Machine Learning Beyond Static Datasets

ESSAI 2023

Dr. Martin Mundt,

Research Group Leader, TU Darmstadt & hessian.AI

Board Member of Directors, ContinualAI

Course: http://owll-lab.com/teaching/essai-23

Day 2 - The Past: Forgetting & Memory

Early definition: lifelong ML

Definition - **Lifelong Machine Learning** - Thrun 1996:

“The system has performed N tasks. When faced with the (N+1)th task, it uses the knowledge gained from the N tasks to help the (N+1)th task.”

“Is Learning The n-th Thing Any Easier Than Learning the First?” (NeurIPS 1996) & “Explanation based Neural Network Learning A Lifelong Learning Approach”, Springer US, 1996

Early definition: ~~lifelong ML~~ transfer learning

Definition - **~~Lifelong Machine~~ Transfer Learning** - Thrun 1996:

“The system has performed N tasks. When faced with the (N+1)th task, it uses the knowledge gained from the N tasks to help the (N+1)th task.”

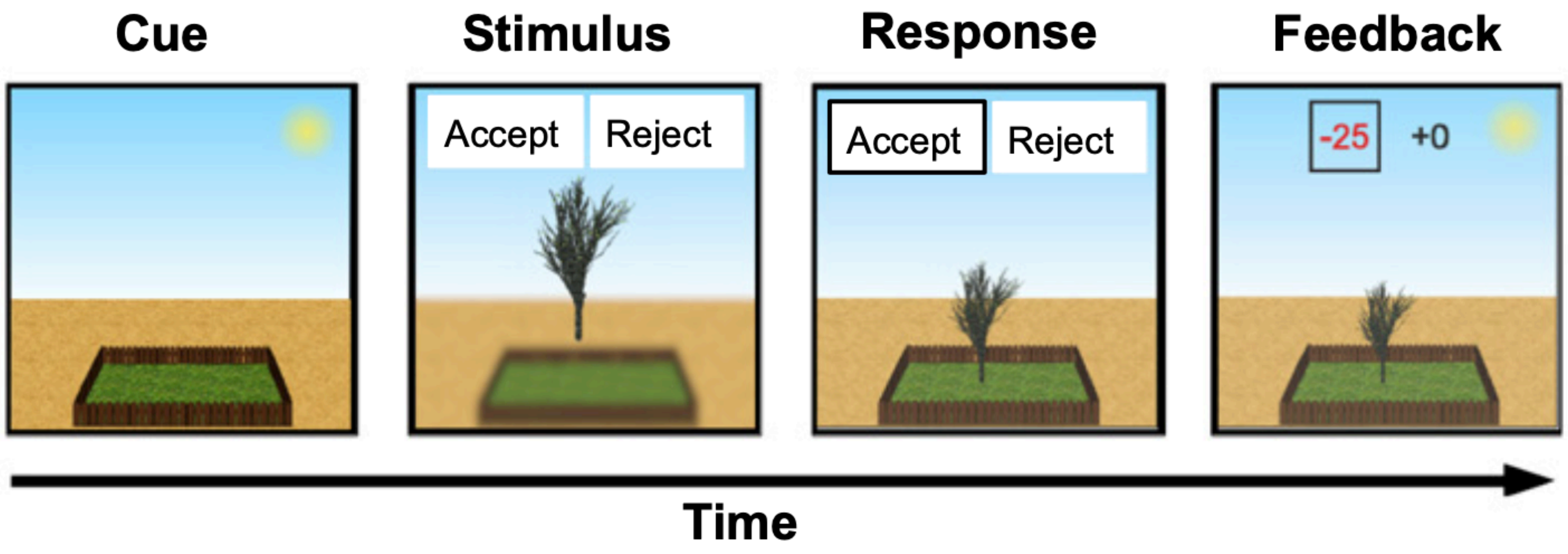
- Transfer learning does not care what happens to the source, it is only concerned with target domain & task!

“Is Learning The n-th Thing Any Easier Than Learning the First?” (NeurIPS 1996) & “Explanation based Neural Network Learning A Lifelong Learning Approach”, Springer US, 1996

What is the challenge of caring about source & target?
How humans learn continually

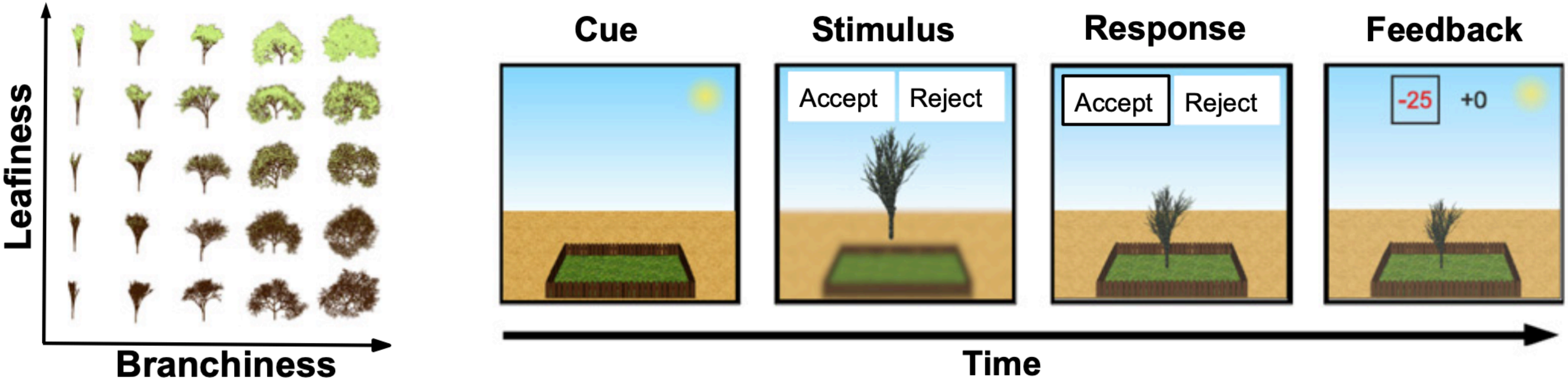
When do you think

humans do well in this?



Flesch et al, "Comparing continual task learning in minds and machines", PNAS 115, 2018

What is the challenge of caring about source & target?
How humans learn continually



Humans seem to actively benefit from temporal correlation during “training”.

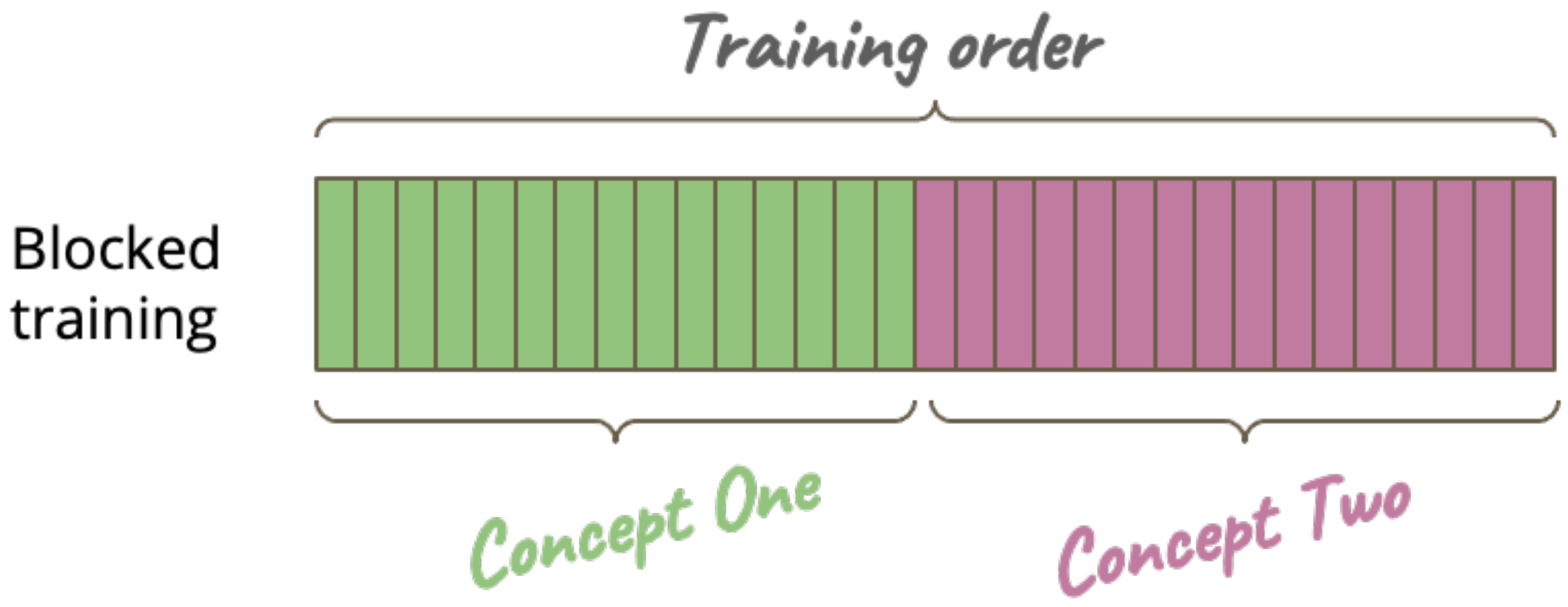
They do well if trees sensibly follow leaf & branch density

Flesch et al, “Comparing continual task learning in minds and machines”, PNAS 115, 2018

What is the challenge of caring about source & target?
How humans learn continually

What do you think will happen if we present such a curriculum in ML?

How do we typically train in ML?

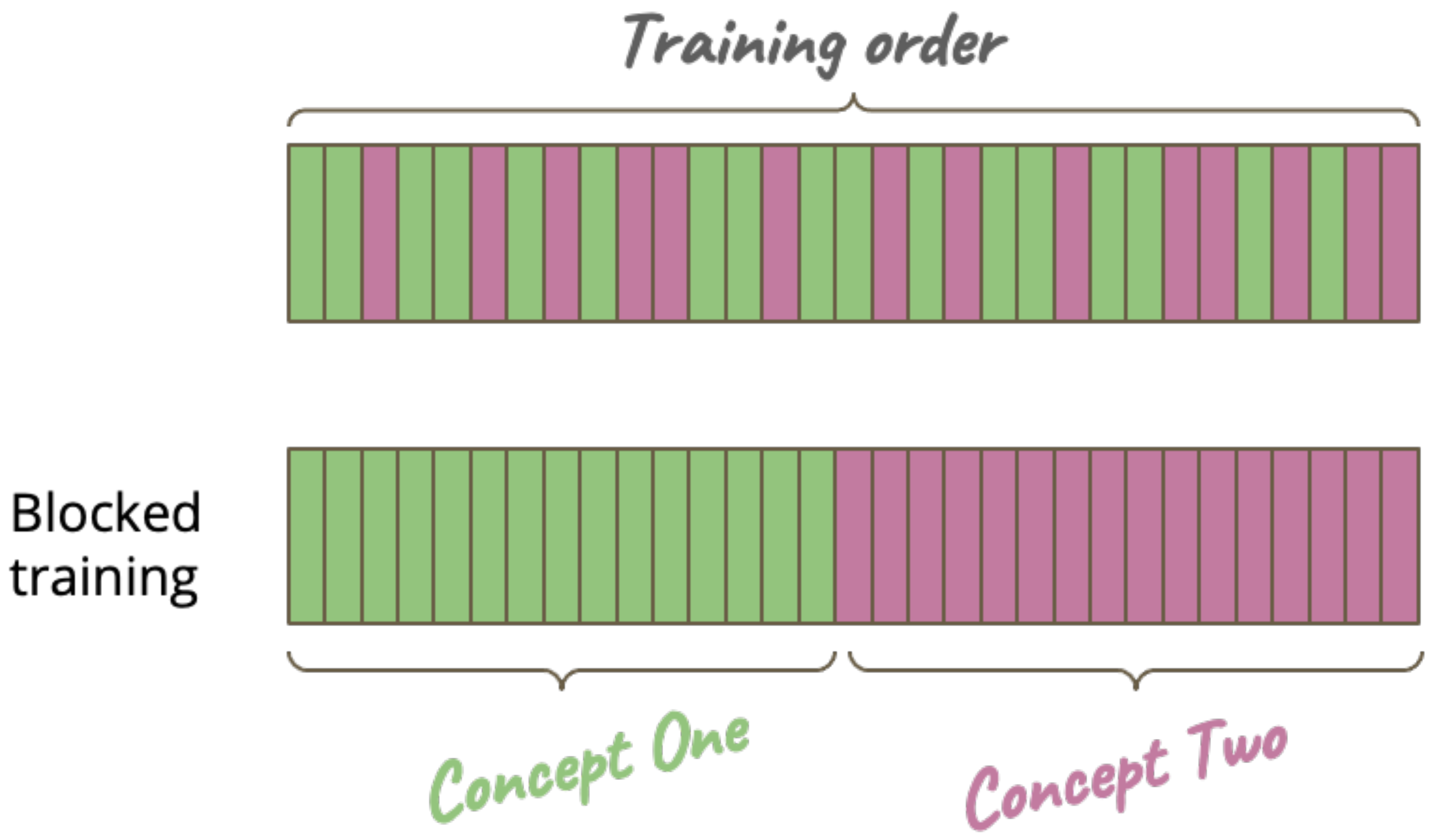


Flesch et al, “Modelling continual learning in humans with Hebbian context gating and exponentially decaying task signals”, PLOS Computational Bio, 2023

What is the challenge of caring about source & target?
How machines learn

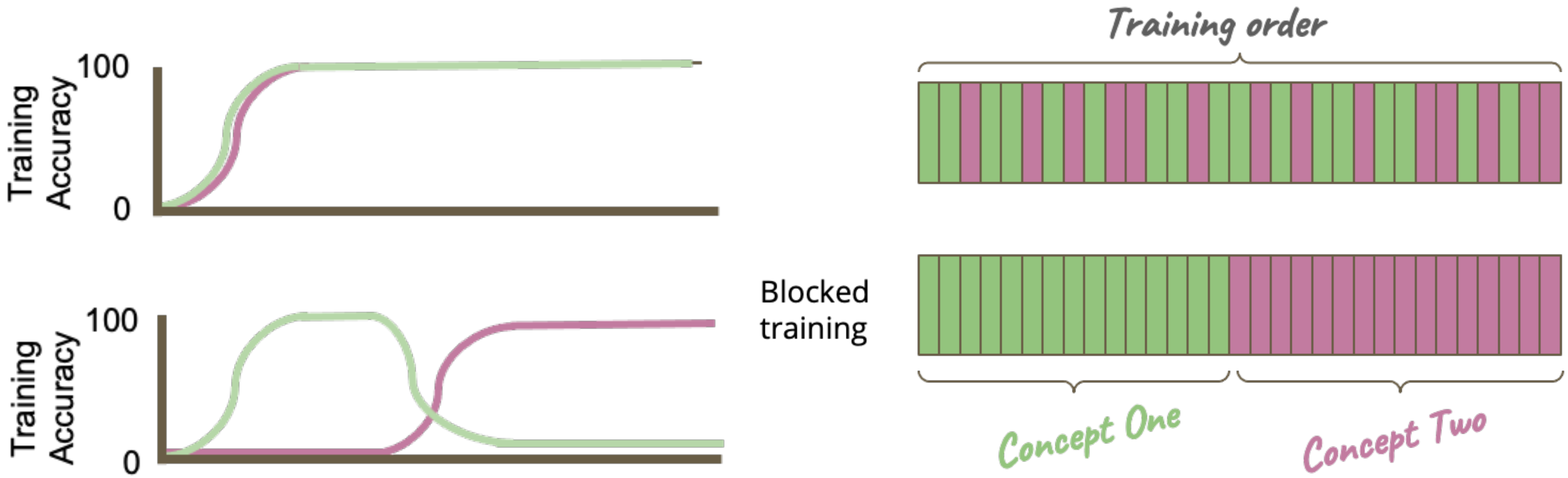
What do you think will happen if we present such a curriculum in ML?

How do we typically train in ML?



Flesch et al, “Modelling continual learning in humans with Hebbian context gating and exponentially decaying task signals”, PLOS Computational Bio, 2023

What is the challenge of caring about source & target?
How machines learn

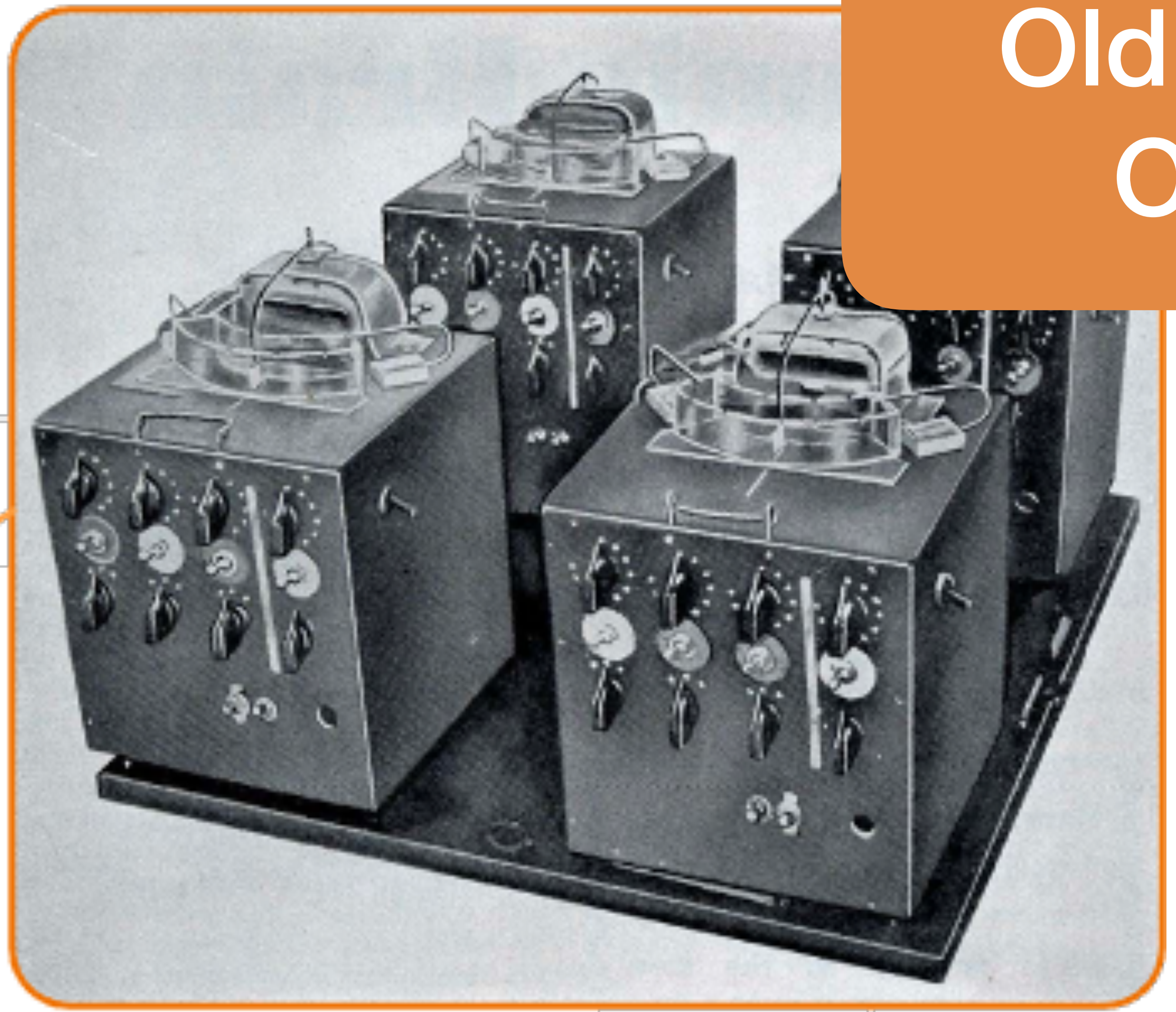


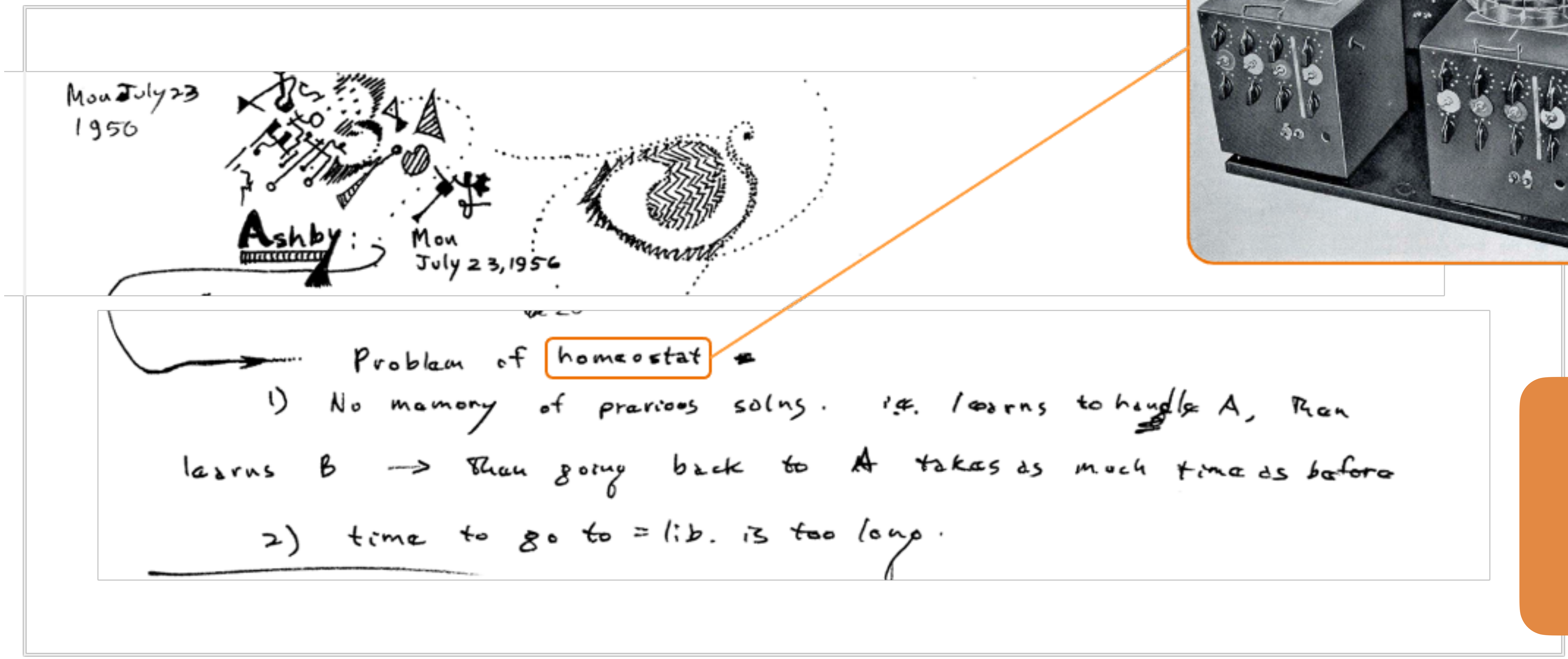
Machine learning typically shuffles data & performs poorly when data is ordered

Flesch et al, "Modelling continual learning in humans with Hebbian context gating and exponentially decaying task signals", PLOS Computational Bio, 2023

Machines don't learn like humans: catastrophic interference (McCloskey & Cohen 89)

Old Problems, Old Ideas





Mon July 23 1956

Ashby: Mon July 23, 1956

Problem of homeostat

1) No memory of previous solns. i.e. learns to handle A, then learns B → then going back to A takes as much time as before

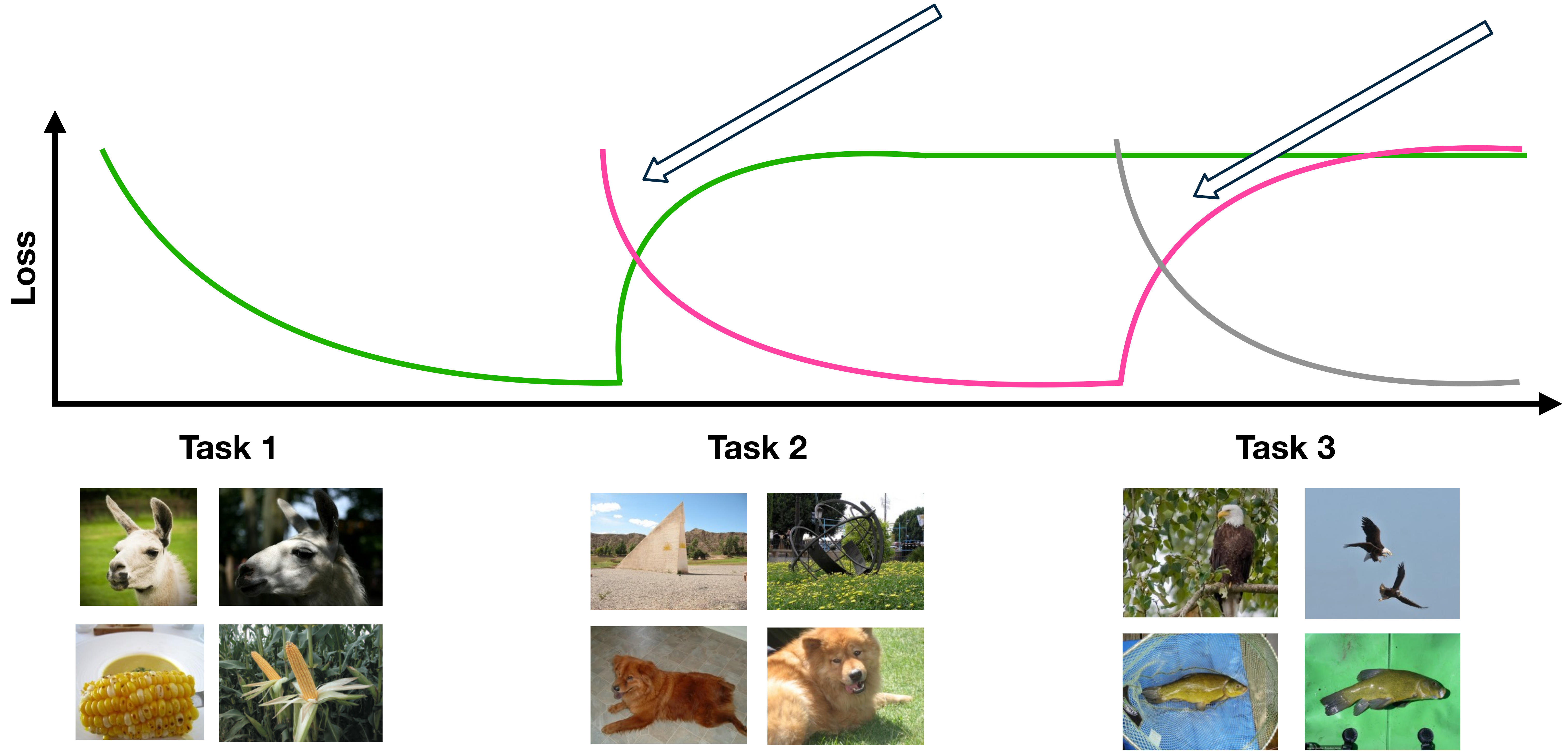
2) time to go to = lib. is too long.

Ray Solomonoff's notes on Ross Ashby's talk, Dartmouth 1956

Why does catastrophic interference/forgetting occur?

A popular example

Catastrophic forgetting



***Key assumption**:*
no access to/revisiting of prior "task" data!

Optimization: risk & losses

What we would like to generally do is minimize the following scenario:

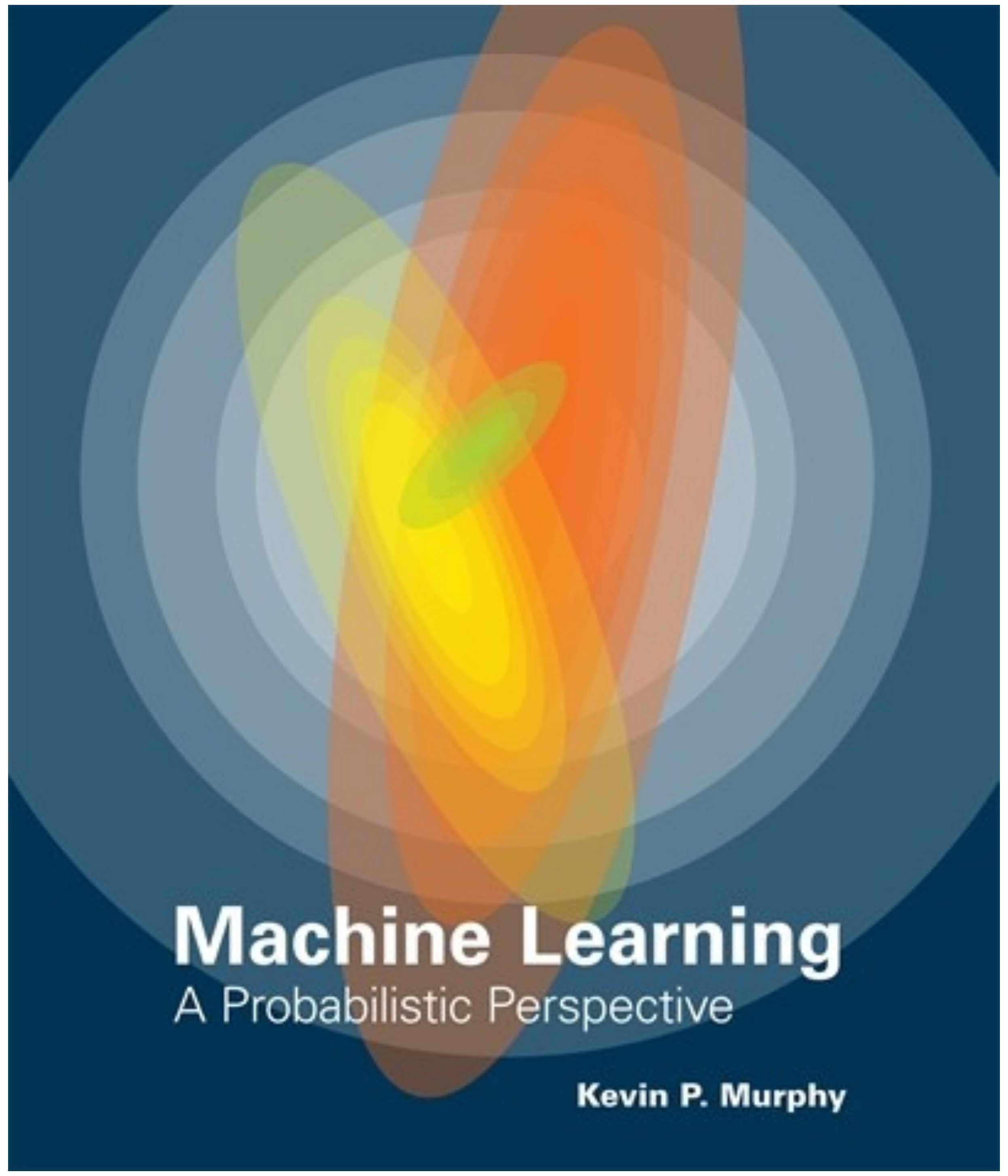
Find a hypothesis or decision procedure: $\delta : \mathscr{X} \rightarrow \mathscr{A}$

and define the risk or expected loss as:

$$R(\theta^*, \delta) = \mathbb{E}_{p(\tilde{D}|\theta^*)}\left[L(\theta^*, \delta(\tilde{D}))\right]$$

Where $\tilde{D}$ is data from the true distribution,

represented by parameter θ^*



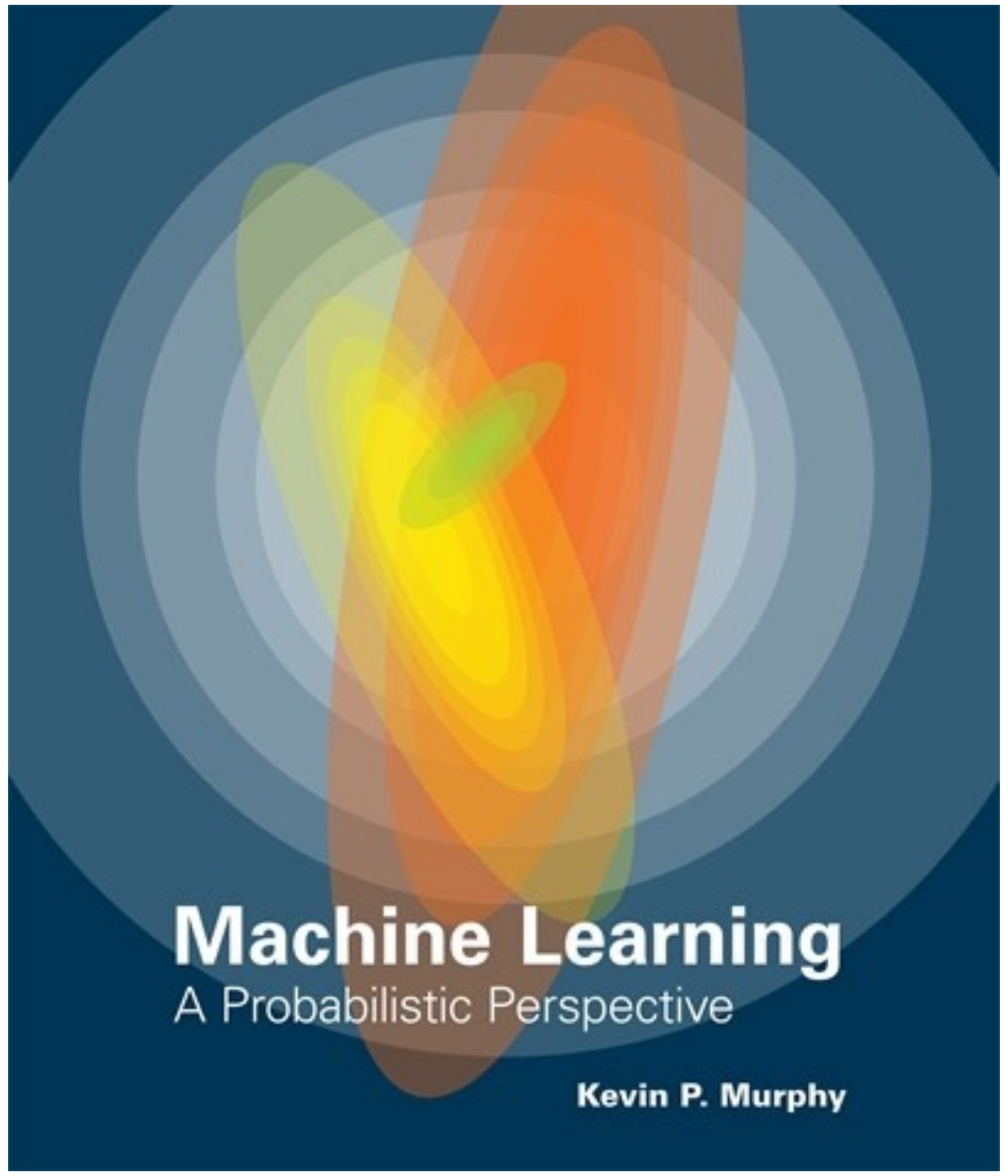
Pages 197-209

Optimization: risk & losses

$$R(\theta^*, \delta) = \mathbb{E}_{p(\tilde{D}|\theta^*)} \left[L(\theta^*, \delta(\tilde{D})) \right]$$

The challenges:

- Cannot actually compute above risk
  (usually don't know the true distribution)
- Besides: if we think of e.g. binary classification, i.e. a
  0-1 measure, it can be hard to optimize



Pages 197-209

Optimization: risk & losses

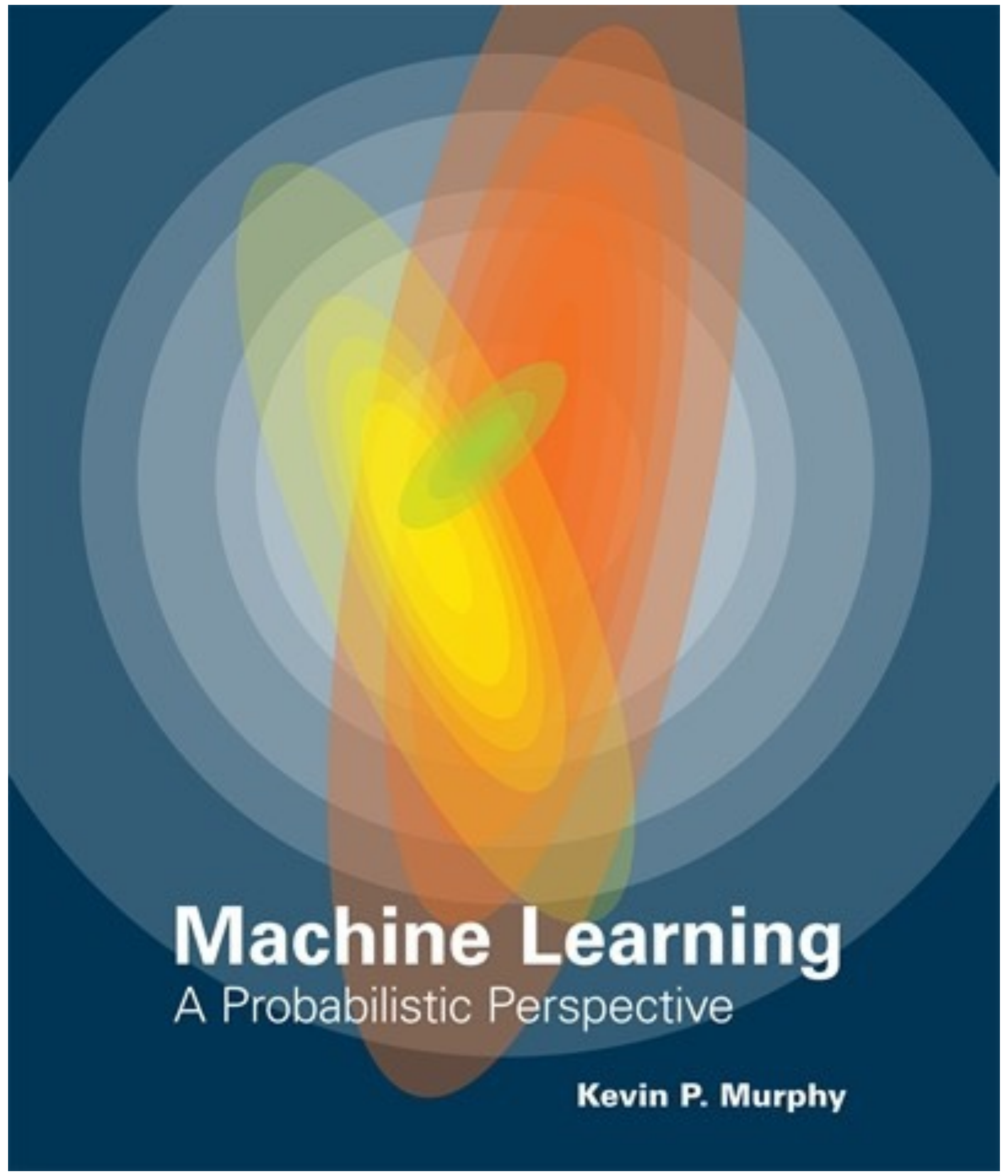
$$R(\theta^*, \delta) = \mathbb{E}_{p(\tilde{D}|\theta^*)}\left[L(\theta^*, \delta(\tilde{D}))\right]$$ instead:

$$R(p^*, \delta) = \mathbb{E}_{(x,y)\sim p^*}\left[L(y, \delta(x))\right]$$

But we can look at the true but unknown response and our predictions $\delta(x)$ given an input x.

We then further use empirical estimates:

$$R_{emp}(D, \delta) = 1/N \sum_{i=1}^{N} L(y_i, \delta(x_i))$$



Pages 197-209

Optimization: risk & losses

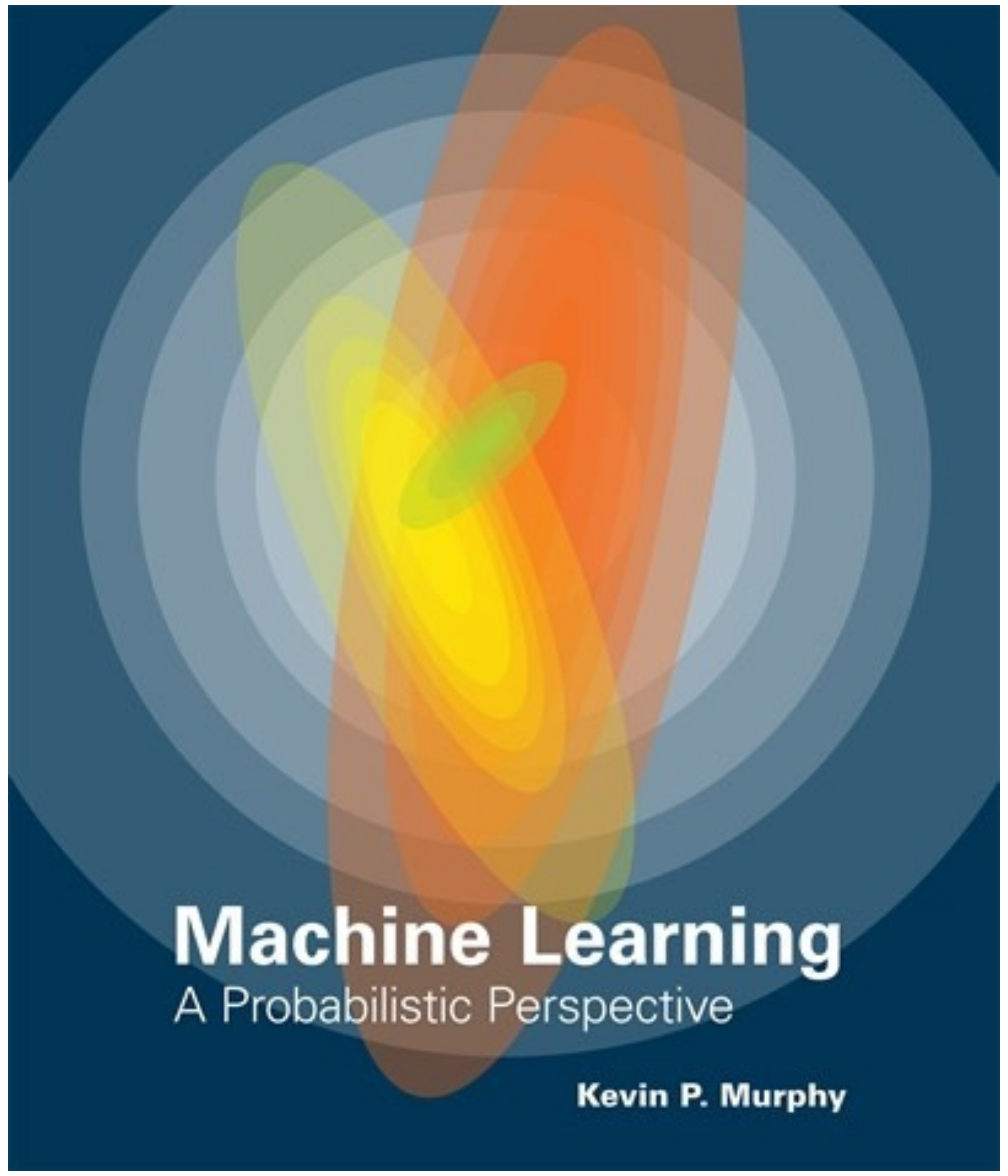
$$R_{emp}(D, \delta) = 1/N \sum_{i=1}^{N} L(y_i, \delta(x_i))$$

We then usually chose a loss function, e.g. MSE:

$$L(y, \delta(x)) = (y - \delta(x))^2$$

or similarly an unsupervised reconstruction surrogate:

$$L(y, \delta(x)) = ||x - \delta(x)||_2^2$$



Pages 197-209

Optimization: gradient descent

There are various optimization algorithms, the most popular ones are perhaps:
(Stochastic) gradient descent (SGD) and expectation maximization (EM)

Let us consider (S)GD here, as the “workhorse” underlying a lot of deep learning:

- Simple form: 1st order optimization to find a minimum of a differentiable function
- Achieved by iteratively taking (small) steps in the gradient direction of a function f in the direction in which it decreases the fastest:

$$x_{n+1} = x_n - \lambda \nabla f(x_n) \quad where \quad f(x_0) \geq f(x_1) \geq \ldots \geq f(x_n)$$

Optimization: gradient descent

We can transfer the SGD concept to the idea of parameters and losses:

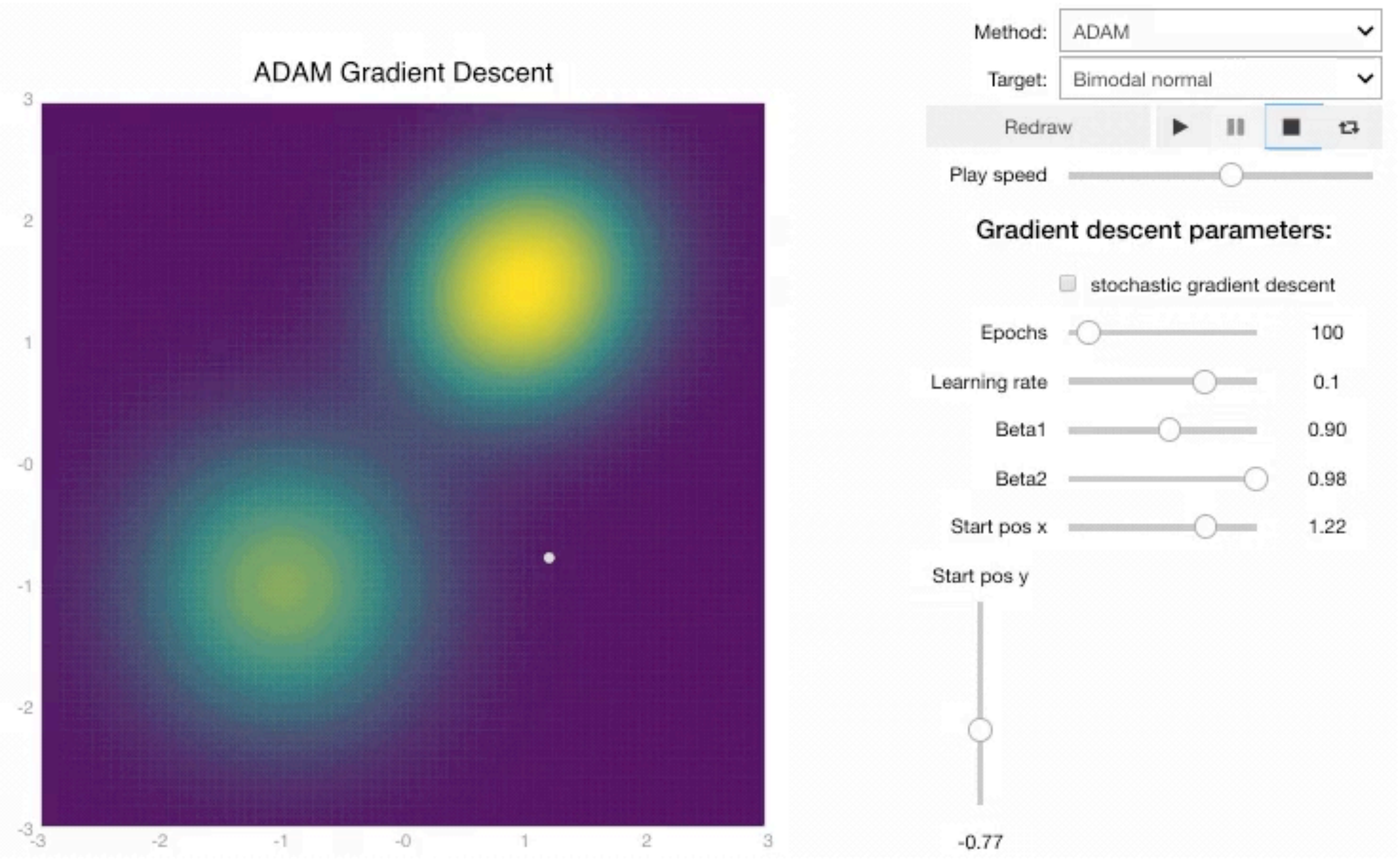
$$L(\theta) = 1/N \sum_{i=1}^{N} L_i(\theta)$$

Then iterative updates become (where in neural nets we backpropagate gradients):

$$\theta \leftarrow \theta - \lambda \nabla L(\theta) = \theta - \lambda/N \sum_{i}^{N} \nabla L_i(\theta)$$

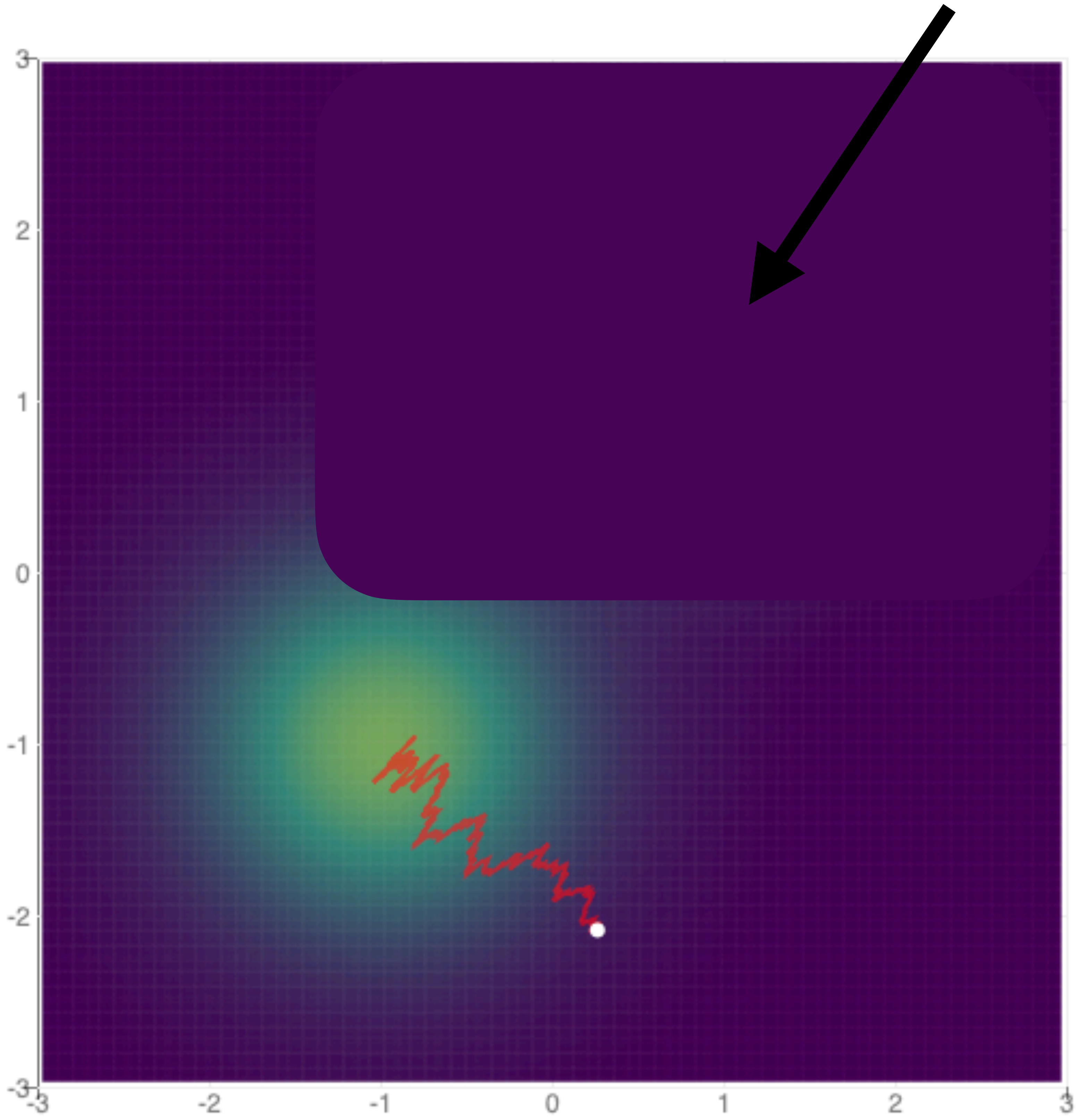
Let us talk about gradient estimates, stochasticity, step sizes, and ultimately forgetting

Forgetting & (stochastic) gradient descent



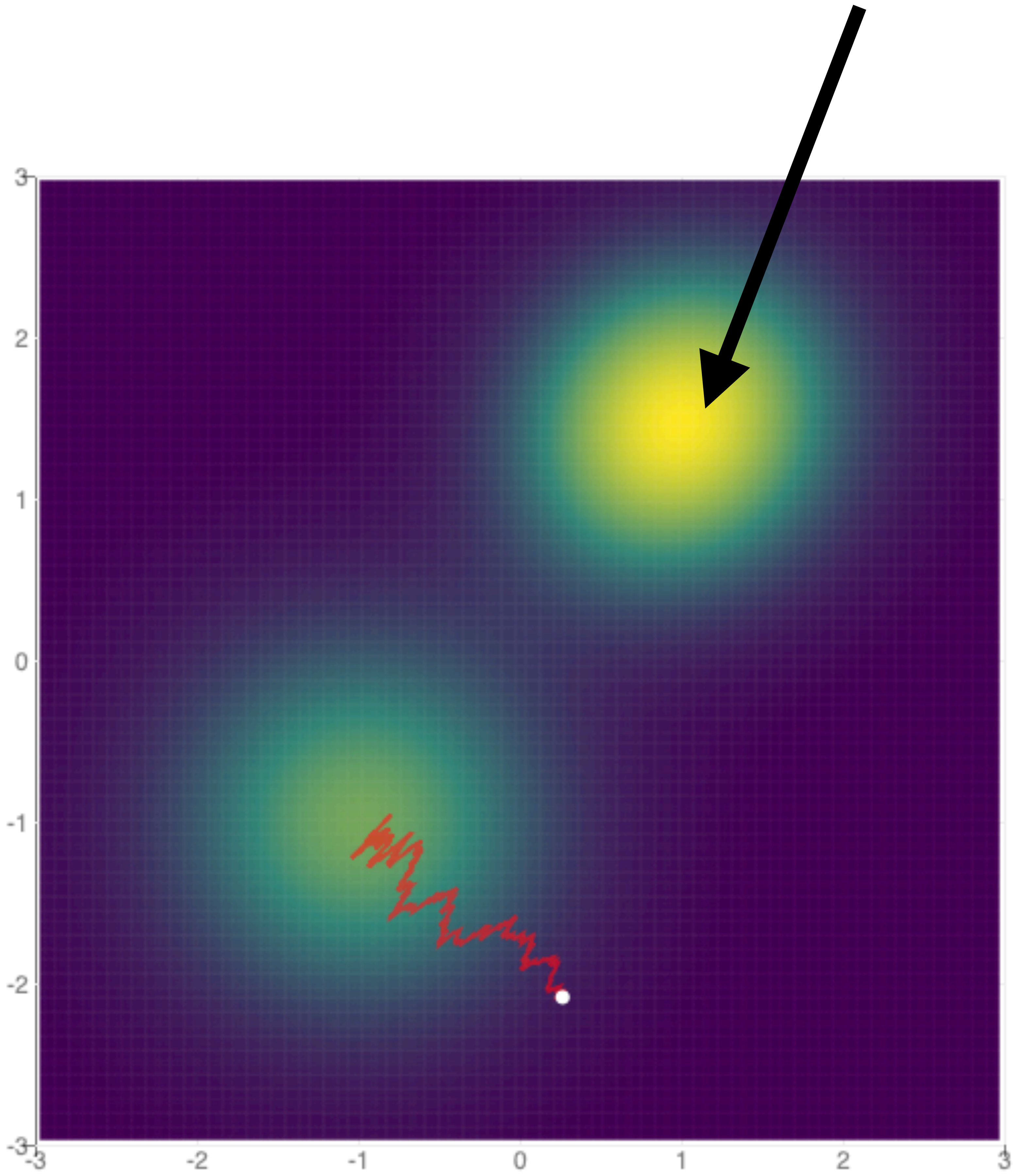
Forgetting & (stochastic) gradient descent

Assume the previous extremum wasn't there in "task" 1



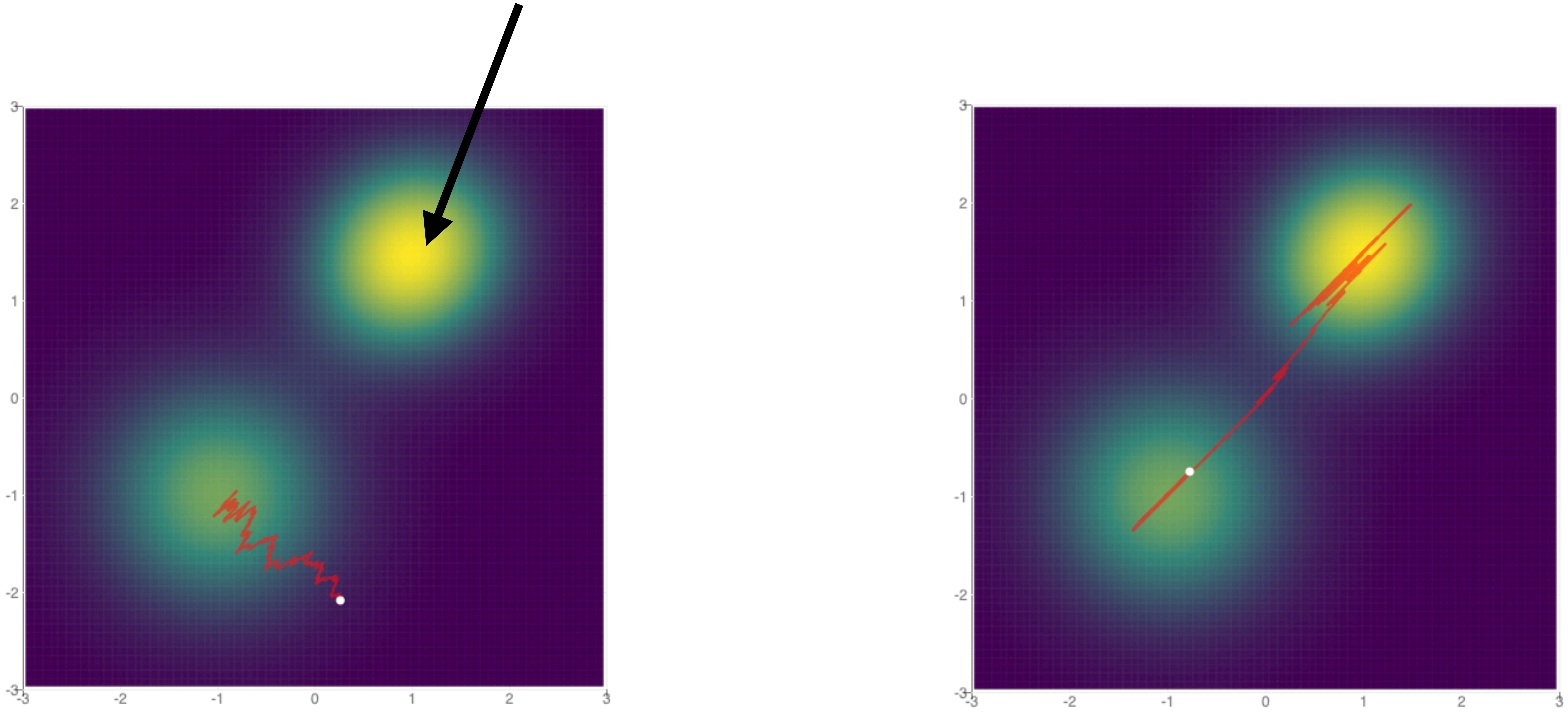
Forgetting & (stochastic) gradient descent

But now it gets added because new data is observed



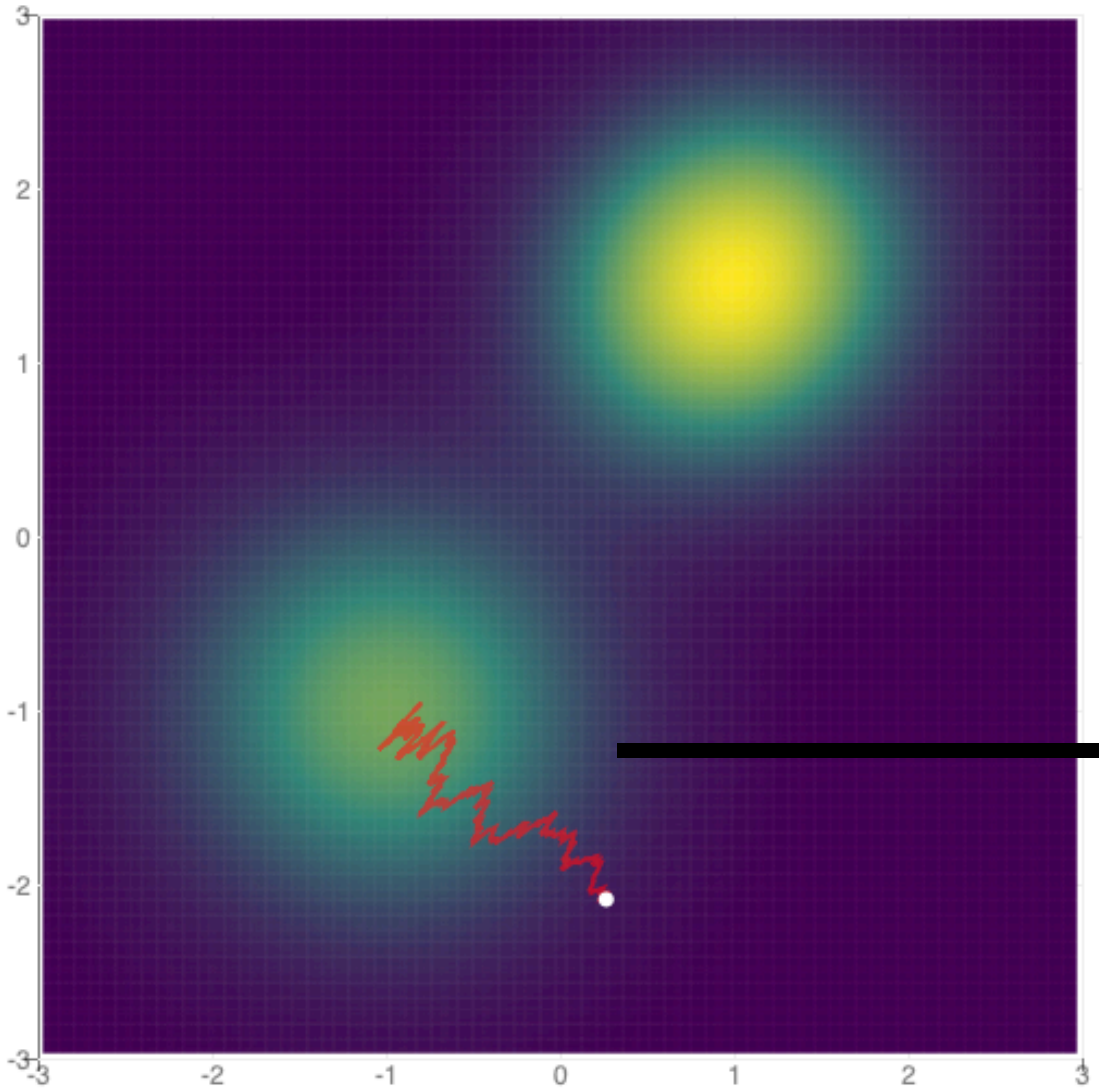
Forgetting & (stochastic) gradient descent

But now it gets added because new data is observed & noise is very large



Forgetting & (stochastic) gradient descent

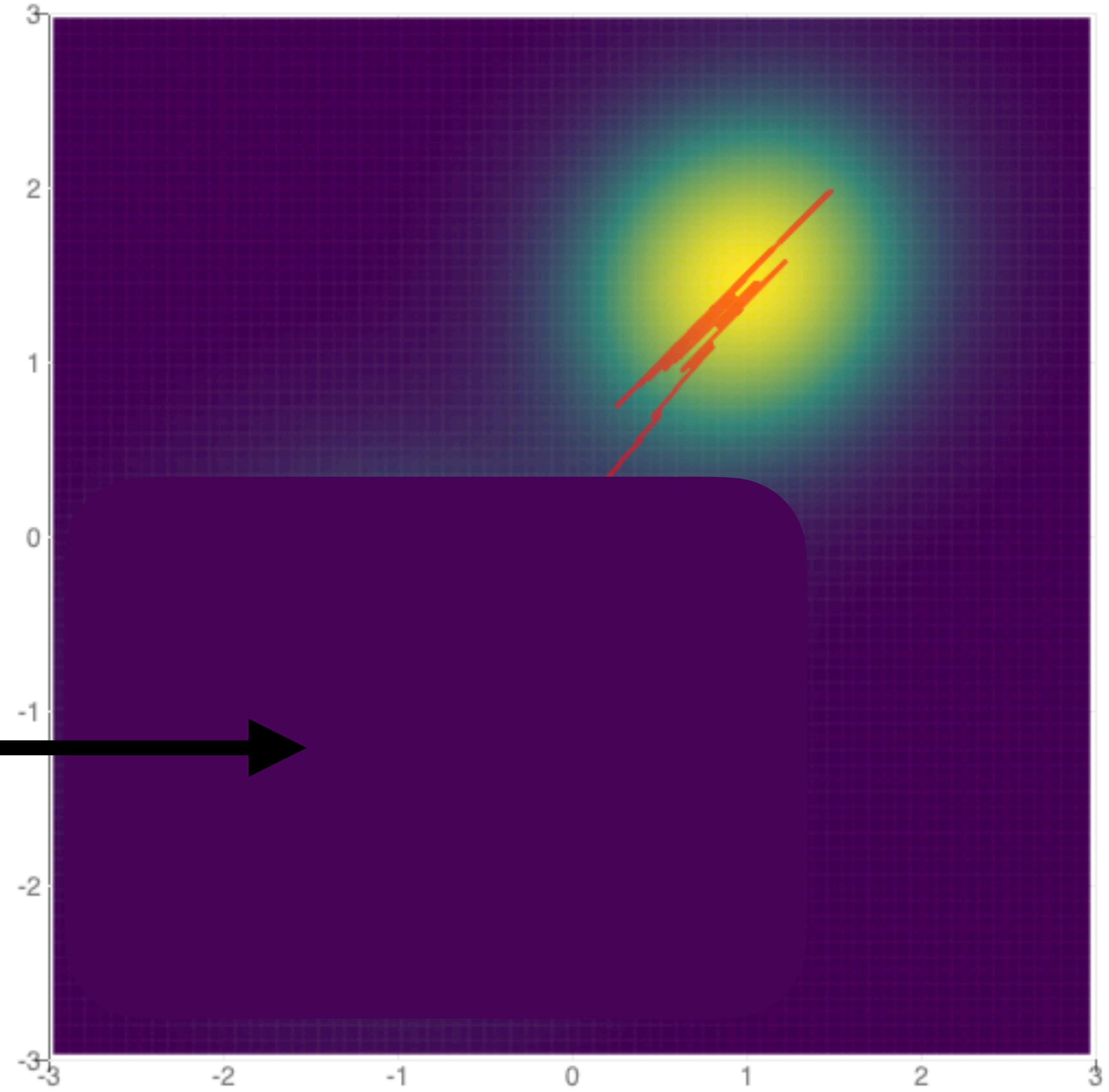
But now it gets added because new data is observed & noise is very large



Or prior data

is no longer

accessible



The stability - plasticity (sensitivity) dilemma

What we are essentially interested in is the so called stability - plasticity (or sensitivity) dilemma (Hebb, “The organization of behavior”, 1949).

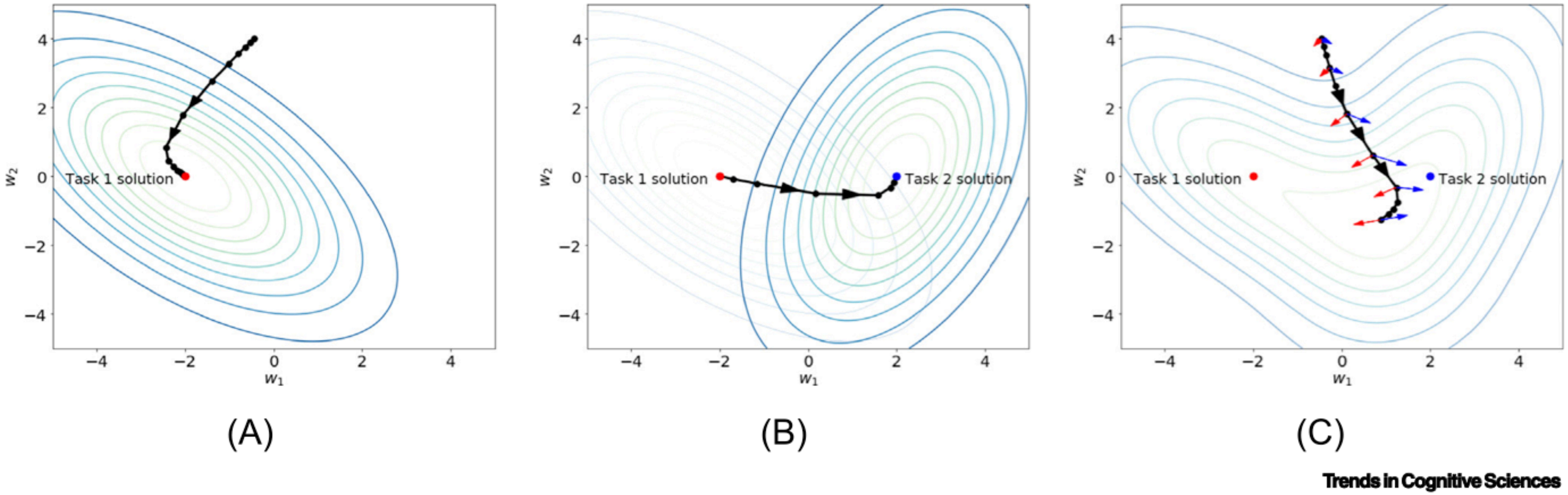


Figure 3. Illustrations of Gradient Descent Optimization for Different Tasks. (A) The trajectory taken by gradient descent optimization when minimizing a loss corresponding to a single task. (B) The optimization trajectory when subsequently training the same model on a second task. (C) The trajectory taken when using the total loss from both tasks (black) and the gradients from each individual task at multiple points during optimization (red and blue). See Box 2 for more detailed discussion.

Hadsell et al, “Embracing Change: Continual Learning in Deep Neural Networks”, Trends in Cognitive Sciences 24:12, 2020

The stability - plasticity (sensitivity) dilemma

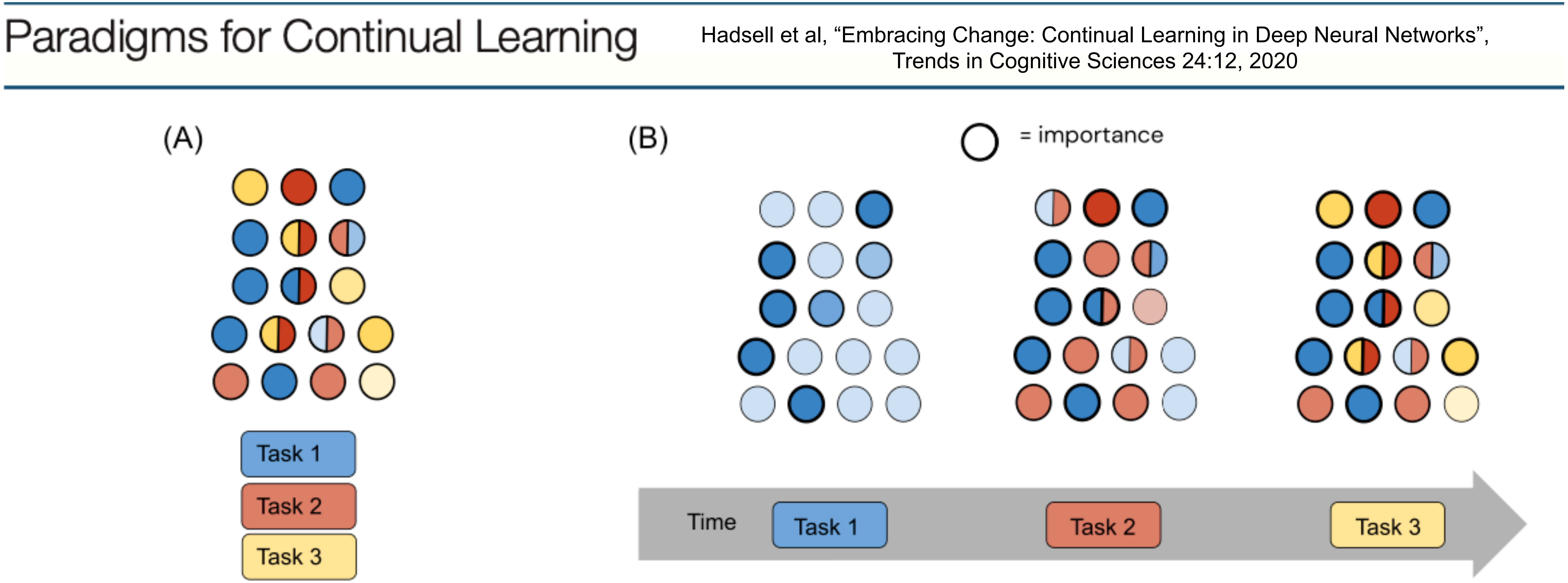
Old Problems,
Old Ideas

"There exists in the mind of man a block of wax … harder, moister, and having more or less of purity in one than another… **the soft are good at learning, but apt to forget; and the hard are the reverse**"

– Plato, Theaetetus, ~369 BCE

Lack of past data access & catastrophic forgetting from the optimization perspective

How do we prevent forgetting?

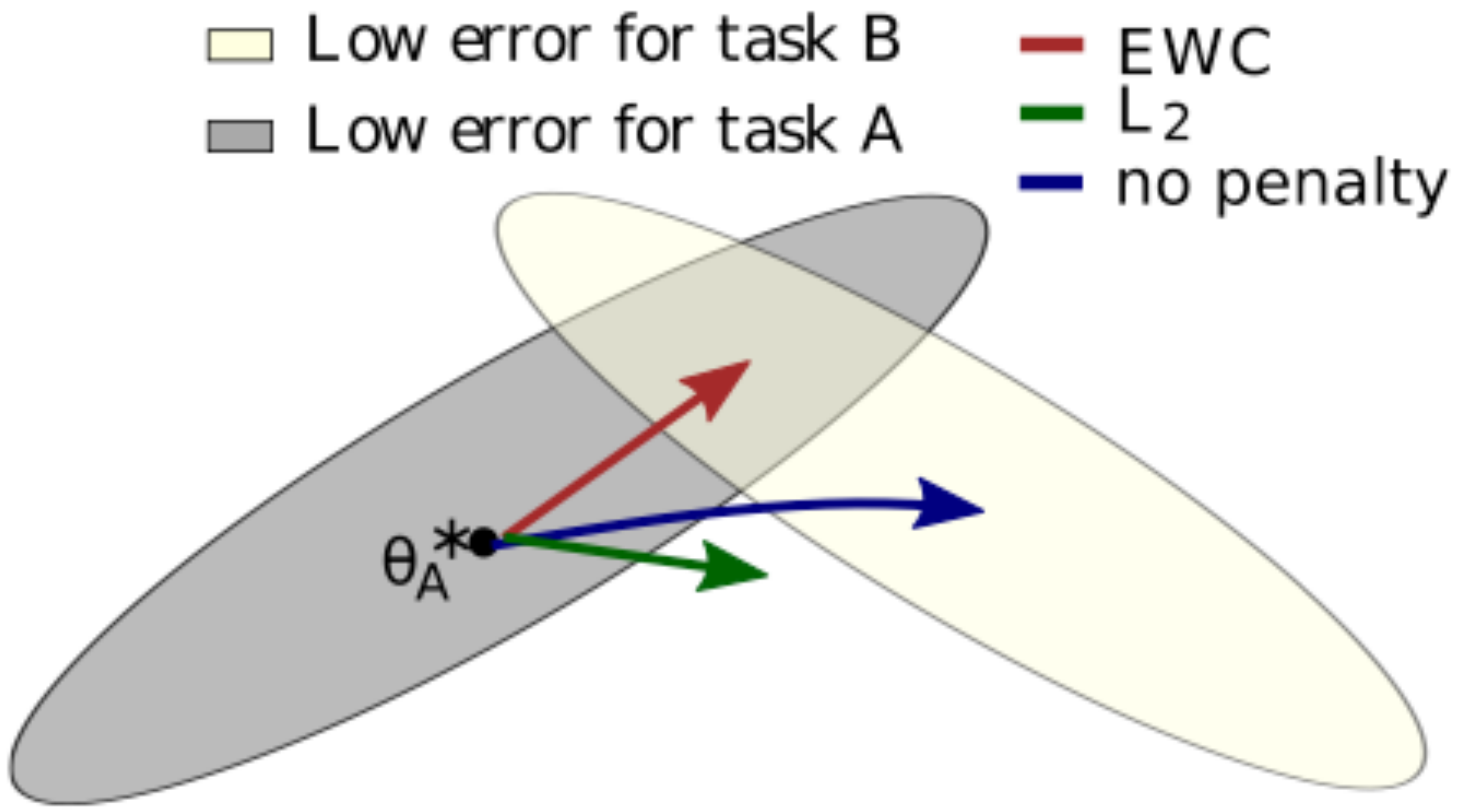


We will look at three perspectives on forgetting.

Let's start with the one directly related to stability vs. plasticity

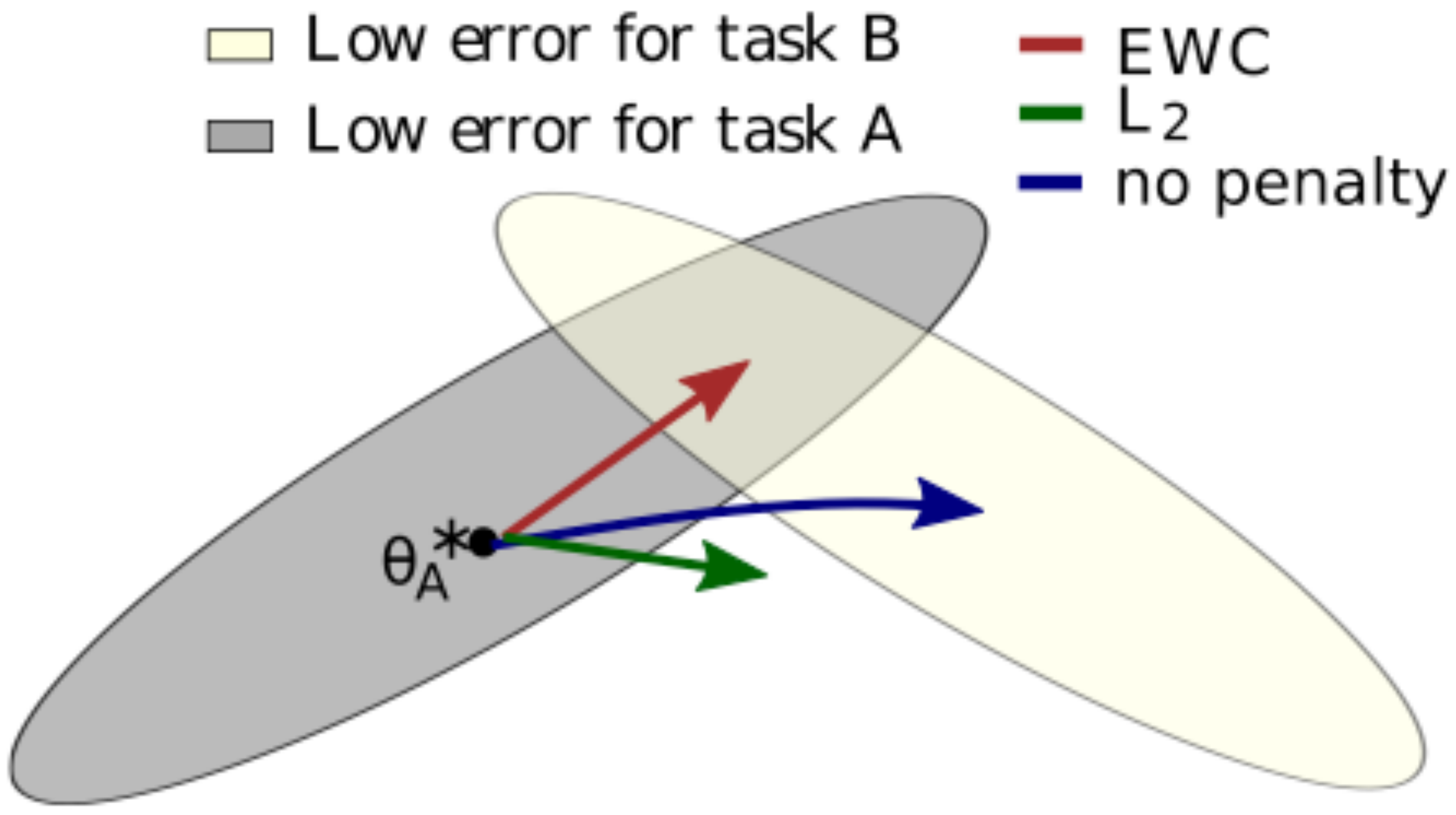
Variant A. Finding & regularizing important parameters

Elastic weight consolidation



Kirkpatrick et al, “Overcoming catastrophic forgetting in neural networks”,
PNAS 114(13), 2017

Elastic weight consolidation

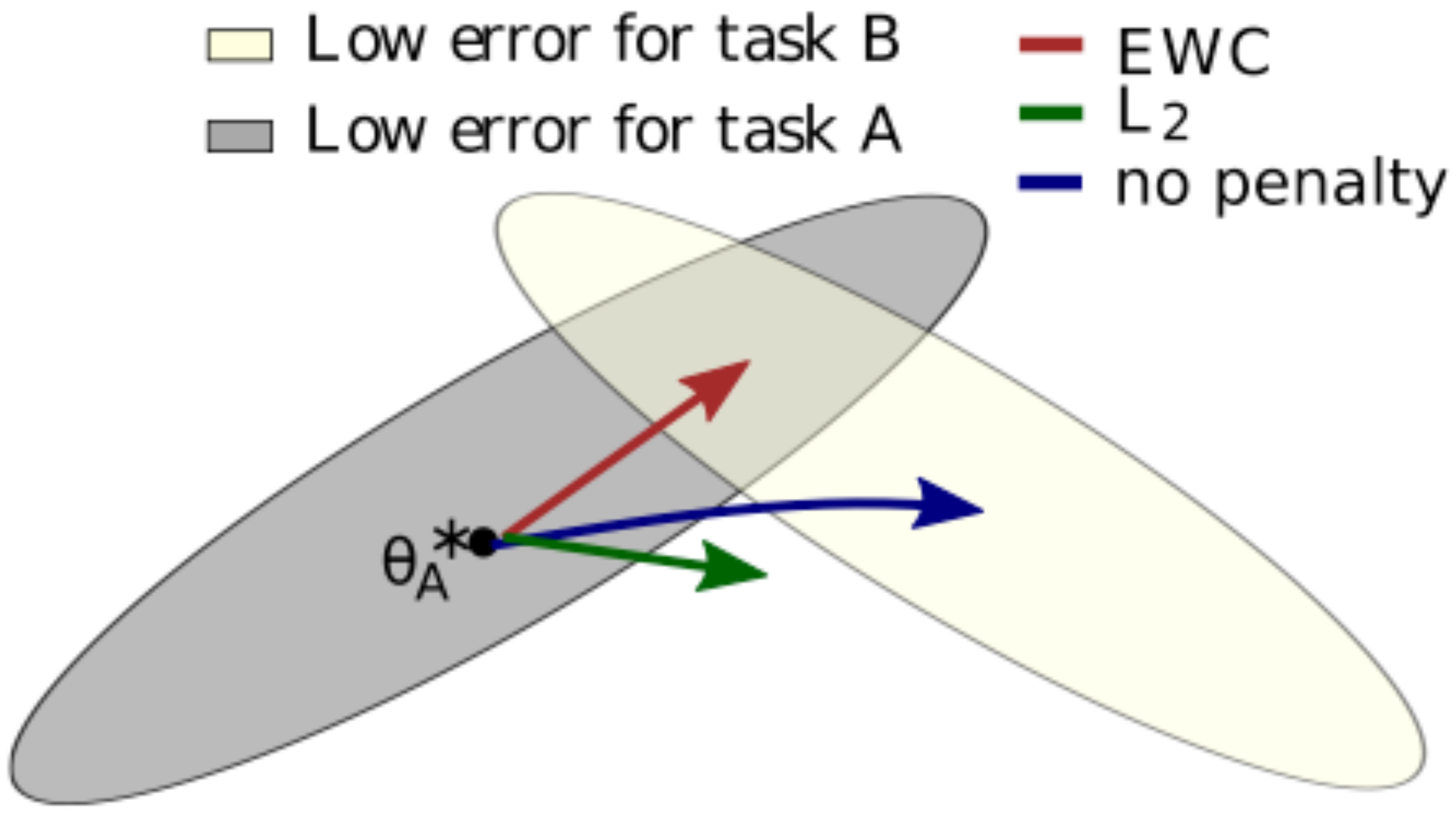


$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta^*_{A,i})^2$$

Instead of naively continuing to optimize task B, we can impose a penalty on previously learned parameters.

Kirkpatrick et al, "Overcoming catastrophic forgetting in neural networks", PNAS 114(13), 2017

Elastic weight consolidation



$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

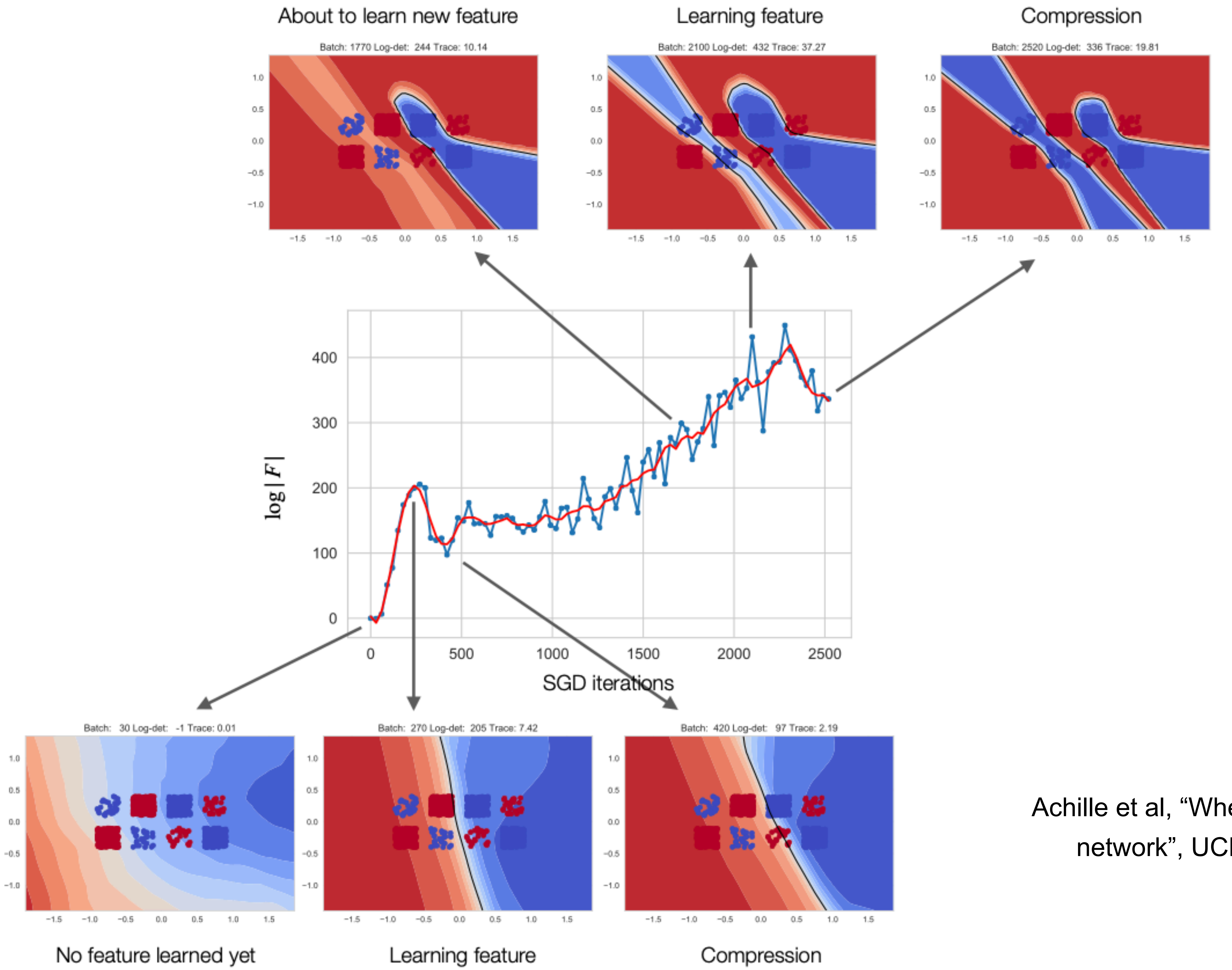
Instead of naively continuing to optimize task B, we can impose a penalty on previously learned parameters.

We will need to find a matrix F that tells us which parameters are most important for task A.

Example: Fisher information (related to natural gradients. (https://agustinus.kristia.de/techblog/2018/03/11/fisher-information/ has a nice summary)

Kirkpatrick et al, "Overcoming catastrophic forgetting in neural networks", PNAS 114(13), 2017

Fisher information & parameter importance intuition



Achille et al, “Where is the information in a deep neural network”, UCLA-TR:190005, 2019

A similar idea: Synaptic Intelligence

Key idea: change (with time t) in loss is well approximated by the gradient (g):

$$L(\theta(t) + \delta(t)) - L(\theta(t)) \approx \sum_k g_k(t)\delta_k(t)$$

A similar idea: Synaptic Intelligence

Key idea: change (with time t) in loss is well approximated by the gradient (g):

$$L(\theta(t) + \delta(t)) - L(\theta(t)) \approx \sum_k g_k(t)\delta_k(t)$$

Each parameter change $\delta_k(t) = \theta_k'(t)$ contributes amount $g_k(t)\delta_k(t)$ to the change in total loss.

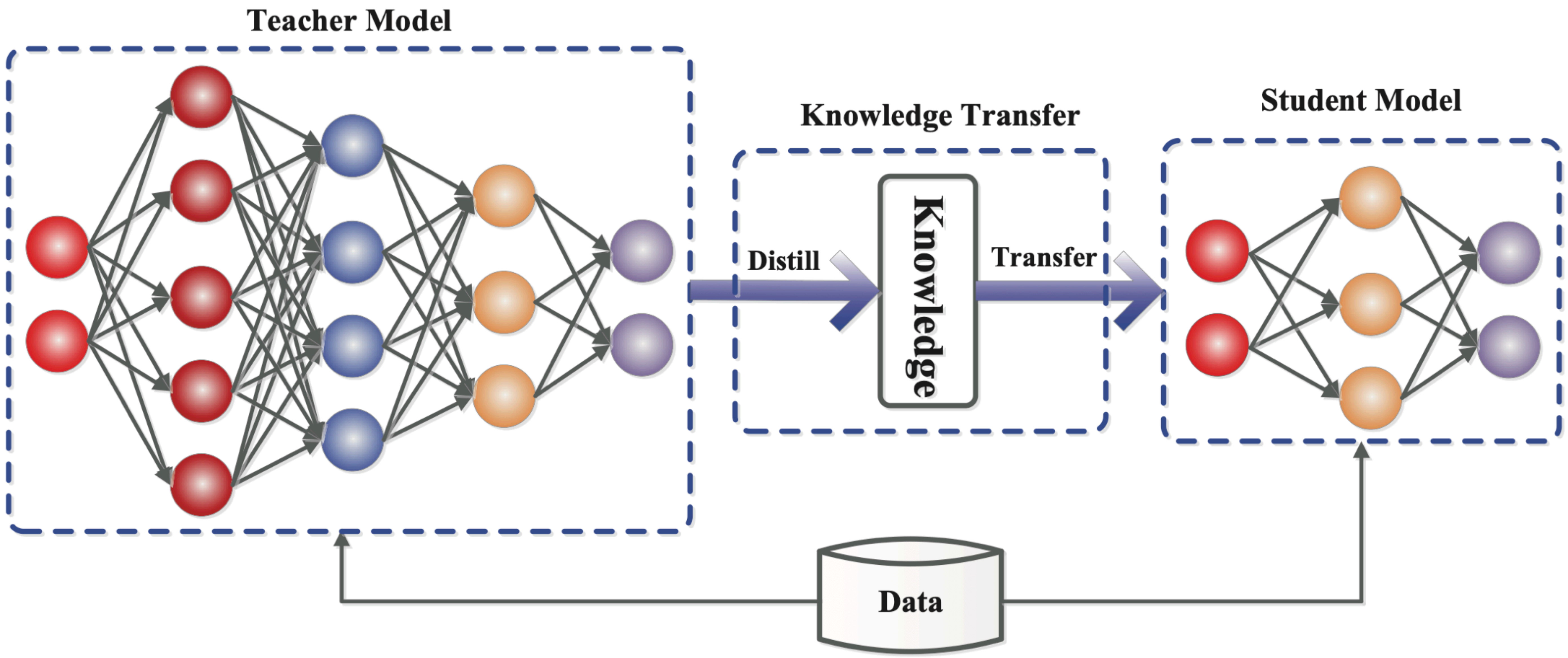
Assign importance to each parameter according to the monitored trajectory and formulate a similar penalty to EWC again (with different importance measure).

Variant B. Maintaining (input-output) relationships

Alternative stability-plasticity: Knowledge distillation

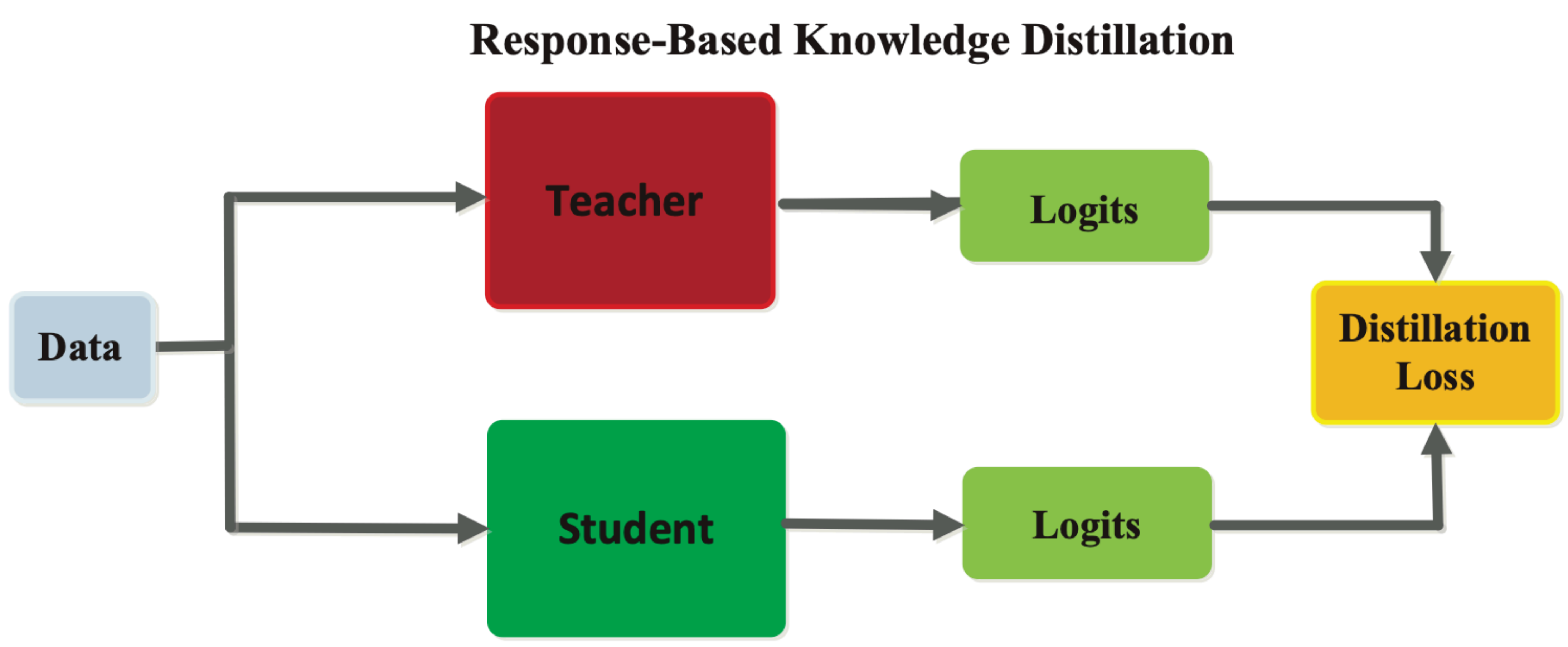
Alternatively, we know that if we have enough parameters, there are many potential solutions to produce the same input-output relationships.

Key idea: Let's try to maintain a task's input-output relationship

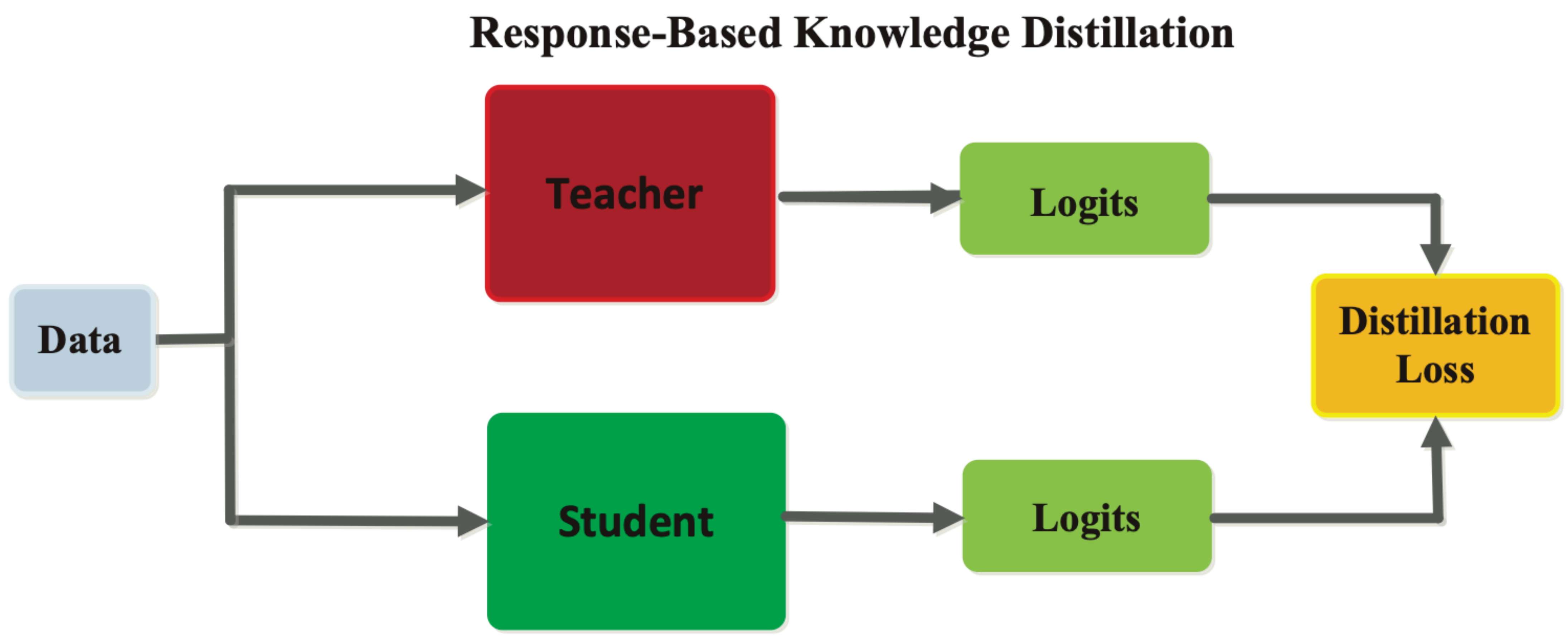


Gou et al, "Knowledge Distillation: A survey", International Journal of Computer Vision 129, 2021

Alternative stability-plasticity: Knowledge distillation

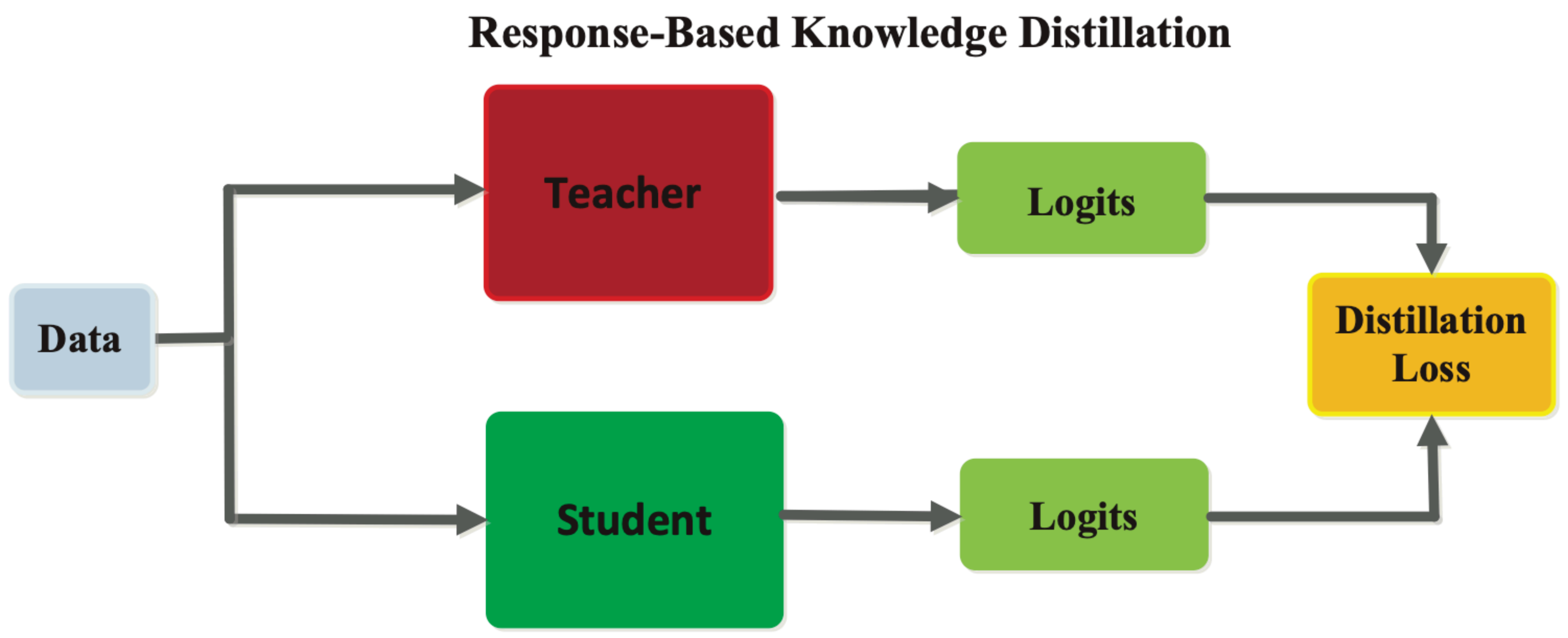


Alternative stability-plasticity: Knowledge distillation



Special case: classifier logits (Hinton et al, "Distilling the Knowledge in A Neural Network", NeurIPS14 Deep Learning Workshop)

Alternative stability-plasticity: Knowledge distillation



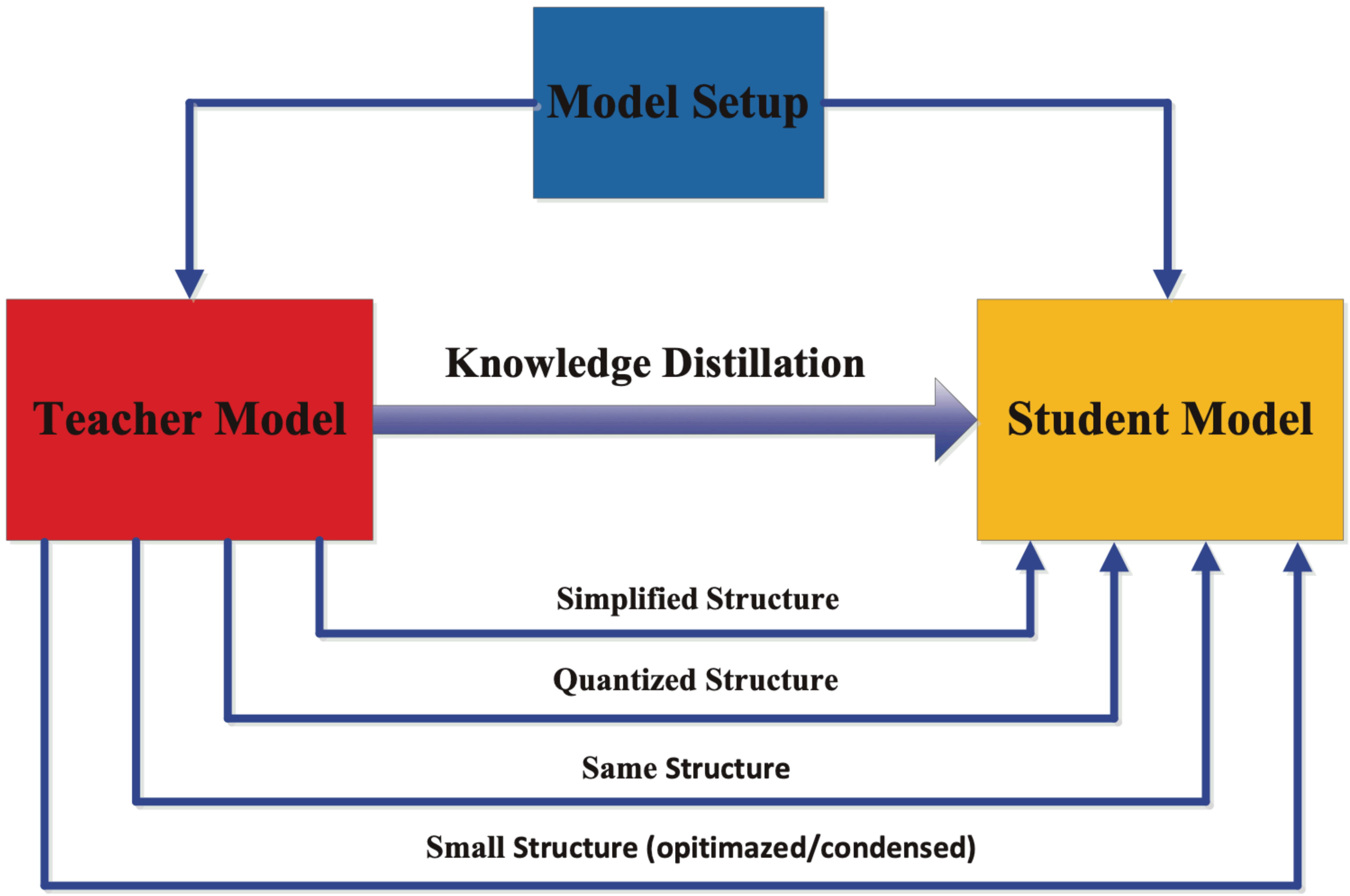
$$\frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{exp(z_i/T)}{\sum_j exp(z_j/T)} - \frac{exp(v_i/T)}{\sum_j exp(v_j/T)}\right)$$

Special case: classifier logits (Hinton et al, “Distilling the Knowledge in A Neural Network”, NeurIPS14 Deep Learning Workshop)

In essence: make sure that the distance between z & v of 2 models is minimized, or more generally minimizing the KL divergence over the 2 probability distributions.

Alternative stability-plasticity: Knowledge distillation

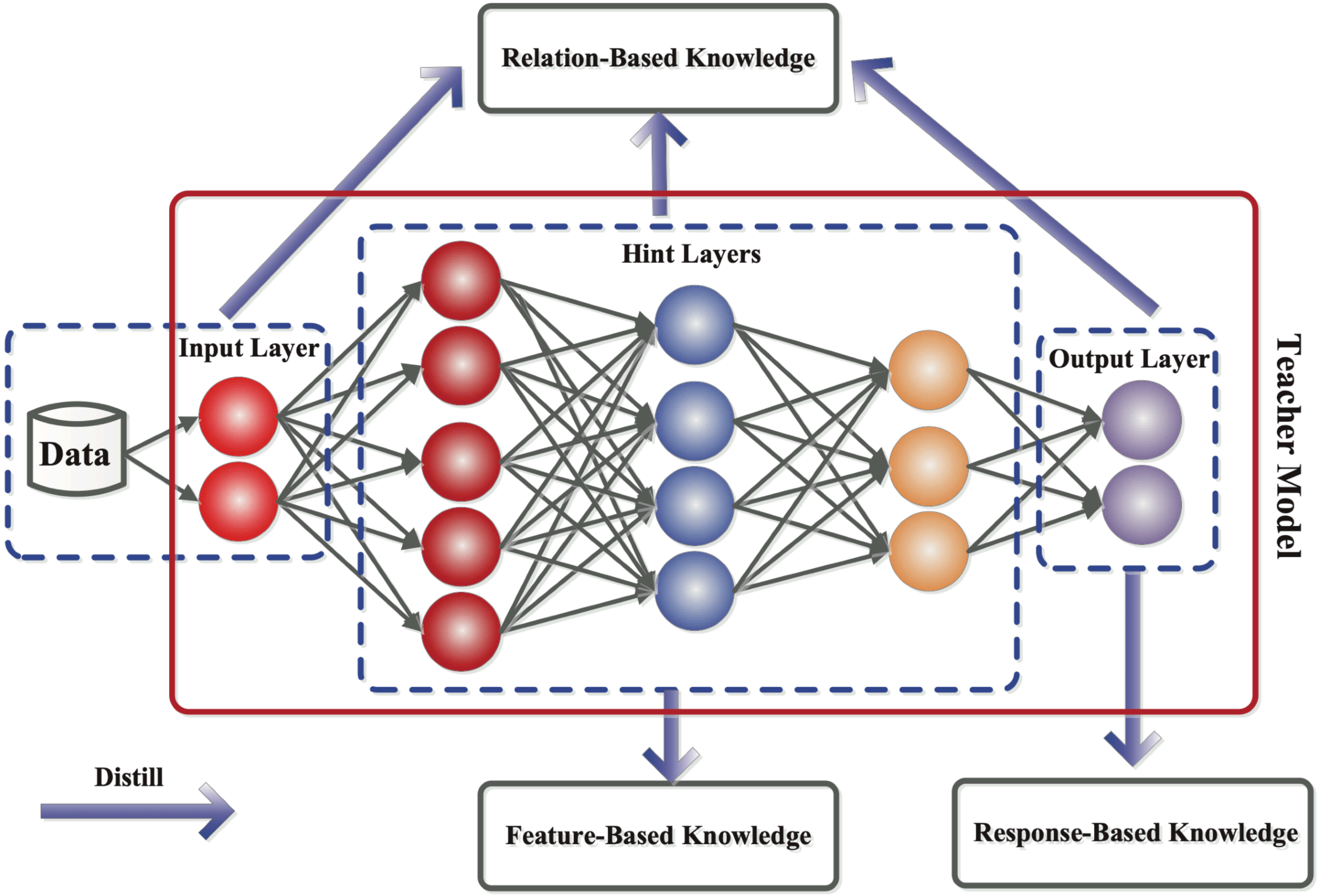
Apart from continual learning (on the next slides), why distill?



Gou et al, “Knowledge Distillation: A survey”, International Journal of Computer Vision 129, 2021

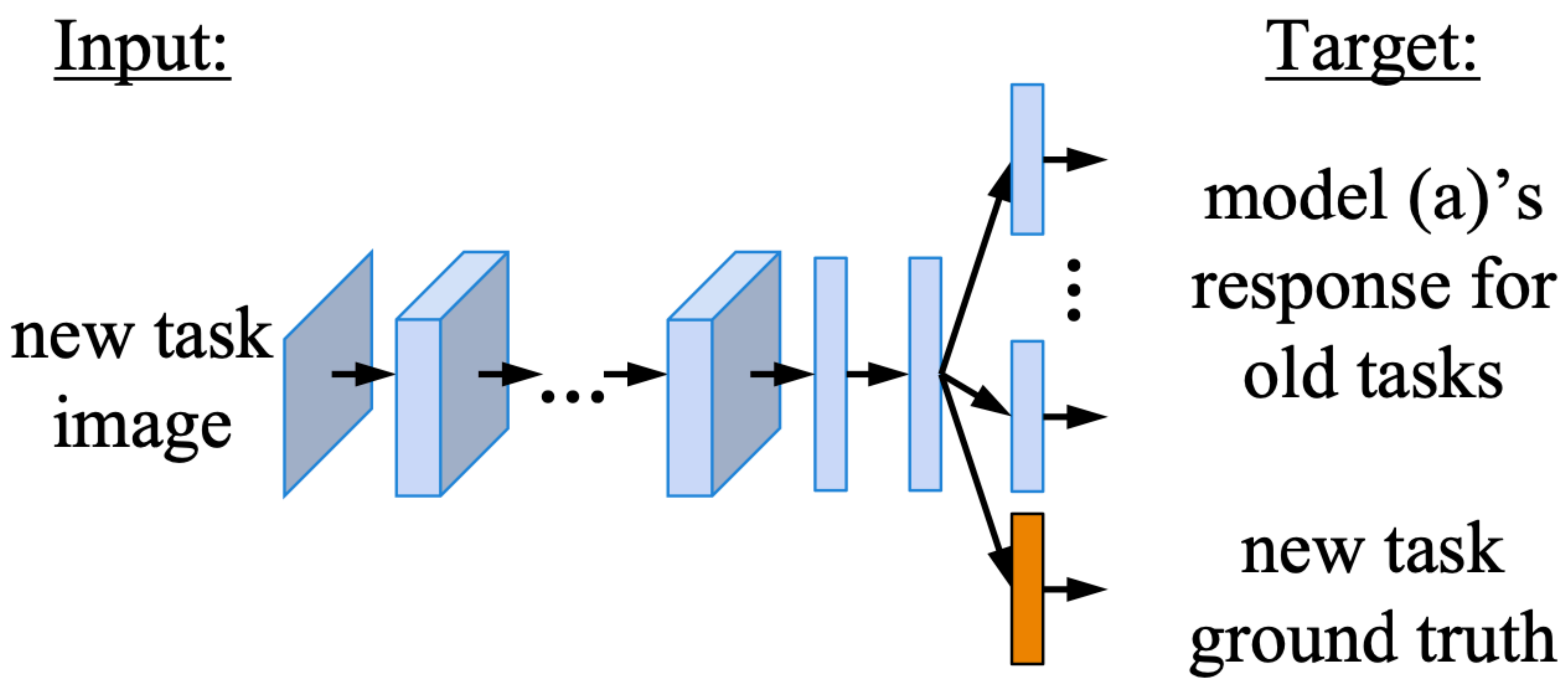
Alternative stability-plasticity: Knowledge distillation

We generally have various choices of what types of relationships we wish to distill (and how)



Gou et al, “Knowledge Distillation: A survey”, International Journal of Computer Vision 129, 2021

Knowledge distillation to alleviate forgetting

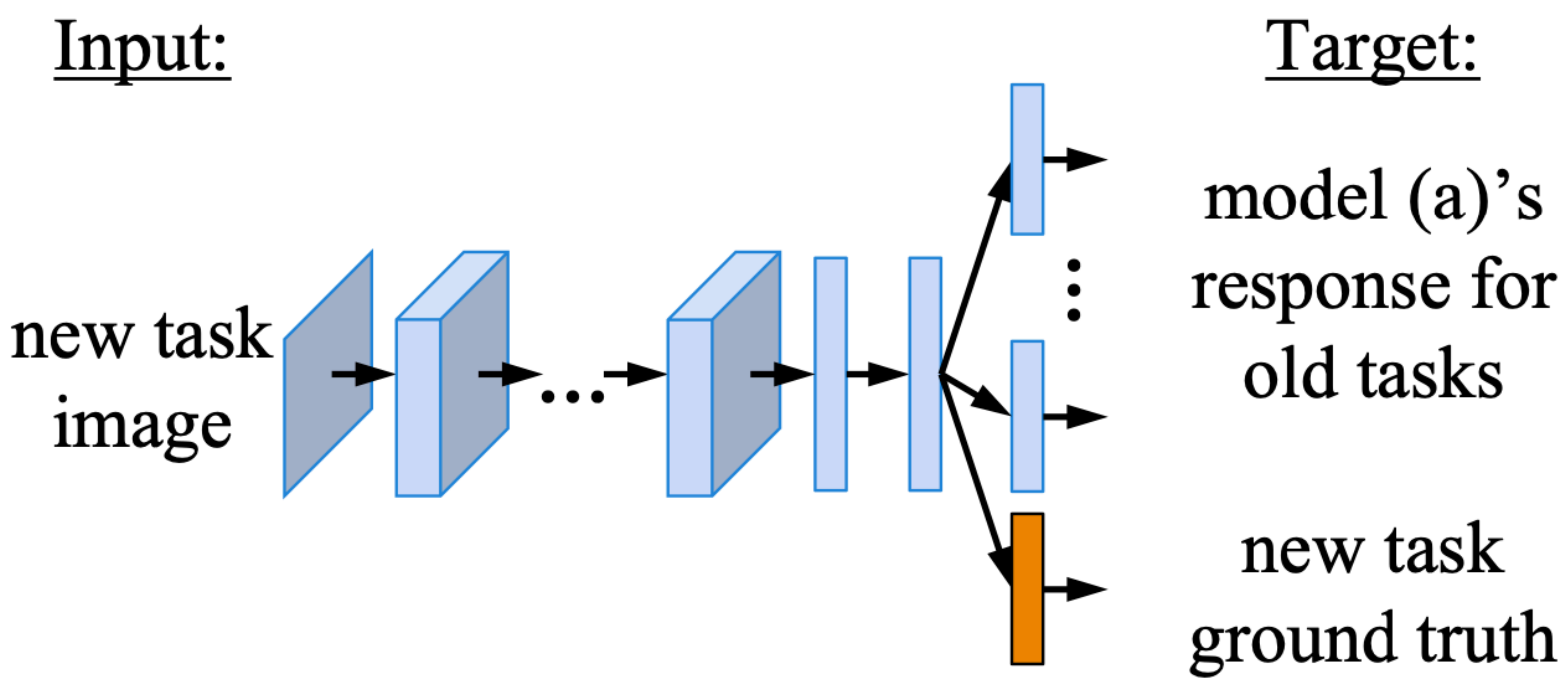


Learning without forgetting

(Li & Hoiem, "Learning without Forgetting", ECCV 2016)

Key idea: compute task "head" with new data and continue to preserve this input-output relationship, while learning a new task "head" simultaneously

Knowledge distillation to alleviate forgetting



LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

θ_o: task specific parameters for each old task

X_n, Y_n: training data and ground truth on the new task

Initialize:

$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*

Train:

Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*

Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*

$$\theta_s^*,\ \theta_o^*,\ \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\operatorname{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$$

Learning without forgetting

(Li & Hoiem, "Learning without Forgetting", ECCV 2016)

Key idea: compute task "head" with new data and continue to preserve this input-output relationship, while learning a new task "head" simultaneously

But knowledge is more than parameters. There are more ways to have "memory" than to regularize

Some early thoughts: parameters & data

Most definitely not the earliest, but very intuitive examples!

Ideas date back to at least the 70s, even the 50s.

Rehearsal

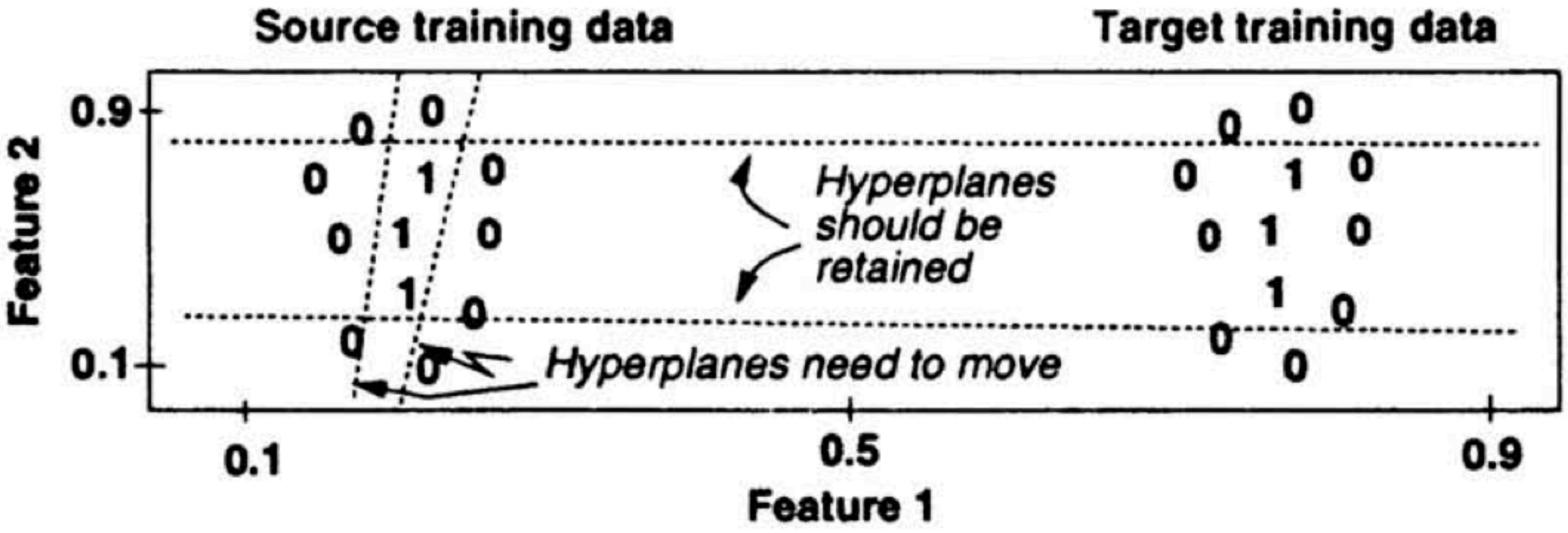
"The sequential acquisition of new data is incompatible with the gradual discovery of structure and can lead to *catastrophic interference* with what has previously been learned. In light of these observations, we suggest that the neocortex may be optimized for the gradual discovery of the shared structure of events and experiences, and that the hippocampal system is there to provide a mechanism for rapid acquisition of new information without interference with previously discovered regularities. After this initial acquisition, the hippocampal system serves as a teacher to the neocortex..."

McClelland et al, "Why there are complementary learning systems in the hippocampus and neocortex", Psychological Review 102, 1995 (see also Robins 1995)

Rehearsal: basic intuition

Assuming privacy is not a concern & that we have auxiliary memory:

some data is more relevant than other, can we retain a subset?

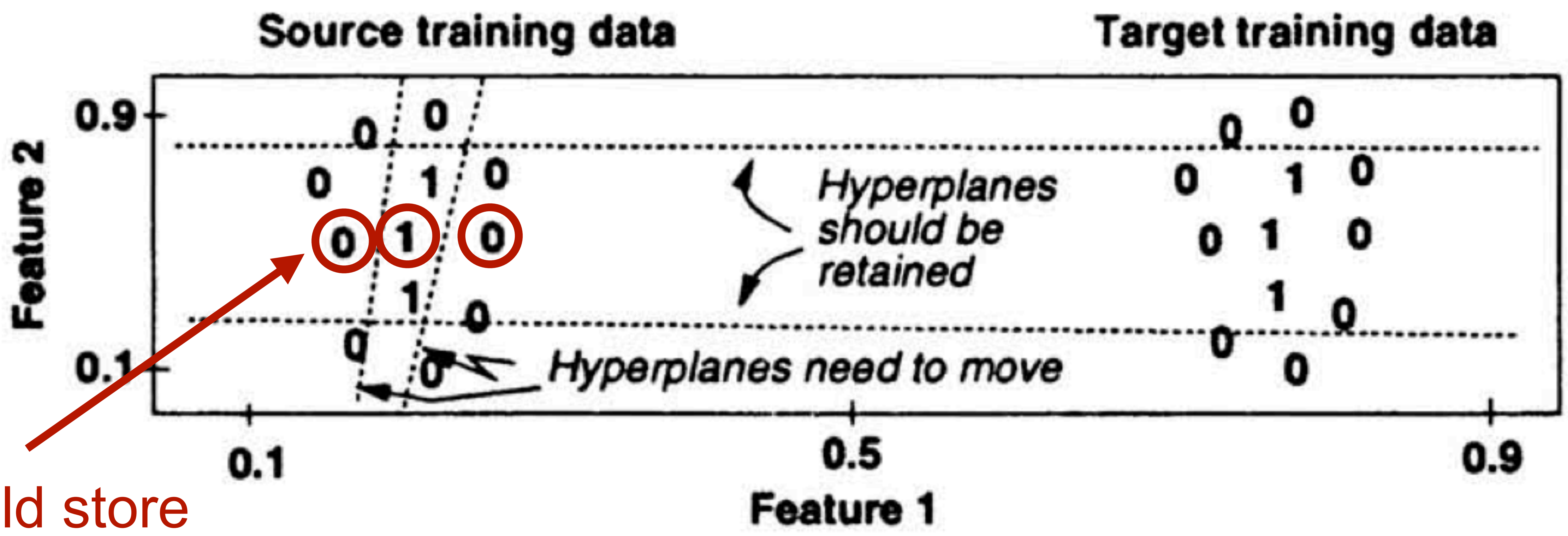


“Discriminability-Based Transfer between Neural Networks”, L. Y. Pratt, NeurIPS 1992

Rehearsal: basic intuition

Assuming privacy is not a concern & that we have auxiliary memory:

some data is more relevant than other, can we retain a subset?



Maybe we could store these few examples?

“Discriminability-Based Transfer between Neural Networks”, L. Y. Pratt, NeurIPS 1992

Let's start with an example to develop desiderata: iCaRL - incremental classifier & representation learning

Algorithm 1 iCaRL CLASSIFY

input x // image to be classified
require $\mathcal{P} = (P_1, \dots, P_t)$ // class exemplar sets
require $\varphi : \mathcal{X} \to \mathbb{R}^d$ // feature map
  for $y = 1, \dots, t$ **do**
    $\mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$ // mean-of-exemplars
  end for
  $y^* \leftarrow \operatorname{argmin}_{y=1,\dots,t} \|\varphi(x) - \mu_y\|$ // nearest prototype
output class label y^*

- Stores a subset of data in a fixed size memory buffer
- Classifies based on nearest class means
- Consecutively replaces parts of memory buffer with new examples

Rebuffi et al, "iCaRL: Incremental Classifier and Representation Learning", CVPR 2017

iCaRL: picking "exemplars"

Algorithm 4 iCaRL ConstructExemplarSet

input image set $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of exemplars
require current feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
  $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$ // current class mean
  for $k = 1, \ldots, m$ **do**
    $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$
  end for
  $P \leftarrow (p_1, \ldots, p_m)$
output exemplar set P

How is our memory buffer filled?

- Iteratively: one by one, based on “herding” (Welling ICML 2009)
- Pick exemplars to best approximate the overall mean
- For a size of k exemplars: loop k times

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL: replacing exemplars

Algorithm 5 iCaRL ReduceExemplarSet

input m // target number of exemplars
input $P = (p_1, \dots, p_{|P|})$ // current exemplar set
$P \leftarrow (p_1, \dots, p_m)$ // *i.e.* keep only first m
output exemplar set P

Our memory buffer is limited, how do we later remove samples?

- Memory buffer is a prioritized list
- Later picked exemplars for a task “weigh” less
- Simply cut and repopulate

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL: incremental training

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \dots, X^t$ // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \dots, s-1$ **do**
$q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y \neq y_i} \log(1 - g_y(x_i)) + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1 - q_i^y) \log(1 - g_y(x_i)) \Big]$$

that consists of *classification* and *distillation* terms.

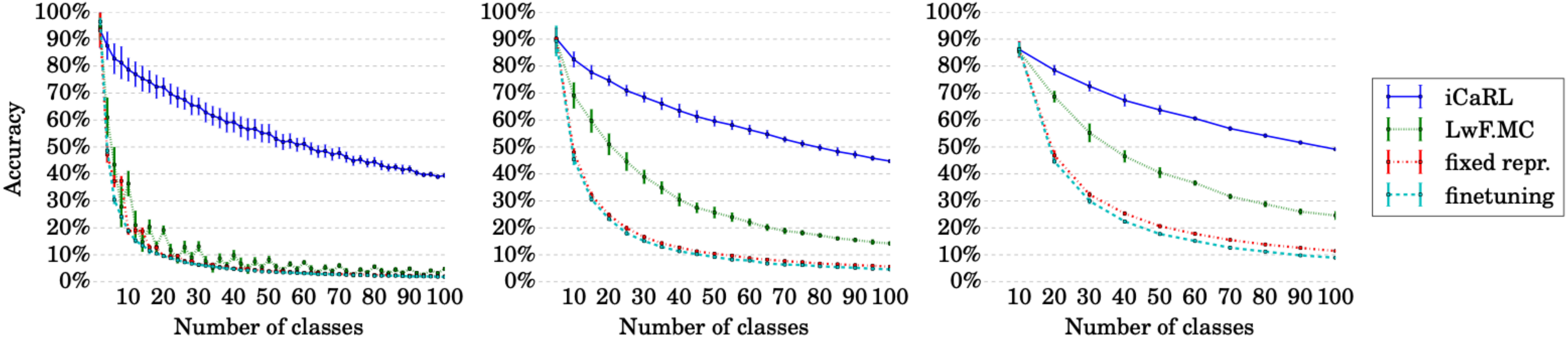
How do we train incrementally?

- Concatenate dataset with exemplars/interleave exemplars into training
- Pick new exemplars (not shown on the right) + replace existing
- Additionally use knowledge distillation

Rebuffi et al, "iCaRL: Incremental Classifier and Representation Learning", CVPR 2017

iCaRL & knowledge distillation

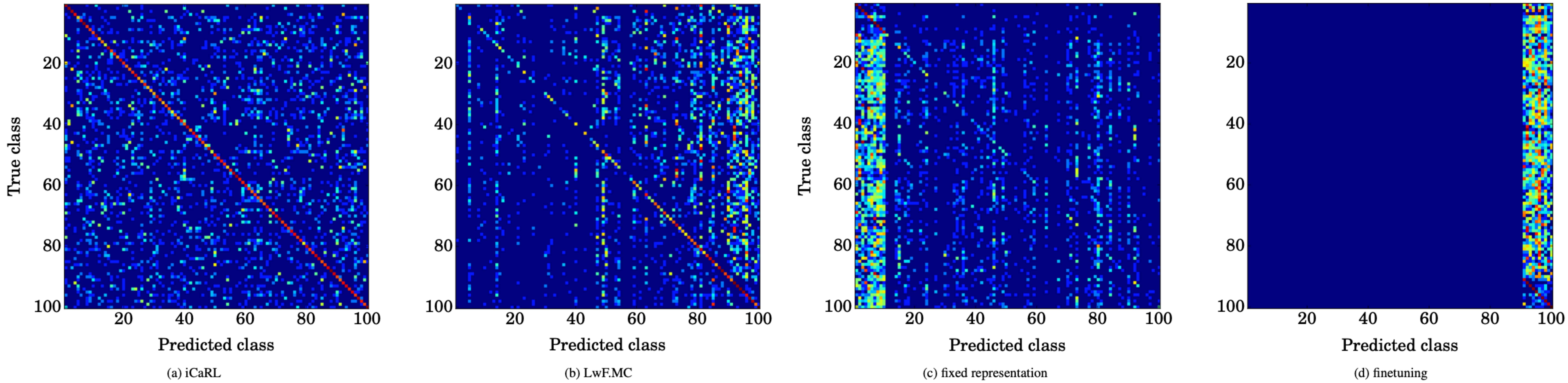
Example: incrementally learning CIFAR100 - exemplars are crucial



Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

iCaRL & knowledge distillation

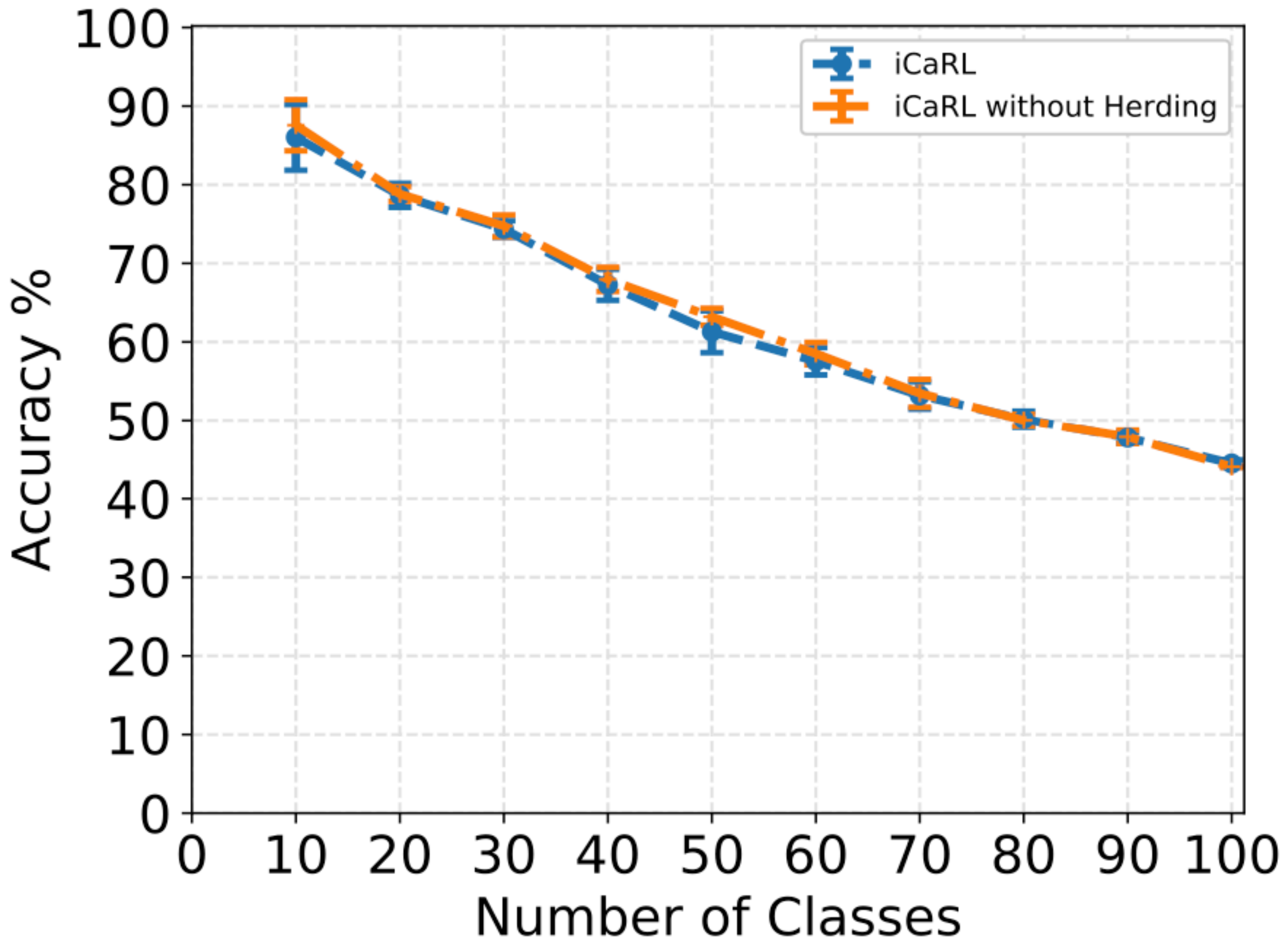
Confusion matrices empirically confirm our intuition



Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

Role of extraction algorithm

Does the herding selection algorithm outperform random selection?



Javed et al, “Revisiting Distillation and Incremental Classifier Learning”, ACCV 2018

Role of memory budget

How expected are our observations?

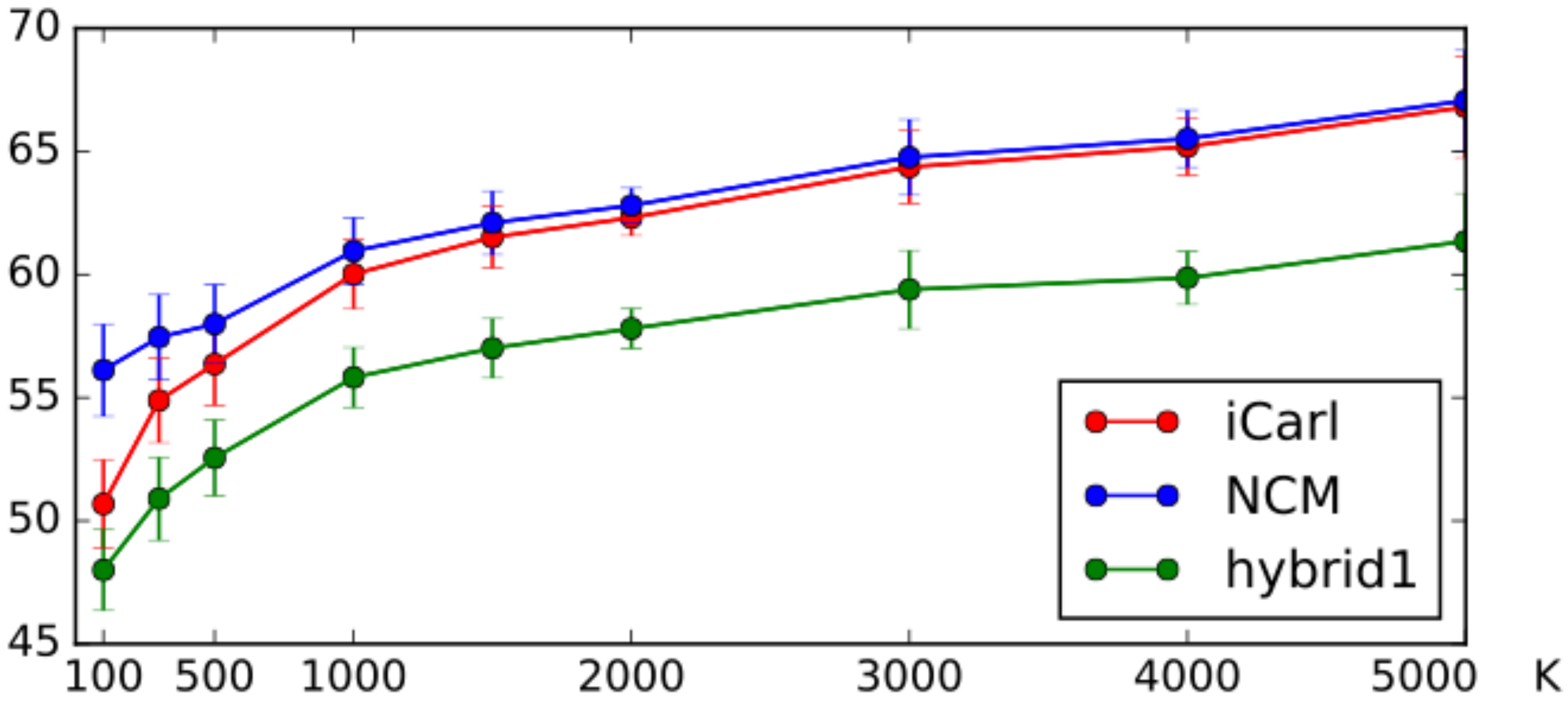


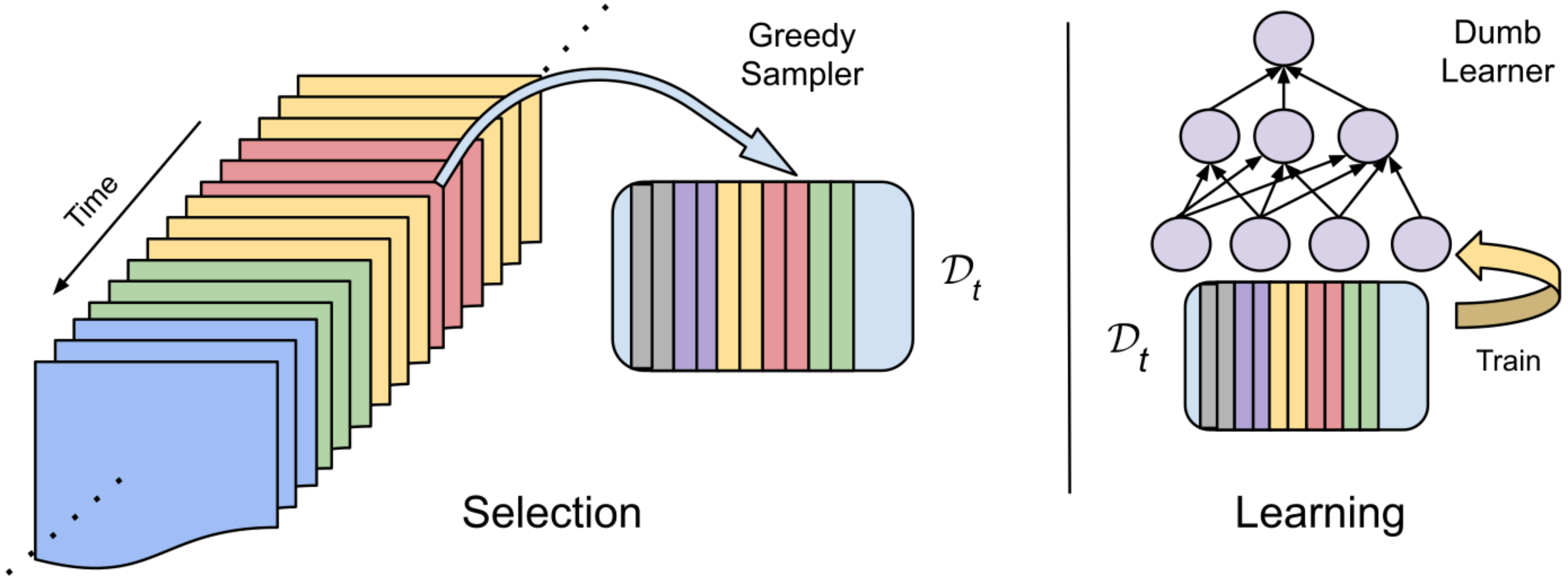
Figure 4: Average incremental accuracy on iCIFAR-100 with 10 classes per batch for different memory budgets K.

Rebuffi et al, “iCaRL: Incremental Classifier and Representation Learning”, CVPR 2017

Role of memory

Is it really just the data subset that we retain?

A “dumb learner” comparison suggests that we may get similar performance if we just train on the exemplar subset



Prabu et al, “GDumb: A Simple Approach that Questions our Approach to Continual Learning”, ECCV 2020

Formally: we may want to find core sets

What is a core set? The term core set is often loosely employed in modern literature to be synonymous to exemplars and sub sets of data

Good introductions are: Bachem et al, “Practical Coreset Constructions for Machine Learning” (2017) or Jubran et al, “Introduction to Coresets: Accurate Coresets” (2019)

Formally: we may want to find core sets

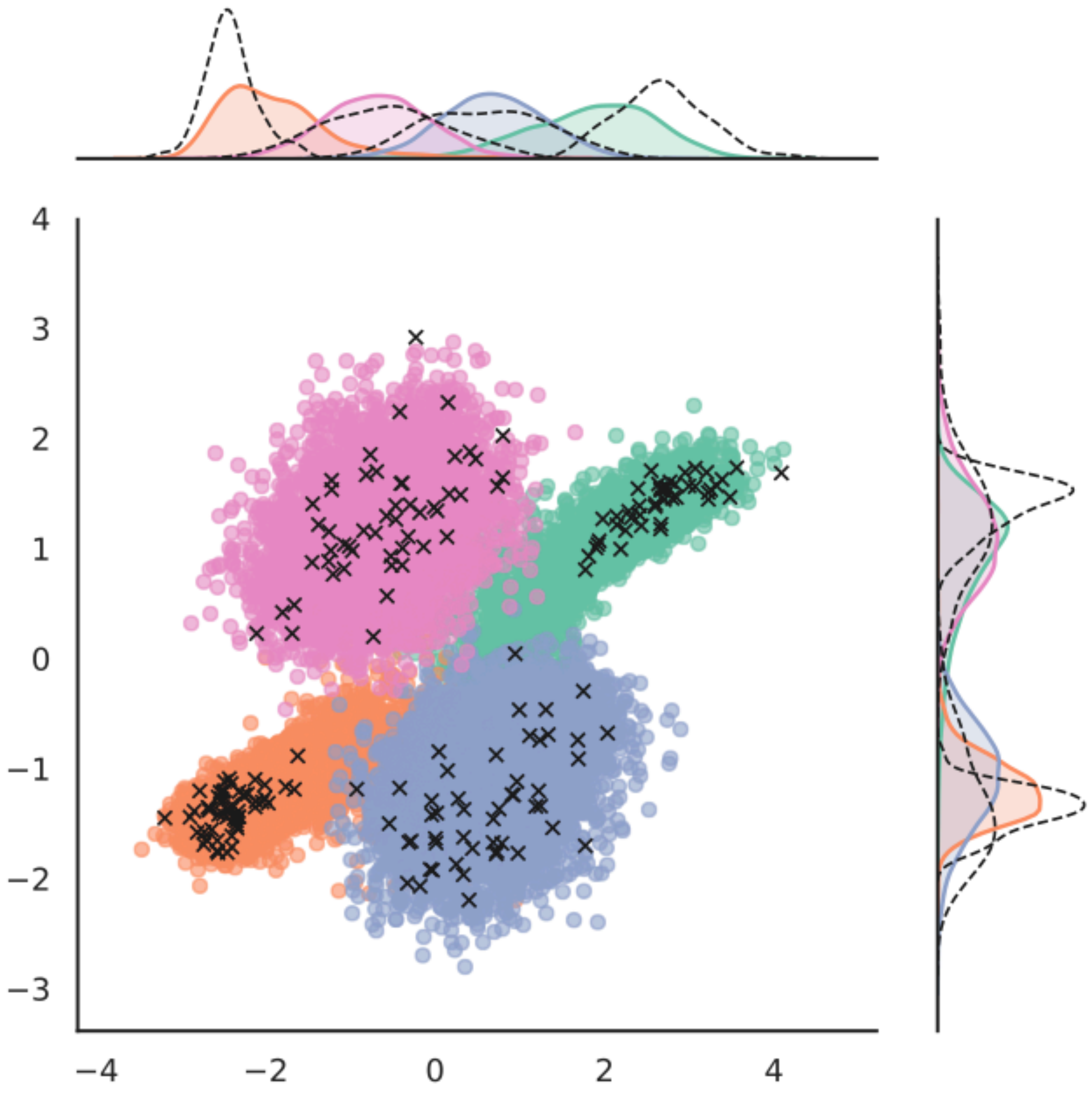
What is a core set? The term core set is often loosely employed in modern literature to be synonymous to exemplars and sub sets of data

"*coresets are small, (weighted) summaries of large data sets such that solutions found on the summary itself are* ***provably competitive*** *with solutions found on the full data set*" $|\texttt{cost}(P, Q) - \texttt{cost}(C, Q)| \leq \varepsilon \cdot \texttt{cost}(P, Q)$

-> specific to data, a set of questions/queries, models + loss/cost functions

Good introductions are: Bachem et al, "Practical Coreset Constructions for Machine Learning" (2017) or Jubran et al, "Introduction to Coresets: Accurate Coresets" (2019)

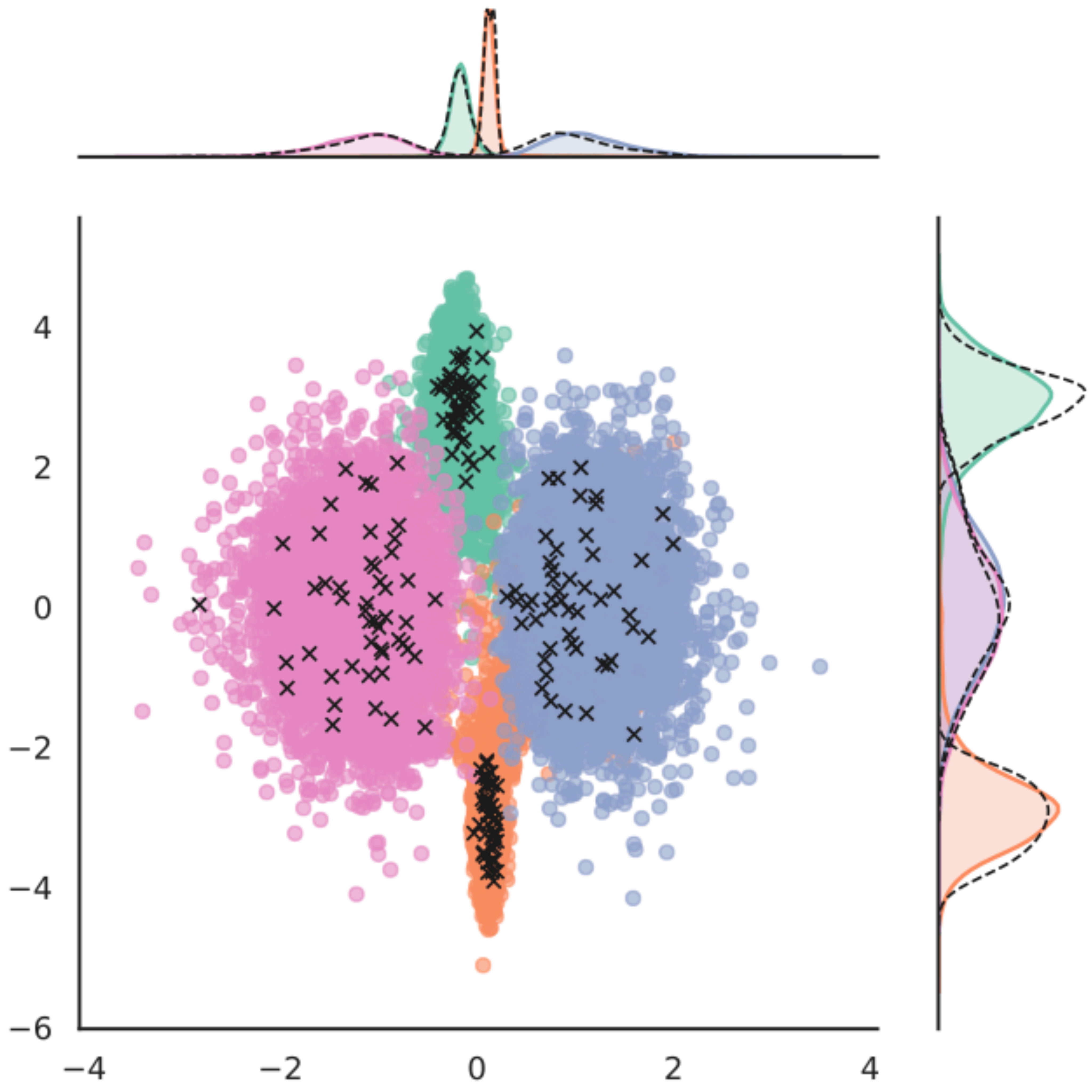
Formally: we may want to find core sets - intuition



- Example of a 2-D latent space with 4 classes/clusters
- Random or k-means (depending on the amount of k) may not mirror the distribution well

Figure from “A Wholistic View of Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning”, Mundt et al, Neural Networks 2023

Formally: we may want to find core sets - intuition



- It's a lot easier if we a notion of the distribution, e.g. we trained a generative model
- (It's not actually that easy in practice for various reasons, but the intuition is that we are somewhat aware of p(x) now)

Figure from "A Wholistic View of Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning", Mundt et al, Neural Networks 2023

Are data memory buffers the solution?
The conjunction of data & model parameters

Role of memory & the brain

“While it is an effective method in ANNs, rehearsal is unlikely to be a realistic model of biological learning mechanisms, as in this context the actual old information (accurate and complete representation of all items ever learned by the organism) is not available.
***Pseudorehearsal is significantly more likely** to be a mechanism which could actually be employed by organisms as it does not require access to this old information, it just requires a way of approximating it.”*

R. French, “Pseudo-recurrent Connectionist Networks: An Approach to the Sensitivity-Plasticity Dilemma”, Connection Science 9:4, 1997

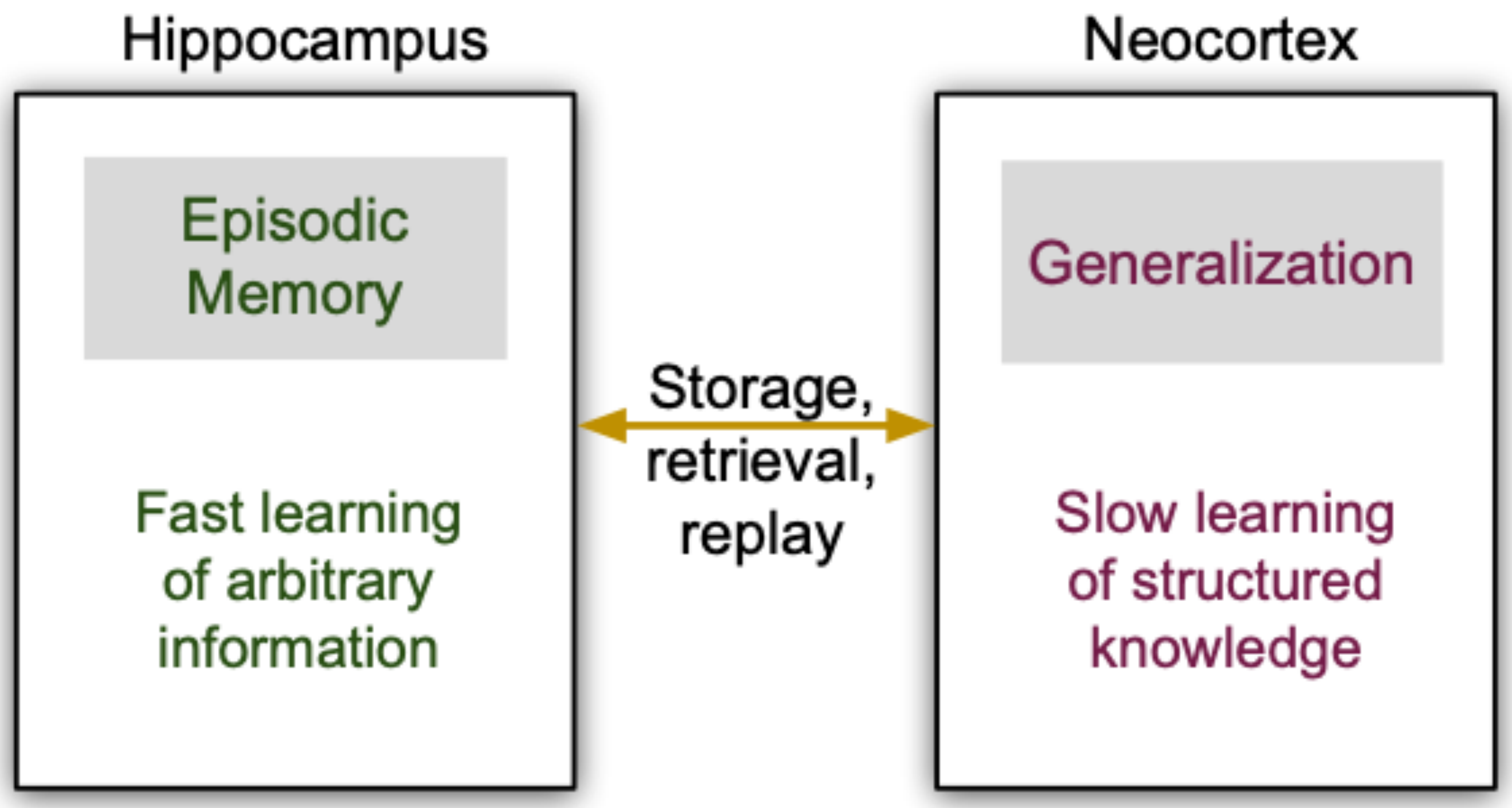
Role of memory & the brain - generative models

*“Pseudorehearsal is based on the use in the rehearsal process of **artificially constructed populations** of “pseudoitems” **instead of the “actual previously learned items**.*

A pseudoitem is constructed by generating a new input vector (setting at random 50% of input elements to 0 and 50% to 1 as usual), and passing it forward through the network in the standard way. Whatever output vector this input generates becomes the associated target output”

A. Robins, “Catastrophic forgetting, rehearsal and pseudorehearsal”, Journal of Neural Computing 7, 1995

Role of memory & the brain - generative models



simplified picture

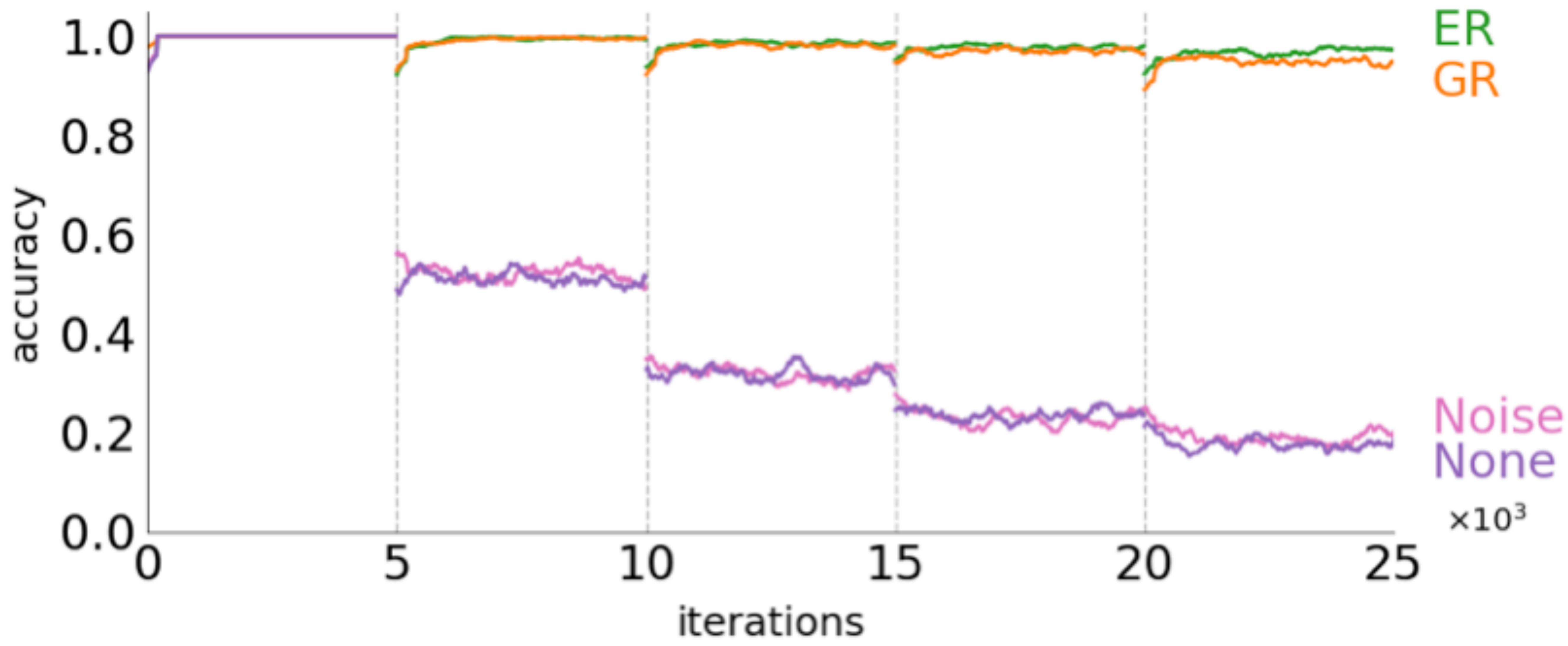
Figure from Parisi et al, "Continual Lifelong Learning with Neural Networks: A Review", Neural Networks 113, 2019

Complementary learning systems

(McClelland et al, Psychological Review 102:3, 1995)

- Hippocampus: short-term adaptation & rapid learning of novel information
- Neocortex: slow learning, to consolidate & build up overlapping representations
- Hippocampus "plays back" over time to neocortex

Exemplar/Generative Rehearsal



- Exemplar Rehearsal (ER) and Generative (Pseudo-)Rehearsal (GR) can work equally well if we have a powerful generator

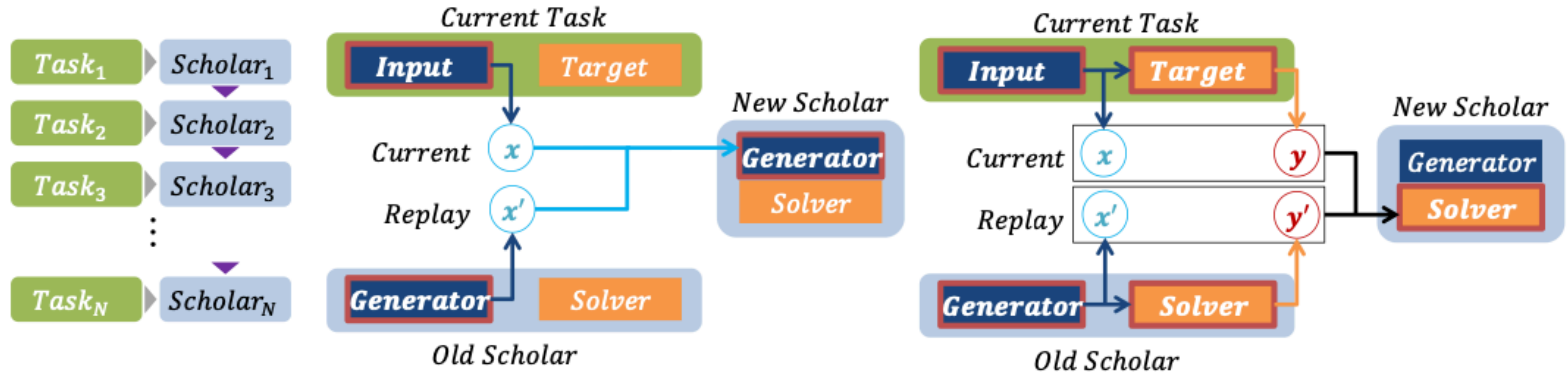
- In contrast, randomly rehearsed sampled noise patterns will no longer work on complex tasks

Shin et al, "Continual Learning with Deep Generative Replay", NeurIPS 2017

Deep Generative Replay

We could train two machine learning models:

a “generator” (there are many types) + “task solver” -> alternate training



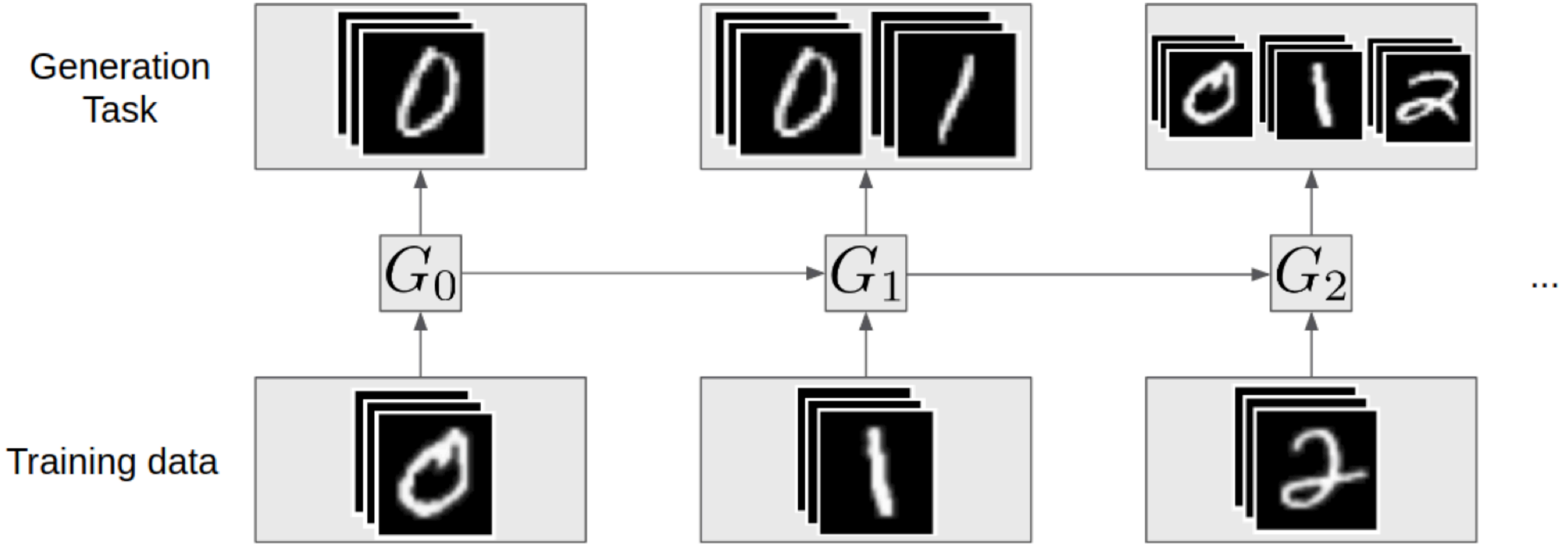
(a) Sequential Training (b) Training Generator (c) Training Solver

Shin et al, “Continual Learning with Deep Generative Replay”, NeurIPS 2017

Deep Generative Replay

We could train two machine learning models:

a “generator” (there are many types) + “task solver” -> alternate training



Lesort et al, “Generative Models from the perspective of Continual Learning”, IJCNN 2019

Let us continue tomorrow with adapting the models we use for both memory of past & encoding of future