Machine Learning Beyond Static Datasets

ESSAI 2023

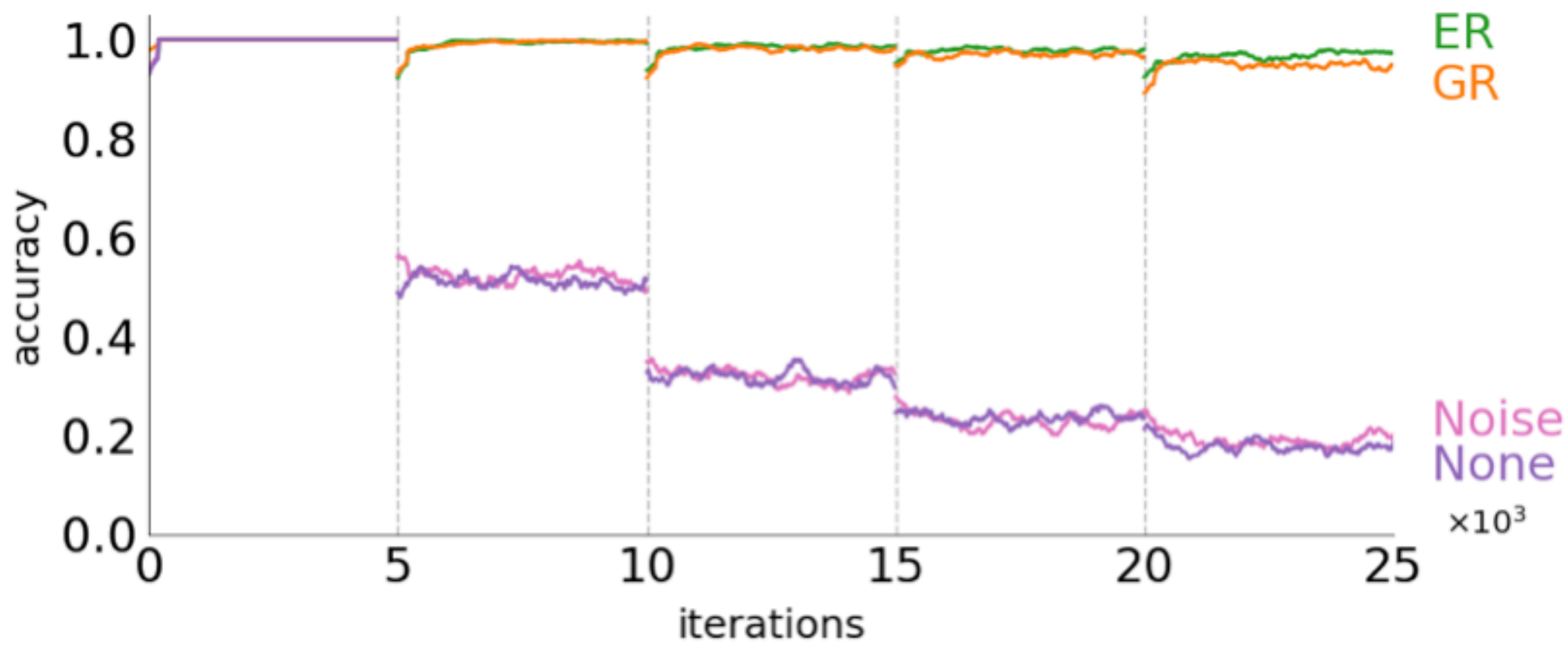
Dr. Martin Mundt,

Research Group Leader, TU Darmstadt & hessian.AI

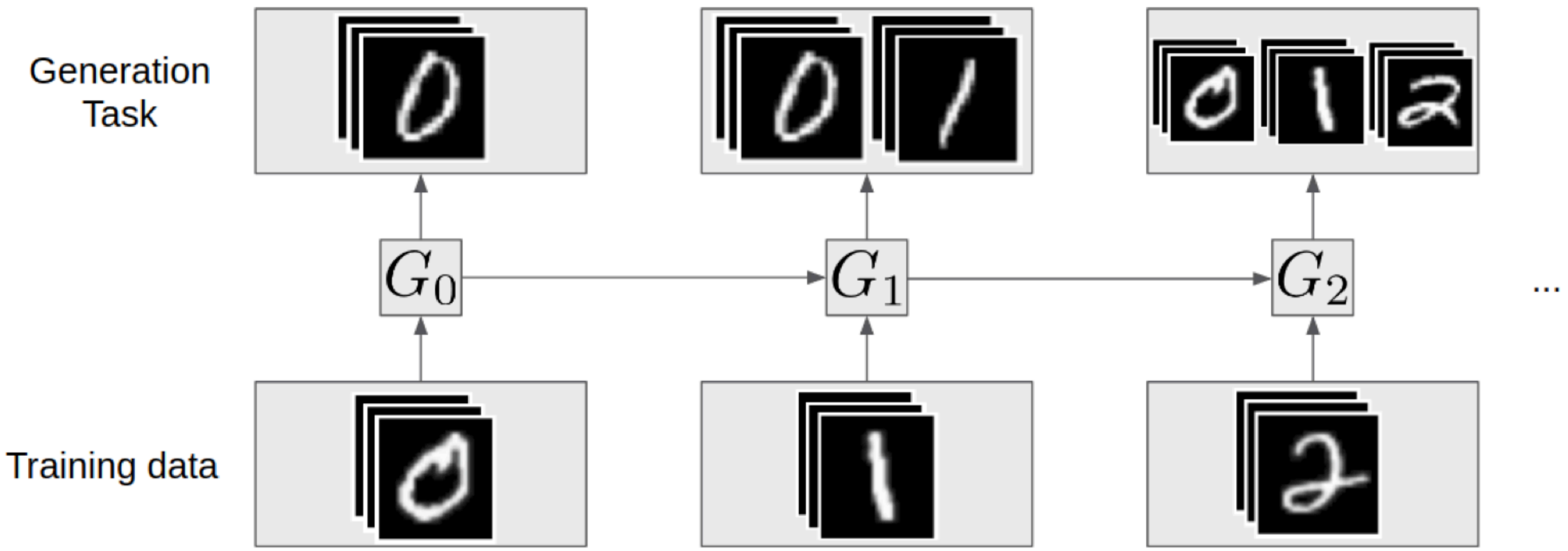
Board Member of Directors, ContinualAI

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

Day 3: From Past to Future Memory & Growth



Pseudo-rehearsal & why generative models fit both perspectives to avoid forgetting seen so far



What are generative models & why should we care?

- A **discriminative** model typically learns something like p(y|x)

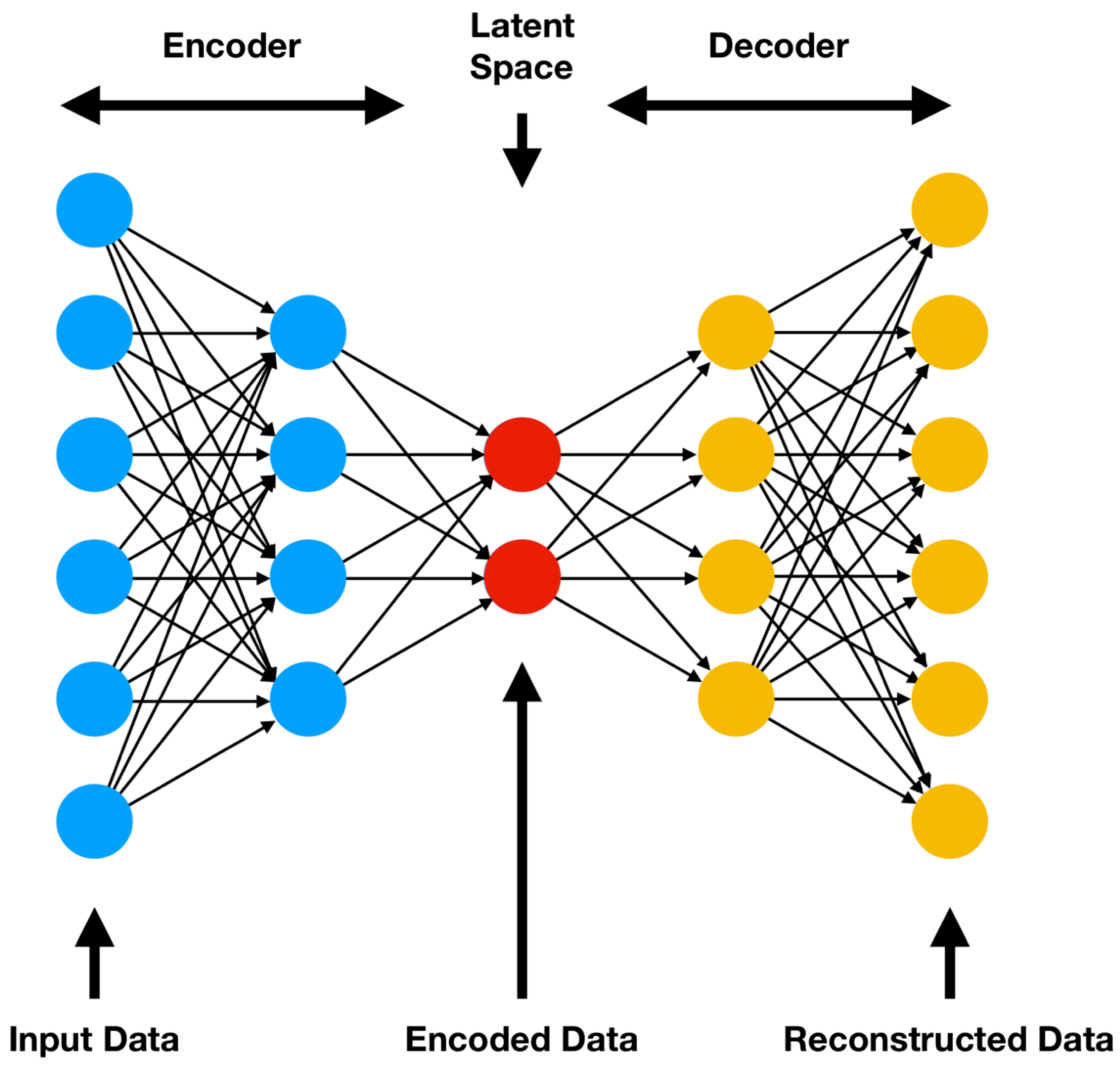
- A **generative** model also learns about the data distribution p(x) & the process by which data is created (the generative factors)

What are generative models & why should we care?

- A **discriminative** model typically learns something like $p(y|x)$
- A **generative** model also learns about the data distribution $p(x)$ & the process by which data is created (the generative factors)
- Having a generative model **does not mean we cannot also solve discriminative** tasks $p(x,y) = p(y|x)p(x)$

Let's pick one type of model specifically to go through:
(variational) autoencoders

Why Autoencoders? To see that we don't necessarily require two models in the ML perspective



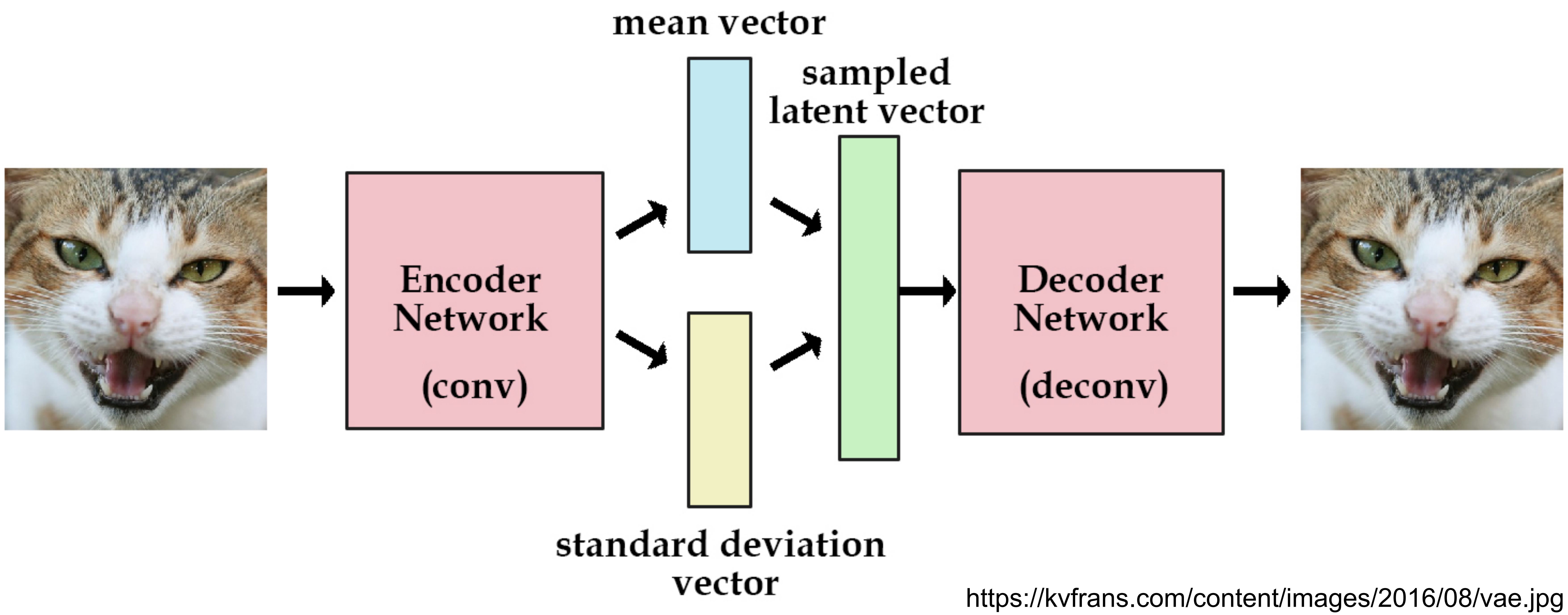
- Learn an "**encoding**" of the data
- Encoder maps to a "latent code"
- Decoder **reconstructs** the input

https://www.compthree.com/blog/autoencoder/

Variational Autoencoders (Kingma & Welling, ICLR 14)

The latent embedding/variables may be difficult to grasp if unconstrained.

But we could constrain the latent space to follow a specific distribution, e.g. a Variational Autoencoder



https://kvfrans.com/content/images/2016/08/vae.jpg

Skipping the VAE derivation to distill its essence

- A dataset with variable $\mathbf{x}$
- Data is generated by a random process involving unobserved random variable $\mathbf{z}$

Skipping the VAE derivation to distill its essence

- A dataset with variable **x**
- Data is generated by a random process involving unobserved random variable **z**
- **z** is generated from some prior distribution $p_\theta(z)$
- A value x is generated from some conditional distribution $p_\theta(x \mid z)$

Skipping the VAE derivation to distill its essence

- A dataset with variable **x**
- Data is generated by a random process involving unobserved random variable **z**
- **z** is generated from some prior distribution $p_\theta(z)$
- A value x is generated from some conditional distribution $p_\theta(x \mid z)$

But the parameters and values of latent variables **z** are not known to us.

$p_\theta(x) = \int p_\theta(x, z)dz$ is intractable

Skipping the VAE derivation to distill its essence

TL;DR; derivation: approximate & get a lower bound to data distribution p(x)

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - KL\left[q_\phi(z \mid x) \mid\mid p_\theta(z)\right]$$

Skipping the VAE derivation to distill its essence

TL;DR; derivation: approximate & get a lower bound to data distribution p(x)

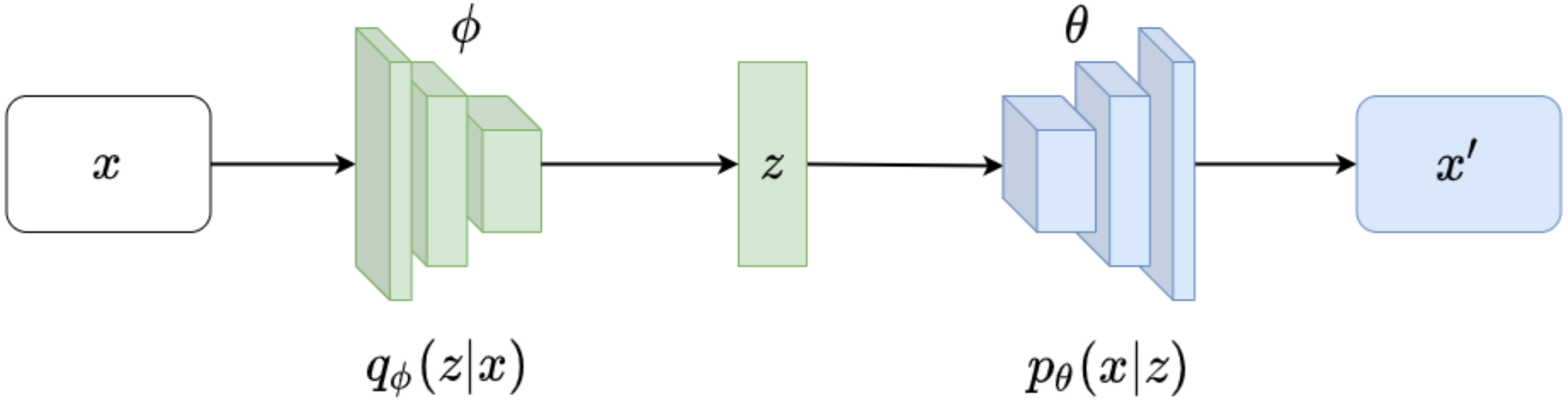
$$\mathscr{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)}\left[\log p_\theta(x\,|\,z)\right] - KL\left[q_\phi(z\,|\,x)\,||\,p_\theta(z)\right]$$

- The 1. term is the expected **reconstruction error** given by the log-likelihood (with sampling)
- The 2. term is a **KL divergence** encouraging the "approximate posterior" to be close to a prior (of our choice)

VAE: summary

- **Probabilistic encoder** -> given a datapoint x it produces a distribution over possible values of z from which it could have been generated
- **Probabilistic decoder** -> produces a distribution over possible values of x given z

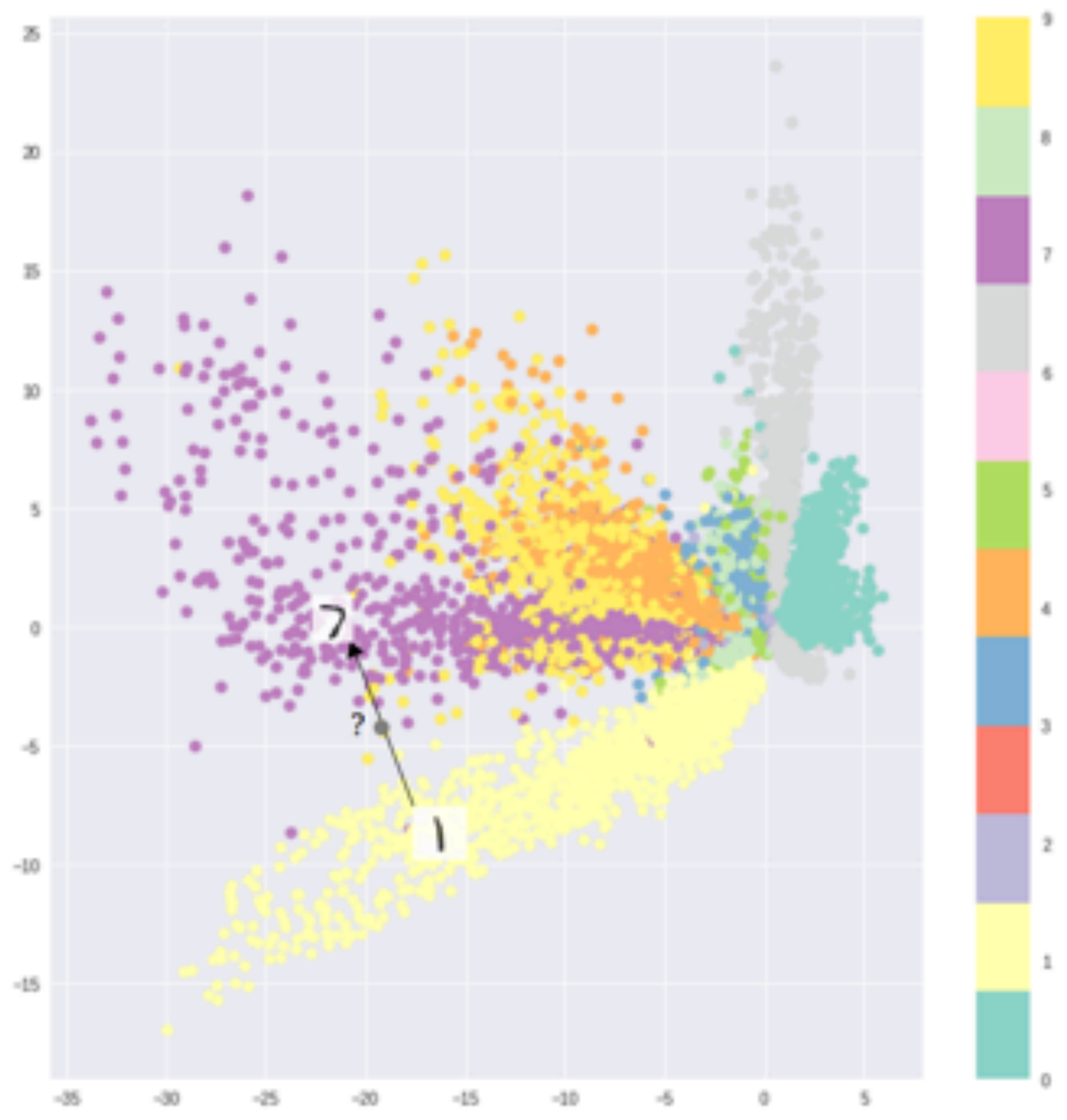
$$\mathrm{KL}[q_\phi(z|x) \| p_\theta(z)] \qquad \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$



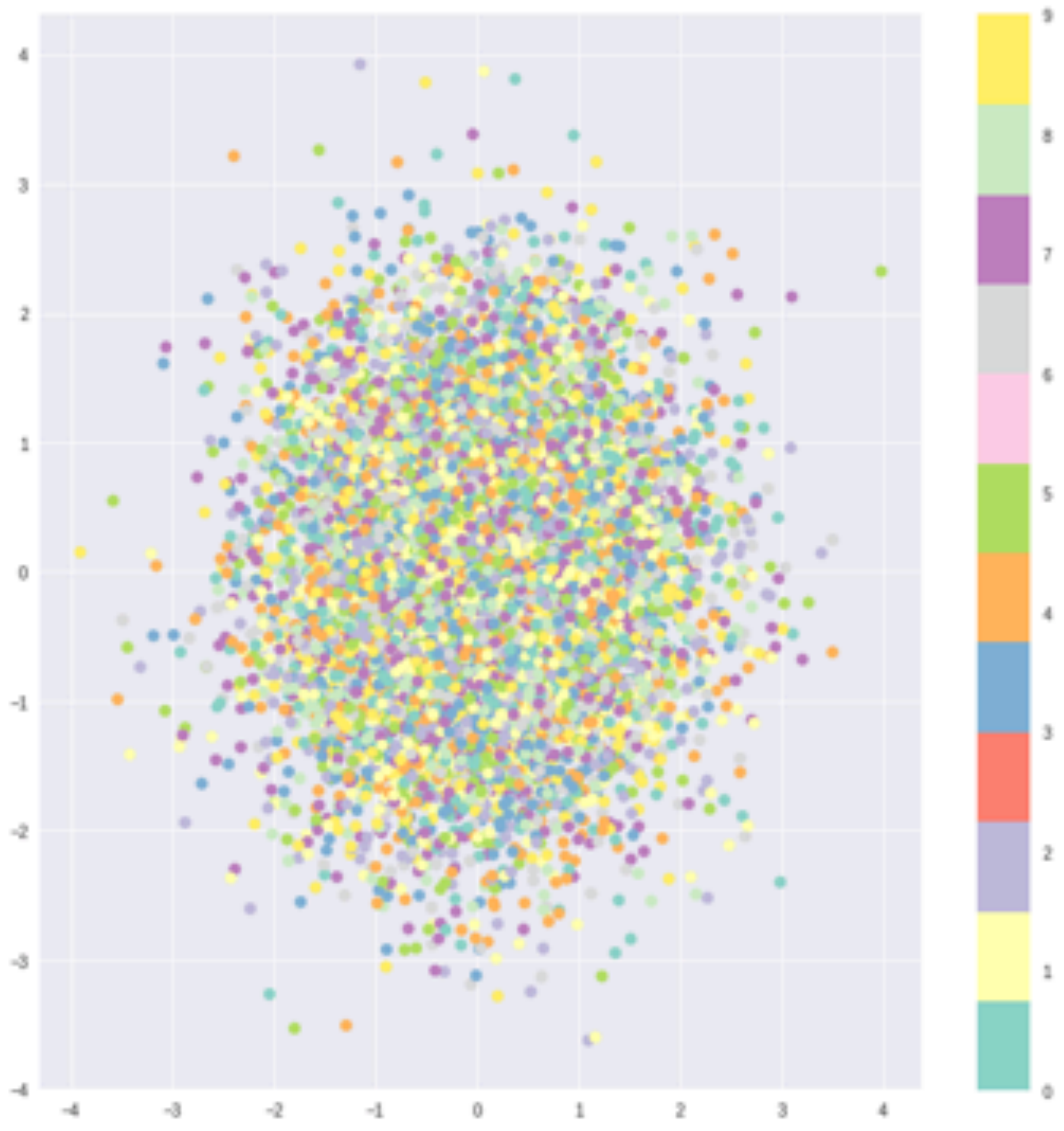
How does this model help us in avoiding forgetting?

What have we gained?

Only reconstruction loss



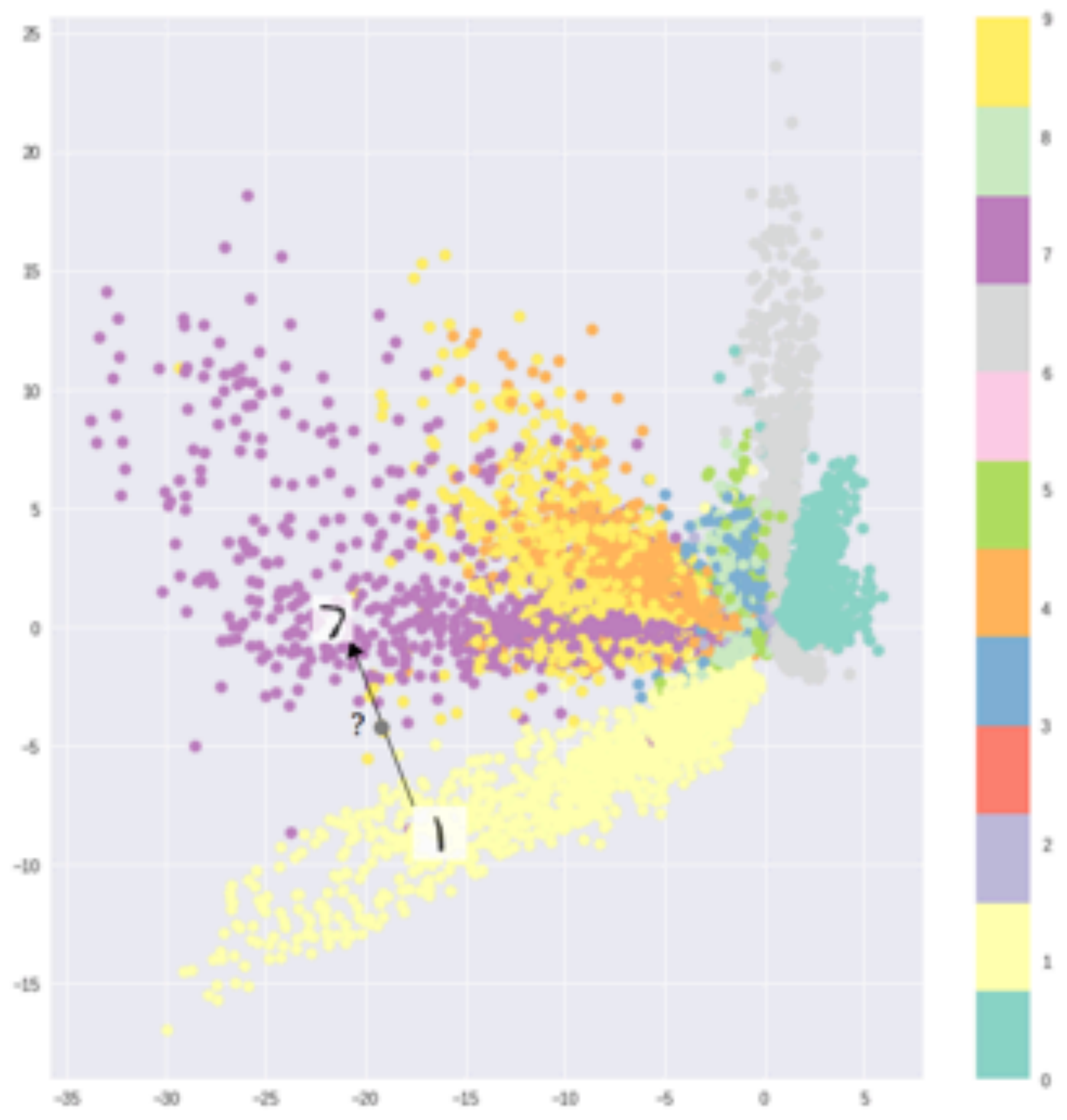
Only KL divergence



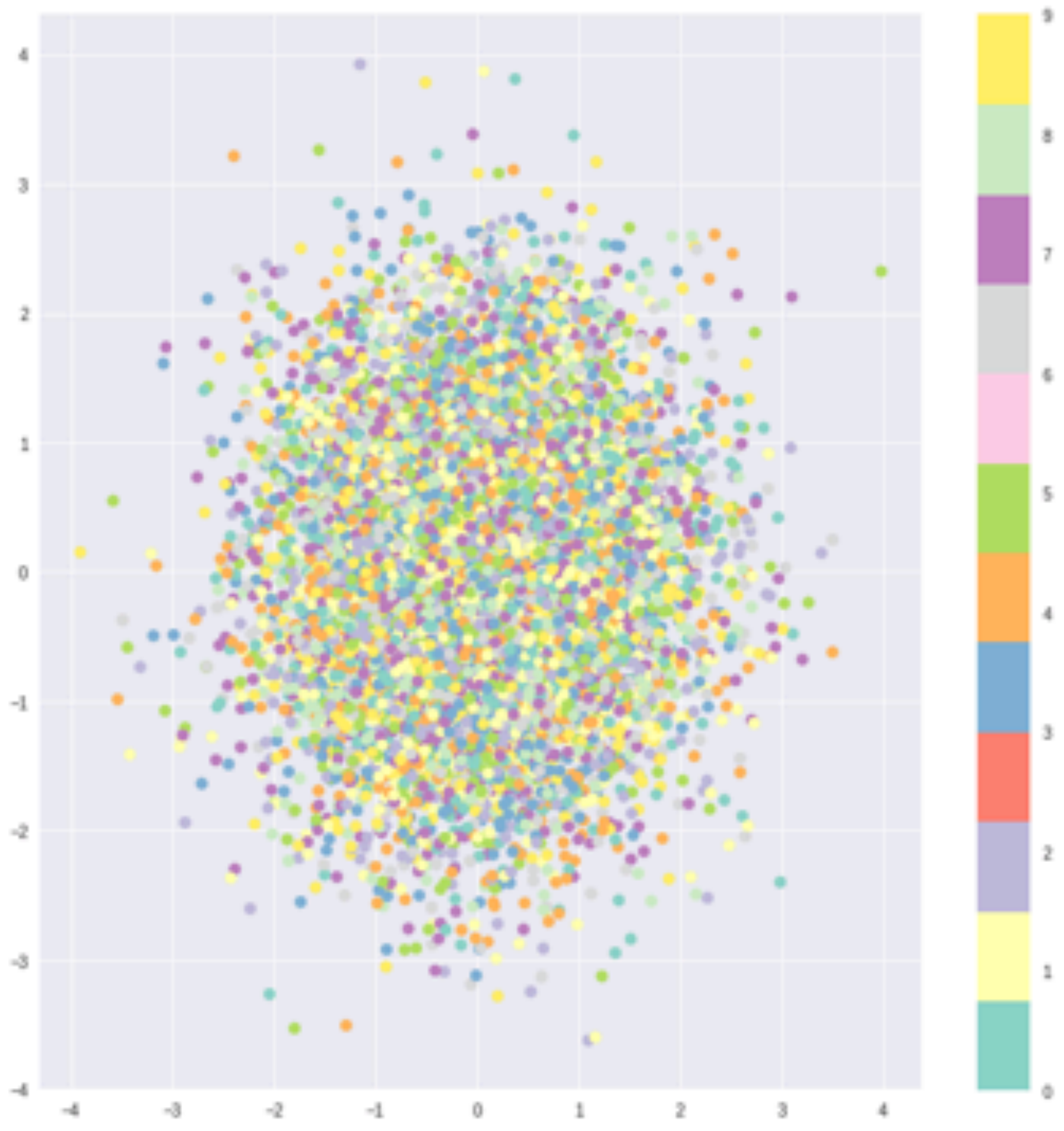
https://www.jeremyjordan.me/variational-autoencoders/ and https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

What have we gained?

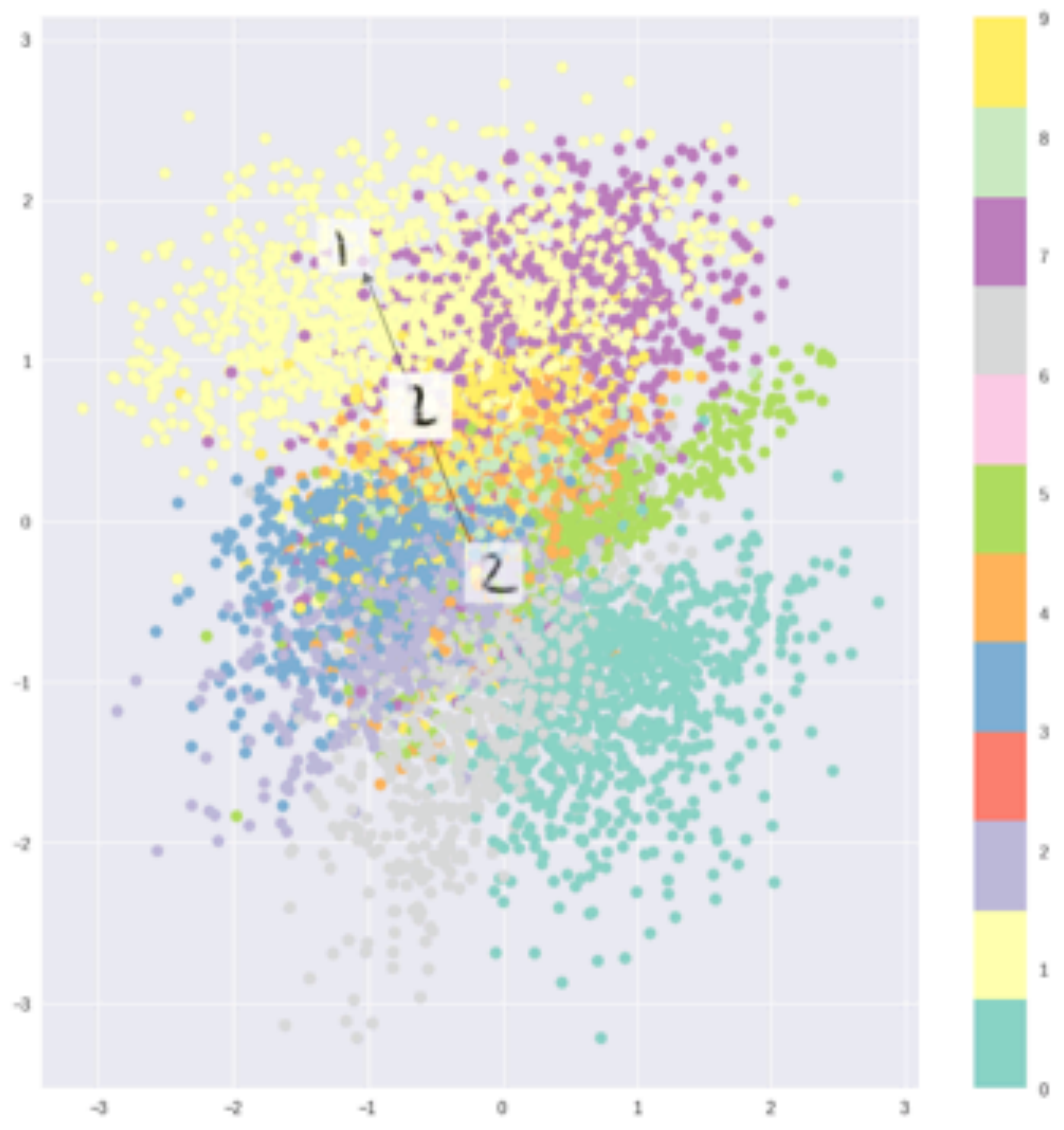
Only reconstruction loss



Only KL divergence



Combination



https://www.jeremyjordan.me/variational-autoencoders/ and https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

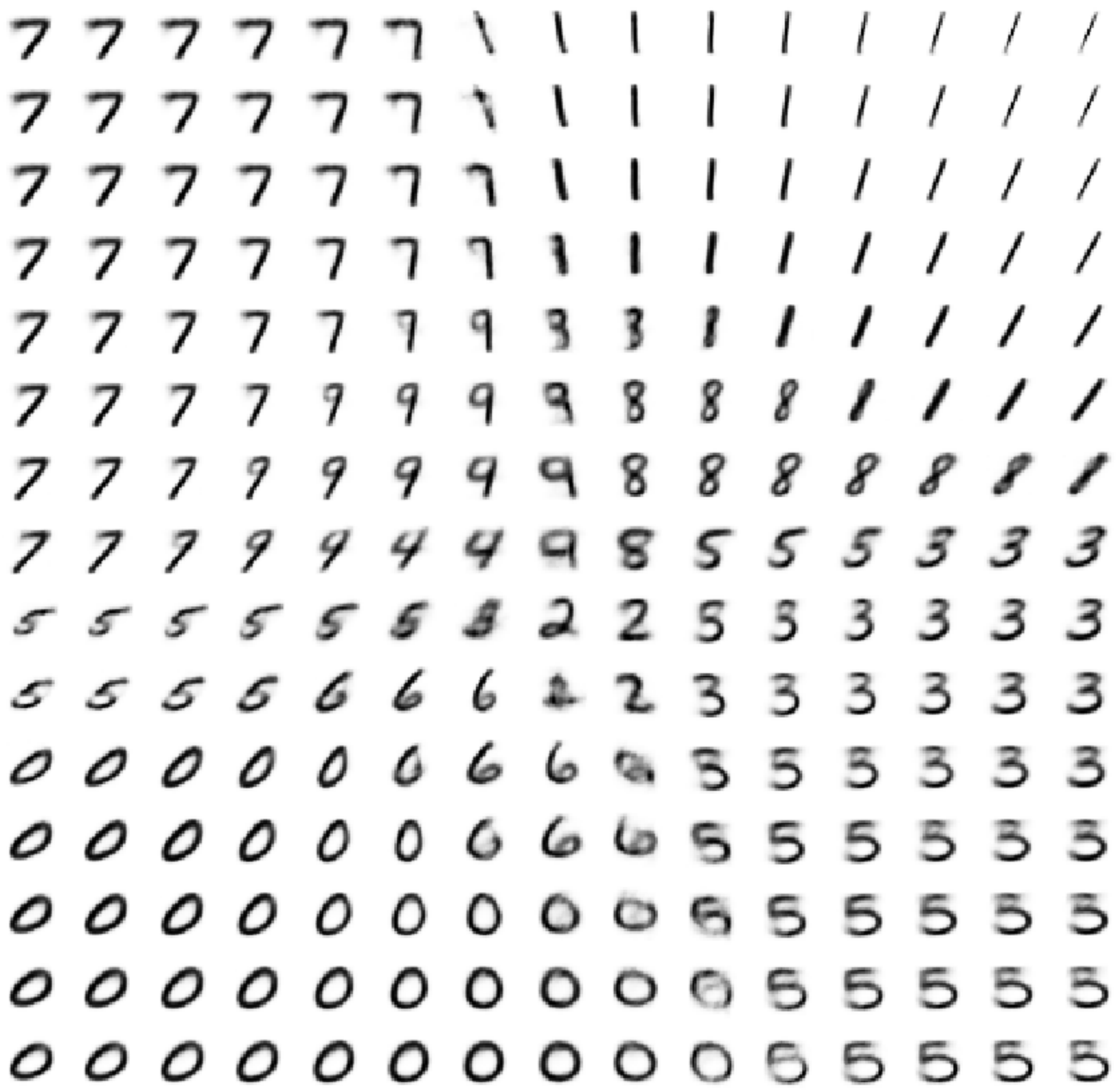
What have we gained?

- We can sample from a trained model:

  $z \sim p(z)$, here $\mathcal{N}(0,I)$,

  and then **generate** (decode) x

- We also have the approximation to our data distribution p(x) that we could **regularize** in continual learning



https://www.compthree.com/blog/autoencoder/

Variational Continual Learning

$$\mathscr{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \,|\, z)\right] - KL \left[q_\phi(z \,|\, x) \,||\, p_\theta(z)\right]$$

The "likelihood focused" perspective:

generative/pseudo rehearsal

- Generate old tasks' data and
  concatenate it with new task data
- Primarily optimize "the likelihood" (left)

See Nguyen et al, "Variational Continual Learning" ICLR 2018 & follow-ups like Farquhar et al "A Unifying Bayesian View of Continual Learning", NeurIPS workshops 2018

Variational Continual Learning

$$\mathscr{L}(\theta, \phi; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \,|\, z)\right] - KL \left[q_\phi(z \,|\, x) \,||\, p_\theta(z)\right]$$

The "likelihood focused" perspective:

generative/pseudo rehearsal

- Generate old tasks' data and concatenate it with new task data
- Primarily optimize "the likelihood" (left)

The "prior focused" perspective:

regularization/distillation

- Only use new task data
- Use the posterior of an old task as the new task's prior $KL\left[q_t(z) \,||\, q_{t-1}(z)\right]$

See Nguyen et al, "Variational Continual Learning" ICLR 2018 & follow-ups like Farquhar et al "A Unifying Bayesian View of Continual Learning", NeurIPS workshops 2018

Variational Continual Learning

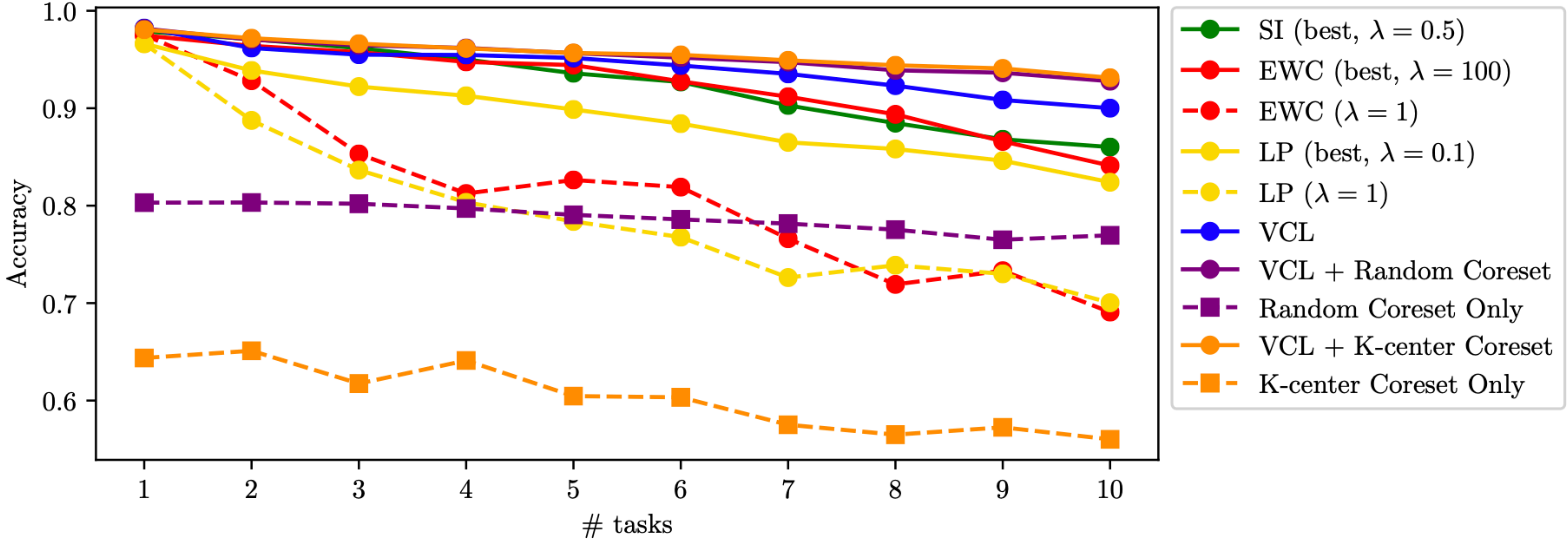


Figure 2: Average test set accuracy on all observed tasks in the Permuted MNIST experiment.

Nguyen et al, “Variational Continual Learning” ICLR 2018

Both perspectives are valuable, but storing data is not always desired & can be a “trivial" solution. What do we desire?

Let's summarize: what could we want?

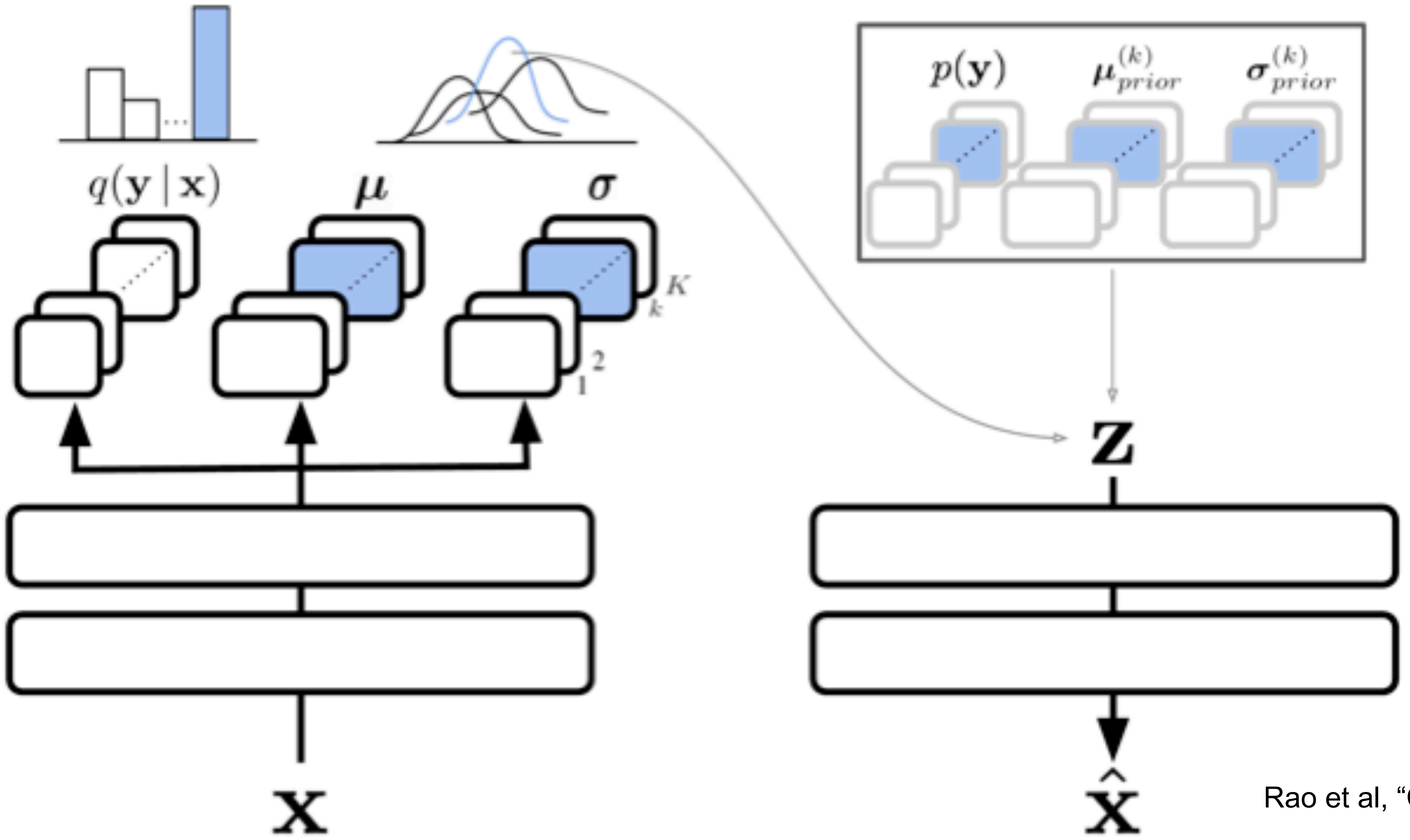
What could our expectations be, what might we desire?

- Constant memory budget?
- Pragmatically? Selection that outperforms randomly stored data points?
- A way to shrink the memory buffer to add new tasks, e.g. recursively select exemplars?
- Knowledge of the distribution(s) and a subset with guarantees?
- A natural formulation to allow (pseudo-)rehearsal, regularization…?
- …. many more …?

There’s a third way to think about forgetting

CURL: task specific Gaussians

In our example: we can now use task-specific priors - a “Gaussian" per task



Rao et al, “Continual Unsupervised Representation Learning”, NeurIPS 2019

The third pillar of forgetting: dynamic architectures

In essence: we are looking at dynamic/modular architectures

"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

The third pillar of forgetting: dynamic architectures

In essence: we are looking at dynamic/modular architectures

“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. … you can’t have it both ways.”

Variant A: Implicit Dynamic/Modular Architectures

The implicit perspective

The **“implicit" perspective**

- Recall regularization: identify important parameters, constrain those

➡We could assume over-parametrization + try to “sparsify” our parameters

➡Route through “sub-models” that are responsible for a specific task

The implicit perspective: activation overlap

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

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

The implicit perspective: activation overlap

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

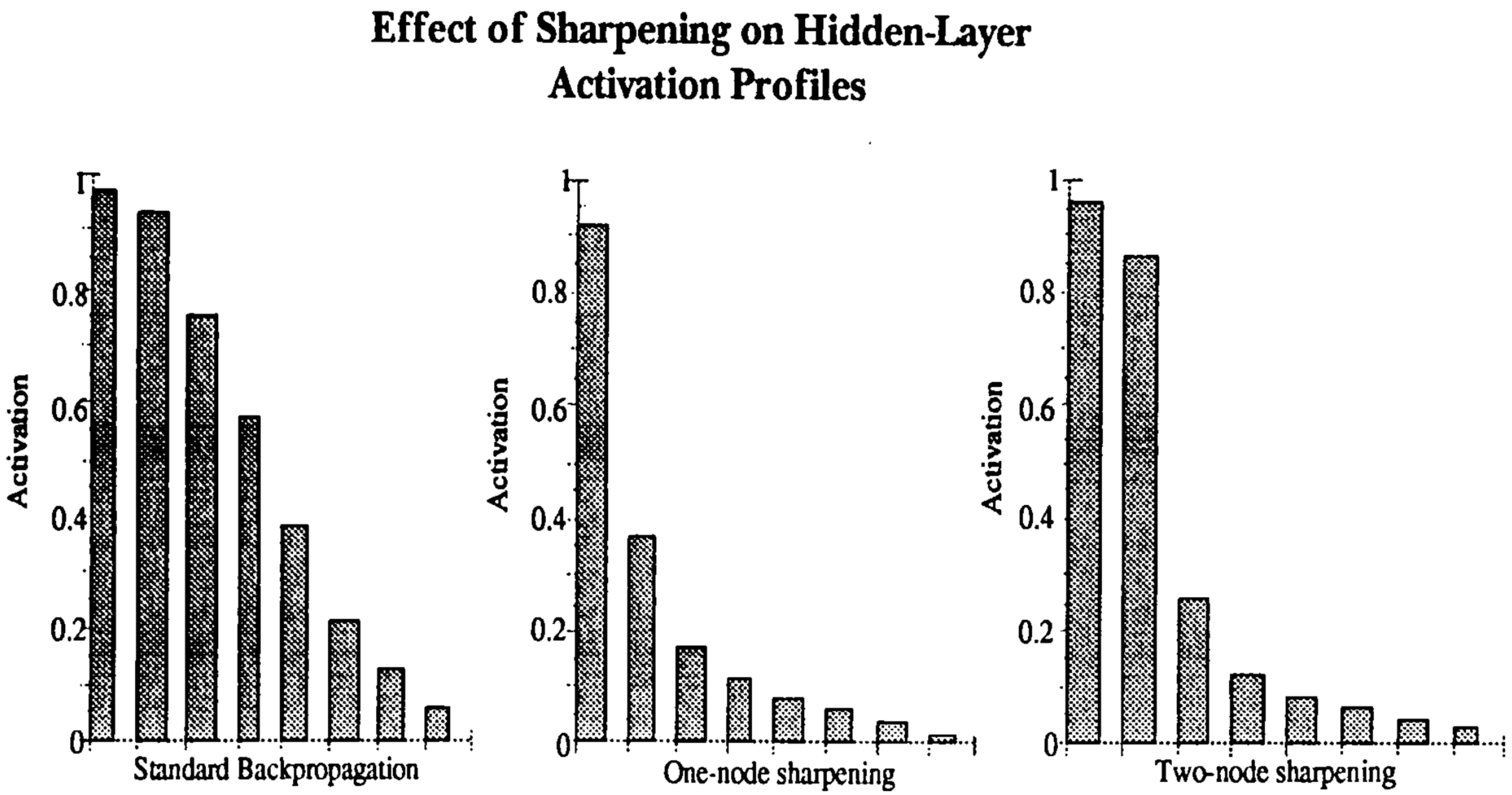
The implicit perspective: activation overlap

An algorithm? increase activation of k nodes, decrease that of others

- 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.

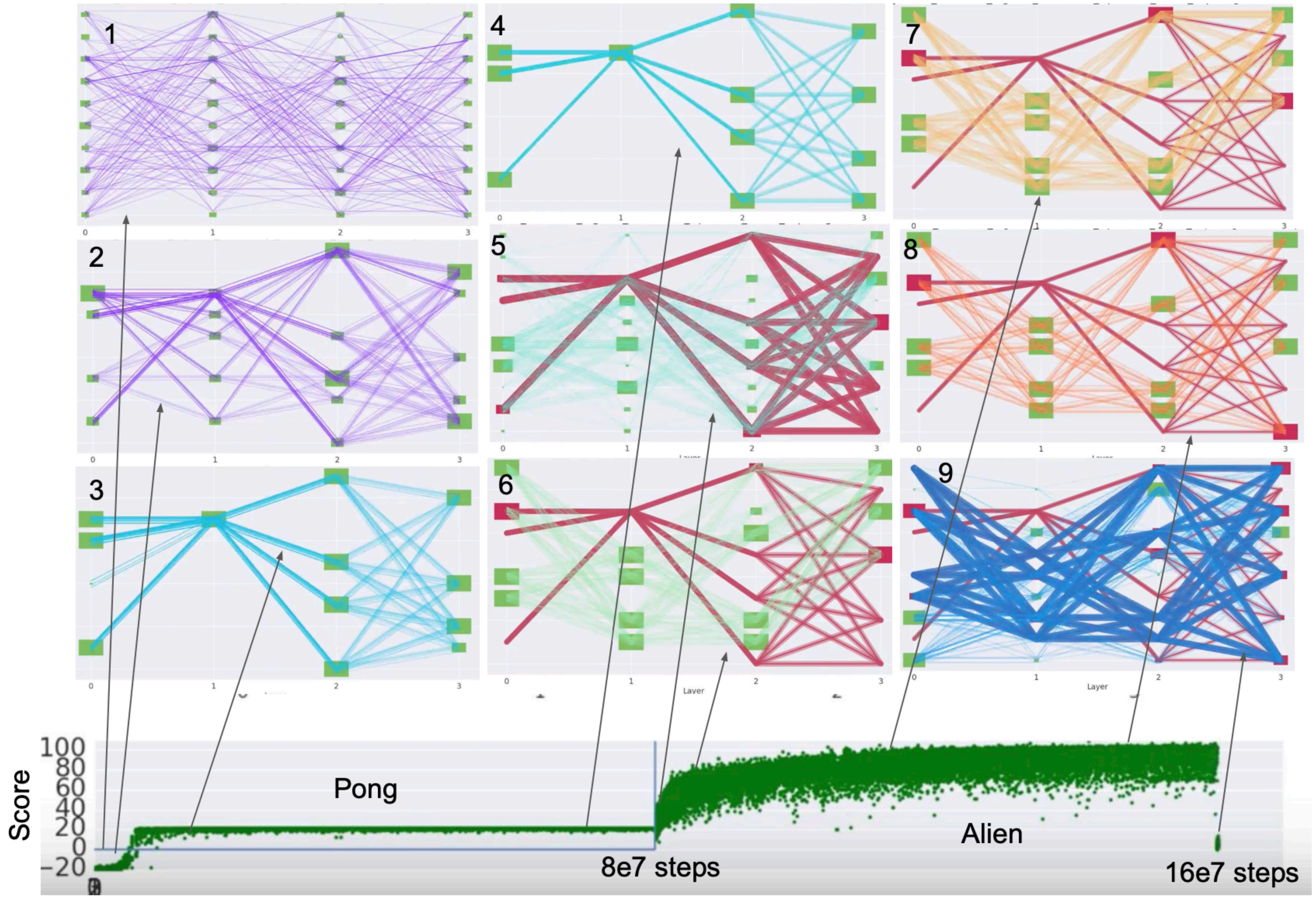
Robert French, "Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks", AAAI 1993

The implicit perspective: activation overlap



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

The implicit perspective: activation overlap



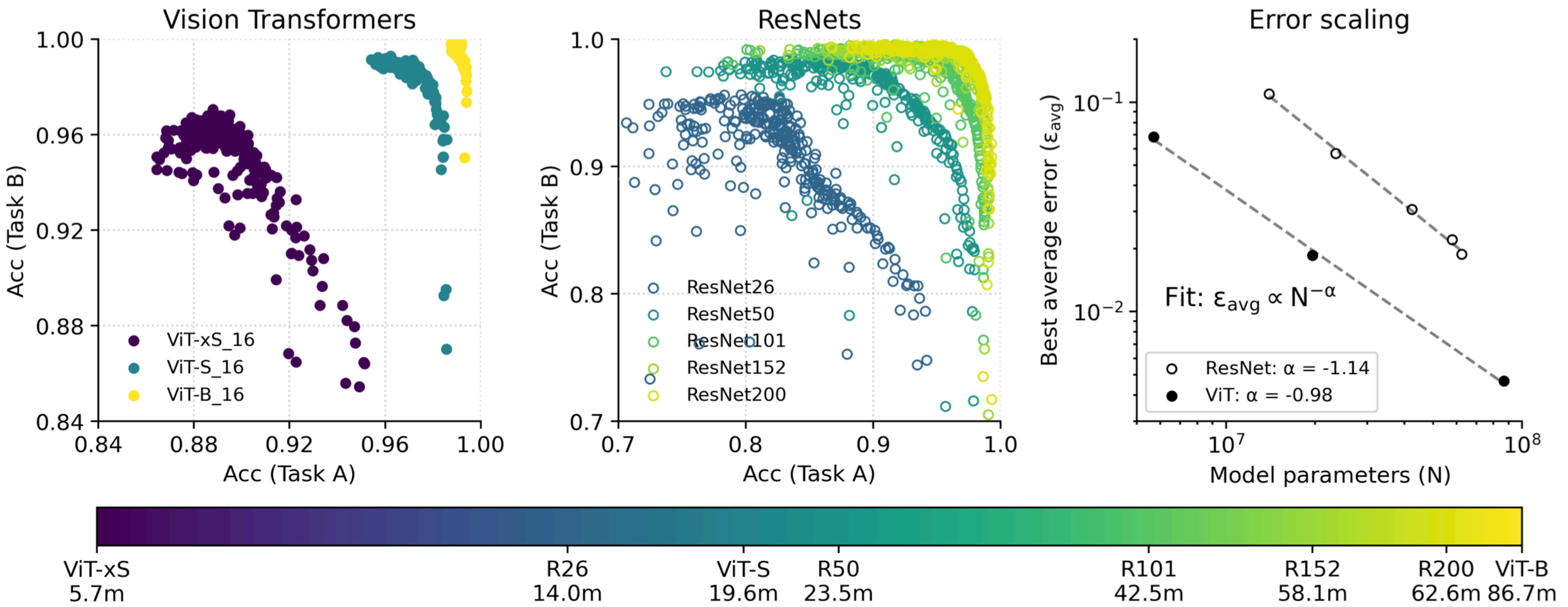
A newer example: **PathNets**

- 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: choice of model & scale

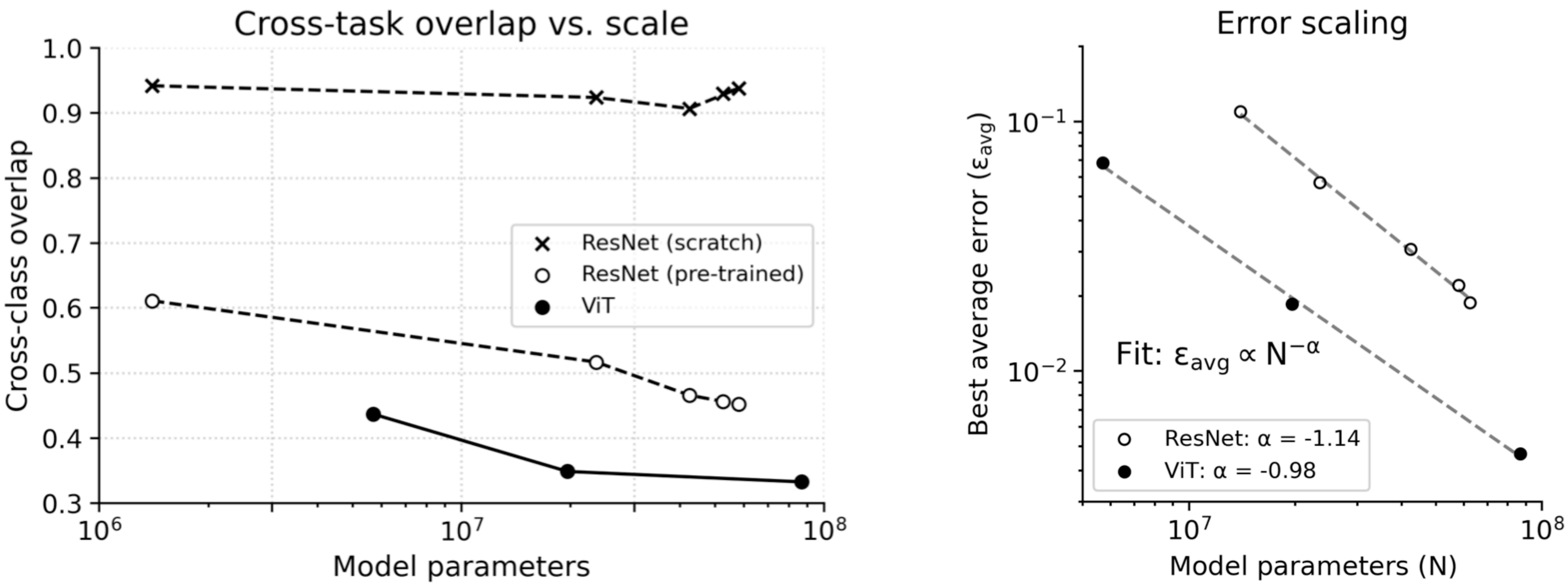
We still need better notions of representation overlap (in deep learning)



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

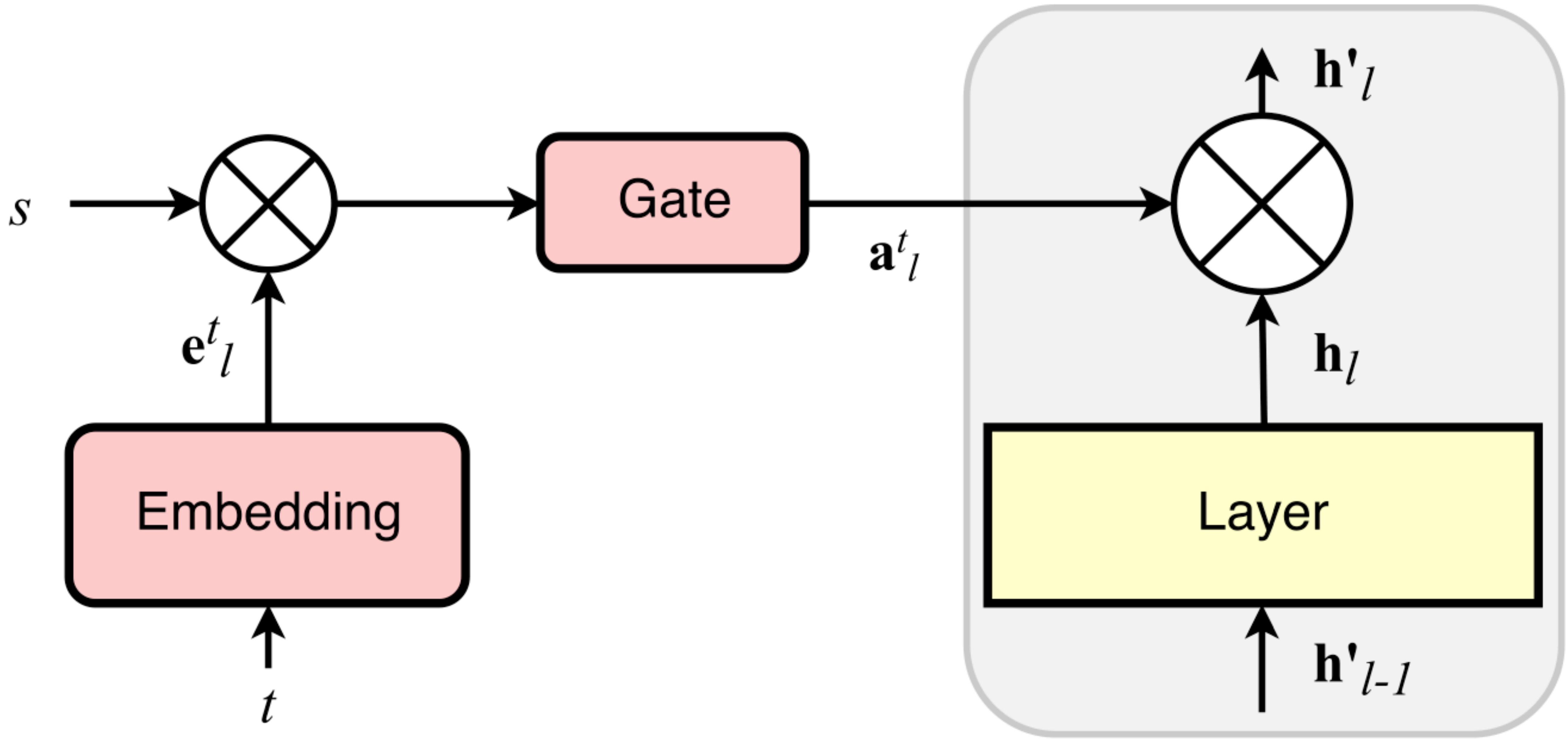
The implicit perspective: choice of model & scale

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

Summary: “implicit” (over-parametrized) perspective



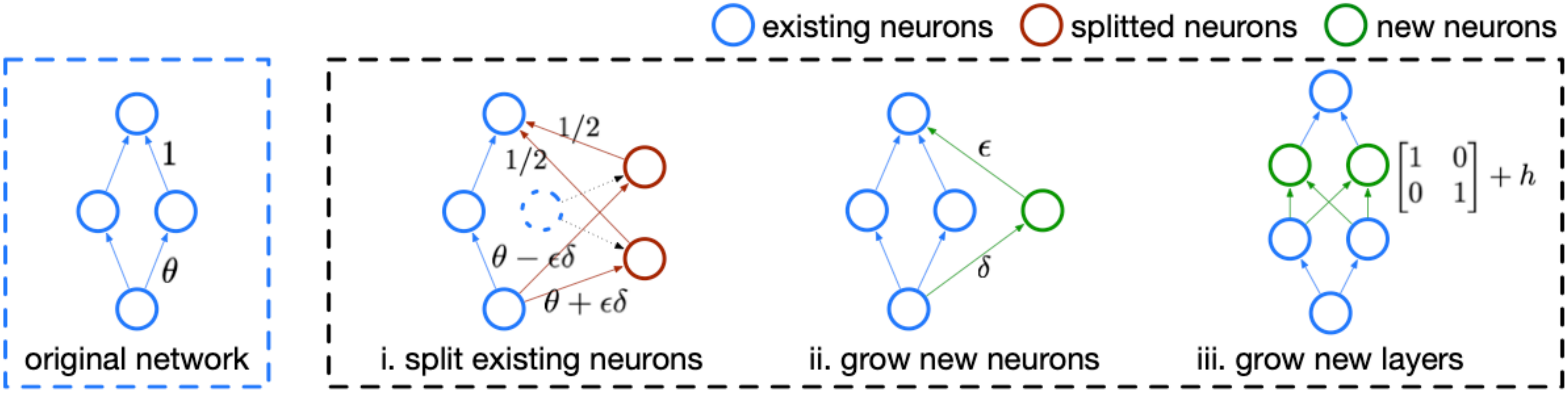
There are many ways to go about task specific subsets of parameters/modules:

- Activation overlap
- Parameter sparsity
- “Attention” masks
- “gates"… etc.

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

Surely interesting, but what about energy & compute?
Variant B: Starting small & growing explicitly

Explicit perspective: changing (neural) model architectures over time



Our initial model choice & its practical realization may not good enough anymore. Complexity might change, inductive bias might be altered …

Wu & Liu et al, "Firefly Neural Architecture Descent: A General Approach for Growing Neural Networks", NeurIPS 2020

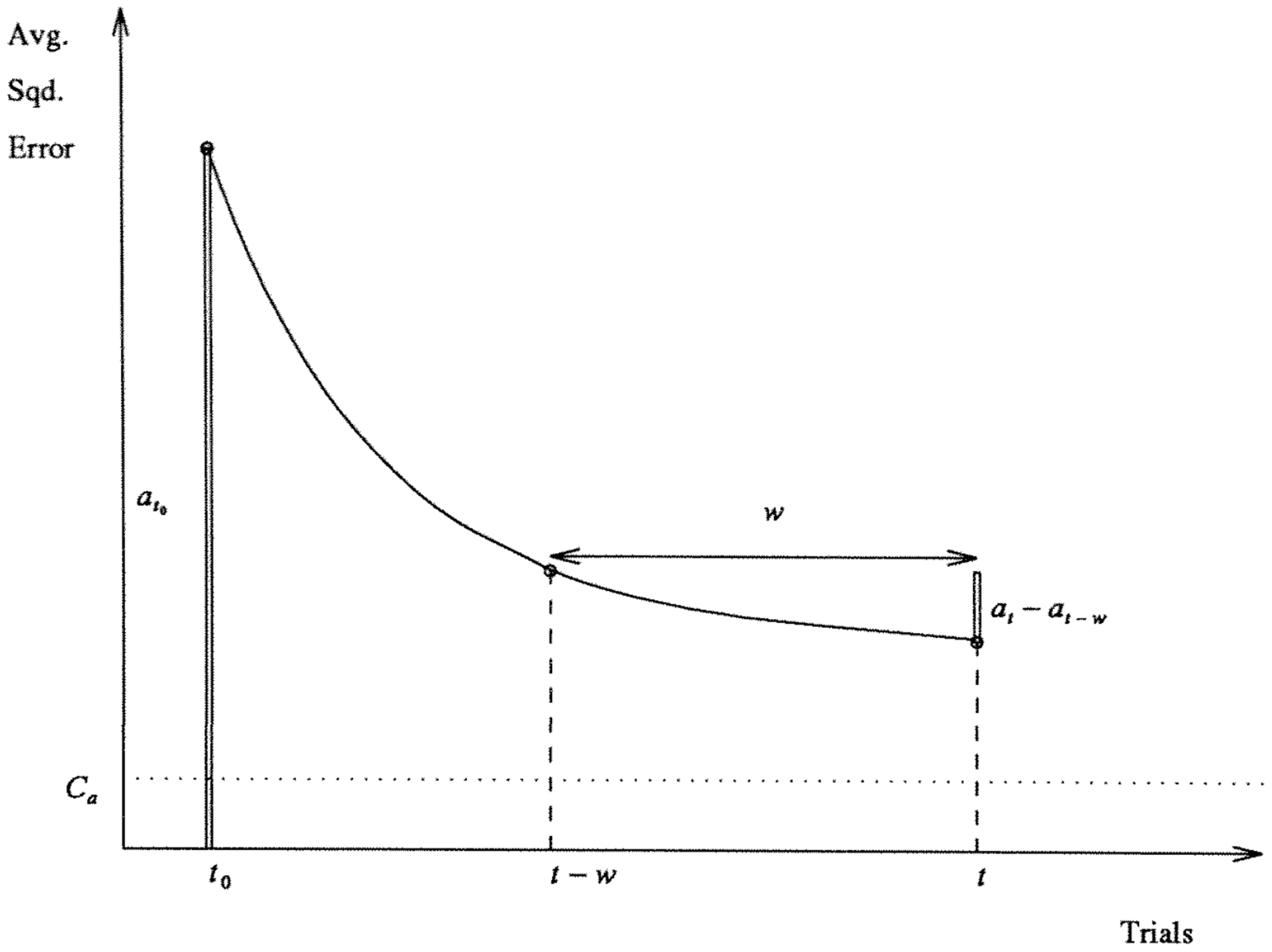
Explicit perspective & neurogenesis

“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)

Inspiration from neurogenesis: dynamic node creation



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

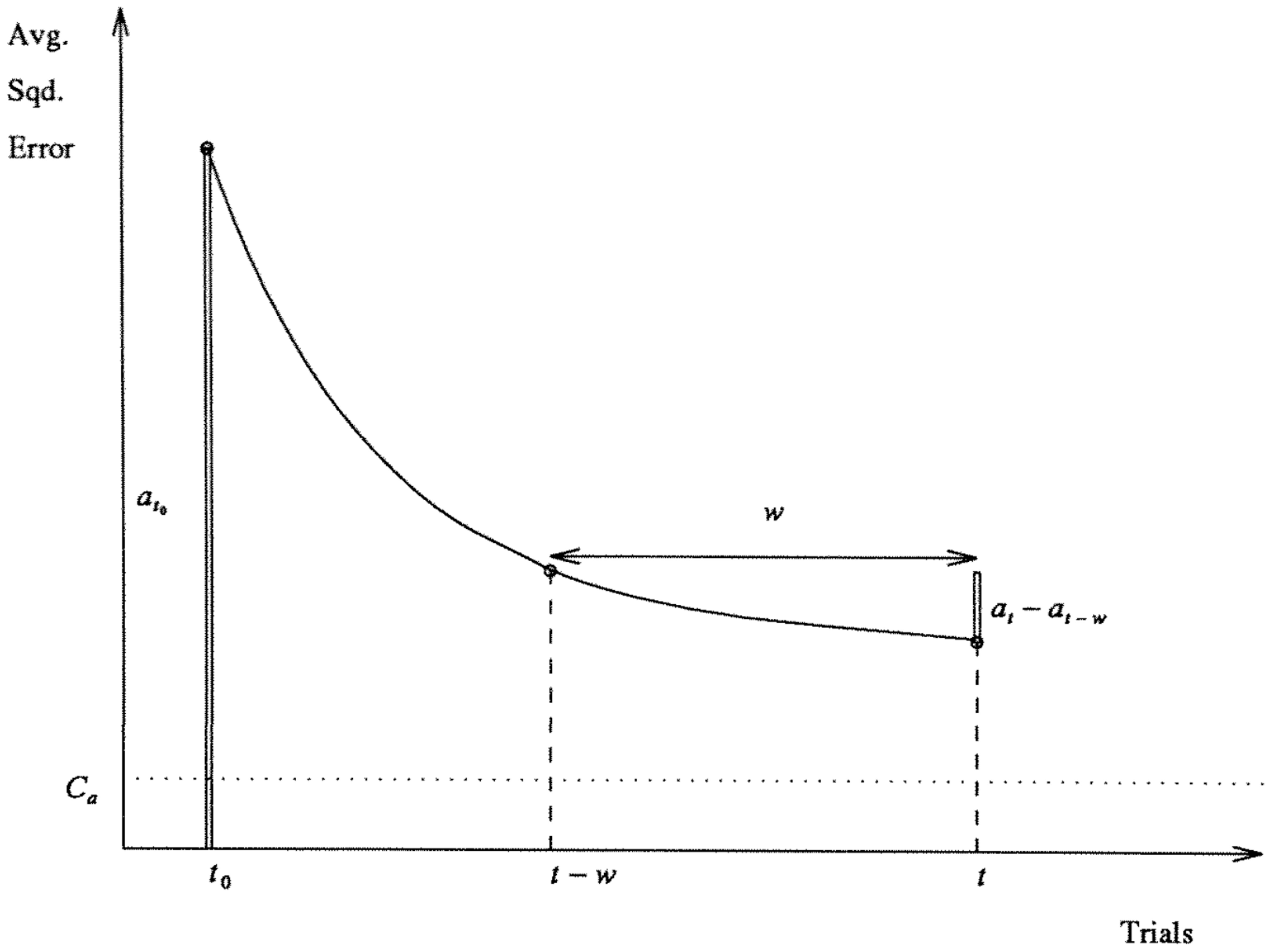
Small initial amount of parameters!

1st crucial question:

When do we add?

- Assumes decaying exponential for the error
- Adds node when error plateaus

Inspiration from neurogenesis: dynamic node creation



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

2nd question: **when do we stop?**

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

Inspiration from neurogenesis: dynamic node creation

Has been empirically investigated on some “simpler” test problems

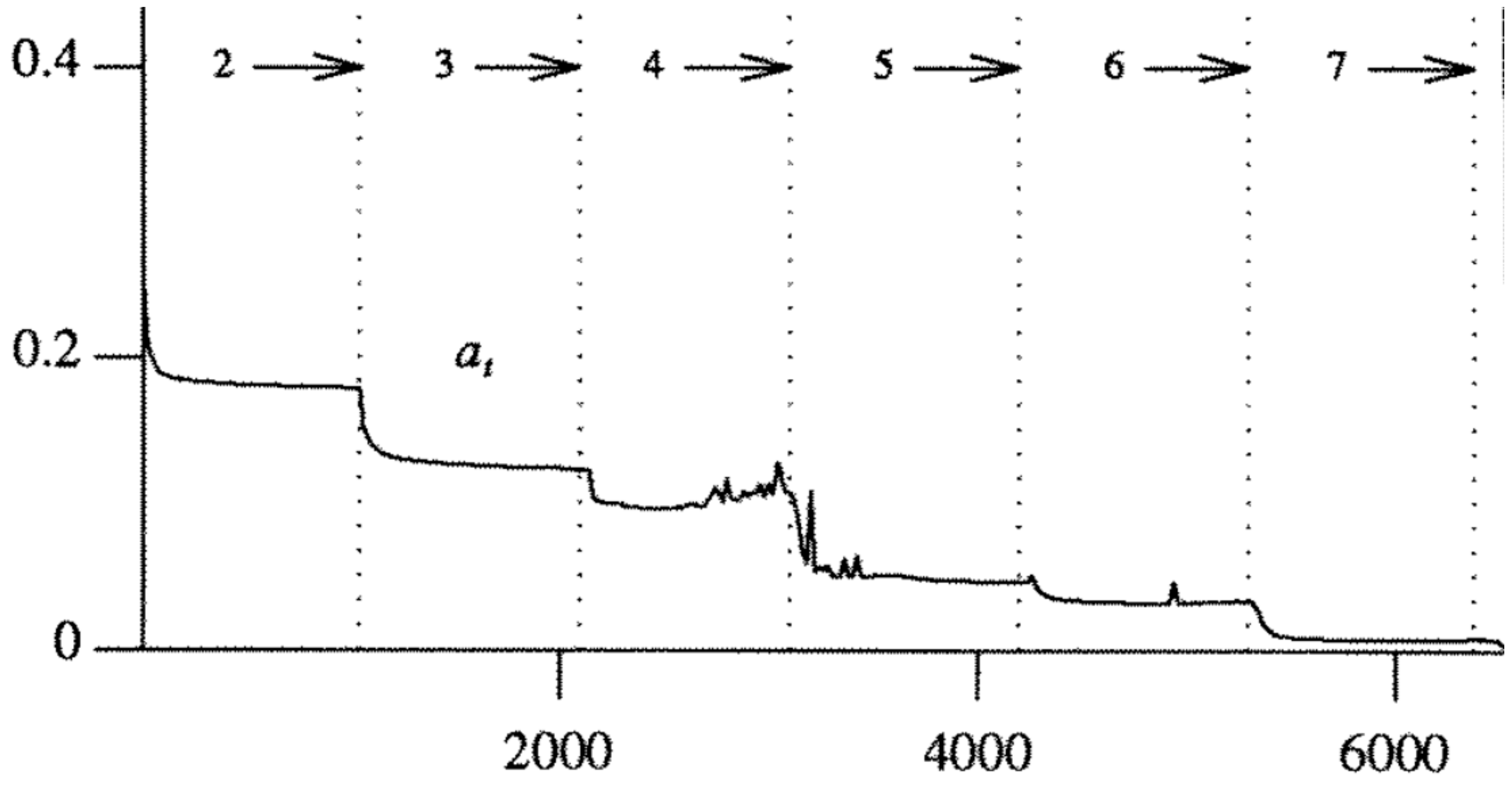
TABLE 2. TEST PROBLEMS ALONG WITH EMPIRICAL UPPER BOUNDS ON THE NUMBER OF HIDDEN LAYER UNITS

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

Inspiration from neurogenesis: dynamic node creation

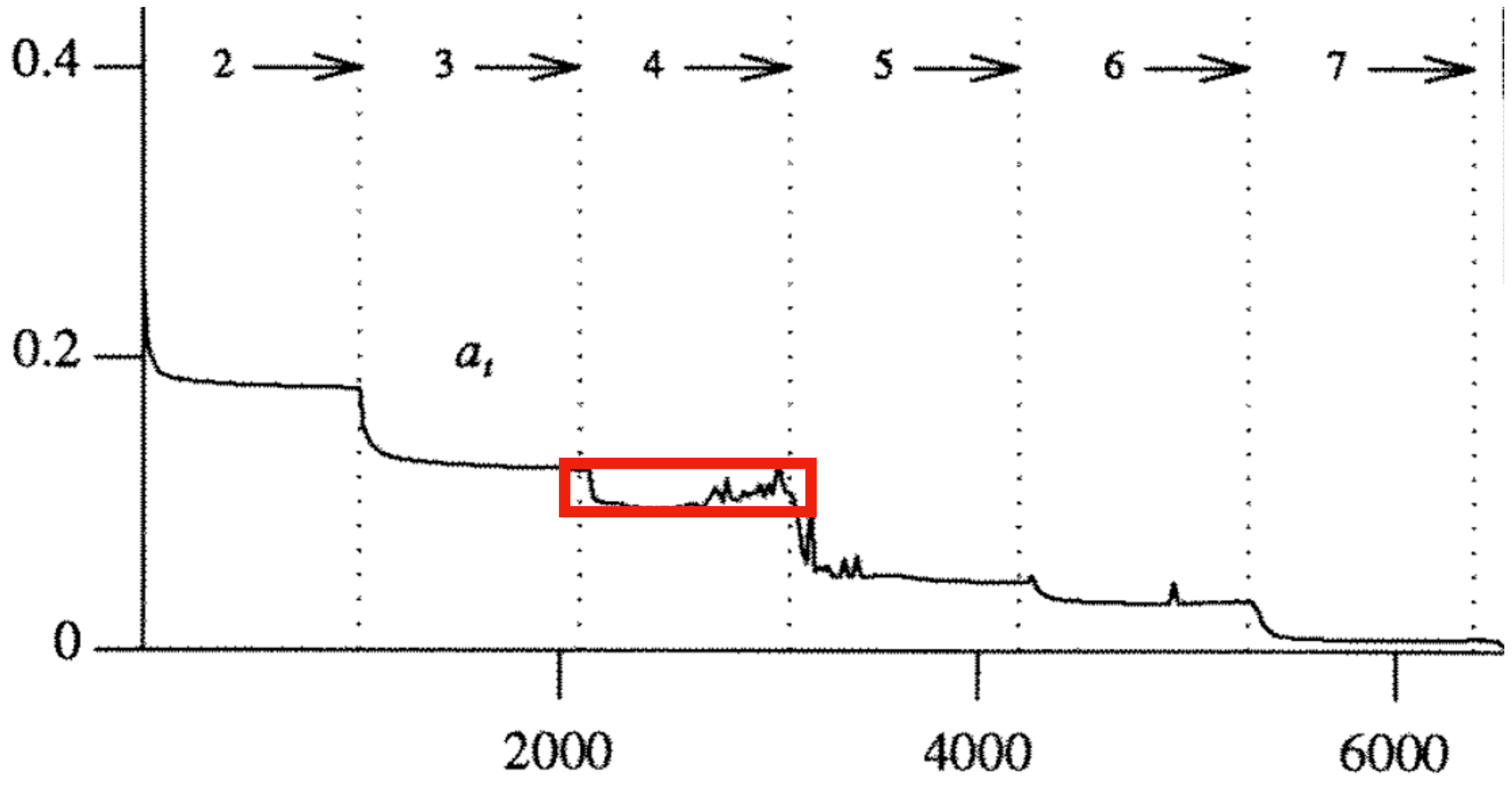
Squared error (y axis) for the ADD3 problem



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

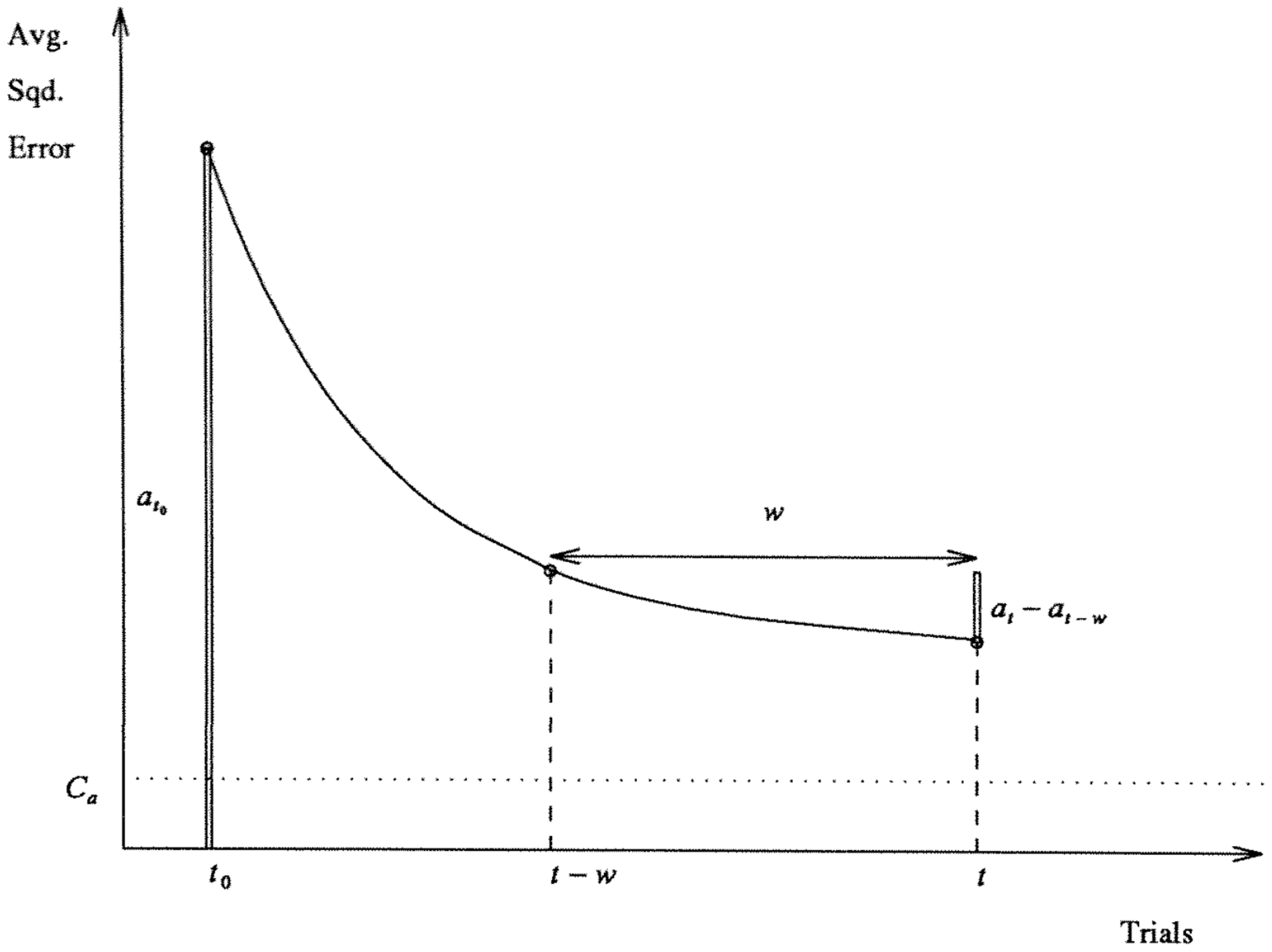
Inspiration from neurogenesis: dynamic node creation

Squared error (y axis) for the ADD3 problem



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

Inspiration from neurogenesis: dynamic node creation

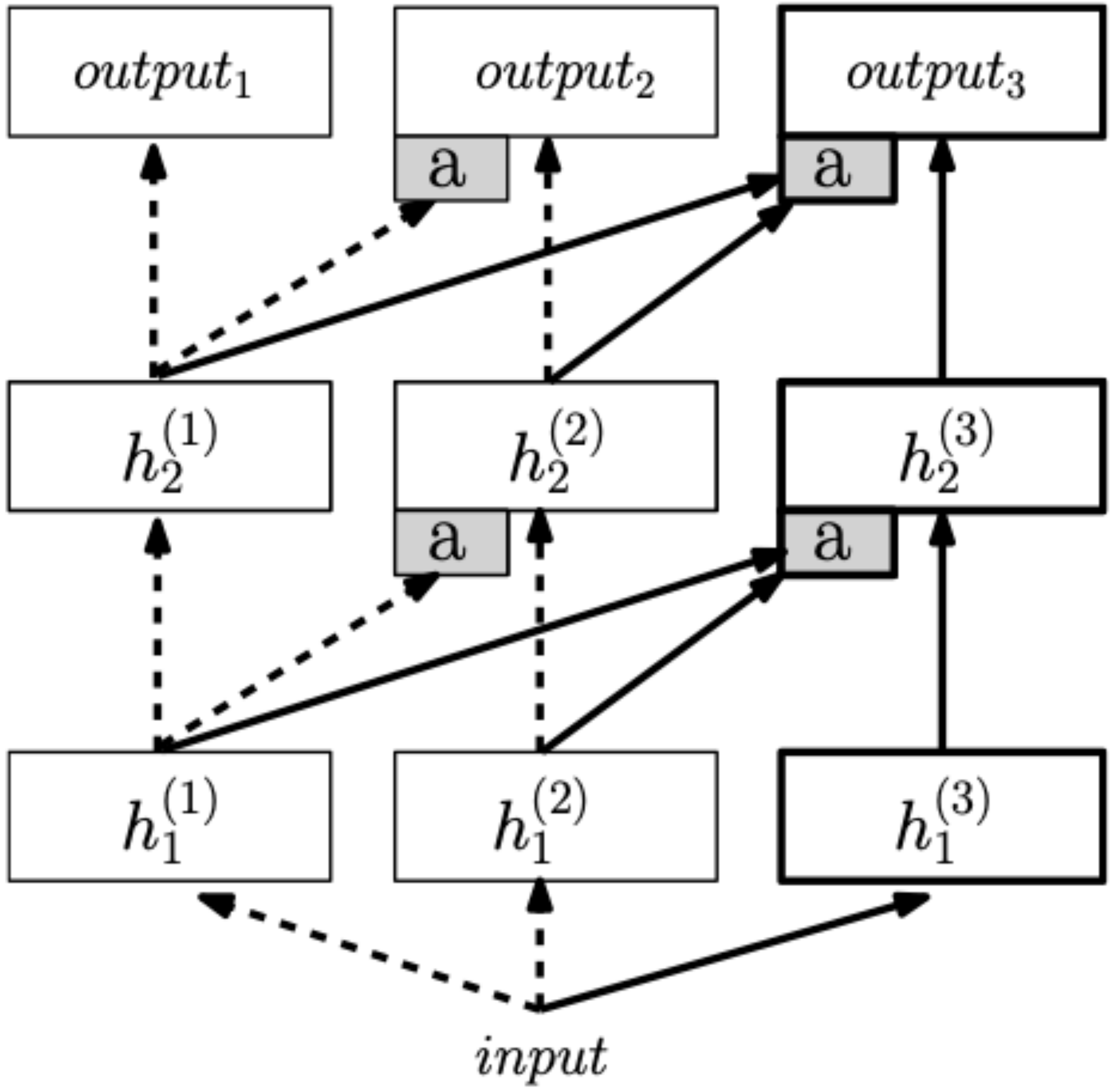


T. Ash, "Dynamic Node Creation in Backpropagation Networks", Connection Science 1:4, 1989

Technically, **3rd crucial question** (not taken into account here): **what/how do we add?**

- one parameter or many?
- neural network layers?
- a different output head if our tasks are different?

A newer example: progressive networks



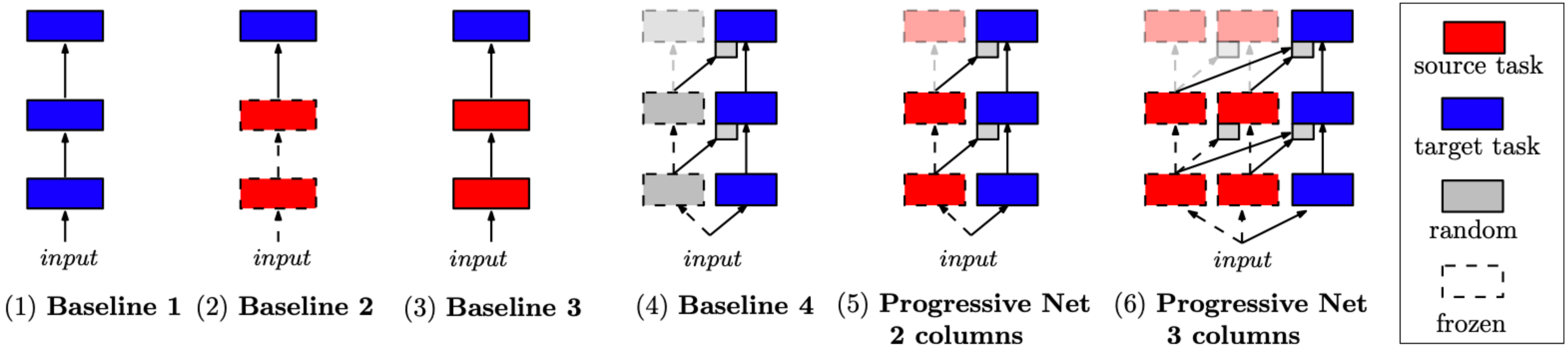
Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

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

➡Transfer where possible & avoid forgetting

A newer example: progressive networks

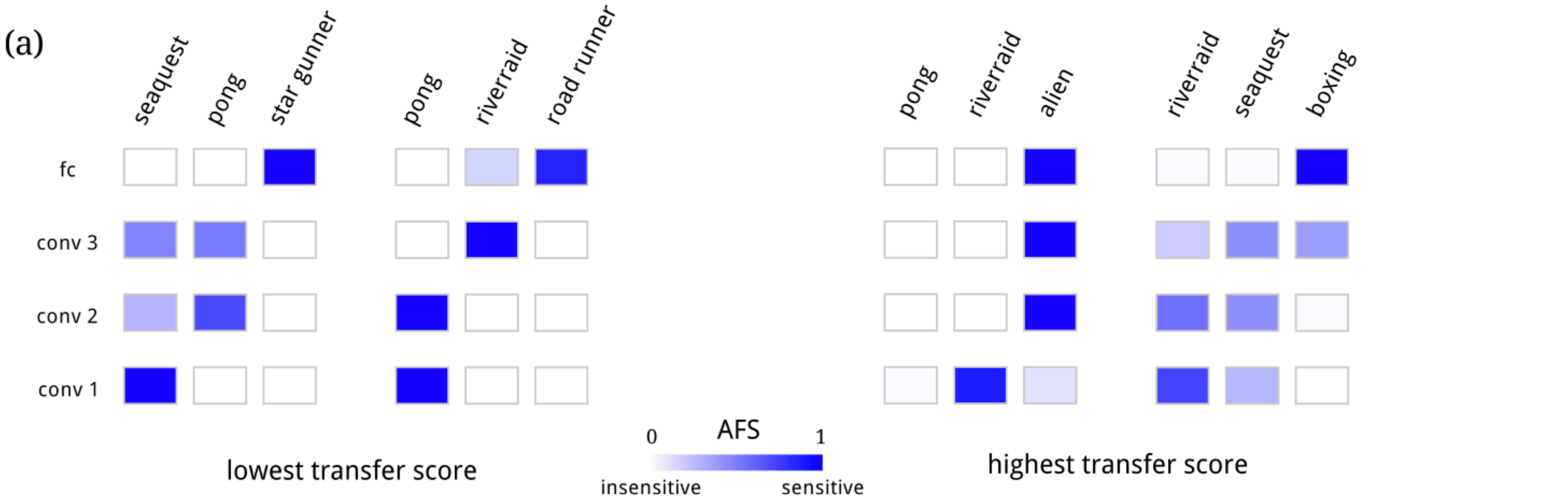
We can evaluate and analyze similarly to what we have already seen, when we talked about knowledge transfer



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

A newer example: progressive networks

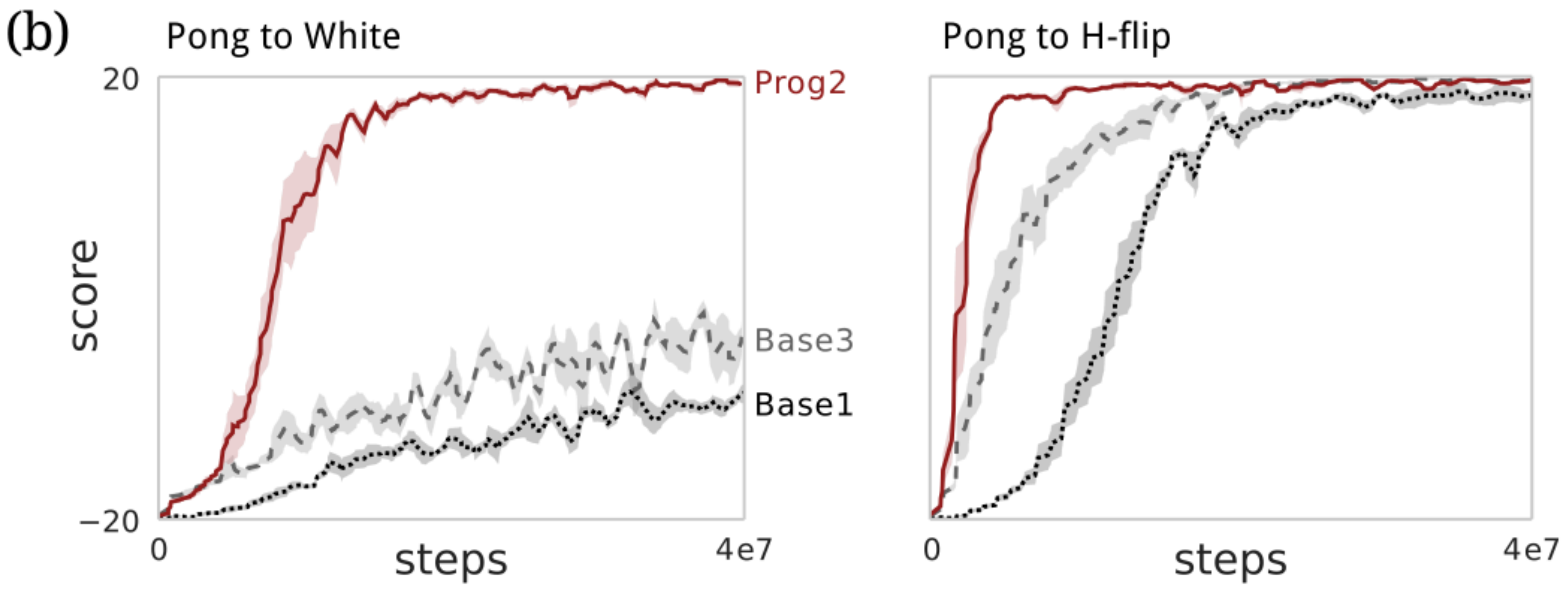
We can evaluate and analyze similarly to what we have already seen, when we talked about knowledge transfer



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

A newer example: progressive networks

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



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

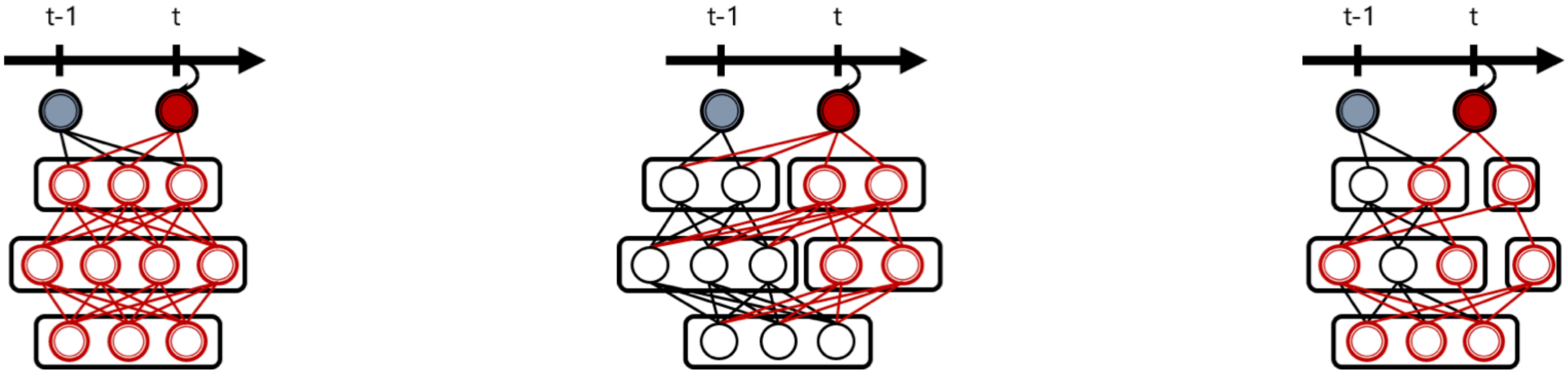
And finally many many more ways that combine ideas: e.g. Dynamically Expandable Nets

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

Intermediate summary: three perspectives to avoid forgetting & a massive elephant in the room

General ways to alleviate forgetting?

Regularize important parameters:
Identify relevant parameters for a task & make sure they do not change much, or make sure the input output relationship remains the same

Rehearsal:
Store a subset of data to rehearse or make use of a generative model to generate

Modify the architecture:
Use task specific masks in an overparameterized model or grow/expand

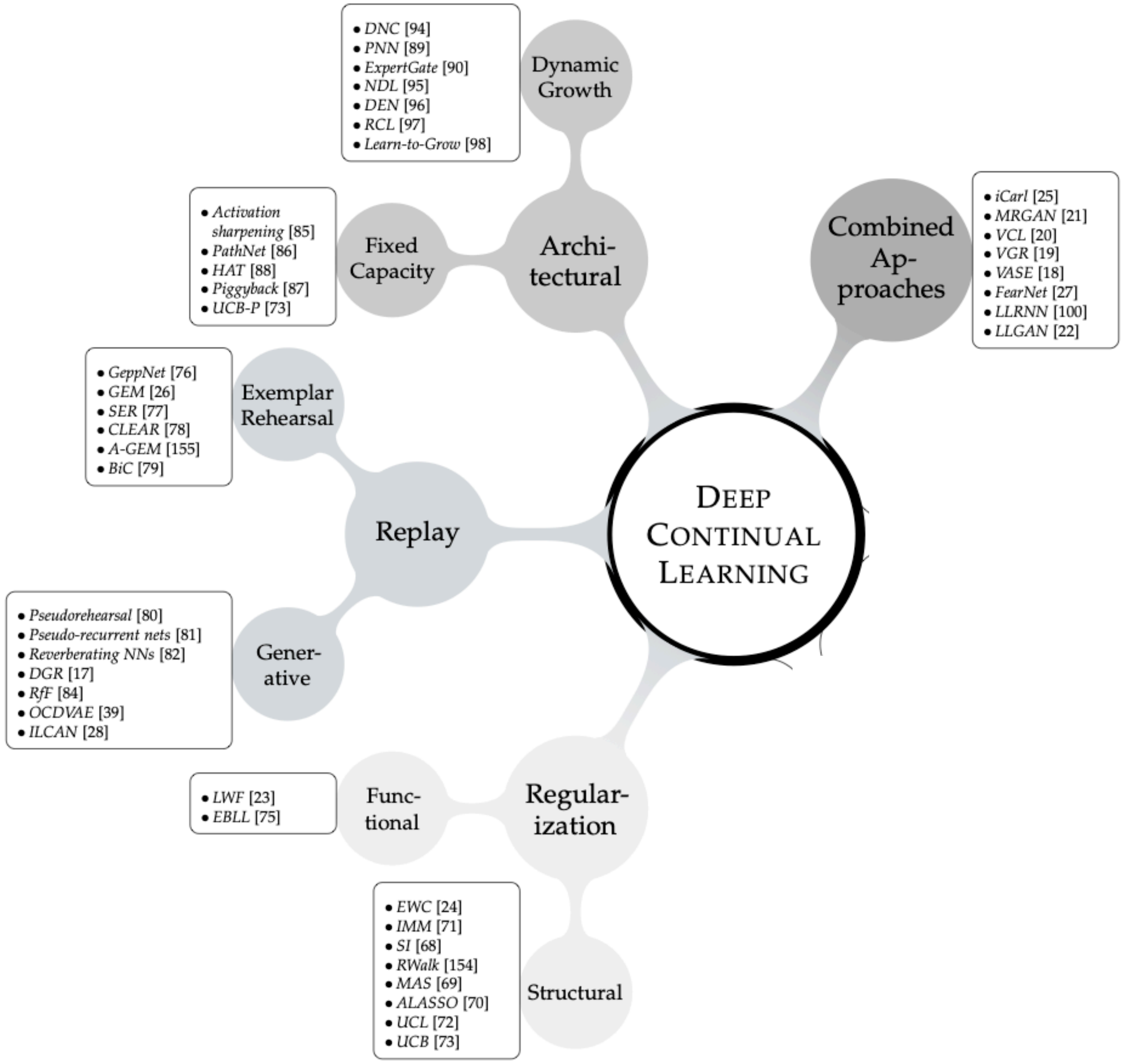
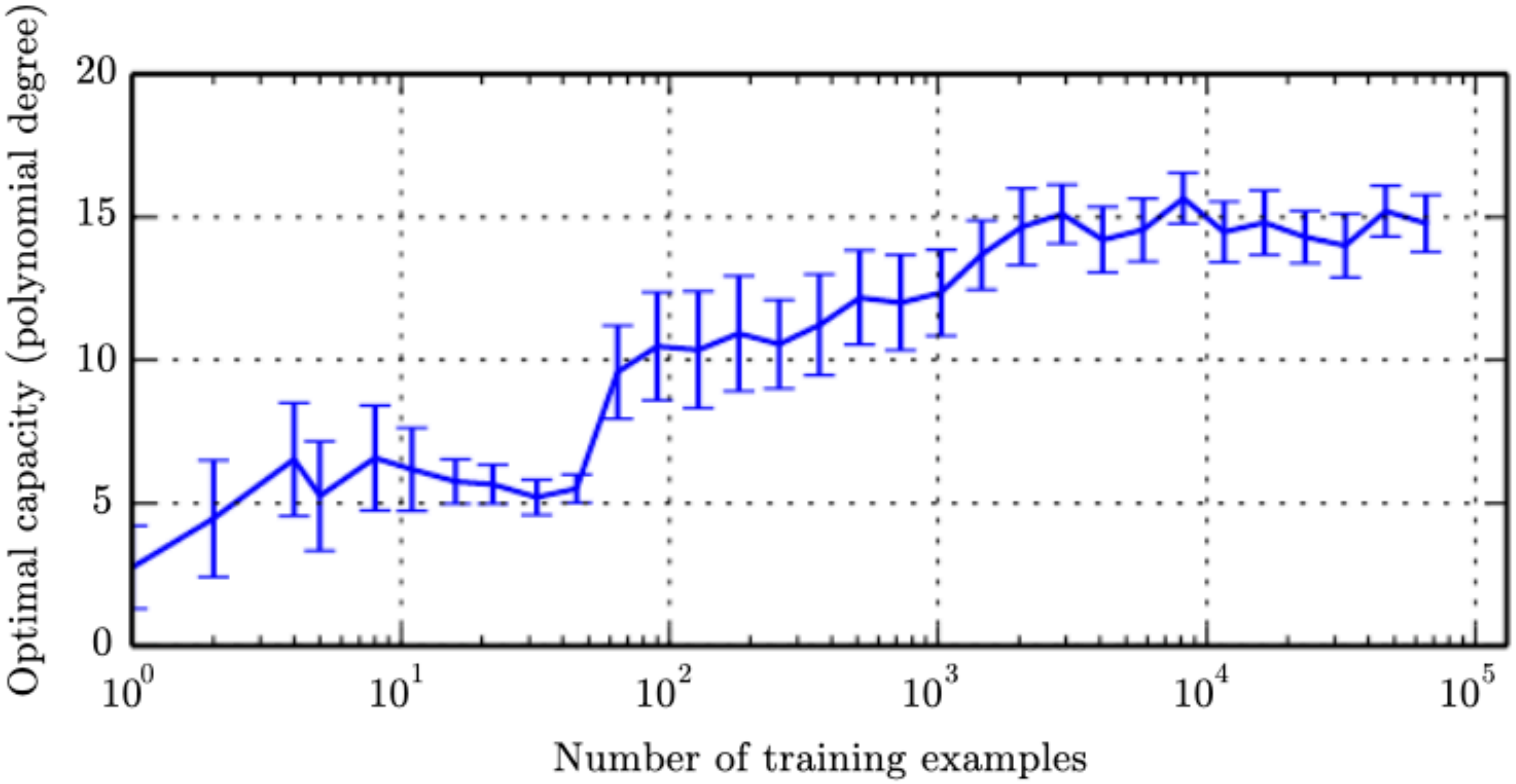


Figure from "A Wholistic View of Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning", Mundt et al, Neural Networks 2023 (Categorization found in several reviews & 30 years ago already)

What is the elephant in the room?

Recall the first lecture's machine learning intro

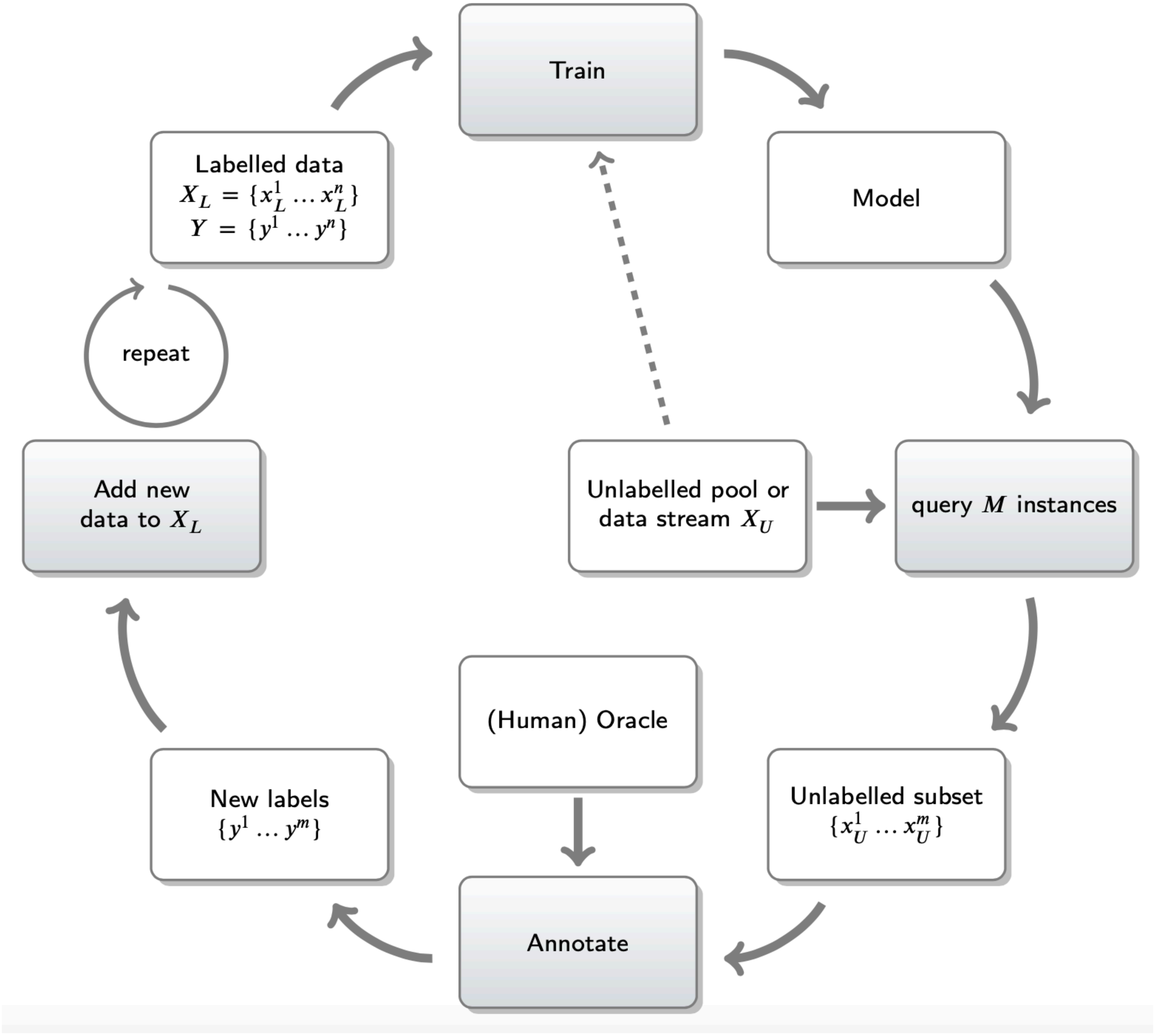
It's not just about forgetting: it's generally about finding suitable capacity



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

Growing models is about finding the right capacity, but also about the ability to handle <u>future data</u>!

Where does (future) data come from? Active learning

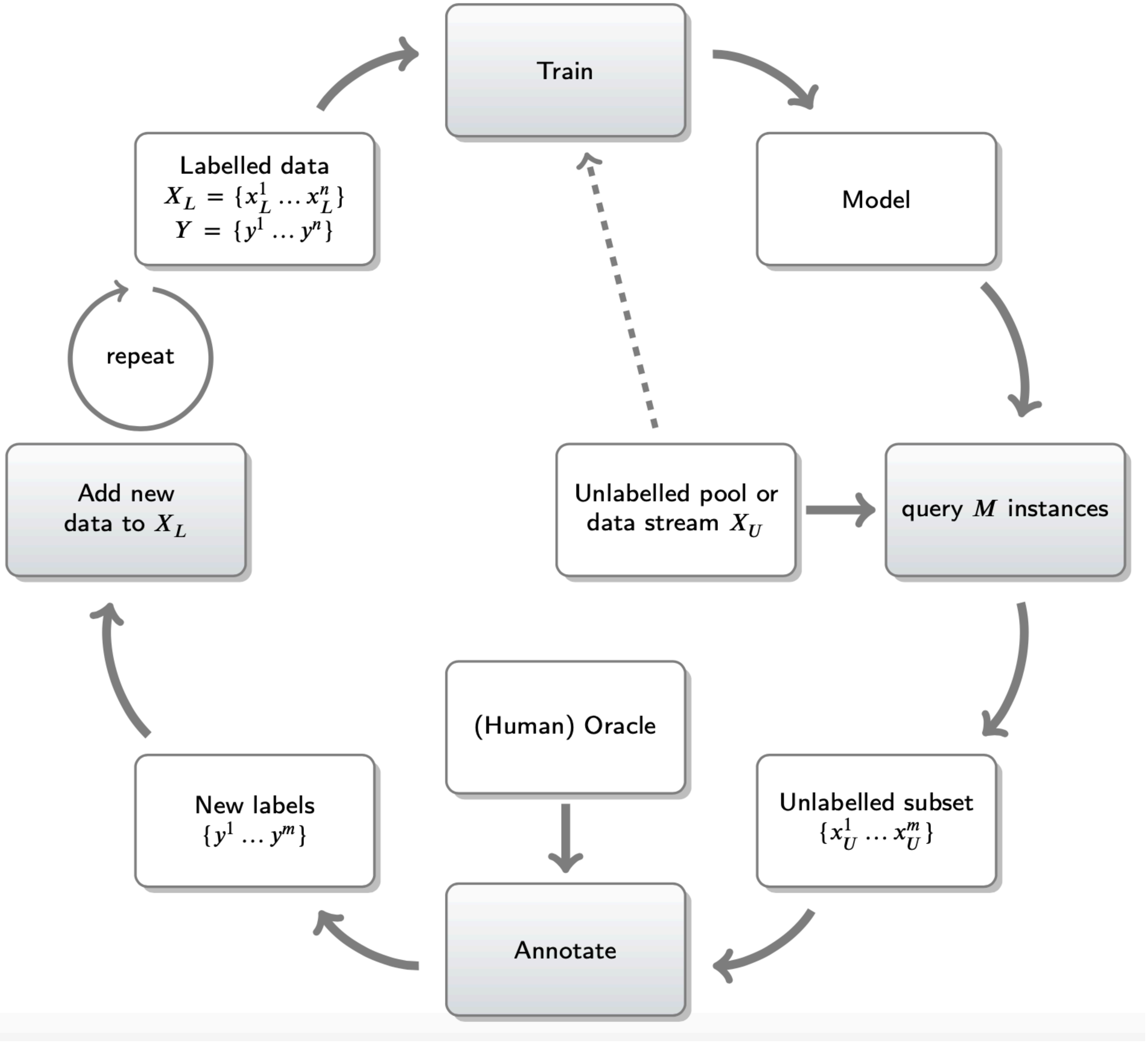


In essence:

How to pick data to add over time?

Figure from Mundt et al, "A Wholistic View of Continual Learning with Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning, Neural Networks 160, 2023

Where does (future) data come from? Active learning



In essence:

How to pick data to add over time?

Before we go through the details:

let's assume we have some way to filter new data & answer how model growth is related to this

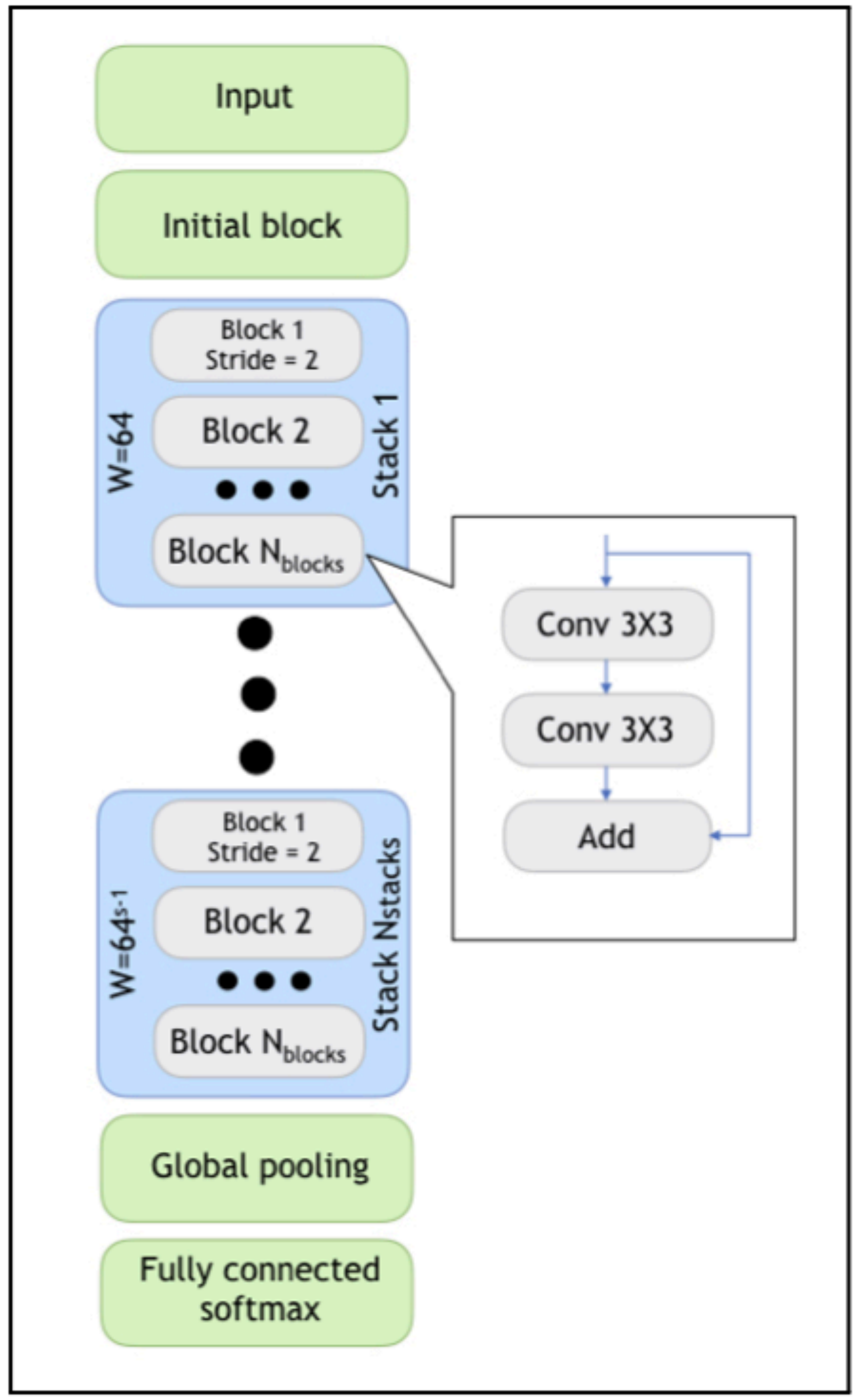
Figure from Mundt et al, "A Wholistic View of Continual Learning with Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning, Neural Networks 160, 2023

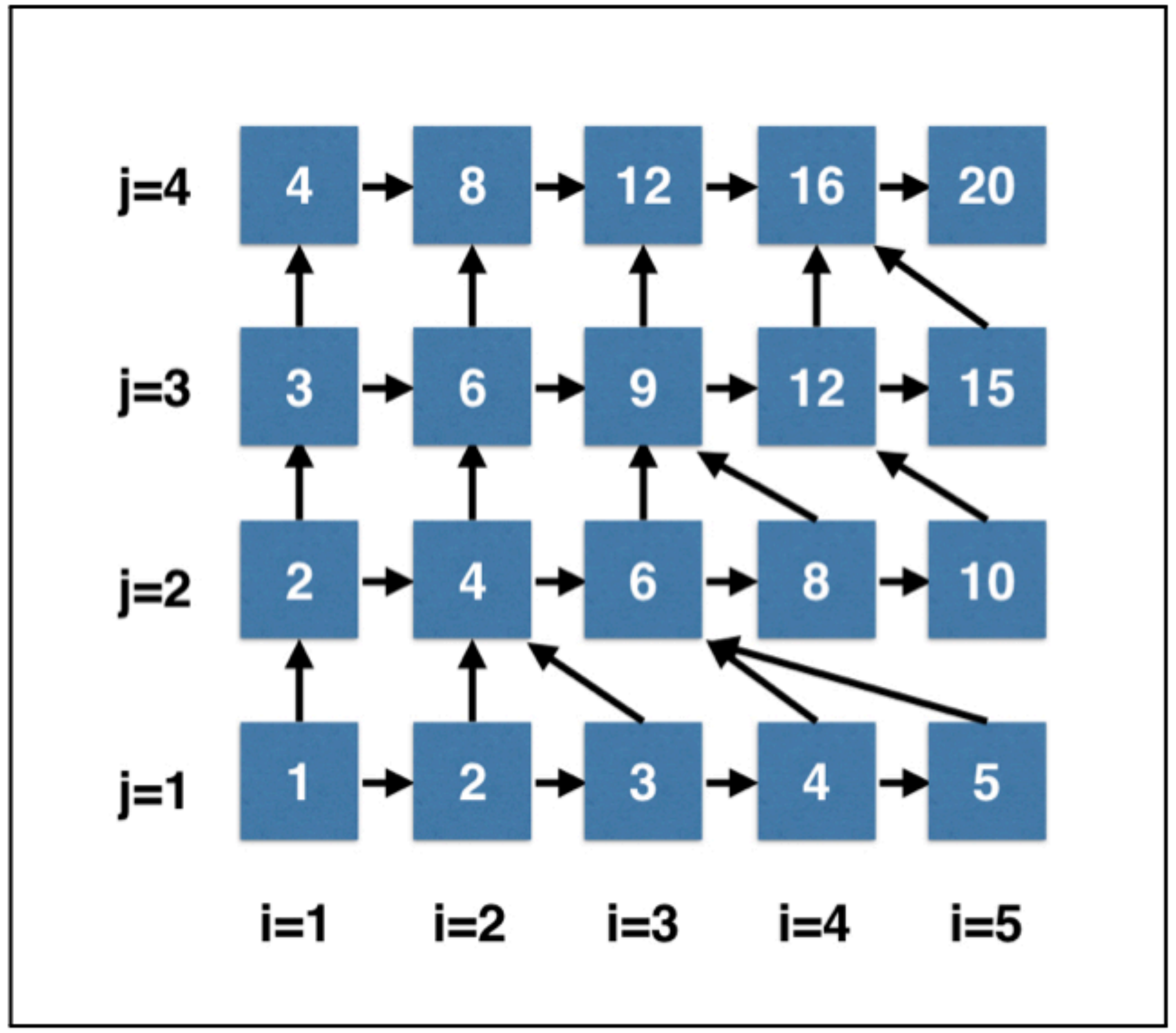
From past to present to future: a growing model with active data queries example

Incremental architecture: For every new data batch, evaluate three architecture choices

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

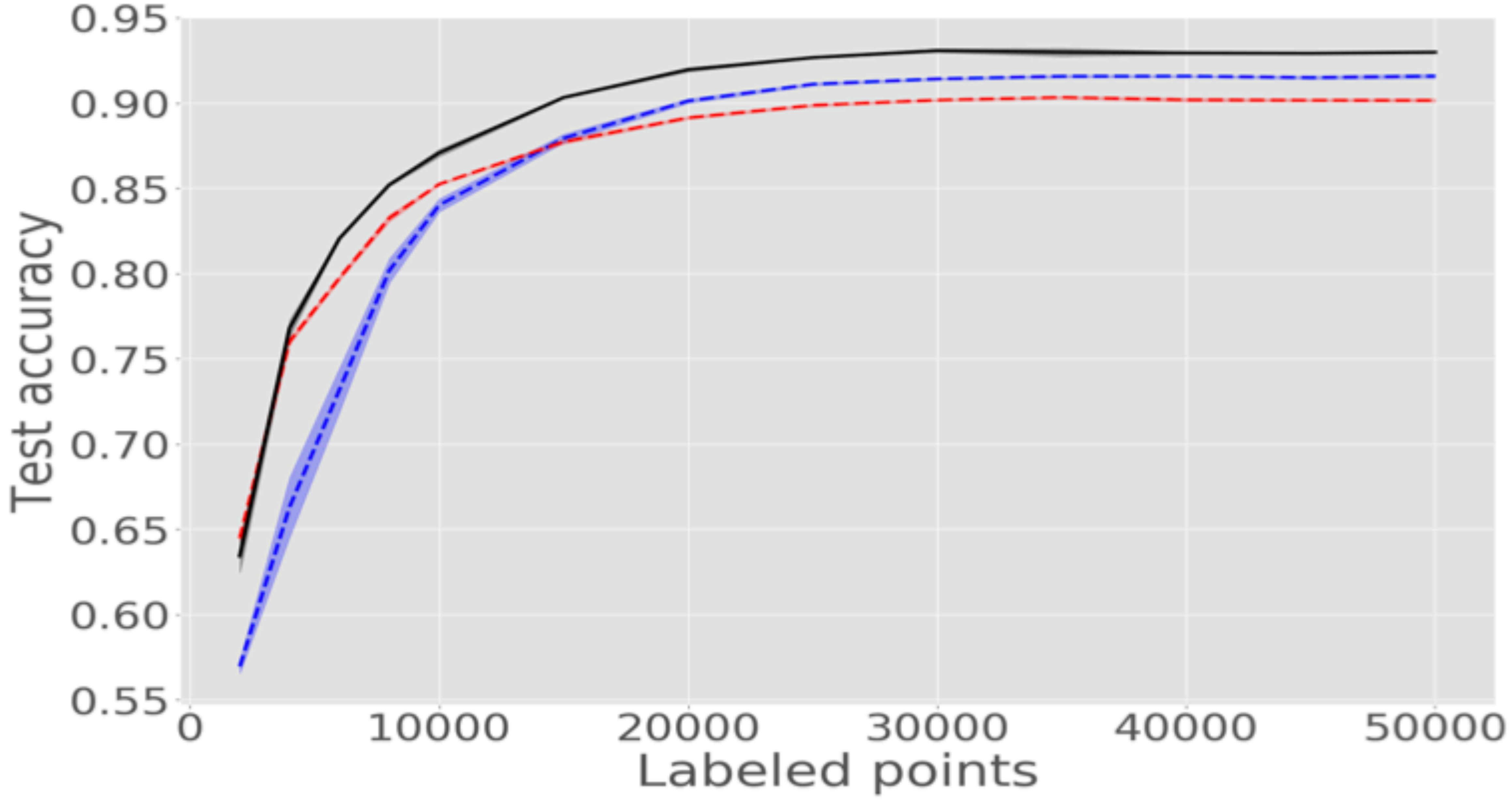
Greedily select the best candidate in terms of a validation dataset





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

From past to present to future: a growing model with active data queries example

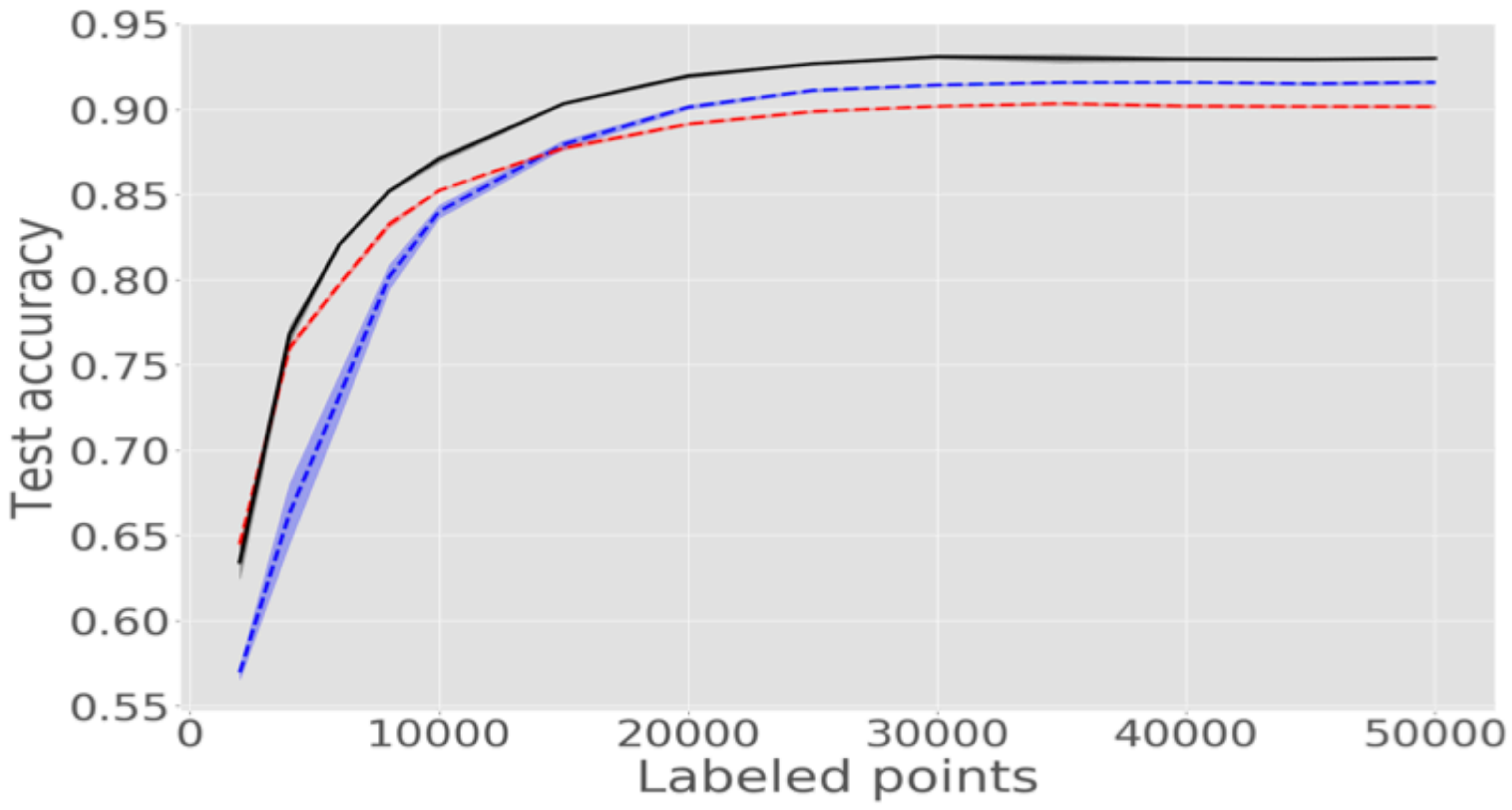


(a) Softmax response

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

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

From past to present to future: a growing model with active data queries example



(a) Softmax response

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

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

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

From past to present to future: a growing model with active data queries example

Consistent for different ways to actively pick data

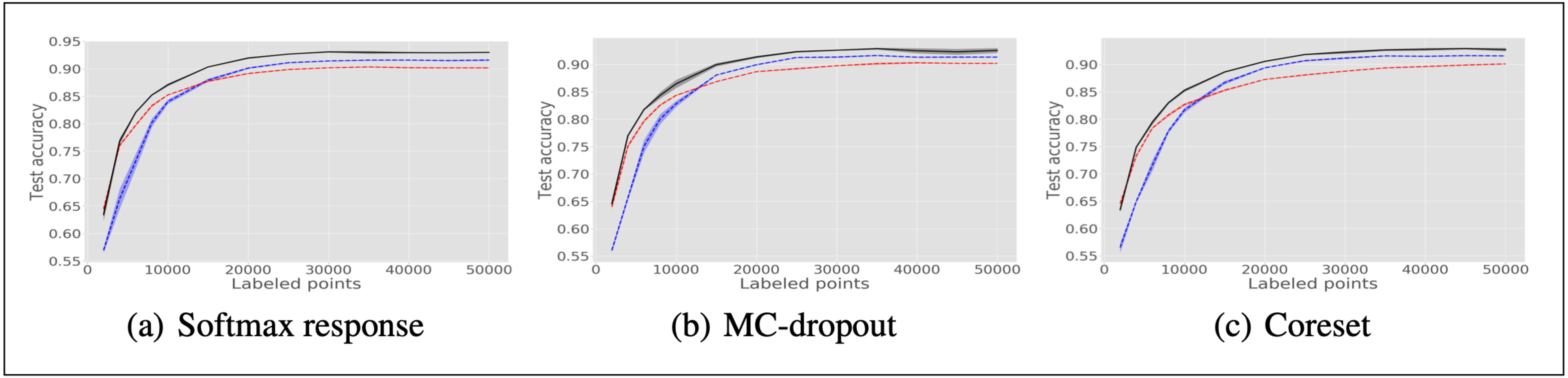


Figure 2: Active learning curves for CIFAR-10 dataset using various query functions, (a) softmax response, (b) MC-dopout, (c) coreset. In black (solid) – Active-iNAS (ours), blue (dashed) – Resnet-18 fixed architecture, and red (dashed) – $A(B_r, 1, 2)$ fixed.

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

Now that we have realized that model growth is about past & future, let's dive into data selection mechanisms