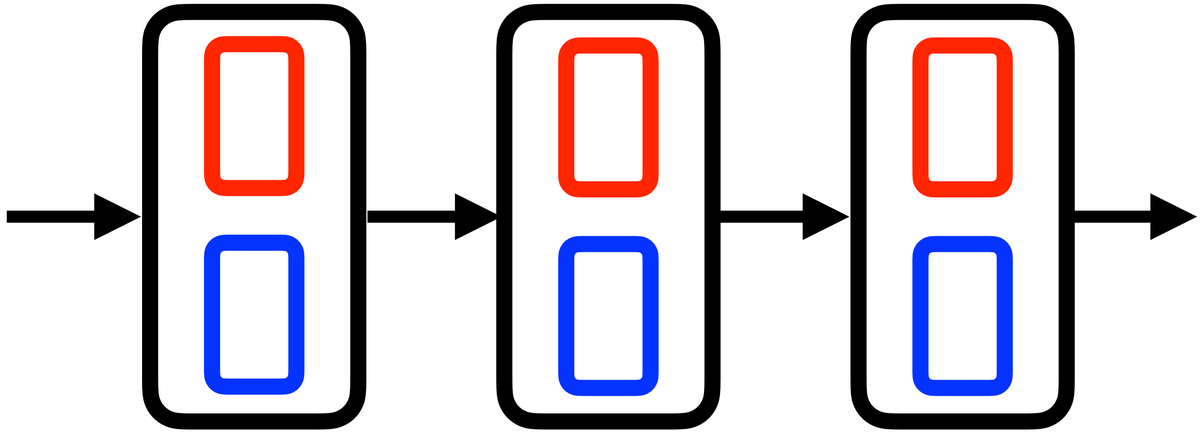
Data Driven Priors

Compositionality
(via modularity)



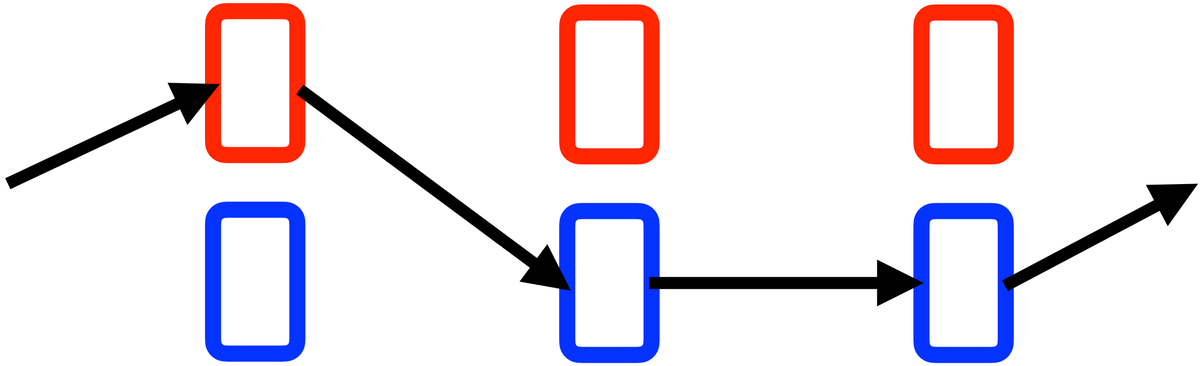
- Perceive
- Run
- Kick
- Communicate
- Plan

Compositionality via modularity:



- Perceive
- Run
- Kick
- Communicate
- Plan

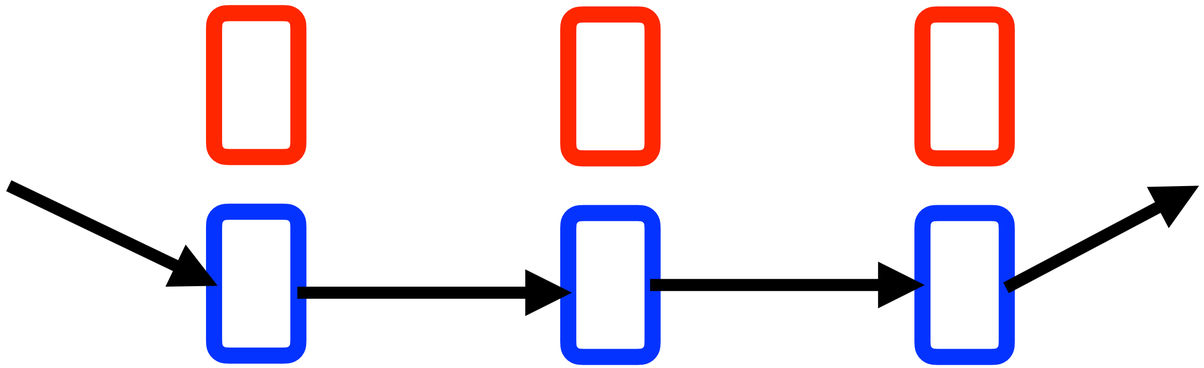
Compositionality via modularity:



Task 1: playing soccer during the day

- Perceive
- Run
- Kick
- Communicate
- Plan

Compositionality via modularity:



Task 2: playing soccer at night

Open questions:

a) What tasks are compositional?
b) How to (learn to) decompose tasks into sub-tasks?
c) How to modularize computation? Implicit or explicit?
d) How to learn modular architectures?
e) How to grow knowledge over time?

Goal: Efficient Learning

What

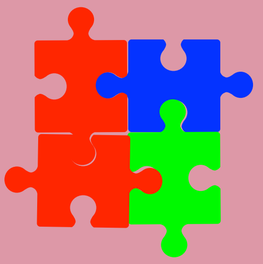
Statistical Efficiency

Computational Efficiency

How

Data Driven Priors

Compositionality
(via modularity)



Outline

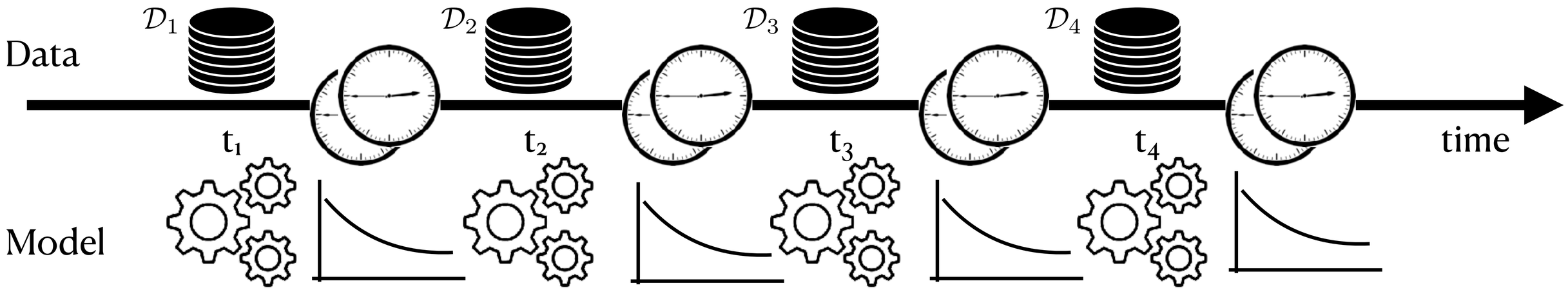
- **Anytime** Learning with Modular Architectures
  - Data arrives in large mega-batches over time
  - Metrics, Benchmarks & Models
  - Representative experiments

L. Caccia, J. Xu, M. Ott, L. Denoyer, M. Ranzato. On Anytime Learning at Macroscale, arXiv 2021

- **Continual** Learning with Modular Architectures
  - Tasks arrive in sequence. Each task has its own distribution.
  - Metrics, Benchmarks & Models
  - Representative experiments

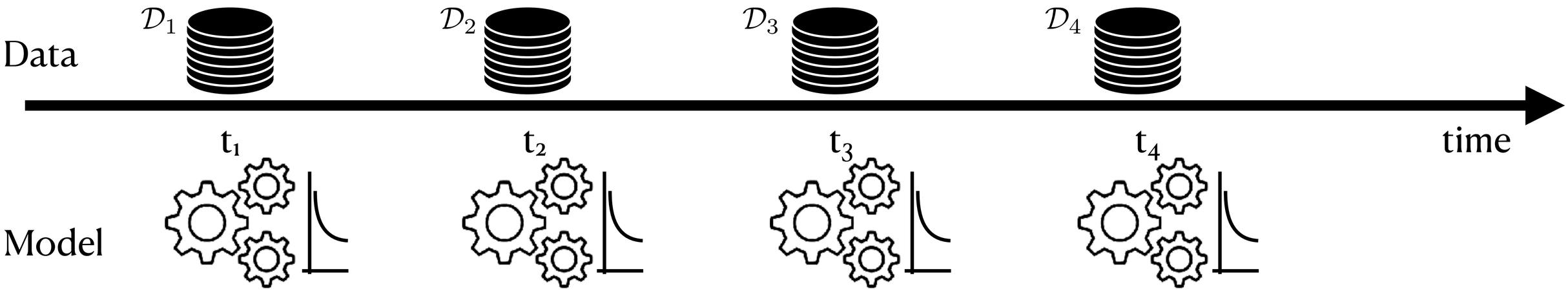
T. Veniat, L. Denoyer, M. Ranzato. Efficient Continual Learning with Modular Networks and Task-Driven Priors, ICLR 2021

Learning Framework



Typically, we assume the rate at which the data arrives matches the rate at which the model can update its parameters.

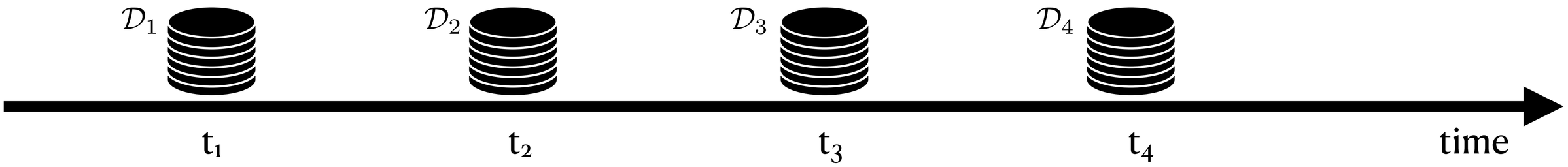
Learning Framework



In practice, the rate at which the data arrives is often **slower** than the rate at which the model can update its parameters.

Warning: if model does multiple passes over each mega-batch, then data is not i.i.d. anymore.

Learning Framework

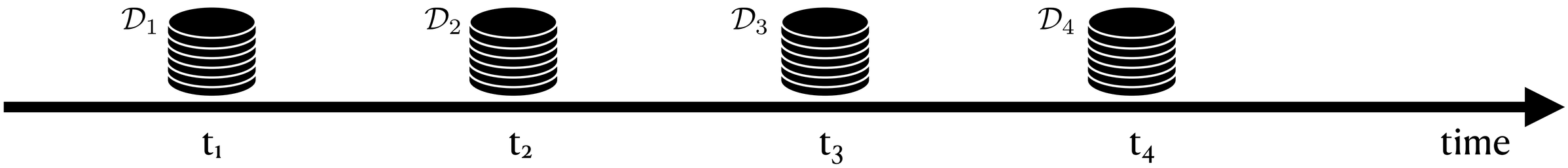


$\mathcal{D}_i$: Dataset (mega-batch) received at the i-th time step. it consists of several mini-batches. All datasets are sampled from the same distribution.
We assume that the time to update the model is negligible compared to time interval between two consecutive mega-batches.

Model Desiderata:

- Low error at any point in time.
- Error rate decreases over time.
- Compute & memory efficiency.

Learning Framework



$\mathcal{D}_i$: Dataset (mega-batch) received at the i-th time step. it consists of several mini-batches. All datasets are sampled from the same distribution.
We assume that the time to update the model is negligible compared to time interval between two consecutive mega-batches.

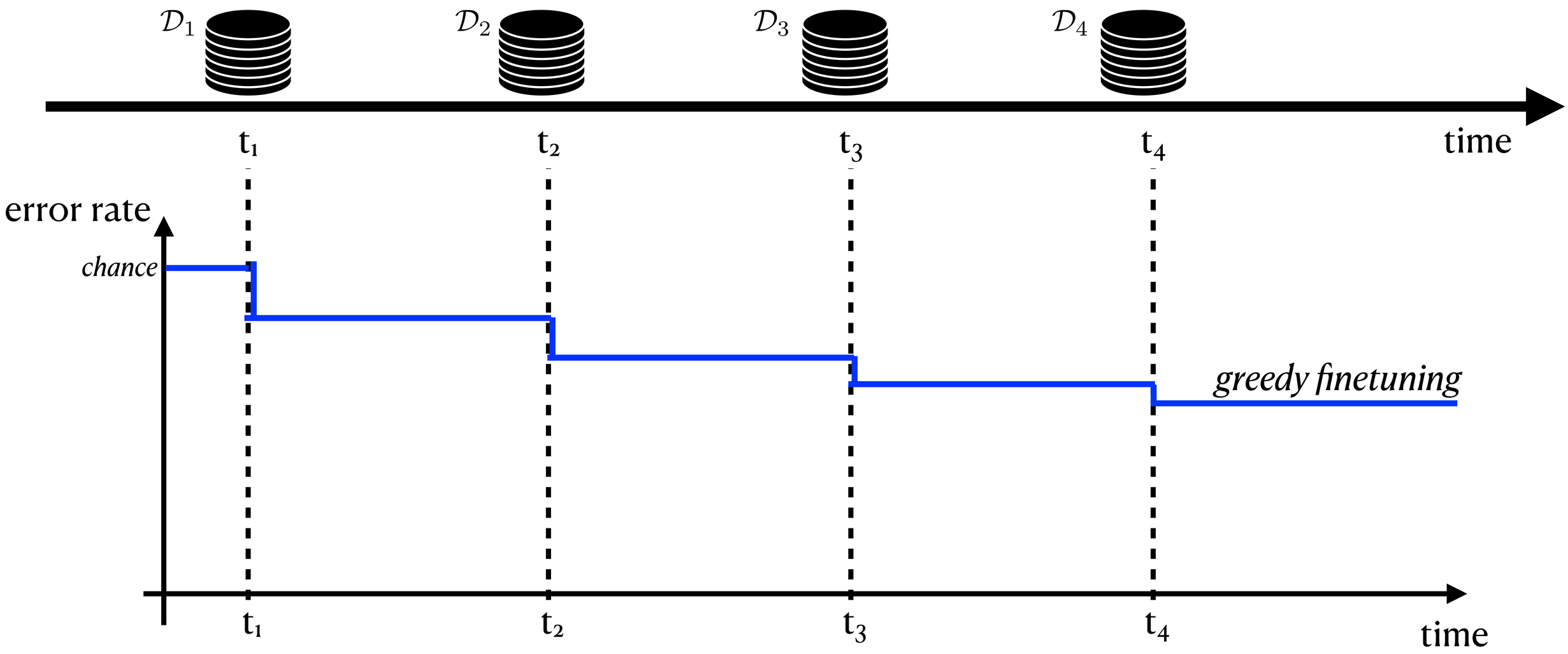
Model Desiderata:

- Low error at any point in time.
- Error rate decreases over time.
- Compute & memory efficiency.

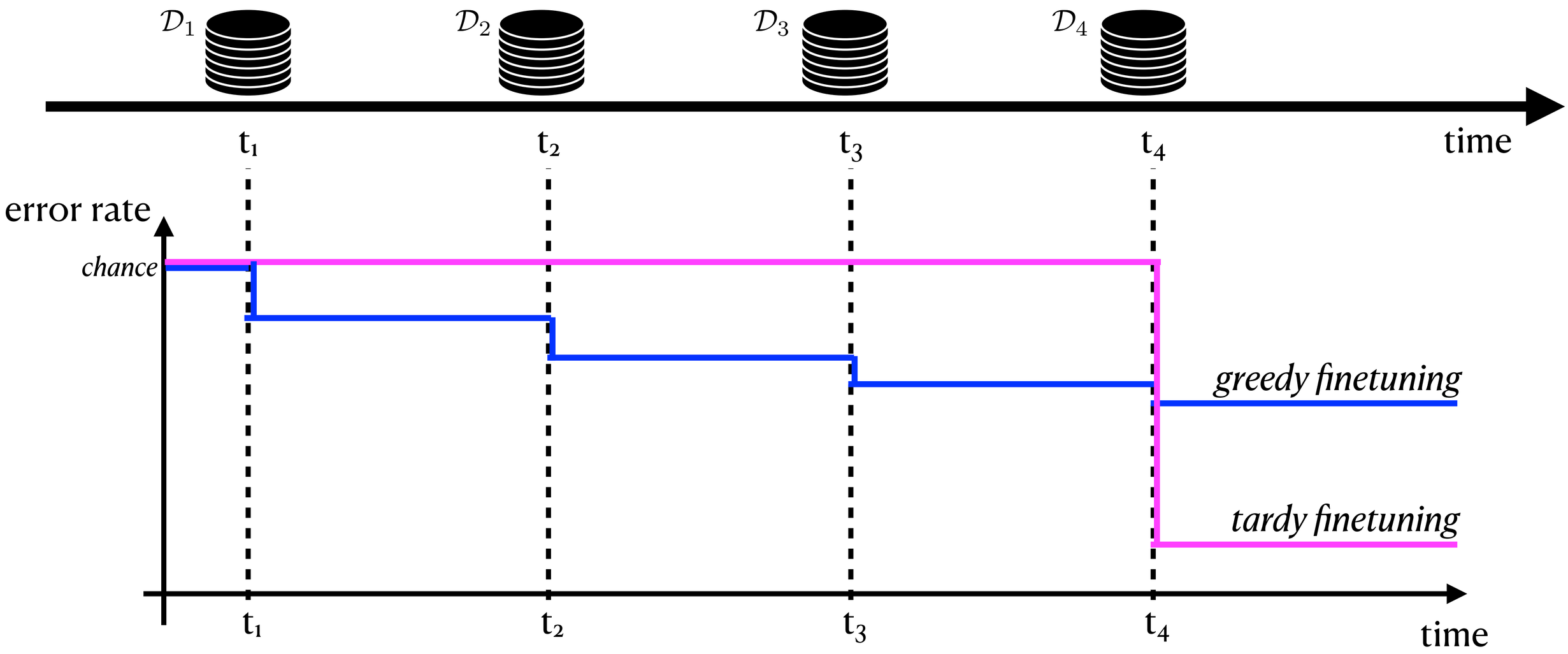
Anytime Learning at Macro-Scale (ALMA)

Grefenstette and Ramsey. Approach to anytime learning. ICML 1992

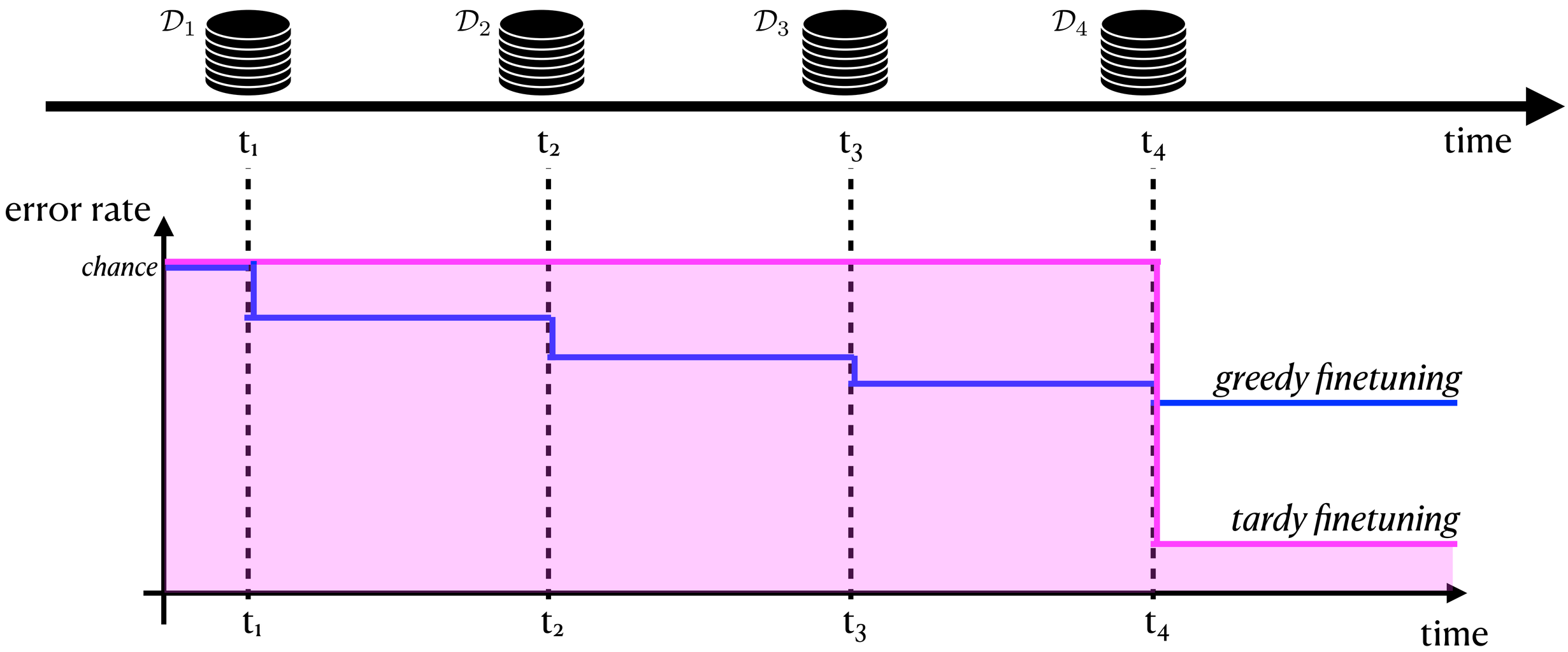
Anytime Learning at Macro-Scale: Models



Anytime Learning at Macro-Scale: Models



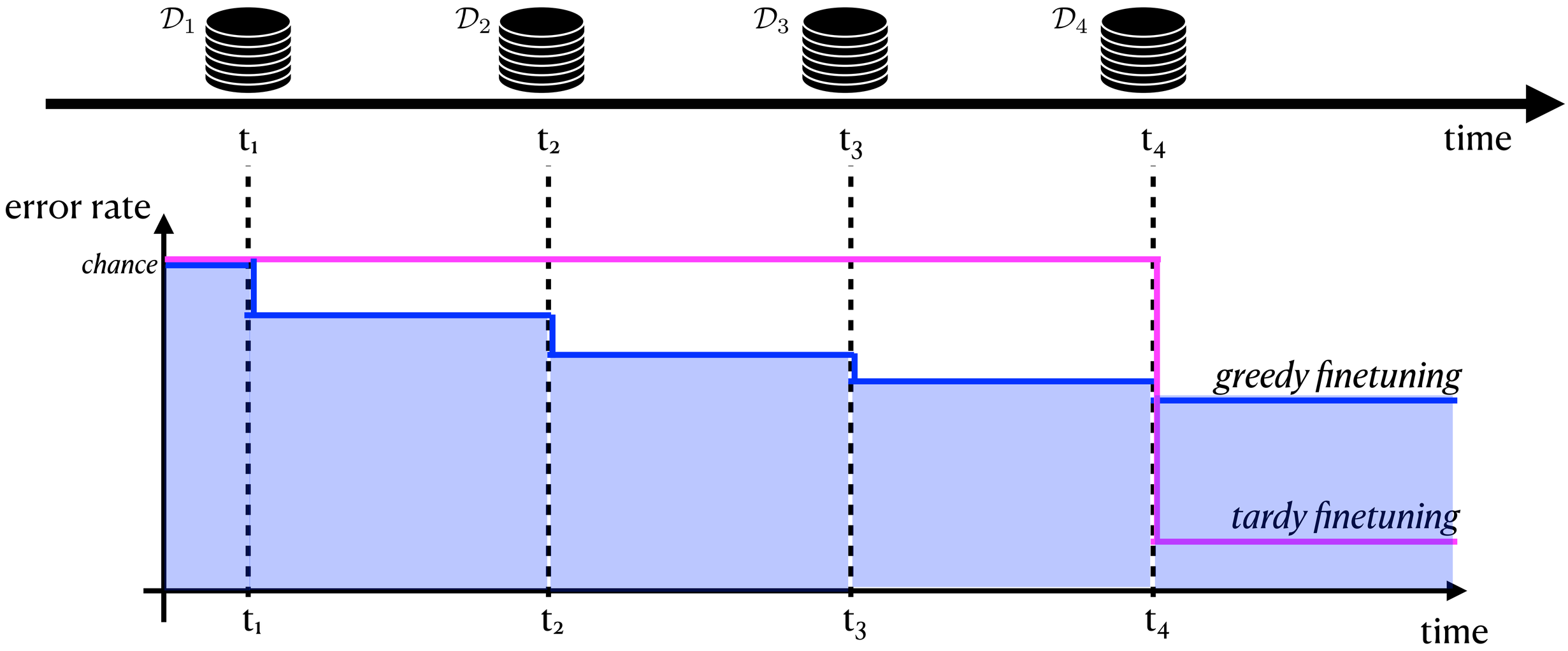
Anytime Learning at Macro-Scale: Models



Area under the learning curve provides metric of performance over the whole learning trajectory.

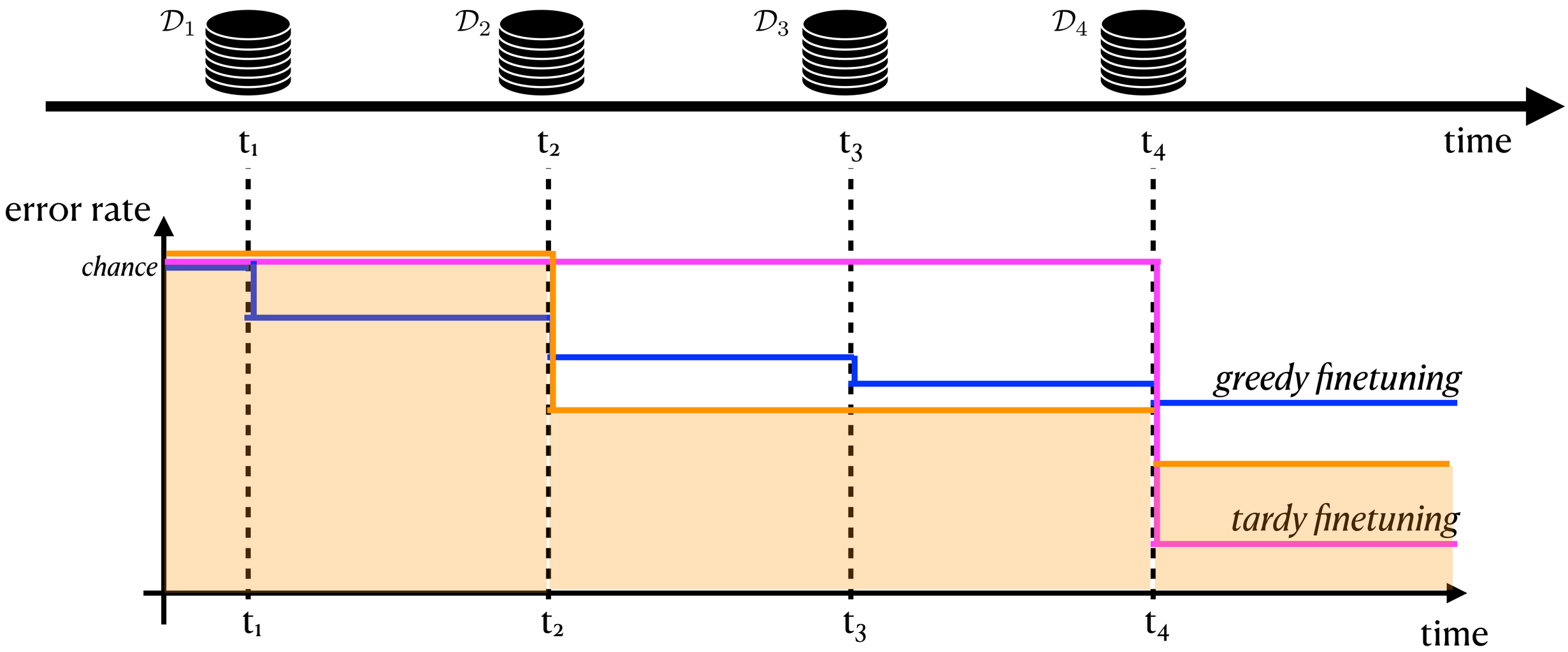
Liu et al. Towards automated deep learning: Analysis of the autodl challenge series 2019. PMLR 2020

Anytime Learning at Macro-Scale: Models



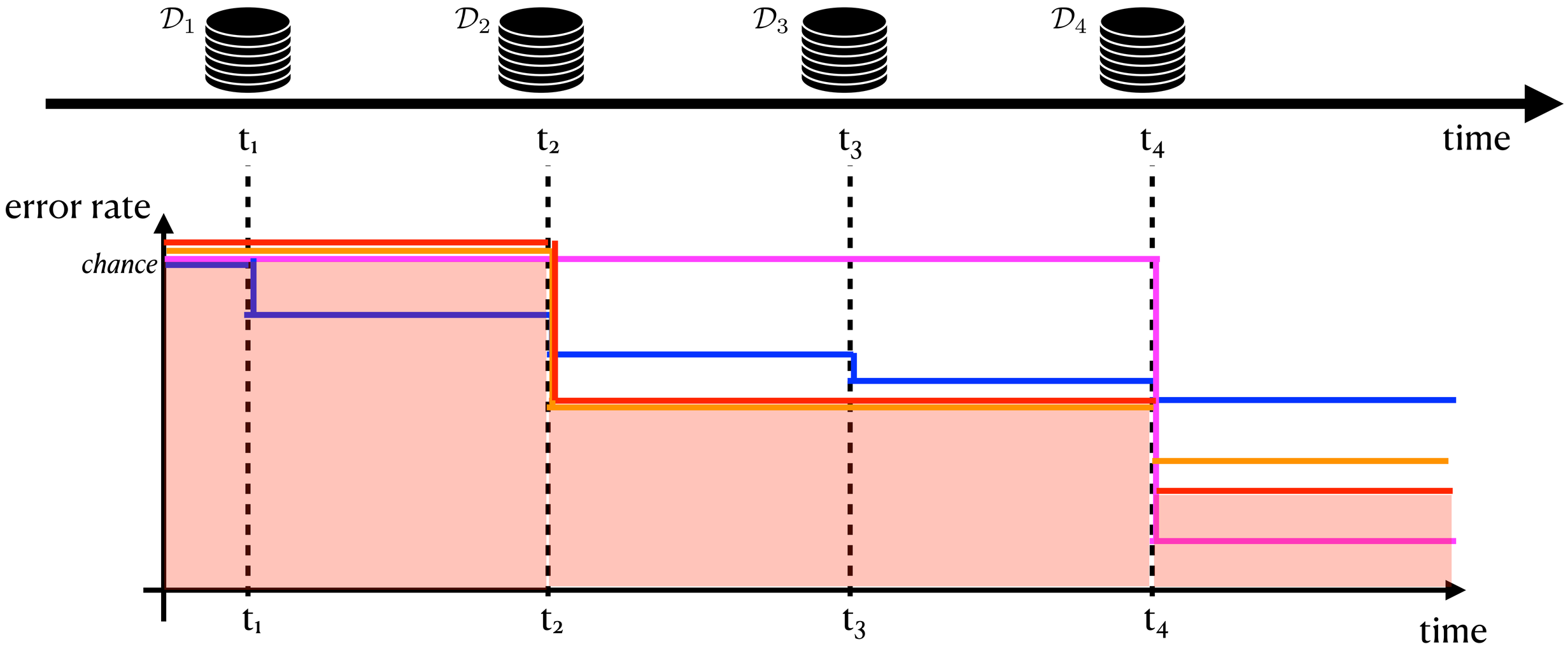
Area under the learning curve provides metric of performance over the whole learning trajectory.

Conjecture #1



Intermediate update rate.

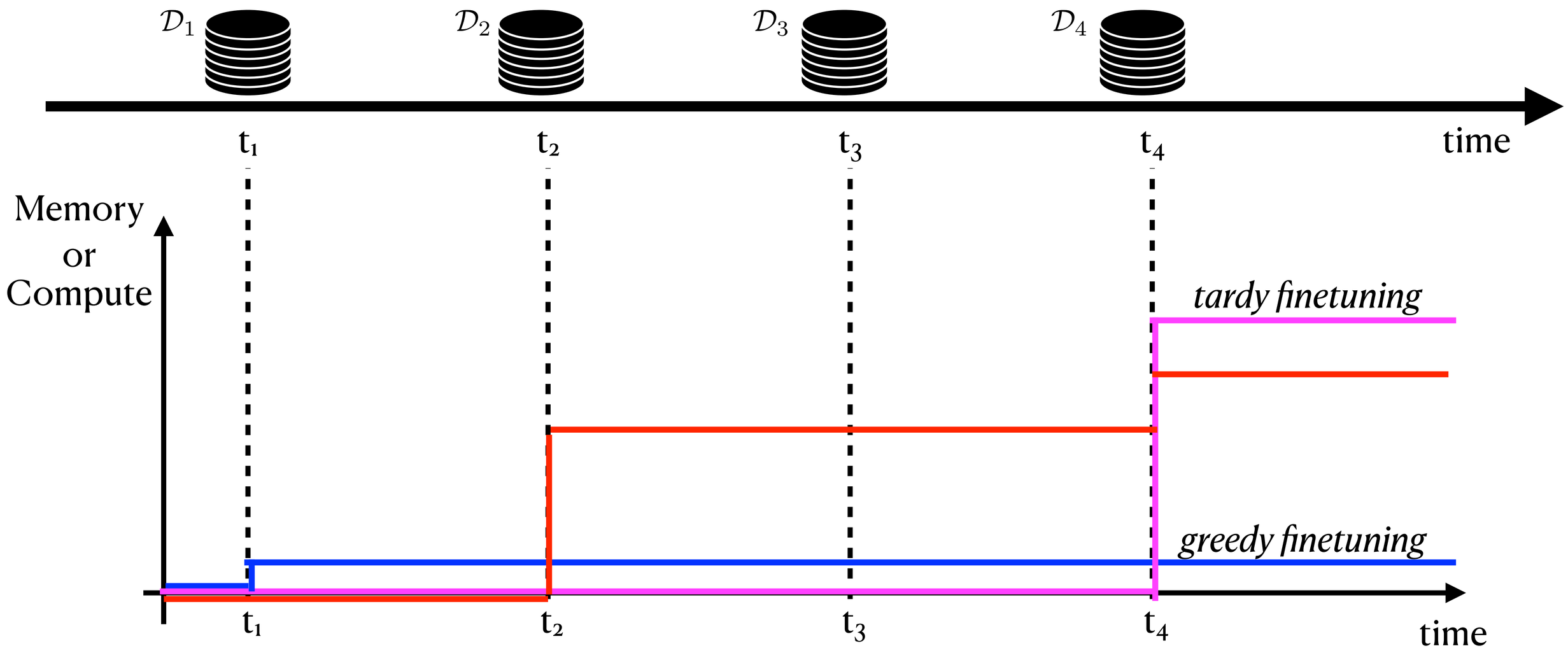
Conjecture #2



Intermediate update rate.

Growing capacity over time.

Anytime Learning at Macro-Scale: Metrics



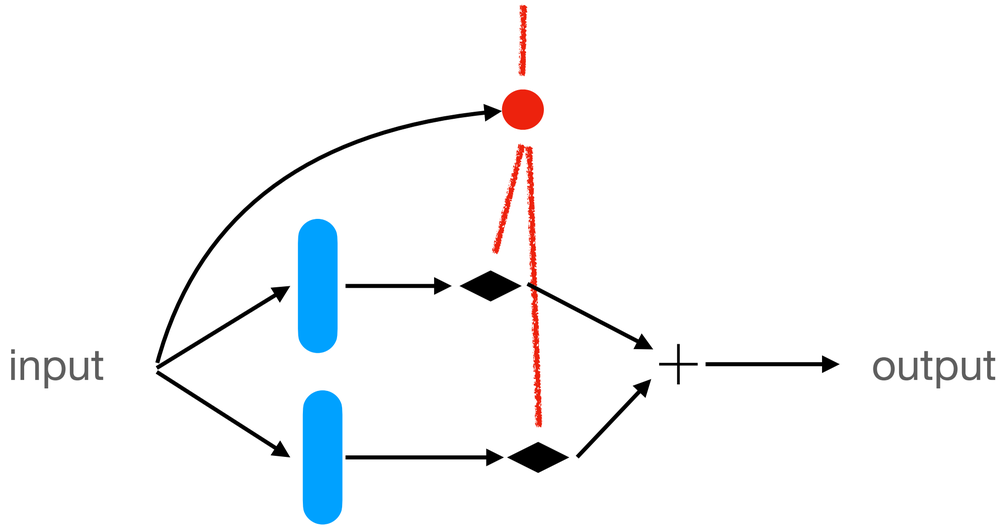
Setting

- Take any dataset, and split it into B disjoint mega-batches.
- Go over the ordered sequences of mega-batches (only once).
- Model can:
  - Loop over each mega-batch as much as it wants to (but it will be reflected in the compute).
  - Extract validation set from each mega-batch for its own cross-validation.
  - Decide how long to wait (nr. of mega-batches) before updating its parameters.
- Metrics: area under the curve of
  - Error rate
  - Memory
  - Compute

Models

- Finetuning, with various waiting time.
    - Fixed capacity.
    - Growing capacity.
- Ensembling.

Mixture of Experts (MoE)

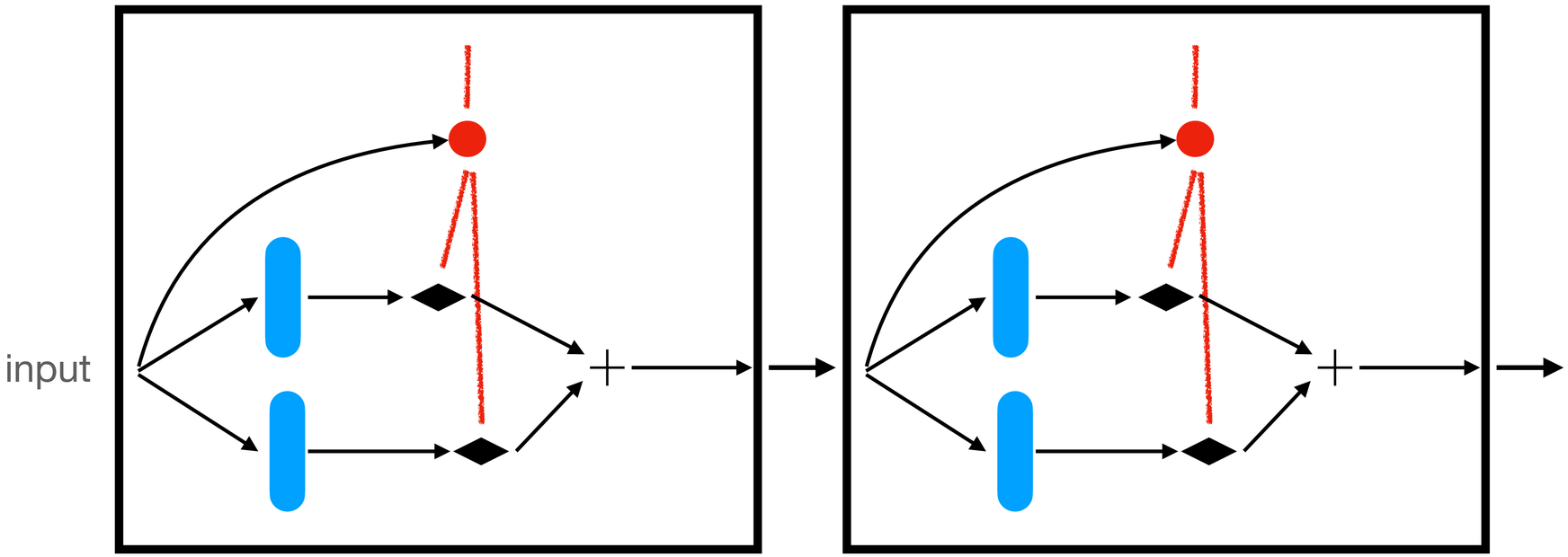


MoE with two experts

$$f(x) = \sum_{j=1}^{k} g(x)[j]\ h(x|j)$$

Jordan and Jacobs. Hierarchical mixture of experts and the EM algorithm. Neural Computation 1994

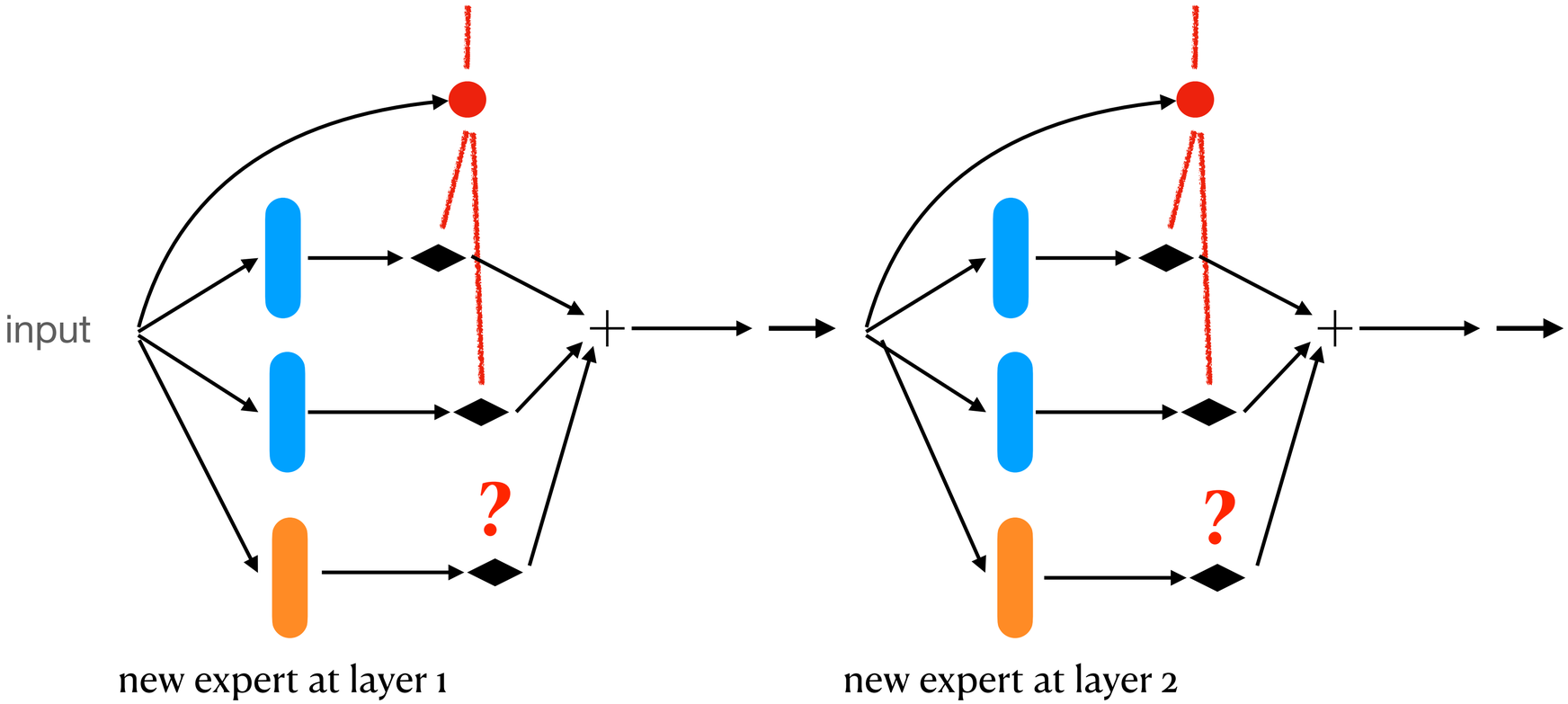
Hierarchical Mixture of Experts (MoE)



- Modular
- Computationally efficient (hard gating)

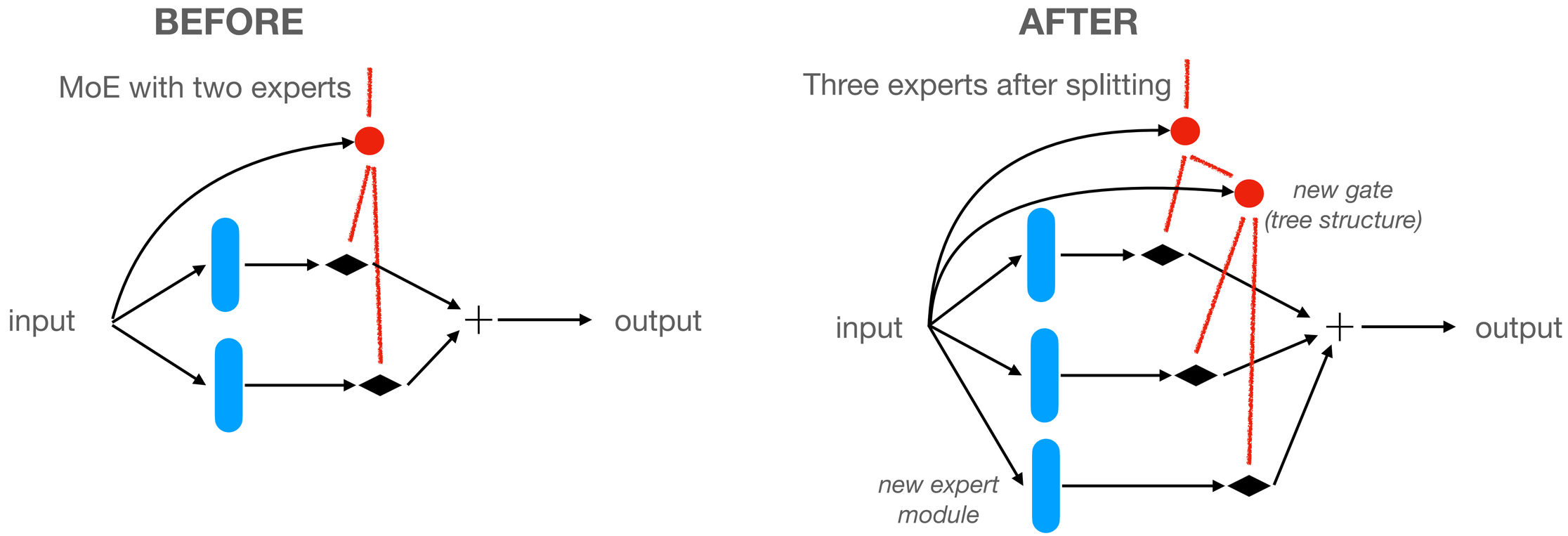
- Hard to train

Growing Mixture of Experts (gMoE)



How to make the growth step smooth and differentiable?

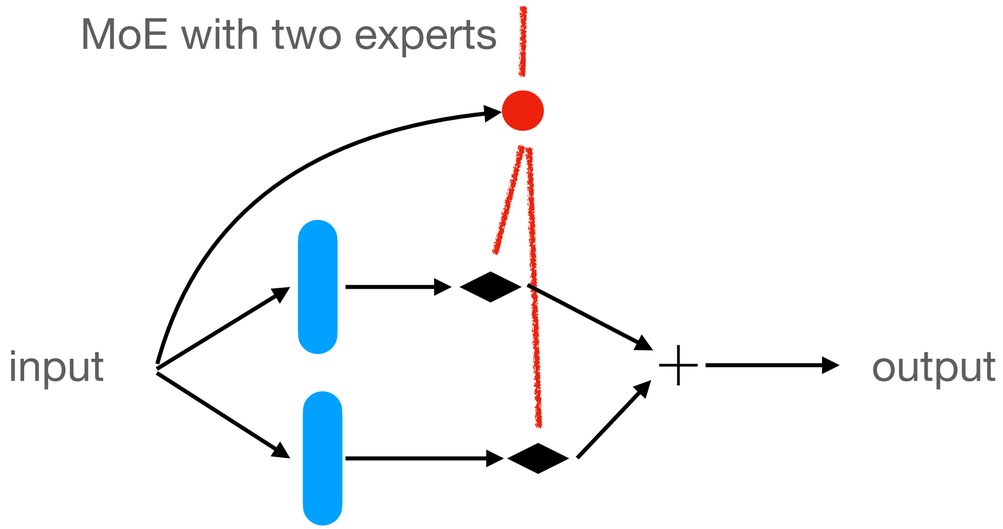
Growing Mixture of Experts (gMoE)

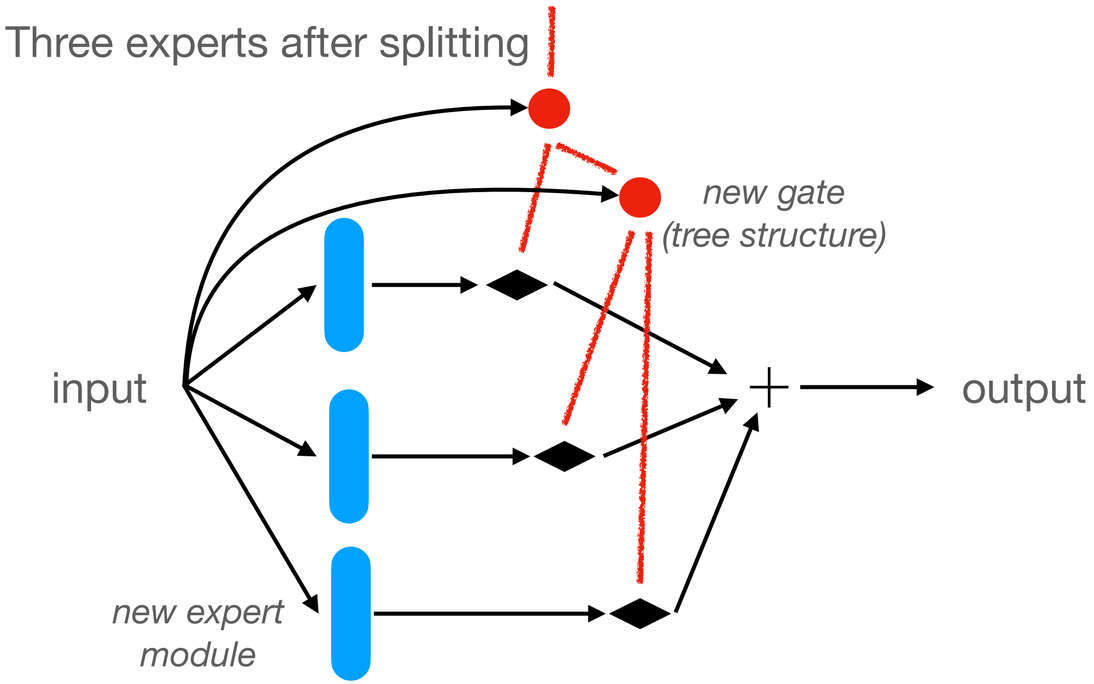


WHEN: ?
WHAT: ?
HOW: ?

Fritsch et al. Adaptively growing hierarchical mixtures of experts. NIPS 1996

Growing Mixture of Experts (gMoE)





WHEN: Every M mega-batches [even better, based on validation loss]

WHAT: By splitting expert incurring the largest cumulative loss (at each layer).

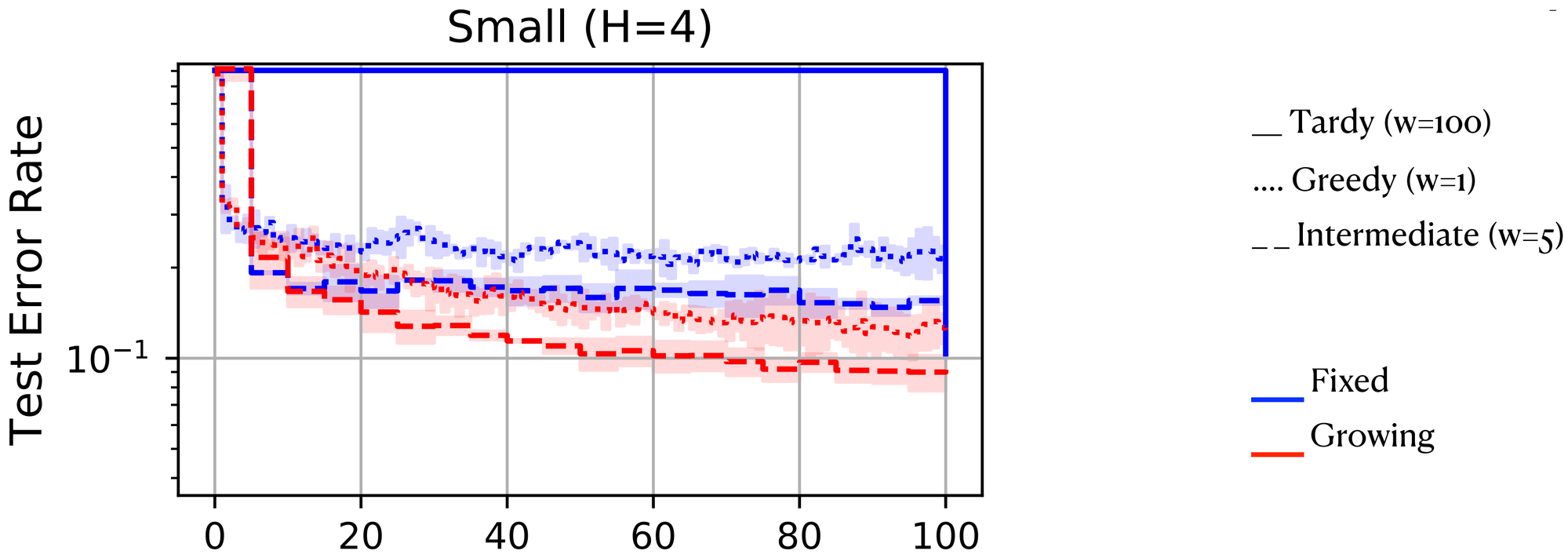
HOW: Experts are copied; Gates are expanded into a tree.

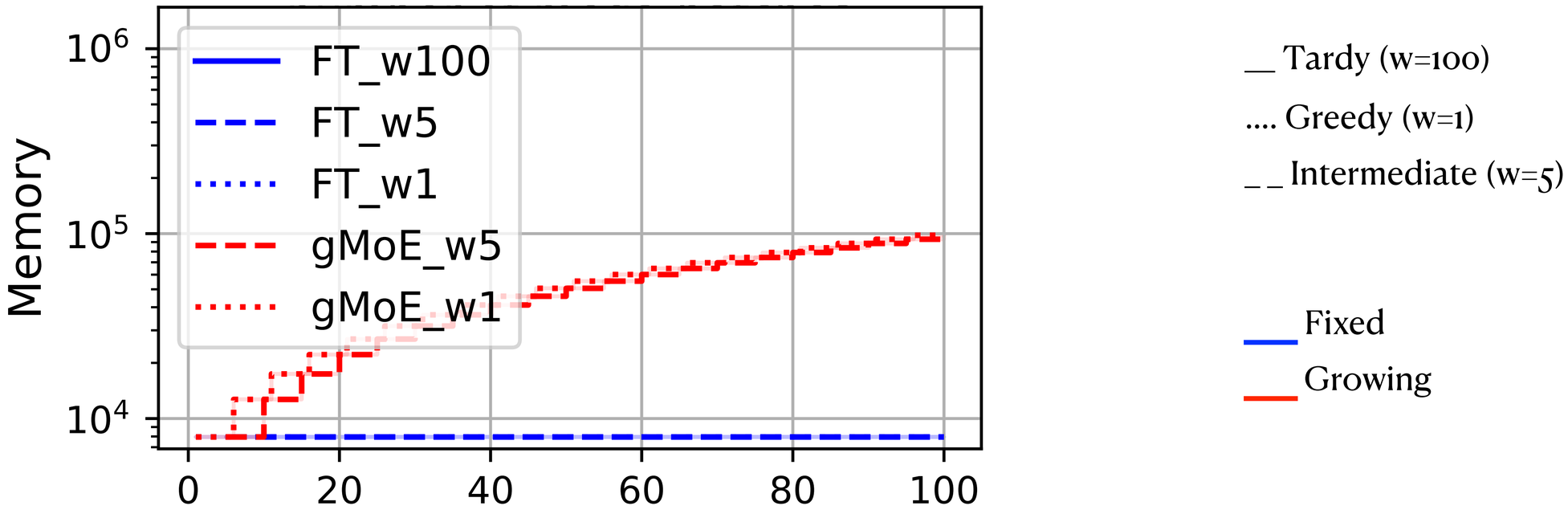
Fritsch et al. Adaptively growing hierarchical mixtures of experts. NIPS 1996

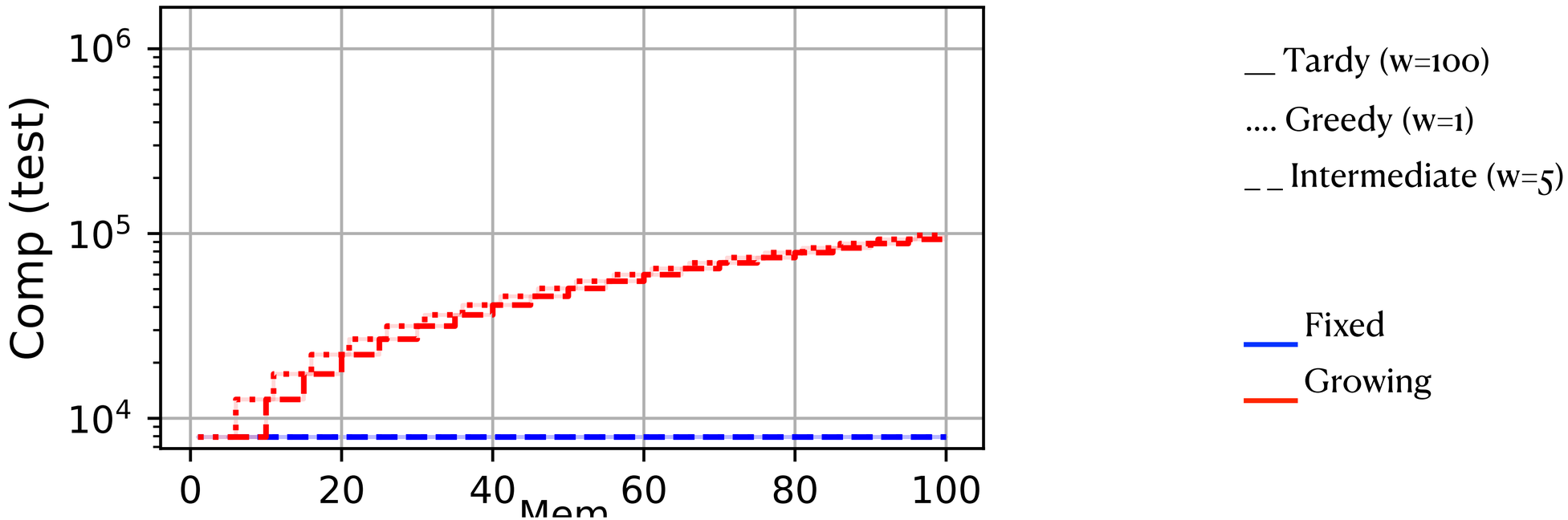
Experiment: MNIST

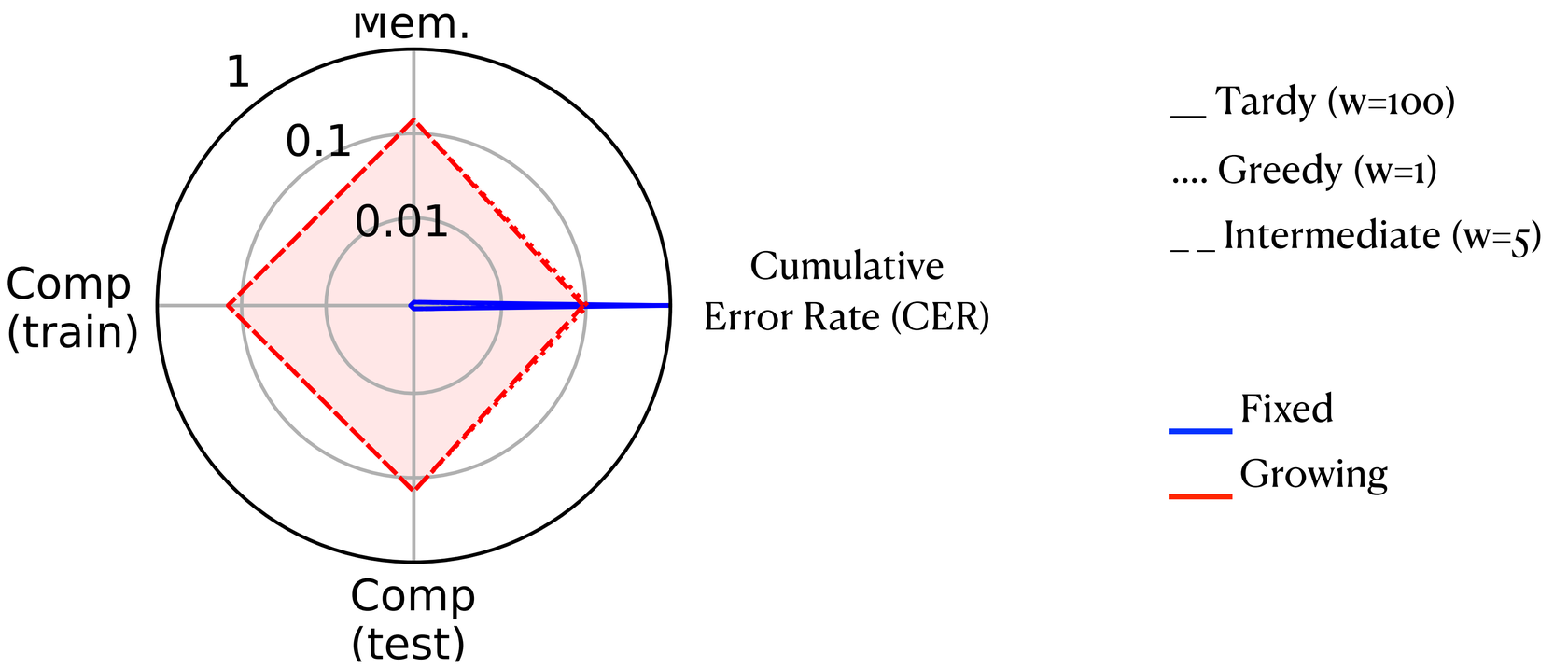
- 100 mega-batches (600 samples per mega-batch).
- 3 layers fully connected neural net.
- Models (with varying waiting time):
  - Fixed MoE with hard gating.
  - Growing MoE with hard gating.
  - Ensembling.
- Learning algorithm:
  - Fine-tuning.

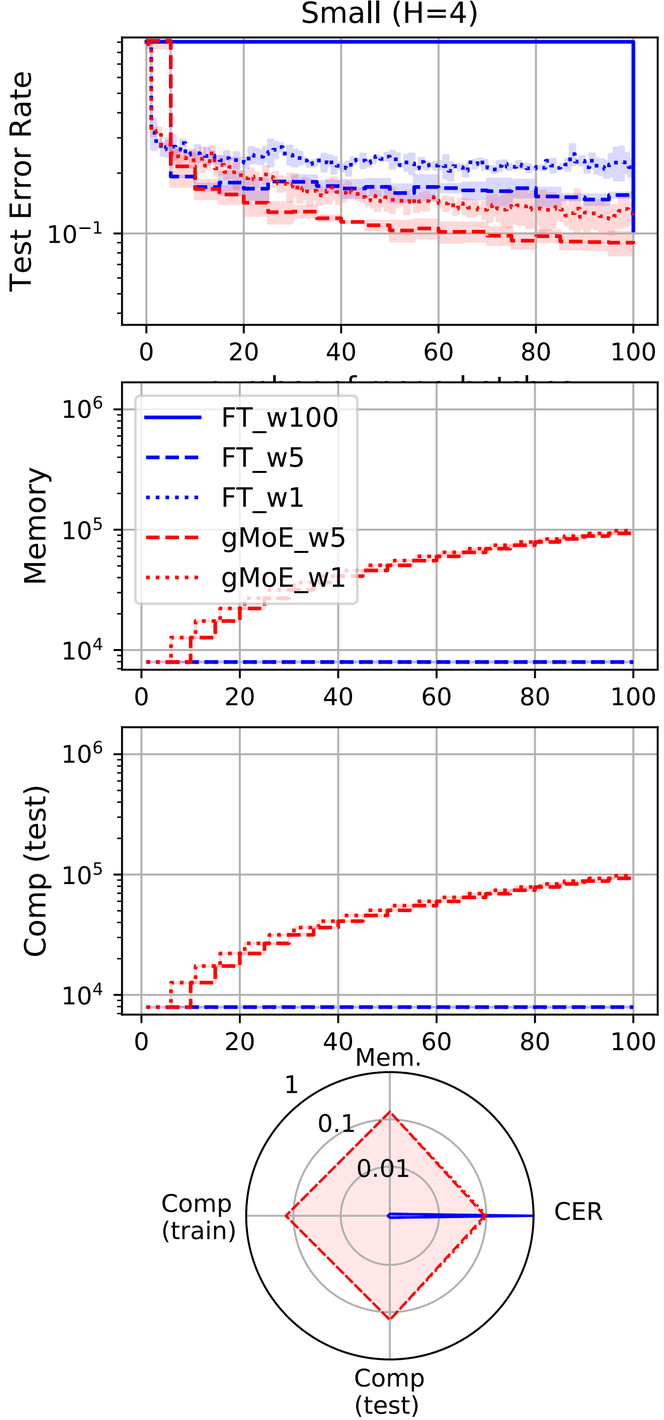
In the arXiv paper you can find experiments on CIFAR 10 and on a large scale language modeling task.


Small (H=4)
Test Error Rate
10^{-1}
0
20
40
60
80
100
Tardy (w=100)
Greedy (w=1)
Intermediate (w=5)
Fixed
Growing


Memory
10^6
10^5
10^4
0
20
40
60
80
100
FT_w100
FT_w5
FT_w1
gMoE_w5
gMoE_w1
__ Tardy (w=100)
.... Greedy (w=1)
_ _ Intermediate (w=5)
Fixed
Growing

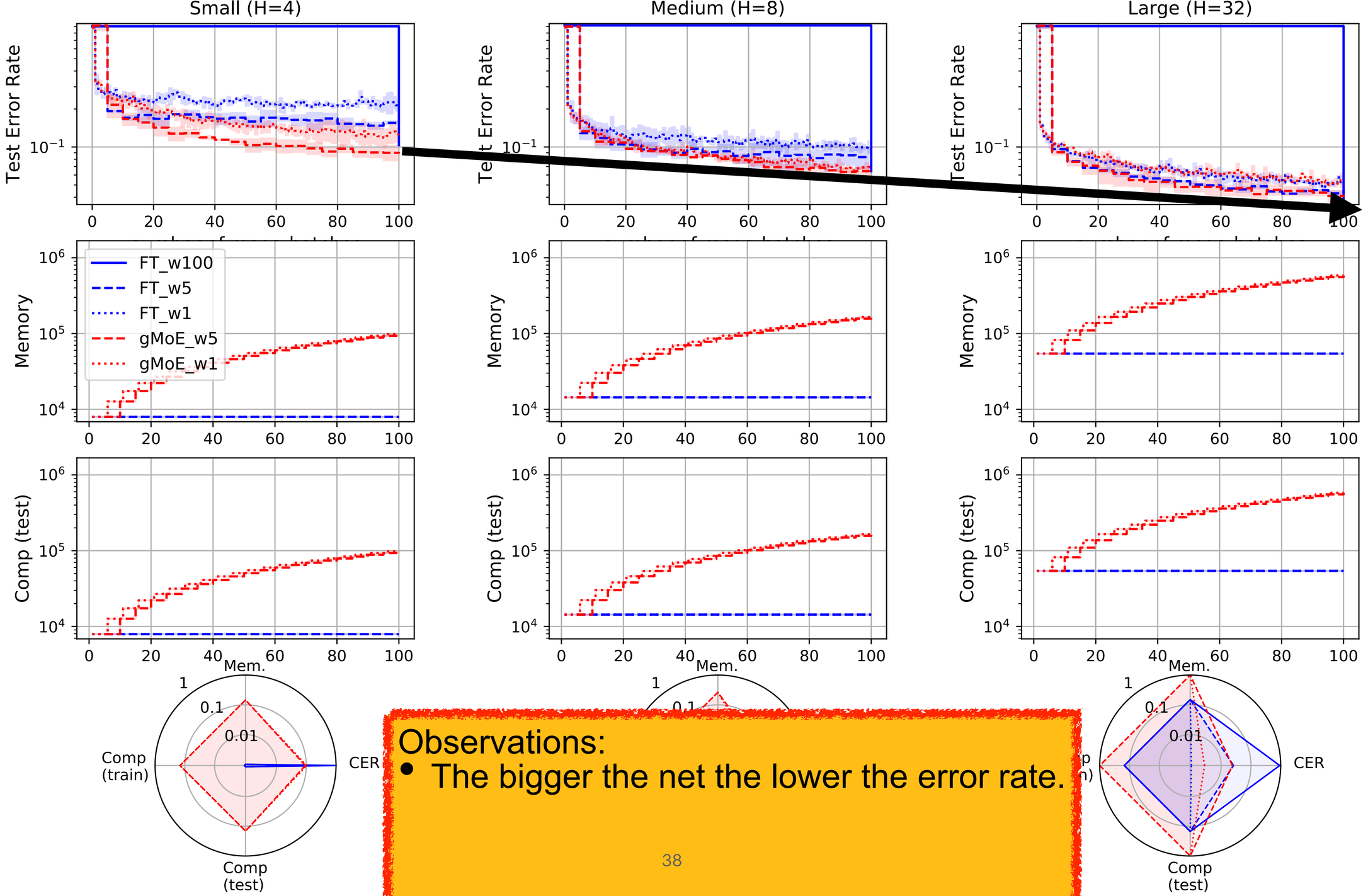

Comp (test)
10^6
10^5
10^4
0
20
40
60
80
100
Mem
Tardy (w=100)
Greedy (w=1)
Intermediate (w=5)
Fixed
Growing


Mem.
1
0.1
0.01
Comp
(train)
Comp
(test)
Cumulative
Error Rate (CER)
Tardy (w=100)
Greedy (w=1)
Intermediate (w=5)
Fixed
Growing



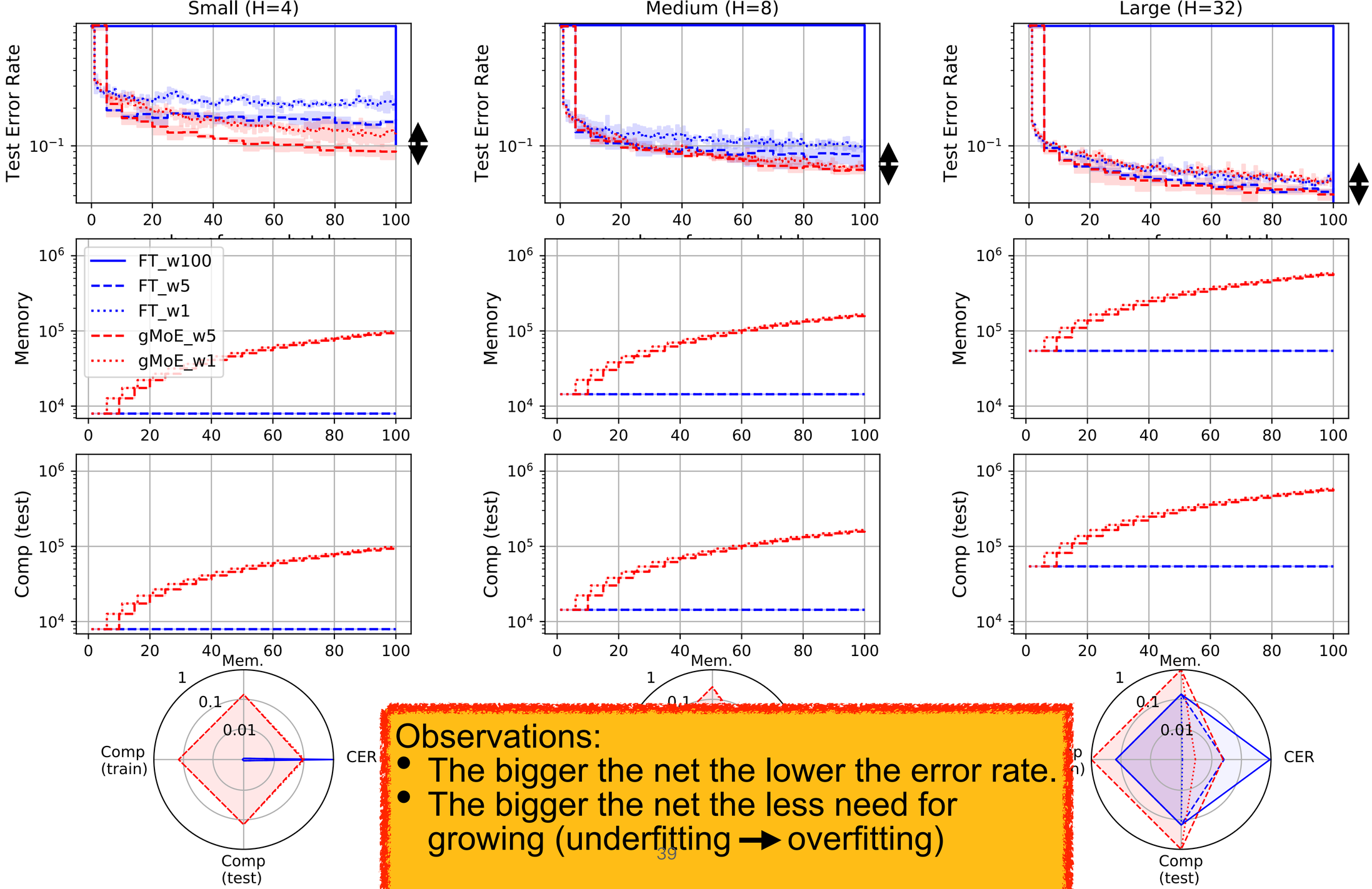
Observations:

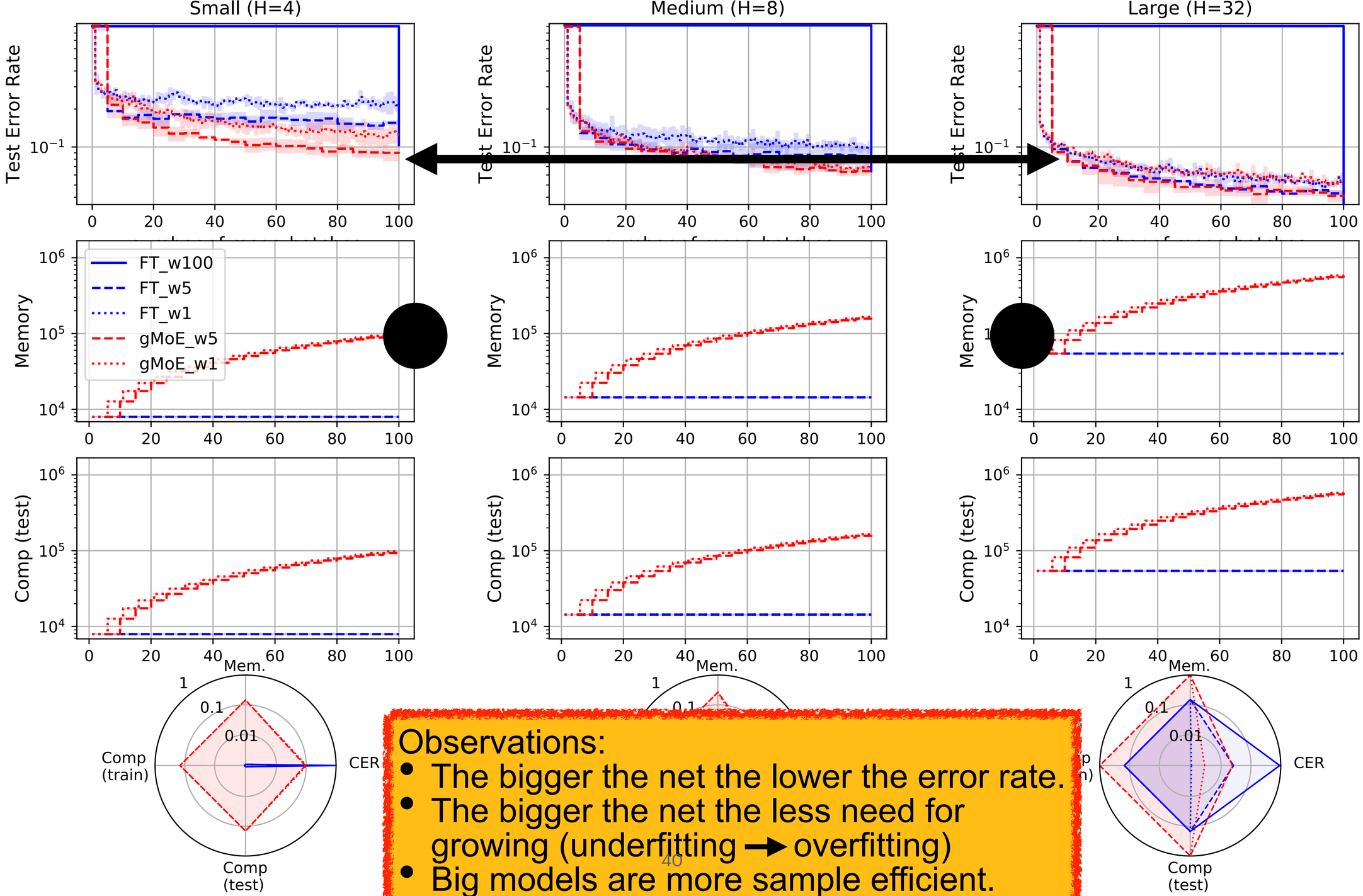
- Growing achieves lower error rate, but at higher memory and computer cost
- Waiting every 5 mega-batches (--), works better than being greedy (..) or tardy (—).



Observations:

- The bigger the net the lower the error rate.


Small (H=4)
Medium (H=8)
Large (H=32)
Test Error Rate
Memory
Comp (test)
FT_w100
FT_w5
FT_w1
gMoE_w5
gMoE_w1
Mem.
Comp (train)
Comp (test)
CER
Observations:
The bigger the net the lower the error rate.
The bigger the net the less need for growing (underfitting → overfitting)


Small (H=4)
Medium (H=8)
Large (H=32)
Test Error Rate
Memory
Comp (test)
FT_w100
FT_w5
FT_w1
gMoE_w5
gMoE_w1
Mem.
CER
Comp (train)
Comp (test)
Observations:
The bigger the net the lower the error rate.
The bigger the net the less need for growing (underfitting → overfitting)
Big models are more sample efficient.

Remarks

- Anytime learning at macro-scale is a realistic setting.
- Preliminary results show some promise in growing architectures over time.
- Bigger is better even at small scale and with fully connected nets, but it'll never be big enough. We still need to figure out how to grow.
- Several open questions:
  - How to better train modular networks?
  - Are there better modular architectures?
  - Why do big models generalize better even on small datasets?

Outline

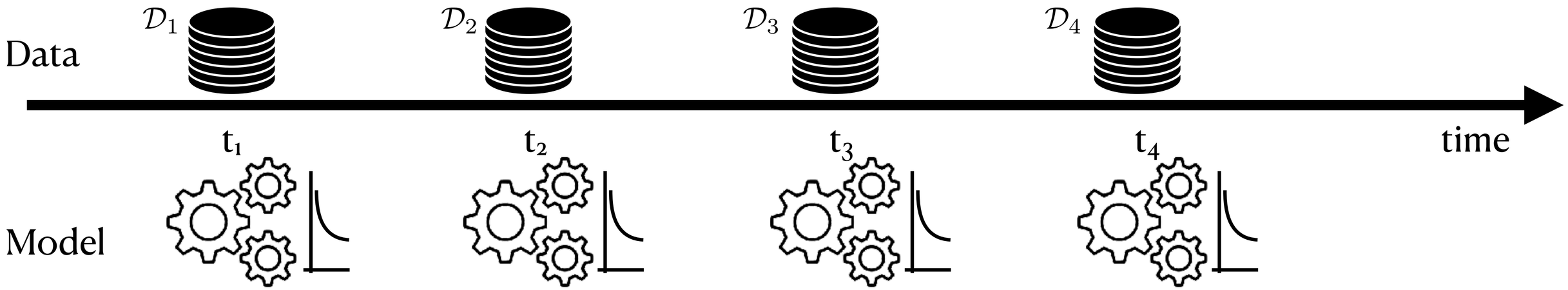
- **Anytime** Learning with Modular Architectures
  - Data arrives in large mega-batches over time
  - Metrics, Benchmarks & Models
  - Representative experiments

L. Caccia, J. Xu, M. Ott, L. Denoyer, M. Ranzato. On Anytime Learning at Macroscale, arXiv 2021

- **Continual** Learning with Modular Architectures
  - Tasks arrive in sequence. Each task has its own distribution.
  - Metrics, Benchmarks & Models
  - Representative experiments

T. Veniat, L. Denoyer, M. Ranzato. Efficient Continual Learning with Modular Networks and Task-Driven Priors, ICLR 2021

Learning Framework



- Each dataset is a task, with its own input/output distribution.
- Tasks may relate to each other, but in unknown ways.
- Task ids are given to the learner both at training and test time.
- At test time, learner can be asked to perform any previous task.

Ring. Continual learning in reinforcement environments. PhD thesis 1994

Thrun. A lifelong learning perspective for mobile robot control. IRS 1994

Continual Learning Today

Typical streams (10 tasks):

- Permuted MNIST
- Incremental CIFAR-100

Metrics:

- Average accuracy on all tasks after learning from the stream.
- Forgetting.

Issue: cannot measure transfer.

- Too few, too related tasks. Too many examples per task.
- There is no suitable metric.

Continual TRansfer Learning (CTRL) Stream

Direct transfer stream: $\mathcal{S}^{-} = (t_1^+, t_2, t_3, t_4, t_5, t_1^-)$

$t_i \perp t_j$

$|t^+| > |t| > |t^-|$

Knowledge update stream: $\mathcal{S}^{+} = (t_1^-, t_2, t_3, t_4, t_5, t_1^+)$

t, t', t'' related

Transfer to similar input/ output distributions:

$\mathcal{S}^{\text{in}} = (t_1, t_2, t_3, t_4, t_5, t_1')$

$\mathcal{S}^{\text{out}} = (t_1, t_2, t_3, t_4, t_5, t_1'')$

Underlying datasets:

- CIFAR 10
- CIFAR 100
- MNIST
- Rainbow MNIST
- Fashion MNIST
- D. Textures
- SVHN

Plasticity stream: $\mathcal{S}^{\text{pl}} = (t_1, t_2, t_3, t_4, t_5)$

Long stream: $\mathcal{S}^{\text{long}}$ Stream with 100 tasks (derived from 5 vision datasets). Various degrees of relatedness.

Continual TRansfer Learning (CTRL) Stream

Direct transfer stream: $\mathcal{S}^{-} = (t_1^+, t_2, t_3, t_4, t_5, t_1^-)$

Knowledge update stream: $\mathcal{S}^{+} = (t_1^-, t_2, t_3, t_4, t_5, t_1^+)$

Transfer to similar input/ output distributions:

$\mathcal{S}^{\text{in}} = (t_1, t_2, t_3, t_4, t_5, t_1')$

$\mathcal{S}^{\text{out}} = (t_1, t_2, t_3, t_4, t_5, t_1'')$

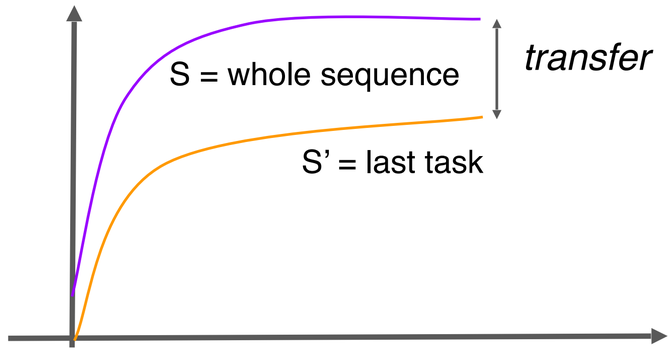
Plasticity stream: $\mathcal{S}^{\text{pl}} = (t_1, t_2, t_3, t_4, t_5)$

$t_i \perp t_j$

$|t^+| > |t| > |t^-|$

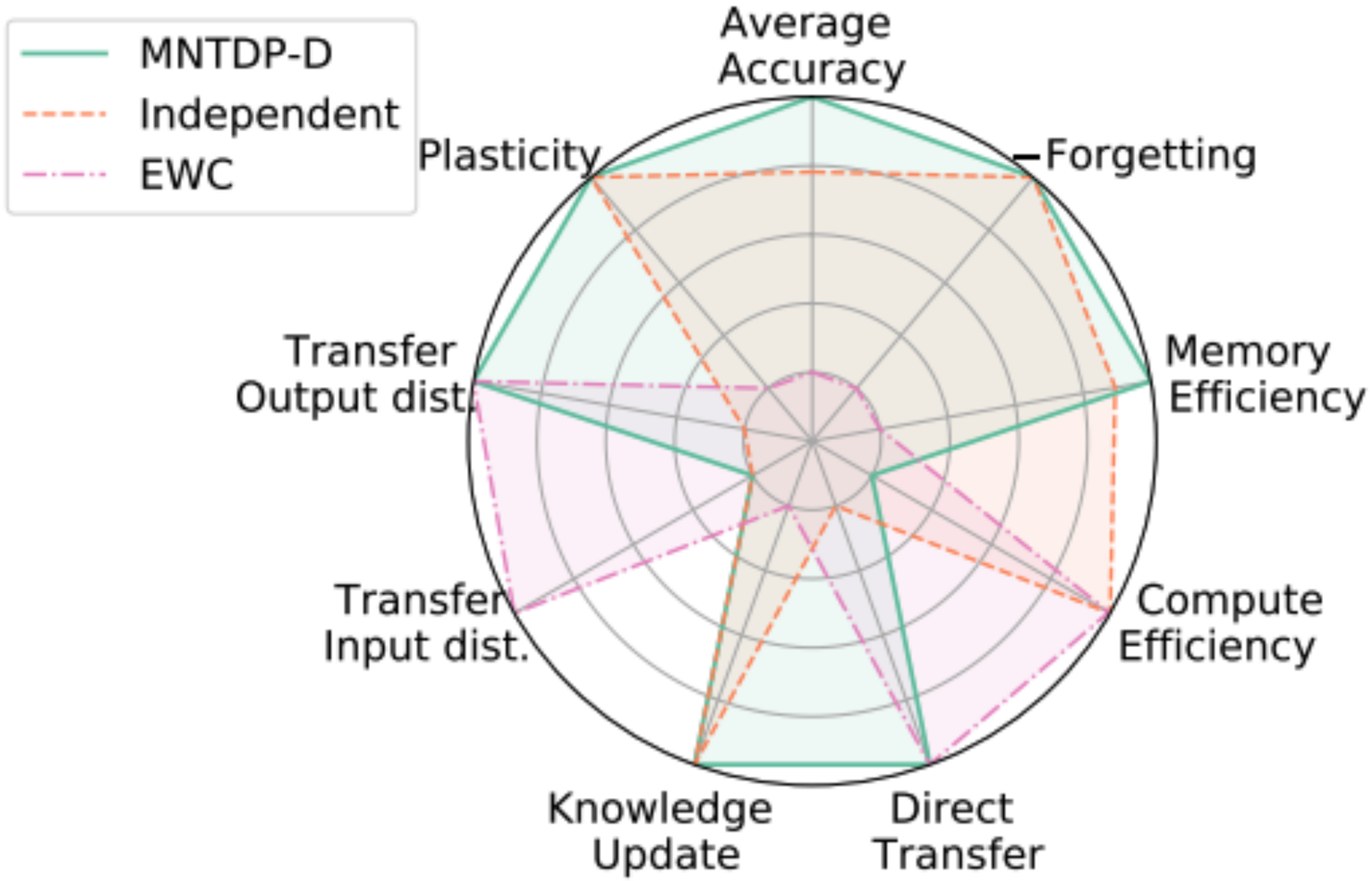
t, t', t'' related

Use transfer metric:



Long stream: $\mathcal{S}^{\text{long}}$ Stream with 100 tasks (derived from 5 vision datasets). Various degrees of relatedness.

Holistic Performance Assessment



Model

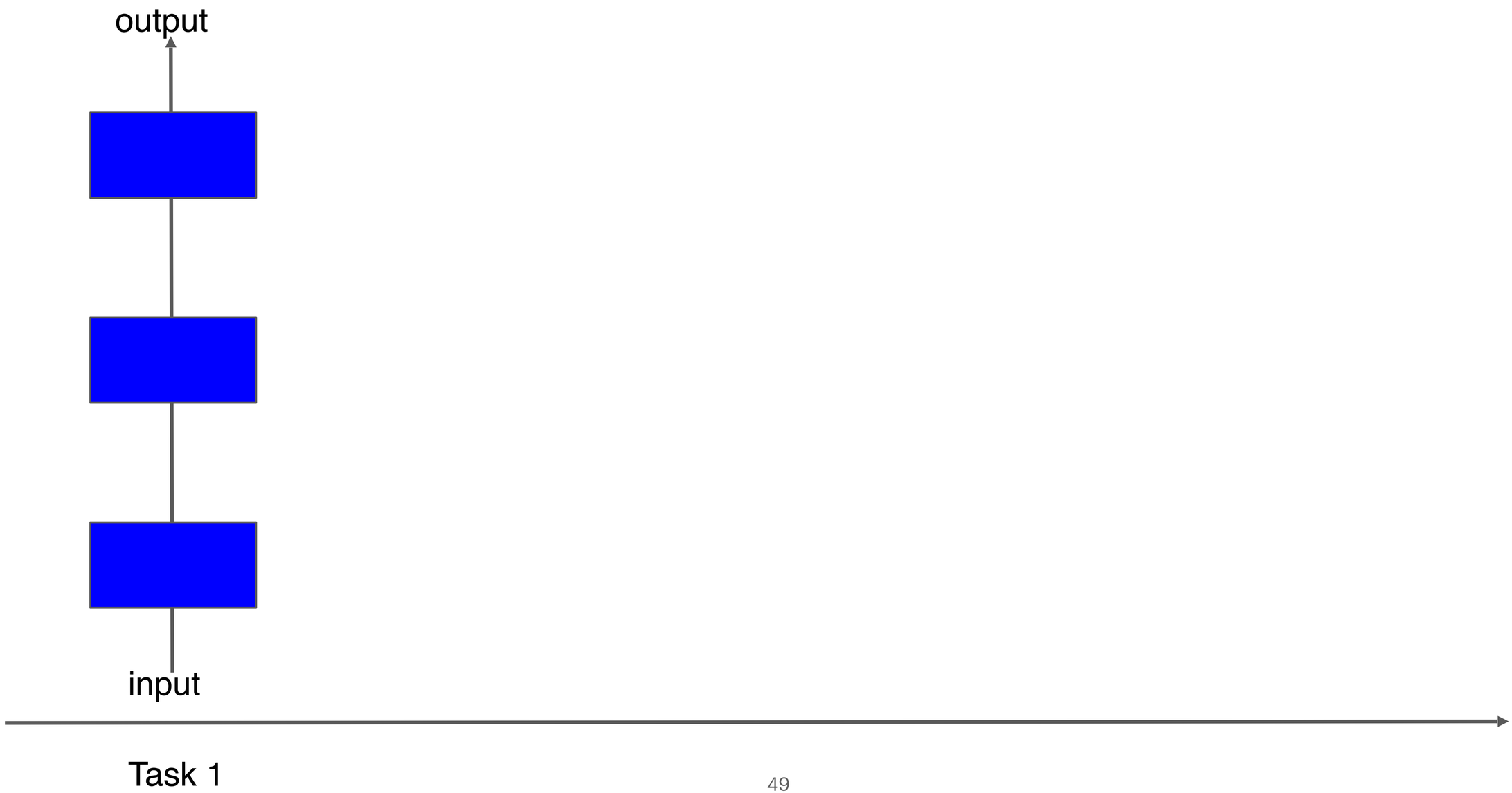
Desiderata:

- Performs well in terms of:
  - average accuracy
  - Forgetting
  - Transfer
- Scale sub-linearly with the number of tasks
- Compositional

Prior work:

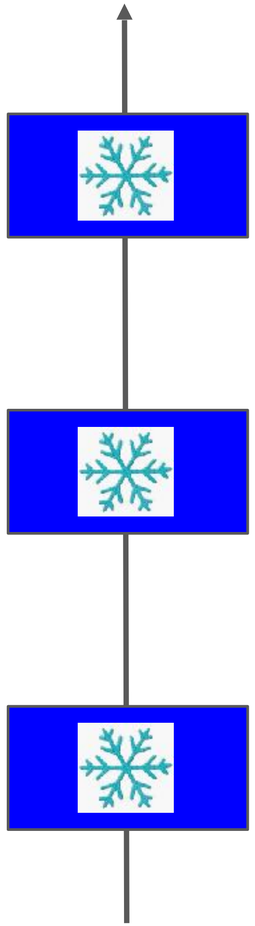
- Regularization-based approaches (e.g., EWC)
- Memory replay approaches
- **Modular architectures**

Modular Network with Task Driven Prior (MNTDP)

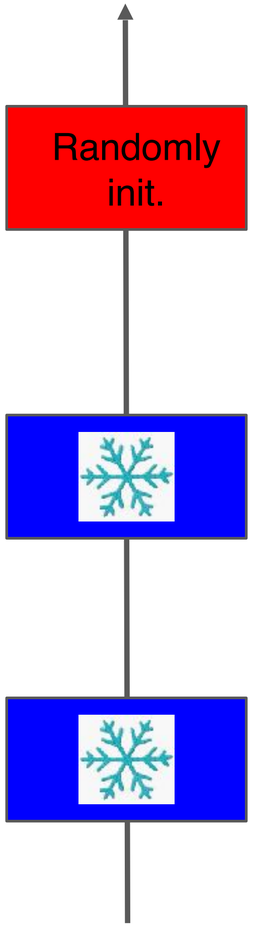


MNTDP

0) Perturb previous net. 1) Train K nets independently.

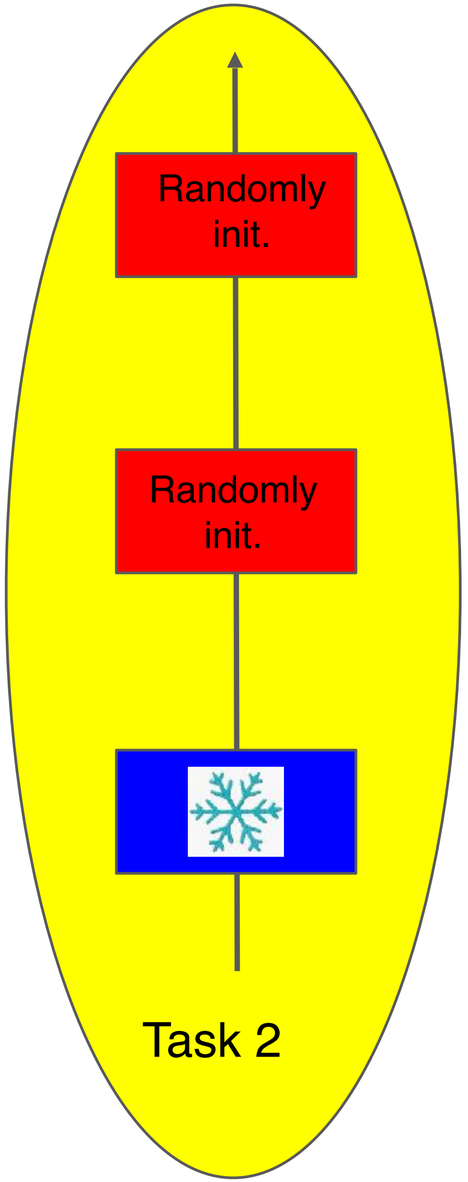


Task 2

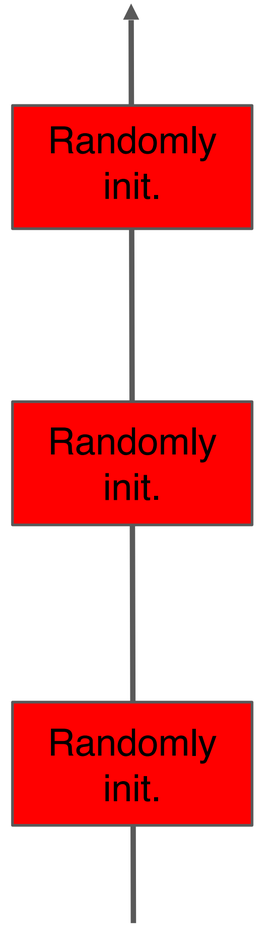


Task 2

2) Select based on validation set.

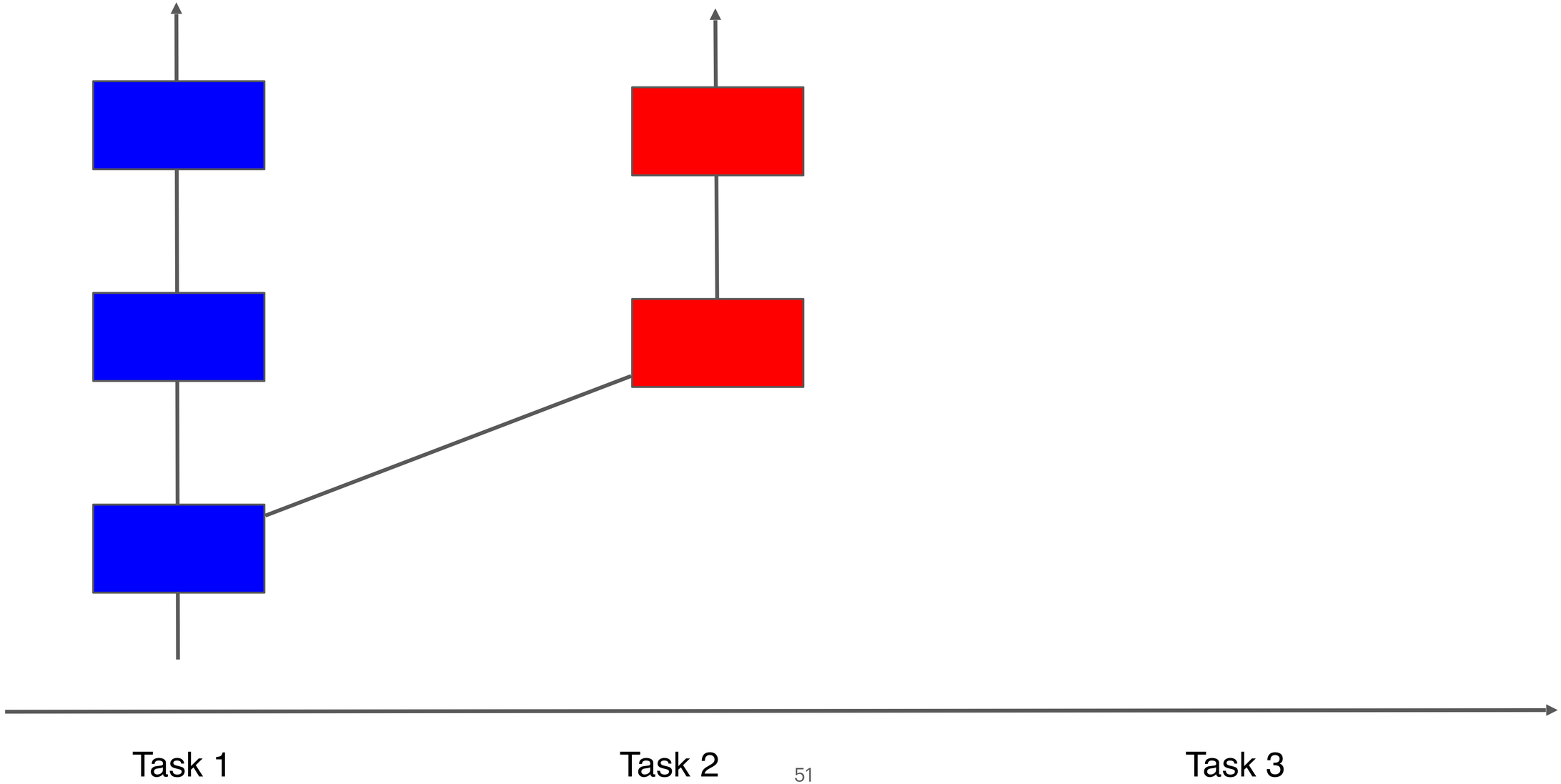


Task 2

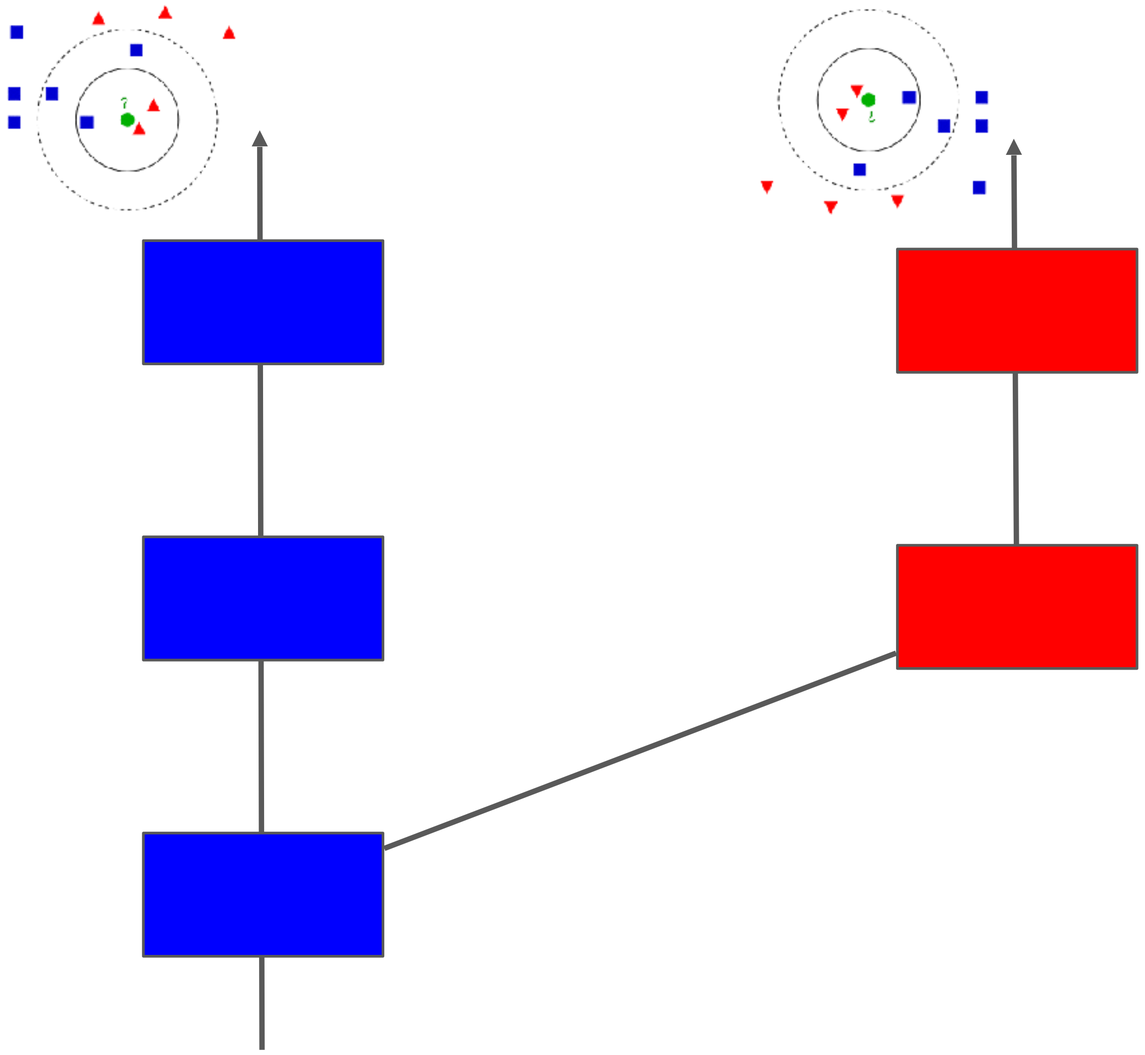


Task 2

MNTDP



MNTDP



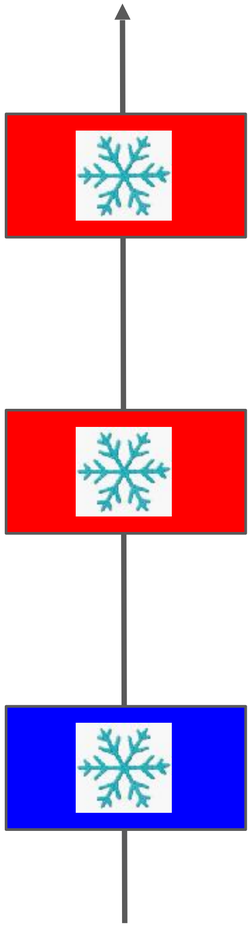
Task 3

Find most similar previous task:
E.g., based on k-NN in feature space.

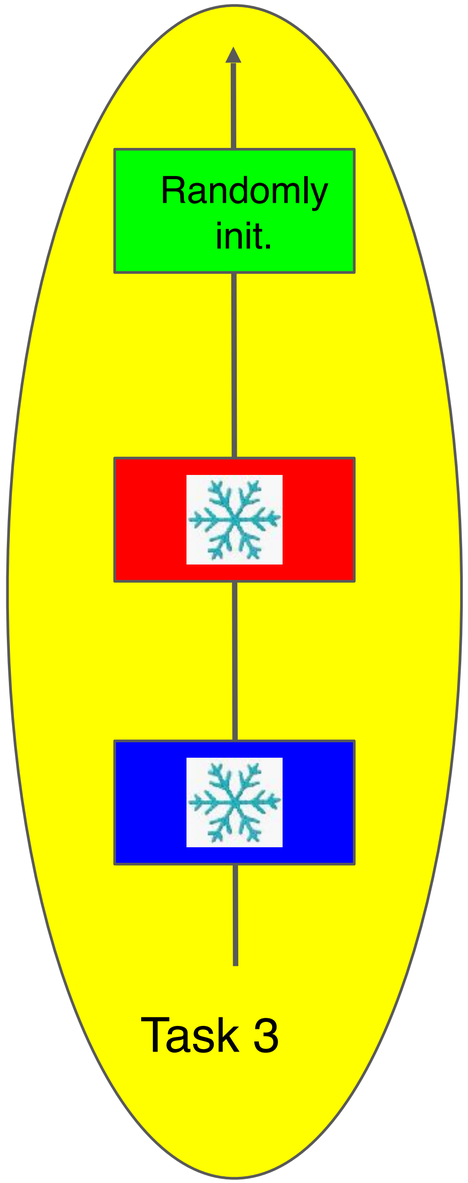
Select corresponding architecture.

MNTDP

0) Perturb previous net. 1) Train K nets independently.

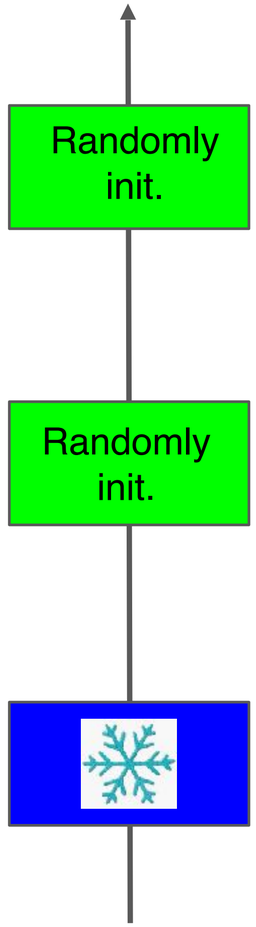


Task 3

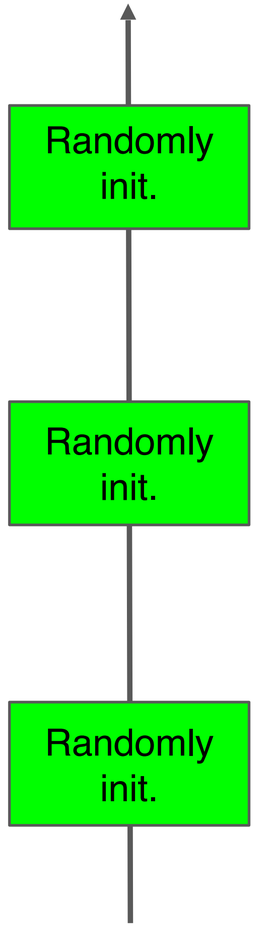


Task 3

2) Select based on validation set.

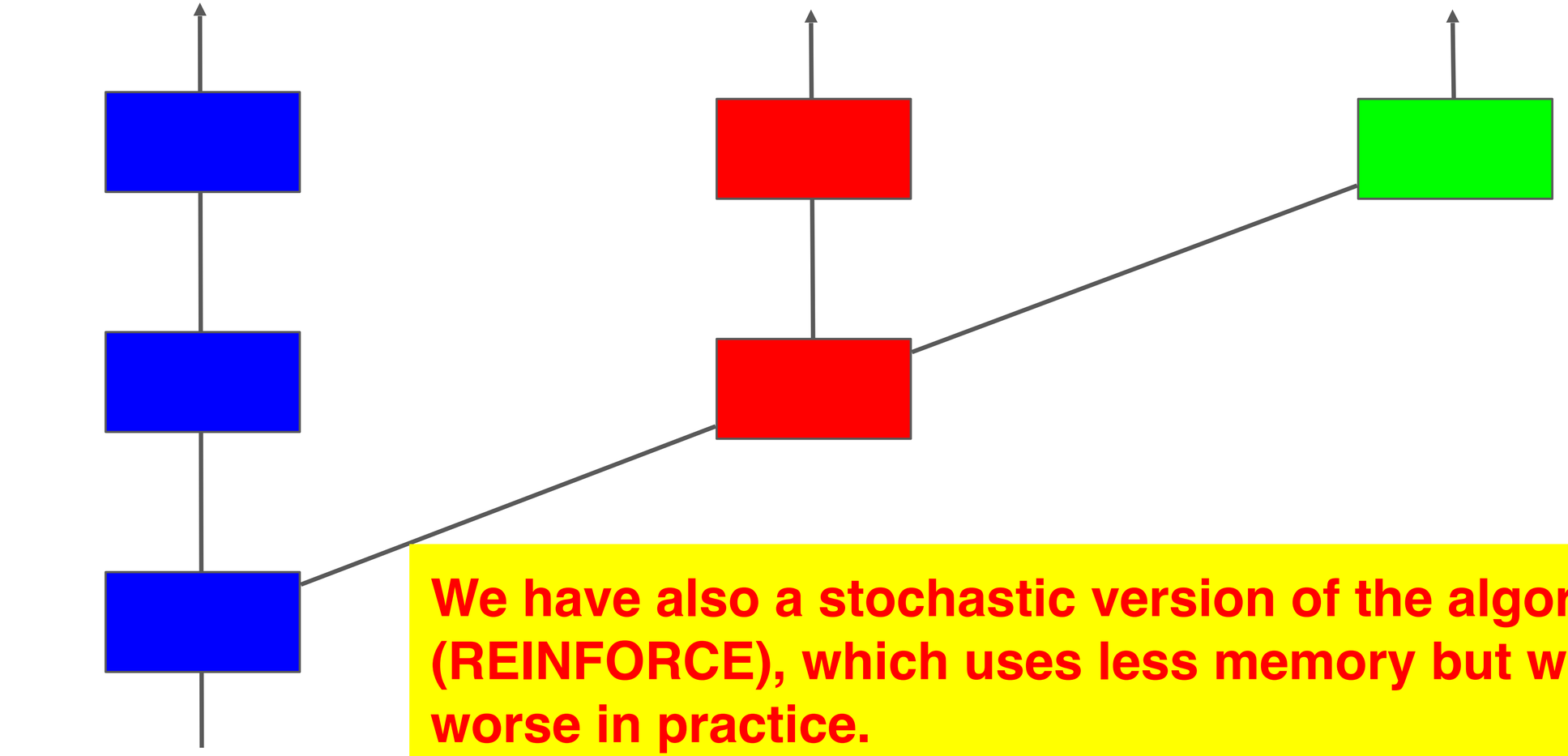


Task 3



Task 3

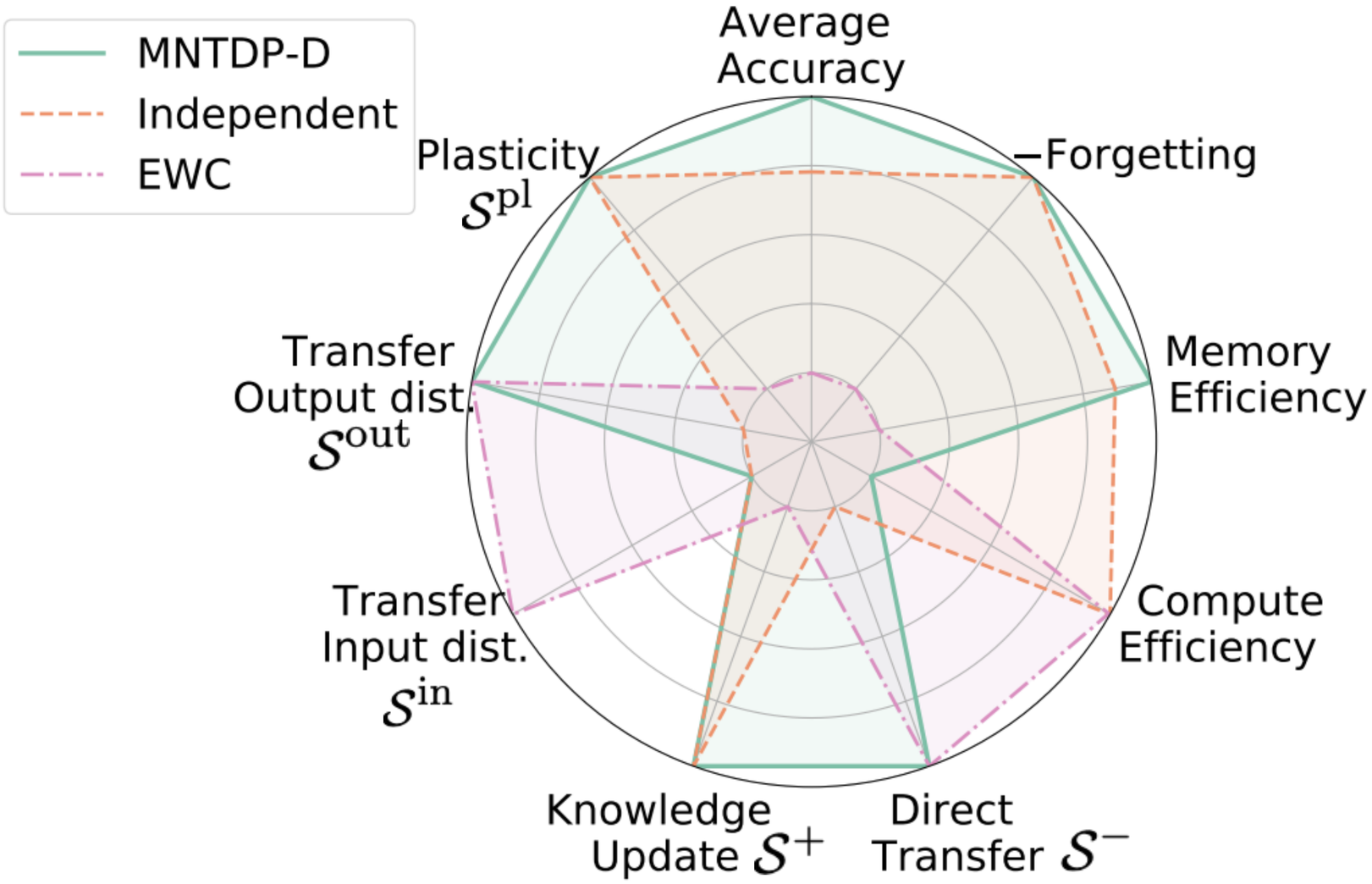
MNTDP



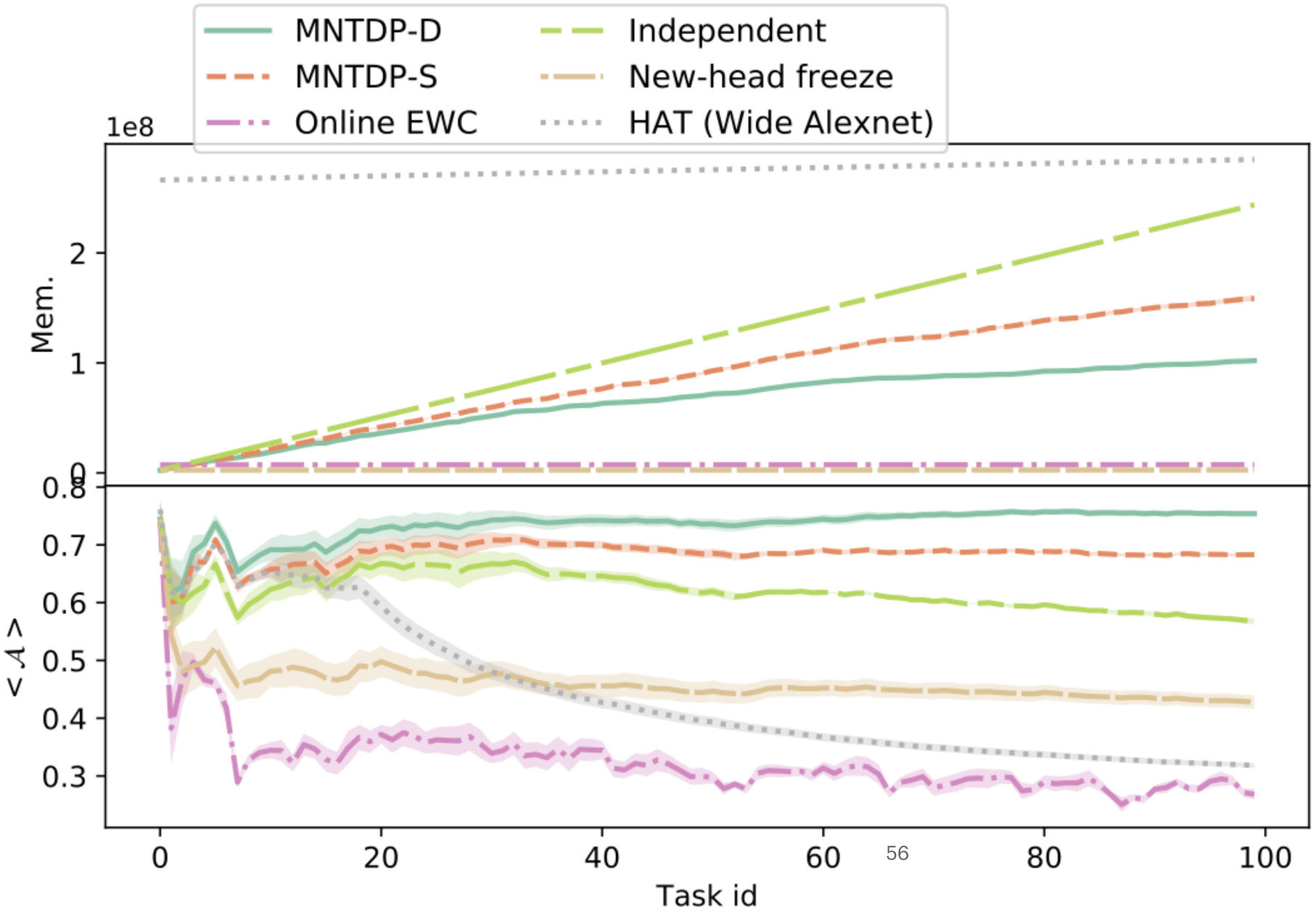
We have also a stochastic version of the algorithm (REINFORCE), which uses less memory but works slightly worse in practice.

Task 1 Task 2 Task 3

Results on CTRL



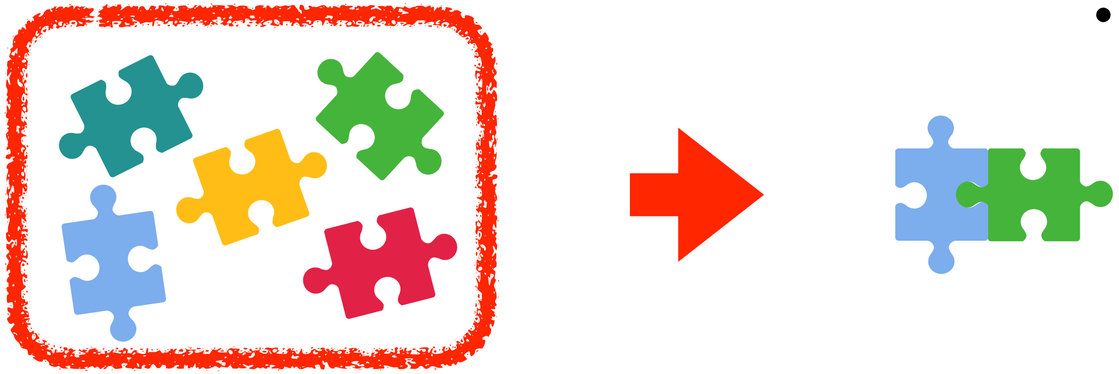
Results on $\mathcal{S}^{\text{long}}$



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

Conclusion

- Datasets are not static, hence models should probably be non static either.
- Great opportunity for making learning more efficient (statistically and computationally).
- How to adapt capacity and retain efficiency both at training and test time?
- How to modularize learning?



- In this talk:
  - Stationary setting (ALMA)
    - Growing MoE
  - Non-stationary setting
    - CTRL
    - MNTDP

- This relates to:
  - Anytime and continual learning
  - Multi-task/few-shot/meta learning
  - AutoML
  - Causal Learning

THANK YOU

Ludovic Denoyer

Tom Veniat

Lucas Caccia

Jing Xu

Myle Ott