Continual Machine Learning

Summer 2024

Teacher

Dr. Martin Mundt,

Research Group on Open World Lifelong Learning

Time

Every Friday 14:25 - 16:05 CEST

Course Homepage

http://owll-lab.com/teaching/cl_lecture_24

https://www.youtube.com/playlist?list=PLm6QXeaB-XkA5-lVBB-h7XeYzFzgSh6sk

Week 6: Dynamic/Modular Neural Architectures

Recall: How to avoid forgetting?

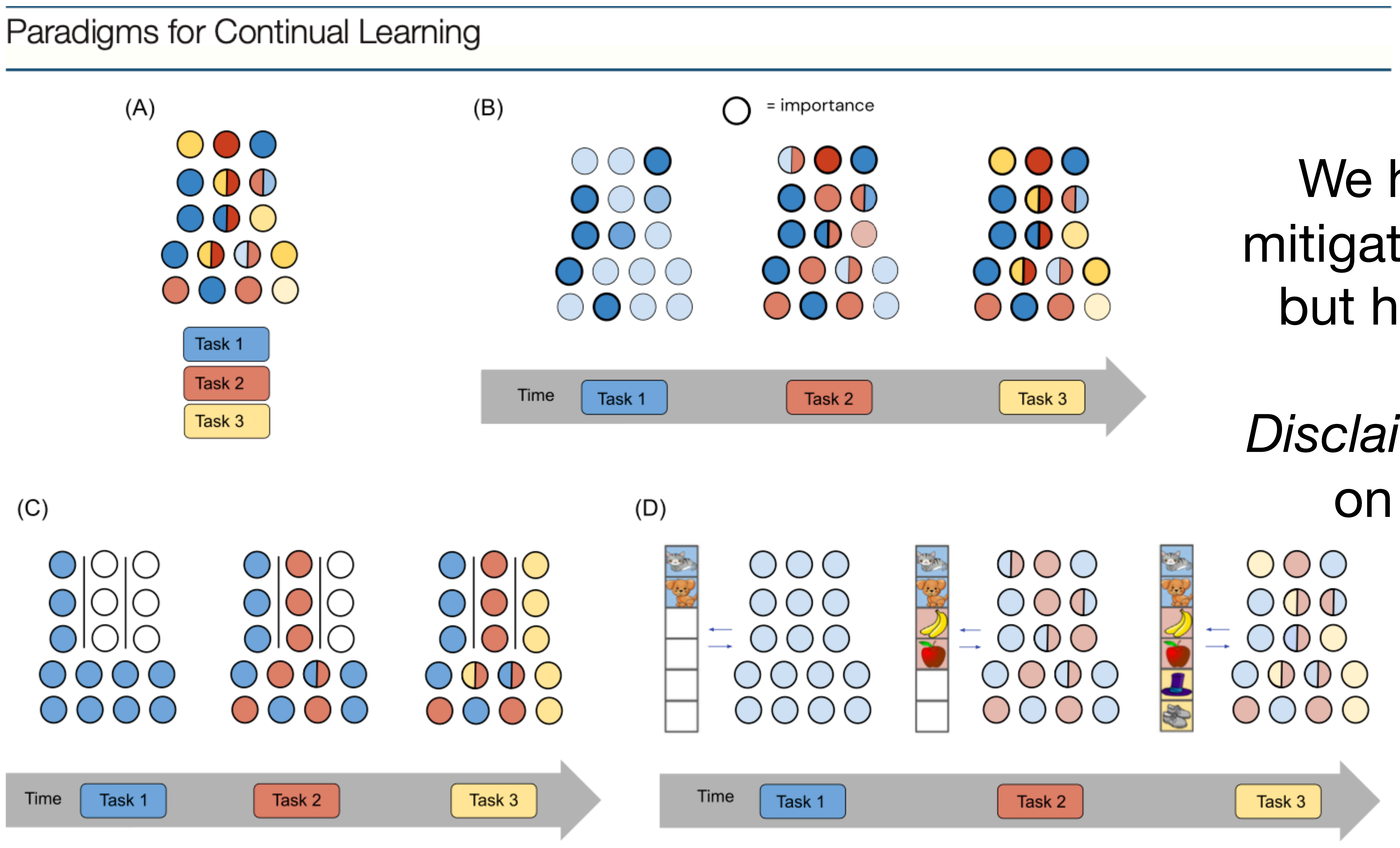


Figure 1. (A) Independent and identically distributed learning methods are standard for nonsequential, multitask learning. In this regime, tasks are learned simultaneously to avoid forgetting and instability. (B) Gradient-based approaches preserve parameters based on their importance to previously learned tasks. (C) Modularity-based methods define hard boundaries to separate task-specific parameters (often accompanied by shared parameters to allow transfer). (D) Memory-based methods write experience to memory to avoid forgetting.

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

We have investigated ways to mitigate **(catastrophic) forgetting** but haven't talked about (C) yet

Disclaimer: we will **focus** primarily on **neural networks** today

Why dynamic architectures?

Why are we talking about dynamic/modular architectures at this point?

"Catastrophic forgetting is a direct consequence of the overlap of distributed representations and can be reduced by reducing this overlap."

Robert French, "Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks", AAAI 1993

Why dynamic architectures?

Why are we talking about dynamic/modular architectures at this point?

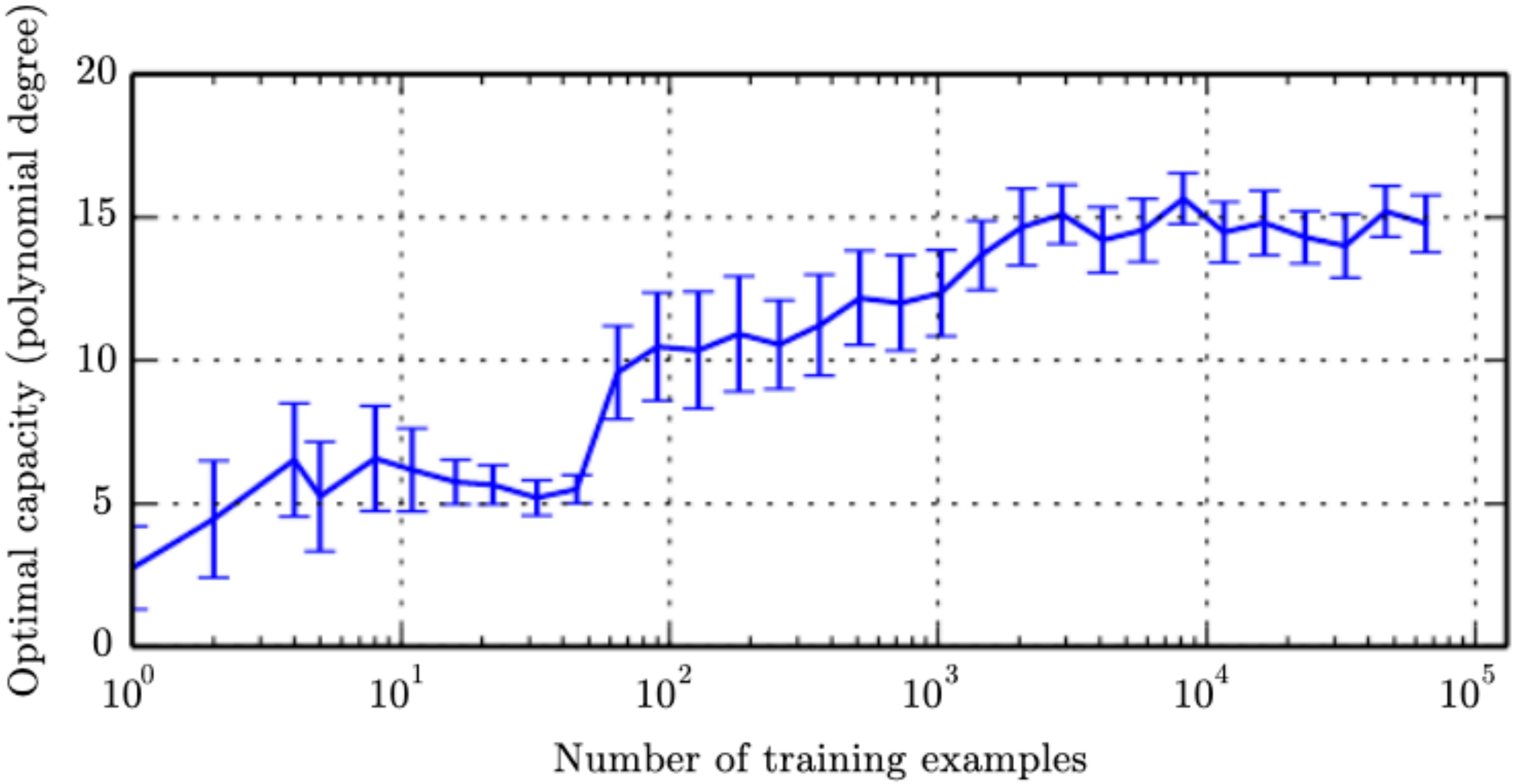
"Catastrophic forgetting is a direct consequence of the overlap of distributed representations and can be reduced by reducing this overlap."

Robert French, "Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks", AAAI 1993

"Very local representations will not exhibit catastrophic forgetting because there is little interaction among representations. However, a look-up table lacks the all-important ability to generalize. The moral of the story is that you can't have it both ways."

Recall lecture 1 on static ML

But it’s not only about catastrophic forgetting: it’s also finding suitable capacity



Deep Learning, Goodfellow, Bengio, Courville, MIT Press 2016,

Machine Learning Basics chapter, page 114.

Two common ways to think about modular architectures

-> **"Implicit"**: over-parametrized and try to create specific sub-modules

"Explicit": add actual parameters/capacity over time

The implicit perspective

The **“implicit" perspective**

- Recall the regularization perspective: identify important parameters, constrain those
- We could assume over-parametrization + try to “sparsify” our parameters
- We create “sub-models” that are primarily responsible for a specific task

The implicit perspective

Example: **activation sharpening** (semi-distributed representations)

- Increase activation of some k nodes, decrease that of others
- Suggestion, overlap as a sum of the smaller activations, the “shared” activation, as a measure of interference
- Four hidden unit example: (0.2, 0.1, 0.9, 0.1) & (0.2, 0.0, 1.0, 0.2)
  Activation overlap: (0.2 + 0.0 + 0.9 + 0.1) / 4 = 0.3
- A non interfering example: (1, 0, 0, 0) & (0, 0, 1, 0) have 0 overlap

Robert French, “Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks”, AAAI 1993

The implicit perspective

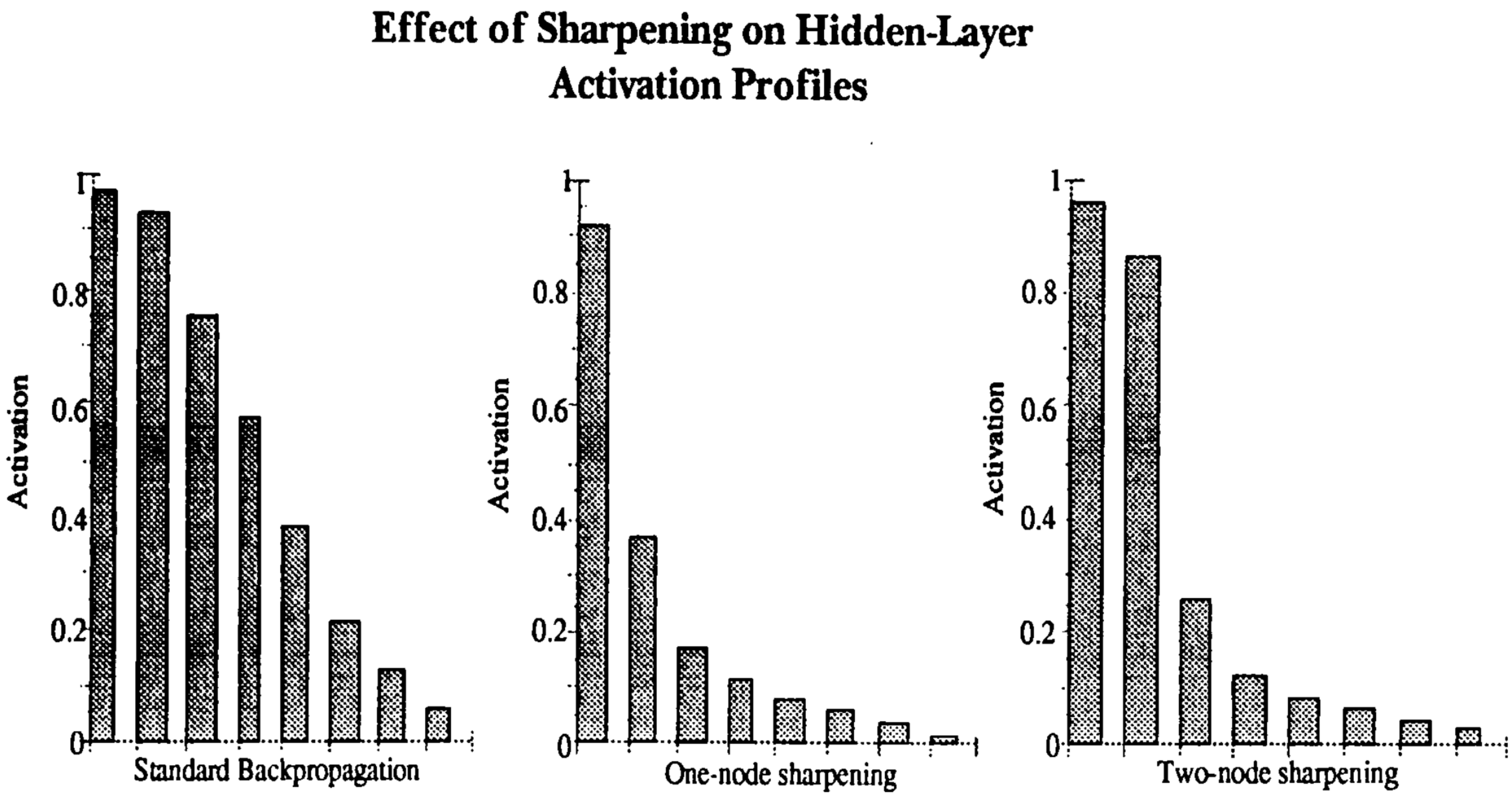
Example: **activation sharpening** (semi-distributed representations)

- Increase activation of some k nodes, decrease that of others
- Suggestion, overlap as a sum of the smaller activations, the “shared” activation

- Perform a forward-activation pass from the input layer to the hidden layer. Record the activations in the hidden layer;
- “Sharpen” the activations of *k* nodes;
- Using the difference between the old activation and the sharpened activation on each node as "error", backpropagate this error to the input layer, modifying the weights between the input layer and the hidden layer appropriately;
- Do a full forward pass from the input layer to the output layer.
- Backpropagate as usual from the output layer to the input layer;
- Repeat.

Robert French, “Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks”, AAAI 1993

The implicit perspective



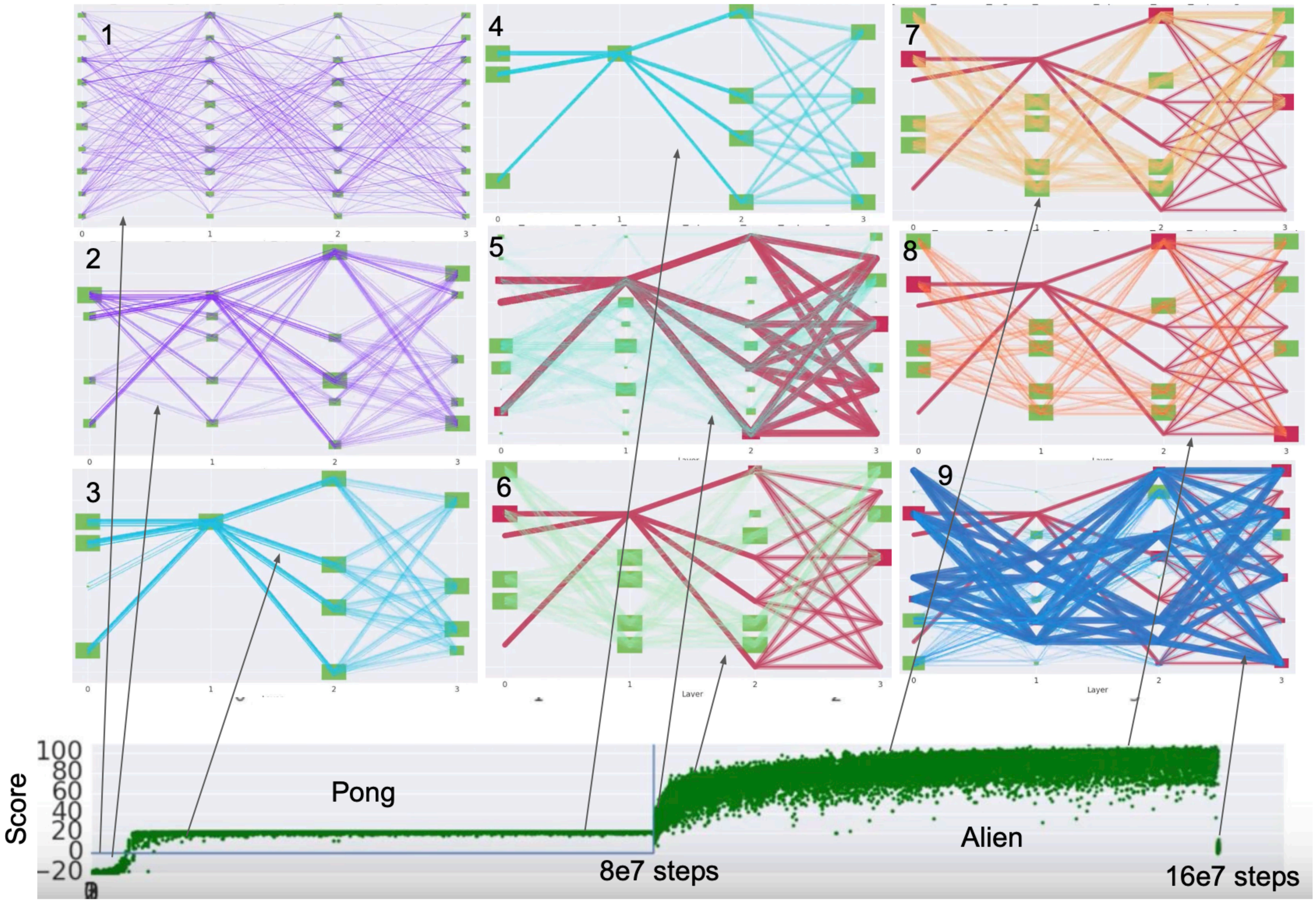
Robert French, “Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks”, AAAI 1993

The implicit perspective

A newer example:

Pathways/PathNets

- Start with an over-parametrized model
- Constrain a task to use a subset of parameters
- Enforce a small/fixed number of active modules/“paths”



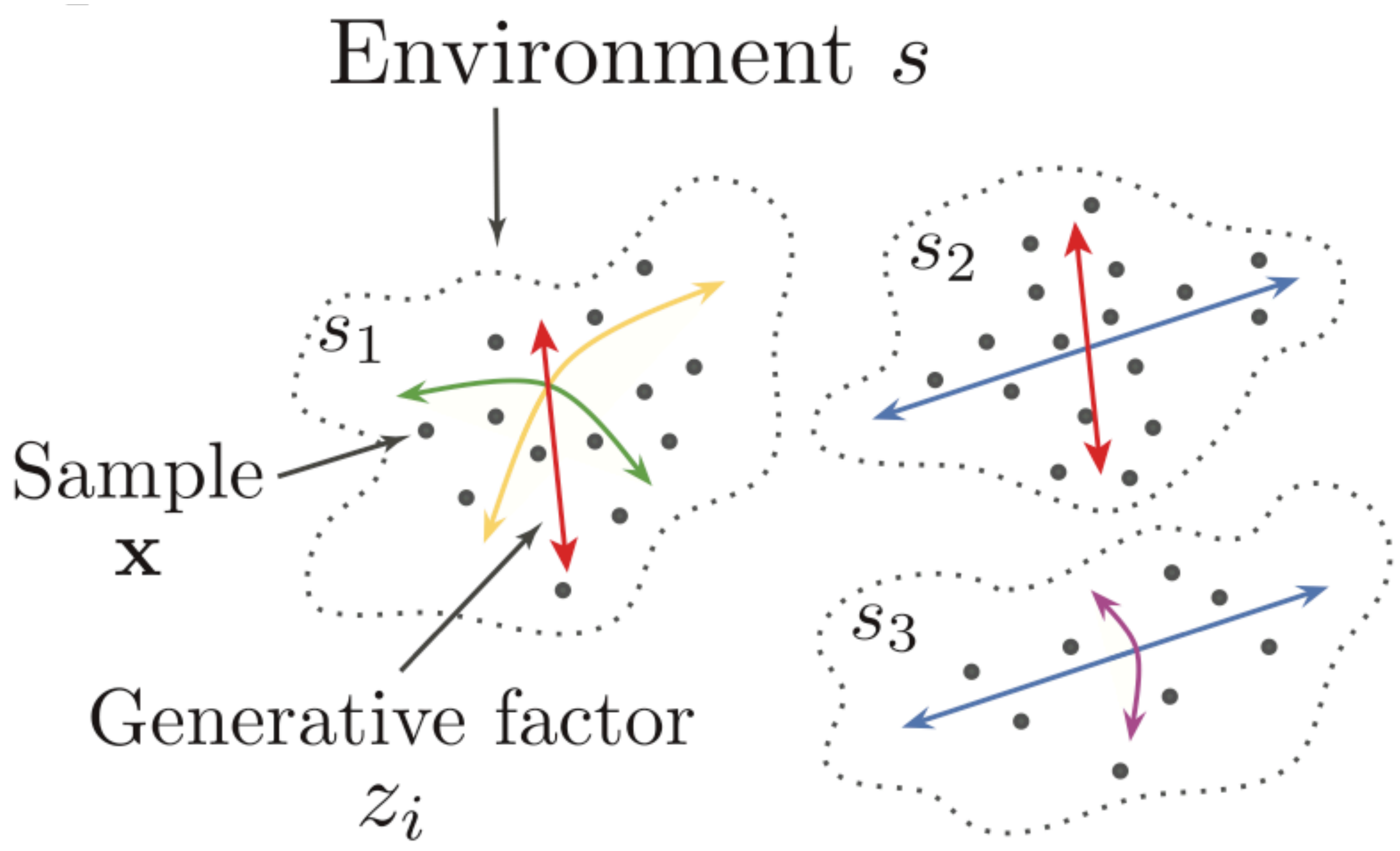
Fernando et al, “PathNet: Evolution Channels Gradient Descent in Super Neural Networks”, arXiv:1701.08734, 2017

The implicit perspective

A different newer example:

Variational Autoencoder with Shared Embeddings (VASE)

- Keep (over-parametrized) encoder/decoder fixed in terms of number of parameters
- Progressively increase latent space capacity in continual learning



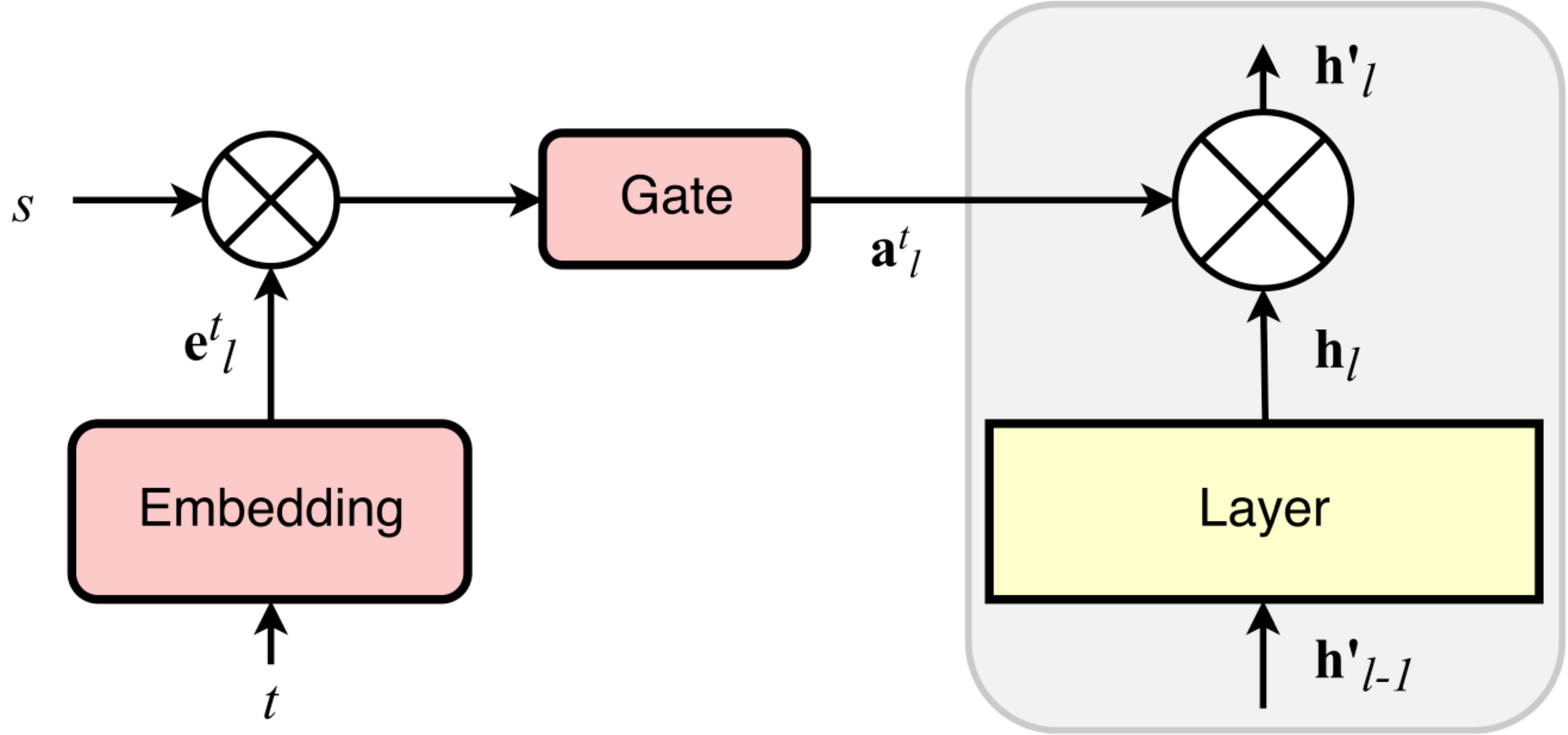
$$\underbrace{\mathbb{E}_{\mathbf{z}^s\sim q_\phi(\cdot|\mathbf{x}^s)}[-\log p_\theta(\mathbf{x}\mid \mathbf{z}^s, s)]}_{\text{Reconstruction error}} + \gamma\Big|\underbrace{\mathbb{KL}(q_\phi(\mathbf{z}^s|\mathbf{x}^s)||p(\mathbf{z}))}_{\text{Representation capacity}} - \underbrace{C}_{\text{Target}}\Big|^2$$

Achille et al, "Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies", NeurIPS 2018

The implicit perspective

There are many ways to go about task specific subsets of parameters/modules:

- Activation overlap
- Parameter sparsity
  (e.g. through L1 regularization)
- "Attention" masks
- … etc.



Serrà et al,"Overcoming Catastrophic Forgetting with Hard Attention to the Task", ICML 2018

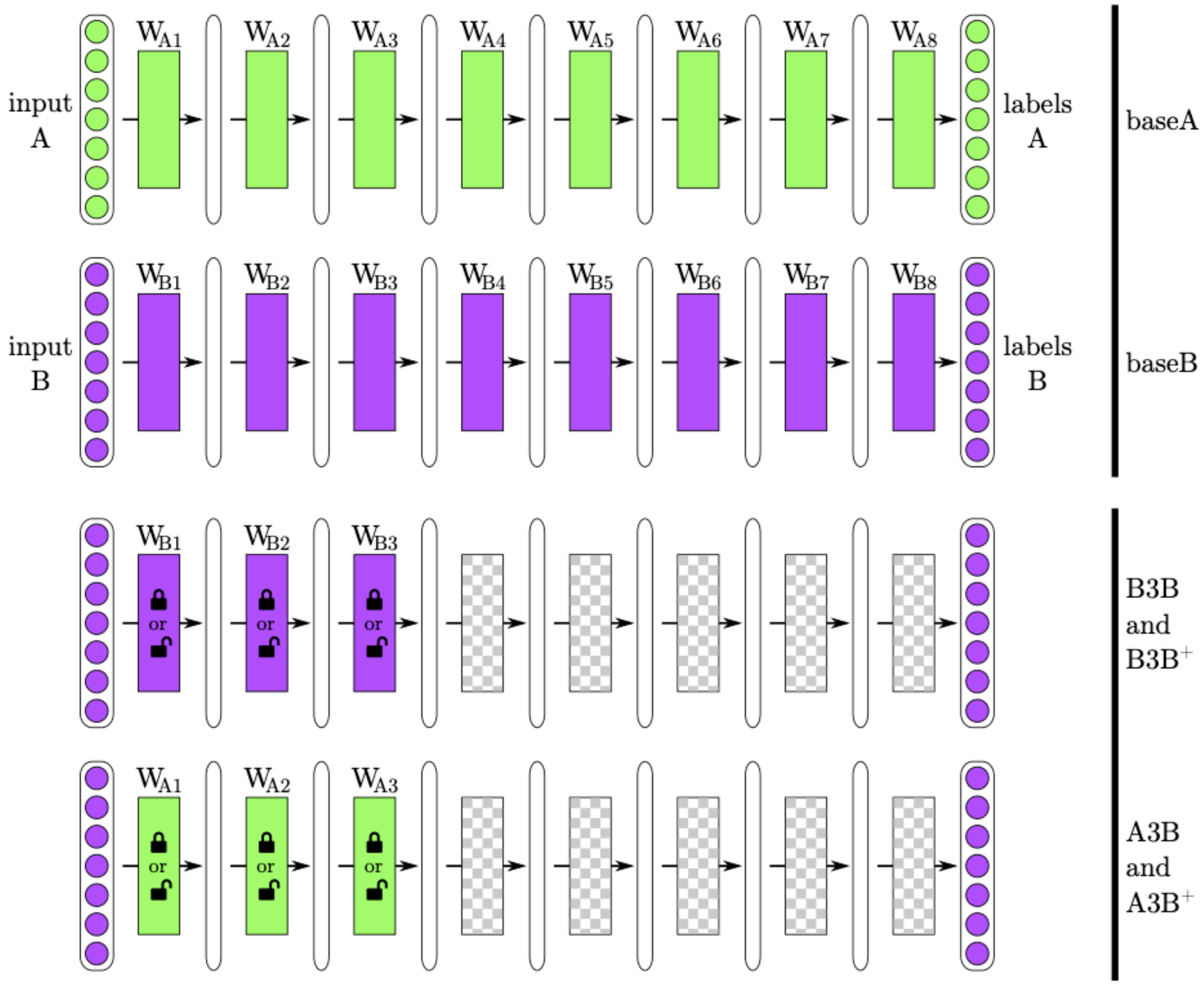
Surely interesting & useful, but what if we don't want to start large/over-parametrized?

Two common ways to think about modular architectures

“Implicit”: over-parametrized and try to create specific sub-modules

-> **“Explicit”**: add actual parameters/capacity over time

Recall lecture 2: transfer



Recall lecture 2 on transfer learning:

some features are more transferable than others

The “**experts**” approach:

- We could share parts + add individual experts on top

“How transferable are features in deep neural networks”, Yosinski et al, NeurIPS 2014

The explicit perspective

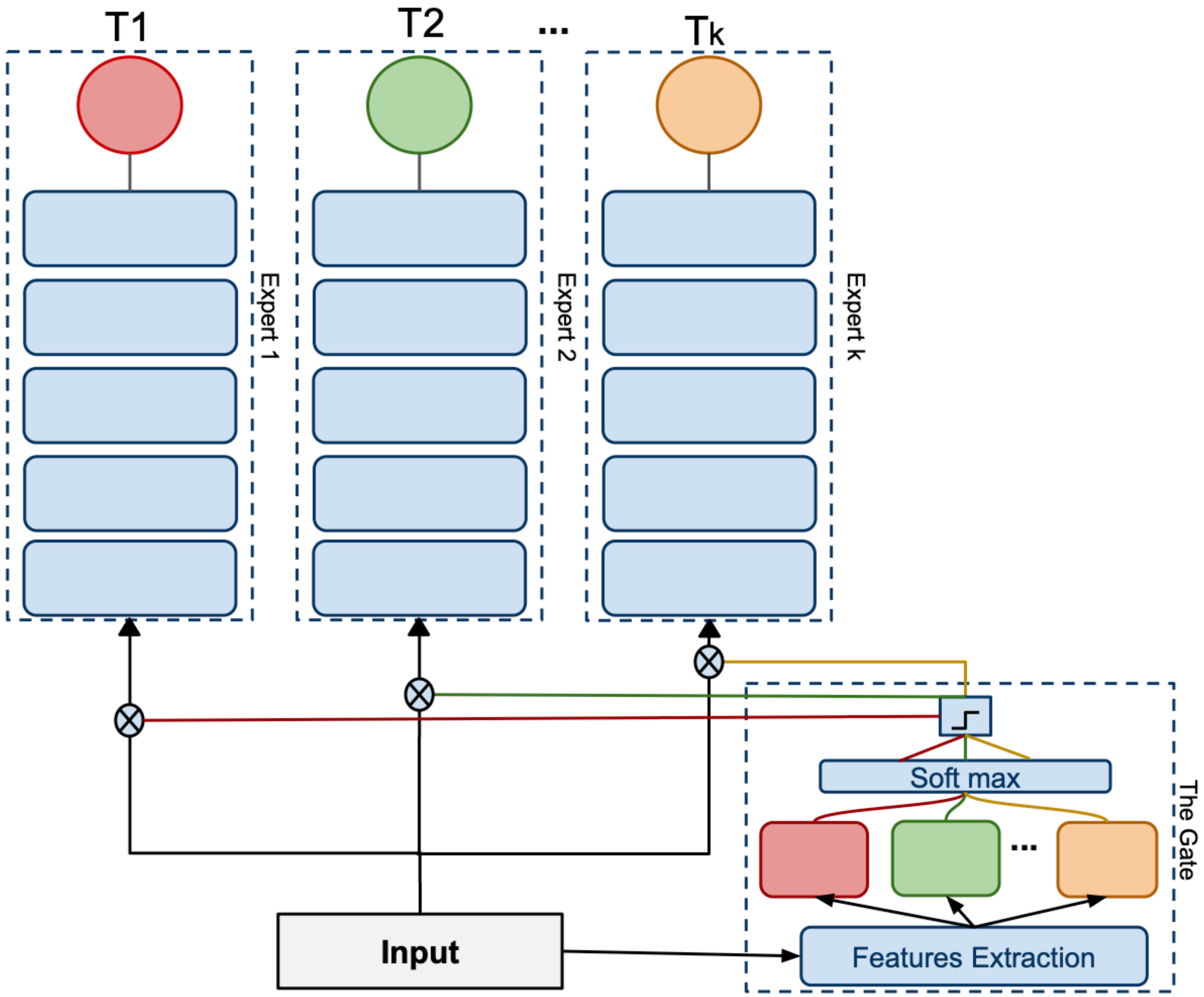


Figure 1. The architecture of our Expert Gate system.

Aljundi et al, “Expert Gate: Lifelong Learning with a Network of Experts”, CVPR 2017

The “**experts**” approach:

- We could share parts + add individual experts on top

+ A solid & somewhat “safe” approach

+- “Backbone” is static & experts don’t share all knowledge (is this a + or -?)

- Can be tough to determine which expert to use

The explicit perspective

The **explicit perspective:** plasticity from a different angle - inspiration from **neurogenesis?**

"After two decades of research, the neurosciences have come a long way from accepting that neural stem/progenitor cells generate new neurons in the adult mammalian hippocampus to unraveling the functional role of adult-born neurons in cognition and emotional control. ***The finding that new neurons are born and become integrated into a mature circuitry throughout life has challenged and subsequently reshaped our understanding of neural plasticity in the adult mammalian brain.****"*

(Quote: Vadodaria & Jessberger, "Functional neurogenesis in the adult hippocampus: then and now", frontiers in neuroscience 8, 2014, see also C. Gross, "Neurogenesis in the adult brain: death of a dogma", Nature Reviews Neuroscience, 2000)

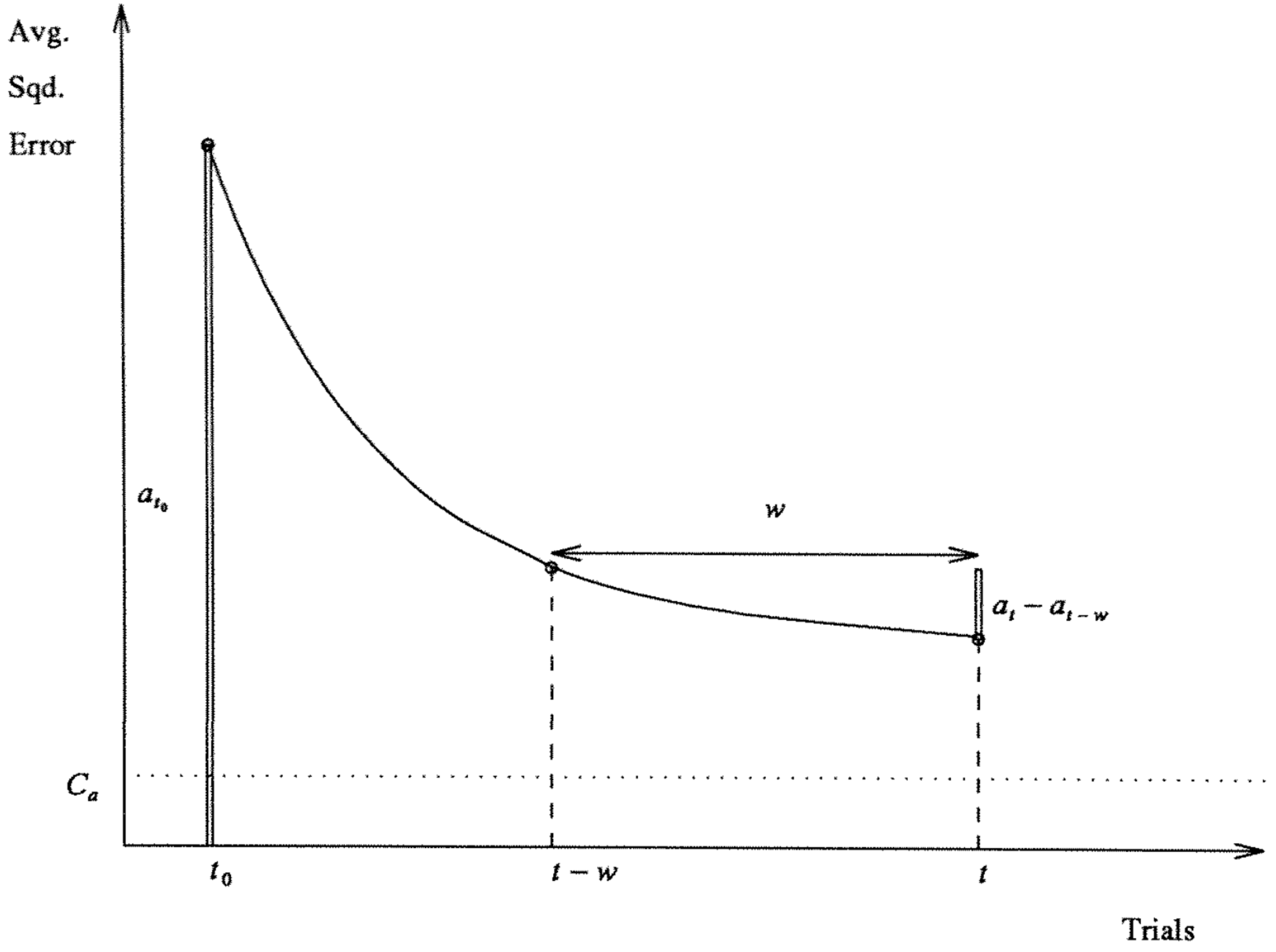
The explicit perspective

Example: **Dynamic Node Creation**

- Small initial amount of parameters

First crucial question: **When should we add?**

- Assumes decaying exponential for error
- Add node when error plateaus

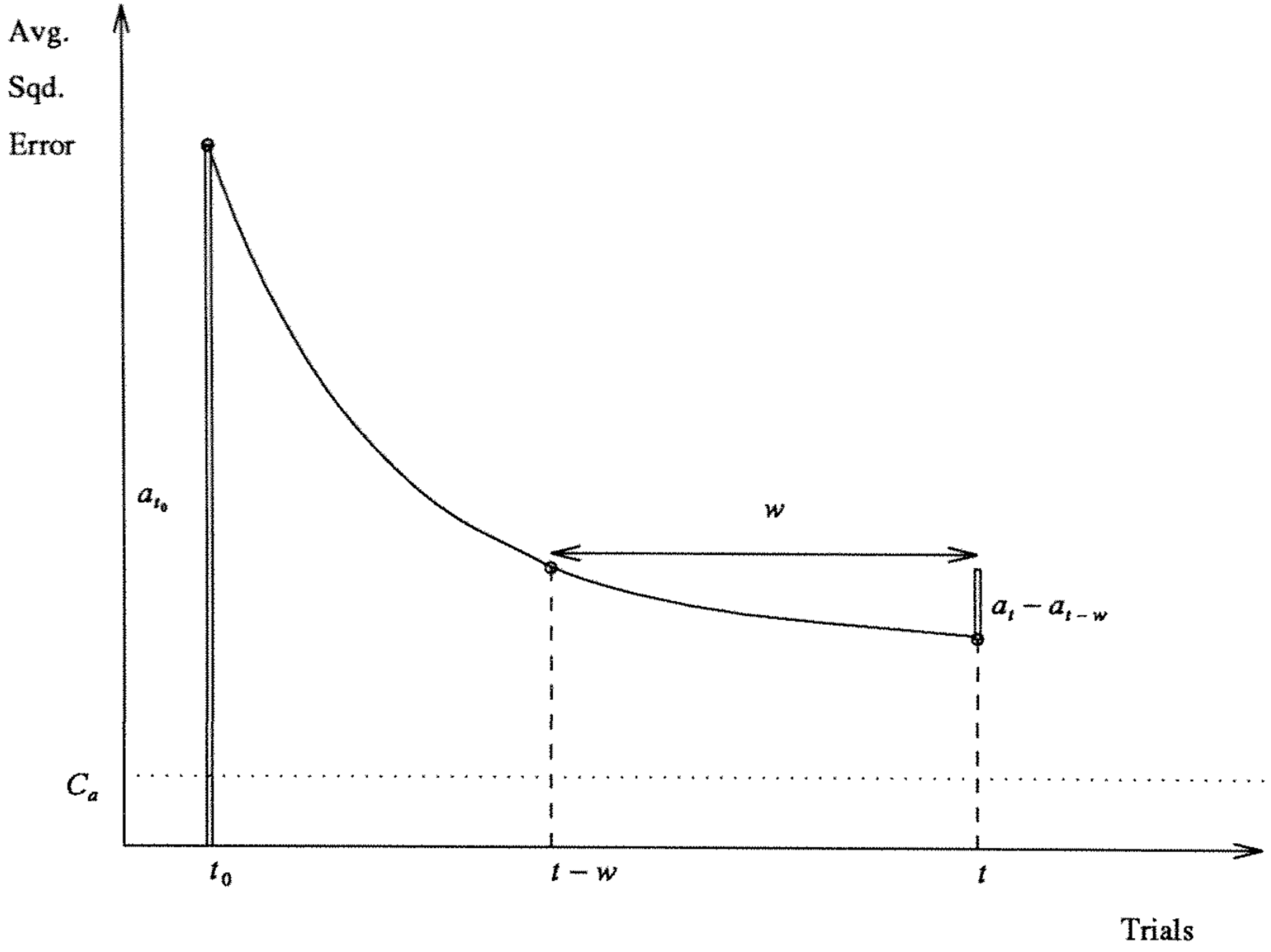


T. Ash, “Dynamic Node Creation in Backpropagation Networks”, Connection Science 1:4, 1989

The explicit perspective

Second crucial question: **when do we stop?**

- Calculate the ratio over the drop in average (squared) error (*a*) across some window (*w*) of time (*t*)
- Stop when relative improvement becomes too small: $\frac{a_t - a_{t-w}}{a_{t_0}} < \Delta_T$
- Stop when acceptable performance/cutoff (C) is reached: $a_t \leq C_a$



T. Ash, "Dynamic Node Creation in Backpropagation Networks", Connection Science 1:4, 1989

The explicit perspective

Has been empirically investigated on some “simpler” test problems

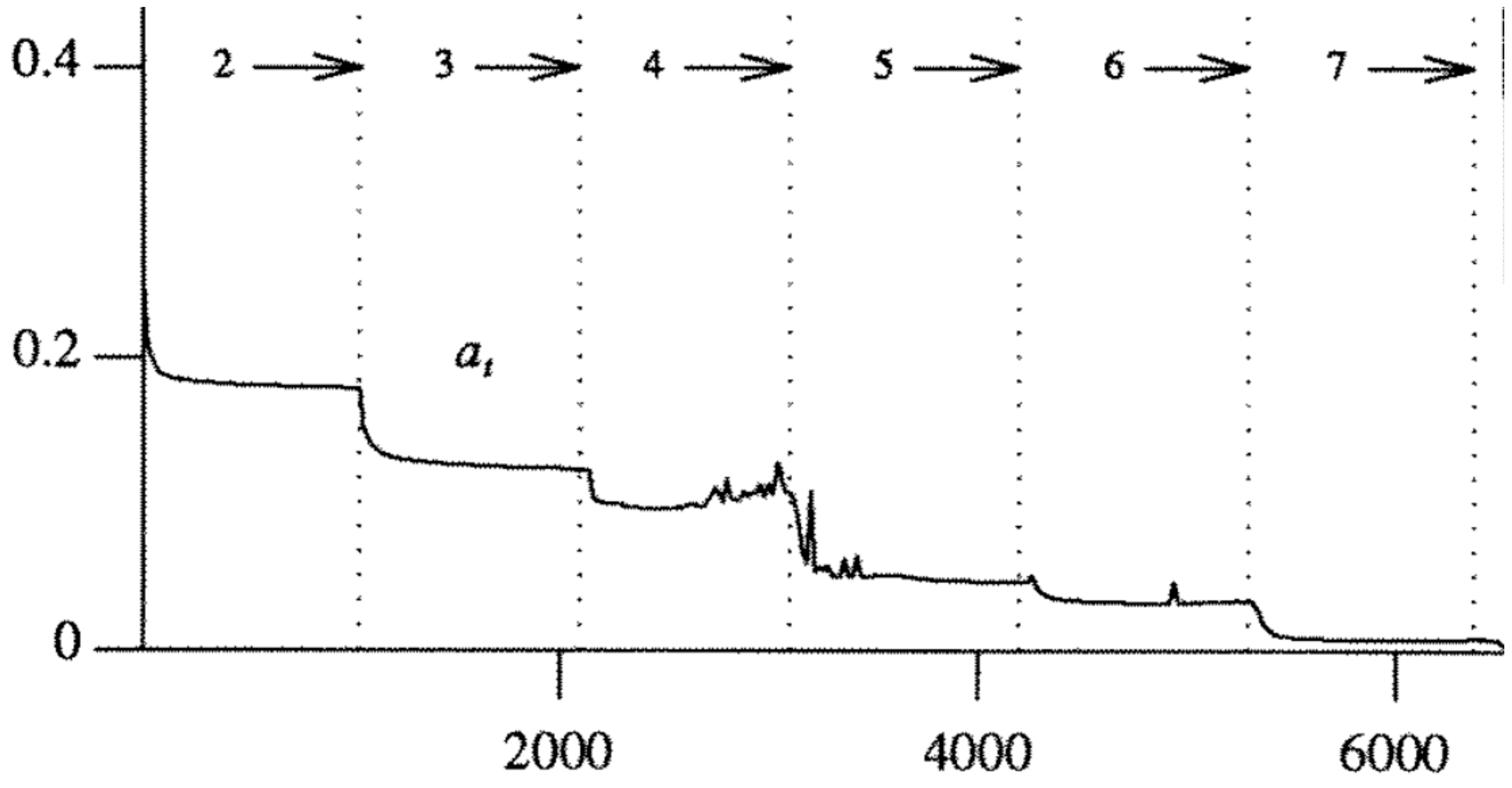
TABLE 2. TEST PROBLEMS ALONG WITH EMPIRICAL UPPER BOUNDS ON THE NUMBER OF HIDDEN LAYER UNITS

Name	Input	Output	Known Solution (# of hidden units)
Encoder Problem (ENC)	N bit binary vector with 1 bit on	Same as input	$\log_2 N$
Symmetry (SYM)	N bit binary vector	1 if symmetric, 0 if asymmetric	2
Parity (PAR)	N bit binary vector	1 if # of 1's is odd, 0 otherwise	N
Binary Addition (ADD)	Two N bit binary vectors	N bit result and 1 carry bit	None known for one hidden layer

T. Ash, “Dynamic Node Creation in Backpropagation Networks”, Connection Science 1:4, 1989

The explicit perspective

Squared error (y axis) for the ADD3 test problem

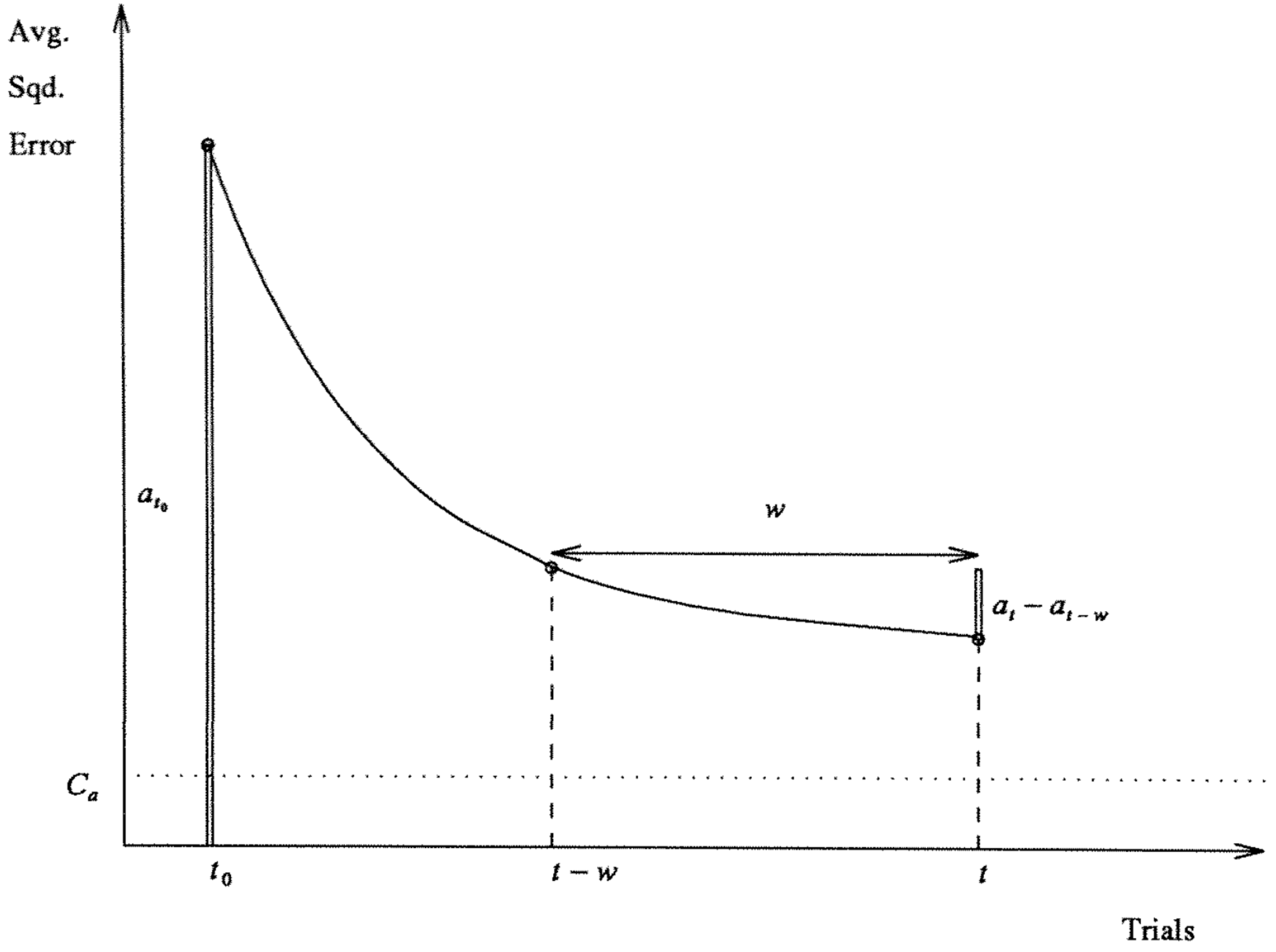


T. Ash, “Dynamic Node Creation in Backpropagation Networks”, Connection Science 1:4, 1989

The explicit perspective

Technically, **third crucial question** (not taken into account here): **how/what do we add?**

- Do we add one parameter or many?
- A neural network layer?
- Do we add a whole new function?
- A different output head if our tasks change?



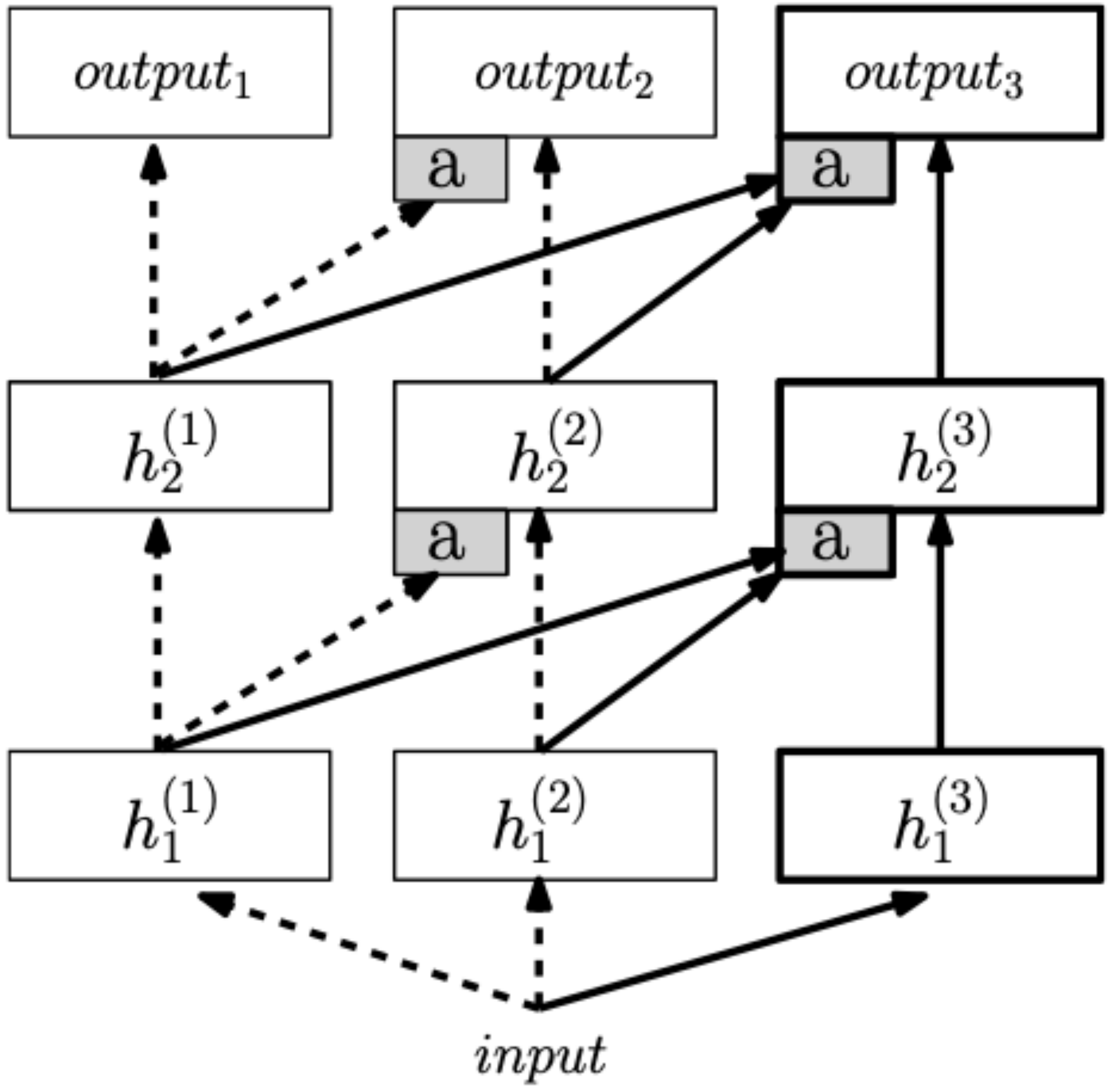
T. Ash, “Dynamic Node Creation in Backpropagation Networks”, Connection Science 1:4, 1989

The explicit perspective

A newer example: progressive networks

- Start with a single "column" of parameters
- Add "column" for new task + freeze old column
- New columns receive lateral connections from old ones

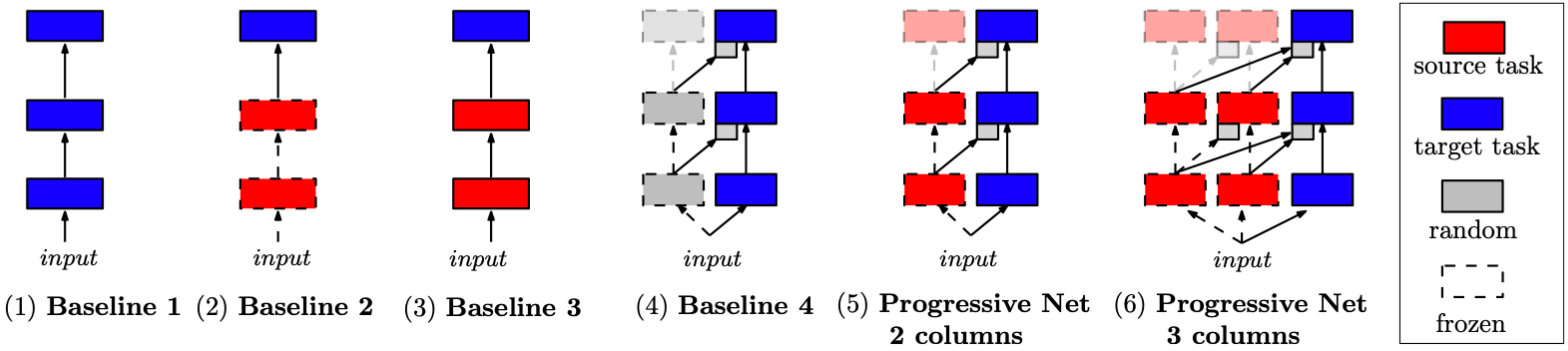
Avoid forgetting & allow transfer where possible



Rusu et al, "Progressive Neural Networks", arXiv:1606.04671, 2016

The explicit perspective

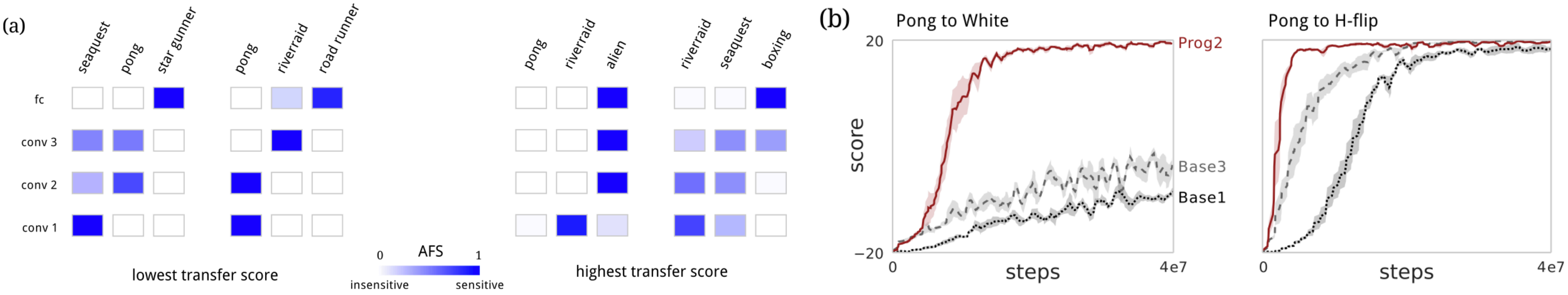
We can evaluate and analyze similarly to what we have already seen in lecture 2, when we talked about knowledge transfer



Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

The explicit perspective

We can evaluate and analyze similarly to what we have already seen in lecture 2, when we talked about knowledge transfer



Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

Aren’t some of these solutions “obvious”?

Aren’t some of these solutions “obvious”?

“While many of the individual ingredients used in progressive nets can be found in the literature, their combination and use in solving complex sequences of tasks is novel”

(Rusu et al, Progressive Neural Networks, 2017)

Aren’t some of these solutions “obvious”?

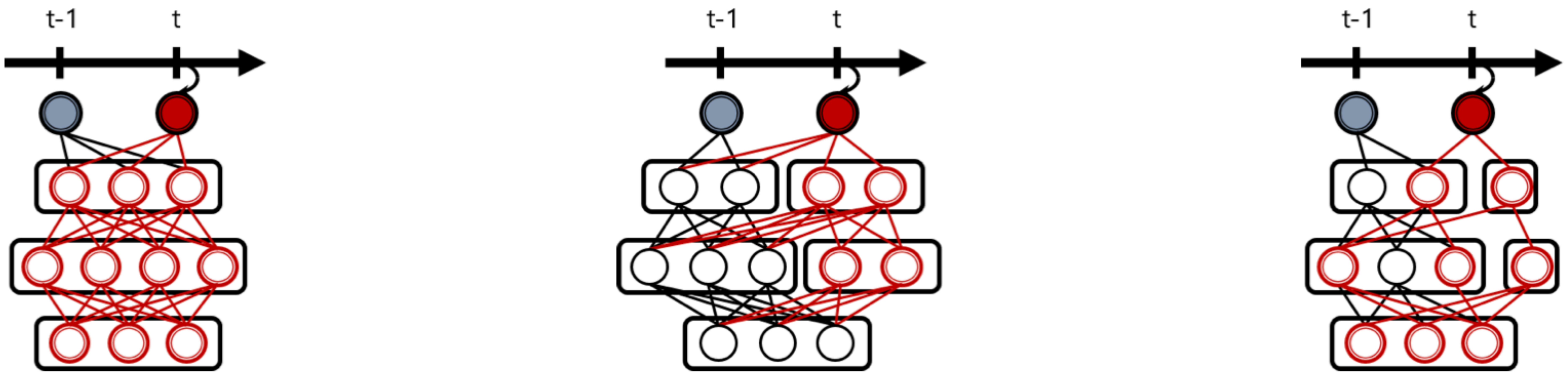
Recall questions: what to start with, when to add/remove - what, how, how much; when to stop …?

!! Developing concrete algorithms & applications is challenging !!

Dynamically Expandable Nets

Various combinations with partial re-training with expansion

EWC | Progressive Nets | DEN



(a) Retraining w/o expansion (b) No-retraining w/ expansion (c) Partial retraining w/ expansion

Yoon et al, “Lifelong Learning with Dynamically Expandable Networks”, ICLR 2018

Dynamically Expandable Nets

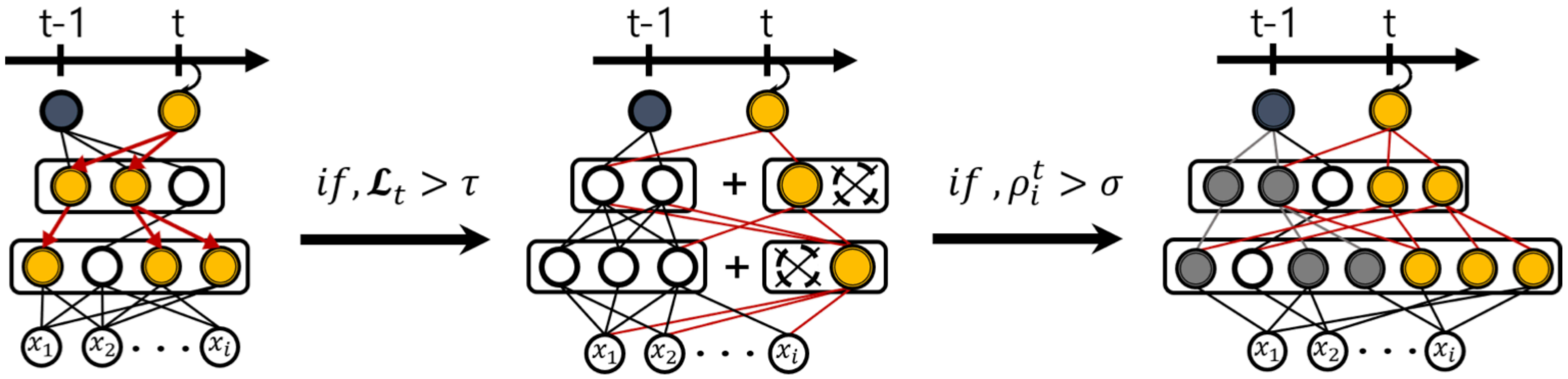
Three key steps:

1. Selective retraining
2. Dynamic expansion
3. Split & duplicate units

Perhaps sidelines the question of how much to add by removing again

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \dots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

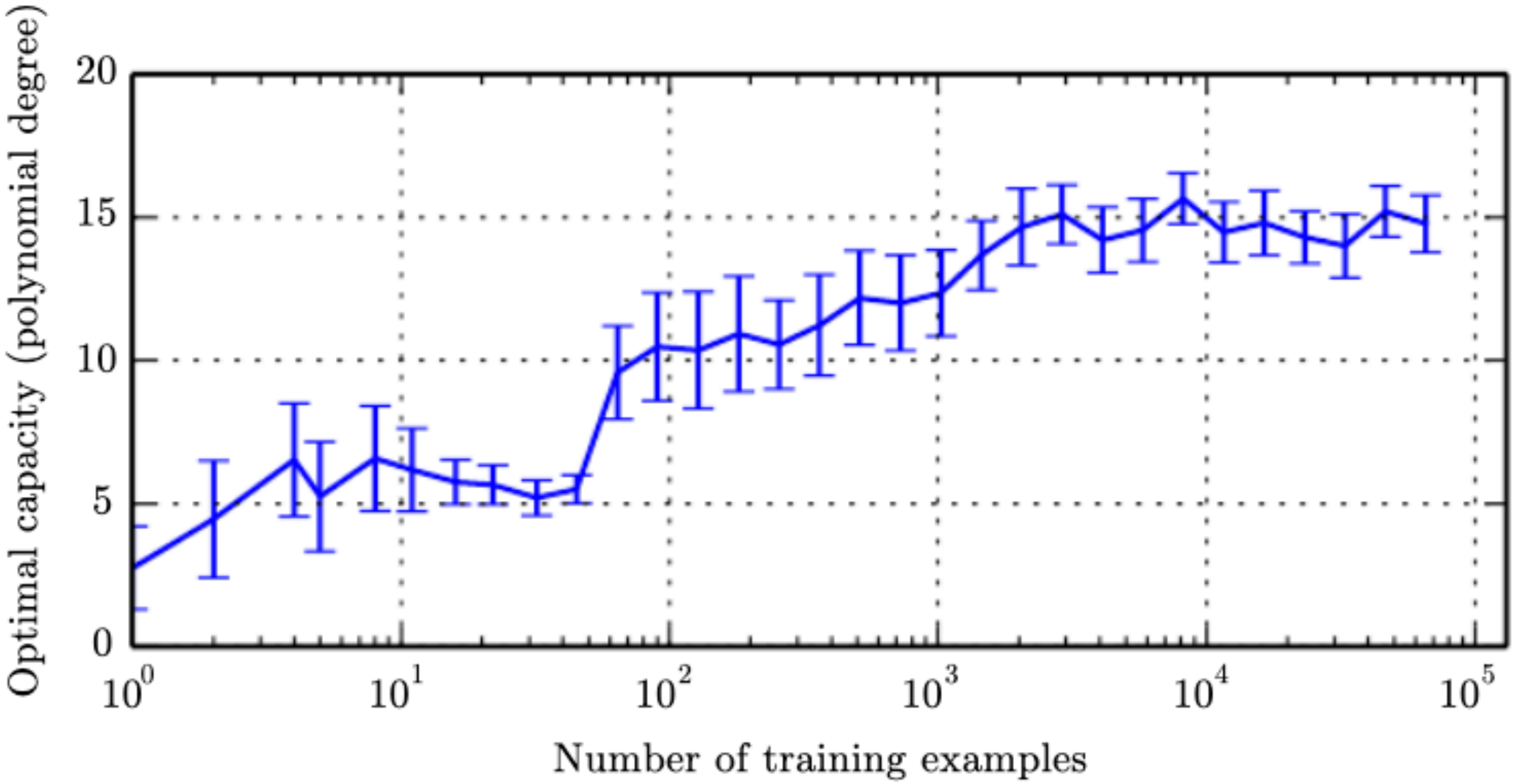


Yoon et al, “Lifelong Learning with Dynamically Expandable Networks”, ICLR 2018

Why is the efficacy of these approaches hard to interpret?
Beyond measuring (catastrophic) forgetting

Recall lecture 1 on static ML

But it's not only about catastrophic forgetting: it's also finding suitable capacity



Deep Learning, Goodfellow, Bengio, Courville, MIT Press 2016,

Machine Learning Basics chapter, page 114.

Small -> large sample scenarios

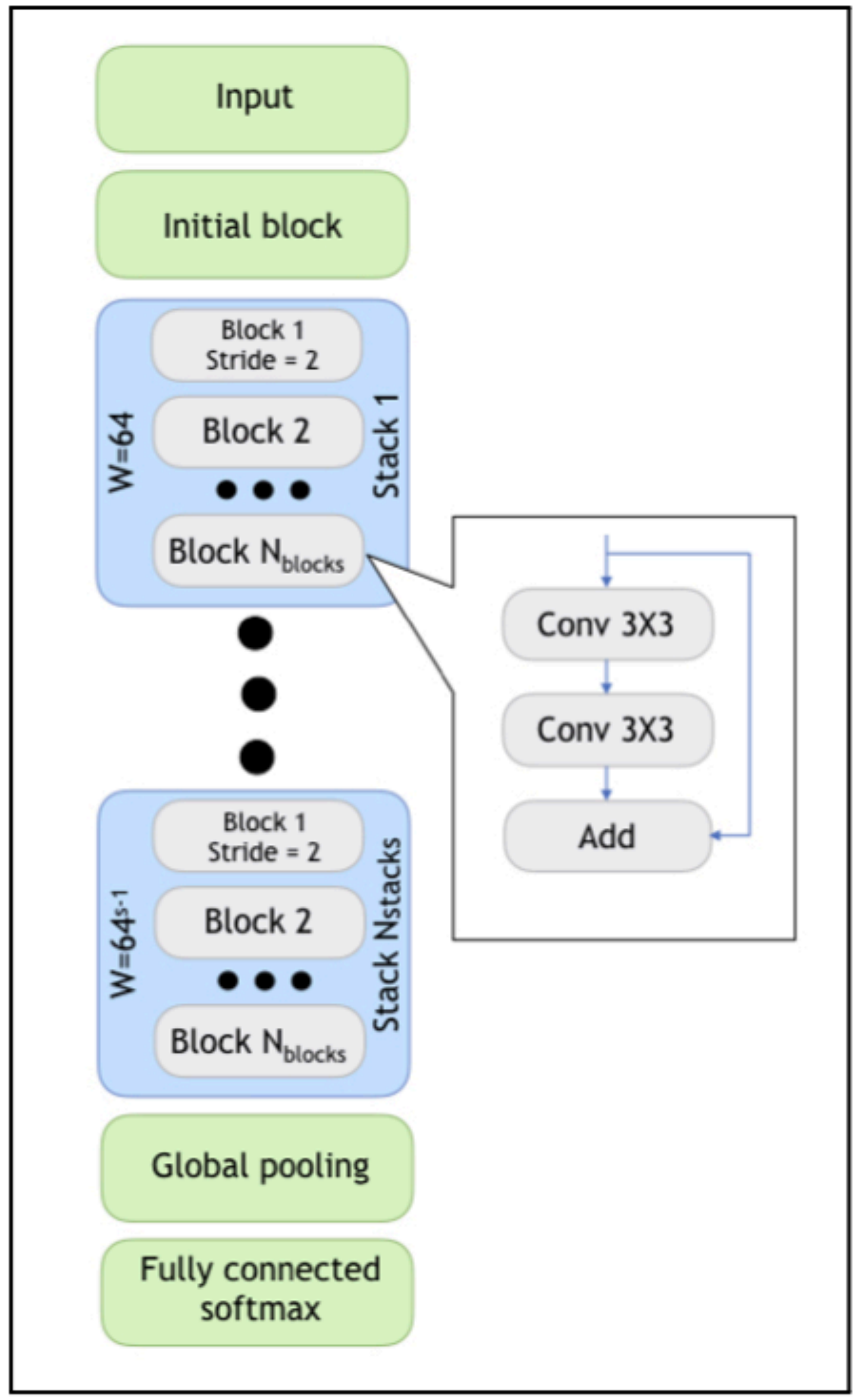
The **active learning perspective**

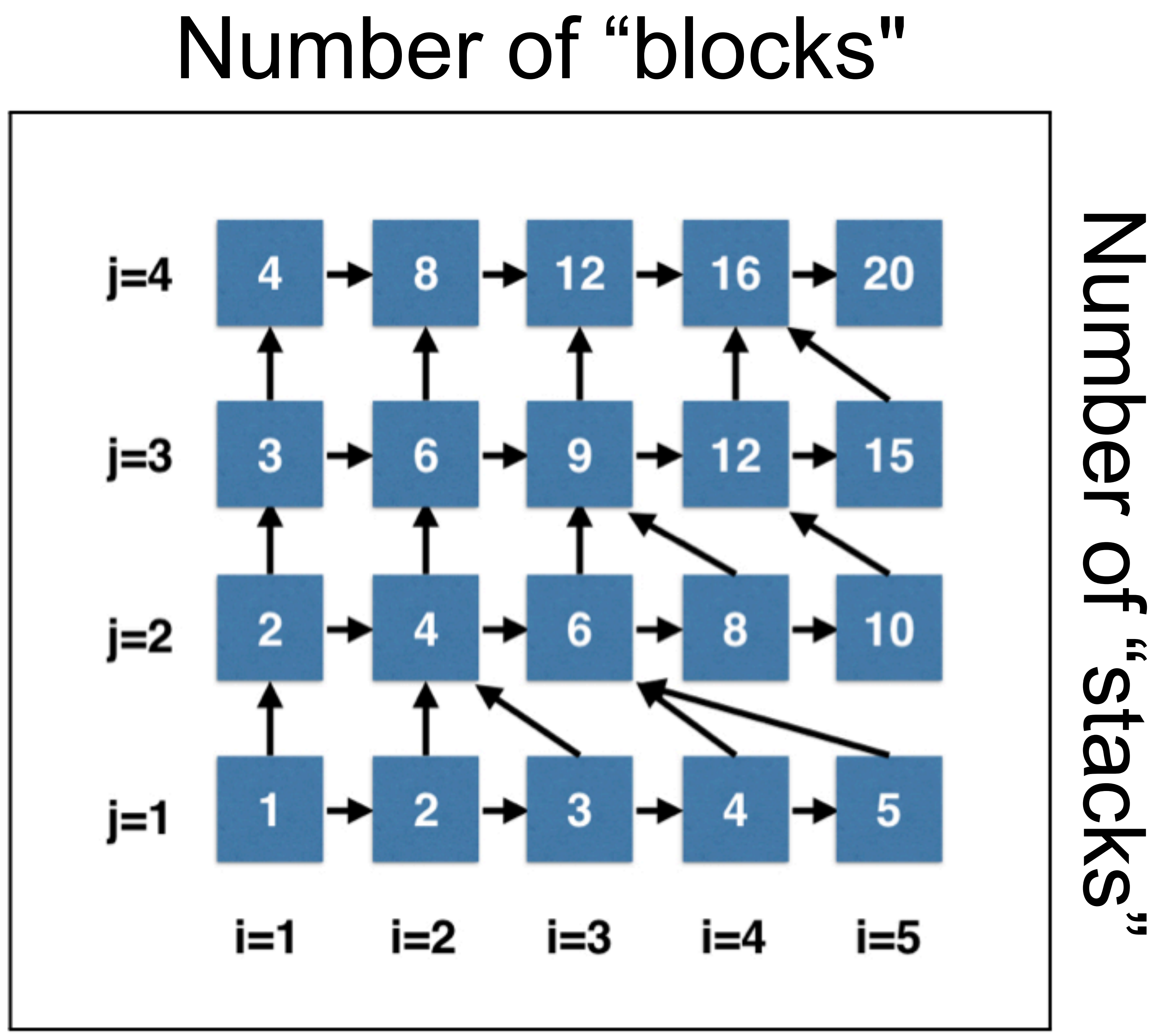
Incremental architecture approach:

For every query, evaluate three architecture choices

1. The present architecture
2. One with expanded width
3. One that also adds layers

Greedily select the best candidate in terms of a validation dataset

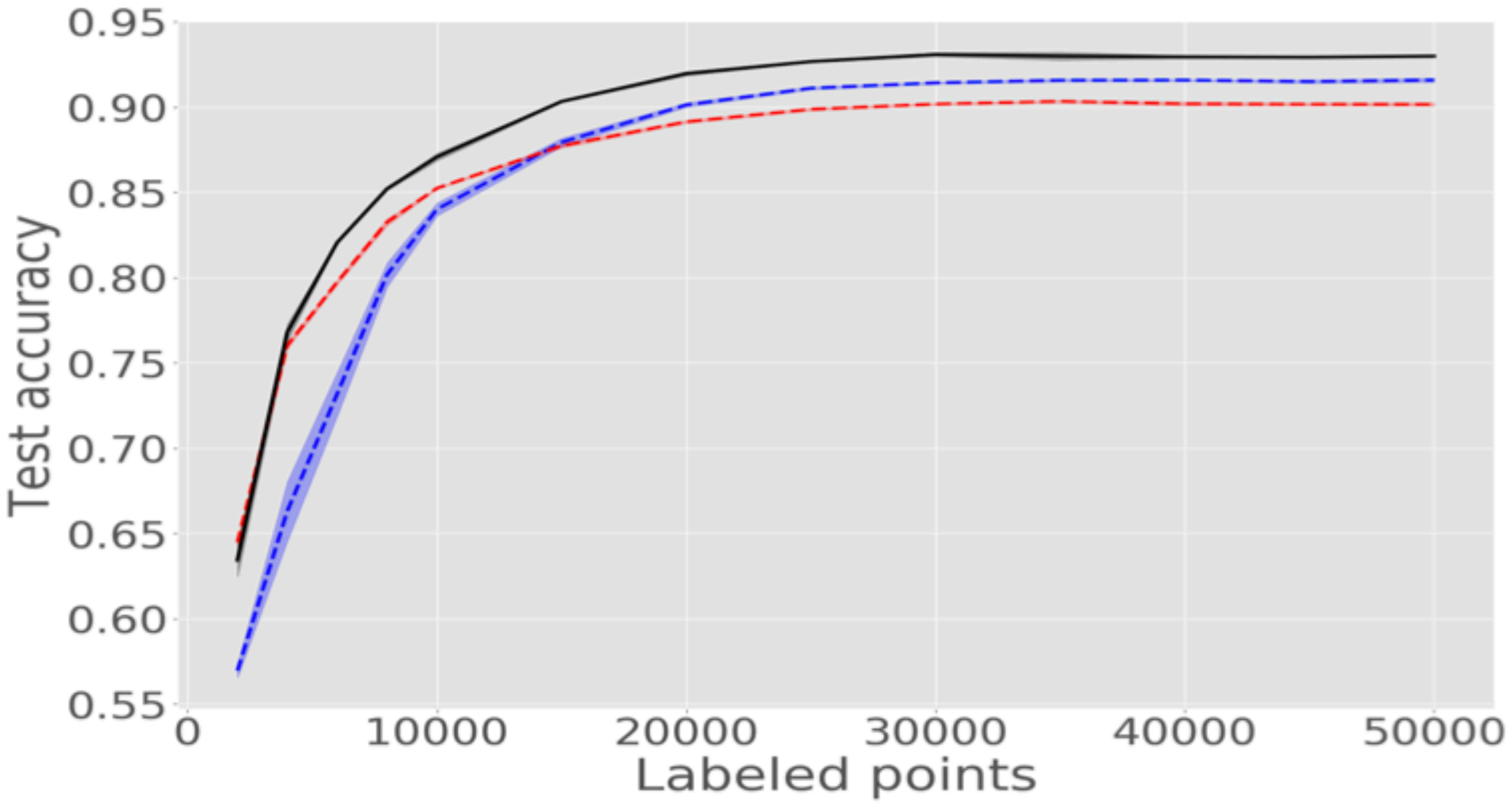




Geifman & El-Yaniv, “Deep Active Learning with a Neural Architecture Search”, NeurIPS 2019

Architecture & active learning

What kind of architecture do you think is depicted in the 3 curves?



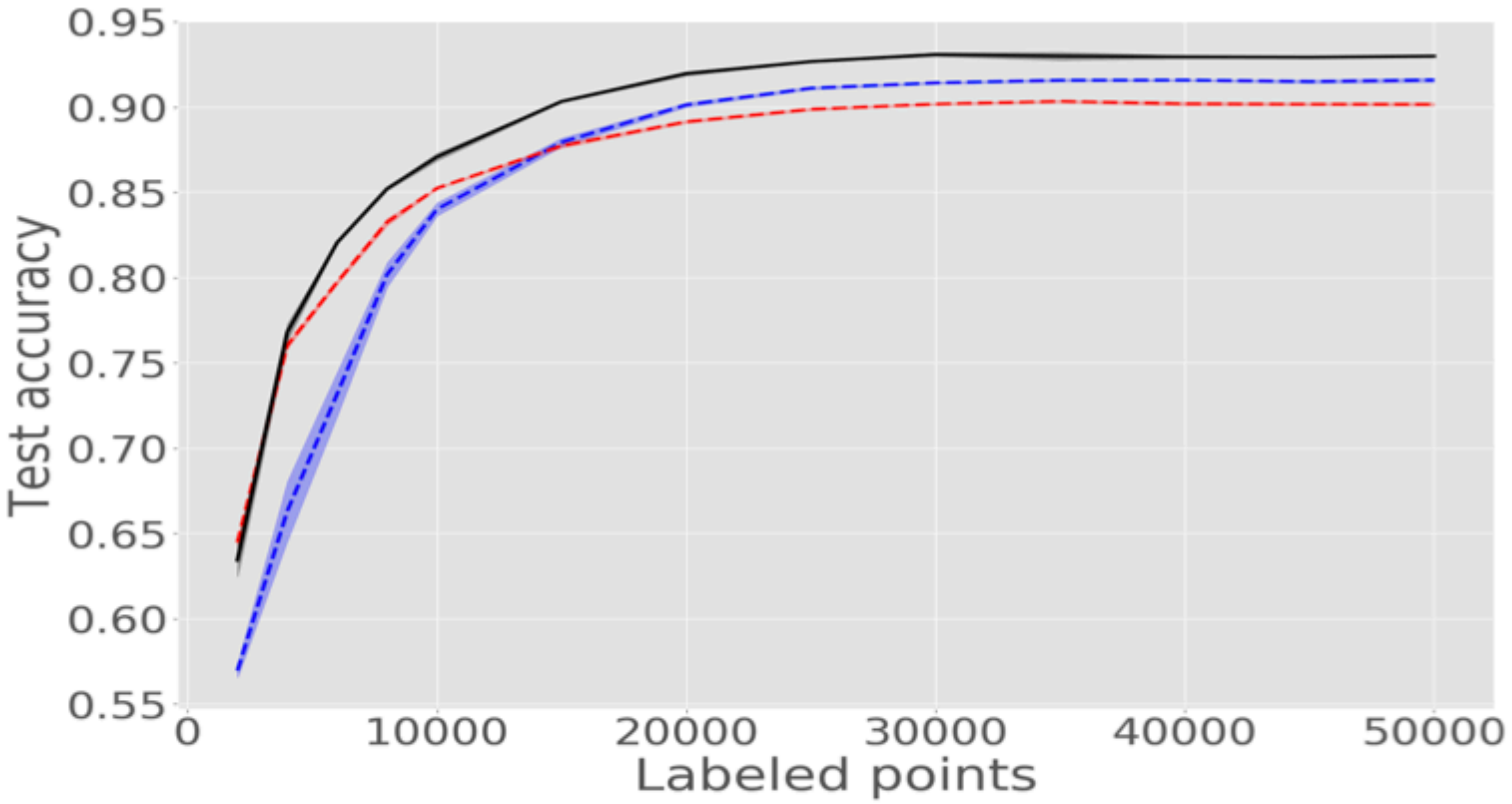
(a) Softmax response

Geifman & El-Yaniv, “Deep Active Learning with a Neural Architecture Search”, NeurIPS 2019

Architecture & active learning

What kind of architecture do you think is depicted in the 3 curves?

1. Black line: incremental architecture
2. Blue line: fixed Resnet (large)
3. Red line: fixed small architecture (start of the incremental one)

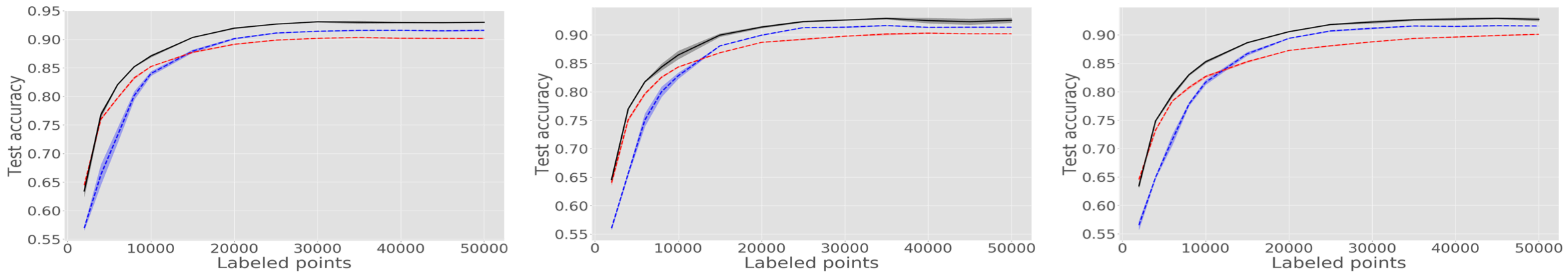


(a) Softmax response

Geifman & El-Yaniv, “Deep Active Learning with a Neural Architecture Search”, NeurIPS 2019

Architecture & active learning

Consistent for different active learning acquisition functions



(a) Softmax response (b) MC-dropout (c) Coreset

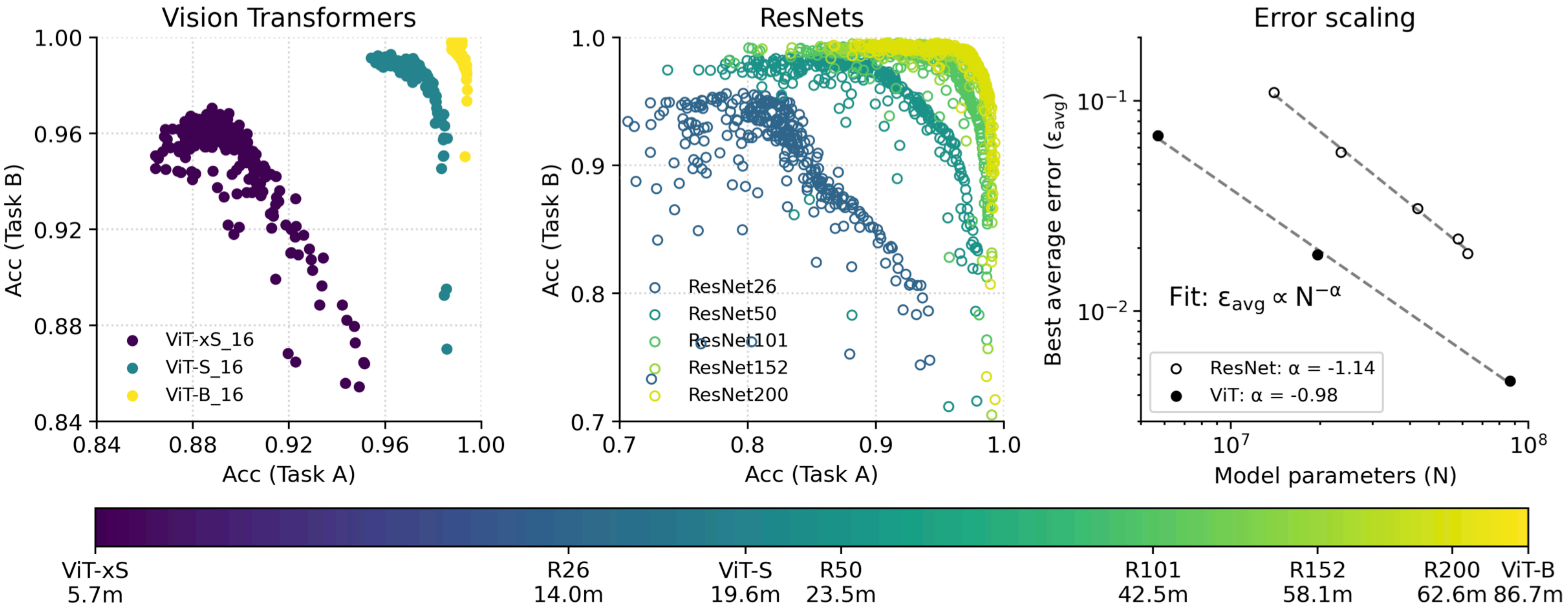
Figure 2: Active learning curves for CIFAR-10 dataset using various query functions, (a) softmax response, (b) MC-dopout, (c) coreset. In black (solid) – Active-iNAS (ours), blue (dashed) – Resnet-18 fixed architecture, and red (dashed) – $A(B_r, 1, 2)$ fixed.

Geifman & El-Yaniv, “Deep Active Learning with a Neural Architecture Search”, NeurIPS 2019

As always: it’s likely even more complicated

Choice of model & scale

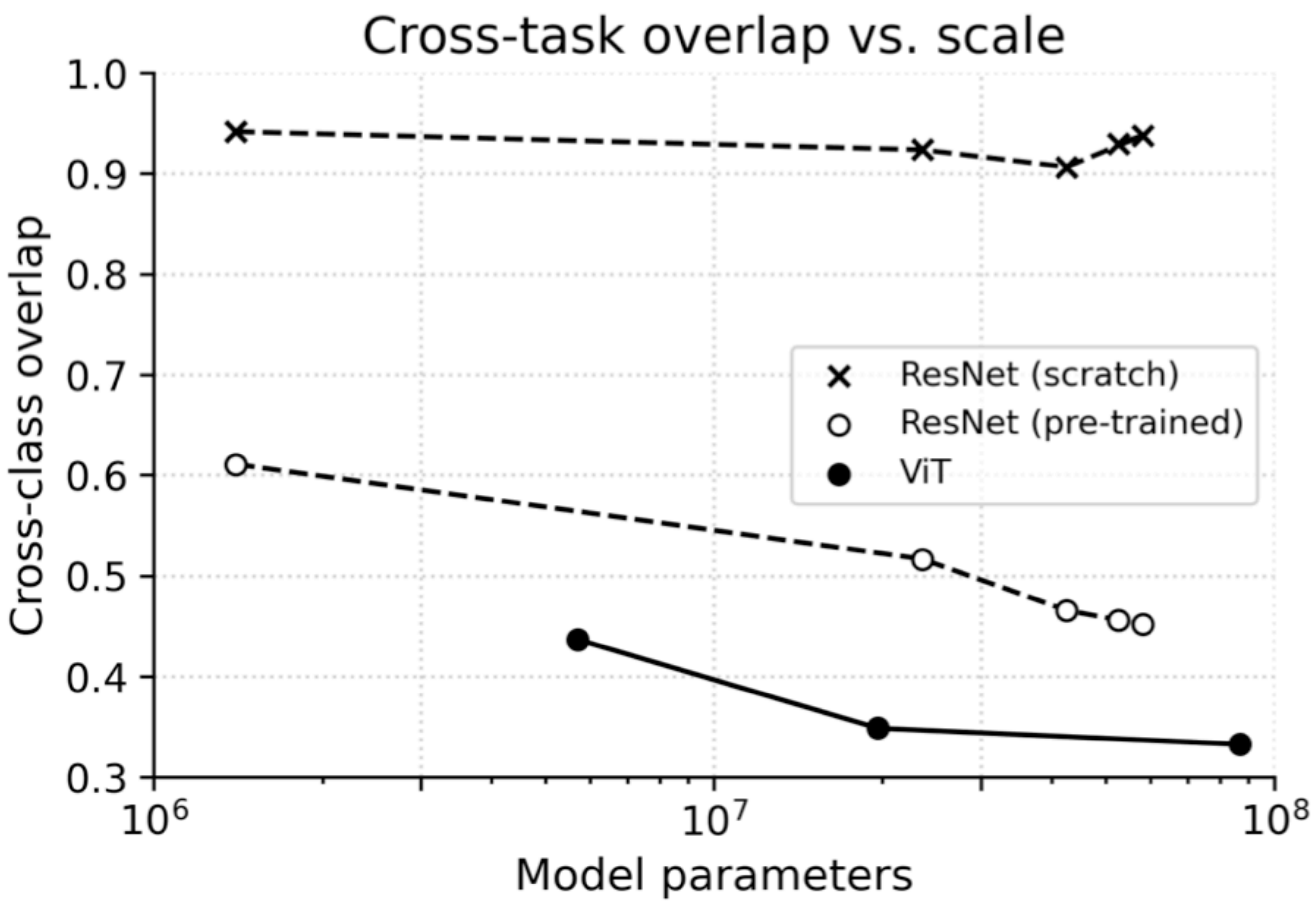
(Opinion?) We don't have a solid idea of representation overlap in deep learning yet

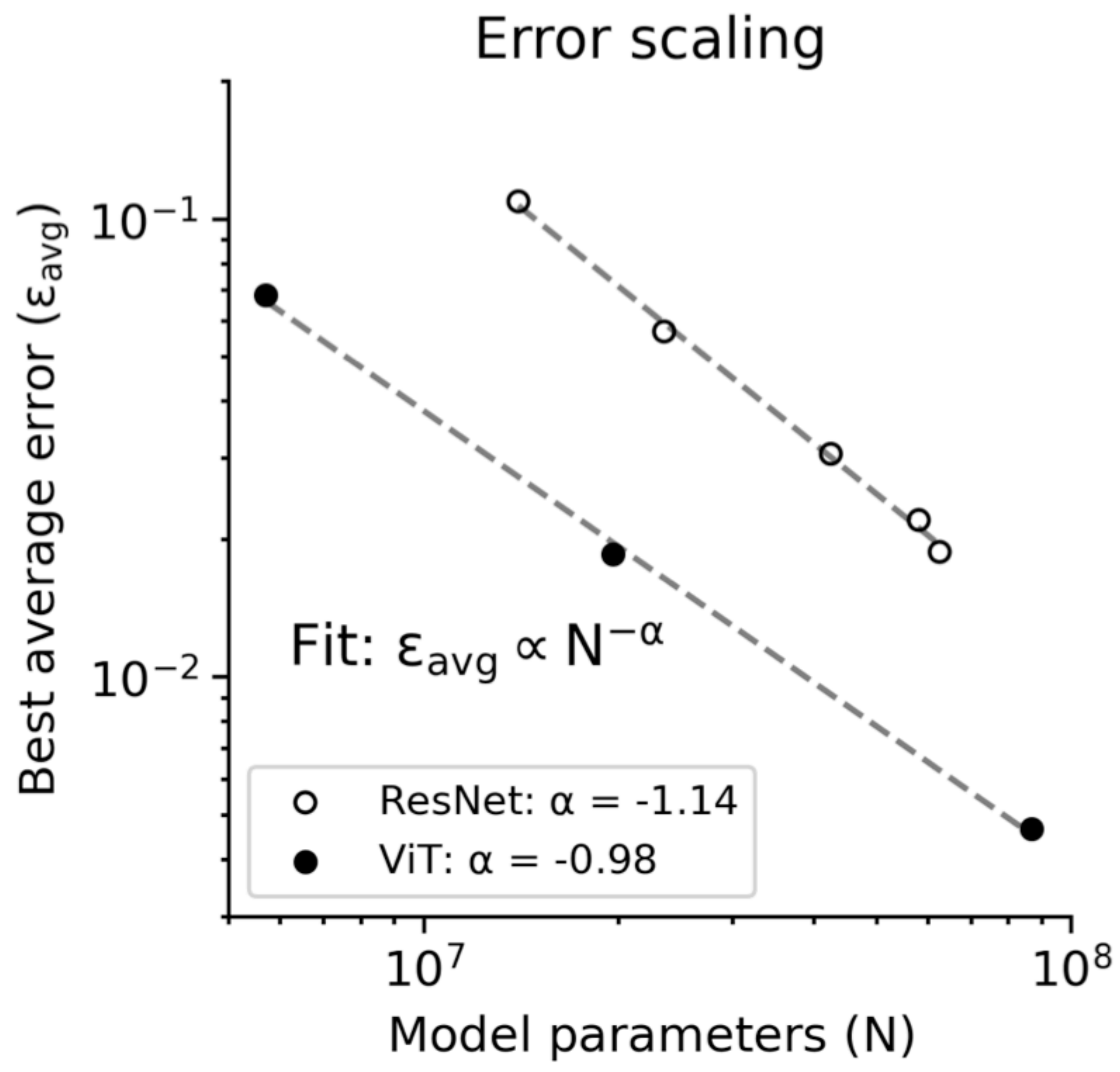


Ramasesh et al, "Effect of Model and Pretraining Scale on Catastrophic Forgetting in Neural Networks", ICLR 2022

Choice of model & scale

Some models may be more suitable than others: orthogonal representations?





Ramasesh et al, "Effect of Model and Pretraining Scale on Catastrophic Forgetting in Neural Networks", ICLR 2022

There are other ways to think about suitable architecture configurations

Meta-learning

The **meta-learning** perspective: **learning to learn**

- Learning to chose a suitable model variant
- Learning to grow
- Architecture search
- Learning loss functions
- Learning optimizers
- ….

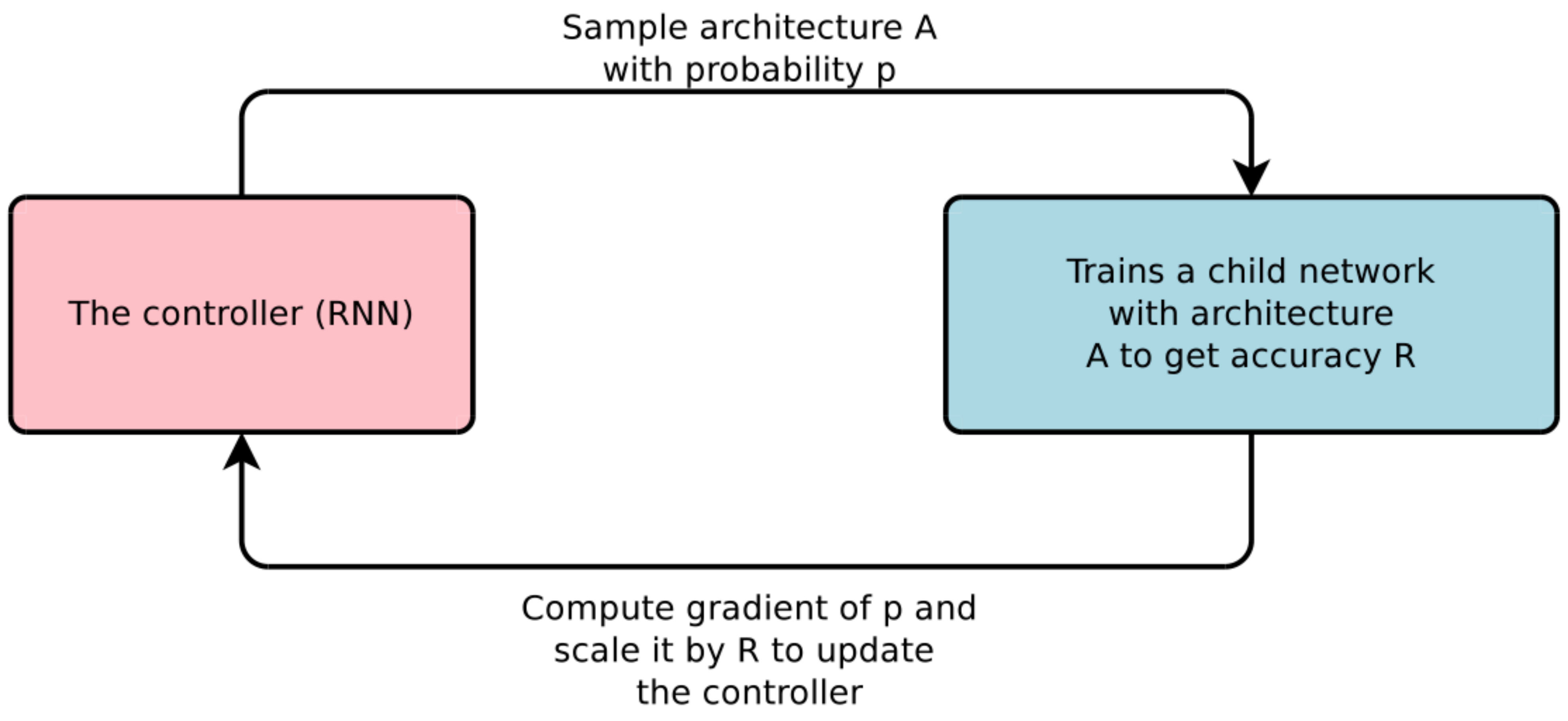


Figure 1: An overview of Neural Architecture Search.

Zopf & Le, “Neural Architecture Search with Reinforcement Learning”, ICLR 2017

It’s half-time: recap & outlook

What have we seen so far?

1. **Intro**: motivation and rough course overview
2. **Transfer** and its forms: from source to target tasks
3. (Catastrophic) **forgetting 1**: optimization, regularization, distillation
4. (Catastrophic) **forgetting 2**: rehearsal & pseudo-rehearsal
5. **Active learning**: querying what data comes next
6. **Dynamic/modular architectures**: more than just “forgetting 3”

We should have a good **initial overview of ways of thinking** & techniques now

Recap

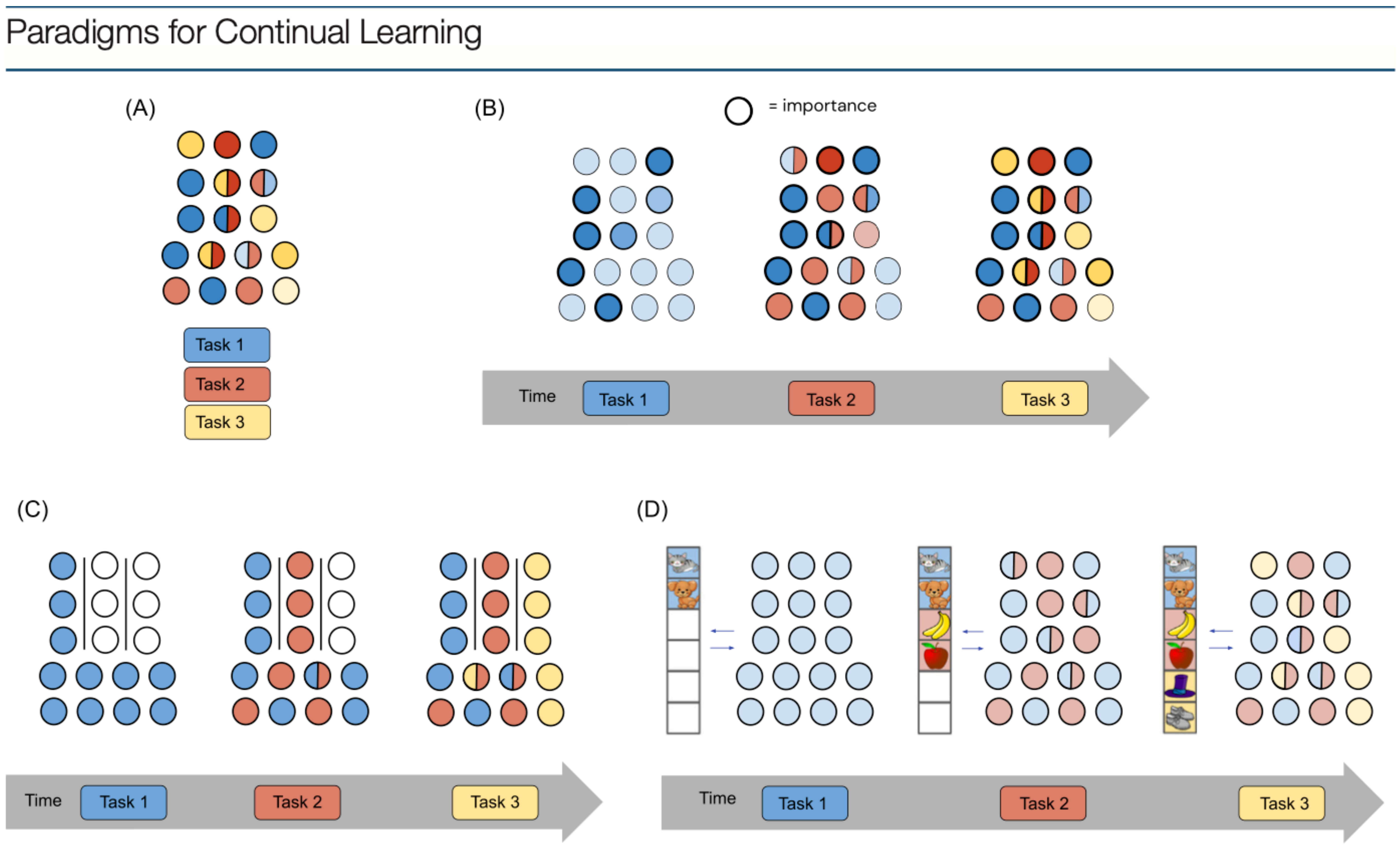


Figure 1. (A) Independent and identically distributed learning methods are standard for nonsequential, multitask learning. In this regime, tasks are learned simultaneously to avoid forgetting and instability. (B) Gradient-based approaches preserve parameters based on their importance to previously learned tasks. (C) Modularity-based methods define hard boundaries to separate task-specific parameters (often accompanied by shared parameters to allow transfer). (D) Memory-based methods write experience to memory to avoid forgetting.

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

We have covered these paradigms & a little more

What's still to come?

- **Learning curricula**
- **Large model intricacies**
- **Evaluation**
- **Open world learning**
- **Role of soft/hardware**
- **(Even more) Frontiers**