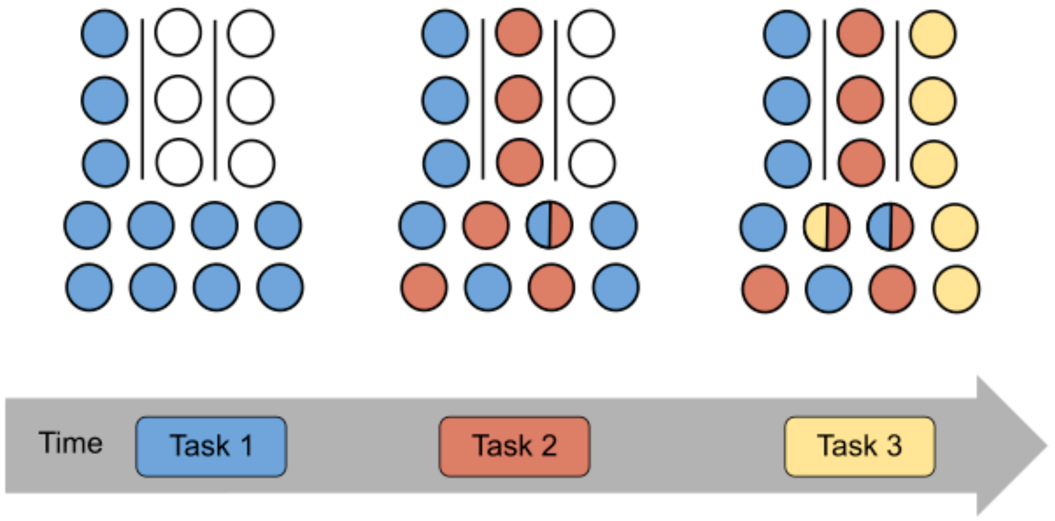
Universität
Bremen



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

Part 2 cont. - Retaining the Past

Experts, Gating & Dynamic Architectures

Lifelong Machine Learning
Summer 2025

Prof. Dr. Martin Mundt

CURL is somewhat implicitly modular

What do we mean by “implicitly modular”?

- It allocates separate Gaussians
- But it relies on a shared encoder/decoder architecture
- Intuitively, we create task-specific sub-modules & sub-spaces
- Requires a significantly over-parametrized model

CURL is somewhat implicitly modular

What do we mean by “implicitly modular”?

- It allocates separate Gaussians
- But it relies on a shared encoder/decoder architecture
- Intuitively, we create task-specific sub-modules & sub-spaces
- Requires a significantly over-parametrized model

Such an **“implicit” approach** is popular in continual learning because it is **“easier”** to get to work.

The alternative would be to **add actual parameters/capacity over time as required** - which is hard because we need to understand when to do this and how to do this in practice. We will look at this in more depth later, but first let us look at different ways for the “implicit” approach.

Universität Bremen

Question time

What are ways in which we could induce task-specificity in large neural networks?

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **Parameters**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **Parameters**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters
- **Activations**: inspired by the motivation behind SI, we could separate out activations for different tasks

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **Parameters**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters
- **Activations**: inspired by the motivation behind SI, we could separate out activations for different tasks
- **Gating/Attention**: we could not modify the architecture at all, and include external gates or “attention” computations for separate tasks

(Some) implicit CL architecture perspectives

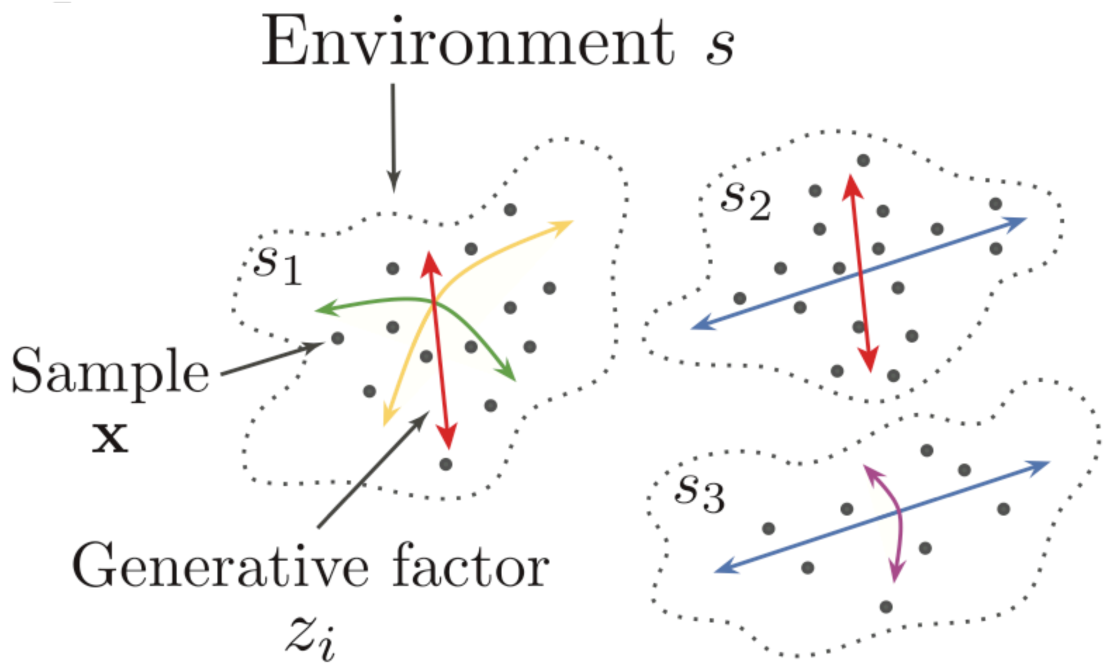
- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **Parameters**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters
- **Activations**: inspired by the motivation behind SI, we could separate out activations for different tasks
- **Gating/Attention**: we could not modify the architecture at all, and include external gates or "attention" computations for separate tasks

Let us look at one example algorithm for each of these categories
(Note: there are numerous nuanced works for each of these approaches)

Universität Bremen

Spaces: VAE with shared embeddings

- Use a VAE and share encoder, but in contrast to CURL assume a priori unknown set S of unknown environments
- For piece-wise stationary data use different sub-sets of generative factors z (which can be the same, but rendered differently)



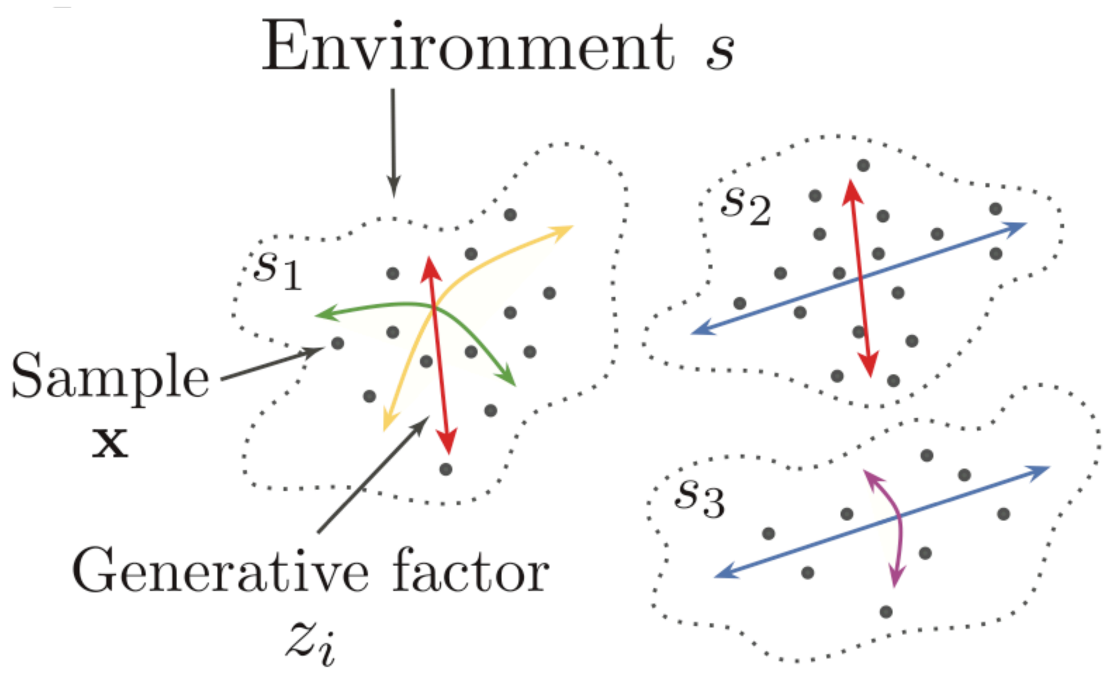
Achille et al, "Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies", NeurIPS 2018

Universität Bremen

Spaces: VAE with shared embeddings

- Intuition: piece-wise stationary observed data can be split into clusters, where each cluster can be mapped to coordinate axes used to parametrize the data

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



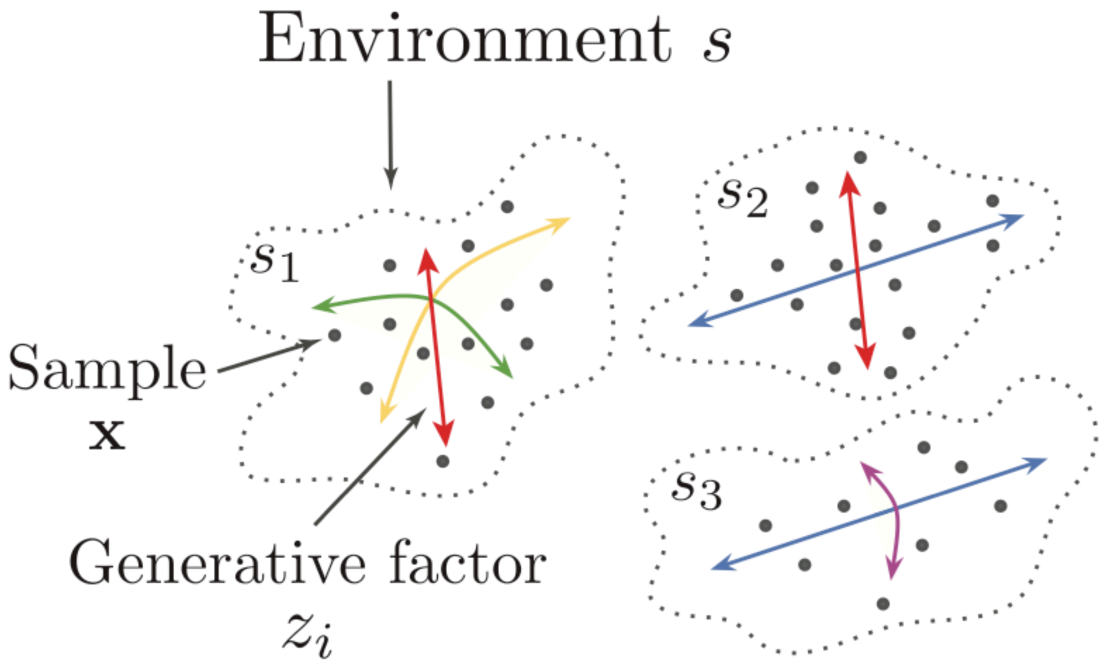
Achille et al, “Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies”, NeurIPS 2018

Spaces: VAE with shared embeddings

- Intuition: piece-wise stationary observed data can be split into clusters, where each cluster can be mapped to coordinate axes used to parametrize the data

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

- To "disentangle" the generative factors, introduce explicit target capacity C, that is progressively increased throughout learning



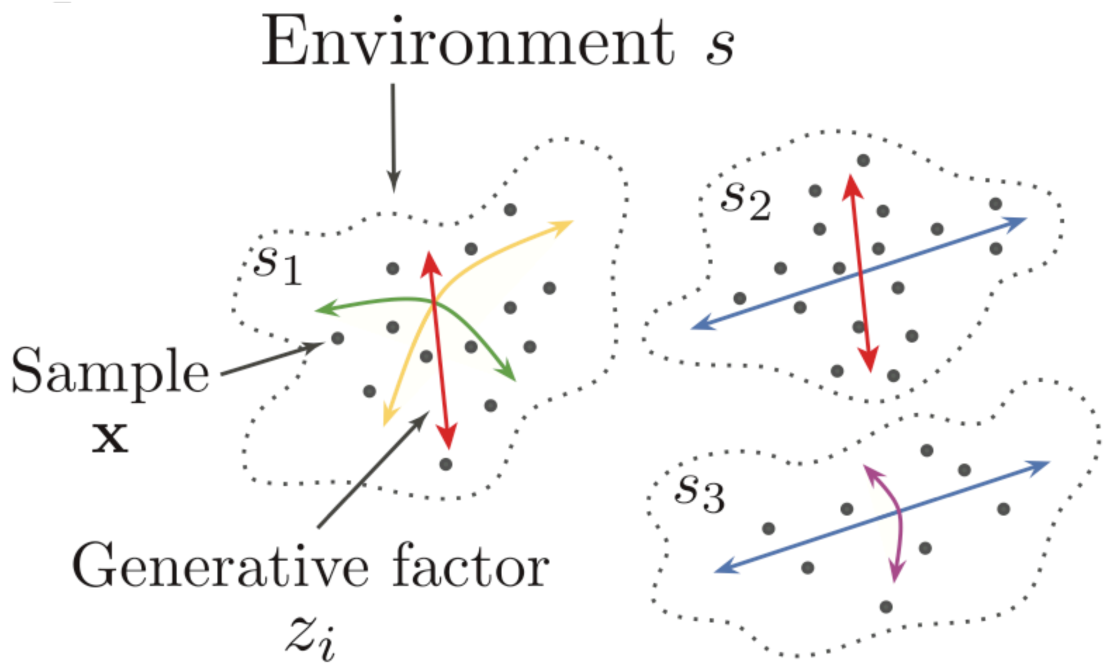
Achille et al, "Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies", NeurIPS 2018

Universität Bremen

Spaces: VAE with shared embeddings

- For prediction, we will however require to keep track which subset is used introducing an environment latent mask a^s:

$$q(z^s \mid x^s) = a^s \circ \mathcal{N}(\mu(x), \sigma(x)) + (1 - a^s) \circ \mathcal{N}(0, \mathbb{I})$$



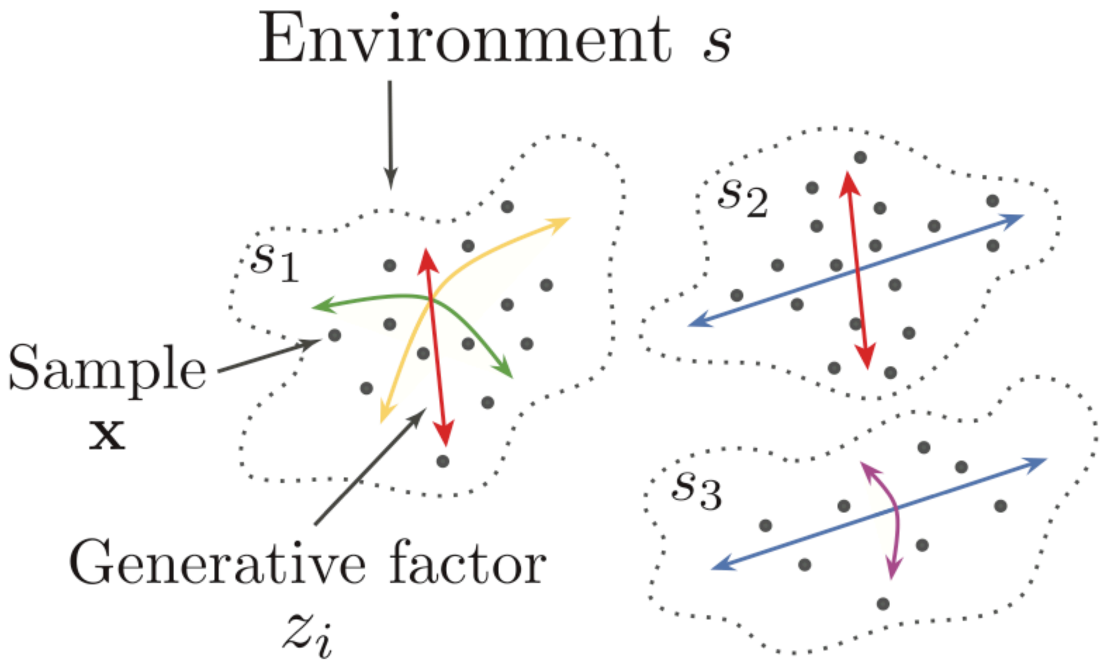
Achille et al, “Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies”, NeurIPS 2018

Universität Bremen

Spaces: VAE with shared embeddings

- For prediction, we will however require to keep track which subset is used introducing an environment latent mask a^s:
  $q(z^s \mid x^s) = a^s \circ \mathcal{N}(\mu(x), \sigma(x)) + (1 - a^s) \circ \mathcal{N}(0, \mathbb{I})$

- We infer the latent mask by testing if the latents stray away from the prior, i.e. behave "atypically" to the environment according to some set threshold:
  $\alpha = \mathbb{KL}(\mathbb{E}_{x^s}[q(z^s \mid x^s)] \,||\, p(z))$

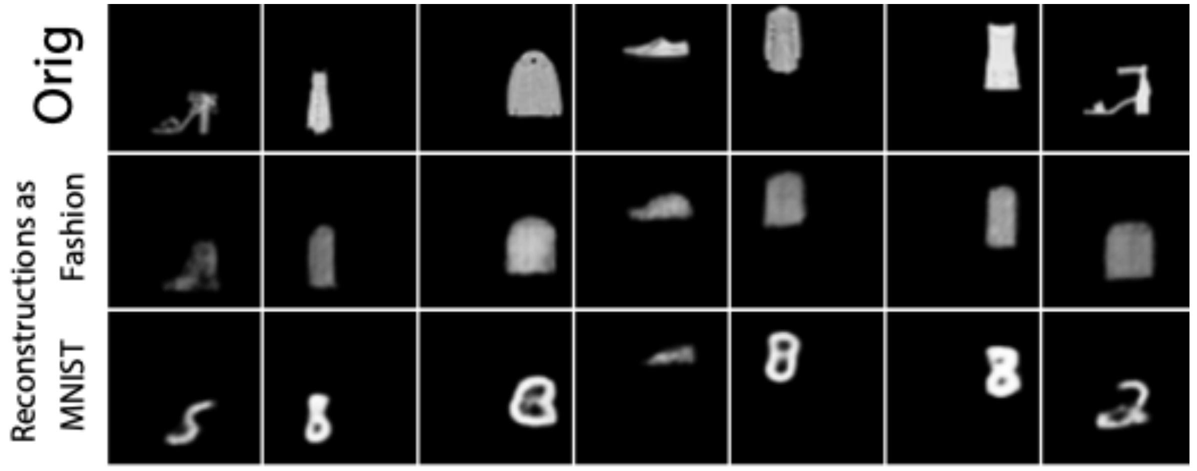


Achille et al, "Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies", NeurIPS 2018

Universität Bremen

Spaces: VAE with shared embeddings

- Summary: learn to associate experiences to an appropriate cluster without disrupting unrelated clusters.
- Changes in the environment are related to increased capacity
- Generative replay is used to avoid forgetting



$$\mathcal{L}(\phi,\theta) = \underbrace{\mathbb{E}_{\mathbf{z}^s \sim q_\phi(\cdot|\mathbf{x}^s))}[-\log p_\theta(\mathbf{x}|\mathbf{z}^s,s)] + \gamma|\mathbb{KL}(q_\phi(\mathbf{z}^s|\mathbf{x}^s)||p(\mathbf{z})) - C|^2}_{\text{MDL on current data}} +$$

$$+ \underbrace{\mathbb{E}_{\mathbf{z},s',\mathbf{x}'}\Big[D[q_\phi(\mathbf{z}|\mathbf{x}'), q_{\phi'}(\mathbf{z}'|\mathbf{x}')] + D[q_\theta(\mathbf{x}|\mathbf{z},s'), q_{\theta'}(\mathbf{x}'|\mathbf{z},s')] \Big]}_{\text{“Dreaming” feedback on past data}}.$$

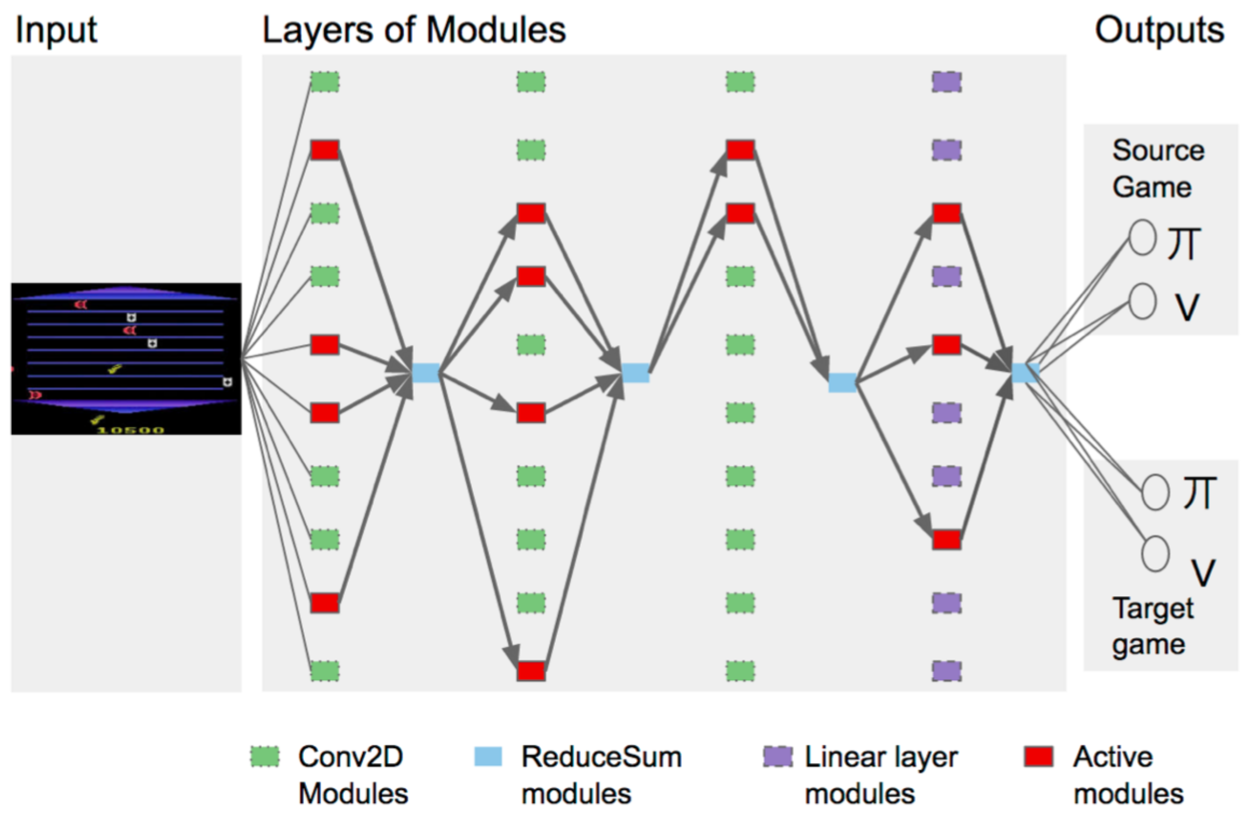
Achille et al, “Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies”, NeurIPS 2018

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **<u>Parameters</u>**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters
- **Activations**: inspired by the motivation behind SI, we could separate out activations for different tasks
- **Gating/Attention**: we could not modify the architecture at all, and include external gates or “attention” computations for separate tasks

Let us look at one example algorithm for each of these categories
(Note: there are numerous nuanced works for each of these approaches)

Parameters: Constraining "Paths" in PathNet

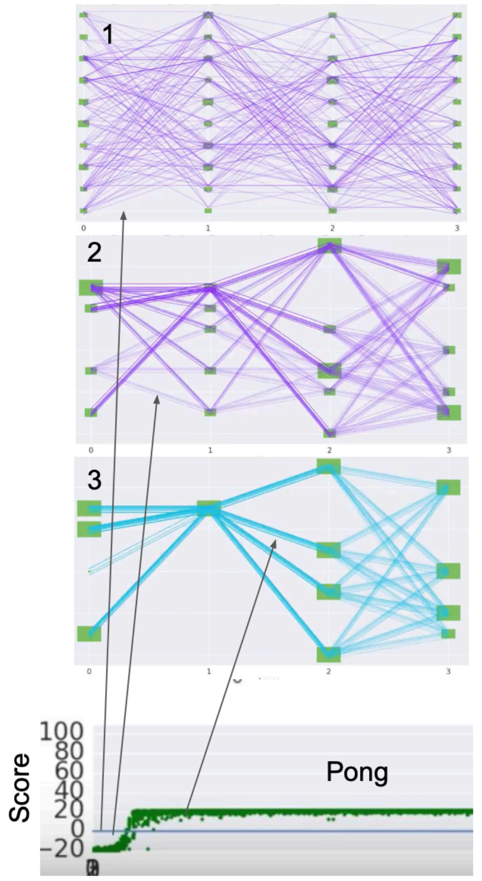


Simple idea: constrain a task to use a subset of parameters, before then freezing them

Intuitively: enforce a small number of active "modules" or "paths" by removing connectivity beyond some pre-set value

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

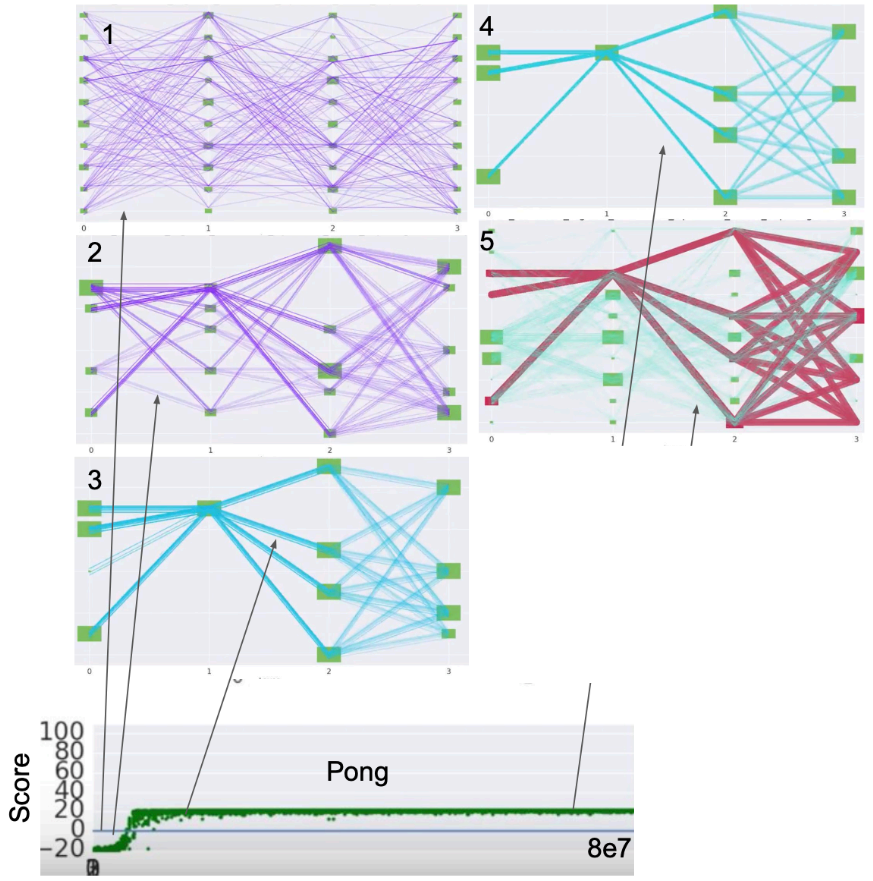
Parameters: Constraining “Paths” in PathNet



Start with a randomly initialized “pathway” (purple lines) while learning a task (e.g. playing Pong)

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

Parameters: Constraining "Paths" in PathNet



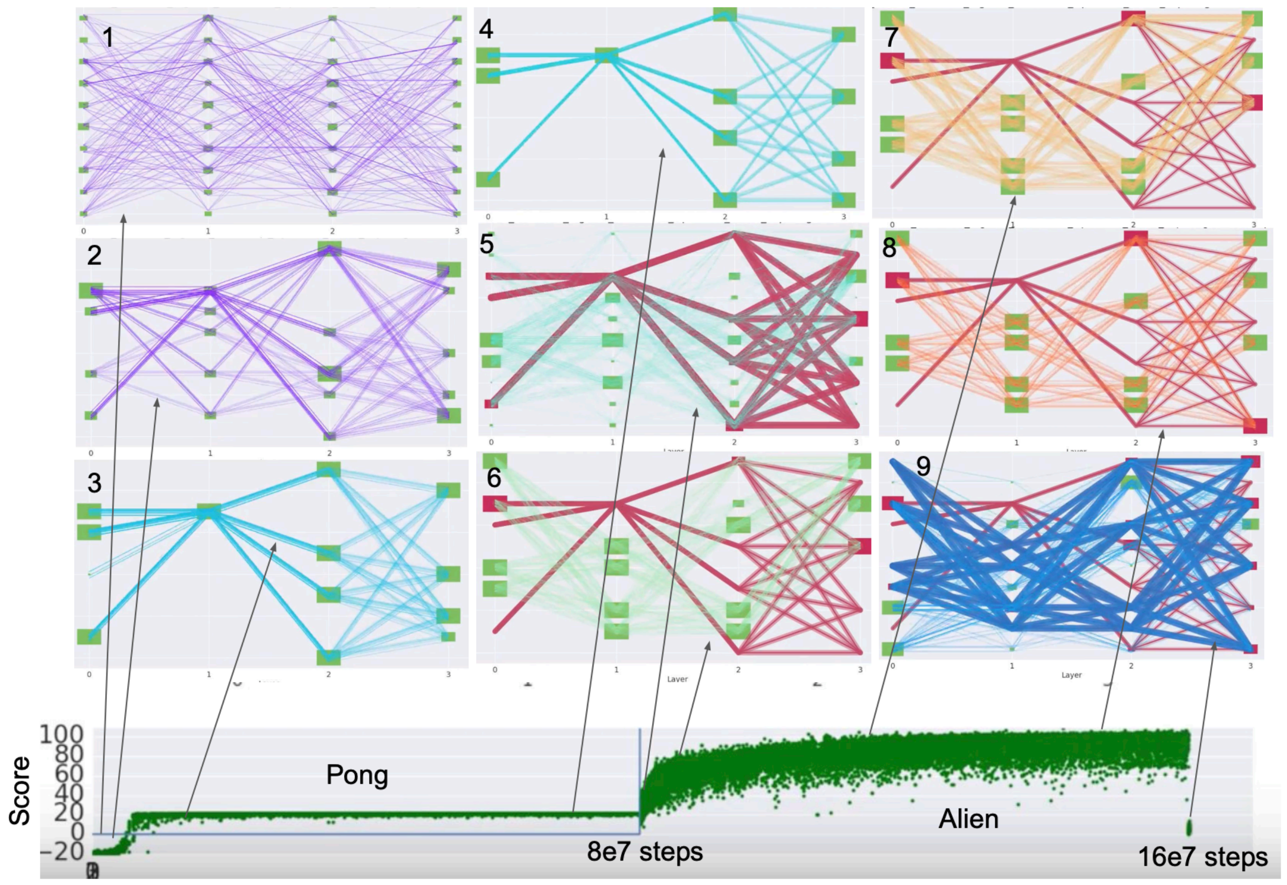
Start with a randomly initialized "pathway" (purple lines) while learning a task (e.g. playing Pong)

Fix the "best" pathway (red) & add a new path population (light blue)

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

Universität Bremen

Parameters: Constraining “Paths” in PathNet



Start with a randomly initialized “pathway” (purple lines) while learning a task (e.g. playing Pong)

Fix the “best” pathway (red) & add a new path population (light blue)

Train new population (here on Alien) until again fixing the pathway at the end (dark blue)

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

Universität
Bremen

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **Parameters**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters
- **Activations**: inspired by the motivation behind SI, we could separate out activations for different tasks
- **Gating/Attention**: we could not modify the architecture at all, and include external gates or “attention” computations for separate tasks

Let us look at one example algorithm for each of these categories
(Note: there are numerous nuanced works for each of these approaches)

Activations: “sharpen” activations (or sparsify)

Intuitively: increase the activation values of some kind of nodes, decrease that of all others

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

Activations: “sharpen” activations (or sparsify)

Intuitively: increase the activation values of some kind of nodes, decrease that of all others

Suggestion: define activation overlap as a sum of the smaller activations (the “shared” activation) as a measure of interference

A four hidden unit example: (0.2, 0.1, 0.9, 0.1) & (0.2, 0.0, 1.0, 0.2)
What is the activation overlap?

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

Activations: “sharpen” activations (or sparsify)

Intuitively: increase the activation values of some kind of nodes, decrease that of all others

Suggestion: define activation overlap as a sum of the smaller activations (the “shared” activation) as a measure of interference

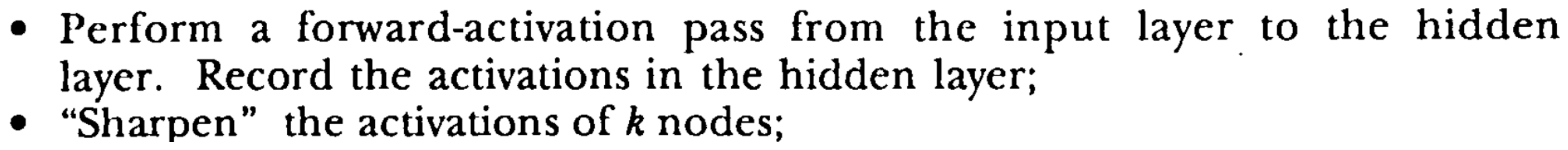
A four hidden unit example: (0.2, 0.1, 0.9, 0.1) & (0.2, 0.0, 1.0, 0.2)
  Activation overlap: (0.2 + 0.0 + 0.9 + 0.1) / 4 = 0.3

A non interfering example: (1, 0, 1, 0) & (0, 1, 0, 1) have no overlap

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

Activations: “sharpen” activations (or sparsify)

Intuitively: increase the activation values of some kind of nodes, decrease that of all others

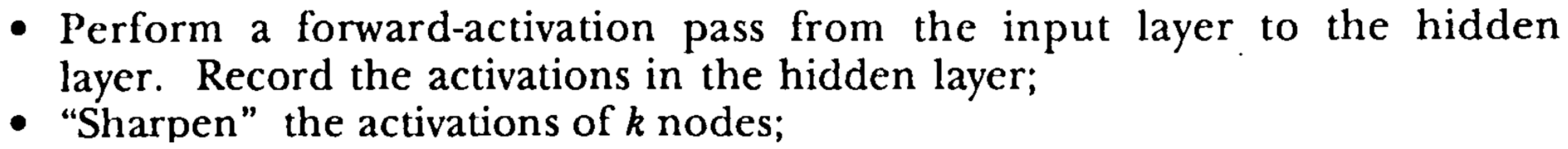


- Perform a forward-activation pass from the input layer to the hidden layer. Record the activations in the hidden layer;
- “Sharpen” the activations of *k* nodes;

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

Activations: “sharpen” activations (or sparsify)

Intuitively: increase the activation values of some kind of nodes, decrease that of all 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;

Sharpen:

$A_{new} = A_{old} + \alpha(1 - A_{old})$ for the nodes to be sharpened

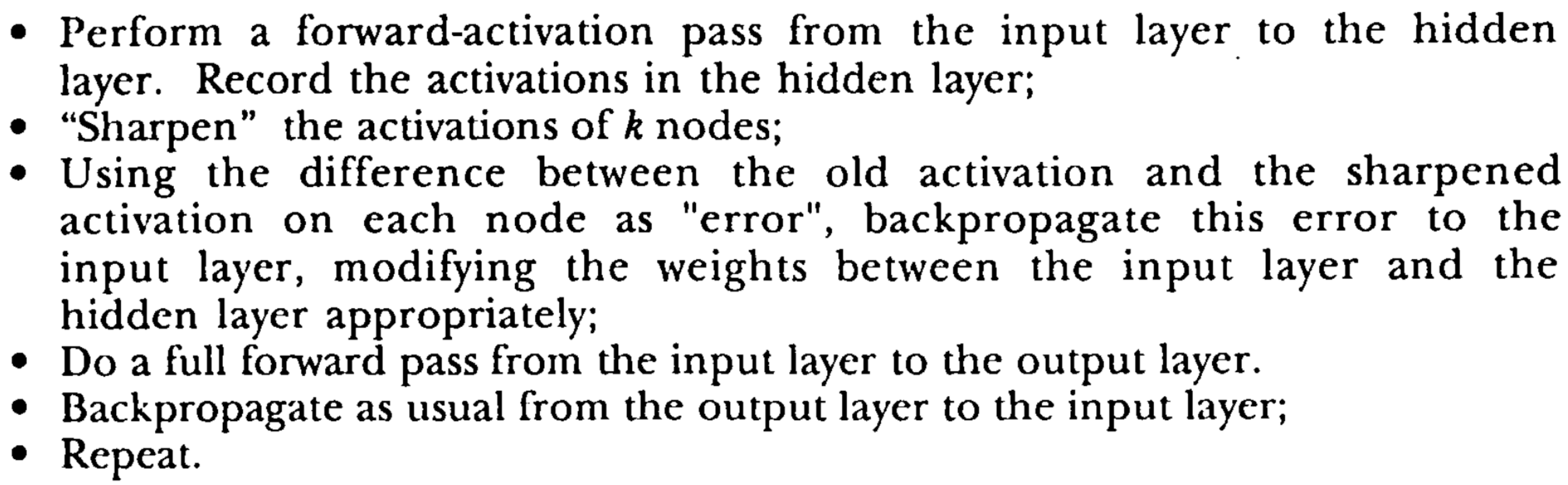
$A_{new} = A_{old} - \alpha A_{old}$ for the other nodes;

according to a chosen sharpening factor α

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

Activations: “sharpen” activations (or sparsify)

Intuitively: increase the activation values of some kind of nodes, decrease that of all others

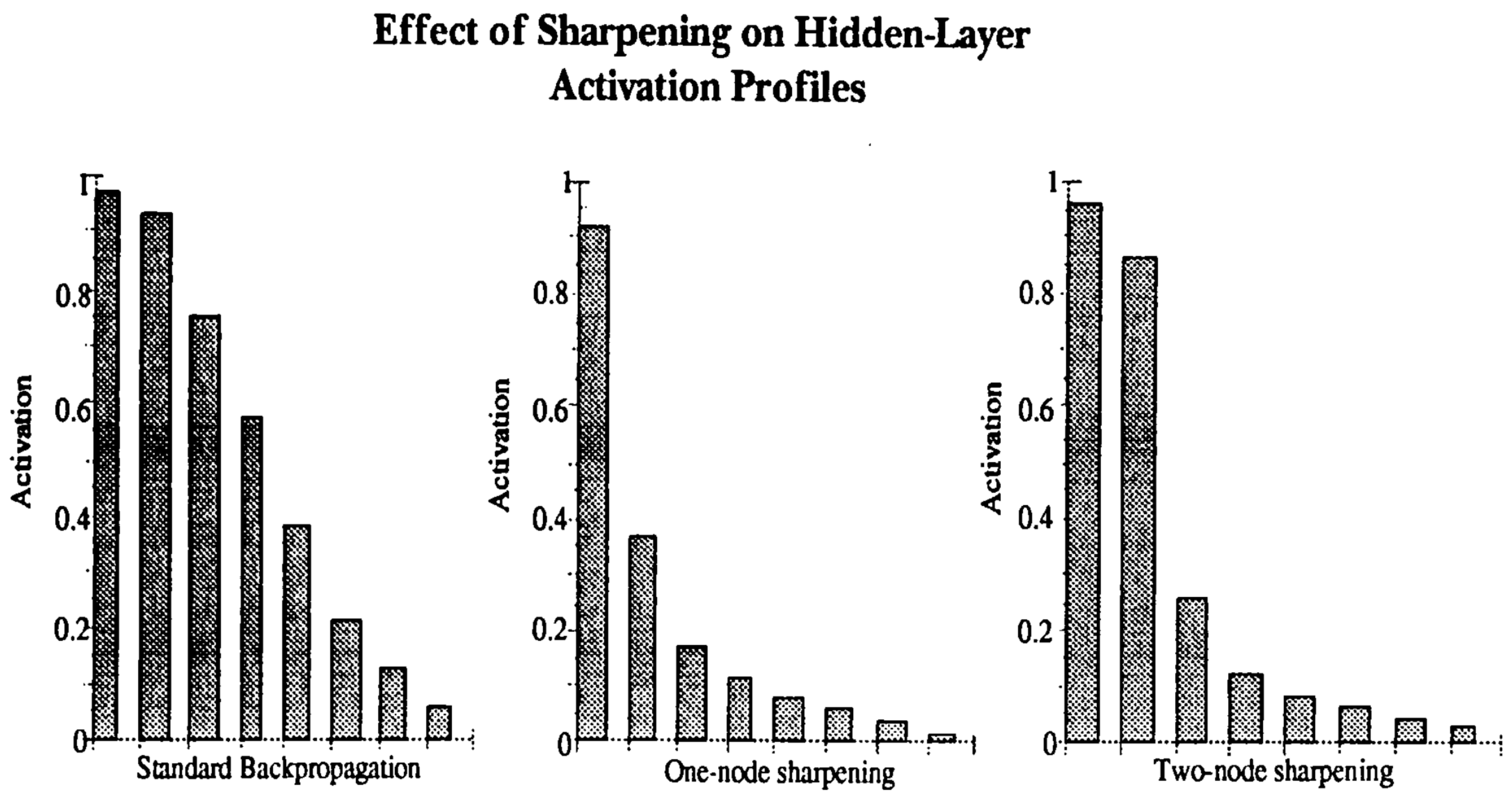


- Perform a forward-activation pass from the input layer to the hidden layer. Record the activations in the hidden layer;
- “Sharpen” the activations of *k* nodes;
- Using the difference between the old activation and the sharpened activation on each node as "error", backpropagate this error to the input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

Universität Bremen

Activations: “sharpen” activations (or sparsify)



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

(Some) implicit CL architecture perspectives

- **Spaces**: similar to CURL, we could try to map different tasks to different sub-spaces
- **Parameters**: inspired by the motivation for EWC & regularization, we could enforce creation of sub-modules through parameters
- **Activations**: inspired by the motivation behind SI, we could separate out activations for different tasks
- **Gating/Attention**: we could not modify the architecture at all, and include external gates or “attention” computations for separate tasks

Let us look at one example algorithm for each of these categories
(Note: there are numerous nuanced works for each of these approaches)

Gating/Attention: conditioning layers

Intuitively: learn something like a “task-identifier” that can be used to condition every layer of a neural network

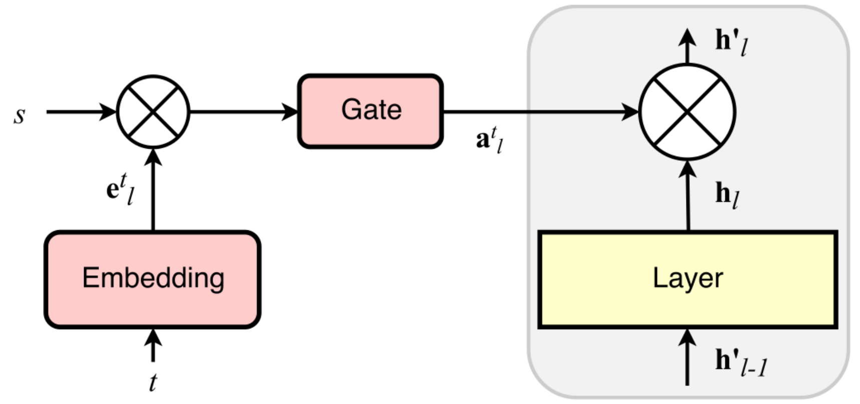
In general, elementweise multiply hidden layers h with an “attention” matrix a: $h_l' = a_l^t h_l$

We could now either form a probability distribution or pay “hard attention” to the task by using a “gating” mechanism to form (binary) attention matrices

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

Gating/Attention: conditioning layers

For instance: use a sigmoid gate $a_l^t = \sigma(s e_l^t)$, where s is a scaling parameter & $\sigma(x) \in [0,1]$ is a gate function based on the single-layer embedding e_l^t



Think of this as similar to PathNet, but instead of constraining entire paths/modules, going down to a single unit level

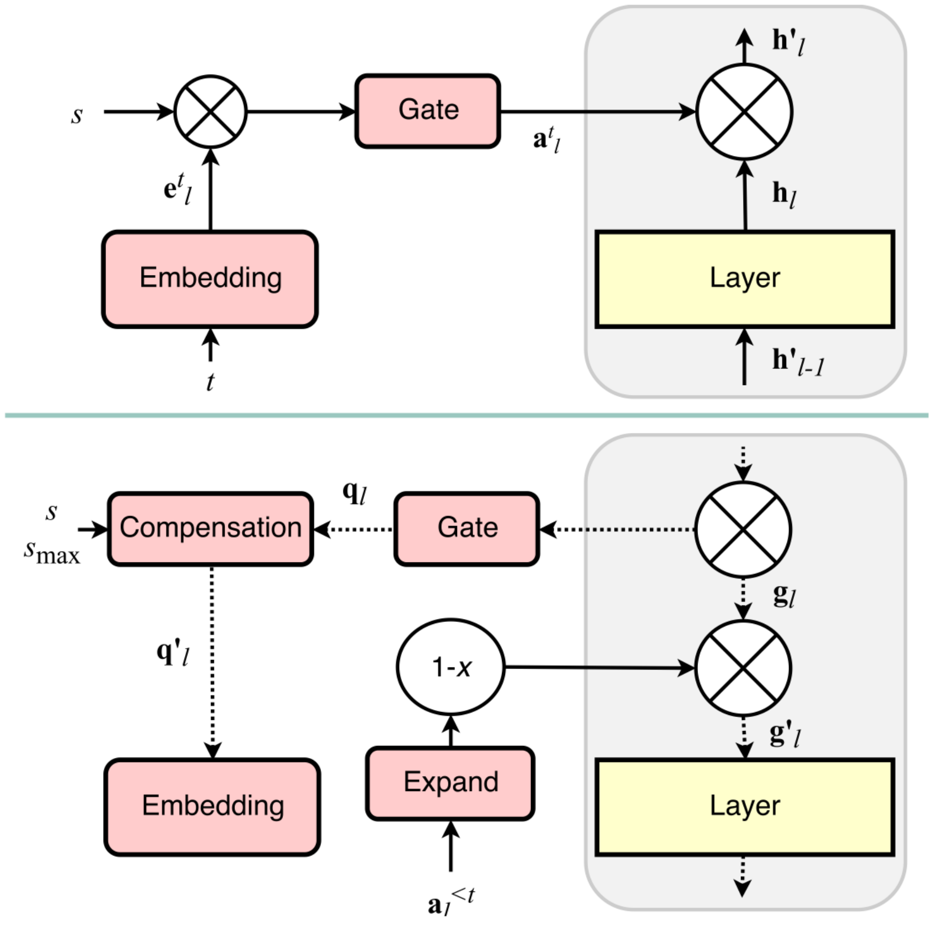
Serrà et al,"Overcoming Catastrophic Forgetting with Hard Attention to the Task", ICML 2018

Universität Bremen

Gating/Attention: conditioning layers

To preserve information, need to condition gradients according to cumulative attention from previous t: recursively $a_l^{\leq t} = \mathtt{max}(a_l^t, a_l^{\leq t})$

and modifying the gradient (which requires "expanding" to match the larger output dimension for more t)
$g'_{l,ij} = [1 - \mathtt{min}(a_{l,i}^{\leq t}, a_{l-1,j}^{\leq t})]g_{l,ij}$



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

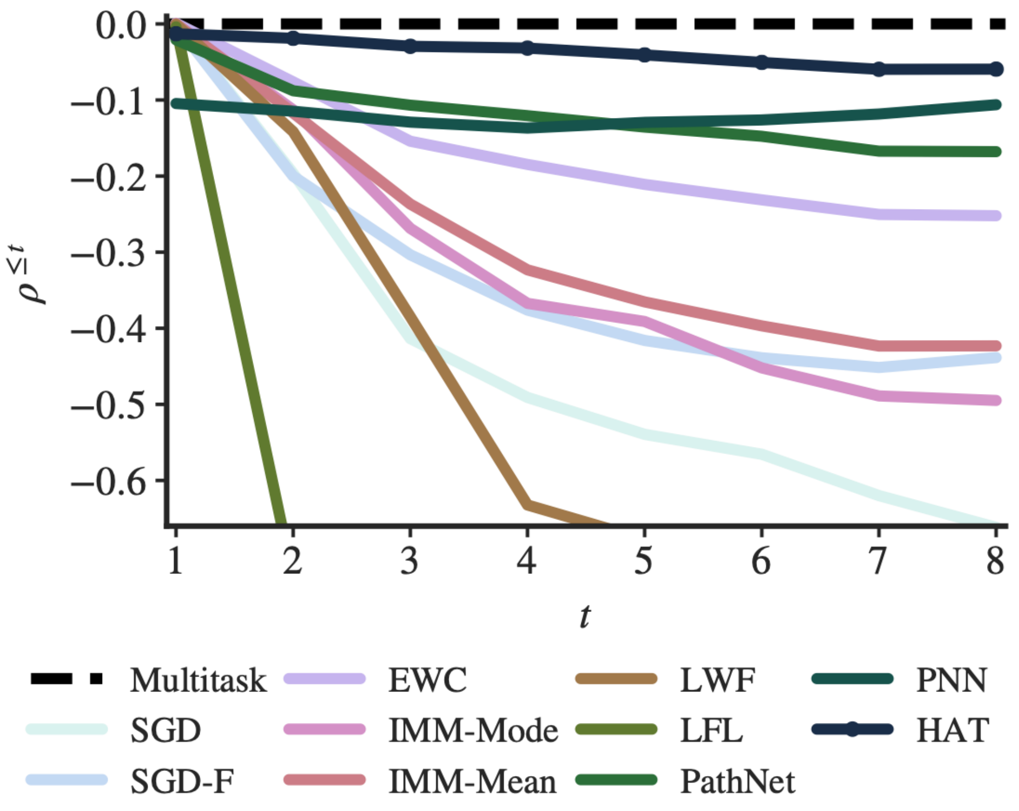
Gating/Attention: conditioning layers

Some extra hyper-parameter tricks we skip here, but "works well" in general (similar to PathNet)

Here "forgetting ratio", defined as:

$$\rho^{\tau\leq t} = \frac{A^{\tau\leq t} - A_R^{\tau}}{A_J^{\tau\leq t} - A_R^{\tau}} - 1$$

($\tau \leq t$ = task τ acc after learning t; A_R = random model trained only on τ; A_J = multi-task joint training)



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

Lots & lots of other approaches: like "experts"

"Expert" based approaches are very popular and are a somewhat "safe" bet if the "backbone" is large enough + tasks "similar"

In general, a main objective/ challenge is to infer the correct task for a novel data point during inference, such as to select the appropriate part of the model

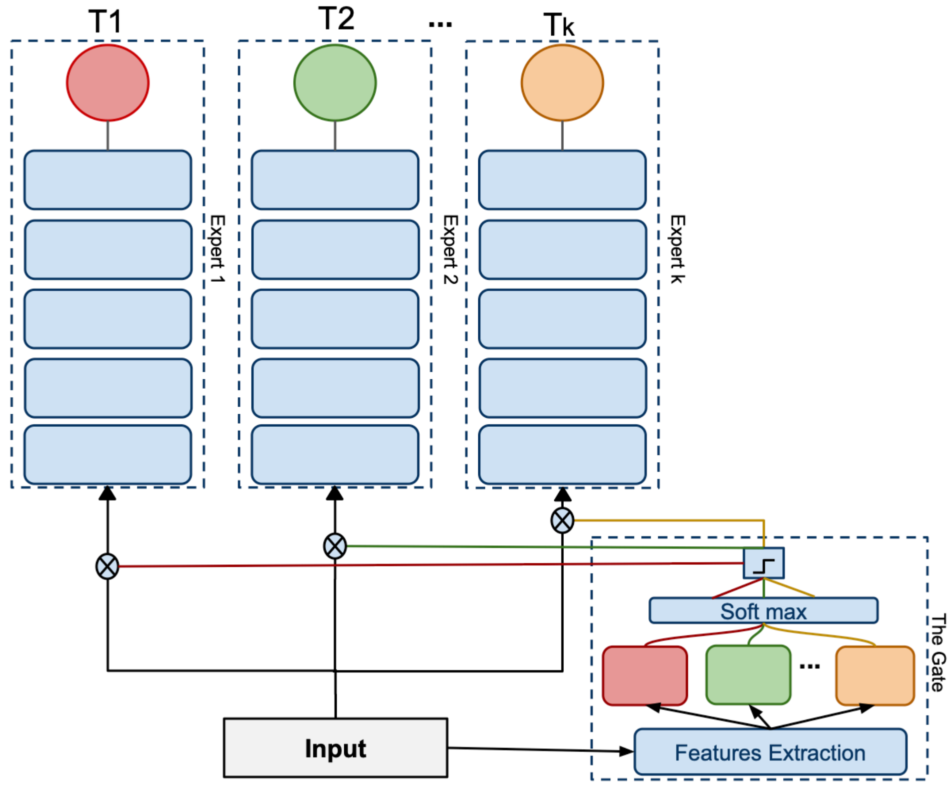


Figure 1. The architecture of our Expert Gate system.

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

Incredibly popular in LLMs: “adapters”

LLMs, that are way too costly to train for anyone, make use of task-specific “adapters” all the time!

Low-rank-adapters (LoRA): inject low rank matrices & only train those.
Instead of full $W_0 \in \mathbb{R}^{d \times k}$ update with low-rank decomposition
$W_0 + \Delta W = W_0 + BA$ where
$B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$ & $r \ll \texttt{min}(d, k)$

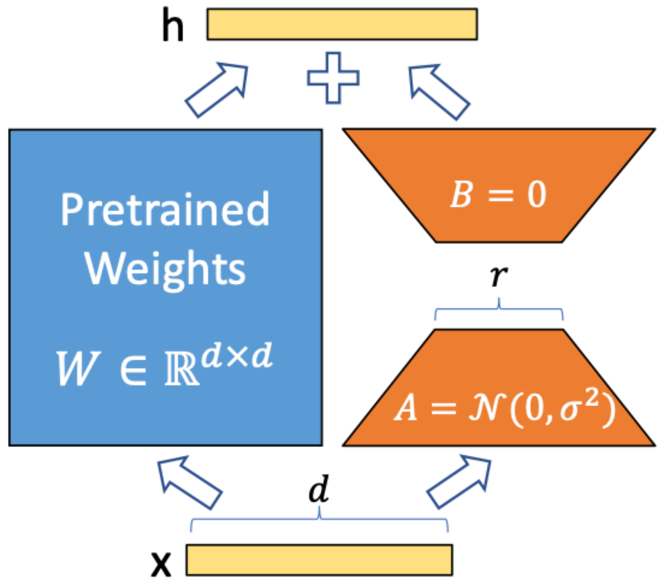

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 1: Our reparametrization. We only train A and B.

E. Hu et al, “LoRA: Low-Rank Adaptation of Large Language Models”,ICLR 2022

Question time

Why do you think LoRA is so popular for LLMs? What caveats do you see with other approaches we have encountered so far for very large models?

Popular in LLMs (& various large-scale models)

Pragmatically, only adapters & fine-tuning seem to be computationally efficient enough for now :(. EWC has a param^2 matrix, LLM training only passes the data once, so rehearsal is very costly, generation “snowballs”

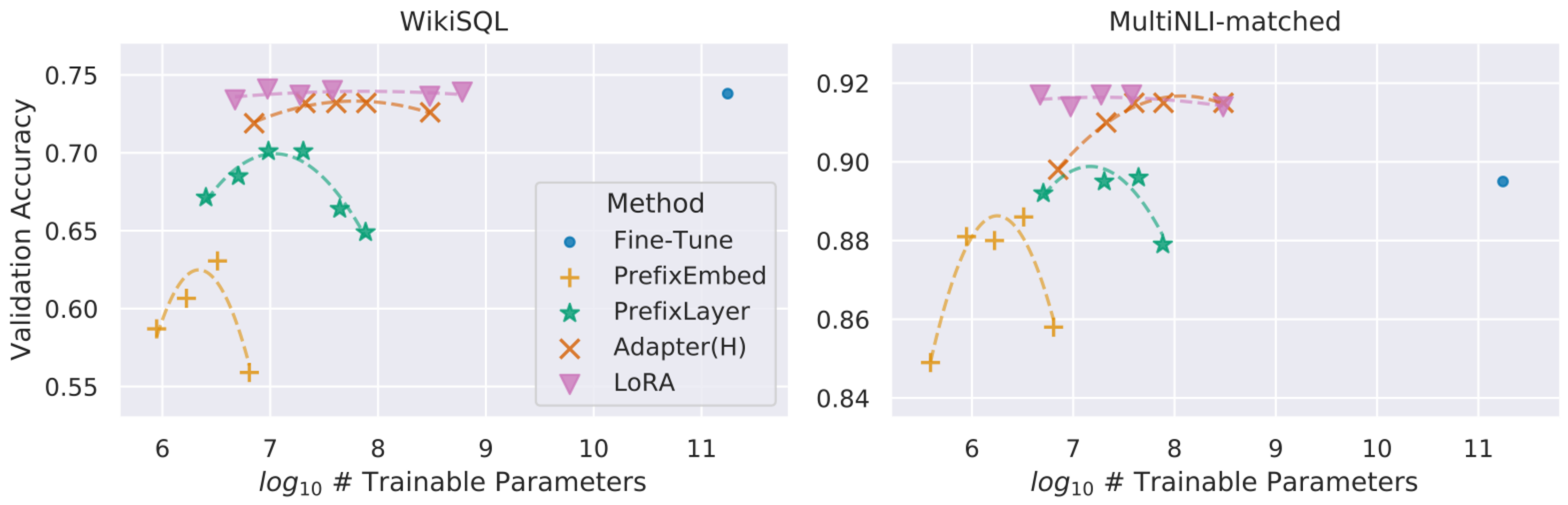


Figure 2: GPT-3 175B validation accuracy vs. number of trainable parameters of several adaptation methods on WikiSQL and MNLI-matched. LoRA exhibits better scalability and task performance.

E. Hu et al, “LoRA: Low-Rank Adaptation of Large Language Models”,ICLR 2022

Important: not all models are the same!

RECALL

“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.”

Important: not all models are the same!

But are representations equally
“dense” in all models?

Are activations equally overlapping
for possible model choices?

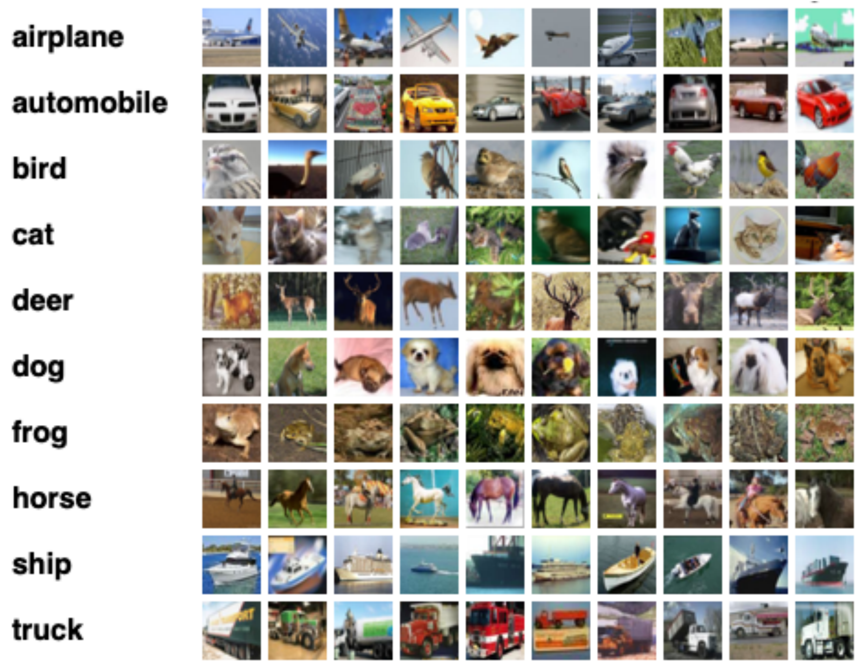
Universität Bremen

Important: not all models are the same!

But are representations equally "dense" in all models?

Are activations equally overlapping for possible model choices?

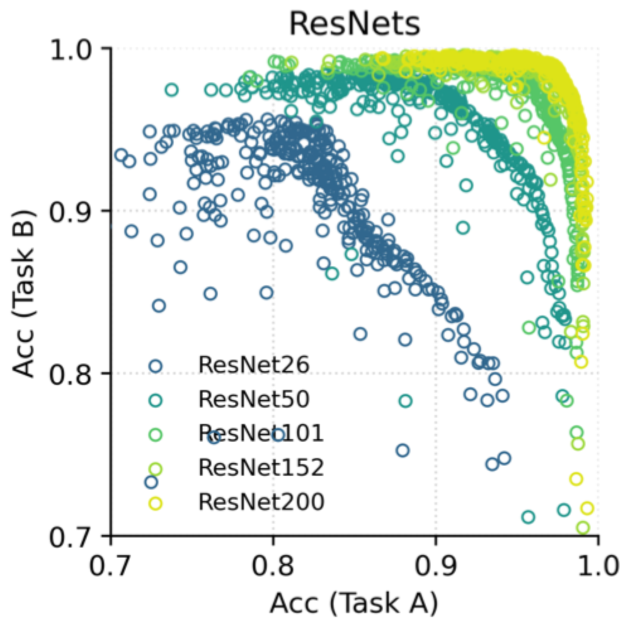
Let's analyze forgetting based on model choice & scale on split CIFAR-10 (going from the first five classes in task A to the second five)



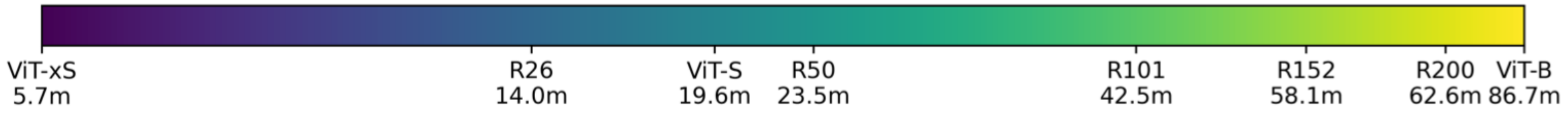
CIFAR images from https://www.cs.toronto.edu/~kriz/cifar.html

Universität
Bremen

Important: not all models are the same!



Average task A/B error improves systematically with model size

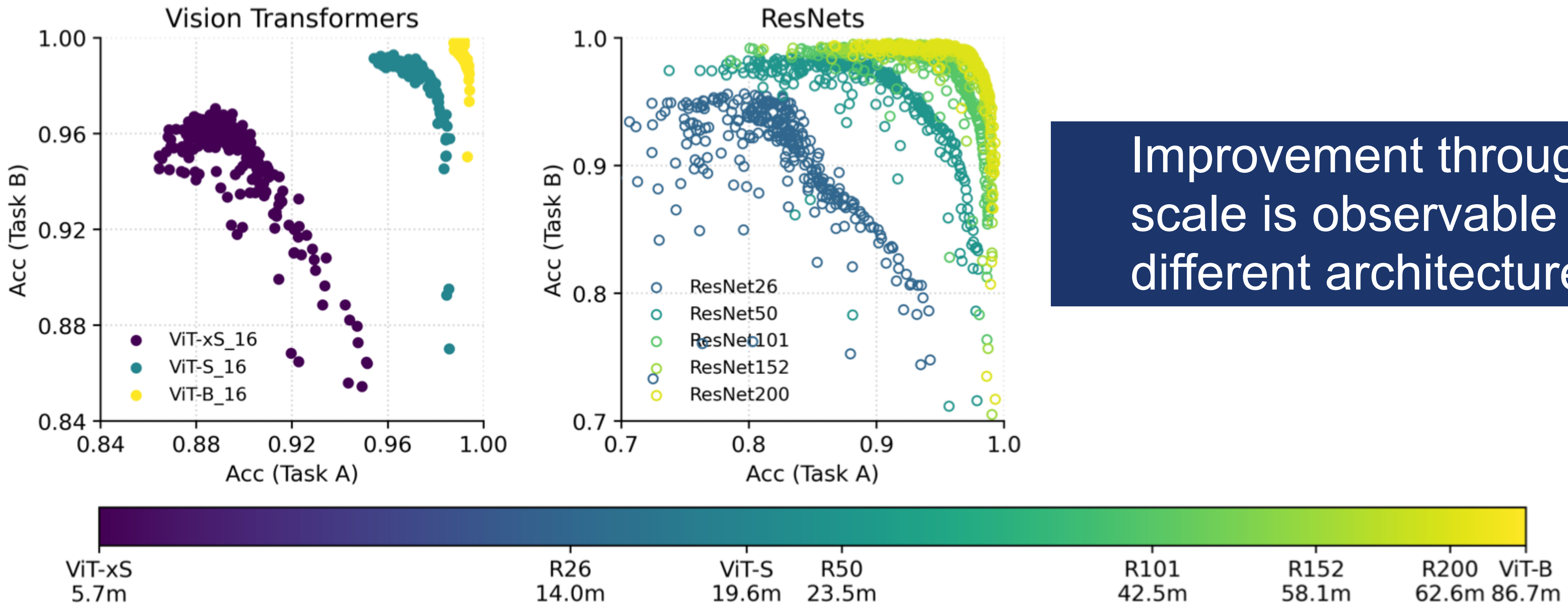


Ramasesh et al, “Effect of Model and Pretraining Scale on Catastrophic Forgetting in Neural Networks”, ICLR 2022

Universität Bremen

OWL ML

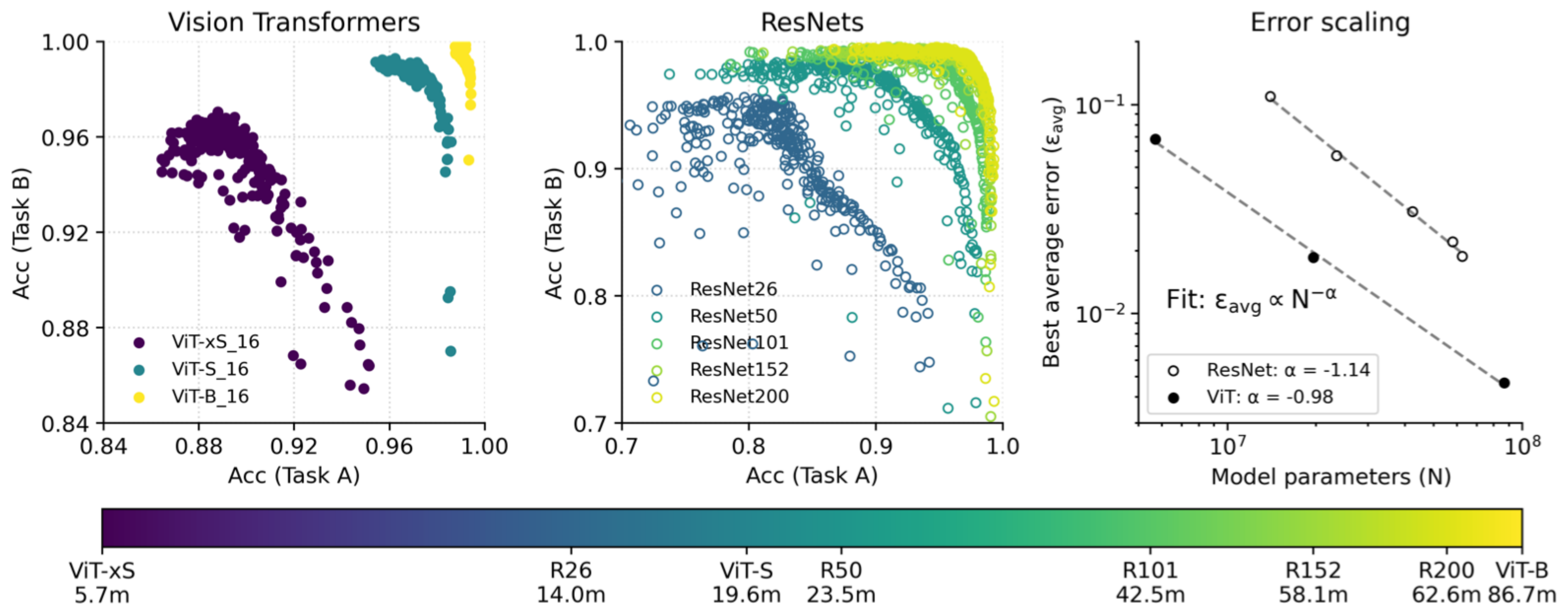
Important: not all models are the same!



Improvement through scale is observable for different architectures

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

For CIFAR-10, ViTs seem to forget less



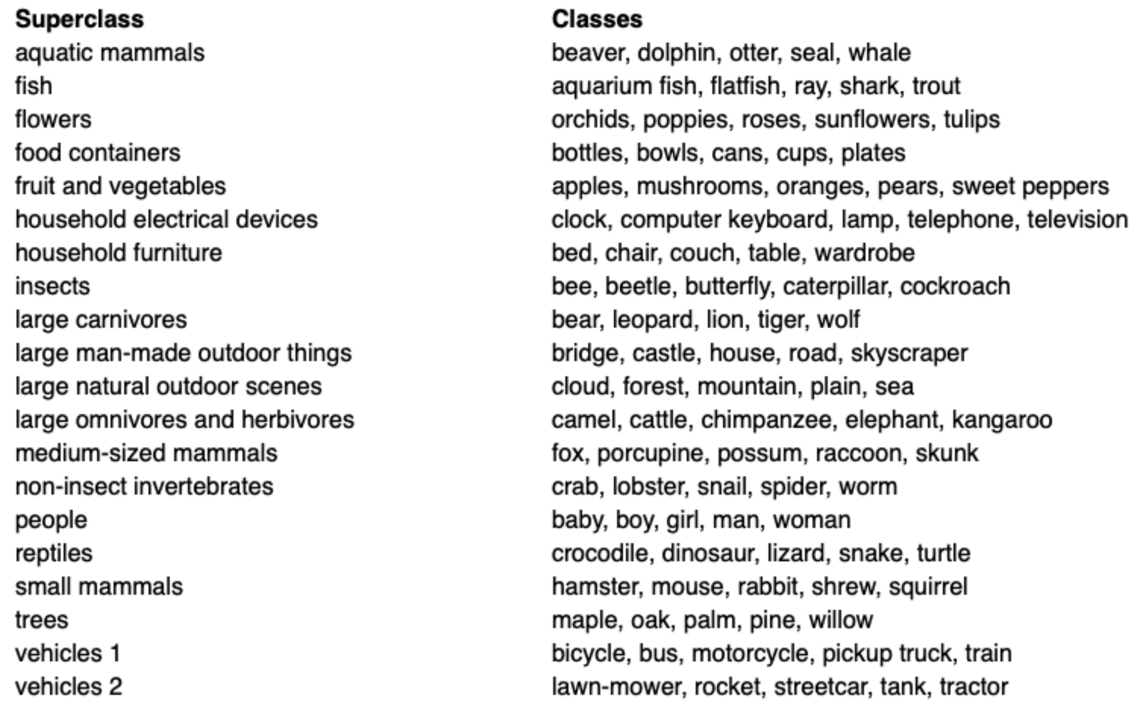
Ramasesh et al, “Effect of Model and Pretraining Scale on Catastrophic Forgetting in Neural Networks”, ICLR 2022

Are ViTs consistently better here?

Let's consider a second, input-distribution-shift, scenario (recall transductive transfer)

Here, based on CIFAR100, where we can sample a different subset from the super classes in task A & B

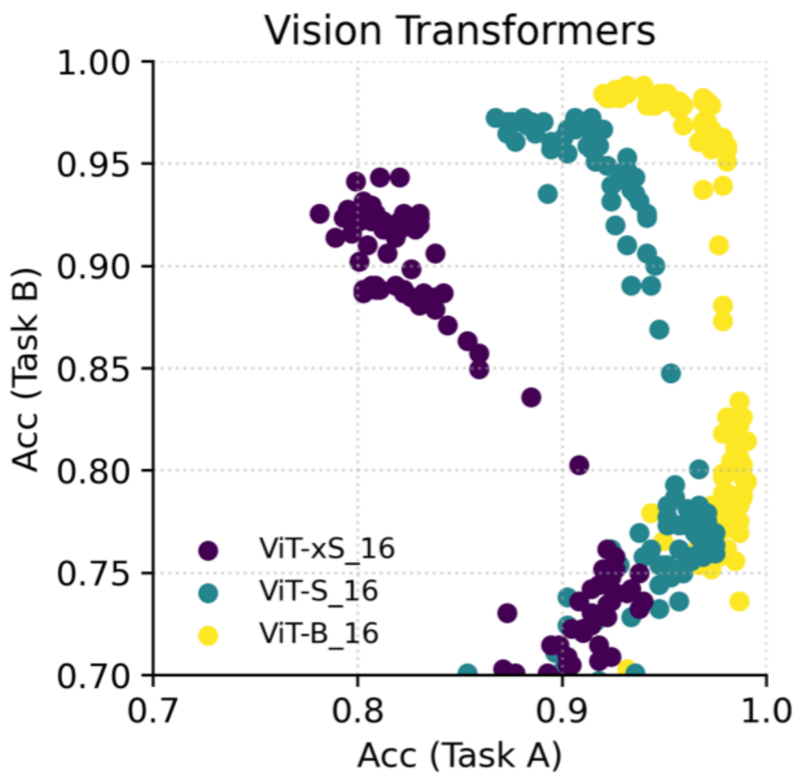
This implies we don't need to modify the task(-head) itself

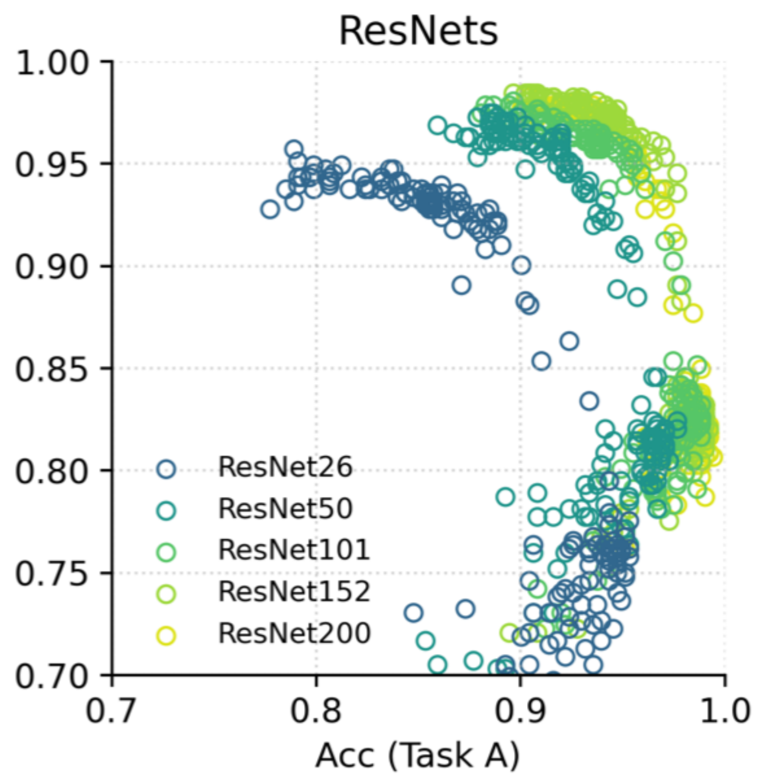


Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

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

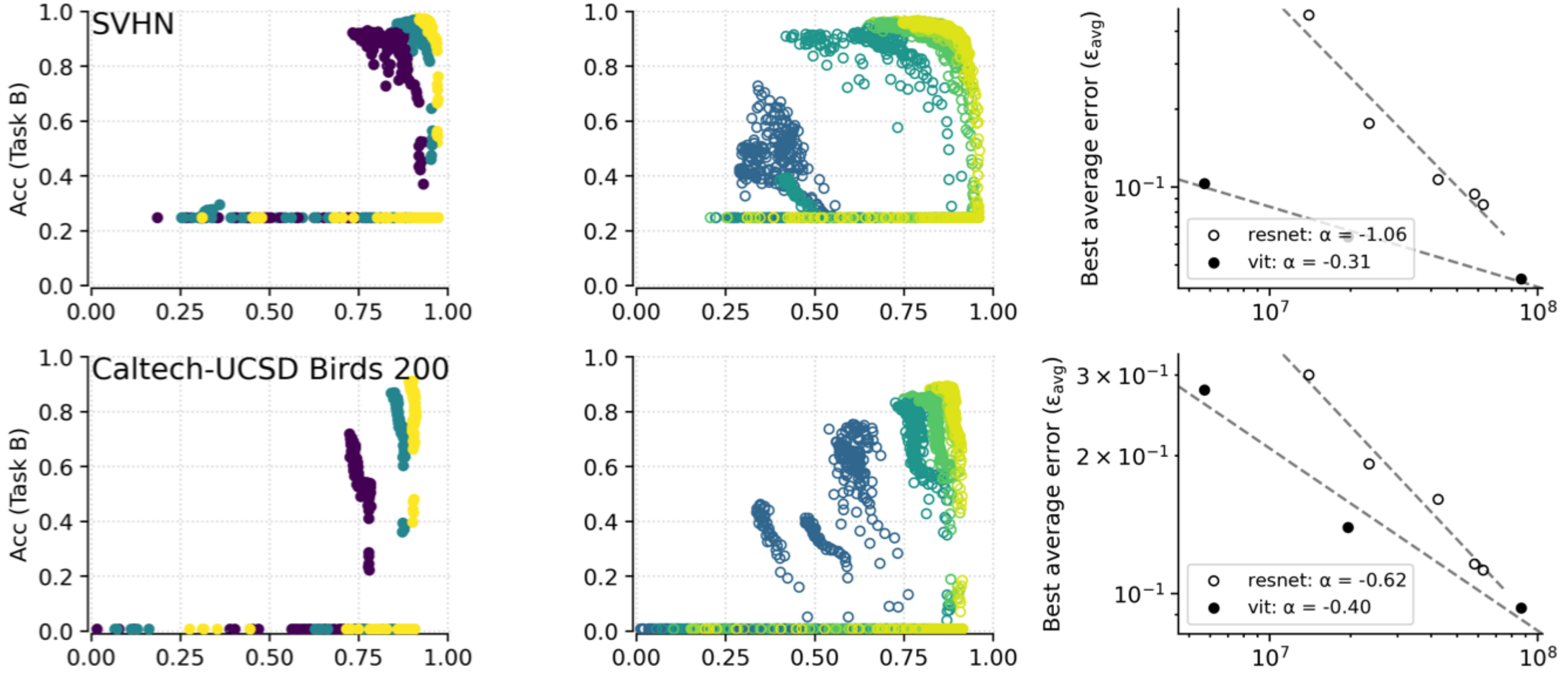
Task dependent, but seems better at small scale





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

Task dependent, but seems better at small scale



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

Question time

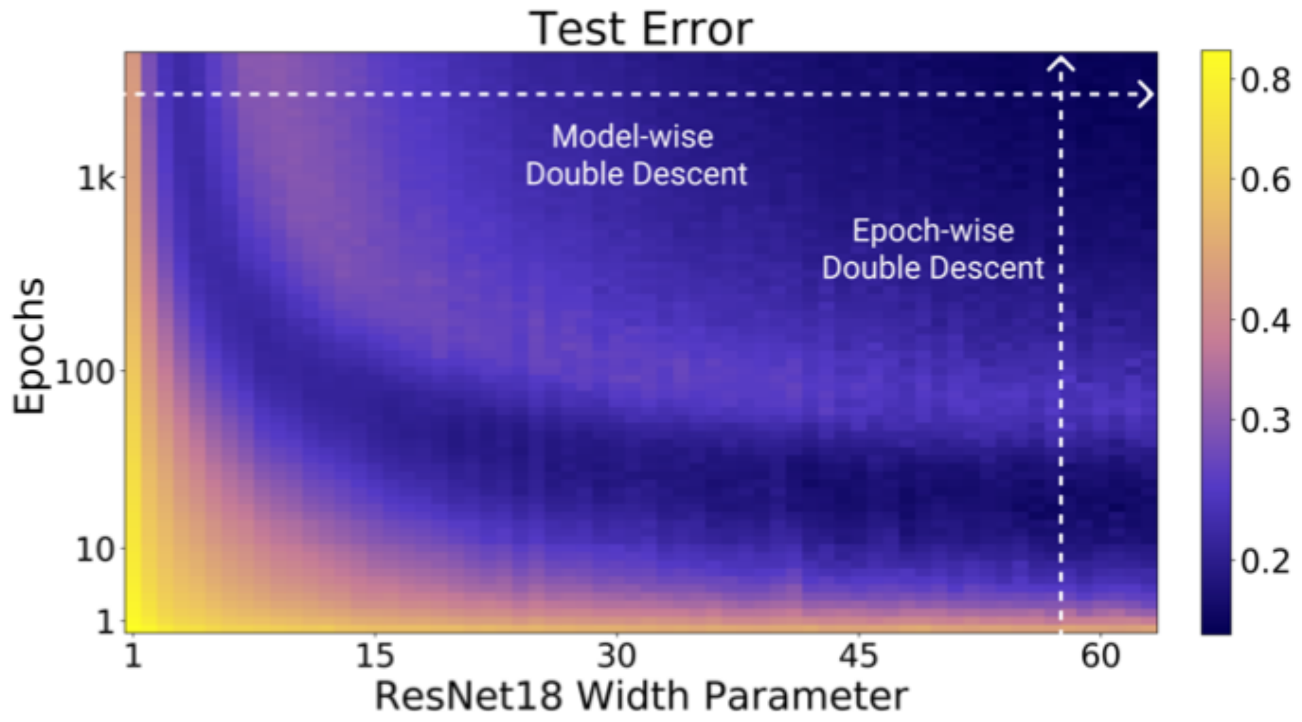
Recall deep double descent (first lecture), which other dimension matters on top of model size?

Universität Bremen

Recall: deep double descent

- With *increased model size*, performance *first gets worse*, *then* gets *better*
- Similar "deep double descent" phenomenon when *increasing training steps*

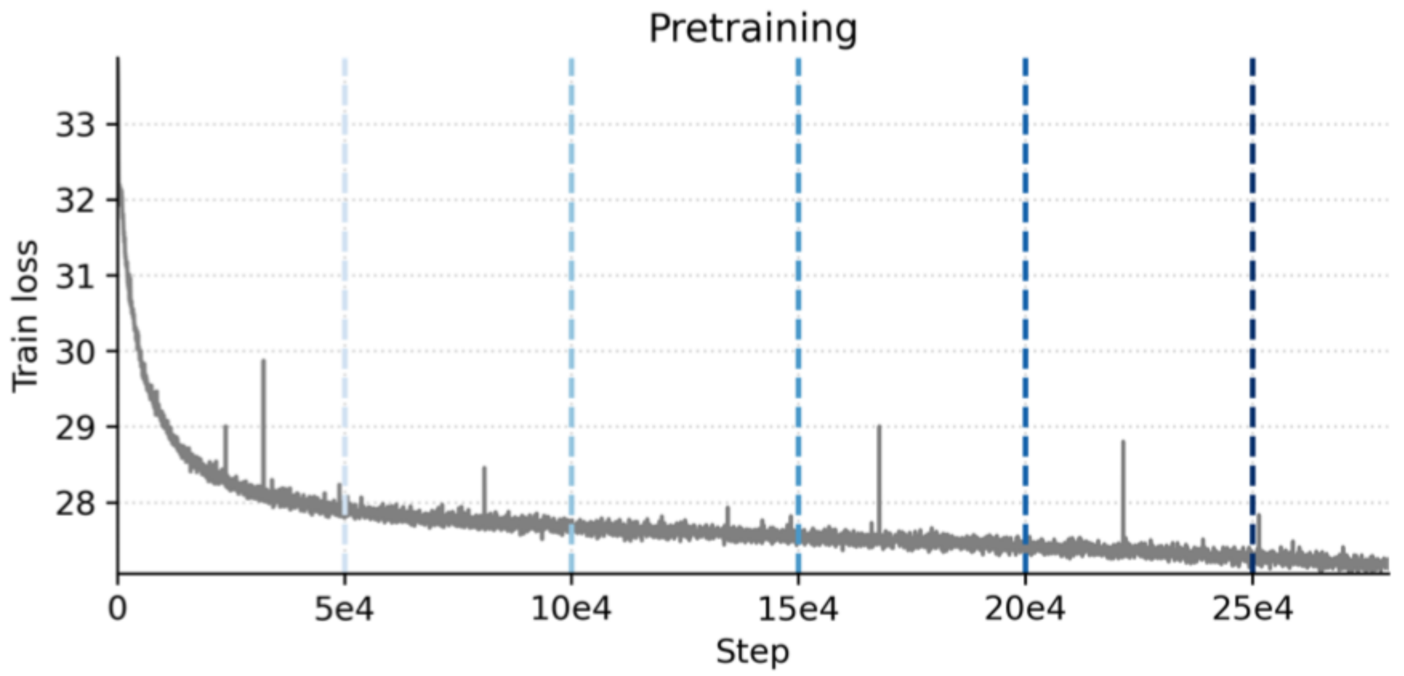
Nakkiran et al, "Deep Double Descent: Where Bigger Models and More Data Hurt", ICLR 2020

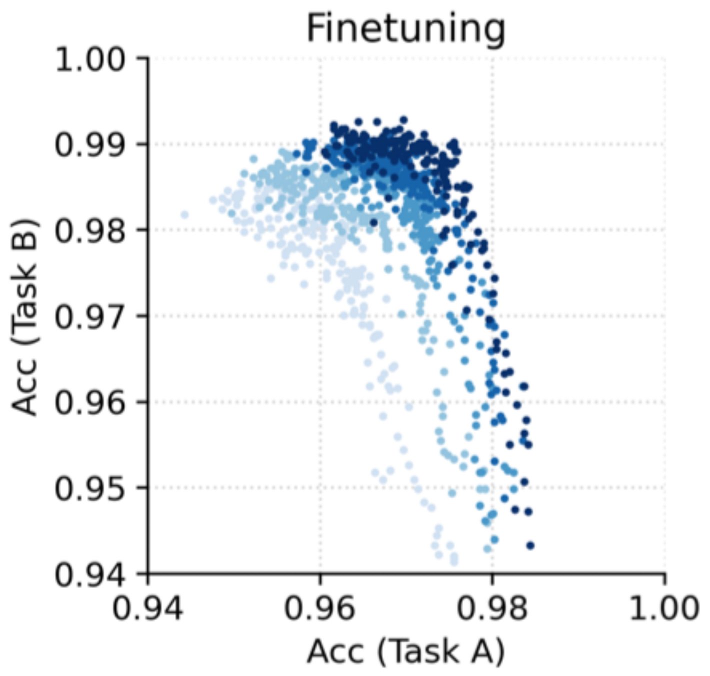


Universität Bremen

Longer “pre-training” seems to help

When we continue training longer on the initial data, fine-tuning seems to increasingly exhibit less overall forgetting





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

Universität Bremen

Pre-training seems to generally be beneficial

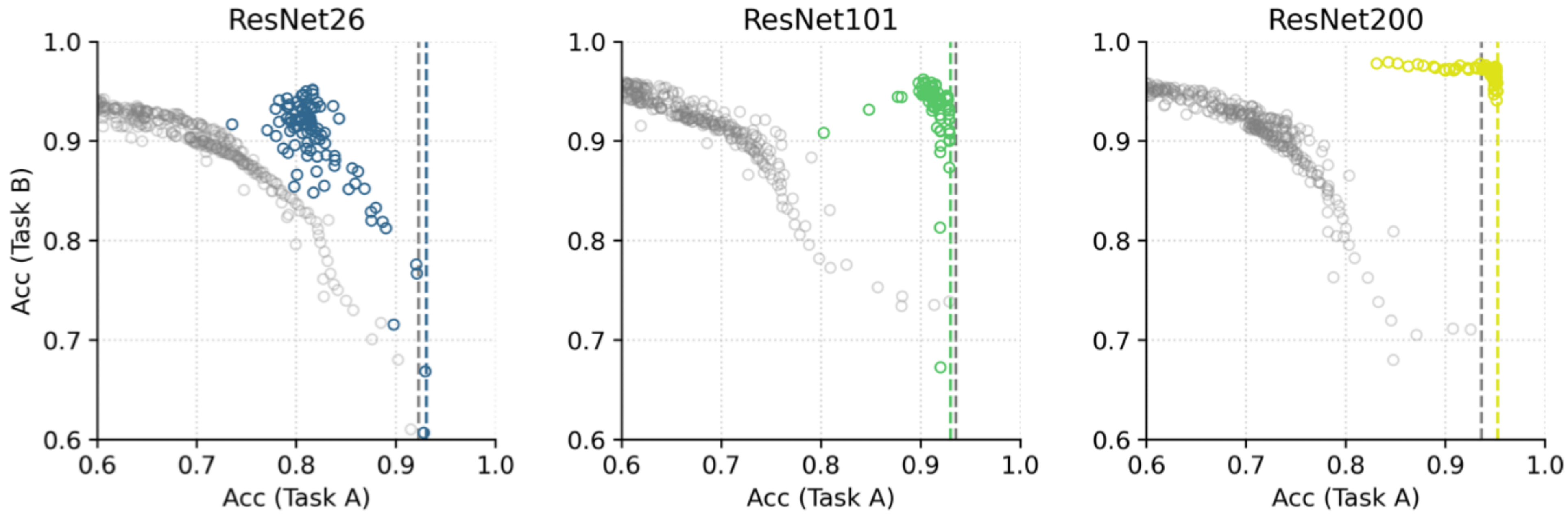


Figure 6: **Pretrained versus trained-from-scratch models.** ResNet models trained from scratch (gray) exhibit inferior forgetting performance than pretrained models (colored), even for the pretrained models shown here, which were handicapped during fine-tuning, in order to match Task A performance. Futhermore, the trained-from-scratch models are unable to take advantage of model scale

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

Deep double descent explains larger size & steps, but what is the intuition behind more pre-training?

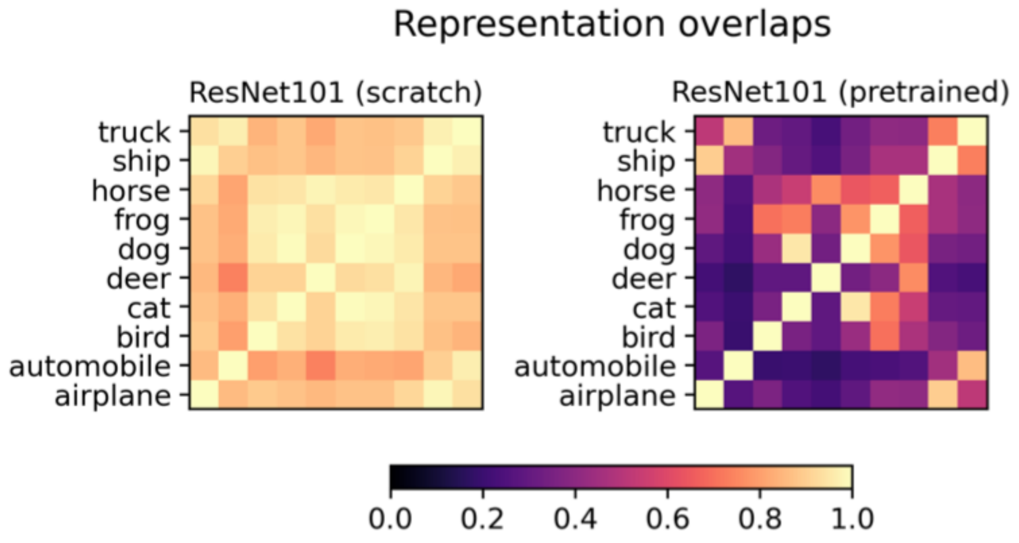
Investigate overlap in representations in terms of model features for Task A and B. As one choice, use trace overlap of the penultimate layer activations based on matrices of inner-products between task A and B features $\Theta_{AB} = f_A f_B^T$ (e.g. for CIFAR-10, 10x10 matrix)

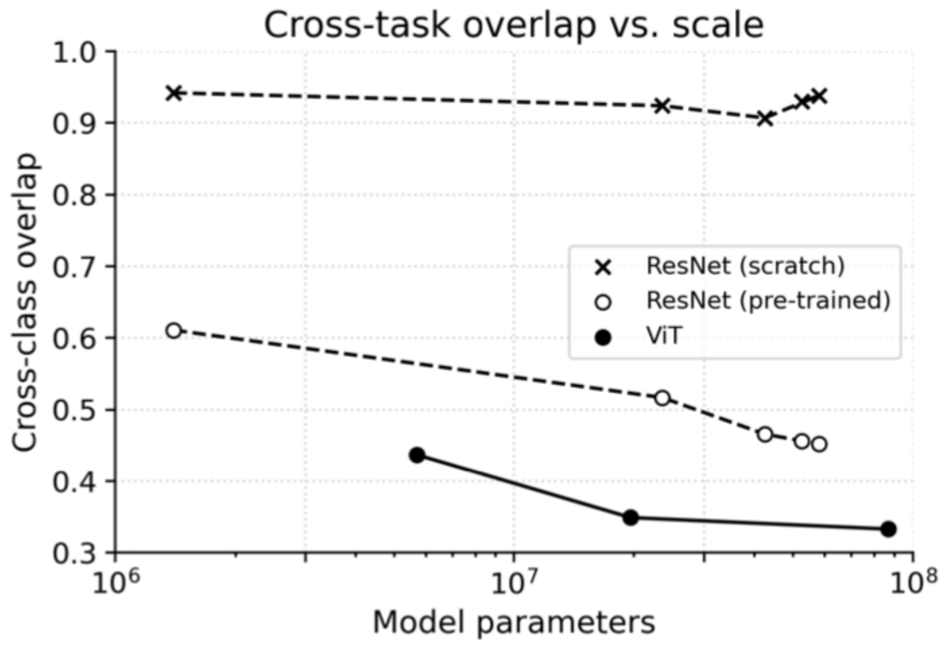
Trace overlap: $S_{AB} = \dfrac{Tr(\Theta_{AB}\Theta_{AB}^T)}{\sqrt{(Tr(\Theta_{AA}\Theta_{AA}^T)Tr(\Theta_{AB}\Theta_{BB}^T))}}$

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

Pre-training & representation overlap

For CIFAR-10: ViTs feature seem to generally have less class-overlap & pre-training reduces overlap significantly for CNNs





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

Question time

Despite being more intuitive (at least for me), why is adding capacity over time hard & unpopular?

We have learned about some empirical observations that fuel the LLMs' "quest for scale", but let's move to the "explicit" perspective.

Let’s find out: the “explicit” growth perspective

Plasticity from a different angle - inspired by **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)

Universität Bremen

An example for intuition: Dynamic Node Creation

Assume a fixed depth (layers) neural networks and start with a small amount of initial parameters

When training you will quickly face:

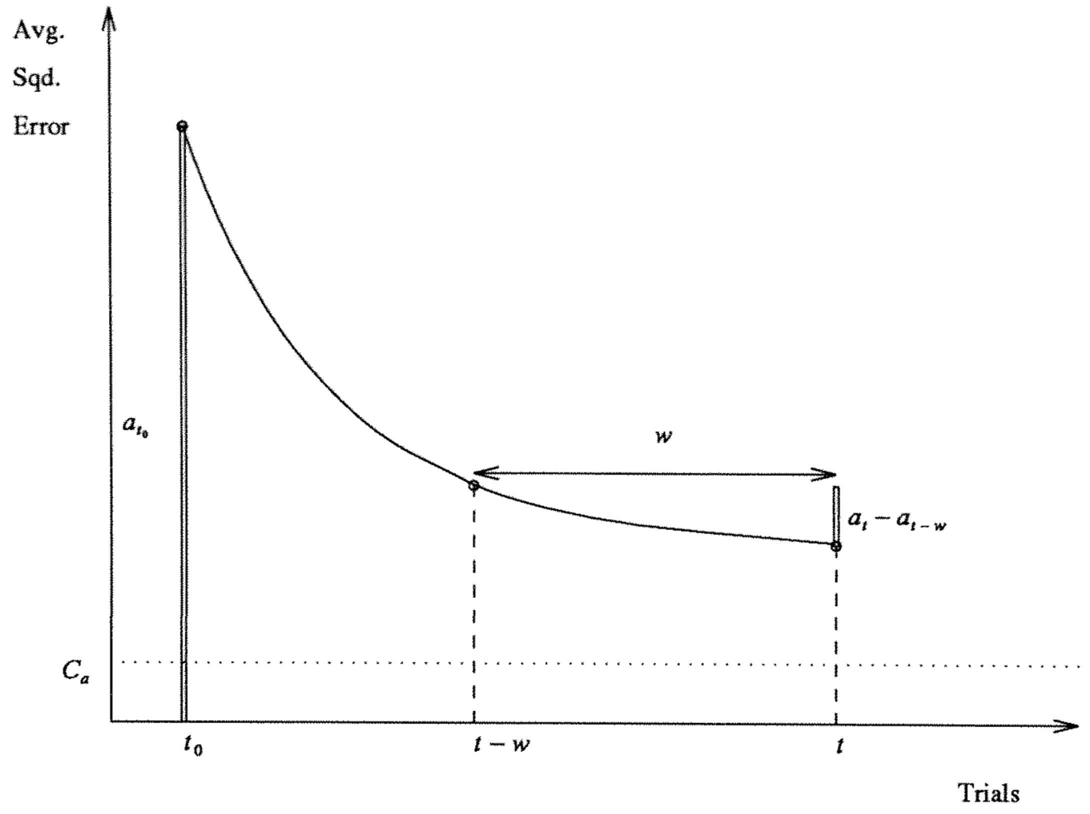
When to add more parameters?

When to add more parameters to the model?

DNC proposition:

- Assume monotonically decaying exponential for error
- Add new node when error plateaus
- Define plateau as relative according to a window: $a_t - a_{t-w}$

When to add more parameters?



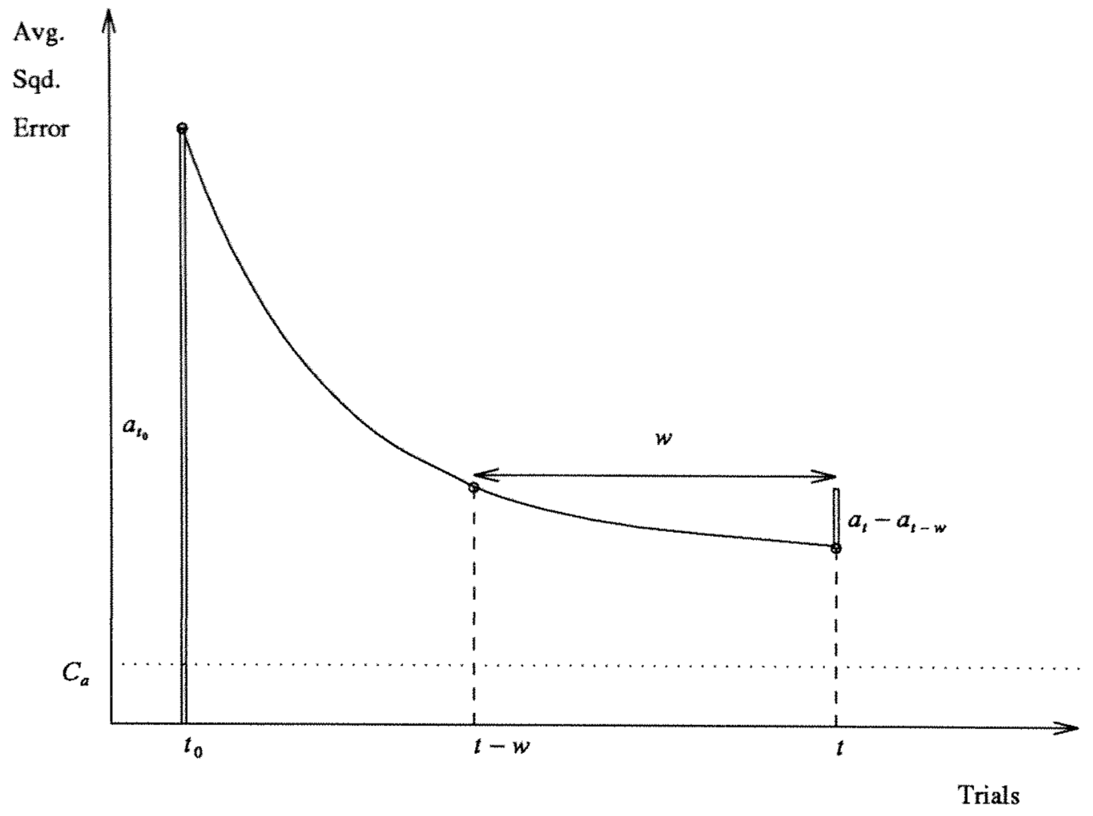
T. Ash, "Dynamic Node Creation in Backpropagation Networks", Connection Science 1:4, 1989

Universität Bremen

When to add more parameters to the model?

- Add parameters given pre-defined "trigger slope" to detect error plateaus:

$$\frac{a_t - a_{t-w}}{a_{t_0}} < \Delta_T$$



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

When to add more parameters to the model?

- Add parameters given pre-defined "trigger slope" to detect error plateaus:

$$\frac{a_t - a_{t-w}}{a_{t_0}} < \Delta_T$$

Note, this is fundamentally different from our easier implicit "freezing solutions"

When adding extra degrees of freedom, freezing constrains the solution a lot

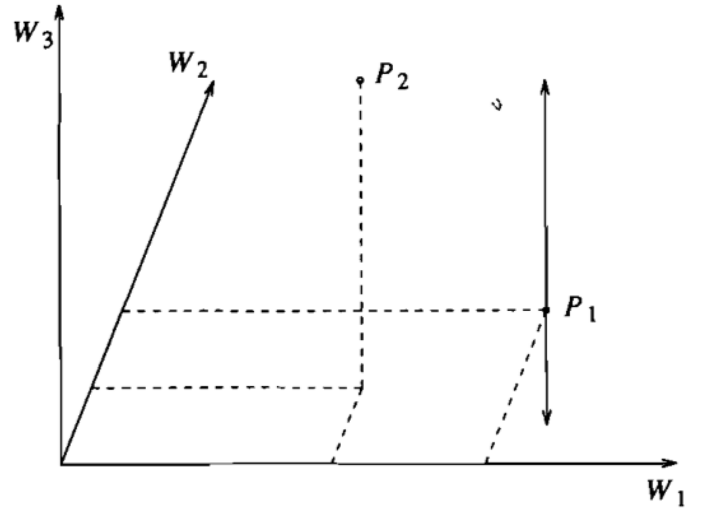


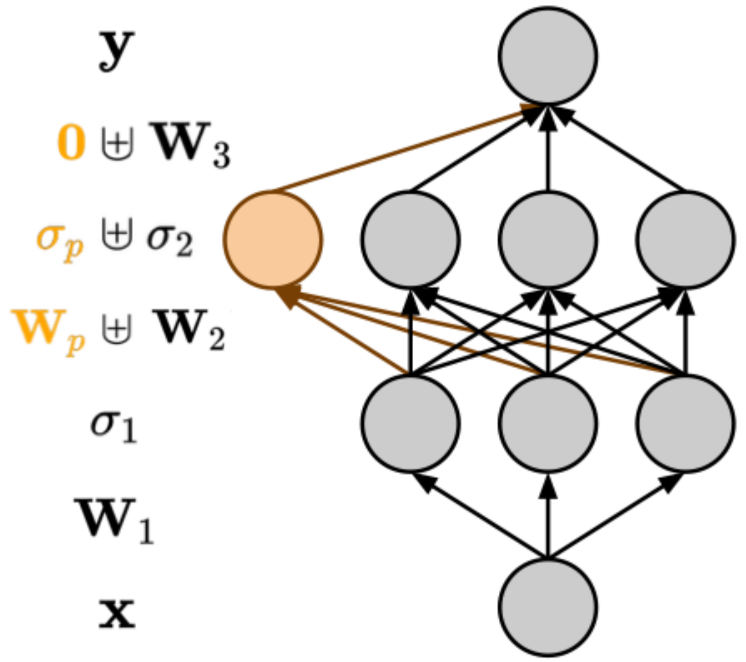
Figure 1. P_1 represents the point of lowest error in the plane defined by $W_1 \times W_2$. If another dimension is introduced (W_3), the global best becomes P_2. But freezing the values along W_1 and W_2 only allows solutions along the line through P_1 to be found. Generalizing to higher dimensions; only solutions in a particular *affine subset* of the weight space can be found.

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

Universität Bremen

Can we just add parameters?

- Increase network "width" by expanding the subsequent transform with zeros
- This leaves the function untouched, but proposes weights W_p are non-zero and eventually gradients will lead to change



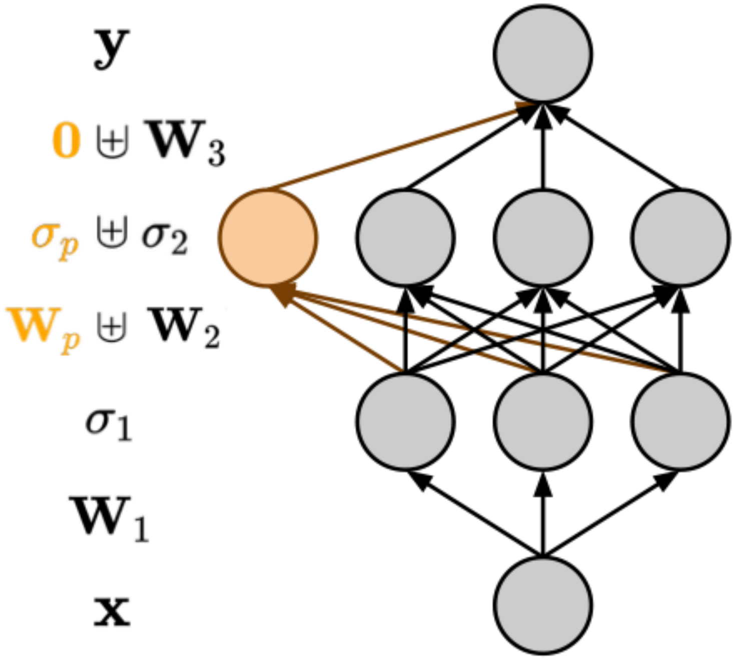
$\mathbf{y}$
$\mathbf{0} \uplus \mathbf{W}_3$
$\sigma_p \uplus \sigma_2$
$\mathbf{W}_p \uplus \mathbf{W}_2$
σ_1
$\mathbf{W}_1$
$\mathbf{x}$

Image from Mitchell et al, "Self-Expanding Neural Networks", arXiv:2307.04526, 2024

Can we just add parameters?

- Increase network "width" by expanding the subsequent transform with zeros
- This leaves the function untouched, but proposes weights W_p are non-zero and eventually gradients will lead to change

With $\frac{a_t - a_{t-w}}{a_{t_0}} < \Delta_T$ we never stop growing. When do we stop?

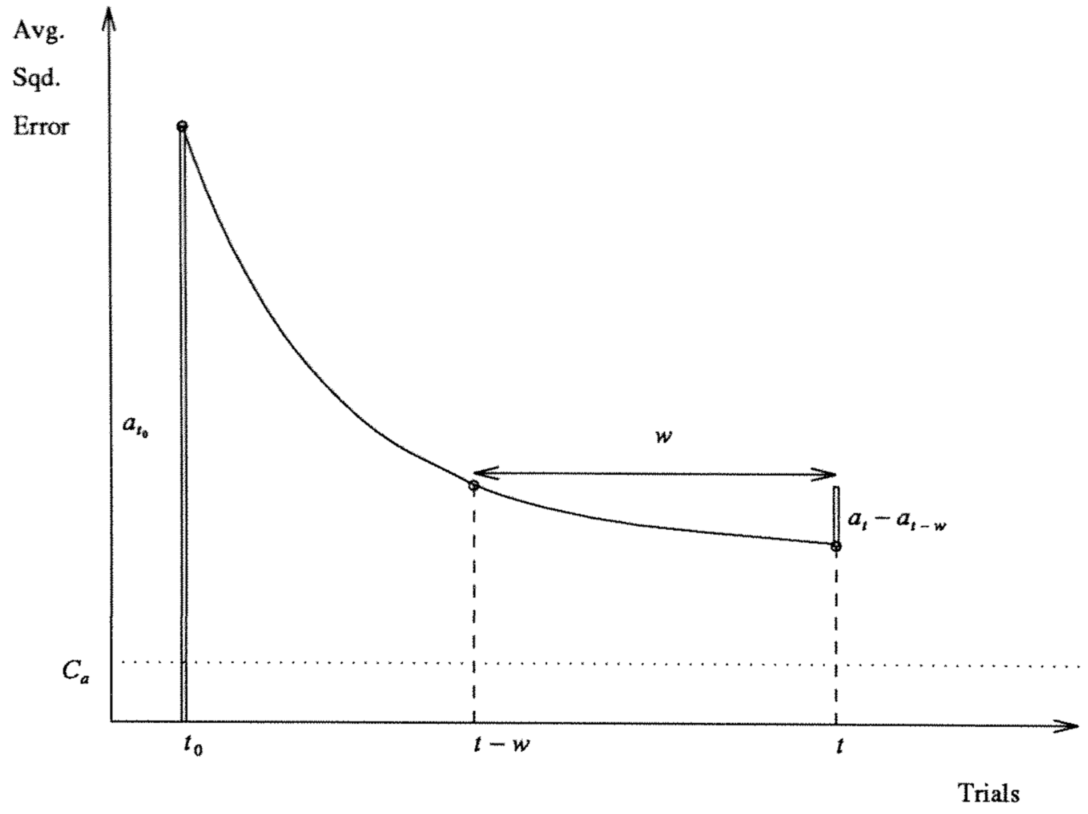


$\mathbf{y}$
$\mathbf{0} \uplus \mathbf{W}_3$
$\sigma_p \uplus \sigma_2$
$\mathbf{W}_p \uplus \mathbf{W}_2$
σ_1
$\mathbf{W}_1$
$\mathbf{x}$

Image from Mitchell et al, "Self-Expanding Neural Networks", arXiv:2307.04526, 2024

When to stop adding more parameters to the model?

- If we are well-aware of the task, we could set an cut-off according to absolute performance (e.g. measured through cross-validation)
- Alternatively, we would create another ratio, to relate the improvement obtained by the last addition to the improvement of the prospective one. I.e. ask "Does adding another unit still offer benefit?" With another threshold

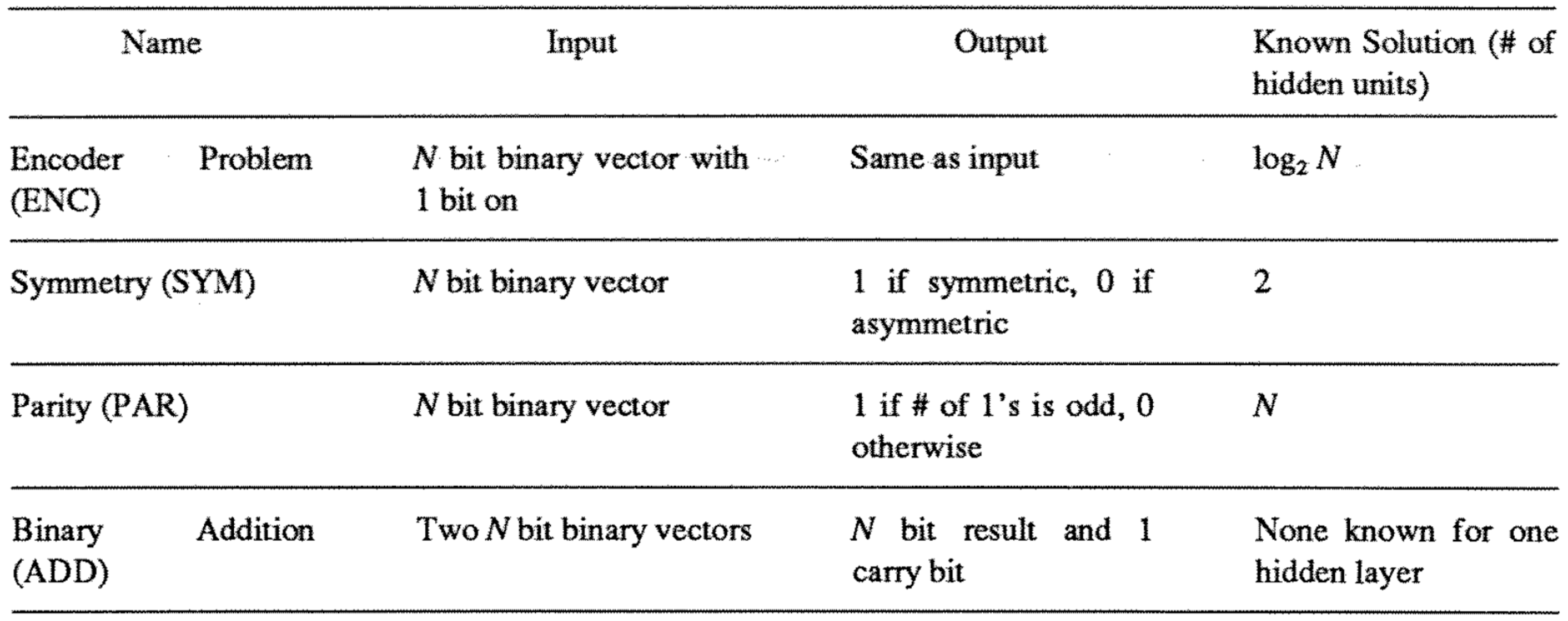


T. Ash, "Dynamic Node Creation in Backpropagation Networks", Connection Science 1:4, 1989

1989 Dynamic Node Creation in practice

Empirical investigation with a single hidden layer on “simpler” tests

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



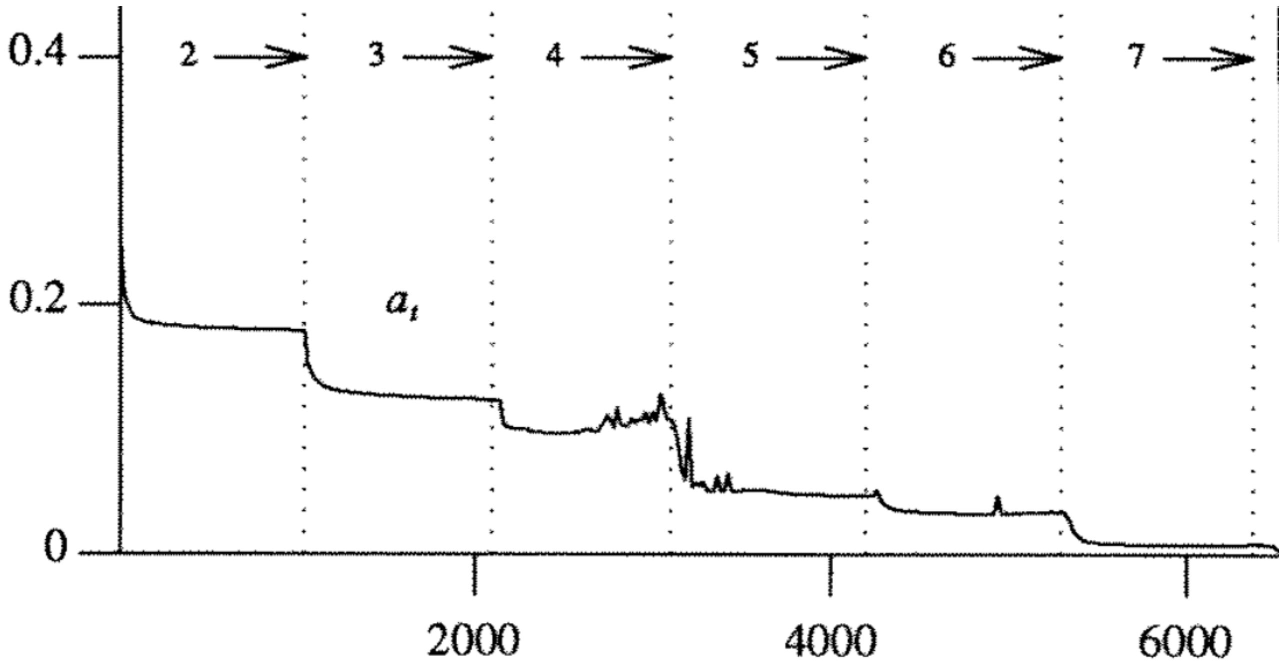
Name	Input	Output	Known Solution (# of hidden units)
Encoder Problem (ENC)	N bit binary vector with 1 bit on	Same as input	$\log_2 N$
Symmetry (SYM)	N bit binary vector	1 if symmetric, 0 if asymmetric	2
Parity (PAR)	N bit binary vector	1 if # of 1's is odd, 0 otherwise	N
Binary Addition (ADD)	Two N bit 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

Universität Bremen

1989 Dynamic Node Creation in practice

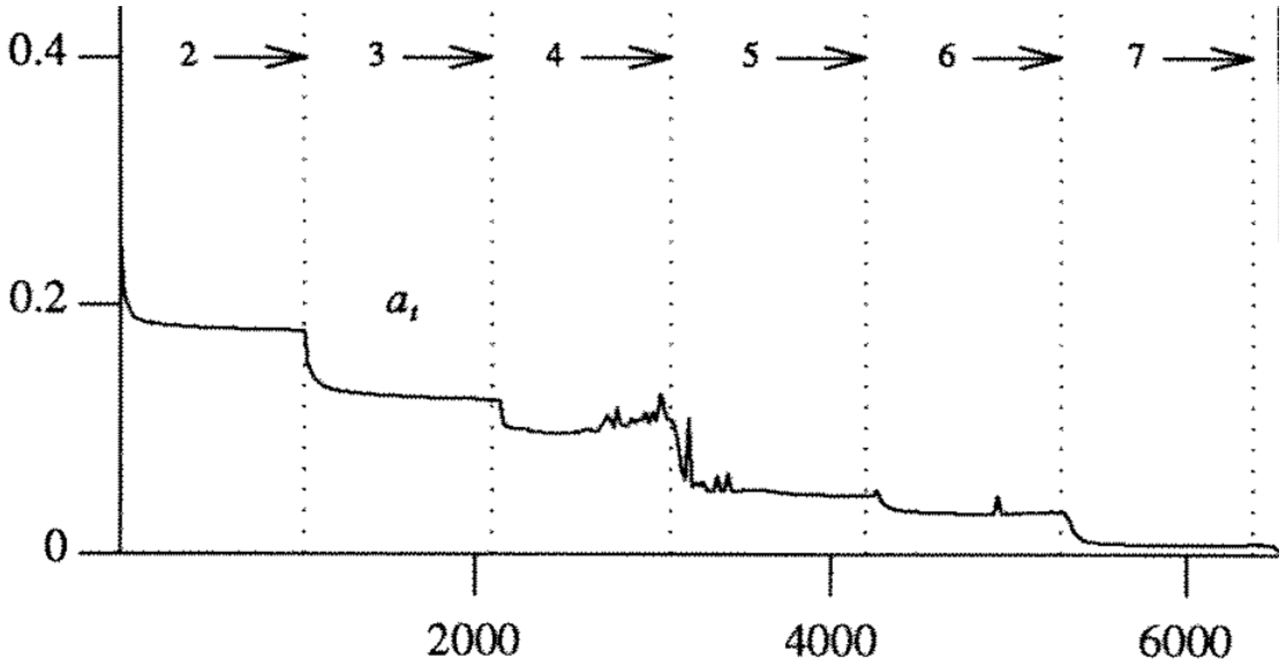
Here, squared error (y-axis) for the ADD3 problem



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

1989 Dynamic Node Creation in practice

Here, squared error (y-axis) for the ADD3 problem



Caution: at step 4, the window choice almost stopped the addition!

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

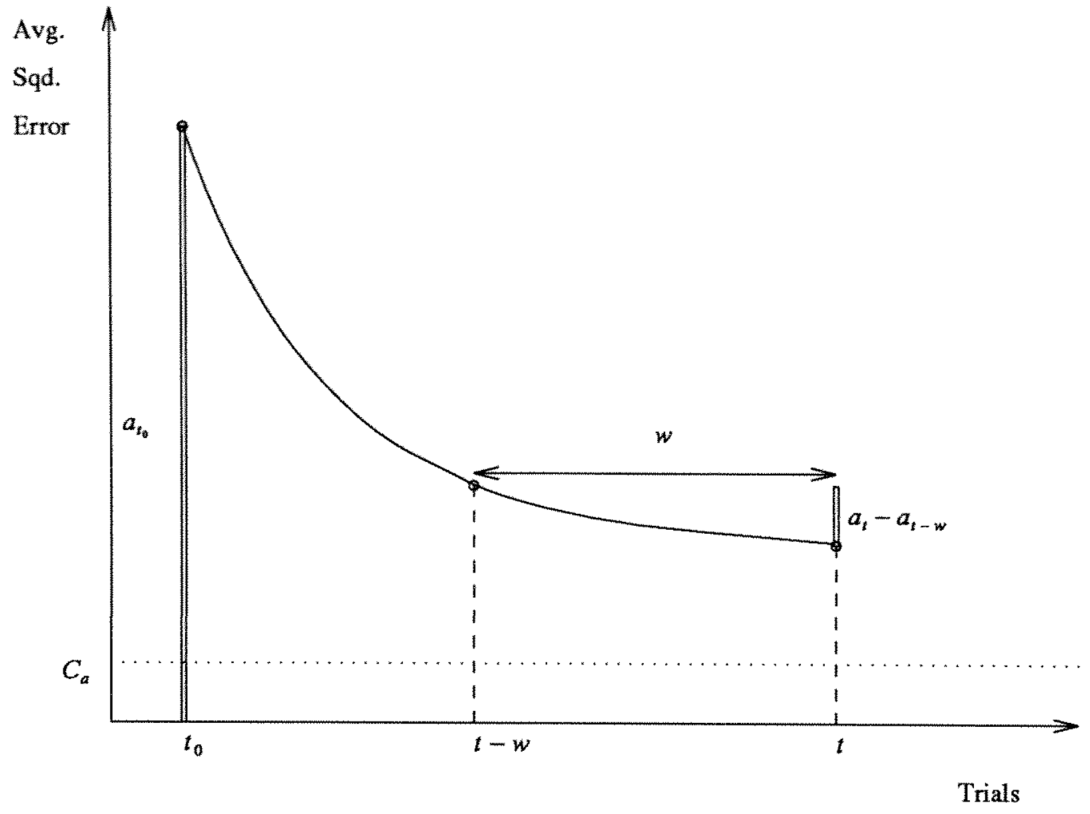
Universität Bremen

What DNC does not ask

DNC doesn't consider many critical questions & leaves them unanswered

Where do we add? What do we add?

- If we have more than one layer, where do we add a new parameter?
- Can/should we add a new layer instead of parameters?
- How should new weights be initialized?



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

Solving all there questions: when, where, what with params & layers is still largely unsolved

Let’s tackle these questions one step at a time, by first moving from 1989 DNC - that essentially only considers when to add/stop - to more modern examples based on neural networks with more layers

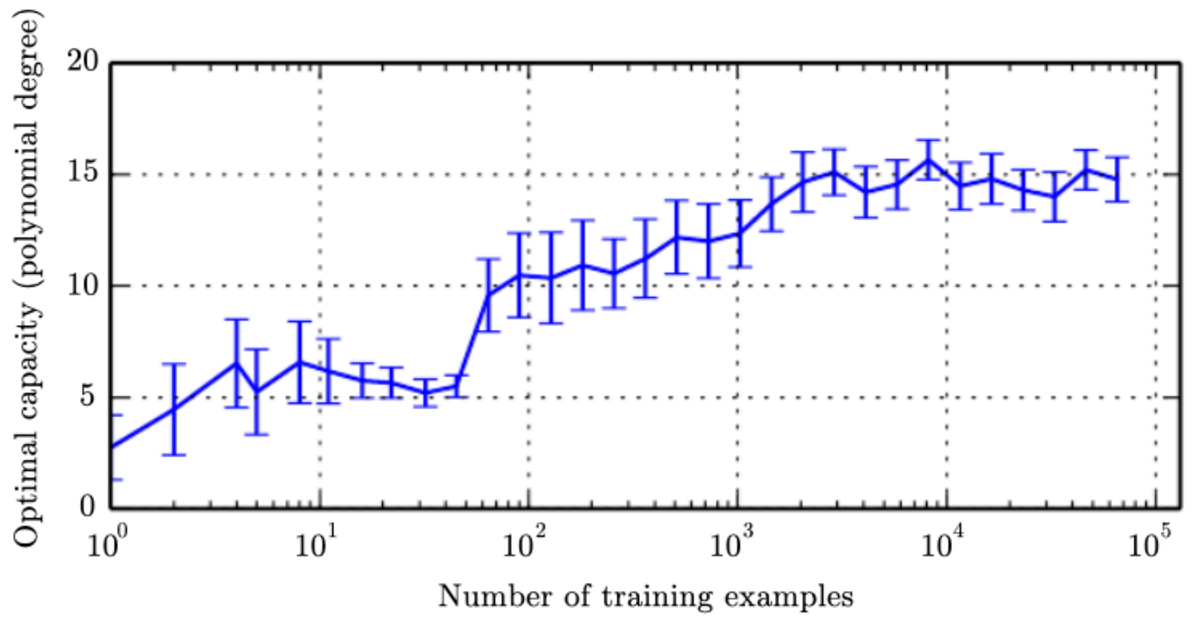
We start with a suggestion called “progressive neural networks”. It takes a look at parameter addition from a less general perspective.

Instead of finding the right capacity for any problem, it adds capacity per *discrete task* at *discrete points in time* to avoid *forgetting*

Universität Bremen

DISCLAIMER: before we proceed

It should have become clear from DNC and our prior discussion, that finding the right capacity is a general challenge for ML.
Recall our below image for optimal capacity vs. training examples

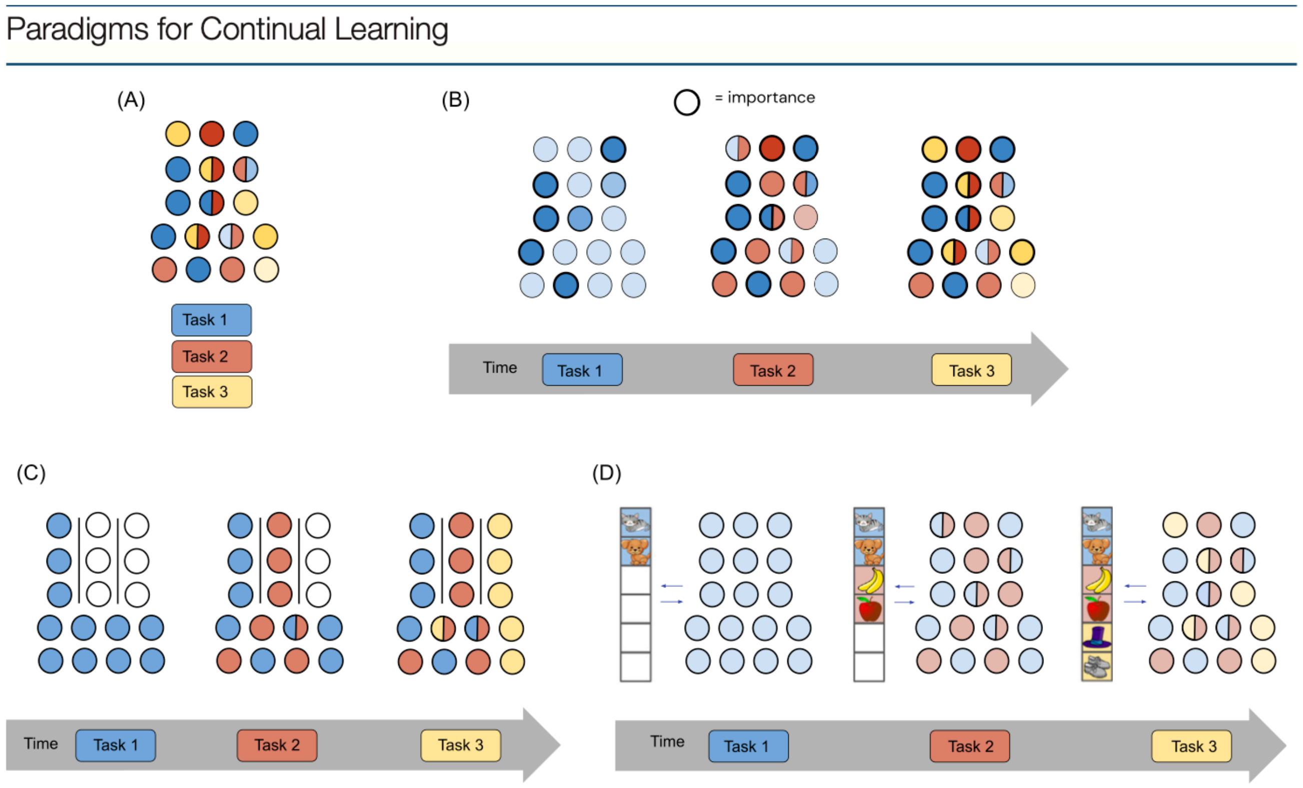


Deep Learning, Goodfellow, Bengio, Courville, MIT Press 2016

DISCLAIMER: before we proceed

“Growth” methods are often framed as one of “3 pillars of continual learning”: “regularization”, “rehearsal”, “architecture” approaches

This is because growth is beneficial to handle plasticity/forgetting, but it’s more than that and somewhat misleading

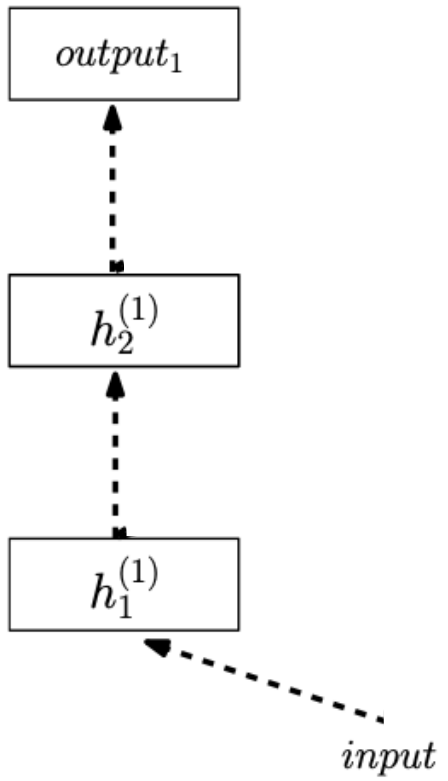


Hadsell et al, “Embracing Change: Continual Learning in Deep Neural Networks”, Trends in Cognitive Sciences 24:12, 2020

Universität Bremen

Progressive Networks

- We start with a single “column”, I.e. a neural network of our choice
- When switching to a second task, we will freeze the parameters of this “column”, and add a second column (i.e. we create a second network)



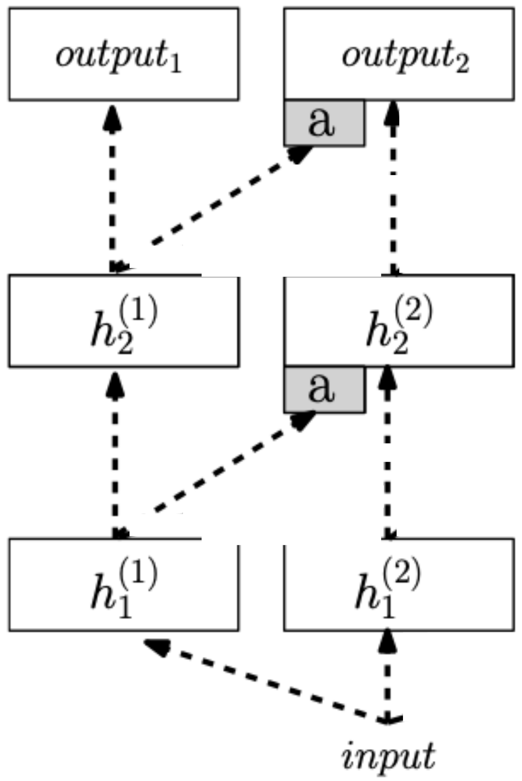
Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

Universität Bremen

Progressive Networks

- We connect the new column laterally to the old one, by introducing an additional set of weights

- For frozen params $\Theta^{(1)}$ and random to be learned column 2 parameters $\Theta^{(2)}$, hidden layer $h_i^{(2)}$ now receives input from both $h_{i-1}^{(2)}$ & $h_{i-1}^{(1)}$.



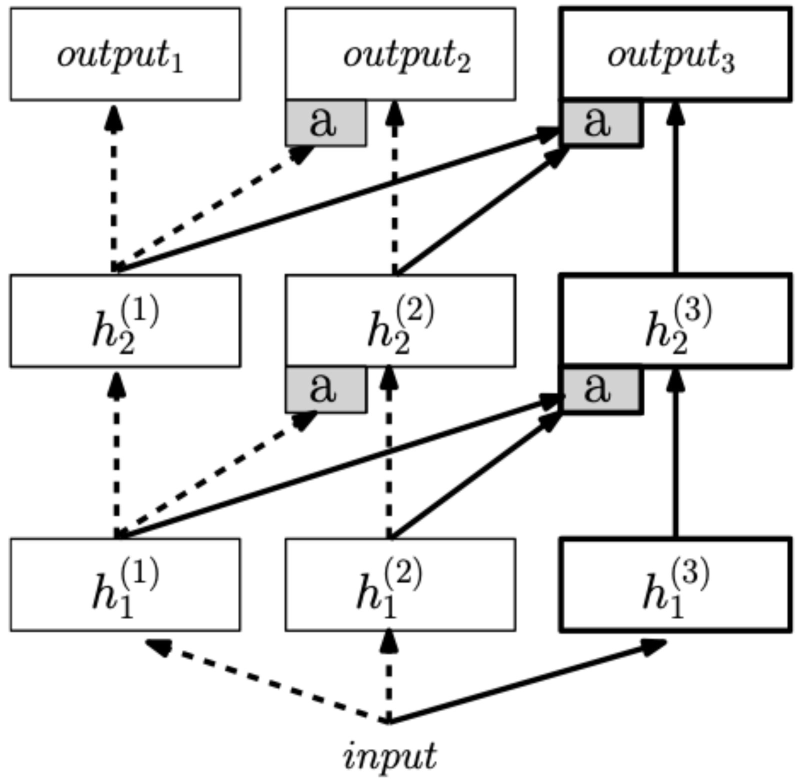
Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

Progressive Networks

- We connect the new column laterally to the old one, by introducing an additional set of weights

- For frozen params $\Theta^{(1)}$ and random to be learned column 2 parameters $\Theta^{(2)}$, hidden layer $h_i^{(2)}$ now receives input from both $h_{i-1}^{(2)}$ & $h_{i-1}^{(1)}$. And for k tasks:

$$h_i^{(k)} = f(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)})$$

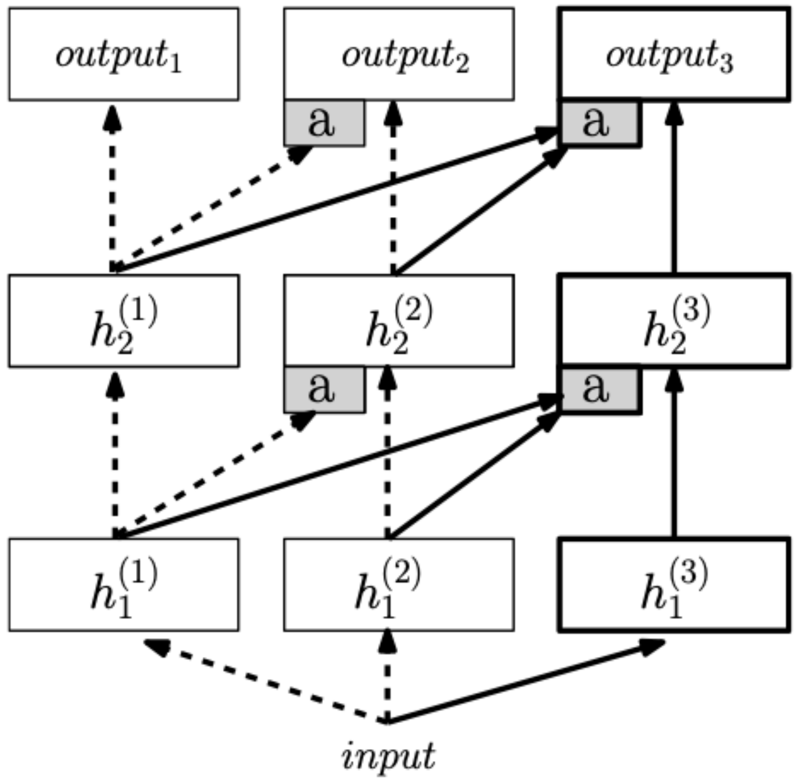


Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

Universität Bremen

Progressive Networks

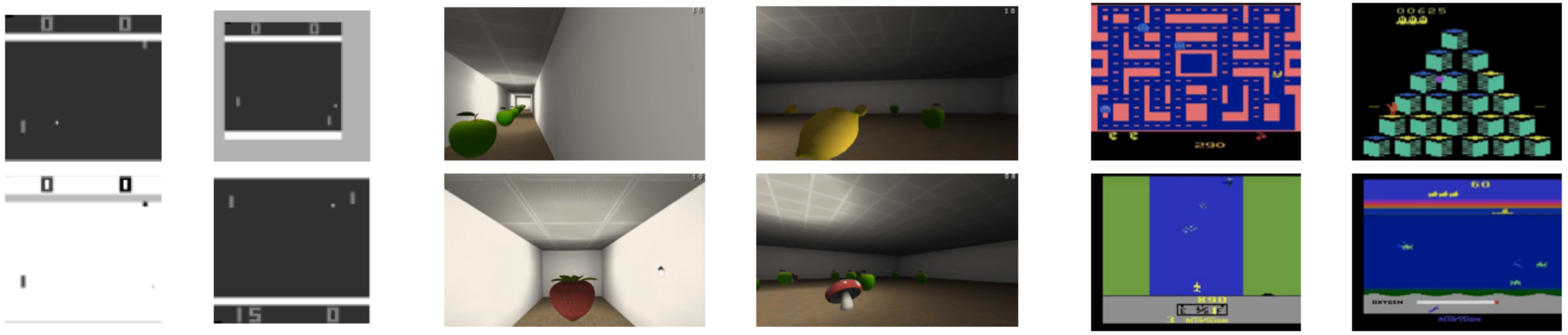
- And for k tasks:
$$h_i^{(k)} = f(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)})$$
- In practice, we can think of ProgNNs as a pre-cursor to LLM fine-tuning with adapters, if we include a projection matrix V & a learnable scalar α: e.g. feed through an MLP
$$h_i^{(k)} = \sigma(W_i^{(k)} h_{i-1}^{(k)} + U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_{i-1}^{(<k)} h_{i-1}^{(<k)}))$$
- This would then also allow to handle different inputs of different scales



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

Progressive NNs: empirical analysis

ProgNNs have been investigated on various reinforcement learning based scenarios initially (and probably on most everything now with more than 3k citations on the work)

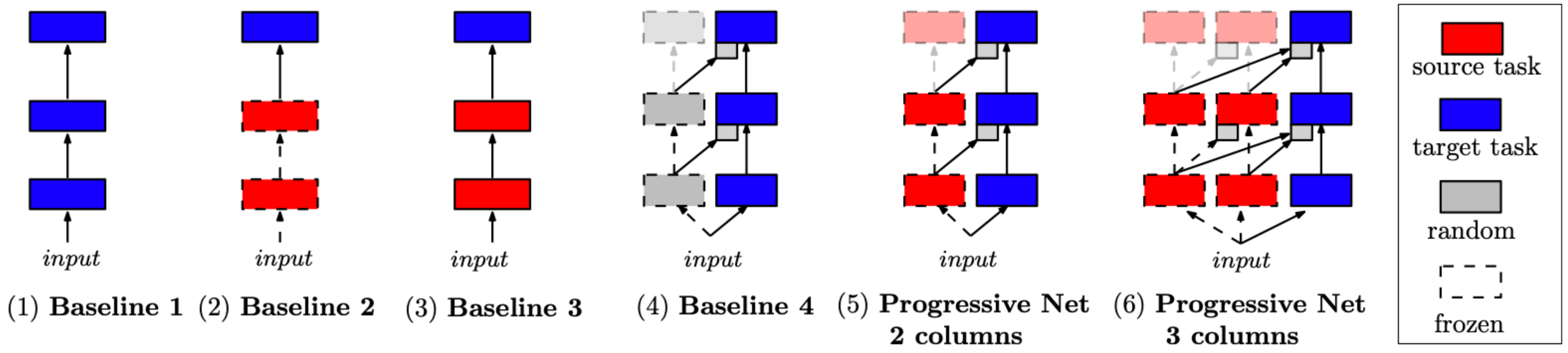

(a) Pong variants (b) Labyrinth games (c) Atari games

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

Progressive NNs: empirical analysis

Their two main goals are arguably: 1) avoid forgetting by freezing old columns and 2) accelerate training through selective transfer

The second aspect relates intuitively to our transfer learning tutorial!

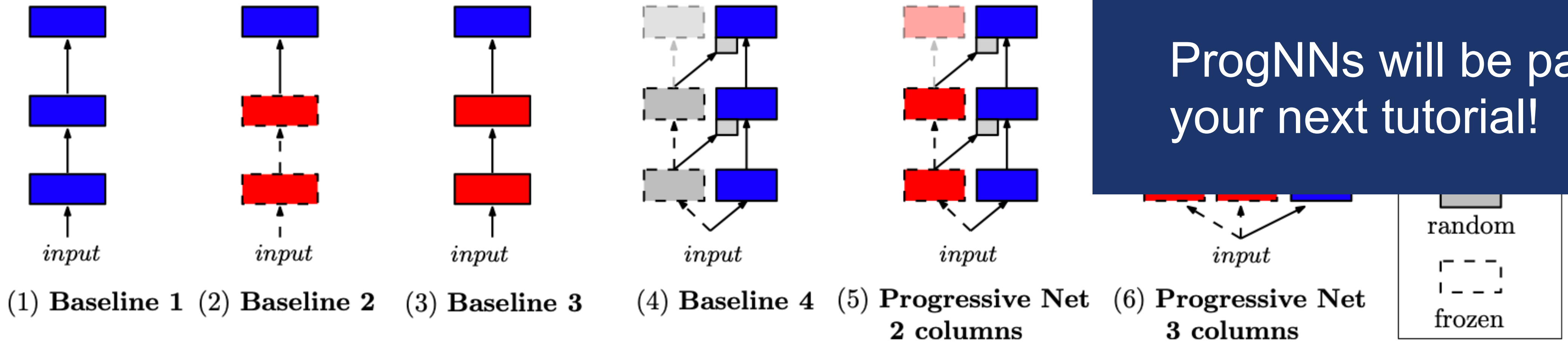


Rusu et al, “Progressive Neural Networks”, arXiv:1606.04671, 2016

Progressive NNs: empirical analysis

Their two main goals are arguably: 1) avoid forgetting by freezing old columns and 2) accelerate training through selective transfer

The second aspect relates intuitively to our transfer learning tutorial!



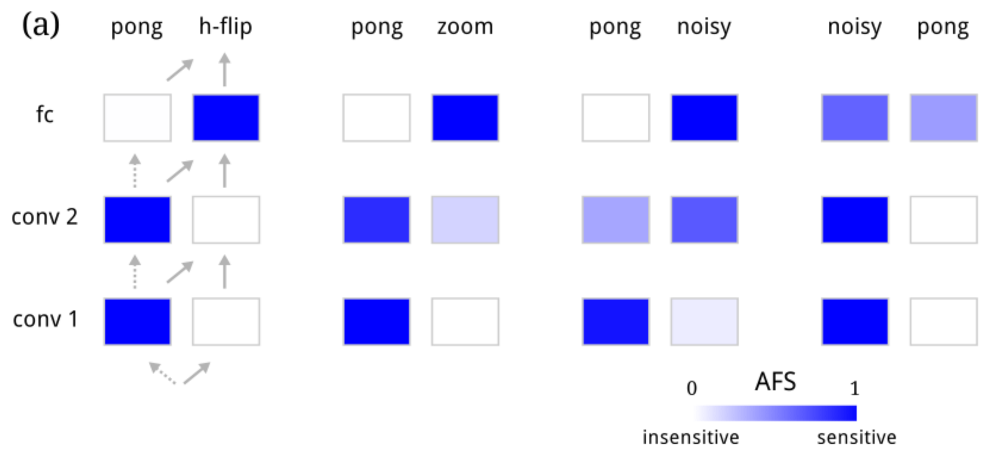
ProgNNs will be part of your next tutorial!

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

Progressive NNs: transfer sensitivity analysis

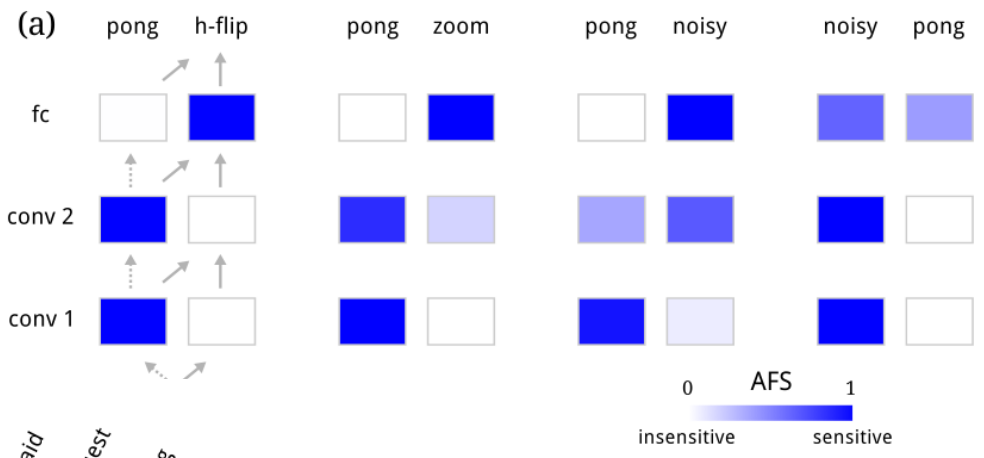
We can check which layers of which columns contribute to tasks, e.g. by looking at average fisher sensitivity

For two columns & distribution shift

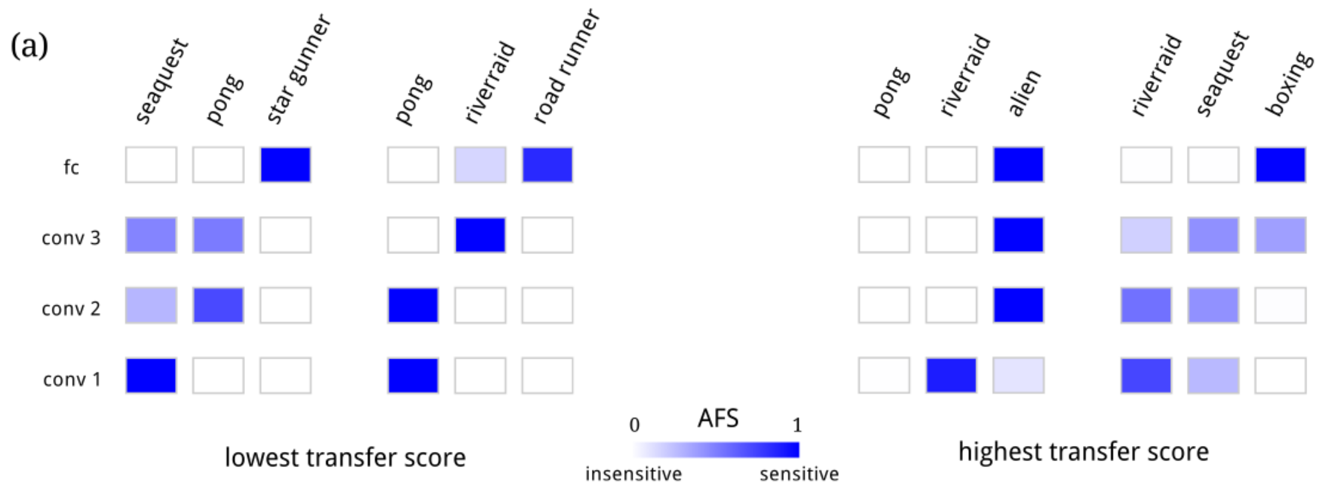


Progressive NNs: transfer sensitivity analysis

We can check which layers of which columns contribute to tasks, e.g. by looking at average fisher sensitivity



For two columns & distribution shift



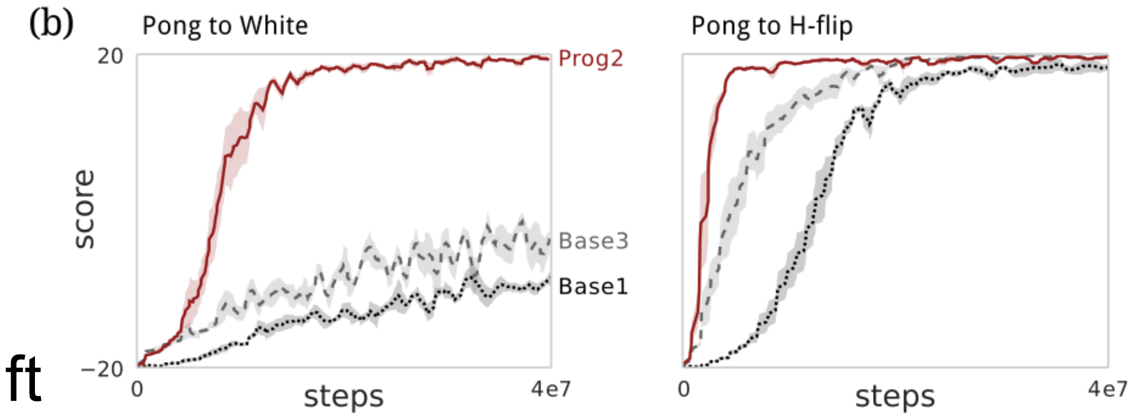
For three columns & different games

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

Progressive NNs: score & speed analysis

We can also check absolute score and speed of adaptation. Note: no forgetting due to freezing, but need to know task identities

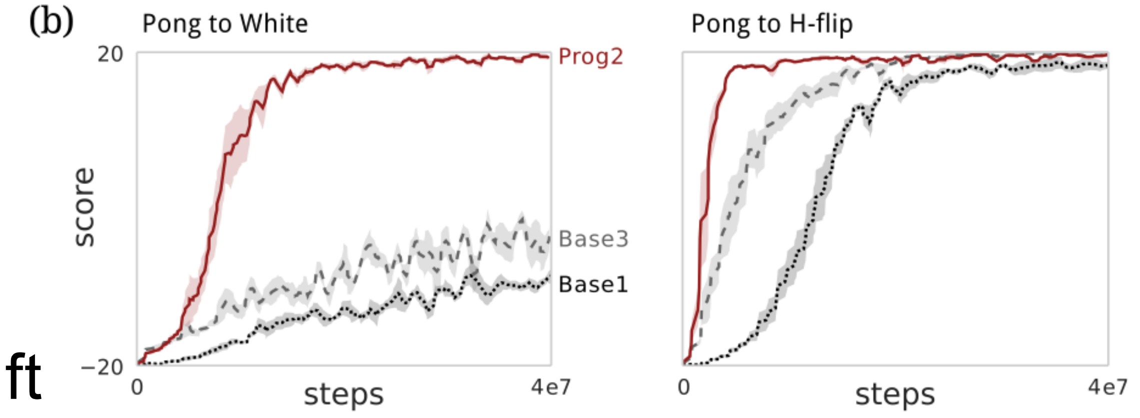
For two columns & distribution shift



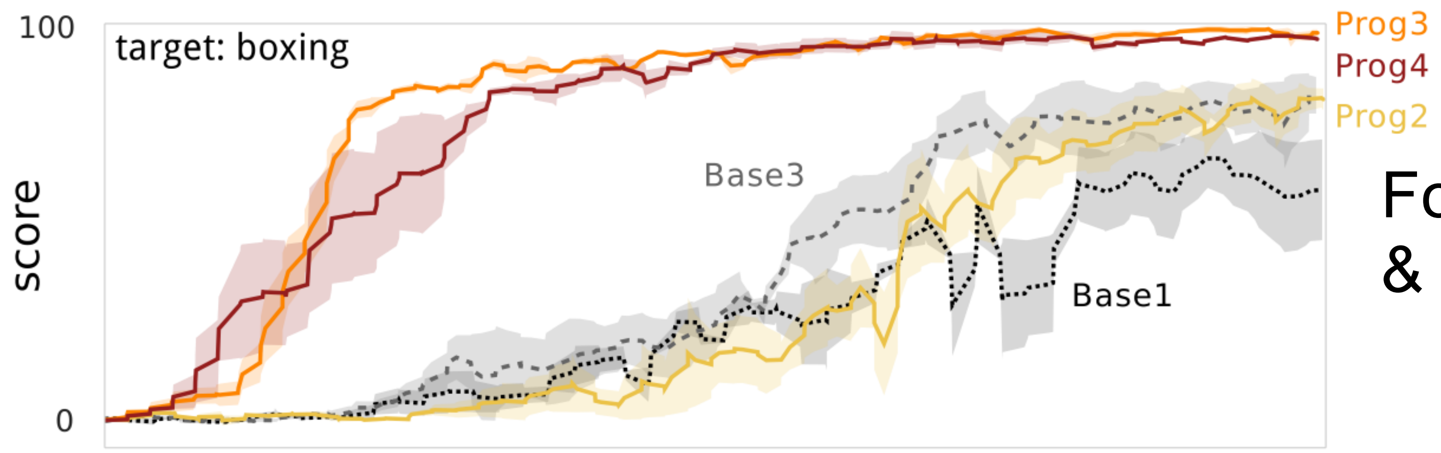
Universität Bremen

Progressive NNs: score & speed analysis

We can also check absolute score and speed of adaptation. Note: no forgetting due to freezing, but need to know task identities



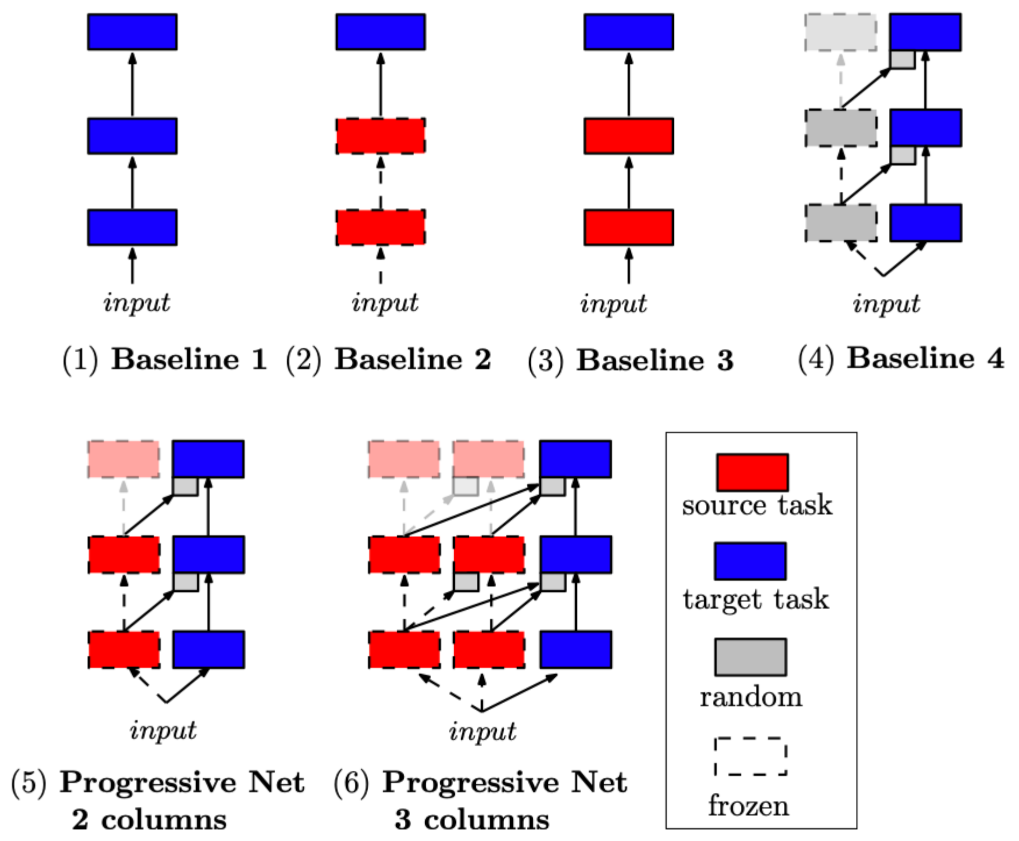
For two columns & distribution shift

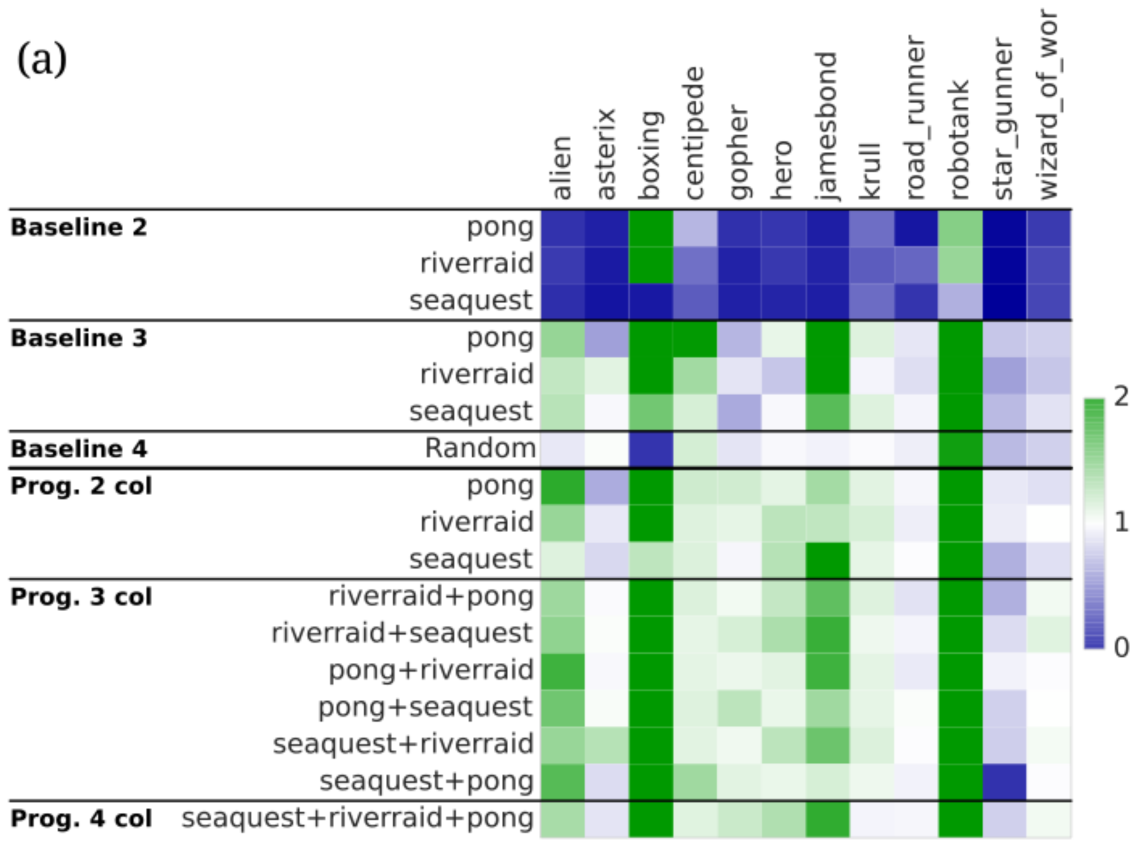


For three or four columns & different games

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

Progressive NNs: transfer analysis





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

Question time

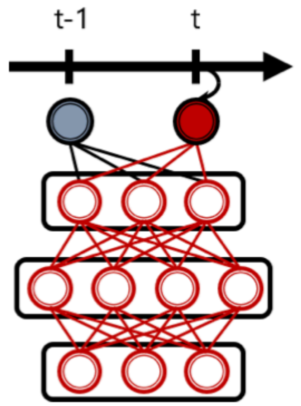
In contrast to our original dynamic node creation motivation, Progressive networks freeze columns. Can we leverage the connectivity of ProgNNs, yet allow for continued adaptation without forgetting?

Universität Bremen

Dynamically Expandable Nets

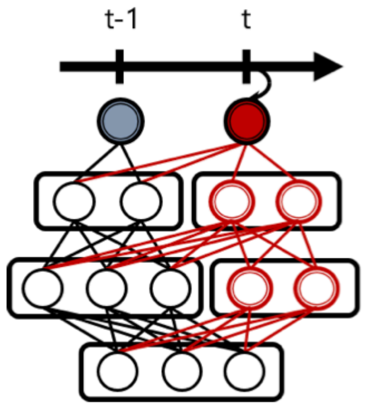
A “straightforward" idea (which is rather involved in DENN practice): combine EWC-style training with Progressive Networks

EWC



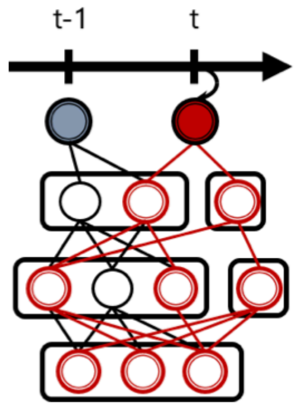
(a) Retraining w/o expansion

ProgNN



(b) No-retraining w/ expansion

DENN



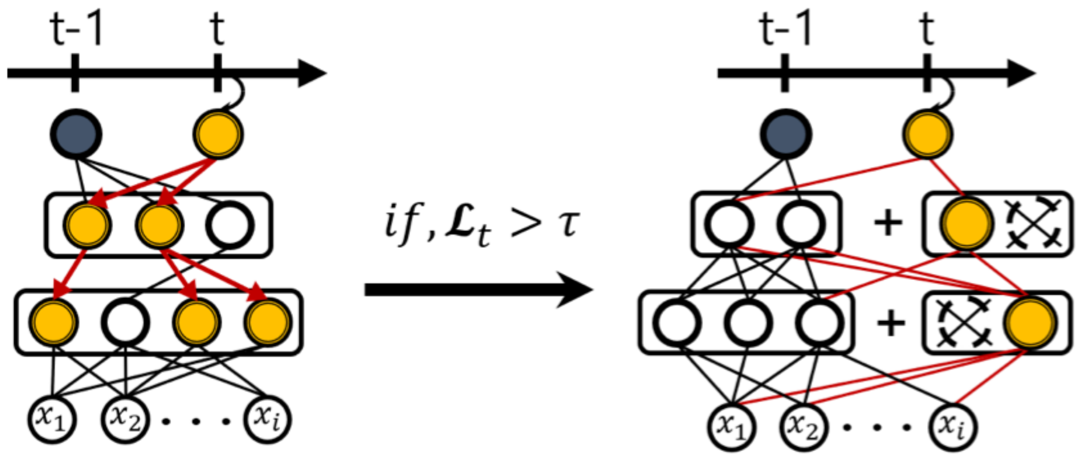
(c) Partial retraining w/ expansion

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

DENN: when to expand?

Step1: Continue selectively training on new task t, if the loss is below a set threshold, expand the network capacity

Step1.1: $\mathtt{min}_W \mathscr{L}(W^{t=1}; D_t) + \mu \sum_{l \in L} || W_l^{t=1} ||_1$ train: promote L1 sparsity

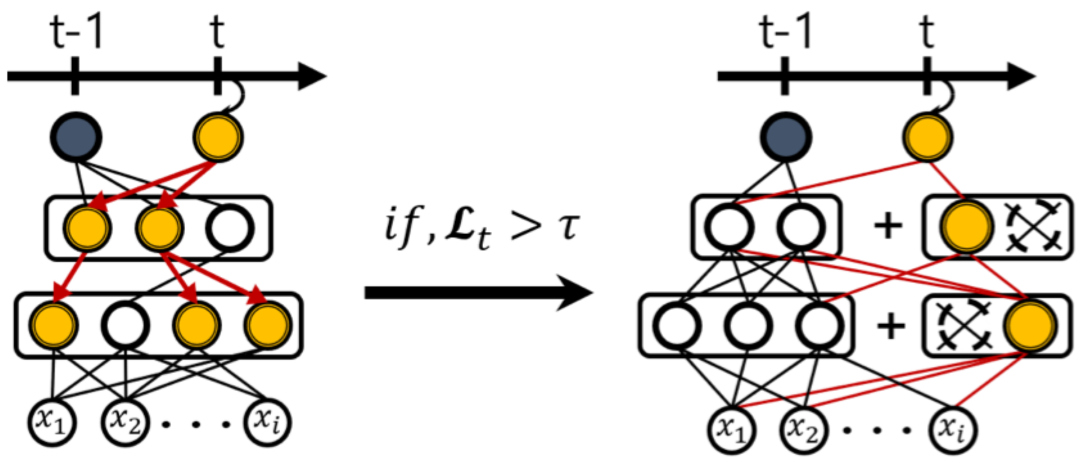


Yoon et al, “Lifelong Learning with Dynamically Expandable Networks”, ICLR 2018

DENN: when to expand?

Step1.2: Identify top connections through breath-first search to identify all relevant units that have paths to the task output, train with L2 regularizer

Step1.3: if selective training $\mathscr{L}_t > \tau$ add units in every layer by the philosophy of “add too much and remove again later”

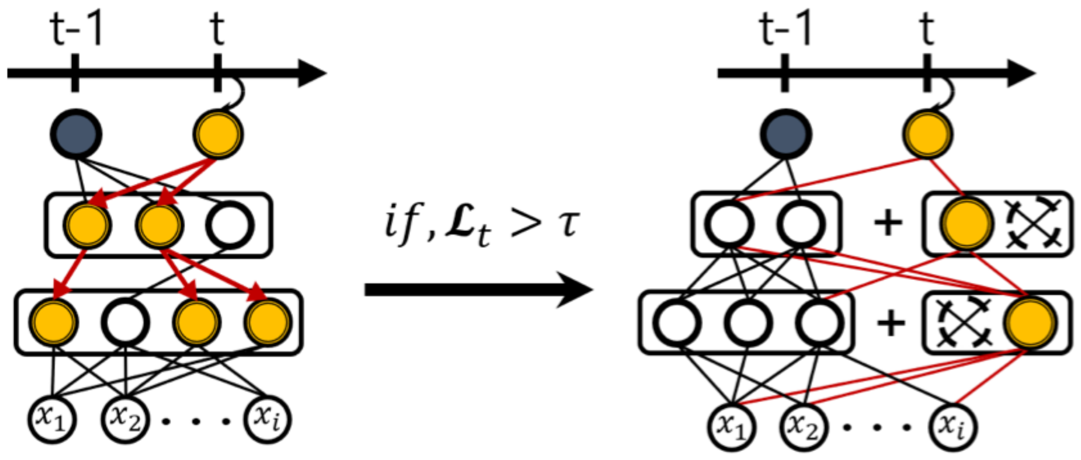


Yoon et al, “Lifelong Learning with Dynamically Expandable Networks”, ICLR 2018

DENN: how much to add?

Step2: Train with regularization on the newly added weights $W_l^{\mathcal{N}}$, then remove "useless" units

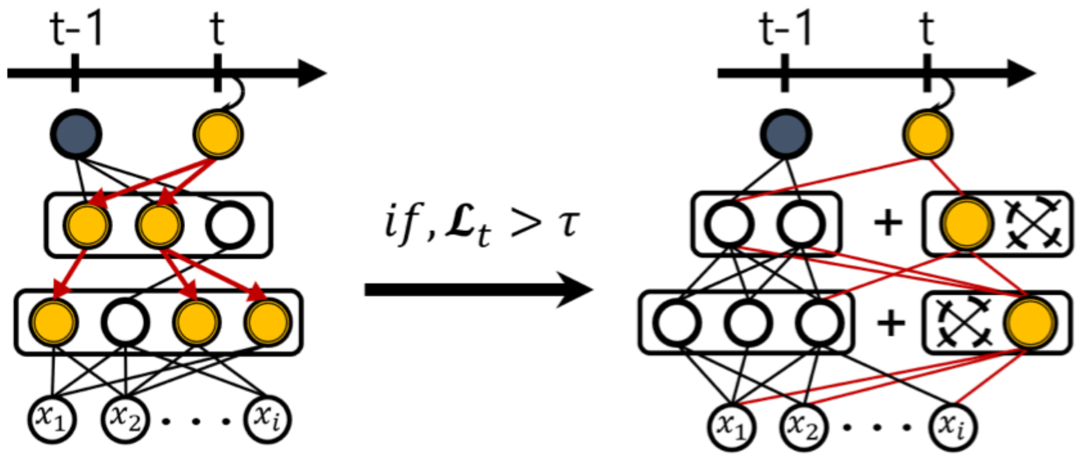
Step2.1: Add L1 (sparse) regularizer on new outgoing weights & L2 on incoming (g): $\mathtt{min}_W \mathscr{L}(W_l^{\mathcal{N}}; W_l^{t-1}, D_t) + \mu \| W_l^{\mathcal{N}} \|_1 + \gamma \sum_g \| W_{l,g}^{\mathcal{N}} \|_2$



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

DENN: how much to add?

Step2.2: Remove excessive (“useless”) units by some criterion, e.g. do not contribute to task, small weights, etc. (not specified in original DENN work)
Step2.3: In principle apply EWC, $\mathtt{min}_W \mathscr{L}(W^t; D_t) + \lambda || W^t - W^{t-1} ||_2^2$, but this can be challenging -> DENN measures “drift” to “copy” units



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

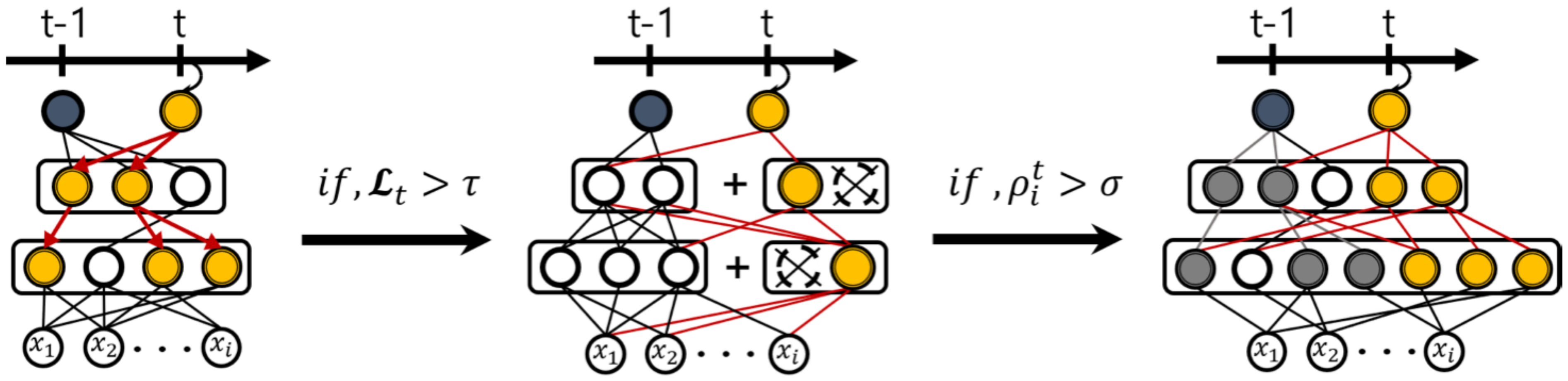
Universität Bremen

DENN: what to add?

Step3: If the difference in weights is “too large”, balancing stability/plasticity is tough. Instead, split neuron i into two copies and train.

Step3.1: Measure EWC-style “drift” $\rho_i^t = ||W_i^t - W_i^{t-1}||_2$

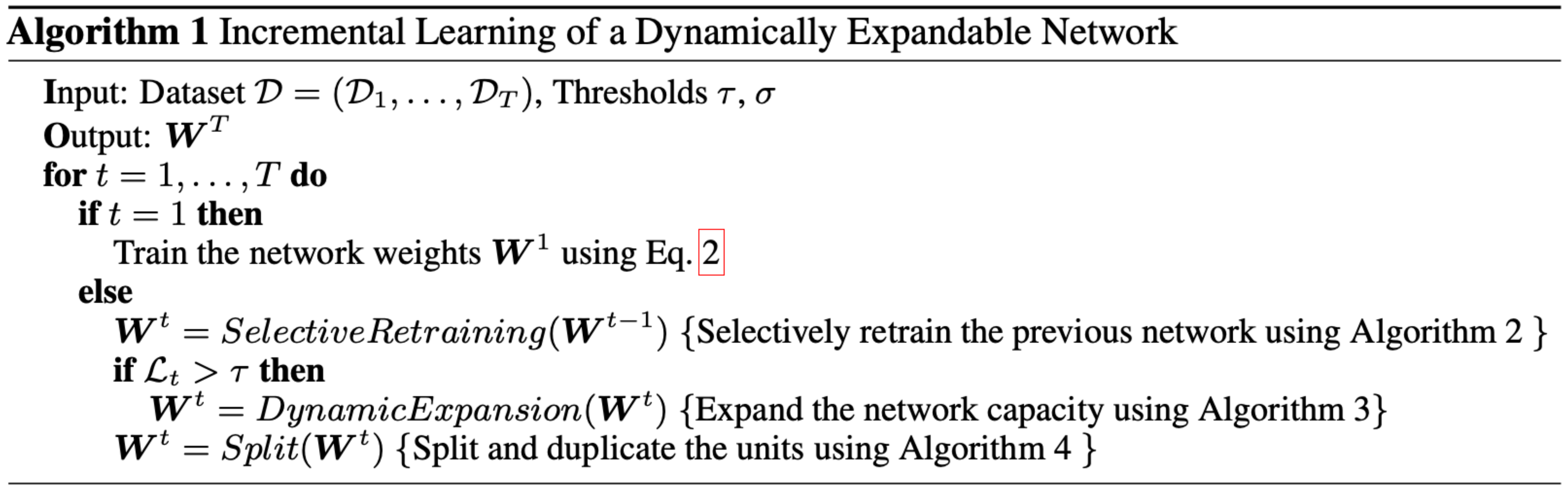
Step3.2: if $\rho_i^t > \sigma$ (some threshold), copy and introduce new lateral edges



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

DENN: in summary

Originally intuitive motivation, but many hyper-parameters/ad-hoc choices.
Sidelines almost all questions in terms of architecture growth (where, what..)



Algorithm 1 Incremental Learning of a Dynamically Expandable Network

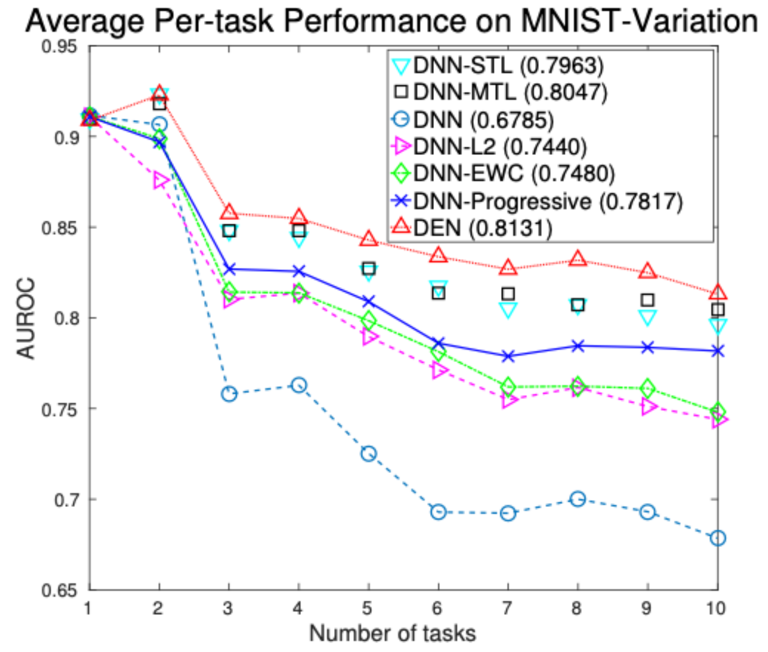
```
Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds $\tau$, $\sigma$
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ do
  if $t = 1$ then
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {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 }
```

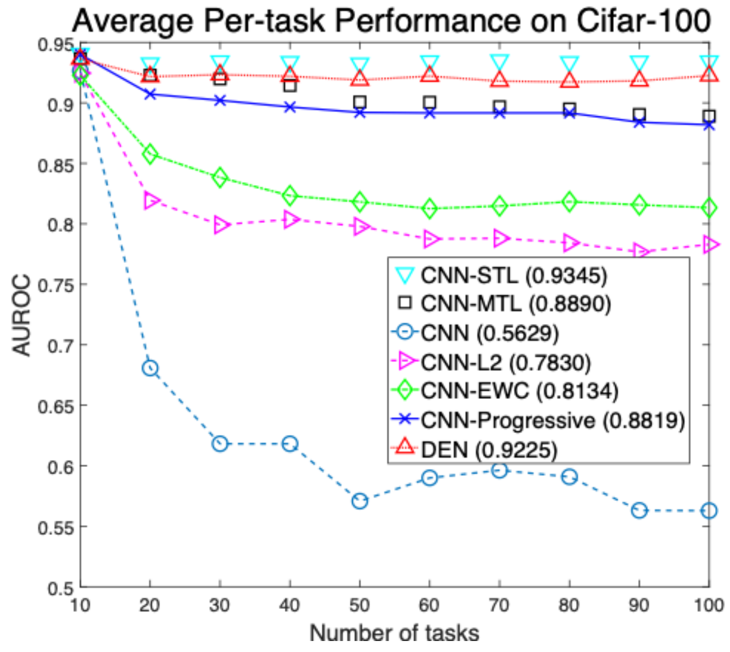
Yoon et al, “Lifelong Learning with Dynamically Expandable Networks”, ICLR 2018

Universität Bremen

DENN: in practice

Has been reported to perform well, but the exact implementation seems to have some reproducibility issues. However, the key idea is very meaningful

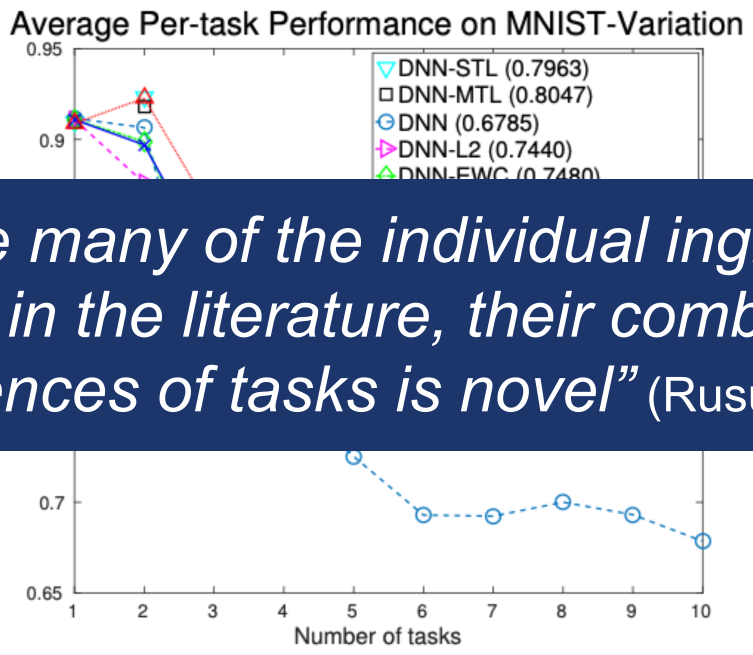


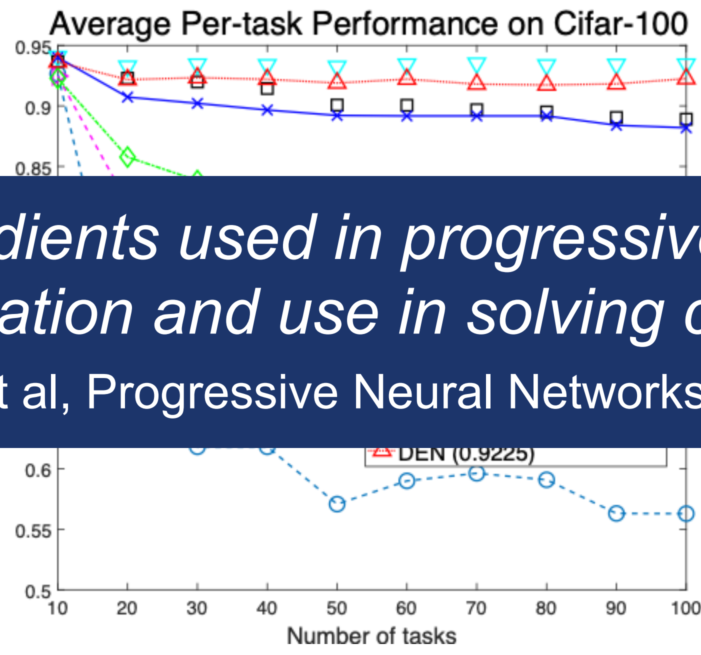


Yoon et al, “Lifelong Learning with Dynamically Expandable Networks”, ICLR 2018

DENN: in practice

Has been reported to perform well, but the exact implementation seems to have some reproducibility issues. However, the key idea is very meaningful





"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)

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

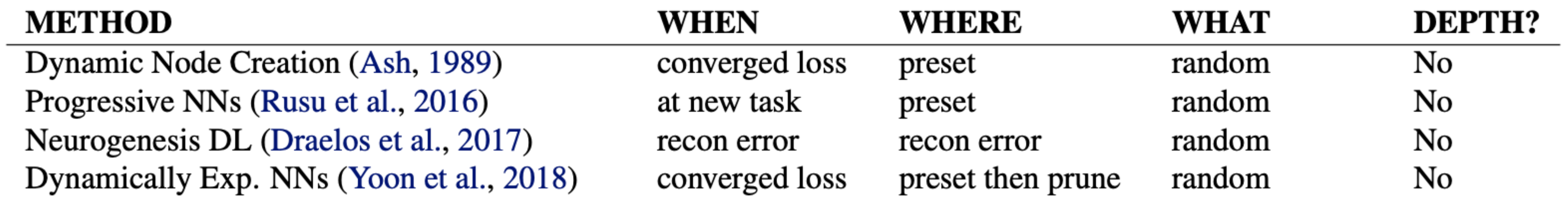
Question time

We still really have only seen works that tackle "when to add" - what about "where to add", "what to add". Do you have any ideas for approaches?

Universität Bremen

When, Where, What + Network Depth

We have seen that DNC, ProgNNs, DENNs (and Neurogenesis DL, which does something similar based on reconstruction error in an auto-encoder) all use presets or ignore some essential questions

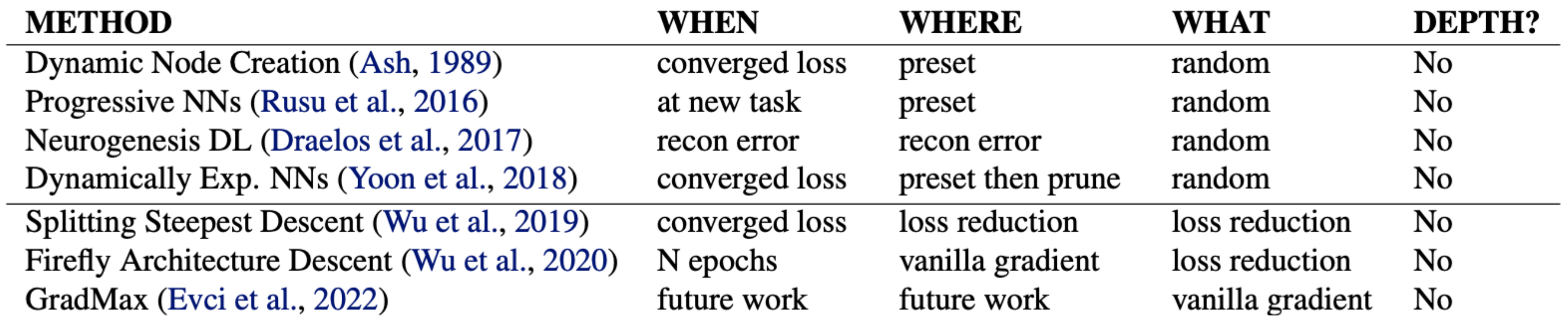


METHOD	WHEN	WHERE	WHAT	DEPTH?
Dynamic Node Creation (Ash, 1989)	converged loss	preset	random	No
Progressive NNs (Rusu et al., 2016)	at new task	preset	random	No
Neurogenesis DL (Draelos et al., 2017)	recon error	recon error	random	No
Dynamically Exp. NNs (Yoon et al., 2018)	converged loss	preset then prune	random	No

Table from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

When, Where, What + Network Depth

Some newer family of methods improve upon splitting of neurons (like in DENN) and think about the “what to add” in the sense of how to set weights/connections. Ideas revolve around weight eigenvalues, not zeroing connections & initializing outgoing weights with e.g. SVD



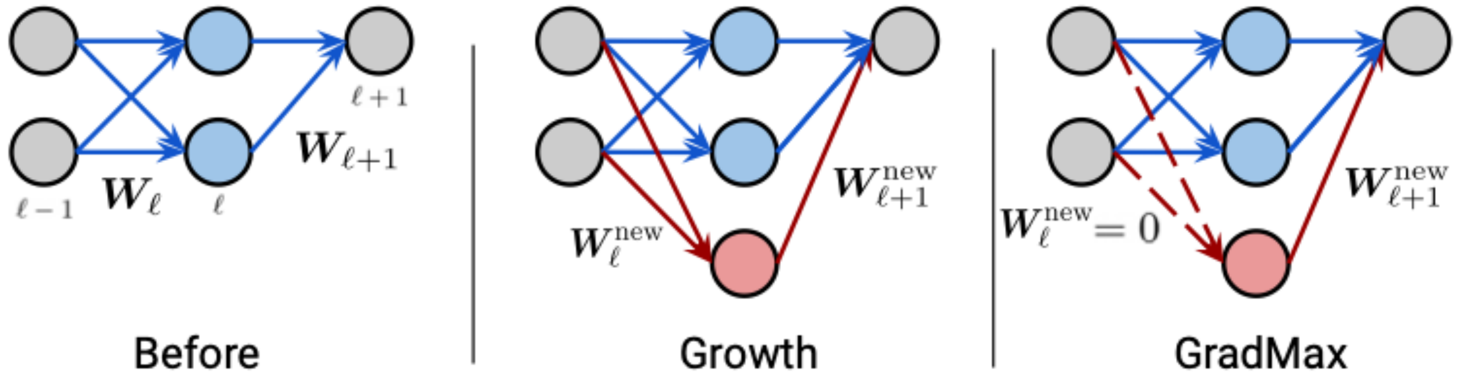
METHOD	WHEN	WHERE	WHAT	DEPTH?
Dynamic Node Creation (Ash, 1989)	converged loss	preset	random	No
Progressive NNs (Rusu et al., 2016)	at new task	preset	random	No
Neurogenesis DL (Draelos et al., 2017)	recon error	recon error	random	No
Dynamically Exp. NNs (Yoon et al., 2018)	converged loss	preset then prune	random	No
Splitting Steepest Descent (Wu et al., 2019)	converged loss	loss reduction	loss reduction	No
Firefly Architecture Descent (Wu et al., 2020)	N epochs	vanilla gradient	loss reduction	No
GradMax (Evci et al., 2022)	future work	future work	vanilla gradient	No

Table from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität
Bremen

Example from GradMax

*“We mainly focus on the question of **how** and introduce a new initialization method for the new neurons… we keep the growing schedule (**where** and **when**) fixed”*



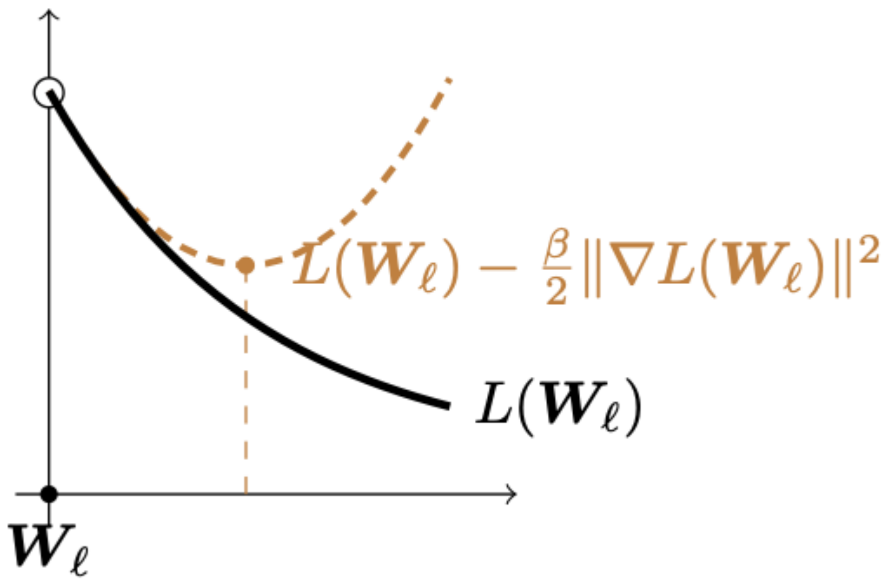
Follows the same intuition as before: set incoming weights to zero to keep output unchanged, but initialize outgoing weights using SVD!

Evci et al, “GradMax: Growing Neural Networks Using Gradient Information”, ICLR, 2022

Example from GradMax

Intuition - in the name ”Max Gradient”:

- Larger gradients lead to large objective decrease
- Prior theory (Nesterov 2003) has tied decrease in objective to an upper-bound that increases as the norm of the gradient increases
- GradMax adds new units in a step that maximizes the gradient norm


$L(\boldsymbol{W}_\ell) - \frac{\beta}{2}\|\nabla L(\boldsymbol{W}_\ell)\|^2$
$L(\boldsymbol{W}_\ell)$
$\boldsymbol{W}_\ell$

Example from GradMax

- Solving the general problem to leave gradients of existing weights unchanged and maximize gradients on the new weights is hard to solve:

$$\texttt{argmax}_{W^{new}} ||\mathbb{E}_D[\frac{\delta L}{\delta W_l^{new}}]||_F^2 + ||\mathbb{E}_D[\frac{\delta L}{\delta W_{l+1}^{new}}]||_F^2$$

$$\text{s.t. } ||W_l^{new}||_F, ||W_{l+1}^{new}||_F \leq c \ \& \ W_{l+1}^{new} h_l^{new} = 0$$

Example from GradMax

- Solving the general problem to leave gradients of existing weights unchanged and maximize gradients on the new weights is hard to solve:

$$\texttt{argmax}_{W^{new}} ||\mathbb{E}_D[\frac{\delta L}{\delta W_l^{new}}]||_F^2 + ||\mathbb{E}_D[\frac{\delta L}{\delta W_{l+1}^{new}}]||_F^2$$

s.t. $||W_l^{new}||_F, ||W_{l+1}^{new}||_F \leq c \;\&\; W_{l+1}^{new} h_l^{new} = 0$

- Important: weights need to be constrained to avoid trivial solution of having the gradient norm go towards infinity
- GradMax introduces an approximate solution based on singular value decomposition (SVD)

Evci et al, “GradMax: Growing Neural Networks Using Gradient Information”, ICLR, 2022

Example from GradMax

- Recall (fully connected layer "pre-activations" and "activations"):
  $z_l = W_l h_{l-1}$ and $h_l = f(z_l)$

- We can derive gradients of the new weights using the chain rule:

$$\frac{\delta L}{\delta W_l^{new}} = (f'(z_l^{new}) \odot W_{l+1}^{new,T} \frac{\delta L}{\delta z_{l+1}}) h_{l-1}^T$$

$$\frac{\delta L}{\delta W_{l+1}^{new}} = \frac{\delta L}{\delta z_{l+1}} h_l^{new,T}$$

Universität Bremen

Example from GradMax

- Recall (fully connected layer “pre-activations” and “activations”): $z_l = W_l h_{l-1}$ and $h_l = f(z_l)$

- We can derive gradients of the new weights using the chain rule:
$$\frac{\delta L}{\delta W_l^{new}} = (f'(z_l^{new}) \odot W_{l+1}^{new,T} \frac{\delta L}{\delta z_{l+1}}) h_{l-1}^T$$
$$\frac{\delta L}{\delta W_{l+1}^{new}} = \frac{\delta L}{\delta z_{l+1}} h_l^{new,T}$$
- Simplifying assumption is to set $W_l^{new} = 0$ and $f(0) = 0$ with $f'(0) = 1$

Evci et al, “GradMax: Growing Neural Networks Using Gradient Information”, ICLR, 2022

Example from GradMax

- Simplifies gradients to $\frac{\delta L}{\delta W_l^{new}} = W_{l+1}^{new,T} \frac{\delta L}{\delta z_{l+1}} h_{l-1}^T$ and $\frac{\delta L}{\delta W_{l+1}^{new}} = 0$

Example from GradMax

- Simplifies gradients to $\frac{\delta L}{\delta W_l^{new}} = W_{l+1}^{new,T} \frac{\delta L}{\delta z_{l+1}} h_{l-1}^T$ and $\frac{\delta L}{\delta W_{l+1}^{new}} = 0$

- And reduces initial problem to $\mathrm{argmax}_{W_{l=1}^{new}} || W_{l=1}^{new,T} \mathbb{E}_D[\frac{\delta L}{\delta z_{l+1}} h_{l-1}^T] ||_F^2$

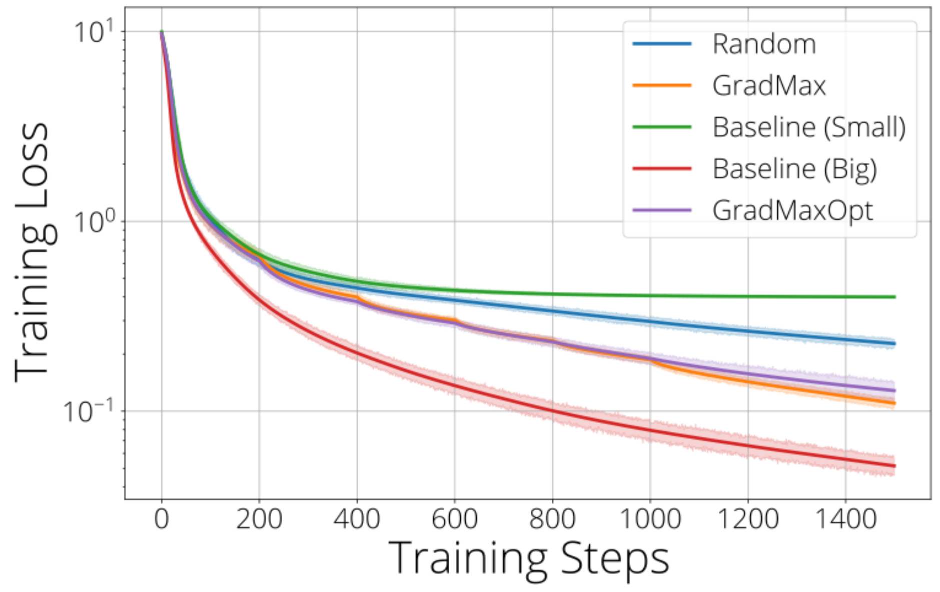
  s.t. $|| W_{l+1}^{new} ||_F \leq c$

- Solution to this maximization problem is by setting columns of W_{l+1}^{new} as top k singular vectors of the matrix $\mathbb{E}_D[\frac{\delta L}{\delta z_{l+1}} h_{l-1}^T]$

Evci et al, “GradMax: Growing Neural Networks Using Gradient Information”, ICLR, 2022

Example from GradMax

- GradMax init is better than random



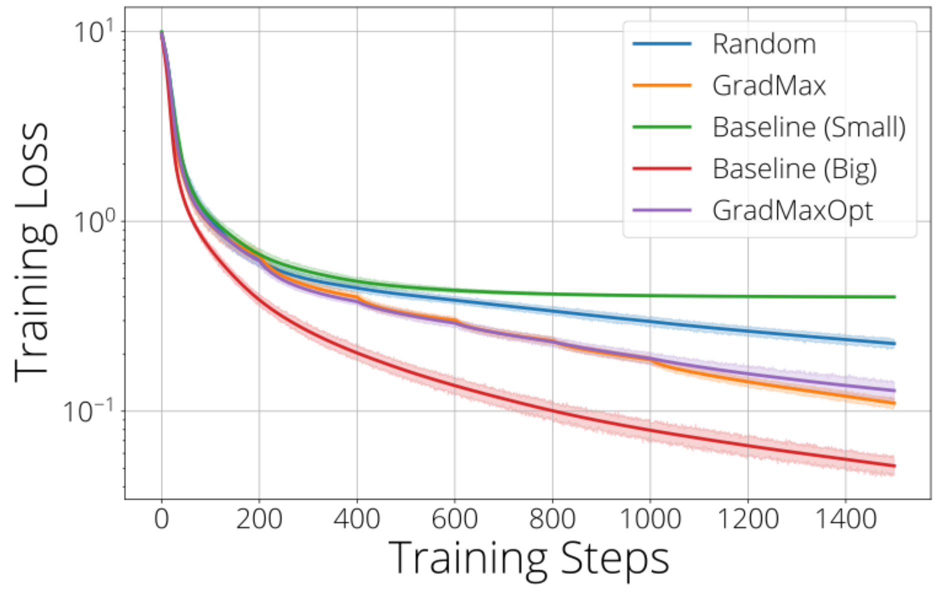
Evci et al, “GradMax: Growing Neural Networks Using Gradient Information”, ICLR, 2022

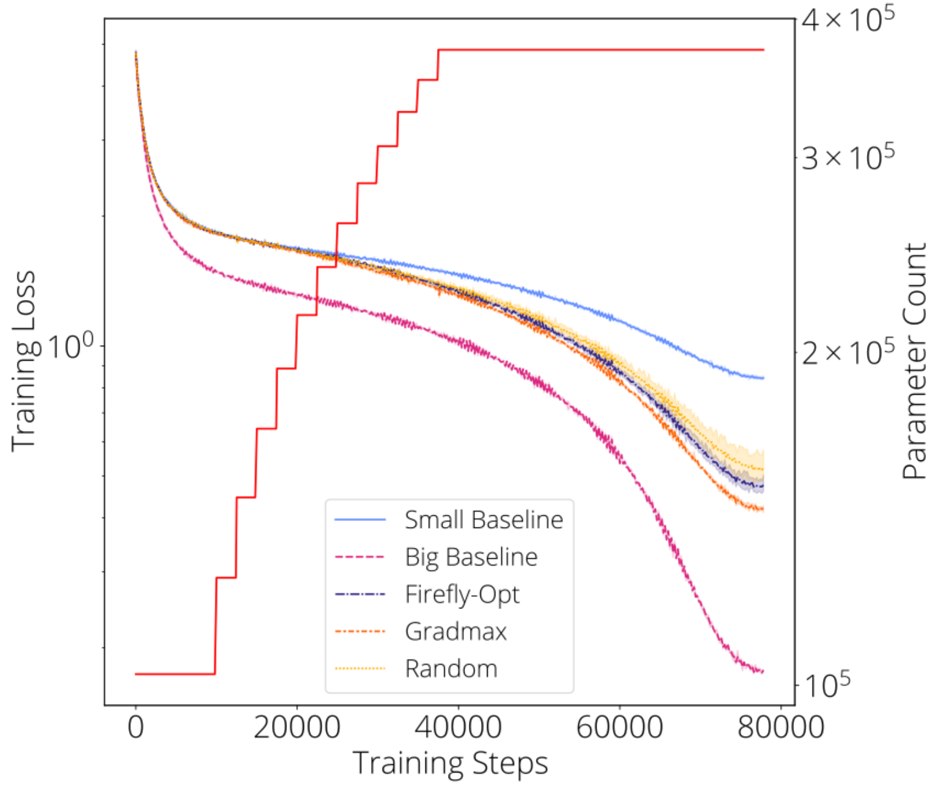
Universität Bremen

OWL ML

Example from GradMax

- GradMax init is better than random
- But we don't answer all questions, so "big baseline" is still much better





(c) Wide-Resnet-28-1 / CIFAR-100

Evci et al, "GradMax: Growing Neural Networks Using Gradient Information", ICLR, 2022

Question time

Why don't any of these works seem to add layers?
(or systematically answer the other questions?)

When, Where, What + Network Depth

Adding layers is hard because we need to avoid perturbing the learned function.

This is significantly more tricky than zeroing connections due to invertibility constraints and non-linear activations.

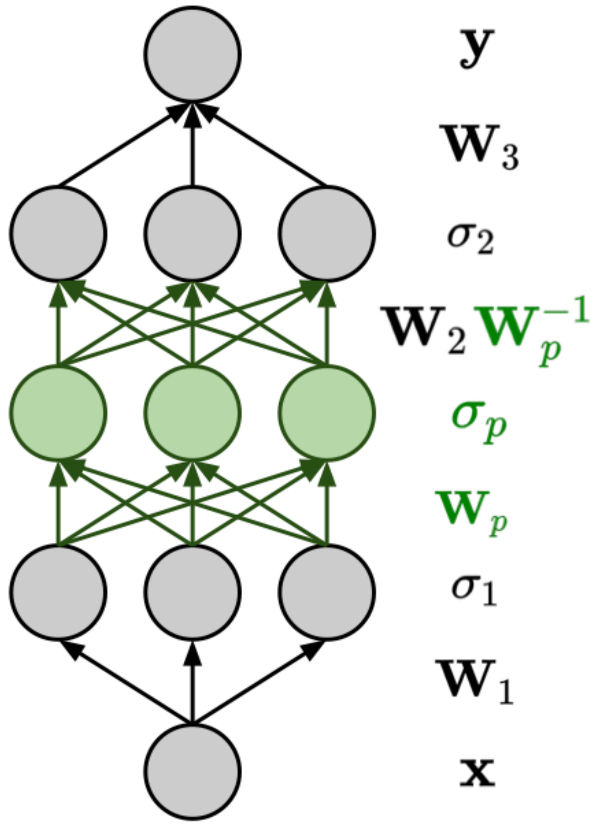


Image from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

When, Where, What + Network Depth

Adding layers is hard because we need to avoid perturbing the learned function.

This is significantly more tricky than zeroing connections due to invertibility constraints and non-linear activations.

Our addition needs to represent the identity, replacing e.g. W_2 with some $(W_2 W_p^{-1})(\sigma_p = \mathbb{I})W_q$ with appropriate σ

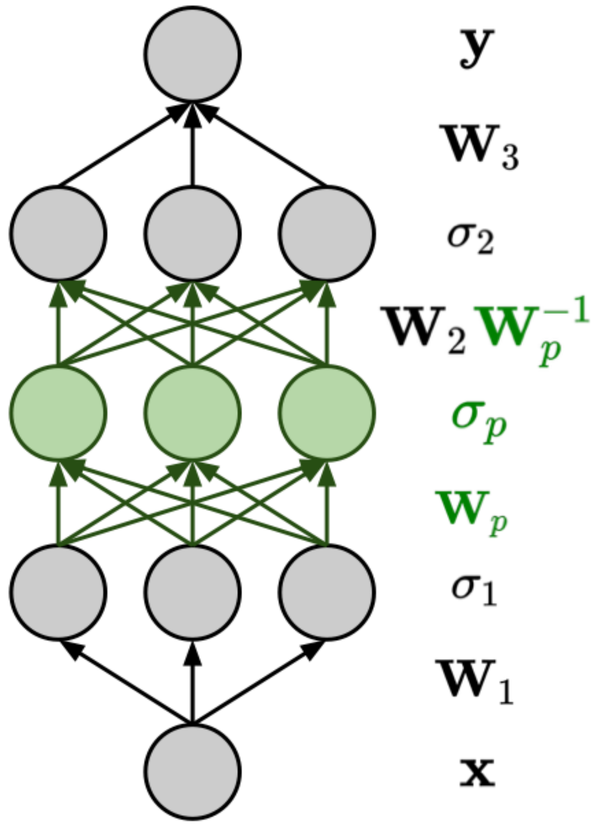


Image from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

When, Where, What + Network Depth

Example: rational activation functions (Molina et al, ICLR 2020)

$$\sigma_\theta(x) = \alpha x + \frac{\beta = \gamma x}{1 + x^2}$$

Setting theta to 1,0,0 -> identity

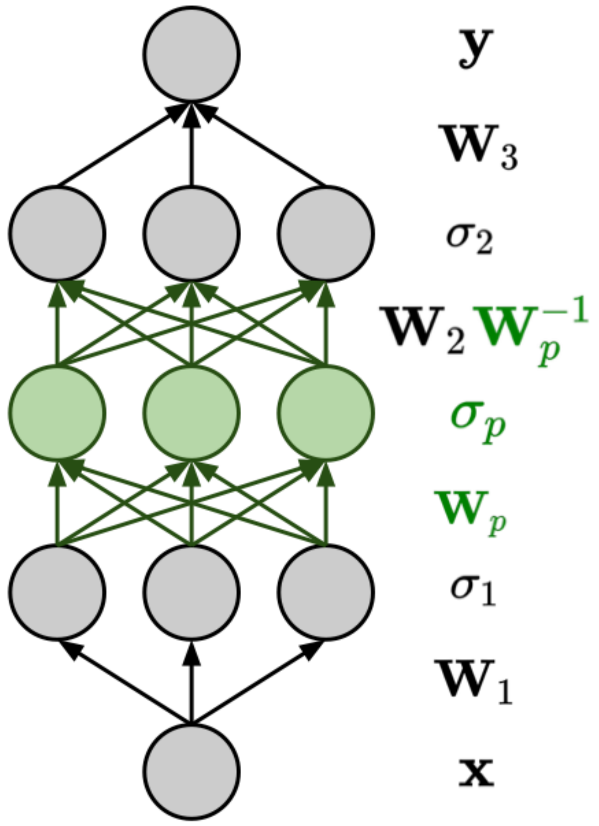


Image from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

When, Where, What + Network Depth

Example: rational activation functions (Molina et al, ICLR 2020)

$$\sigma_\theta(x) = \alpha x + \frac{\beta = \gamma x}{1 + x^2}$$

Setting theta to 1,0,0 -> identity

In convolutional neural networks, we can lift this constraint by working with skip connections

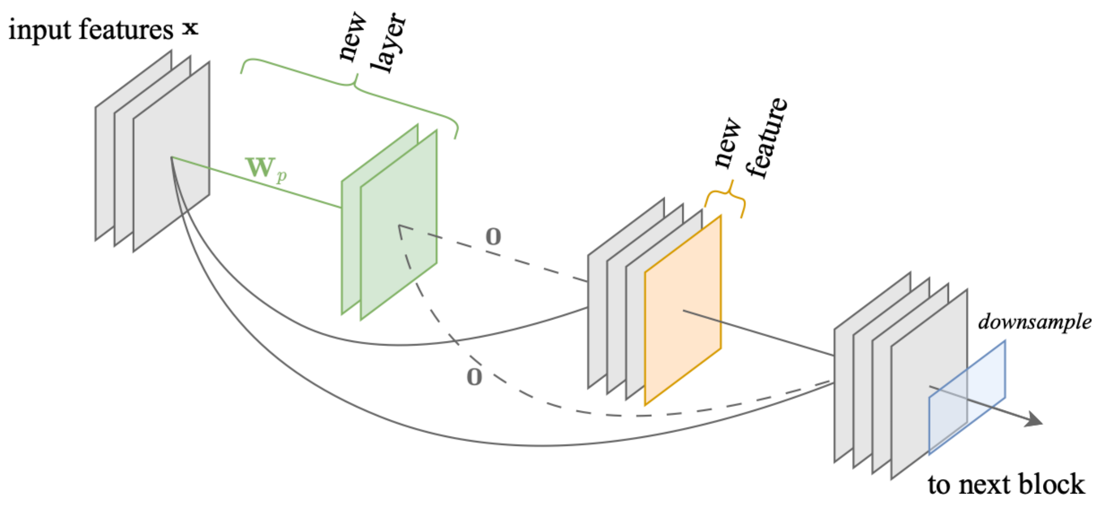
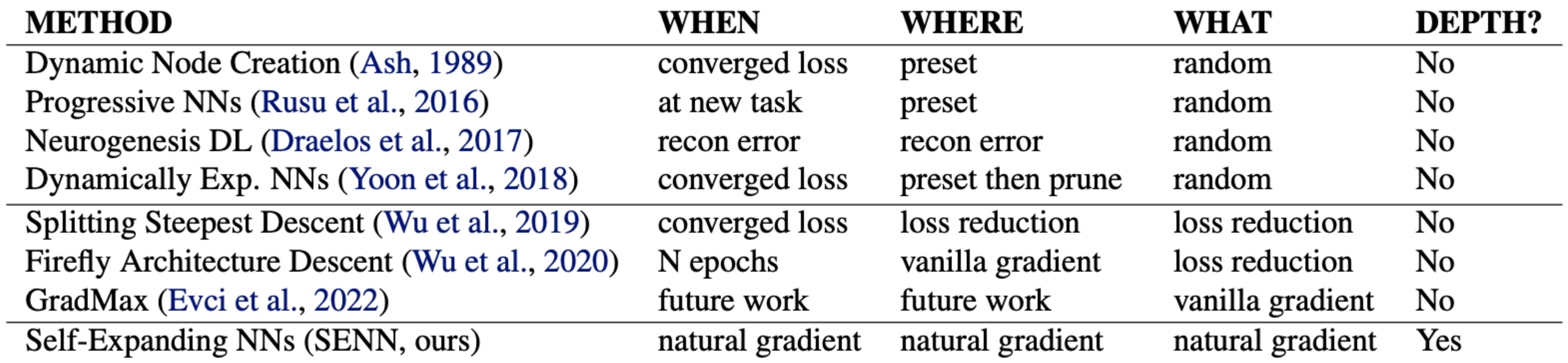


Image from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

When, Where, What + Network Depth

Additions require us to think about perturbations to the function!
And finally, answering “when, where, what” requires us to operate in function space -> “natural gradients”



METHOD	WHEN	WHERE	WHAT	DEPTH?
Dynamic Node Creation (Ash, 1989)	converged loss	preset	random	No
Progressive NNs (Rusu et al., 2016)	at new task	preset	random	No
Neurogenesis DL (Draelos et al., 2017)	recon error	recon error	random	No
Dynamically Exp. NNs (Yoon et al., 2018)	converged loss	preset then prune	random	No
Splitting Steepest Descent (Wu et al., 2019)	converged loss	loss reduction	loss reduction	No
Firefly Architecture Descent (Wu et al., 2020)	N epochs	vanilla gradient	loss reduction	No
GradMax (Evci et al., 2022)	future work	future work	vanilla gradient	No
Self-Expanding NNs (SENN, ours)	natural gradient	natural gradient	natural gradient	Yes

Table from Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Self-Expanding Neural Networks

Intuitively: Consider the set of all possible output values of the parameter space under the Jacobian, which gives achievable directions of change in function space

Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

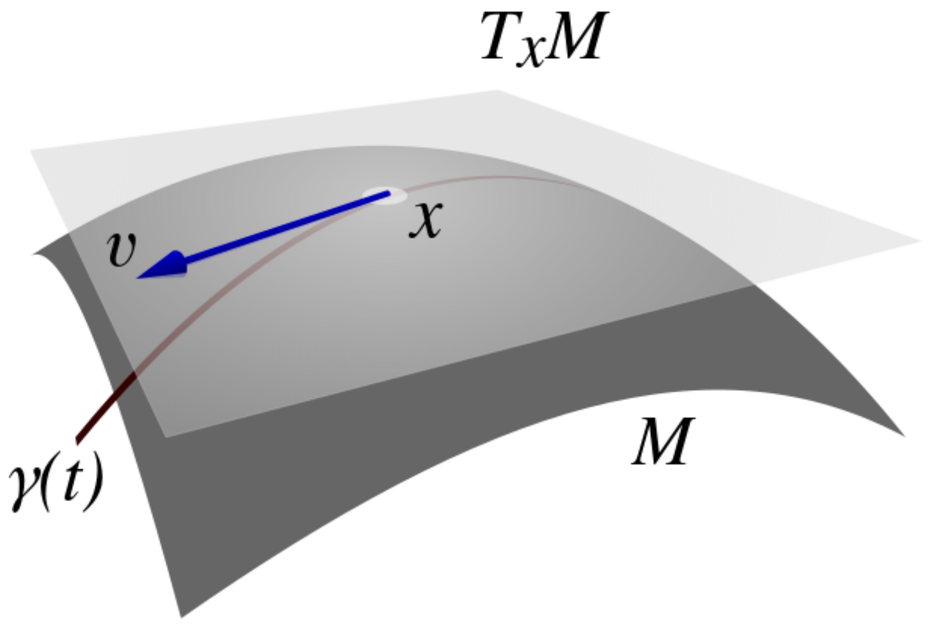
Self-Expanding Neural Networks

Intuitively: Consider the set of all possible output values of the parameter space under the Jacobian, which gives achievable directions of change in function space

Natural gradients give projection of regular gradient onto this subspace

We can maximize "alignment" of these achievable directions with the gradient:

$$\eta = g^T F^{-1} g$$



Universität Bremen

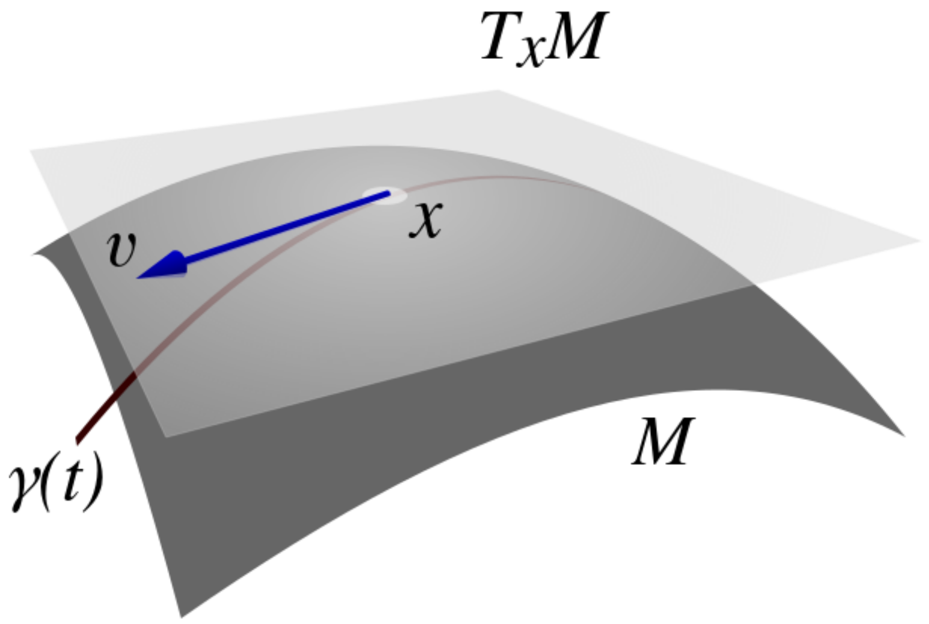
Self-Expanding Neural Networks

We can maximize “alignment” of these achievable directions with the gradient:

$$\eta = g^T F^{-1} g$$

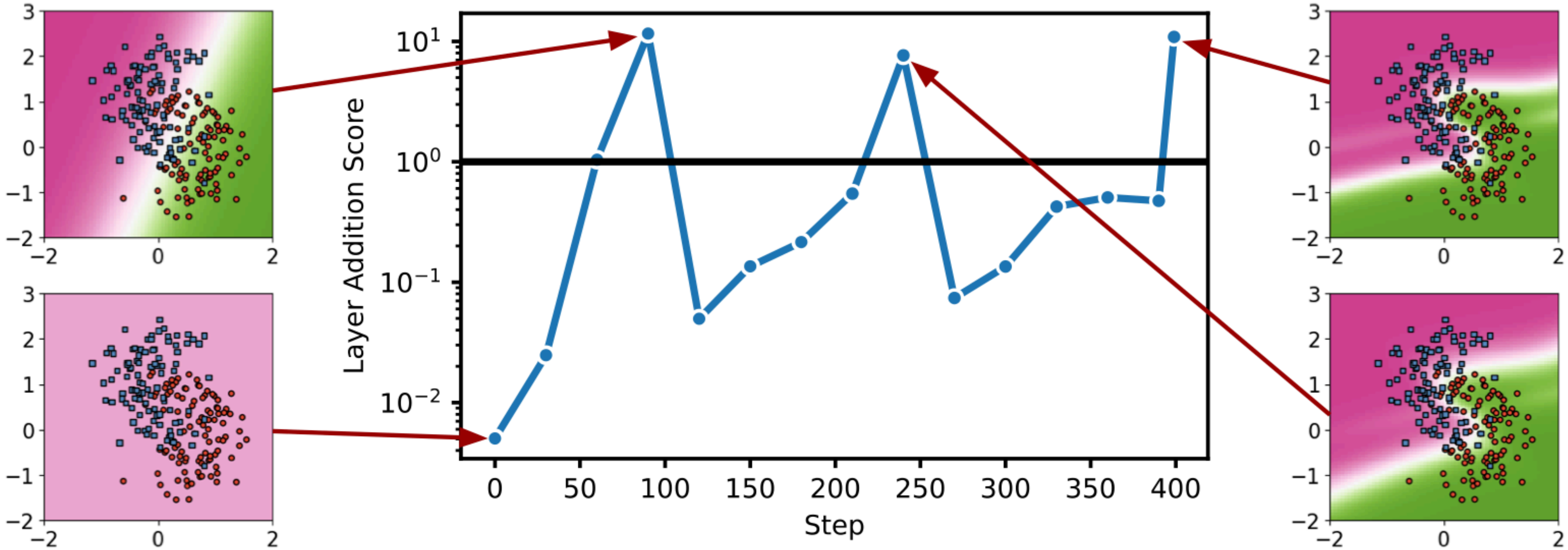
Intuitively “vanilla” vs. “natural”:

- Vanilla: euclidean norm of gradient measures rate of loss reduction. Adding redundant neuron copies increases score
- Natural: measures “size” of gradient in function space. Redundancy implies exact same semantics in function space



Universität
Bremen

SENN in practice: binary classification

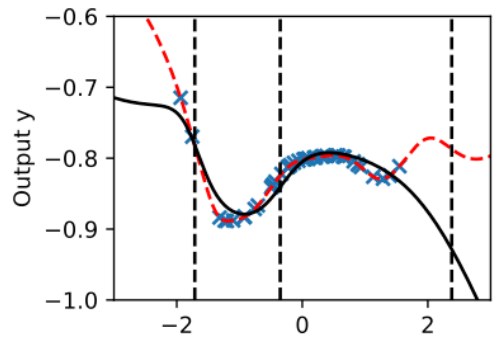


Score tells us when adding layers is beneficial, here from 0-2 MLP layers

Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

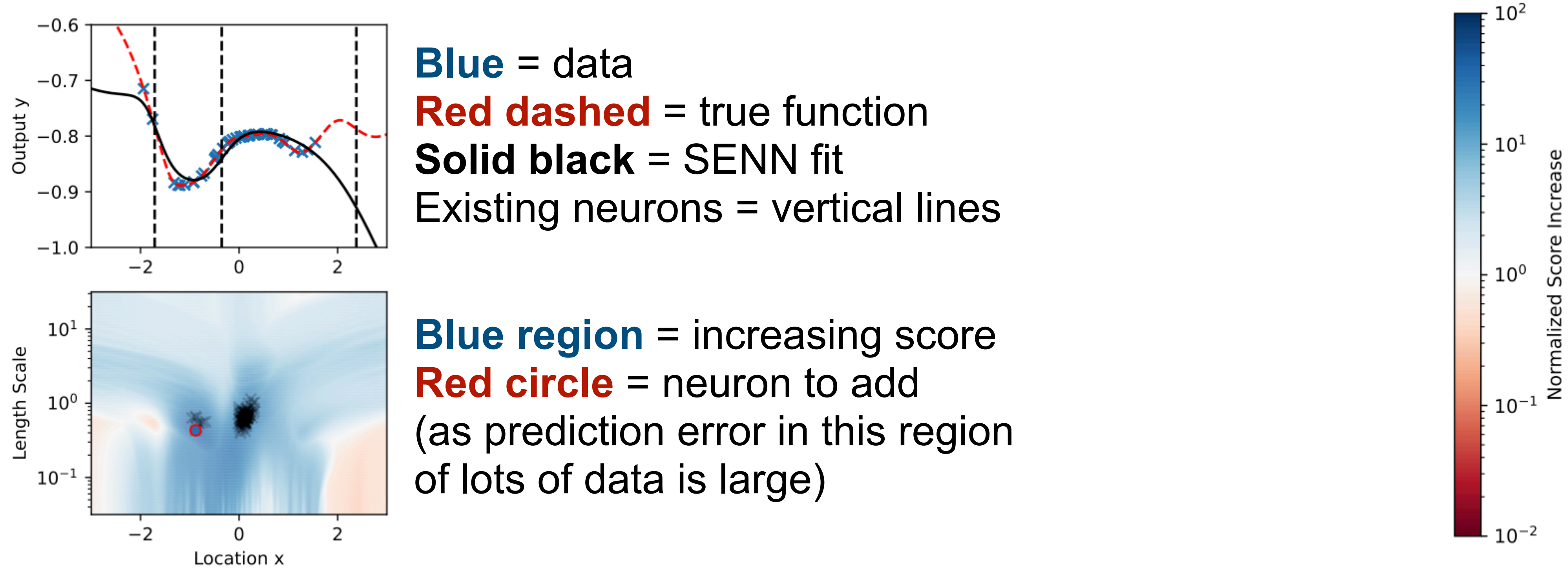
SENN in practice: least squares regression



Consider: single layer least squares regression, where we'll add best neurons for each basis function location & scale

Mitchell et al, "Self-Expanding Neural Networks", arXiv:2307.04526, 2024

SENN in practice: least squares regression



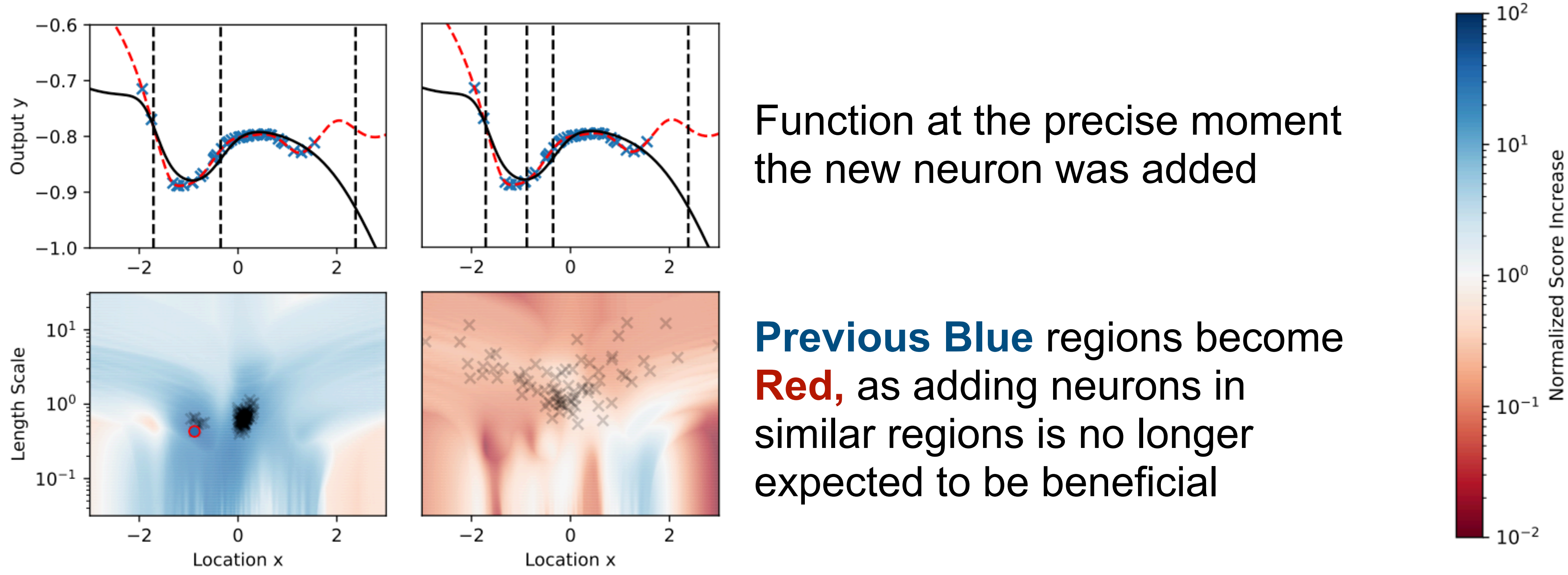
Blue = data
Red dashed = true function
Solid black = SENN fit
Existing neurons = vertical lines

Blue region = increasing score
Red circle = neuron to add
(as prediction error in this region of lots of data is large)

Score: when/where to add neurons for each basis function location & scale

Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

SENN in practice: least squares regression



Function at the precise moment the new neuron was added

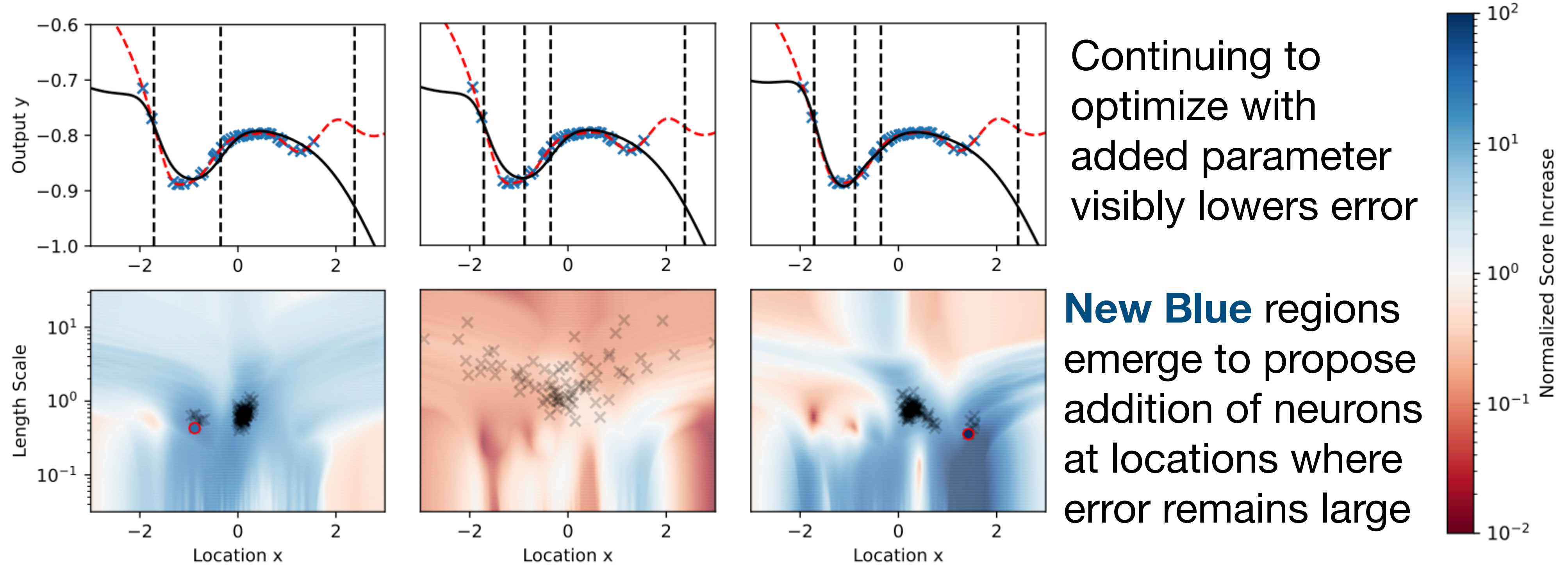
Previous Blue regions become **Red,** as adding neurons in similar regions is no longer expected to be beneficial

Adding a neuron reduces addition score, even prior to optimizing

Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

SENN in practice: least squares regression



Continuing to optimize with added parameter visibly lowers error

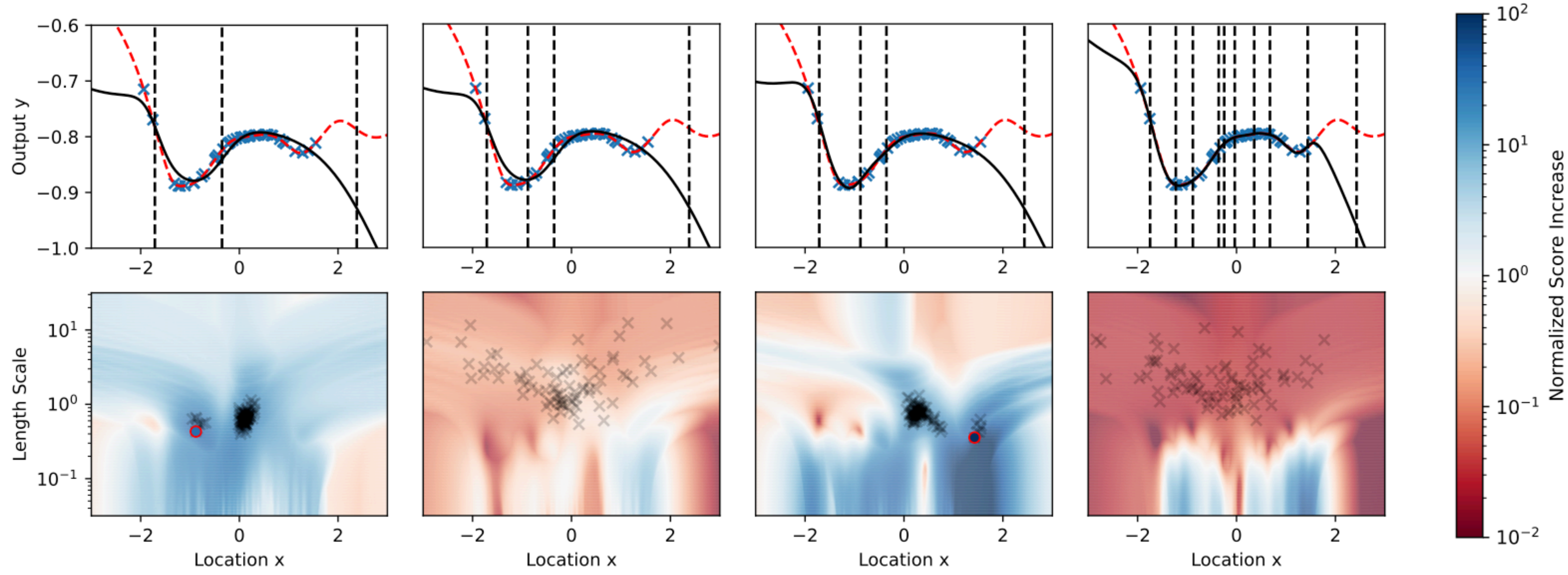
New Blue regions emerge to propose addition of neurons at locations where error remains large

Continuing to optimize new neuron reduces error for part of the data

Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

SENN in practice: least squares regression



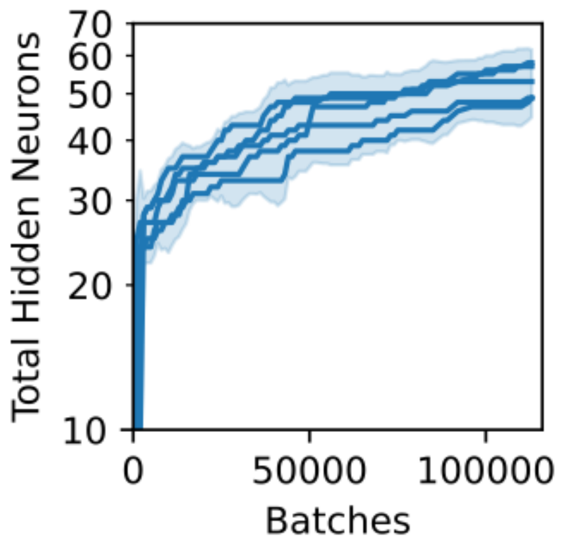
Neurons are added whenever/wherever error can be lowered for data

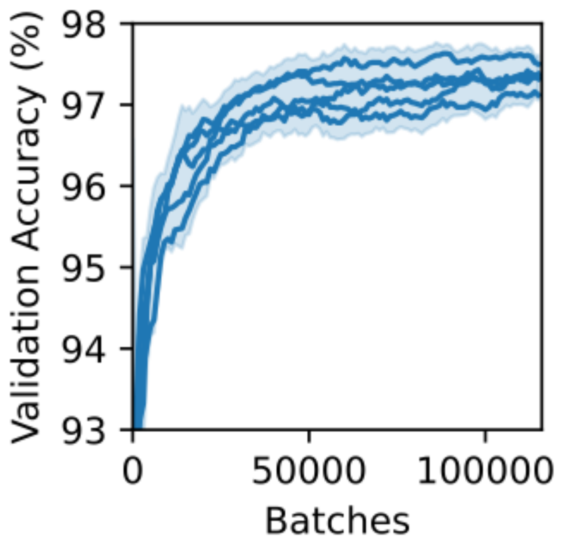
Mitchell et al, "Self-Expanding Neural Networks", arXiv:2307.04526, 2024

Universität Bremen

SENN in practice: MNIST classification

- Reproducible growth during training without perturbing function.
- Accuracy curve looks like a “regular static” neural network.



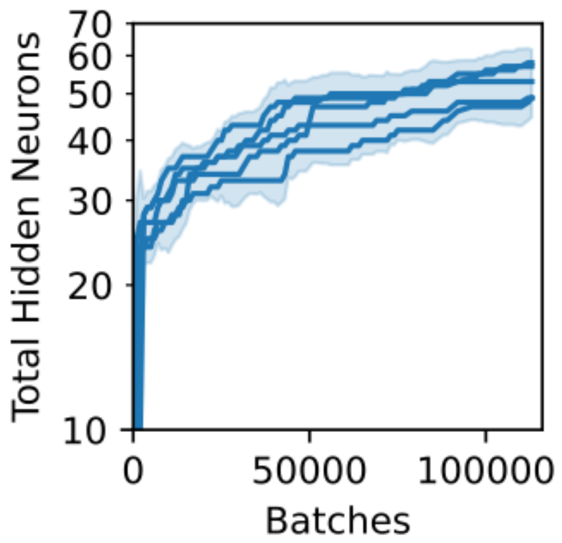


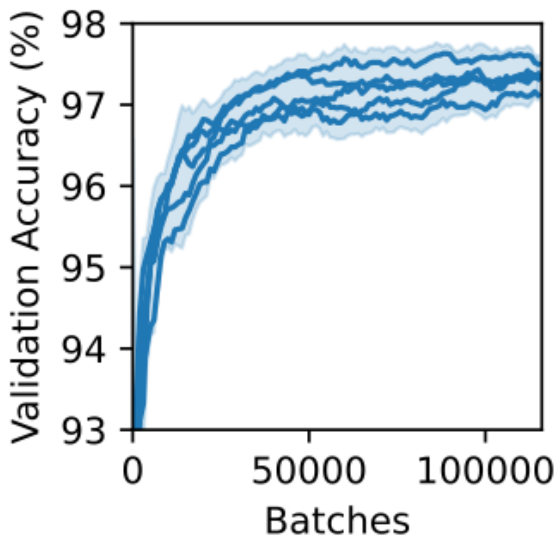
Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

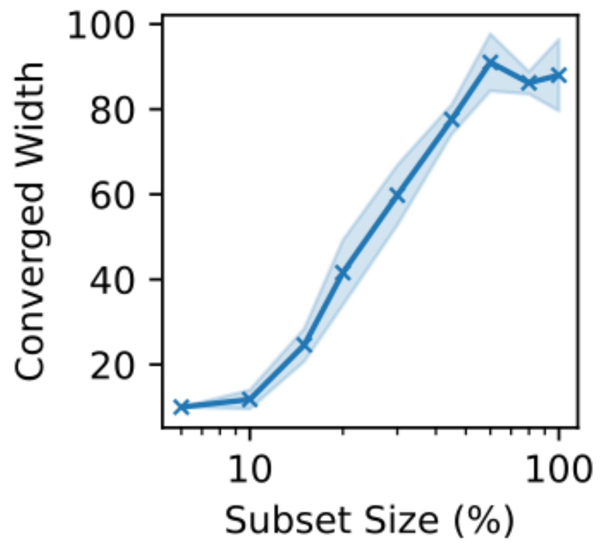
Universität
Bremen

SENN in practice: MNIST classification

- Reproducible growth during training without perturbing function.
- Accuracy curve looks like a “regular static” neural network.
- But layers scale with dataset complexity.





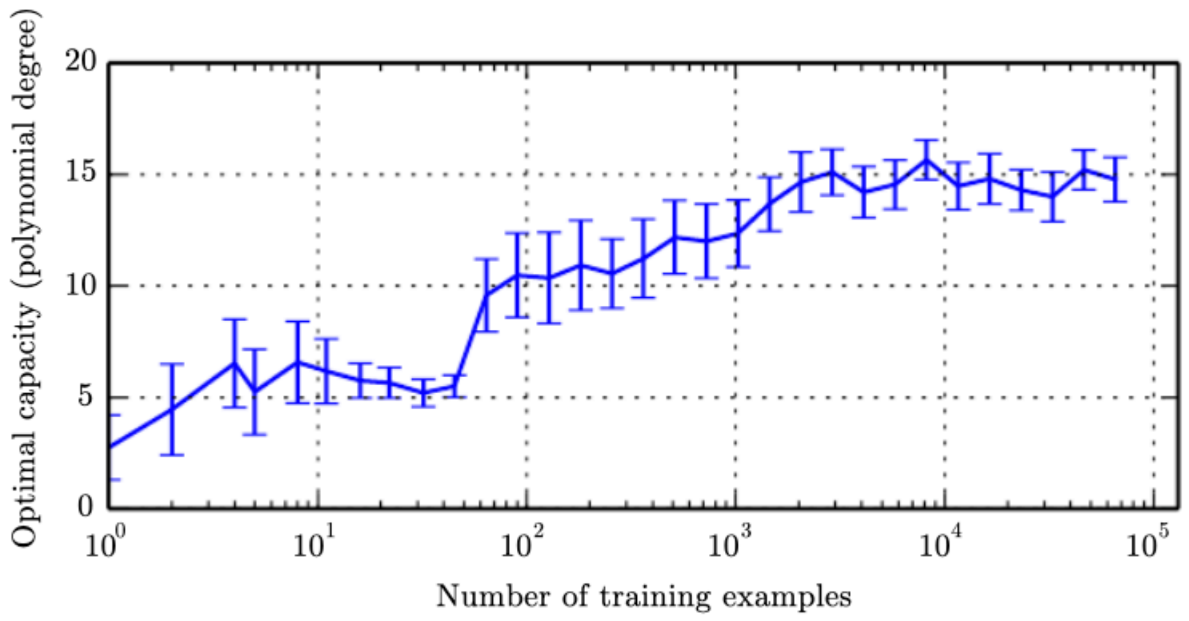


Mitchell et al, “Self-Expanding Neural Networks”, arXiv:2307.04526, 2024

Universität Bremen

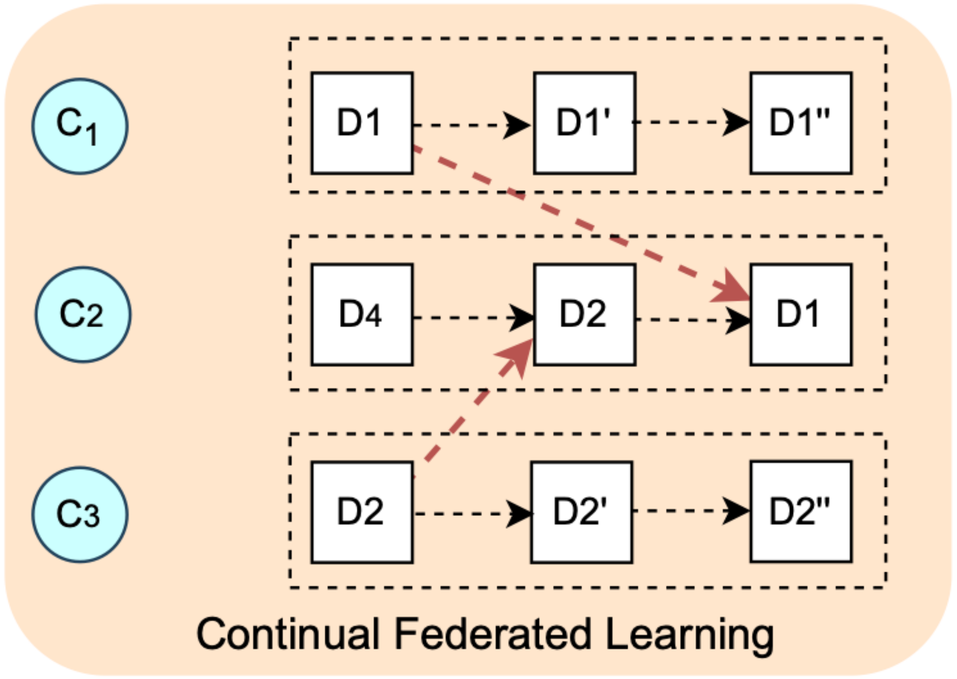
REMINDER: more than stability-plasticity

If it wasn't clear enough already it hopefully is now:
finding the right capacity is a general challenge for ML.
Recall our below image for optimal capacity vs. training examples



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

Universität
Bremen



Paul et al, “Masked Autoencoders are Efficient Continual Federated Learners”, CoLLAs 2024

Part 2 cont. - Retaining the Past

Learning & Evaluation Across Time & Space

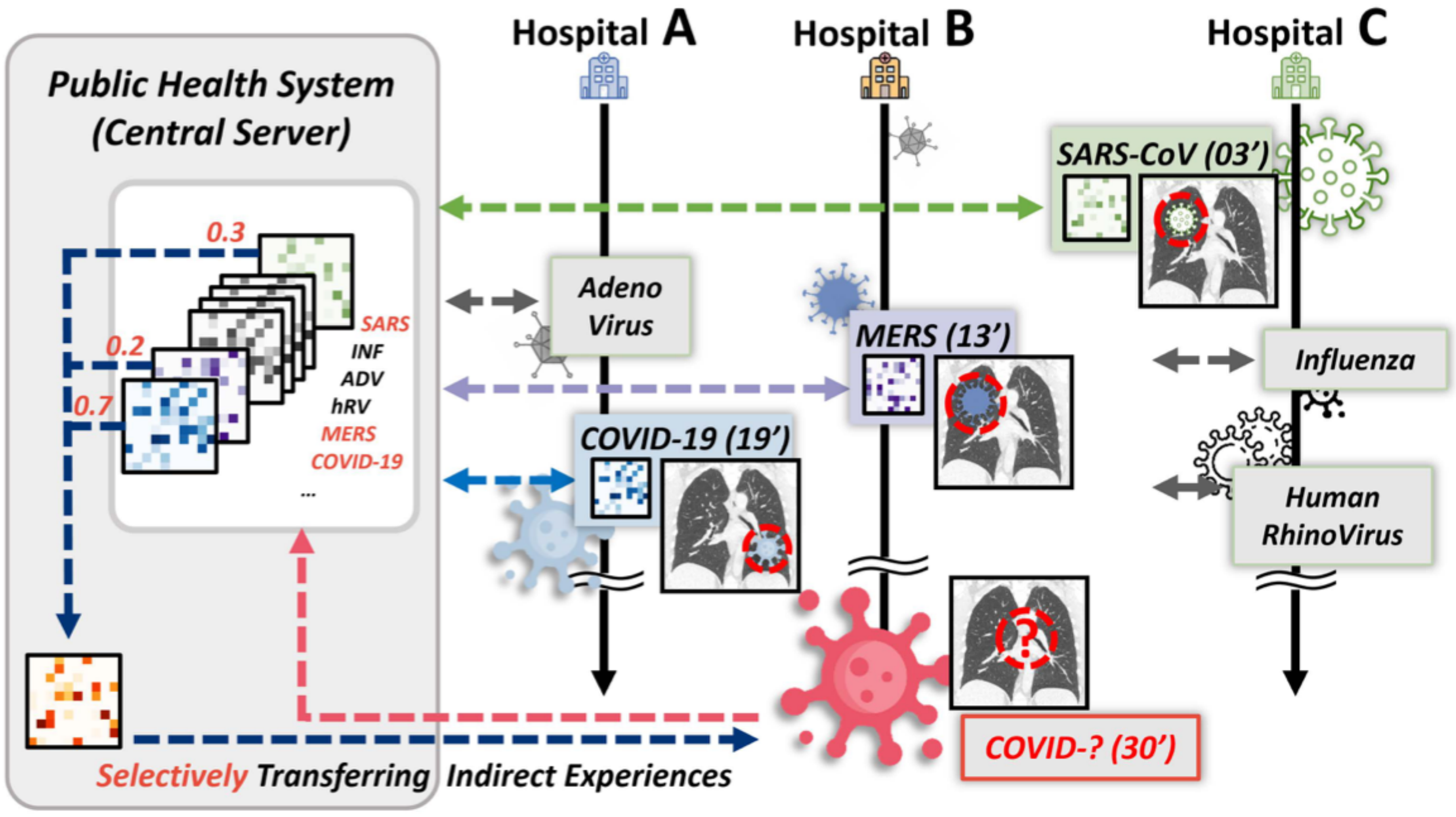
Lifelong Machine Learning
Summer 2025

Prof. Dr. Martin Mundt

Universität
Bremen

OWL ML

Data distributions don’t just change across time

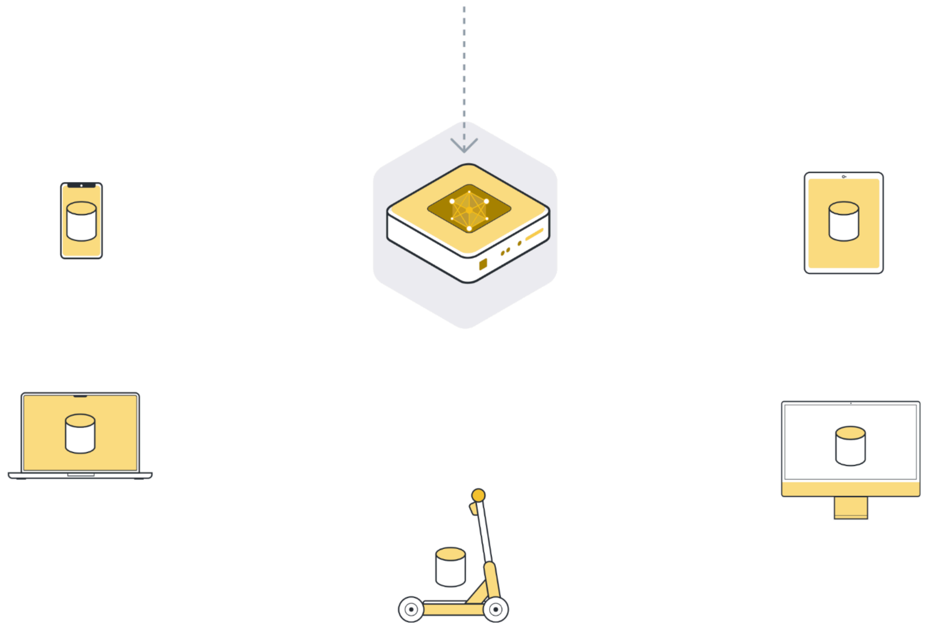


Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

Universität Bremen

(Centralized) Federated Learning

Step 0: Initialize a “global” model

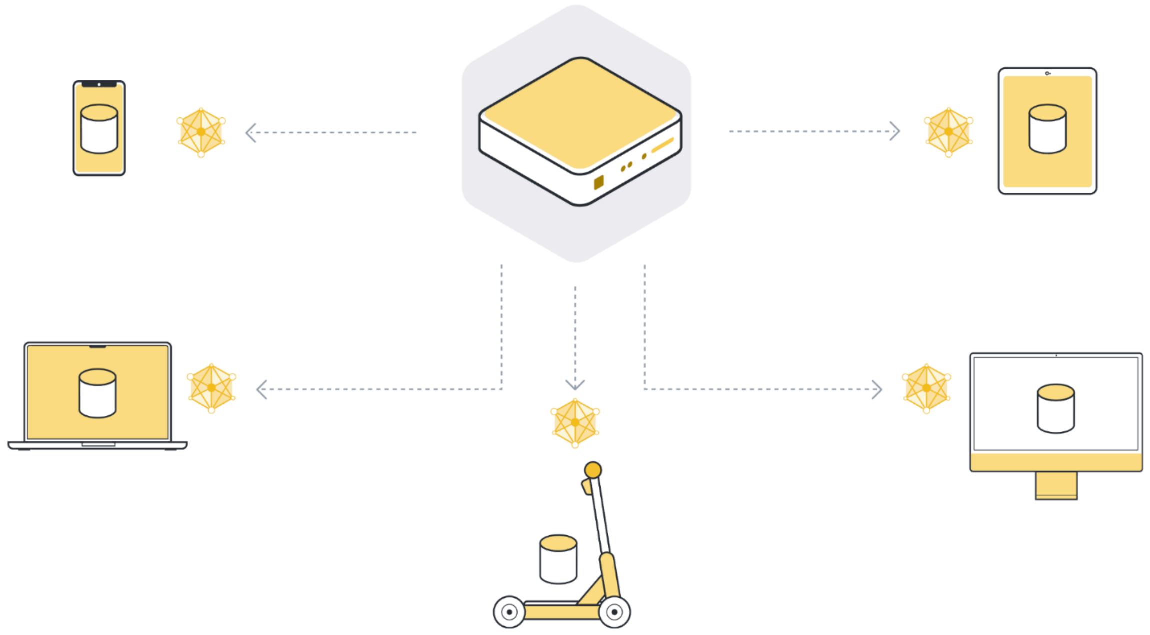


“Federated Learning in Five Steps” from the Flower Framework’s Tutorial:
https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html

Universität Bremen

(Centralized) Federated Learning

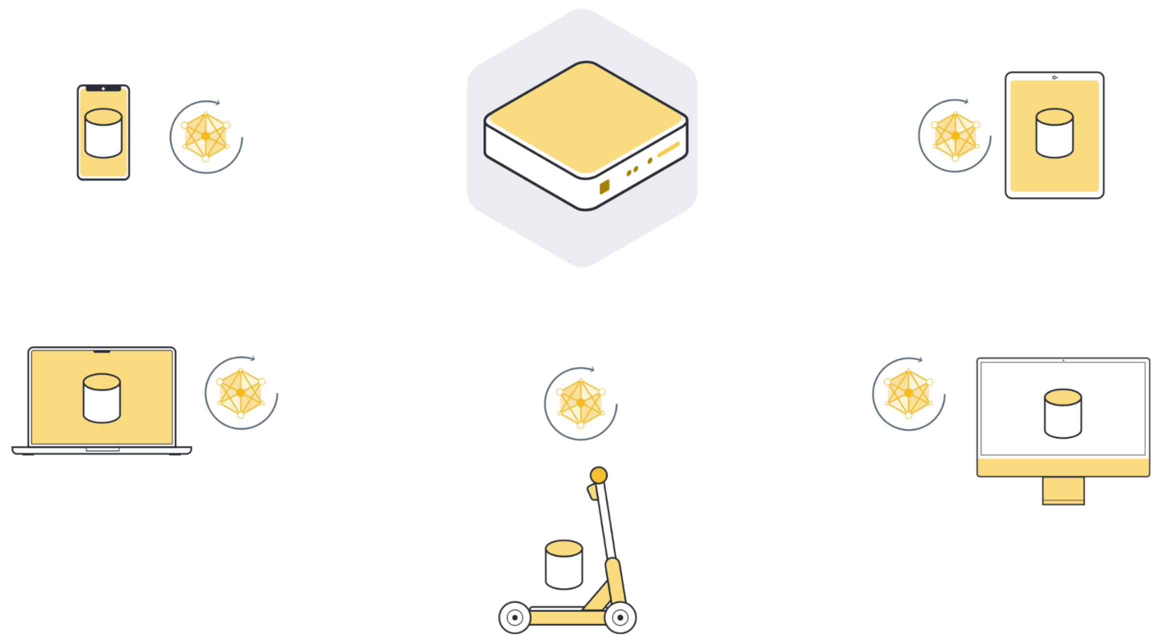
Step 1: Send model to a number of connected devices (client nodes)



“Federated Learning in Five Steps” from the Flower Framework’s Tutorial:
https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html

(Centralized) Federated Learning

Step 2: Train model locally on the data of each device (client node)

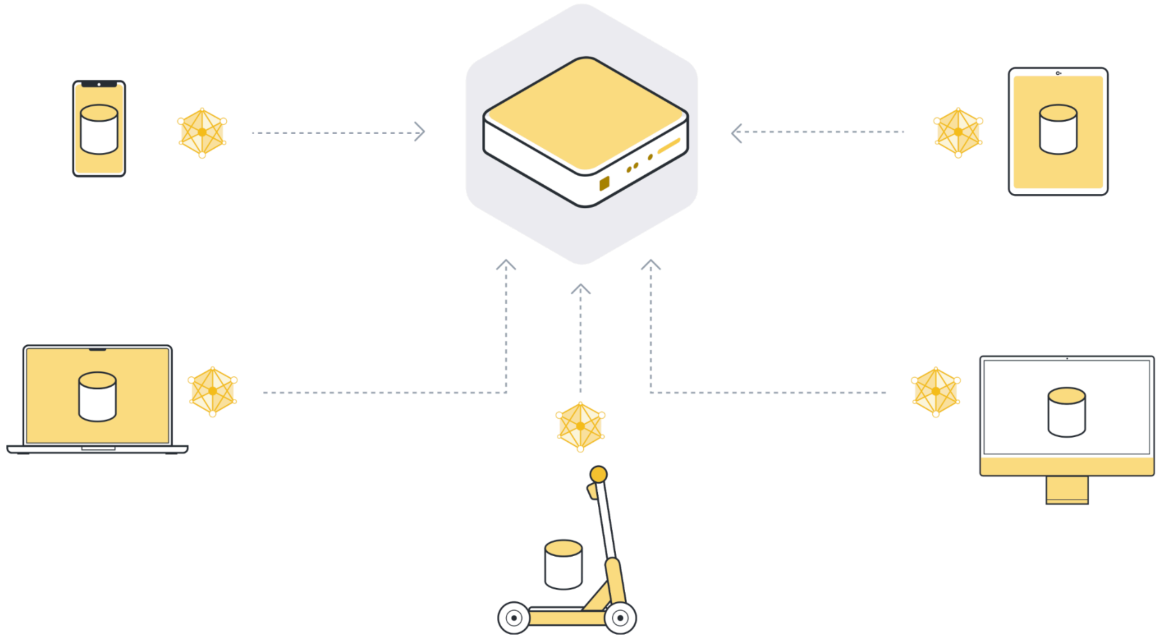


“Federated Learning in Five Steps” from the Flower Framework’s Tutorial:
https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html

Universität Bremen

(Centralized) Federated Learning

Step 3: Return model updates back to the server

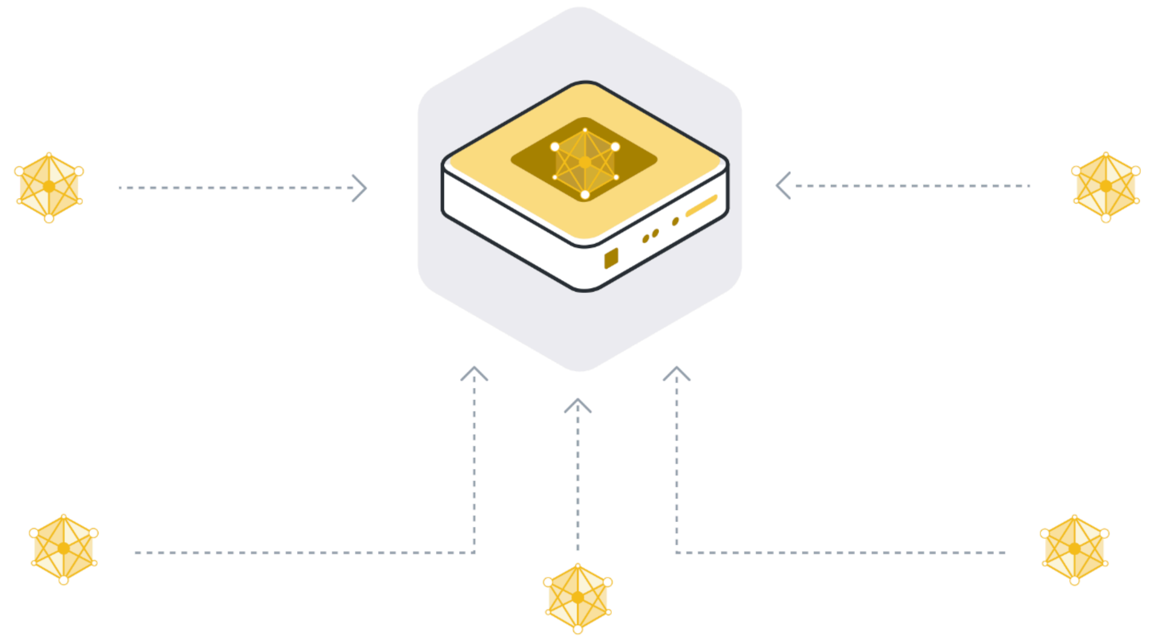


“Federated Learning in Five Steps” from the Flower Framework’s Tutorial:
https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html

Universität Bremen

(Centralized) Federated Learning

Step 4: Aggregate model updates into a new global model



“Federated Learning in Five Steps” from the Flower Framework’s Tutorial:
https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html

Federated Averaging

From an optimization point of view, federated learning is minimization over a combination of non-local objectives.

For K clients over which data is partitioned, with n_k data points on client k, the (weighted) objective is:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

McMahan et al, “Communication-Efficient Learning of Deep Networks from Decentralized Data”, AISTATS 2017

Federated Averaging

From an optimization point of view, federated learning is minimization over a combination of non-local objectives.

For K clients over which data is partitioned, with n_k data points on client k, the (weighted) objective is:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

When partitioning the data uniformly at random, the expectation over $F_k(w)$ over the set of examples per client recovers the IID assumption

McMahan et al, “Communication-Efficient Learning of Deep Networks from Decentralized Data”, AISTATS 2017

Federated Averaging

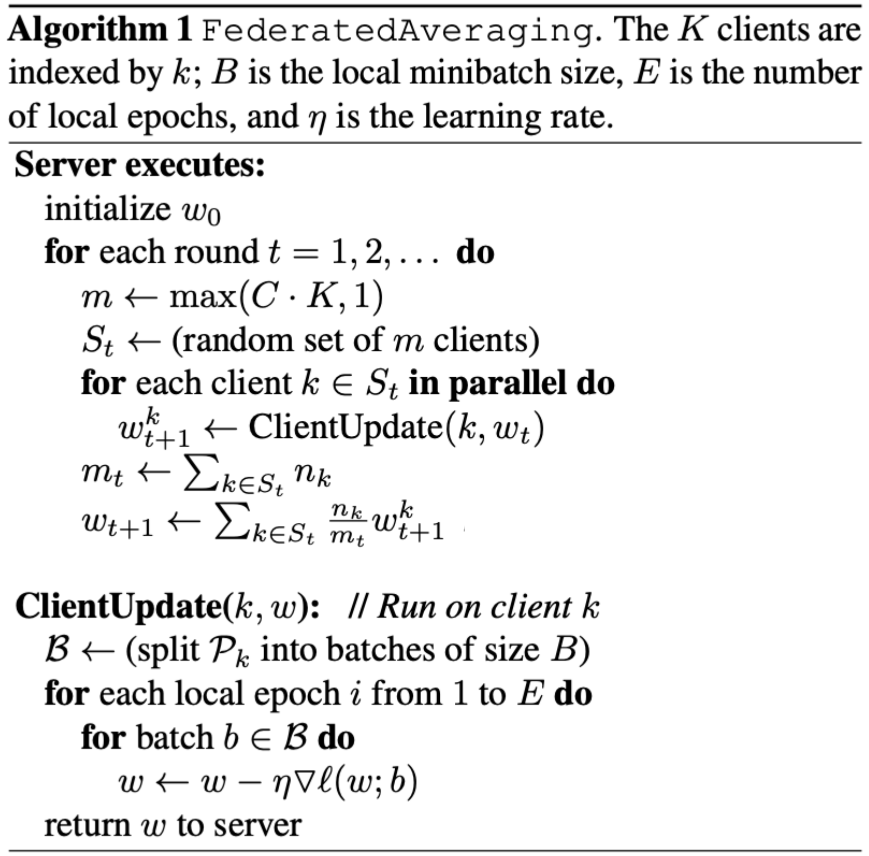
Typical server aggregation takes a weighted average (“FedAvg”):

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

However, we do not necessarily have to communicate after every step. So in practice, it is common to send “pseudo-gradients” after multiple steps (or even epochs)

McMahan et al, “Communication-Efficient Learning of Deep Networks from Decentralized Data”, AISTATS 2017

Federated Averaging: Rounds vs. Steps



Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
  initialize $w_0$
  for each round $t = 1, 2, \dots$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $m_t \leftarrow \sum_{k \in S_t} n_k$
    $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

ClientUpdate($k, w$): // Run on client $k$
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

Additional choices on top of existing hyper-parameters (learning rate, batch size…):

- **Rounds**
- **Epochs** (or local steps)

- Number of participating clients
- Aggregation strategy

McMahan et al, "Communication-Efficient Learning of Deep Networks from Decentralized Data", AISTATS 2017

When does naive federated averaging work?

“Client-server communication is often too slow and expensive. To speed up training (often x10-100) we can make clients spend more time at each round on local training (e.g., do more local SGD steps), thereby reducing the total number of communication rounds.

Muruan Al-Shedivat, “An Inferential Perspective on Federated Learning”, ML-CMU Blog, Feb 2021: https://blog.ml.cmu.edu/2021/02/19/an-inferential-perspective-on-federated-learning/

When does naive federated averaging work?

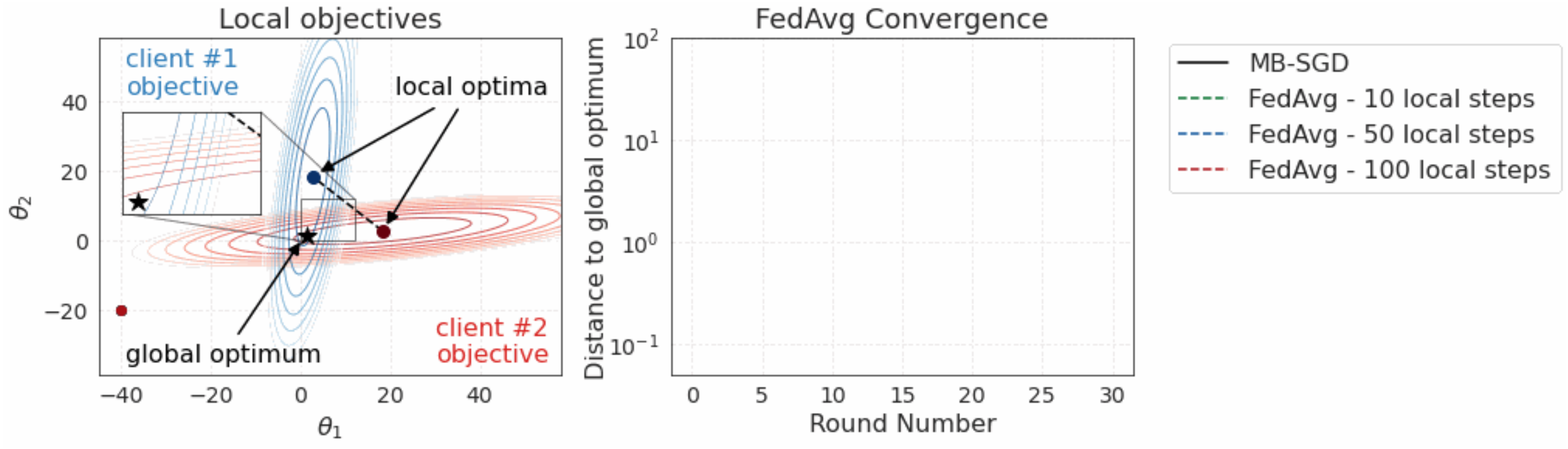
“Client-server communication is often too slow and expensive. To speed up training (often x10-100) we can make clients spend more time at each round on local training (e.g., do more local SGD steps), thereby reducing the total number of communication rounds.

However, because of client data heterogeneity (natural in practice), it turns out that increasing the amount of local computation per round results in convergence to inferior models!”

Muruan Al-Shedivat, “An Inferential Perspective on Federated Learning”, ML-CMU Blog, Feb 2021: https://blog.ml.cmu.edu/2021/02/19/an-inferential-perspective-on-federated-learning/

When does naive federated averaging work?

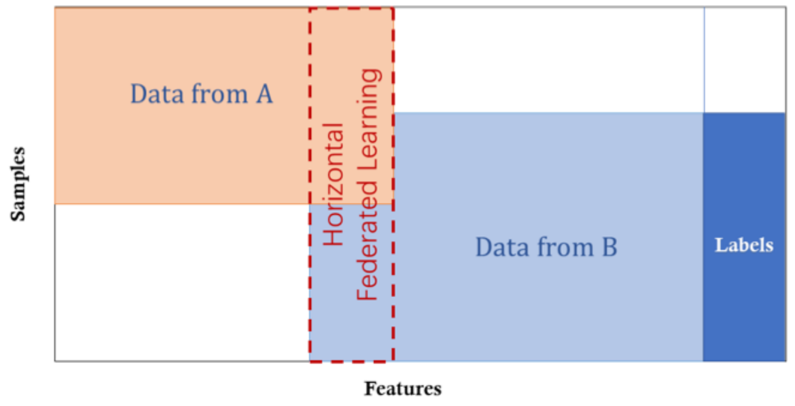
The trade-off between epochs (local steps) & communication rounds



Muruan Al-Shedivat, "An Inferential Perspective on Federated Learning", ML-CMU Blog, Feb 2021: https://blog.ml.cmu.edu/2021/02/19/an-inferential-perspective-on-federated-learning/

Universität
Bremen

Horizontal & Vertical Federated Learning

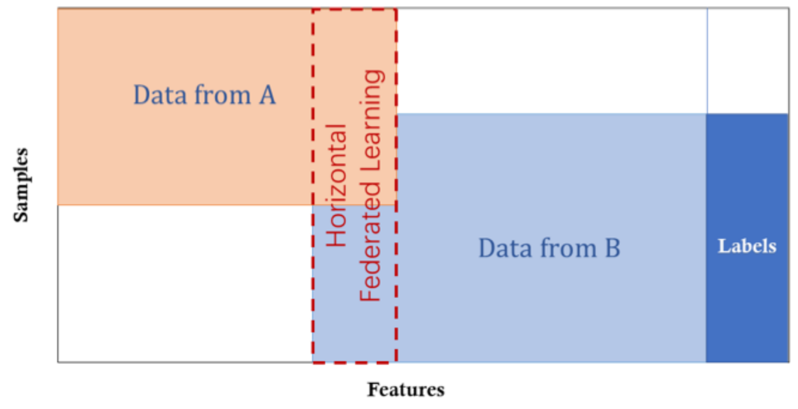


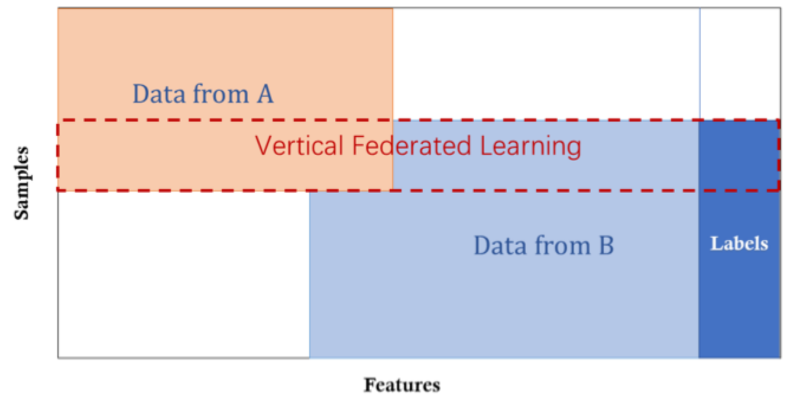
This trade-off is particularly severe when we are in non-IID settings!

Images from Yang et al, “Federated Machine Learning: Concept and Applications”, ACM Trans. Intell. Syst. Technol. 10, 2019

Universität Bremen

Horizontal & Vertical Federated Learning



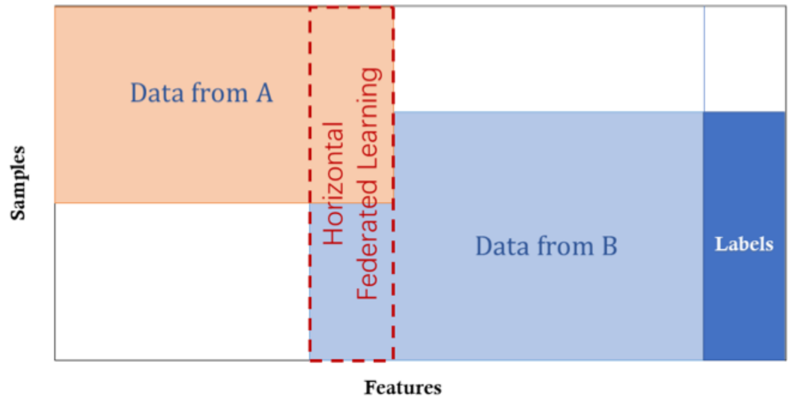


This trade-off is particularly severe when we are in non-IID settings!

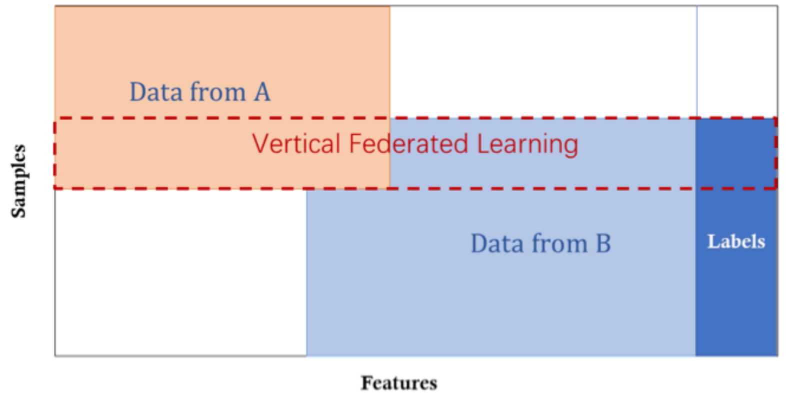
Images from Yang et al, "Federated Machine Learning: Concept and Applications", ACM Trans. Intell. Syst. Technol. 10, 2019

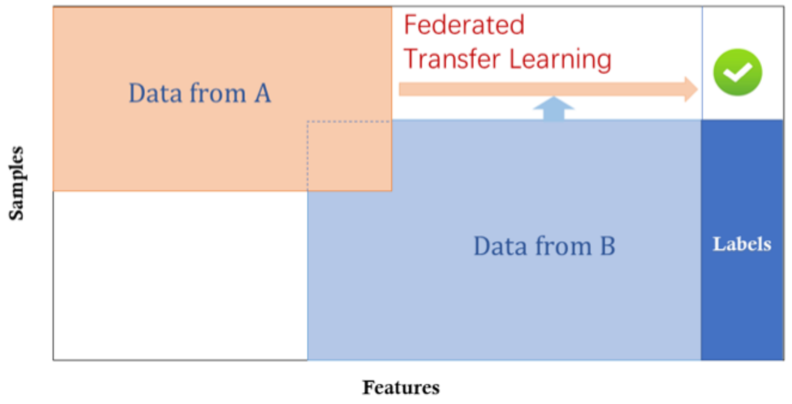
Universität Bremen

Horizontal & Vertical Federated Learning



This trade-off is particularly severe when we are in non-IID settings! Federated literature may call these differently, but you should already be familiar with the concepts.



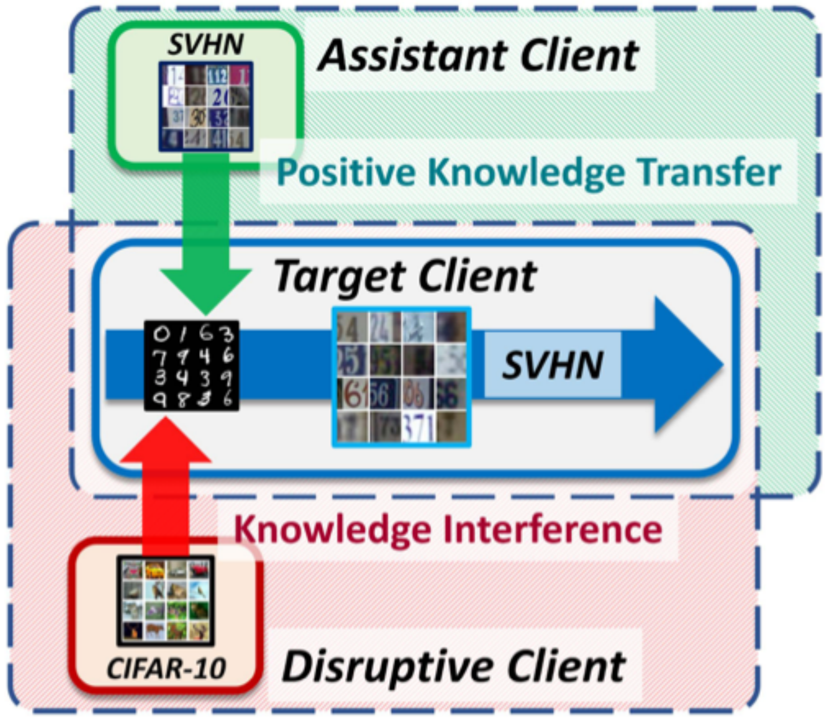


Images from Yang et al, "Federated Machine Learning: Concept and Applications", ACM Trans. Intell. Syst. Technol. 10, 2019

Horizontal & Vertical Federated Learning

The challenges arising from data heterogeneity in federated learning are quite similar to those of continual learning.

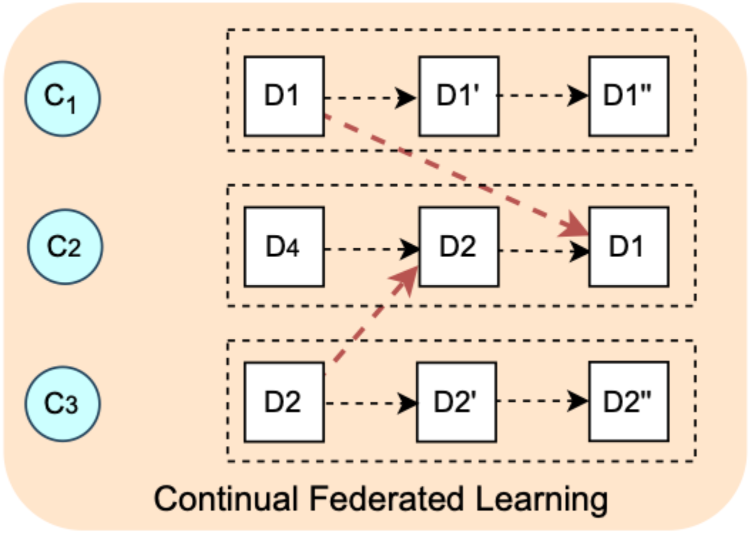
- If we give two clients different tasks, we will encounter catastrophic interference.



Horizontal & Vertical Federated Learning

The challenges arising from data heterogeneity in federated learning are quite similar to those of continual learning.

- If we give two clients different tasks, we will encounter catastrophic interference.
- If the data distribution on a client drifts, the client will suffer from catastrophic forgetting



More involved set-up: experiences can now be heterogeneous across time and space

Image from Paul et al, “Masked Autoencoders are Efficient Continual Federated Learners”, CoLLAs 2024

Universität Bremen

Question time

How do we avoid client interference in non-IID federated (continual) learning settings?

Federated Learning Improvements

Initially federated learning might sound like a different field with different objectives - in fact it was treated as such initially - until the realization came up that optimization challenges are analogous.

We’ve already learned about helpful approaches that we can apply:

1. Curvature estimates
2. Knowledge distillation
3. Architecture decomposition

Federated Learning Improvements

Initially federated learning might sound like a different field with different objectives - in fact it was treated as such initially - until the realization came up that optimization challenges are analogous.

We’ve already learned about helpful approaches that we can apply:

1. Curvature estimates
2. Knowledge distillation
3. Architecture decomposition

But keep in mind that in federated settings, we need to also consider communication efficiency and privacy concerns: no replay!

1. Federated Curvature

Since we’ve done a full EWC derivation, it’s quite intuitive to explain: essentially think of EWC with communication of the Fisher

At round t each node $s \in S$ optimizes the following loss:

$$\tilde{L}_{t,s}(\theta) = L_s(\theta) + \lambda \sum_{j \in S \setminus s} (\theta - \hat{\theta}_{t-1,j})^T \operatorname{diag}(\hat{\mathcal{I}}_{t-1,j})(\theta - \hat{\theta}_{t-1,j}), \qquad (3)$$

On each round t, starting from initial point $\hat{\theta}_t = \frac{1}{N}\sum_{i=1}^{N} \hat{\theta}_{t-1,i}$, the nodes optimize their local loss by running SGD for E local epochs.

Shoham et al, “Overcoming Forgetting in Federated Learning on Non-IID Data”, NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality

1. Federated Curvature

Since we’ve done a full EWC derivation, it’s quite intuitive to explain: essentially think of EWC with communication of the Fisher

At round t each node $s \in S$ optimizes the following loss:

$$\tilde{L}_{t,s}(\theta) = L_s(\theta) + \lambda \sum_{j \in S \setminus s} (\theta - \hat{\theta}_{t-1,j})^T \operatorname{diag}(\hat{\mathcal{I}}_{t-1,j})(\theta - \hat{\theta}_{t-1,j}), \tag{3}$$

On each round t, starting from initial point $\hat{\theta}_t = \frac{1}{N}\sum_{i=1}^{N} \hat{\theta}_{t-1,i}$, the nodes optimize their local loss by running SGD for E local epochs. At the end of each round t, each node j sends to the rest of the nodes the SGD result $\hat{\theta}_{t,j}$ and $\operatorname{diag}(\hat{\mathcal{I}}_{t,j})$ (where $\hat{\mathcal{I}}_{t,j} = \mathcal{I}(\hat{\theta}_{t,j})$). $\hat{\theta}_{t,j}$ and $\operatorname{diag}(\hat{\mathcal{I}}_{t,j})$ will be used for the loss of round $t+1$. We switched from θ^* to $\hat{\theta}$ to signify that local tasks are optimized for E epochs and not until they converge (as was the case for EWC). However, (2) (its generalization to N tasks) supports using large values of E, so $\hat{\theta}_{t,j} \approx \theta^*_{t,j}$ and then $\tilde{L}_{t,j} \approx L$.

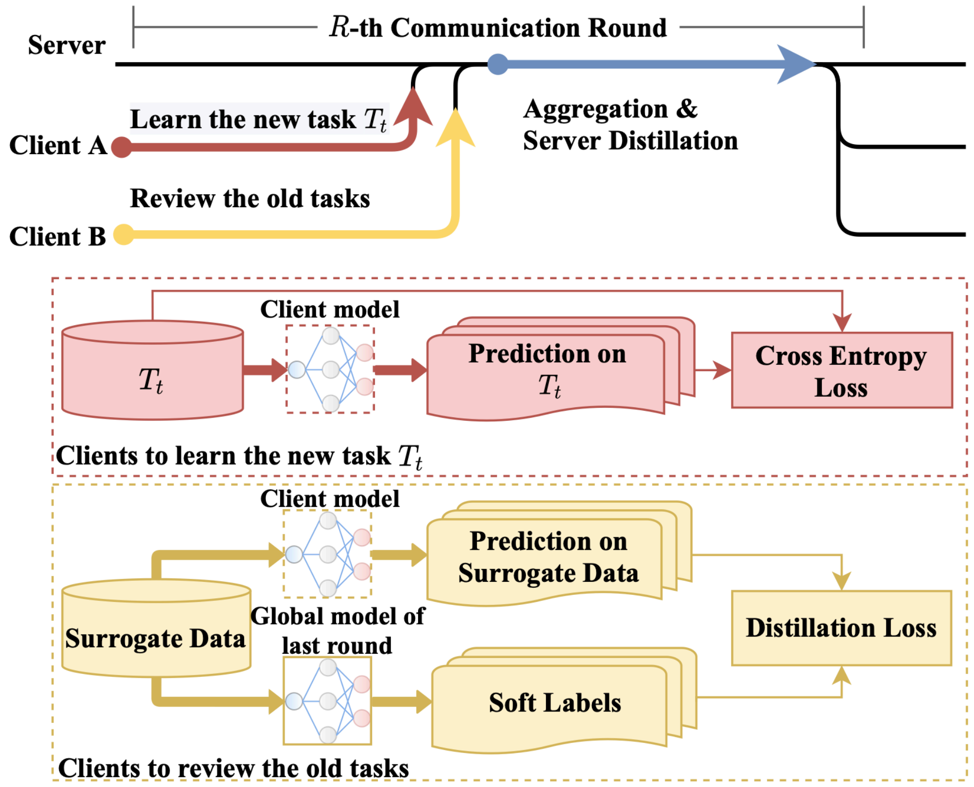
Shoham et al, “Overcoming Forgetting in Federated Learning on Non-IID Data”, NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality

Universität Bremen

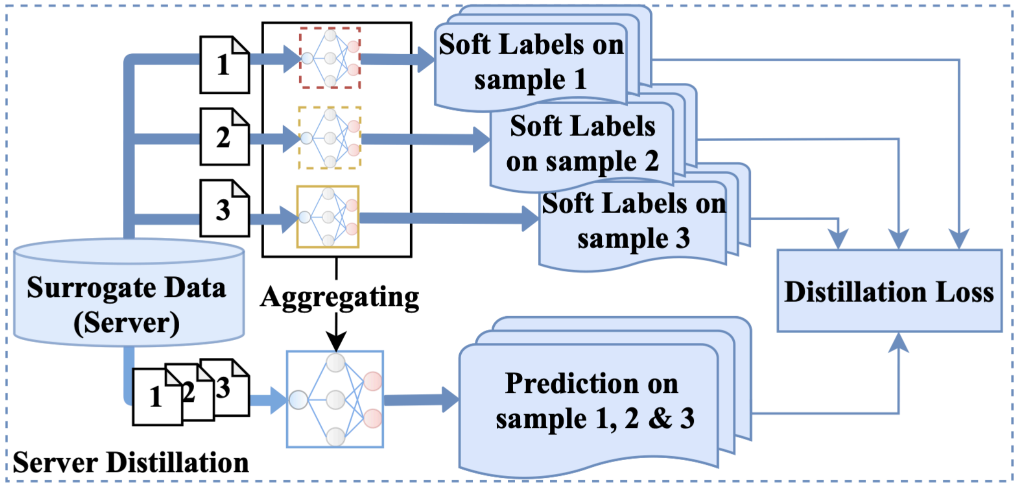
OWL ML

2. Federated Knowledge Distillation

We don't even need equations to understand: KD + multiple models



R-th Communication Round
Server
Client A
Client B
Learn the new task T_t
Review the old tasks
Aggregation & Server Distillation
Client model
T_t
Prediction on T_t
Cross Entropy Loss
Clients to learn the new task T_t
Client model
Surrogate Data
Prediction on Surrogate Data
Global model of last round
Distillation Loss
Soft Labels
Clients to review the old tasks



Ma et al, "Continual Federated Learning Based on Knowledge Distillation", IJCAI 2022

Universität Bremen

3. Federated Architecture Decomposition

Perhaps you already expected this based on the “continual learning” approaches, but this part is more involved than EWC/KD

In general, inspiration is similar as before, use masks or “attention” to find which subsets of parameters to isolate in the architecture.

In other words: “decompose” the architecture into “general” and “task-adaptive” parts. We’ll look into a specific example called “Federated Weighted Inter-Client Transfer (FedWeIT, Yoon et al 21)”

3. Federated Architecture Dec.: FedWeIT

FedWeIT attempts to decompose parameters into:

- **General parameters G**:
  that capture the global and generic knowledge across all clients
- **Base parameters B**:
  which capture generic knowledge for each client
- **Task adaptive parameters A**:
  which capture knowledge for each specific task per client

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

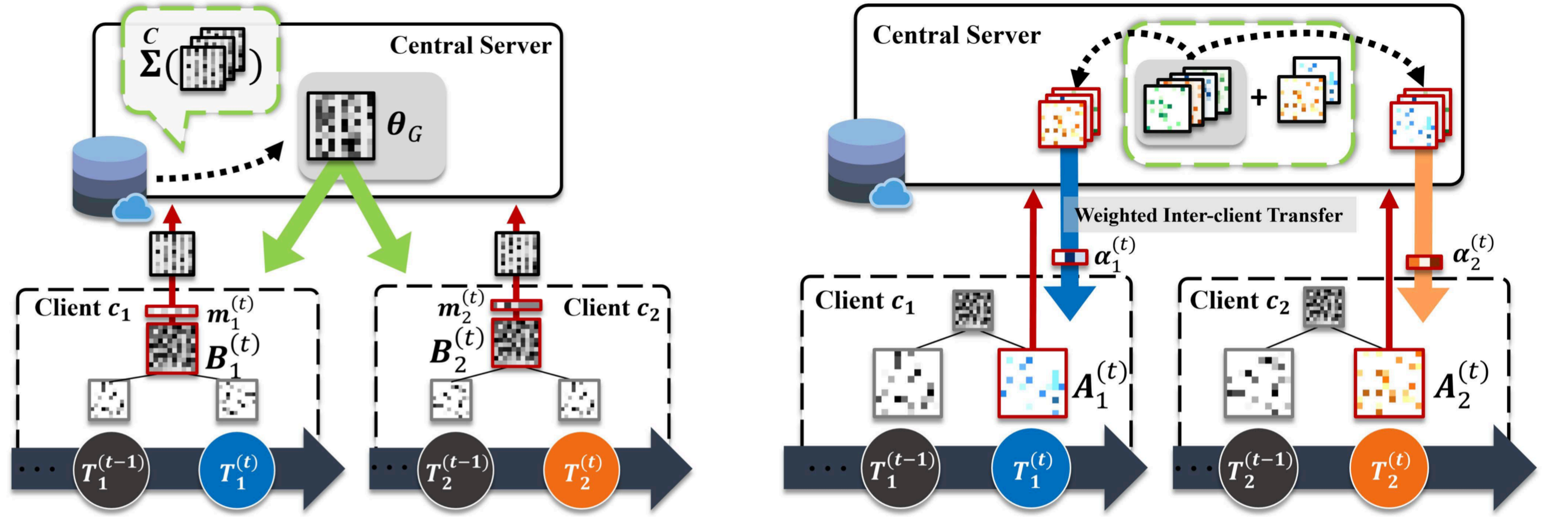
3. Federated Architecture Dec.: FedWeIT

In practice: $\theta_c^{(t)} = B_c^{(t)} \circ m_c^{(t)} + A_c^{(t)} + \sum_{i \in C_{\backslash c}} \sum_{j < |t|} \alpha_{i,j}^{(t)} A_i^{(j)}$

- Base parameters (useful for all client tasks) are multiplied with a learned mask to avoid interfering with general global knowledge G
- Adaptive parameters are transmitted to global/central server
- All task-adaptive parameters are sent from the server to all clients and an attention matrix determines whether select parameter are helpful to the task each client is learning at the present point in time (think of a task seen by another client before)

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

3. Federated Architecture Dec.: FedWeIT



(a) Communication of General Knowledge (b) Communication of Task-adaptive Knowledge

Yoon et al, "Federated Continual Learning with Weighted Inter-client Transfer", ICML 2021

3. Federated Architecture Dec.: FedWeIT

How do we learn to actually decompose the parameters?

$$\mathtt{min}_{B_c^{(t)}, m_c^{(t)}, \alpha_c^{(t)}, A_c^{(1:t)}} \underline{\mathcal{L}(\theta_c^{(t)}; \mathcal{T}_c^{(t)})} + \lambda_1 \Omega(\{m_c^{(t)}, A_c^{(1:t)}\}) + \lambda_2 \sum_{i=1}^{t-1} || \Delta B_c^{(t)} \circ m_c^{(i)} + \Delta A_c^{(i)} ||_2^2$$

Looks more complicated than it is:

1. term: regular loss

3. Federated Architecture Dec.: FedWeIT

How do we learn to actually decompose the parameters?

$$\mathtt{min}_{B_c^{(t)}, m_c^{(t)}, \alpha_c^{(t)}, A_c^{(1:t)}} \mathcal{L}(\theta_c^{(t)}; \mathcal{T}_c^{(t)}) + \underline{\lambda_1 \Omega(\{m_c^{(t)}, A_c^{(1:t)}\})} + \lambda_2 \sum_{i=1}^{t-1} || \Delta B_c^{(t)} \circ m_c^{(i)} + \Delta A_c^{(i)} ||_2^2$$

Looks more complicated than it is:
1. term: regular loss
2. term: L1 sparsity regularizer to ensure A is a small set of params

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

3. Federated Architecture Dec.: FedWeIT

How do we learn to actually decompose the parameters?

$$\mathtt{min}_{B_c^{(t)}, m_c^{(t)}, \alpha_c^{(t)}, A_c^{(1:t)}} \mathscr{L}(\theta_c^{(t)}; \mathscr{T}_c^{(t)}) + \lambda_1 \Omega(\{m_c^{(t)}, A_c^{(1:t)}\}) + \underline{\lambda_2 \sum_{i=1}^{t-1} || \Delta B_c^{(t)} \circ m_c^{(i)} + \Delta A_c^{(i)} ||_2^2}$$

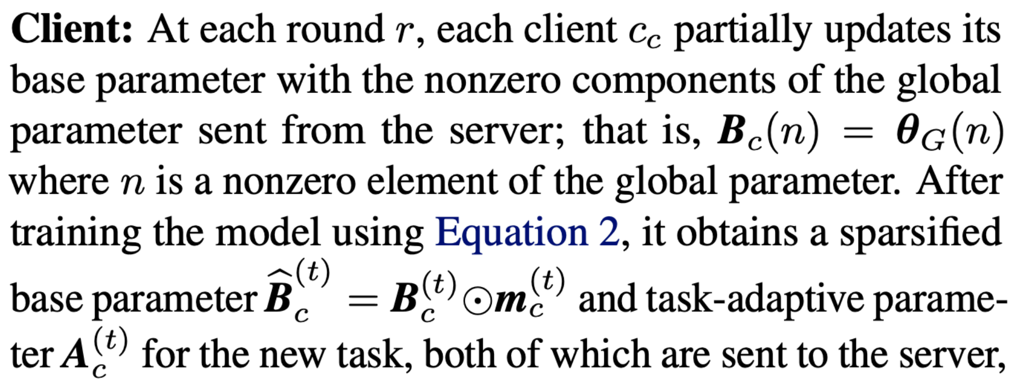
Looks more complicated than it is:

1. term: regular loss
2. term: L1 sparsity regularizer to ensure A is a small set of params
3. term: $\Delta B_c^{(t)} = B_c^{(t)} - B_c^{(t-1)}$ (& for A) regularizes params between t & t-1. Prevents forgetting like our known L2 regularizers, but in a masked + decomposed version that has to account for shifts in B & A

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

3. Federated Architecture Dec.: FedWeIT

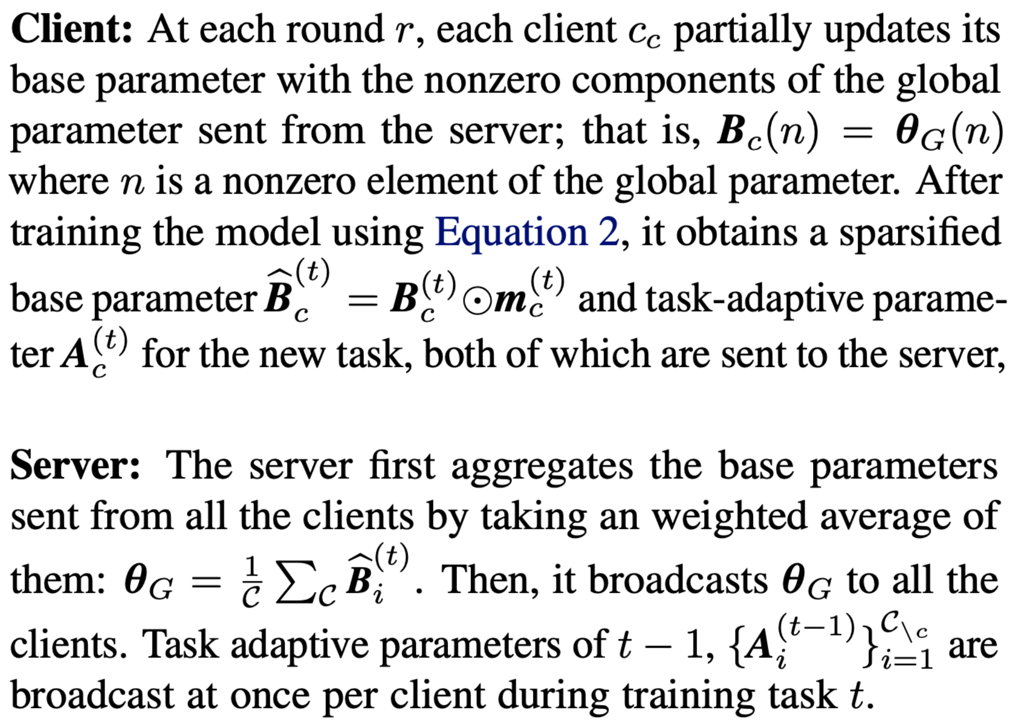
In summary:



Client: At each round r, each client c_c partially updates its base parameter with the nonzero components of the global parameter sent from the server; that is, $\boldsymbol{B}_c(n) = \boldsymbol{\theta}_G(n)$ where n is a nonzero element of the global parameter. After training the model using Equation 2, it obtains a sparsified base parameter $\widehat{\boldsymbol{B}}_c^{(t)} = \boldsymbol{B}_c^{(t)} \odot \boldsymbol{m}_c^{(t)}$ and task-adaptive parameter $\boldsymbol{A}_c^{(t)}$ for the new task, both of which are sent to the server,

3. Federated Architecture Dec.: FedWeIT

In summary:



Client: At each round r, each client c_c partially updates its base parameter with the nonzero components of the global parameter sent from the server; that is, $\boldsymbol{B}_c(n) = \boldsymbol{\theta}_G(n)$ where n is a nonzero element of the global parameter. After training the model using Equation 2, it obtains a sparsified base parameter $\widehat{\boldsymbol{B}}_c^{(t)} = \boldsymbol{B}_c^{(t)} \odot \boldsymbol{m}_c^{(t)}$ and task-adaptive parameter $\boldsymbol{A}_c^{(t)}$ for the new task, both of which are sent to the server,

Server: The server first aggregates the base parameters sent from all the clients by taking an weighted average of them: $\boldsymbol{\theta}_G = \frac{1}{\mathcal{C}} \sum_{\mathcal{C}} \widehat{\boldsymbol{B}}_i^{(t)}$. Then, it broadcasts $\boldsymbol{\theta}_G$ to all the clients. Task adaptive parameters of $t-1$, $\{\boldsymbol{A}_i^{(t-1)}\}_{i=1}^{\mathcal{C}\setminus c}$ are broadcast at once per client during training task t.

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

Universität Bremen

OWL ML

3. Federated Architecture Dec.: FedWeIT

In summary:

Client: At each round r, each client c_c partially updates its base parameter with the nonzero components of the global parameter sent from the server; that is, $\boldsymbol{B}_c(n) = \boldsymbol{\theta}_G(n)$ where n is a nonzero element of the global parameter. After training the model using Equation 2, it obtains a sparsified base parameter $\widehat{\boldsymbol{B}}_c^{(t)} = \boldsymbol{B}_c^{(t)} \odot \boldsymbol{m}_c^{(t)}$ and task-adaptive parameter $\boldsymbol{A}_c^{(t)}$ for the new task, both of which are sent to the server,

Server: The server first aggregates the base parameters sent from all the clients by taking an weighted average of them: $\boldsymbol{\theta}_G = \frac{1}{\mathcal{C}} \sum_{\mathcal{C}} \widehat{\boldsymbol{B}}_i^{(t)}$. Then, it broadcasts $\boldsymbol{\theta}_G$ to all the clients. Task adaptive parameters of $t-1$, $\{\boldsymbol{A}_i^{(t-1)}\}_{i=1}^{\mathcal{C}_{\setminus c}}$ are broadcast at once per client during training task t.

Algorithm 1 Federated Weighted Inter-client Transfer

```
input  Dataset $\{\mathcal{D}_c^{(1:t)}\}_{c=1}^{C}$, global parameter $\boldsymbol{\theta}_G$,
       hyperparameters $\lambda_1, \lambda_2$, knowledge base $kb \leftarrow \{\}$
output $\{\boldsymbol{B}_c, \boldsymbol{m}_c^{(1:t)}, \boldsymbol{\alpha}_c^{(1:t)}, \boldsymbol{A}_c^{(1:t)}\}_{c=1}^{C}$
 1: Initialize $\boldsymbol{B}_c$ to $\boldsymbol{\theta}_G$ for all clients $\mathcal{C} \equiv \{c_1, ..., c_C\}$
 2: for task $t = 1, 2, ...$ do
 3:   Randomly sample knowledge base $kb^{(t)} \sim kb$
 4:   for round $r = 1, 2, ...$ do
 5:     Collect communicable clients $\mathcal{C}^{(r)} \sim \mathcal{C}$
 6:     Distribute $\boldsymbol{\theta}_G$ and $kb^{(t)}$ to client $c_c \in \mathcal{C}^{(r)}$ if $c_c$ meets
        $kb^{(t)}$ first, otherwise distribute only $\boldsymbol{\theta}_G$
 7:     Minimize Equation 2 for solving local CL problems
 8:     $\boldsymbol{B}_c^{(t,r)} \odot \boldsymbol{m}_c^{(t,r)}$ are transmitted from $\mathcal{C}^{(r)}$ to the server
 9:     Update $\boldsymbol{\theta}_G \leftarrow \frac{1}{|\mathcal{C}^{(r)}|} \sum_c \boldsymbol{B}_c^{(t,r)} \odot \boldsymbol{m}_c^{(t,r)}$
10:   end for
11:   Update knowledge base $kb \leftarrow kb \cup \{\boldsymbol{A}_j^{(t)}\}_{j \in \mathcal{C}}$
12: end for
```

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

Universität Bremen

3. Federated Architecture Dec.: FedWeIT

A brief note on communication cost with clients C & rounds R:

Client side

Naive: $|C| \times R \times |\theta|$

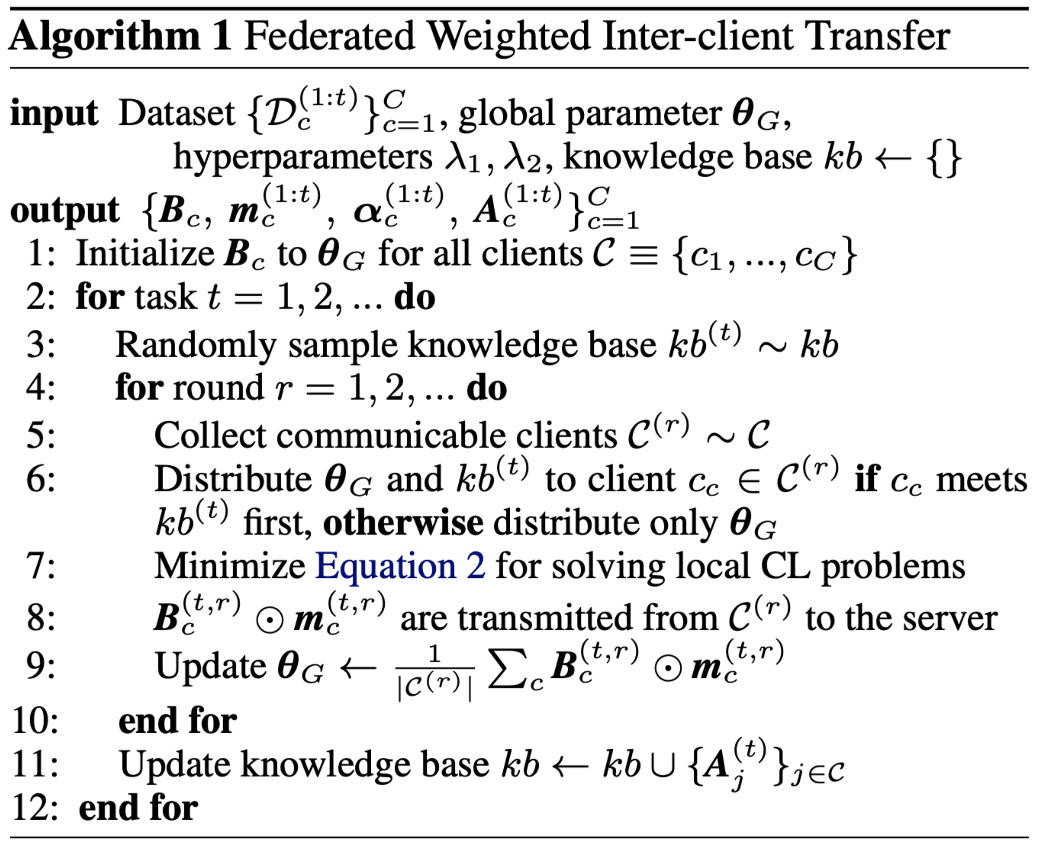
FedWeIT: $|C| \times (R \times |\hat{B}| + |A|)$

Server side

Naive: $|C| \times R \times |\theta|$

FedWeIT:

$|C| \times (R \times |\theta_G| + (|C| - 1) \times |A|)$



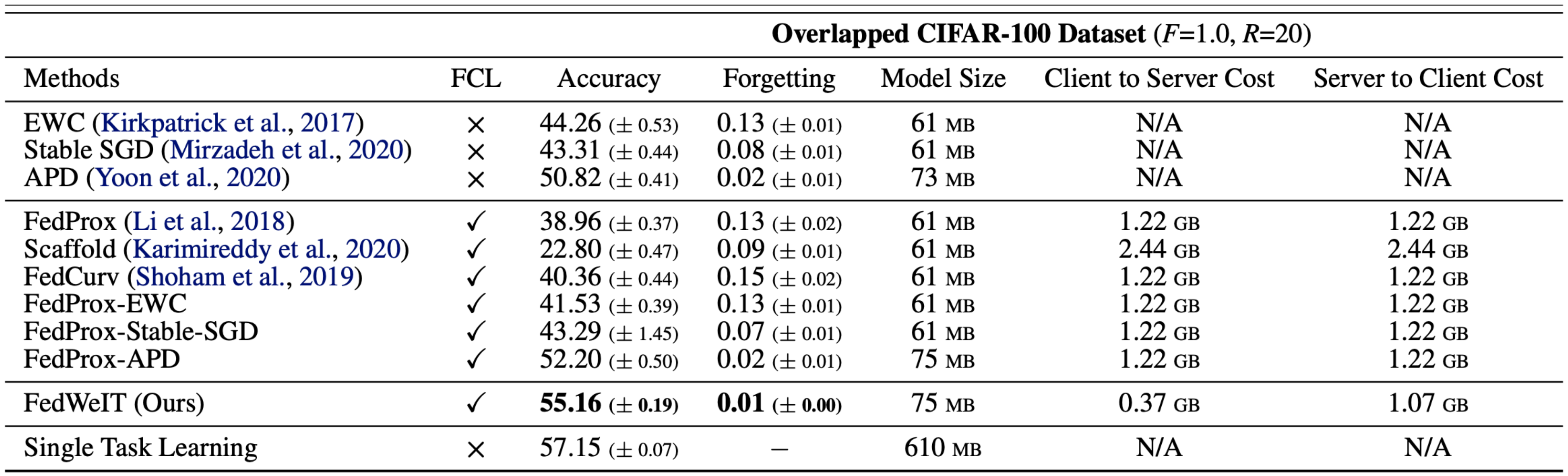
Algorithm 1 Federated Weighted Inter-client Transfer

```
input  Dataset $\{\mathcal{D}_c^{(1:t)}\}_{c=1}^{C}$, global parameter $\boldsymbol{\theta}_G$,
       hyperparameters $\lambda_1, \lambda_2$, knowledge base $kb \leftarrow \{\}$
output $\{\boldsymbol{B}_c, \boldsymbol{m}_c^{(1:t)}, \boldsymbol{\alpha}_c^{(1:t)}, \boldsymbol{A}_c^{(1:t)}\}_{c=1}^{C}$
 1: Initialize $\boldsymbol{B}_c$ to $\boldsymbol{\theta}_G$ for all clients $\mathcal{C} \equiv \{c_1, ..., c_C\}$
 2: for task $t = 1, 2, ...$ do
 3:     Randomly sample knowledge base $kb^{(t)} \sim kb$
 4:     for round $r = 1, 2, ...$ do
 5:         Collect communicable clients $\mathcal{C}^{(r)} \sim \mathcal{C}$
 6:         Distribute $\boldsymbol{\theta}_G$ and $kb^{(t)}$ to client $c_c \in \mathcal{C}^{(r)}$ if $c_c$ meets
            $kb^{(t)}$ first, otherwise distribute only $\boldsymbol{\theta}_G$
 7:         Minimize Equation 2 for solving local CL problems
 8:         $\boldsymbol{B}_c^{(t,r)} \odot \boldsymbol{m}_c^{(t,r)}$ are transmitted from $\mathcal{C}^{(r)}$ to the server
 9:         Update $\boldsymbol{\theta}_G \leftarrow \frac{1}{|\mathcal{C}^{(r)}|} \sum_c \boldsymbol{B}_c^{(t,r)} \odot \boldsymbol{m}_c^{(t,r)}$
10:     end for
11:     Update knowledge base $kb \leftarrow kb \cup \{\boldsymbol{A}_j^{(t)}\}_{j \in \mathcal{C}}$
12: end for
```

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

3. Federated Architecture Dec.: FedWeIT

And finally, empirical comparison: “*We group 100 classes of CIFAR-100 dataset into 20 non-iid superclasses tasks. Then, we randomly sample 10 tasks out of 20 tasks & split instances to create a task sequence for each of the clients with overlapping tasks*”

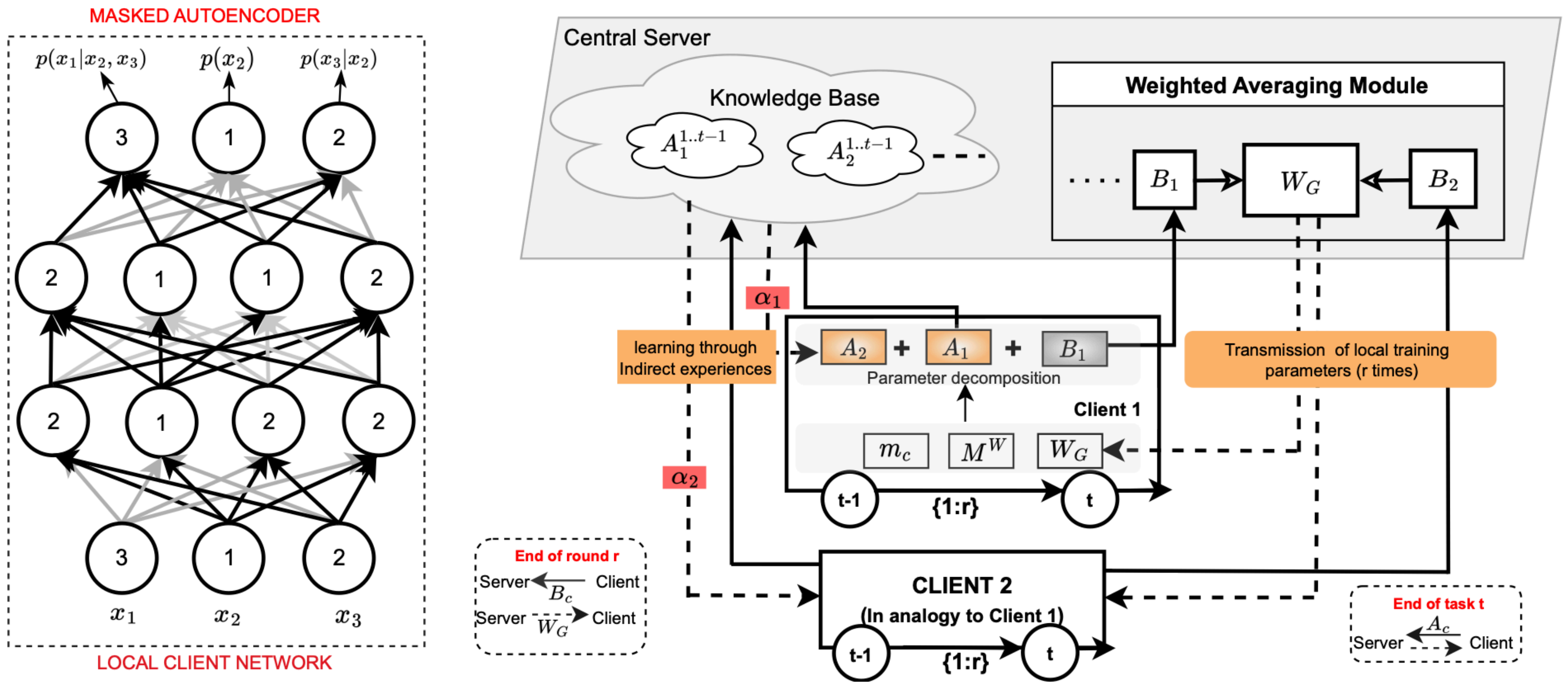


		Overlapped CIFAR-100 Dataset (F=1.0, R=20)				
Methods	FCL	Accuracy	Forgetting	Model Size	Client to Server Cost	Server to Client Cost
EWC (Kirkpatrick et al., 2017)	×	44.26 (± 0.53)	0.13 (± 0.01)	61 MB	N/A	N/A
Stable SGD (Mirzadeh et al., 2020)	×	43.31 (± 0.44)	0.08 (± 0.01)	61 MB	N/A	N/A
APD (Yoon et al., 2020)	×	50.82 (± 0.41)	0.02 (± 0.01)	73 MB	N/A	N/A
FedProx (Li et al., 2018)	✓	38.96 (± 0.37)	0.13 (± 0.02)	61 MB	1.22 GB	1.22 GB
Scaffold (Karimireddy et al., 2020)	✓	22.80 (± 0.47)	0.09 (± 0.01)	61 MB	2.44 GB	2.44 GB
FedCurv (Shoham et al., 2019)	✓	40.36 (± 0.44)	0.15 (± 0.02)	61 MB	1.22 GB	1.22 GB
FedProx-EWC	✓	41.53 (± 0.39)	0.13 (± 0.01)	61 MB	1.22 GB	1.22 GB
FedProx-Stable-SGD	✓	43.29 (± 1.45)	0.07 (± 0.01)	61 MB	1.22 GB	1.22 GB
FedProx-APD	✓	52.20 (± 0.50)	0.02 (± 0.01)	75 MB	1.22 GB	1.22 GB
FedWeIT (Ours)	✓	**55.16** (**± 0.19**)	**0.01** (**± 0.00**)	75 MB	0.37 GB	1.07 GB
Single Task Learning	×	57.15 (± 0.07)	–	610 MB	N/A	N/A

Yoon et al, “Federated Continual Learning with Weighted Inter-client Transfer”, ICML 2021

3. Federated Architecture Dec.: ConFedMADE

As in CL, Auto-encoders seem particularly useful because we can mask inherently. We won't go into full detail, but Paul has published on this :)

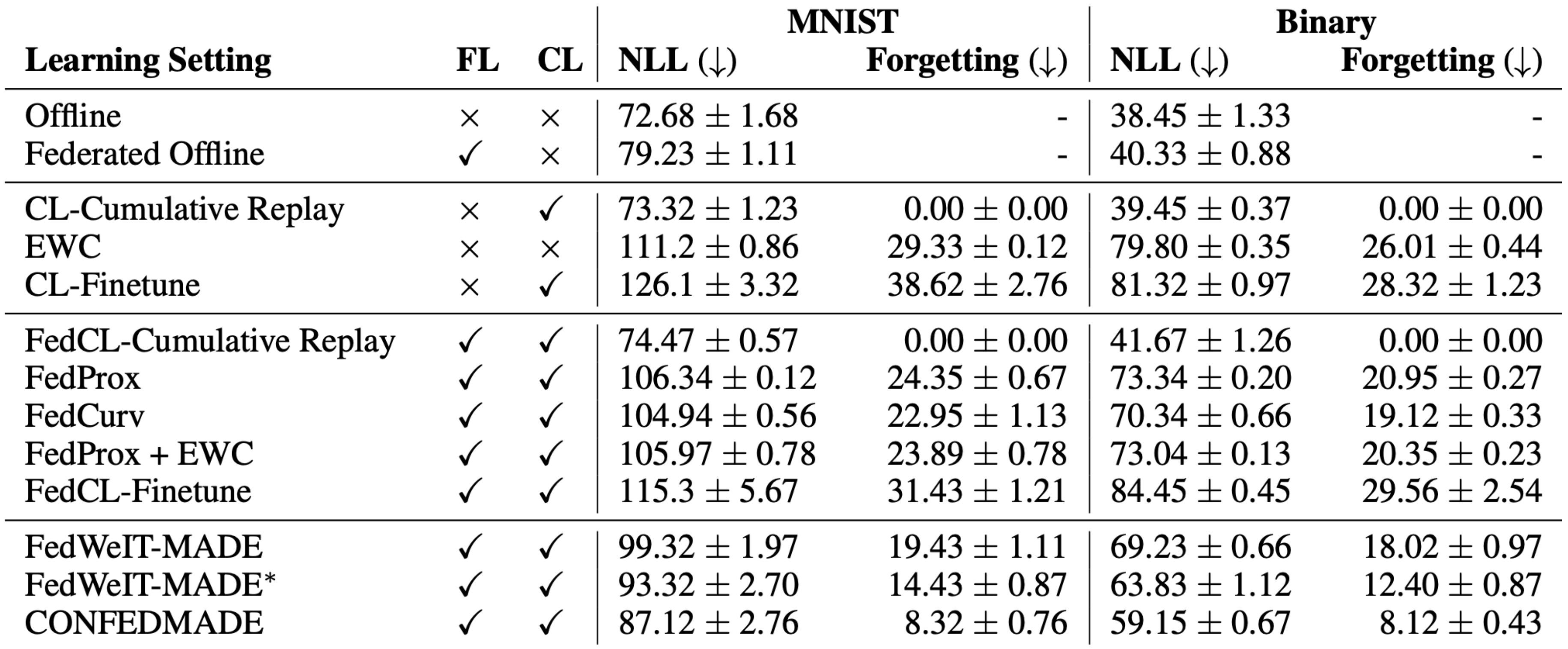


Paul et al, "Masked Autoencoders are Efficient Continual Federated Learners", CoLLAs 2024

Universität Bremen

3. Federated Architecture Dec.: ConFedMADE

This allows more efficient masking + enables unsupervised learning

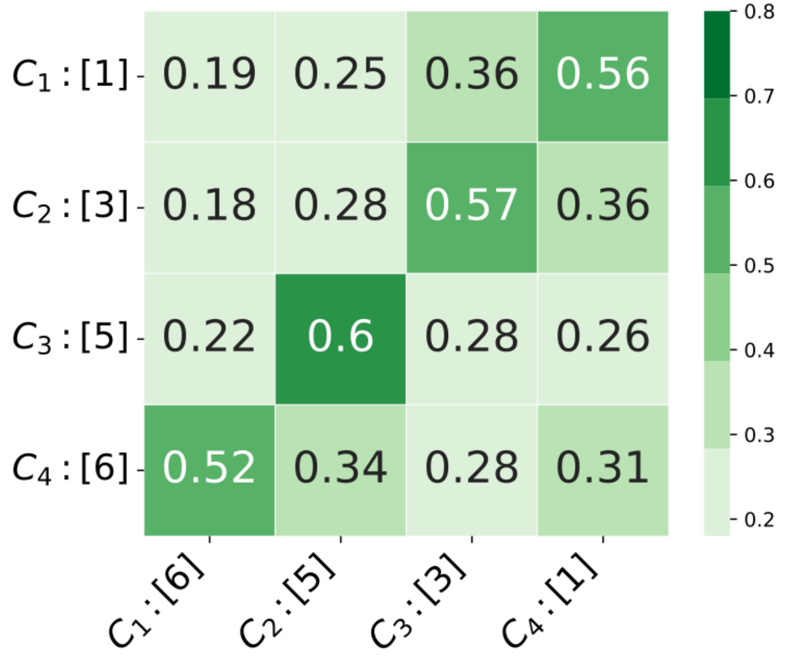


Learning Setting	FL	CL	MNIST		Binary	
			NLL (↓)	Forgetting (↓)	NLL (↓)	Forgetting (↓)
Offline	×	×	72.68 ± 1.68	-	38.45 ± 1.33	-
Federated Offline	✓	×	79.23 ± 1.11	-	40.33 ± 0.88	-
CL-Cumulative Replay	×	✓	73.32 ± 1.23	0.00 ± 0.00	39.45 ± 0.37	0.00 ± 0.00
EWC	×	×	111.2 ± 0.86	29.33 ± 0.12	79.80 ± 0.35	26.01 ± 0.44
CL-Finetune	×	✓	126.1 ± 3.32	38.62 ± 2.76	81.32 ± 0.97	28.32 ± 1.23
FedCL-Cumulative Replay	✓	✓	74.47 ± 0.57	0.00 ± 0.00	41.67 ± 1.26	0.00 ± 0.00
FedProx	✓	✓	106.34 ± 0.12	24.35 ± 0.67	73.34 ± 0.20	20.95 ± 0.27
FedCurv	✓	✓	104.94 ± 0.56	22.95 ± 1.13	70.34 ± 0.66	19.12 ± 0.33
FedProx + EWC	✓	✓	105.97 ± 0.78	23.89 ± 0.78	73.04 ± 0.13	20.35 ± 0.23
FedCL-Finetune	✓	✓	115.3 ± 5.67	31.43 ± 1.21	84.45 ± 0.45	29.56 ± 2.54
FedWeIT-MADE	✓	✓	99.32 ± 1.97	19.43 ± 1.11	69.23 ± 0.66	18.02 ± 0.97
FedWeIT-MADE*	✓	✓	93.32 ± 2.70	14.43 ± 0.87	63.83 ± 1.12	12.40 ± 0.87
CONFEDMADE	✓	✓	87.12 ± 2.76	8.32 ± 0.76	59.15 ± 0.67	8.12 ± 0.43

Paul et al, “Masked Autoencoders are Efficient Continual Federated Learners”, CoLLAs 2024

3. Federated Architecture Dec.: ConFedMADE

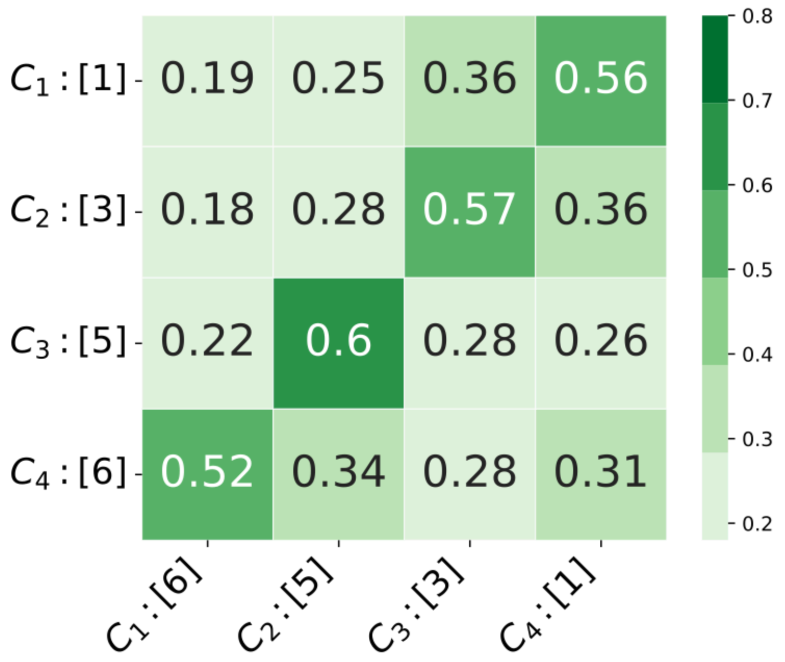
More importantly, let’s look at the intuition behind adaptive parameters on the server & “attention”, here for MNIST digits across clients with 2 tasks

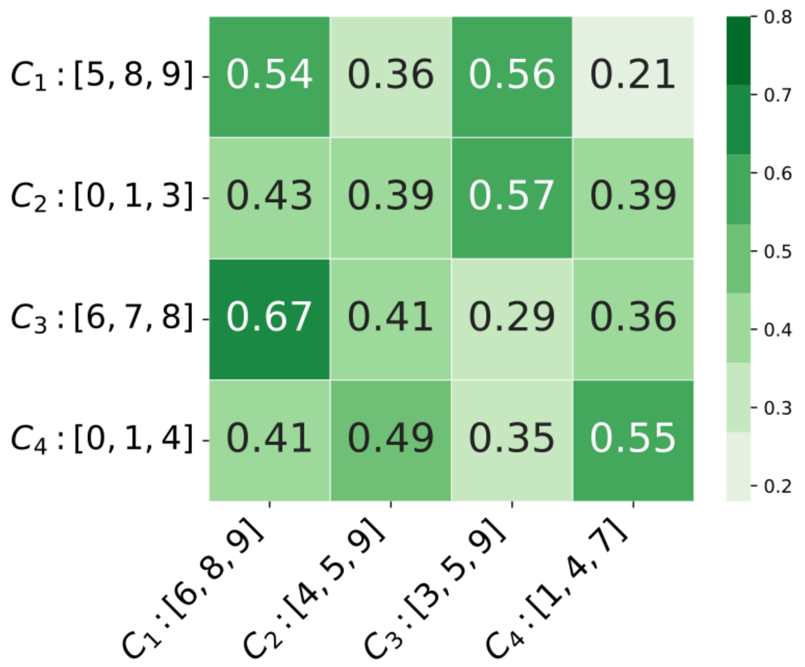


Paul et al, “Masked Autoencoders are Efficient Continual Federated Learners”, CoLLAs 2024

3. Federated Architecture Dec.: ConFedMADE

More importantly, let's look at the intuition behind adaptive parameters on the server & "attention", here for MNIST digits across clients with 2 tasks





Paul et al, "Masked Autoencoders are Efficient Continual Federated Learners", CoLLAs 2024

An important remark before moving on

We won’t consider all aspects relevant to federated learning to maintain the focus of the course on lifelong learning.

Specifically, we will not dive into more detail on various aspects that are critical to distributed/federated learning set-ups:

- Privacy & encryption techniques
- How to (further) reduce communication costs/overheads
- Client participation & client dropping
- Asynchronous set-ups
- Decentralized settings without a central/global server/model

(A)synchronous

To give an illustration why these matter:
here are examples of asynchronous

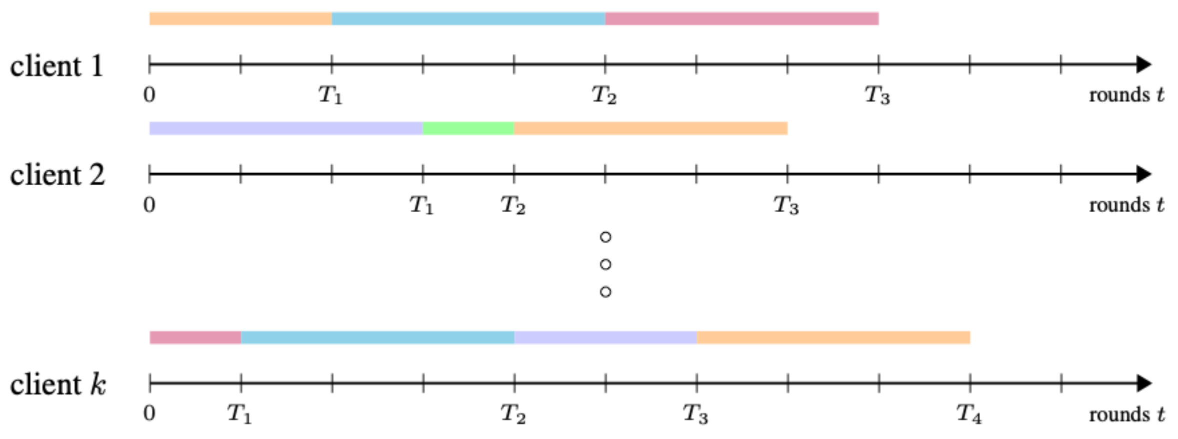


Figure 2. Example timeline showing the progression of the different tasks on the various clients. *(Best viewed in colors)*

Left Image from Shenaj et al, “Asynchronous Federated Continual Learning”, CVPR-W 2023
Right Image from ur Rehman et al, “FairFed: Cross-Device Fair Federated Learning”, IEEE AIPR 2020

(A)synchronous & (De)centralized

To give an illustration why these matter: here are examples of asynchronous & decentralized set-ups

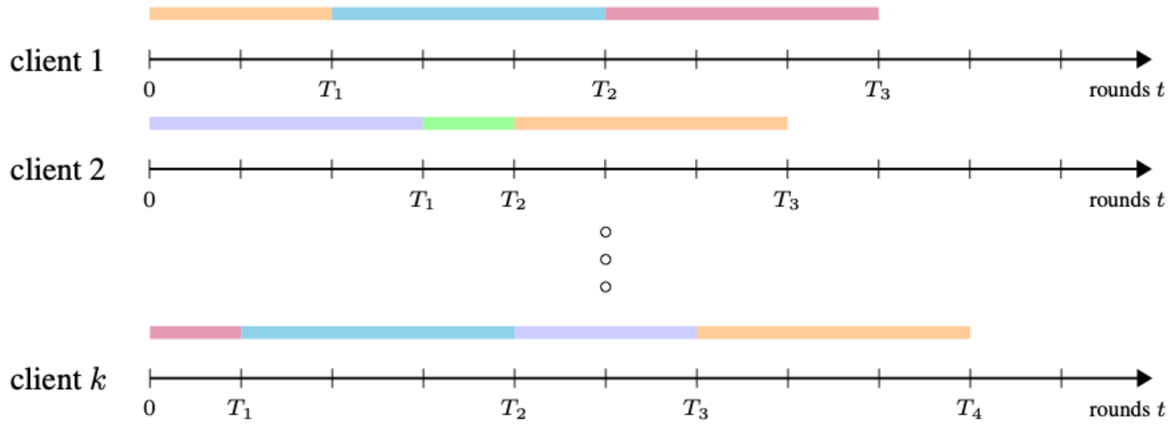
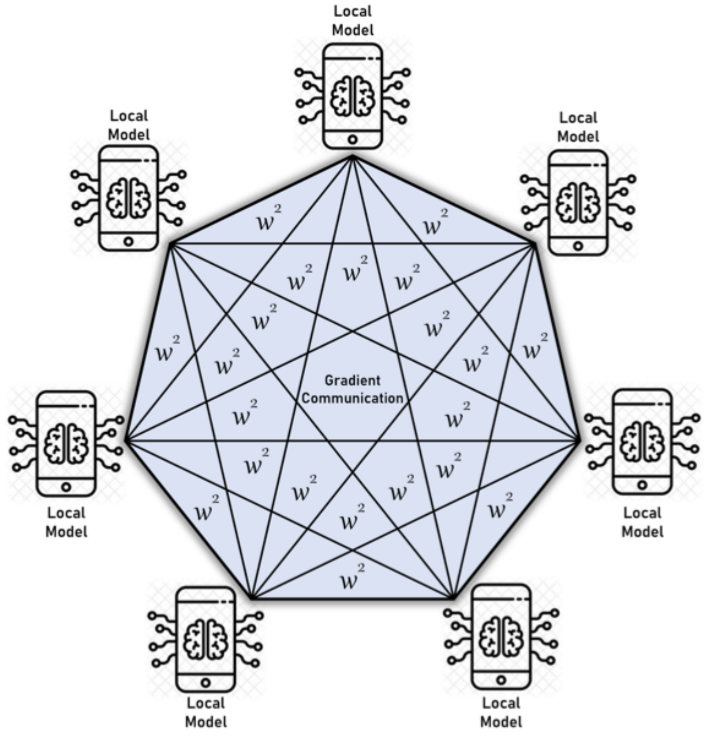


Figure 2. Example timeline showing the progression of the different tasks on the various clients. *(Best viewed in colors)*



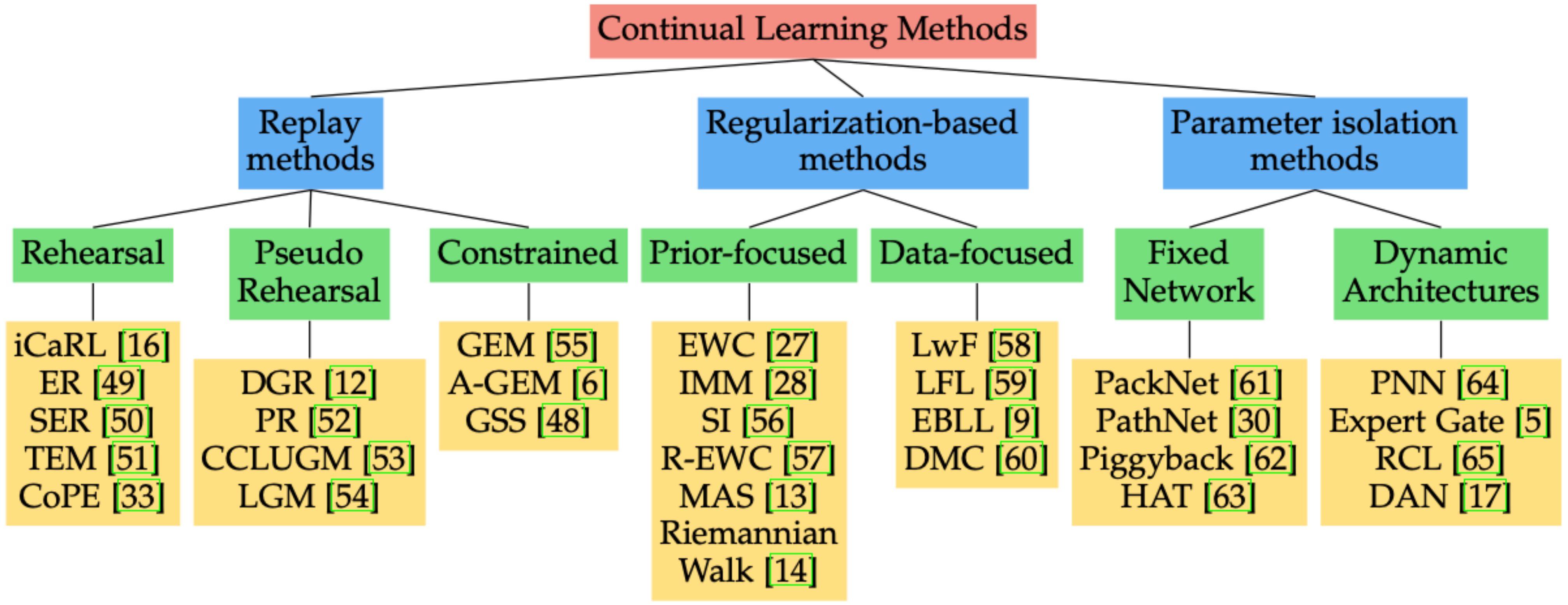
Left Image from Shenaj et al, "Asynchronous Federated Continual Learning", CVPR-W 2023
Right Image from ur Rehman et al, "FairFed: Cross-Device Fair Federated Learning", IEEE AIPR 2020

Question time

We have learned about a lot of settings for heterogeneous data (streams) & approaches to address challenges.
How do we evaluate & compare them?

Universität Bremen

Summary of “continual learning” methods


Continual Learning Methods
Replay methods
Regularization-based methods
Parameter isolation methods
Rehearsal
Pseudo Rehearsal
Constrained
Prior-focused
Data-focused
Fixed Network
Dynamic Architectures
iCaRL [16] ER [49] SER [50] TEM [51] CoPE [33]
DGR [12] PR [52] CCLUGM [53] LGM [54]
GEM [55] A-GEM [6] GSS [48]
EWC [27] IMM [28] SI [56] R-EWC [57] MAS [13] Riemannian Walk [14]
LwF [58] LFL [59] EBLL [9] DMC [60]
PackNet [61] PathNet [30] Piggyback [62] HAT [63]
PNN [64] Expert Gate [5] RCL [65] DAN [17]

De Lange et al, “A continual learning survey: Defying forgetting in classification tasks”, TPAMI 2021

How perspective influences evaluation

Rehearsal methods:
- What do you think matters for evaluating rehearsal?

Regularization methods:
- …

Architecture/“isolation” methods:
- …

Universität Bremen

How perspective influences evaluation

Rehearsal methods:

- Original data amount, generated data, (constant?) memory size, computational expense…

Regularization methods:

- What do you think matters for regularization approaches?

Architecture/“isolation” methods:

- …

How perspective influences evaluation

Rehearsal methods:
- Original data amount, generated data, (constant?) memory size, computational expense…

Regularization methods:
- Regularization strength (hyper-parameters), memory expense, computational expense…

Architecture/“isolation” methods:
- What do you think matters for the architecture perspective?

How perspective influences evaluation

Rehearsal methods:

- Original data amount, generated data, (constant?) memory size, computational expense…

Regularization methods:

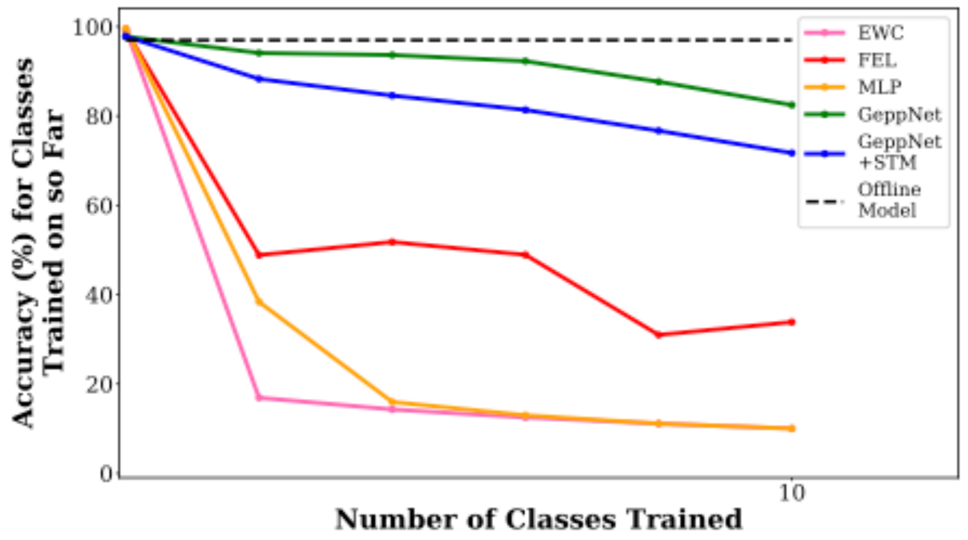
- Regularization strength (hyper-parameters), memory expense, computational expense…

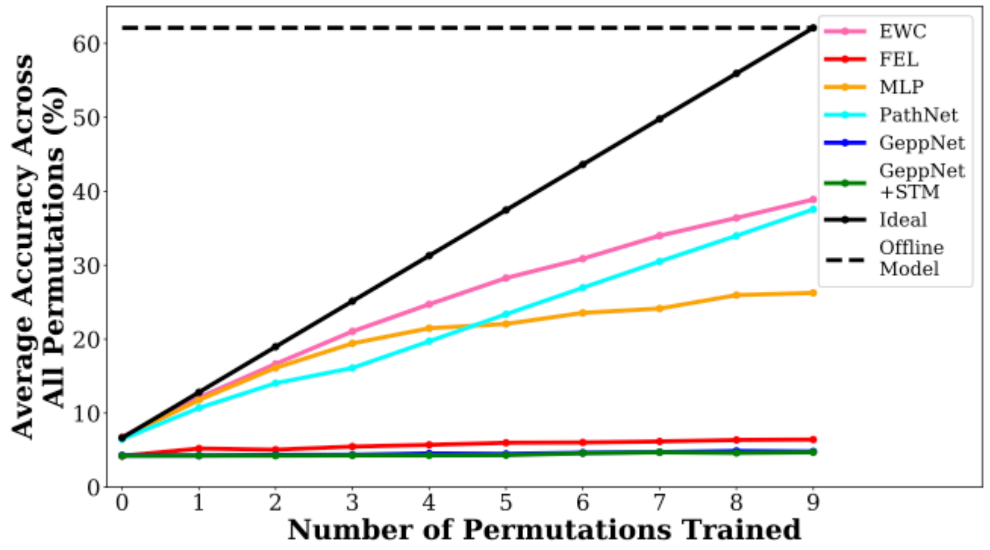
Architecture/“isolation” methods:

- Number of parameters, number of models, expert heads, memory expense, computational expense…

In general: final average loss is insufficient

Do we care about any-time performance or all task performance?





Kemker et al, “Measuring Catastrophic Forgetting in Neural Networks”, AAAI 2018

Always recommended: per “task” loss

- “**Base**” loss
  The initial (an old) task after i new experiences

$$\Omega_{base} = \frac{1}{T-1} \sum_{i=2}^{T} \frac{\alpha_{base,i}}{\alpha_{ideal}}$$

Kemker et al, “Measuring Catastrophic Forgetting in Neural Networks”, AAAI 2018

Always recommended: per “task” loss

- “**Base**” loss
  The initial (an old) task after i new experiences

$$\Omega_{base} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}}$$

- “**New**” loss
  The newest task exclusively

$$\Omega_{new} = \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i}$$

Always recommended: per "task" loss

- "**Base**" loss
  The initial (an old) task after i new experiences

$$\Omega_{base} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}}$$

- "**New**" loss
  The newest task exclusively

$$\Omega_{new} = \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i}$$

- "**All**" loss
  The average up to the present point in time

$$\Omega_{all} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{all,i}}{\alpha_{ideal}}$$

Always recommended: per "task" loss

- "**Base**" loss
  The initial (an old) task after i new experiences

$$\Omega_{base} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}}$$

- "**New**" loss
  The newest task exclusively

$$\Omega_{new} = \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i}$$

- "**All**" loss
  The average up to the present point in time

$$\Omega_{all} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{all,i}}{\alpha_{ideal}}$$

- "**Ideal**" loss
  The offline value when trained all at once

Kemker et al, "Measuring Catastrophic Forgetting in Neural Networks", AAAI 2018

Universität Bremen

Always recommended: per "task" loss

- "**Base**" loss
  The initial (an old) task after i new experiences

$$\Omega_{base} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}}$$

- "**New**" loss
  The newest task exclusively

$$\Omega_{new} = \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i}$$

- "**All**" loss
  The average up to the present point in time

$$\Omega_{all} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{all,i}}{\alpha_{ideal}}$$

- "**Ideal**" loss
  The offline value when trained all at once

How does this loss decomposition help quantify continual learning?

Kemker et al, "Measuring Catastrophic Forgetting in Neural Networks", AAAI 2018

Universität Bremen

Always recommended: per "task" loss

- "**Base**" loss -> measures **retention**
  The initial (an old) task after i new experiences

$$\Omega_{base} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}}$$

- "**New**" loss -> measures **plasticity**
  The newest task exclusively

$$\Omega_{new} = \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i}$$

- "**All**" loss -> measures **overall performance**
  The average up to the present point in time

$$\Omega_{all} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{all,i}}{\alpha_{ideal}}$$

- "**Ideal**" loss -> measures **achievable baseline**
  The offline value when trained all at once

How does this loss decomposition help quantify continual learning?

Kemker et al, "Measuring Catastrophic Forgetting in Neural Networks", AAAI 2018

Universität
Bremen

Useful relative quantities: “forgetting”

“We define forgetting for a particular task (or label) as the difference between the maximum knowledge gained about the task throughout the learning process in the past and the knowledge the model currently has about it.”

For the j-th task after being trained up to task k > j:

$$f_j^k = \max_{l \in \{1, \cdots, k-1\}} a_{l,j} - a_{k,j} \ , \quad \forall j < k$$

Chaudhry et al, “Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence”, ECCV 2018

Useful relative quantities: “intransigence”

“We define intransigence as the inability of a model to learn new tasks. Since we wish to quantify the inability to learn, we compare to the standard classification model which has access to all the datasets at all times”

For a reference model for task k (denoted by *):

$$I_k = a_k^* - a_{k,k}$$

Chaudhry et al, “Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence”, ECCV 2018

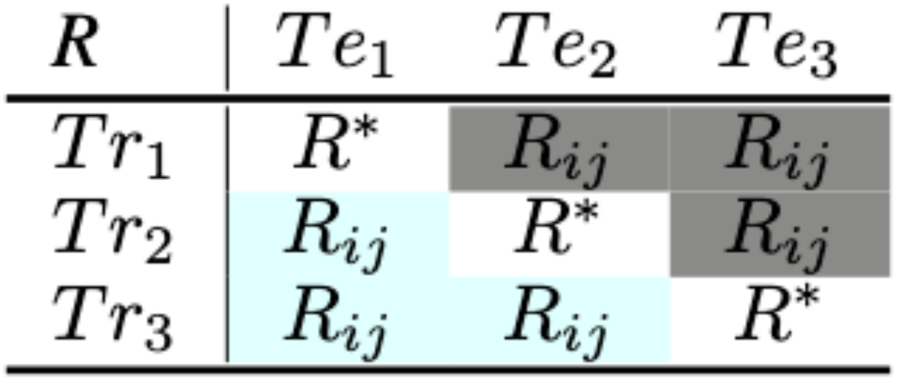
Universität Bremen

Measuring directions of transfer

Forward transfer (with random baseline b): influence of a learning task on future:

$$\mathrm{FWT}_{t,j} = a_{t-1,j} - \bar{b}_j \qquad \mathrm{FWT}_t = \frac{1}{t-1}\sum_{j=2}^{t-1} \mathrm{FWT}_{j-1,j}$$

Which part of the "task" matrix quantifies forward transfer?



R	Te_1	Te_2	Te_3
Tr_1	R^*	R_{ij}	R_{ij}
Tr_2	R_{ij}	R^*	R_{ij}
Tr_3	R_{ij}	R_{ij}	R^*

Lopez-Paz & Ranzato, "Gradient Episodic Memory for Continual Learning", 2017,
See also: Díaz-Rodríguez & Lomonaco et al, "Don't forget, there is more than forgetting: new metrics for Continual Learning", 2018

Measuring directions of transfer

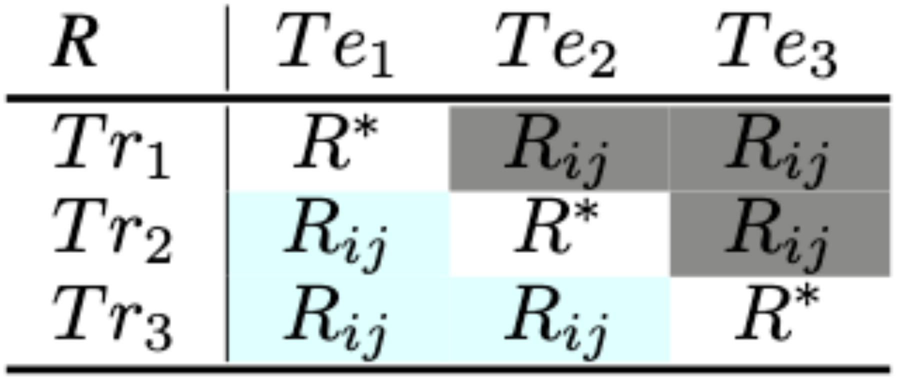
Forward transfer (with random baseline b): influence of a learning task on future:

$$\mathrm{FWT}_{t,j} = a_{t-1,j} - \bar{b}_j \qquad \mathrm{FWT}_t = \frac{1}{t-1}\sum_{j=2}^{t-1} \mathrm{FWT}_{j-1,j}$$

Backward transfer: influence of a task on previous tasks; negative = forgetting, positive = retrospective improvement

$$\mathrm{BWT}_{t,j} = a_{t,j} - a_{j,j} \qquad \mathrm{BWT}_t = \frac{1}{t-1}\sum_{j=1}^{t-1} \mathrm{BWT}_{t,j}$$

Which part of the "task" matrix quantifies backward transfer?



R	Te_1	Te_2	Te_3
Tr_1	R^*	R_{ij}	R_{ij}
Tr_2	R_{ij}	R^*	R_{ij}
Tr_3	R_{ij}	R_{ij}	R^*

Lopez-Paz & Ranzato, "Gradient Episodic Memory for Continual Learning", 2017,
See also: Díaz-Rodríguez & Lomonaco et al, "Don't forget, there is more than forgetting: new metrics for Continual Learning", 2018

Generalizing forward transfer

b-shot performance (b = mini-batch) after the model has been trained on all tasks T:

$$Z_b = \frac{1}{T}\sum_{k=1}^{T} a_{k,b,k}$$

Universität Bremen

Generalizing forward transfer

b-shot performance (b = mini-batch) after the model has been trained on all tasks T:

$$Z_b = \frac{1}{T} \sum_{k=1}^{T} a_{k,b,k}$$

Learning Curve Area (LCA) at beta is the area of the convergence curve Z as a function of b in [0, beta]:

$$\mathrm{LCA}_\beta = \frac{1}{\beta+1} \int_0^\beta Z_b db = \frac{1}{\beta+1} \sum_{b=0}^{\beta} Z_b$$

What does beta = 0 correspond to?

Universität Bremen

Generalizing forward transfer

b-shot performance (b = mini-batch) after the model has been trained on all tasks T:

$$Z_b = \frac{1}{T}\sum_{k=1}^{T} a_{k,b,k}$$

Learning Curve Area (LCA) at beta is the area of the convergence curve Z as a function of b in [0, beta]:

$$\mathrm{LCA}_\beta = \frac{1}{\beta+1}\int_0^\beta Z_b db = \frac{1}{\beta+1}\sum_{b=0}^{\beta} Z_b$$

What does beta = 0 correspond to?

Beta = 0 is equivalent to zero-shot performance == forward transfer

Chaudhry et al, “Efficient Lifelong Learning with A-GEM”, ICLR 2019

And various metrics for memory, size, compute

We can construct similar measures for memory, size & compute
(Here tasks are called N, to conform with the original paper's notation)

$$CE = min(1, \frac{\sum_{i=1}^{N} \frac{Ops\uparrow\downarrow(Tr_i) \cdot \varepsilon}{Ops(Tr_i)}}{N})$$

Computational Efficiency

Quantifies add/multiply ops

(inference & updates)

Díaz-Rodríguez & Lomonaco et al, "Don't forget, there is more than forgetting: new metrics for Continual Learning", 2018

Universität Bremen

And various metrics for memory, size, compute

We can construct similar measures for memory, size & compute
(Here tasks are called N, to conform with the original paper's notation)

$$CE = min(1, \frac{\sum_{i=1}^{N} \frac{Ops\uparrow\downarrow(Tr_i)\cdot\varepsilon}{Ops(Tr_i)}}{N})$$

Computational Efficiency

Quantifies add/multiply ops (inference & updates)

$$MS = min(1, \frac{\sum_{i=1}^{N} \frac{Mem(\theta_1)}{Mem(\theta_i)}}{N})$$

Model Size Efficiency

Quantifies parameter growth

Díaz-Rodríguez & Lomonaco et al, "Don't forget, there is more than forgetting: new metrics for Continual Learning", 2018

Universität Bremen

And various metrics for memory, size, compute

We can construct similar measures for memory, size & compute
(Here tasks are called N, to conform with the original paper's notation)

$$CE = min(1, \frac{\sum_{i=1}^{N} \frac{Ops\uparrow\downarrow(Tr_i)\cdot\varepsilon}{Ops(Tr_i)}}{N})$$

Computational Efficiency

Quantifies add/multiply ops (inference & updates)

$$MS = min(1, \frac{\sum_{i=1}^{N} \frac{Mem(\theta_1)}{Mem(\theta_i)}}{N})$$

Model Size Efficiency

Quantifies parameter growth

$$SSS = 1 - min(1, \frac{\sum_{i=1}^{N} \frac{Mem(M_i)}{Mem(D)}}{N})$$

Sample Storage Size Efficiency

Quantifies stored amount of data (for rehearsal)

Díaz-Rodríguez & Lomonaco et al, "Don't forget, there is more than forgetting: new metrics for Continual Learning", 2018

Question time

There are plenty of other interesting aspects to measure - perhaps you can think of some.
But importantly, how should we report & compare?

Universität Bremen

Question time

There are plenty of other interesting aspects to measure - perhaps you can think of some.
But importantly, how should we report & compare?

This is the one point in the lecture, where I/we don’t have answers yet. But we can look at current practice

First: a (philosophical?) question

Should we opt to maximize one metric - and potentially risk drawing too general conclusions from too limited evidence?

For instance: “x is a good continual learning algorithm”

Universität Bremen

First: a (philosophical?) question

Should we opt to maximize one metric - and potentially risk drawing too general conclusions from too limited evidence?

For instance: “x is a good continual learning algorithm”

Or should we report as exhaustively as possible - and potentially risk comparison being near impossible?

For instance: “all algorithms have value in some niche dimension”

Universität Bremen

1. Challenges when improving a single metric

Recall “task-incremental” set-up, expert heads etc. (here in VCL)

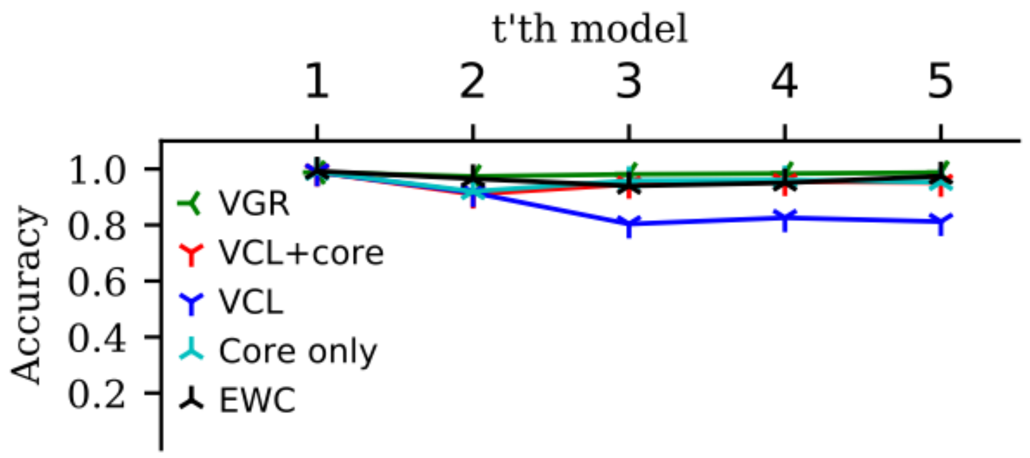


Figure 5. **Multi-headed Split FashionMNIST.**

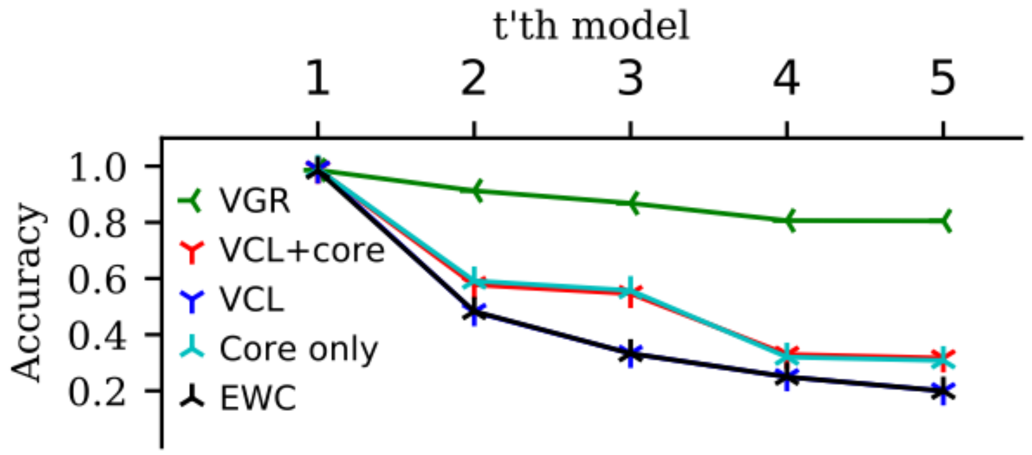


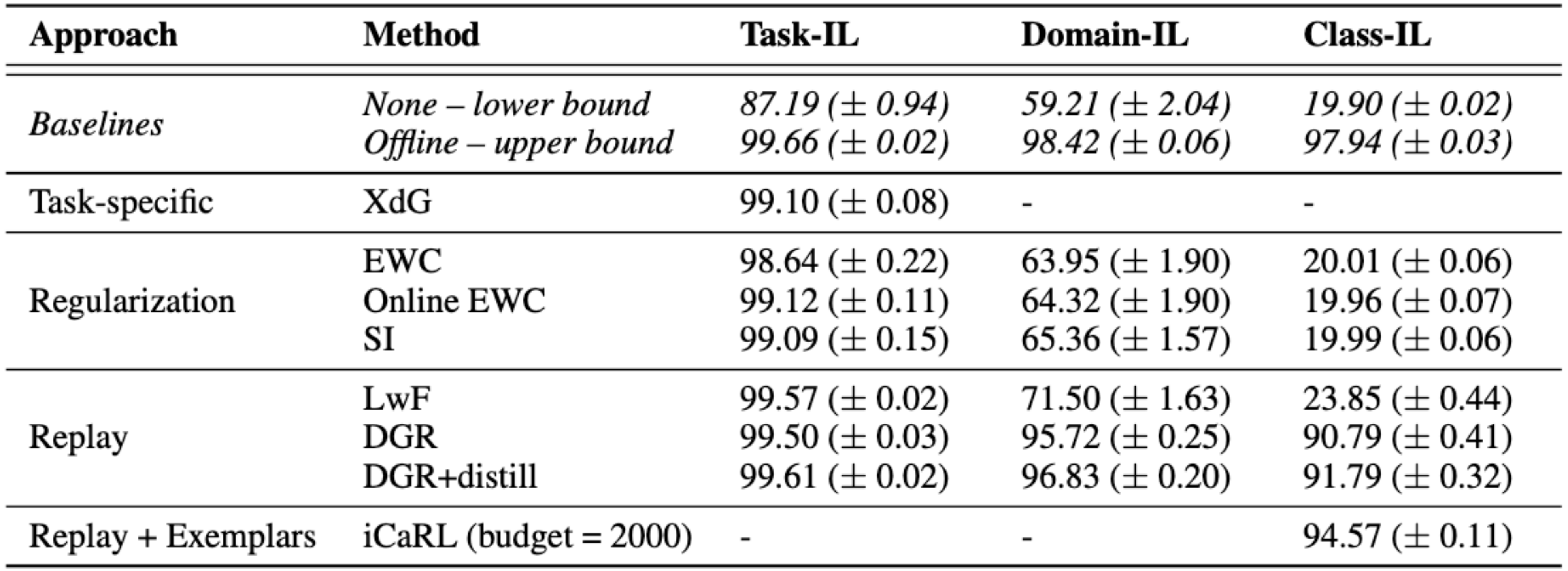
Figure 3. **Single-headed Split Fashion MNIST.**

Farquhar & Gal, “Towards Robust Evaluations of Continual Learning”, Lifelong Learning workshop at ICML 2018

Universität
Bremen

1. Challenges when improving a single metric

Recall replay being “best”, when only considering forgetting

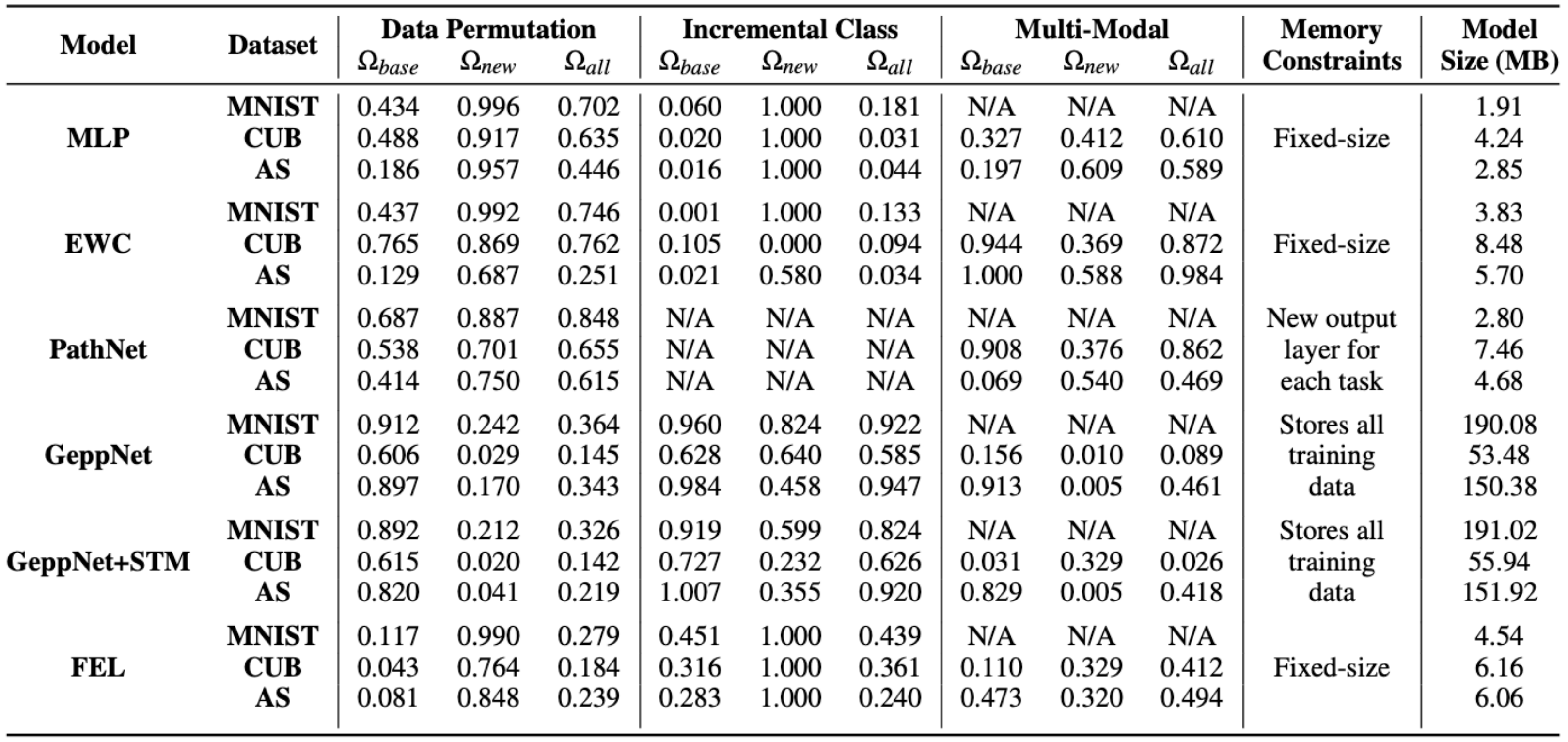


Approach	**Method**	**Task-IL**	**Domain-IL**	**Class-IL**
Baselines	*None – lower bound*	*87.19 (± 0.94)*	*59.21 (± 2.04)*	*19.90 (± 0.02)*
	Offline – upper bound	*99.66 (± 0.02)*	*98.42 (± 0.06)*	*97.94 (± 0.03)*
Task-specific	XdG	99.10 (± 0.08)	-	-
Regularization	EWC	98.64 (± 0.22)	63.95 (± 1.90)	20.01 (± 0.06)
	Online EWC	99.12 (± 0.11)	64.32 (± 1.90)	19.96 (± 0.07)
	SI	99.09 (± 0.15)	65.36 (± 1.57)	19.99 (± 0.06)
Replay	LwF	99.57 (± 0.02)	71.50 (± 1.63)	23.85 (± 0.44)
	DGR	99.50 (± 0.03)	95.72 (± 0.25)	90.79 (± 0.41)
	DGR+distill	99.61 (± 0.02)	96.83 (± 0.20)	91.79 (± 0.32)
Replay + Exemplars	iCaRL (budget = 2000)	-	-	94.57 (± 0.11)

van de Ven et al , “Three types of incremental learning”, Nature MI 4, 2022

2. Challenges when reporting exhaustively

How do we compare approaches with different assumptions?

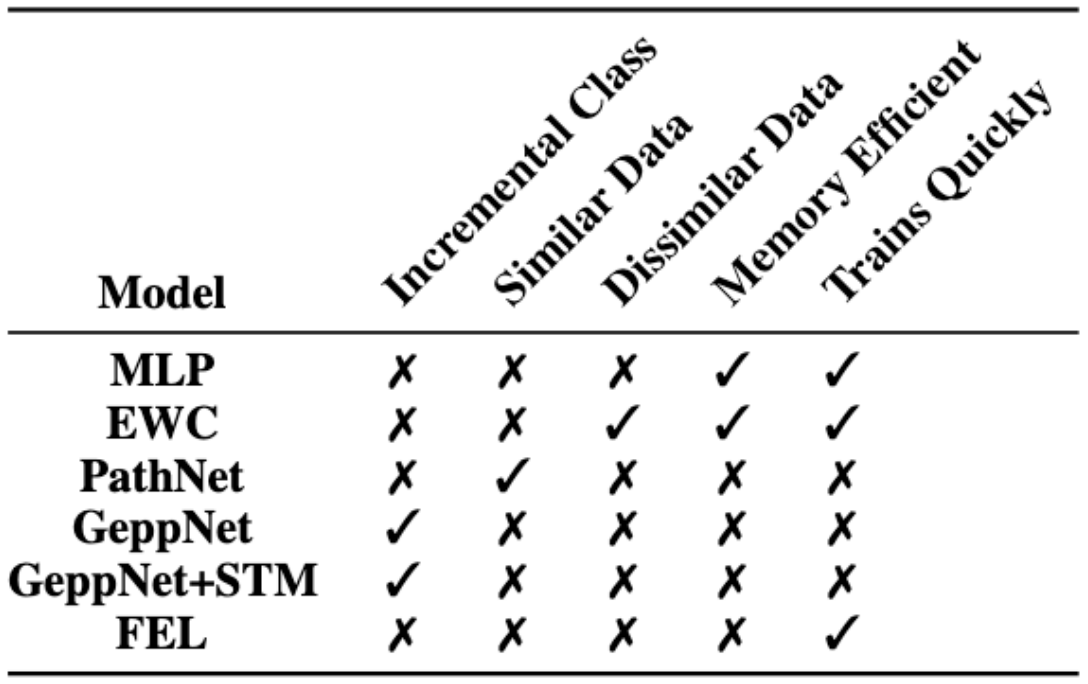


Model	Dataset	Data Permutation			Incremental Class			Multi-Modal			Memory Constraints	Model Size (MB)
		Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}		
MLP	**MNIST**	0.434	0.996	0.702	0.060	1.000	0.181	N/A	N/A	N/A	Fixed-size	1.91
	CUB	0.488	0.917	0.635	0.020	1.000	0.031	0.327	0.412	0.610		4.24
	AS	0.186	0.957	0.446	0.016	1.000	0.044	0.197	0.609	0.589		2.85
EWC	**MNIST**	0.437	0.992	0.746	0.001	1.000	0.133	N/A	N/A	N/A	Fixed-size	3.83
	CUB	0.765	0.869	0.762	0.105	0.000	0.094	0.944	0.369	0.872		8.48
	AS	0.129	0.687	0.251	0.021	0.580	0.034	1.000	0.588	0.984		5.70
PathNet	**MNIST**	0.687	0.887	0.848	N/A	N/A	N/A	N/A	N/A	N/A	New output layer for each task	2.80
	CUB	0.538	0.701	0.655	N/A	N/A	N/A	0.908	0.376	0.862		7.46
	AS	0.414	0.750	0.615	N/A	N/A	N/A	0.069	0.540	0.469		4.68
GeppNet	**MNIST**	0.912	0.242	0.364	0.960	0.824	0.922	N/A	N/A	N/A	Stores all training data	190.08
	CUB	0.606	0.029	0.145	0.628	0.640	0.585	0.156	0.010	0.089		53.48
	AS	0.897	0.170	0.343	0.984	0.458	0.947	0.913	0.005	0.461		150.38
GeppNet+STM	**MNIST**	0.892	0.212	0.326	0.919	0.599	0.824	N/A	N/A	N/A	Stores all training data	191.02
	CUB	0.615	0.020	0.142	0.727	0.232	0.626	0.031	0.329	0.026		55.94
	AS	0.820	0.041	0.219	1.007	0.355	0.920	0.829	0.005	0.418		151.92
FEL	**MNIST**	0.117	0.990	0.279	0.451	1.000	0.439	N/A	N/A	N/A	Fixed-size	4.54
	CUB	0.043	0.764	0.184	0.316	1.000	0.361	0.110	0.329	0.412		6.16
	AS	0.081	0.848	0.239	0.283	1.000	0.240	0.473	0.320	0.494		6.06

Kemker et al, “Measuring Catastrophic Forgetting in Neural Networks”, AAAI 2018

2. Challenges when reporting exhaustively

What do we conclude when performance varies across set-ups?

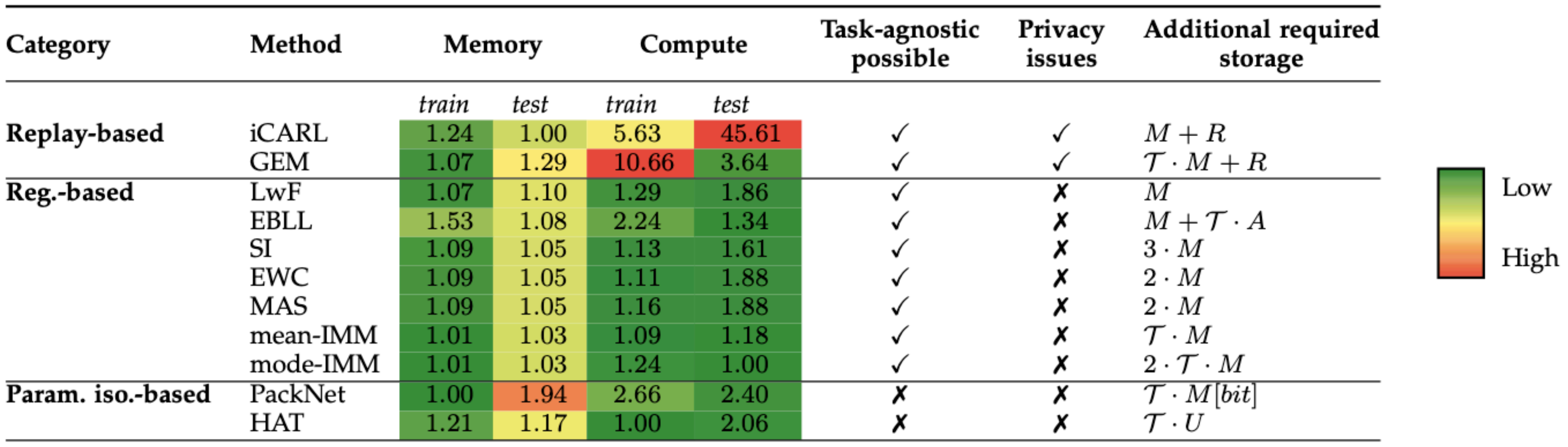


Model	Incremental Class	Similar Data	Dissimilar Data	Memory Efficient	Trains Quickly
MLP	✗	✗	✗	✓	✓
EWC	✗	✗	✓	✓	✓
PathNet	✗	✓	✗	✗	✗
GeppNet	✓	✗	✗	✗	✗
GeppNet+STM	✓	✗	✗	✗	✗
FEL	✗	✗	✗	✗	✓

Kemker et al, “Measuring Catastrophic Forgetting in Neural Networks”, AAAI 2018

2. Challenges when reporting exhaustively

How do we weight the different trade-offs that algorithms make? What is & isn't acceptable, what makes for a "better" algorithm?

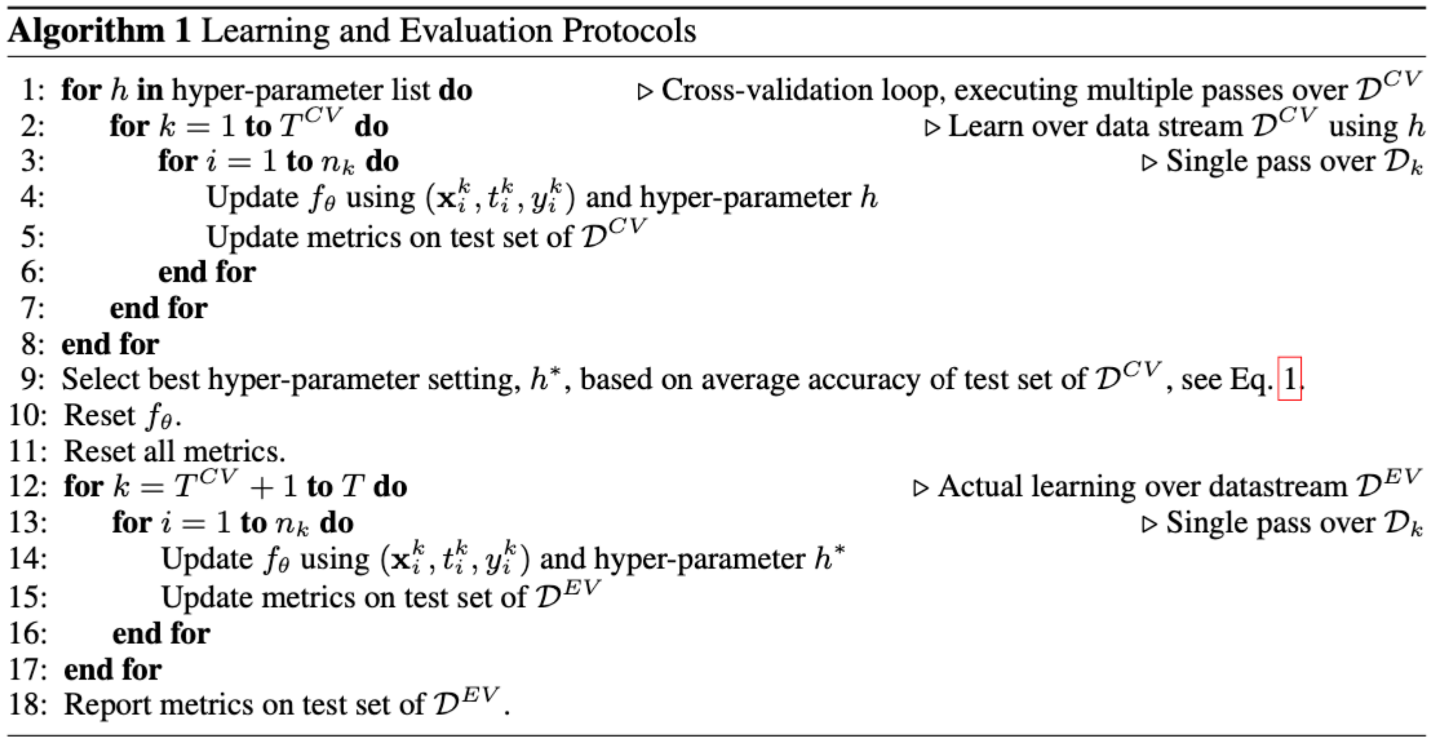


Category	Method	Memory		Compute		Task-agnostic possible	Privacy issues	Additional required storage
		train	*test*	*train*	*test*			
Replay-based	iCARL	1.24	1.00	5.63	45.61	✓	✓	$M + R$
	GEM	1.07	1.29	10.66	3.64	✓	✓	$\mathcal{T} \cdot M + R$
Reg.-based	LwF	1.07	1.10	1.29	1.86	✓	✗	M
	EBLL	1.53	1.08	2.24	1.34	✓	✗	$M + \mathcal{T} \cdot A$
	SI	1.09	1.05	1.13	1.61	✓	✗	$3 \cdot M$
	EWC	1.09	1.05	1.11	1.88	✓	✗	$2 \cdot M$
	MAS	1.09	1.05	1.16	1.88	✓	✗	$2 \cdot M$
	mean-IMM	1.01	1.03	1.09	1.18	✓	✗	$\mathcal{T} \cdot M$
	mode-IMM	1.01	1.03	1.24	1.00	✓	✗	$2 \cdot \mathcal{T} \cdot M$
Param. iso.-based	PackNet	1.00	1.94	2.66	2.40	✗	✗	$\mathcal{T} \cdot M[bit]$
	HAT	1.21	1.17	1.00	2.06	✗	✗	$\mathcal{T} \cdot U$

De Lange et al, "A continual learning survey: Defying forgetting in classification tasks", TPAMI 2021

3. Challenges surrounding hyper-parameters

And finally, how to select hyper-parameters in continual learning?



Algorithm 1 Learning and Evaluation Protocols

```
1:  for $h$ in hyper-parameter list do        ▷ Cross-validation loop, executing multiple passes over $\mathcal{D}^{CV}$
2:      for $k = 1$ to $T^{CV}$ do        ▷ Learn over data stream $\mathcal{D}^{CV}$ using $h$
3:          for $i = 1$ to $n_k$ do        ▷ Single pass over $\mathcal{D}_k$
4:              Update $f_\theta$ using $(\mathbf{x}_i^k, t_i^k, y_i^k)$ and hyper-parameter $h$
5:              Update metrics on test set of $\mathcal{D}^{CV}$
6:          end for
7:      end for
8:  end for
9:  Select best hyper-parameter setting, $h^*$, based on average accuracy of test set of $\mathcal{D}^{CV}$, see Eq. 1.
10: Reset $f_\theta$.
11: Reset all metrics.
12: for $k = T^{CV} + 1$ to $T$ do        ▷ Actual learning over datastream $\mathcal{D}^{EV}$
13:     for $i = 1$ to $n_k$ do        ▷ Single pass over $\mathcal{D}_k$
14:         Update $f_\theta$ using $(\mathbf{x}_i^k, t_i^k, y_i^k)$ and hyper-parameter $h^*$
15:         Update metrics on test set of $\mathcal{D}^{EV}$
16:     end for
17: end for
18: Report metrics on test set of $\mathcal{D}^{EV}$.
```

Chaudhry et al, “Efficient Lifelong Learning with A-GEM”, ICLR 2019

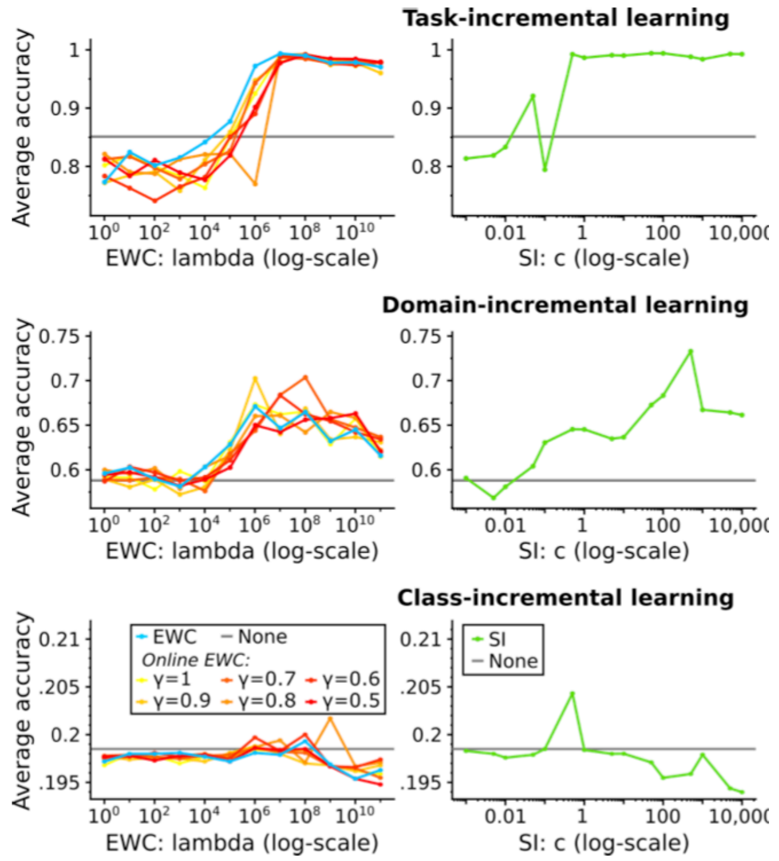
Universität Bremen

3. Challenges surrounding hyper-parameters

Recall: plasticity-sensitivity trade-off (algorithms such as EWC, SI…)

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

Not only hard to chose approximation & hyper-param, but the setting also influences the choice significantly



Chaudhry et al, “Efficient Lifelong Learning with A-GEM”, ICLR 2019

Question time

In your opinion, is the "best" CL algorithm the one

- *with the least assumptions?*
- *that is least hyper parameter sensitive?*
- *the most widely applicable?*
- *that beats all others in some application?*
- *…. ?*

The quest for consensus & general desiderata?

Some suggestions in the literature:

- Cross-task resemblance
- Shared output head
- No test time task labels
- No unconstrained re-training on old tasks
- More than two tasks

Farquhar & Gal, “Towards Robust Evaluations in Continual Learning”, ICML Lifelong Learning Workshop, 2018

Universität Bremen

The quest for consensus & general desiderata?

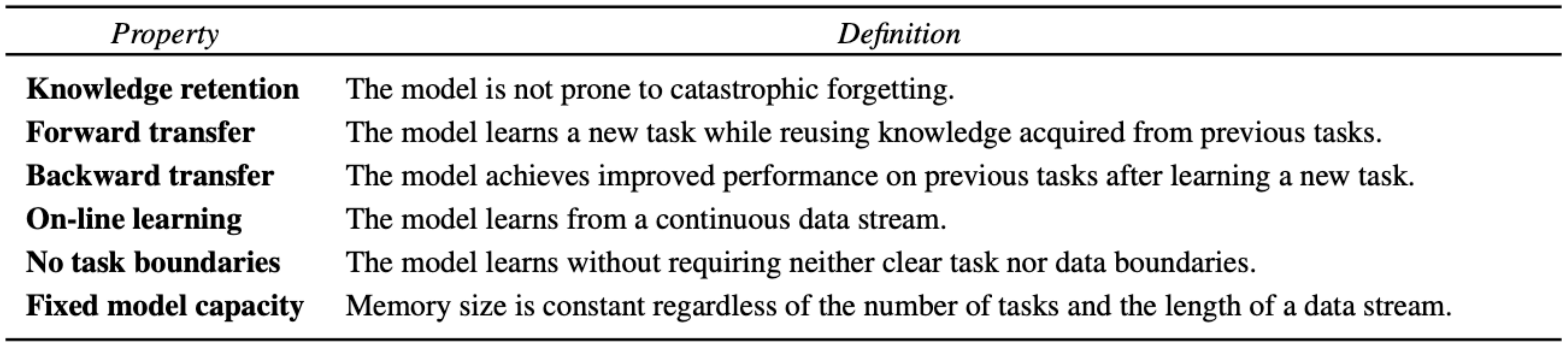
Some suggestions in the literature:

- Cross-task resemblance
- Shared output head
- No test time task labels
- No unconstrained re-training on old tasks
- More than two tasks

And also questions: unclear task boundaries, continuous tasks, overlapping vs. disjoint tasks, long task sequences, time/compute/memory constraints, strict privacy guarantees…

Farquhar & Gal, “Towards Robust Evaluations in Continual Learning”, ICML Lifelong Learning Workshop, 2018

The quest for consensus & general desiderata?



Property	*Definition*
Knowledge retention	The model is not prone to catastrophic forgetting.
Forward transfer	The model learns a new task while reusing knowledge acquired from previous tasks.
Backward transfer	The model achieves improved performance on previous tasks after learning a new task.
On-line learning	The model learns from a continuous data stream.
No task boundaries	The model learns without requiring neither clear task nor data boundaries.
Fixed model capacity	Memory size is constant regardless of the number of tasks and the length of a data stream.

Table 1: Desiderata of continual learning.

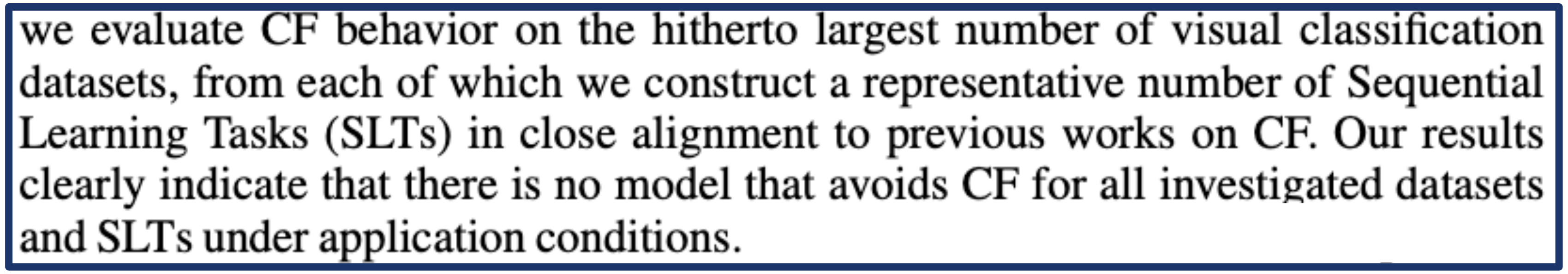
Biesialska et al, “Continual Learning in Natural Language Processing: A Survey”, COLING 2020

Universität Bremen

Question time

Do you agree with all of these desired properties?

Reproducibility crisis: continual ML

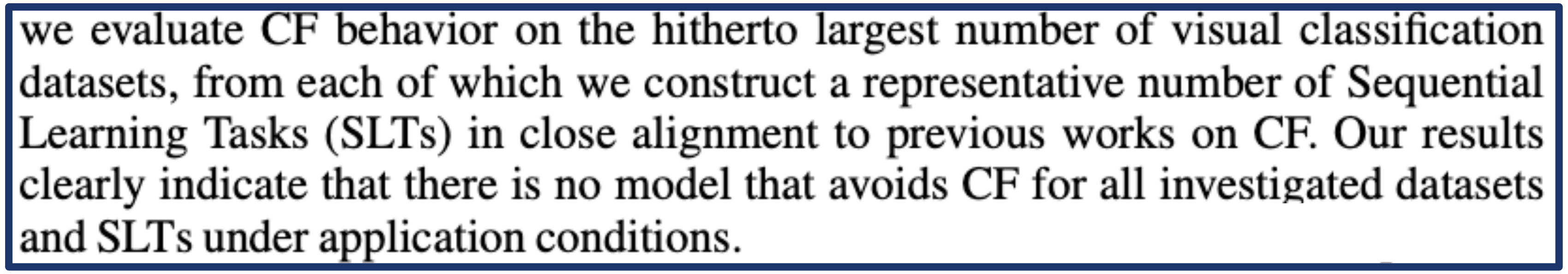

we evaluate CF behavior on the hitherto largest number of visual classification datasets, from each of which we construct a representative number of Sequential Learning Tasks (SLTs) in close alignment to previous works on CF. Our results clearly indicate that there is no model that avoids CF for all investigated datasets and SLTs under application conditions.

"A comprehensive, application-oriented study of catastrophic forgetting in DNNs", Pfuelb & Gepperth, ICLR 2019

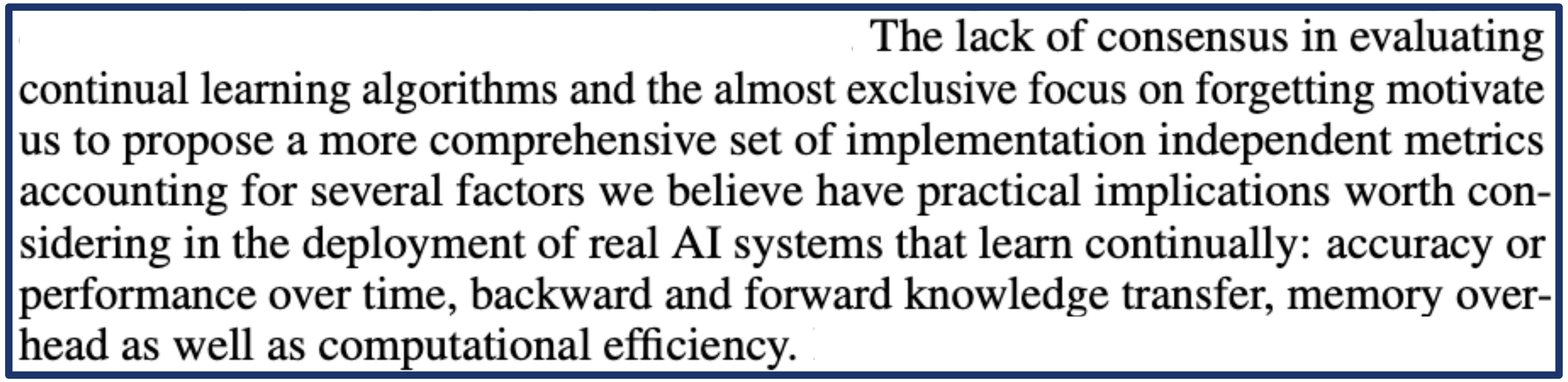
Universität Bremen

OWL ML

Reproducibility crisis: continual ML


we evaluate CF behavior on the hitherto largest number of visual classification datasets, from each of which we construct a representative number of Sequential Learning Tasks (SLTs) in close alignment to previous works on CF. Our results clearly indicate that there is no model that avoids CF for all investigated datasets and SLTs under application conditions.

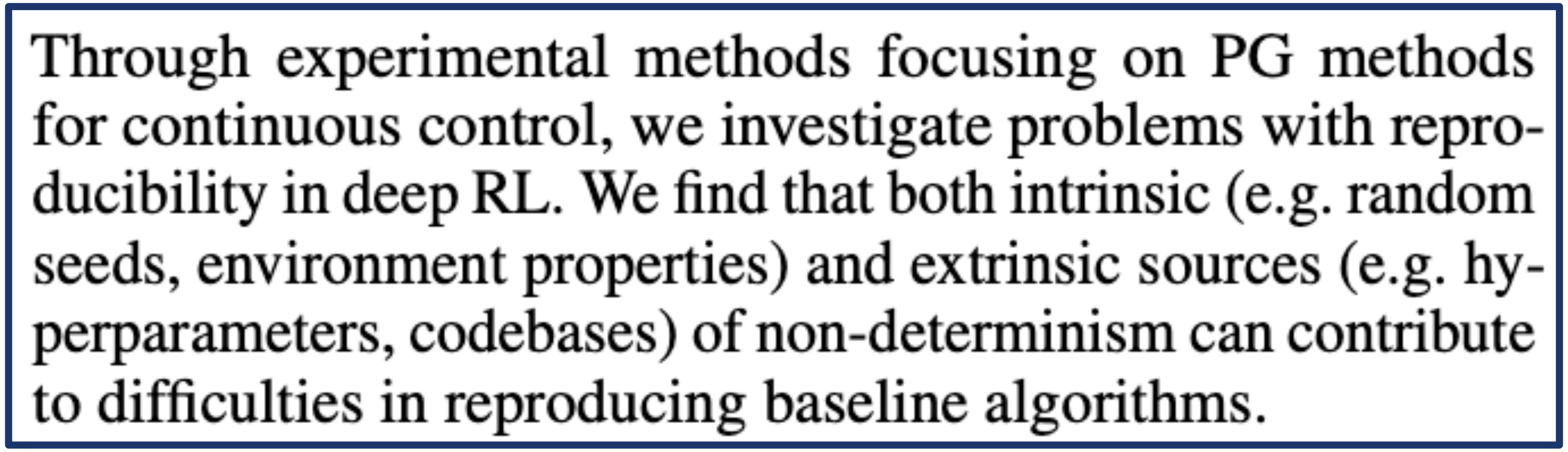
"A comprehensive, application-oriented study of catastrophic forgetting in DNNs", Pfuelb & Gepperth, ICLR 2019


The lack of consensus in evaluating continual learning algorithms and the almost exclusive focus on forgetting motivate us to propose a more comprehensive set of implementation independent metrics accounting for several factors we believe have practical implications worth considering in the deployment of real AI systems that learn continually: accuracy or performance over time, backward and forward knowledge transfer, memory overhead as well as computational efficiency.

"Don't forget, there is more than forgetting: new metrics for Continual Learning", Díaz-Rodríguez et al, Continual Learning Workshop at NeurIPS 2018

Universität Bremen

Reproducibility crisis: ~~continual~~ ML


Through experimental methods focusing on PG methods for continuous control, we investigate problems with reproducibility in deep RL. We find that both intrinsic (e.g. random seeds, environment properties) and extrinsic sources (e.g. hyperparameters, codebases) of non-determinism can contribute to difficulties in reproducing baseline algorithms.

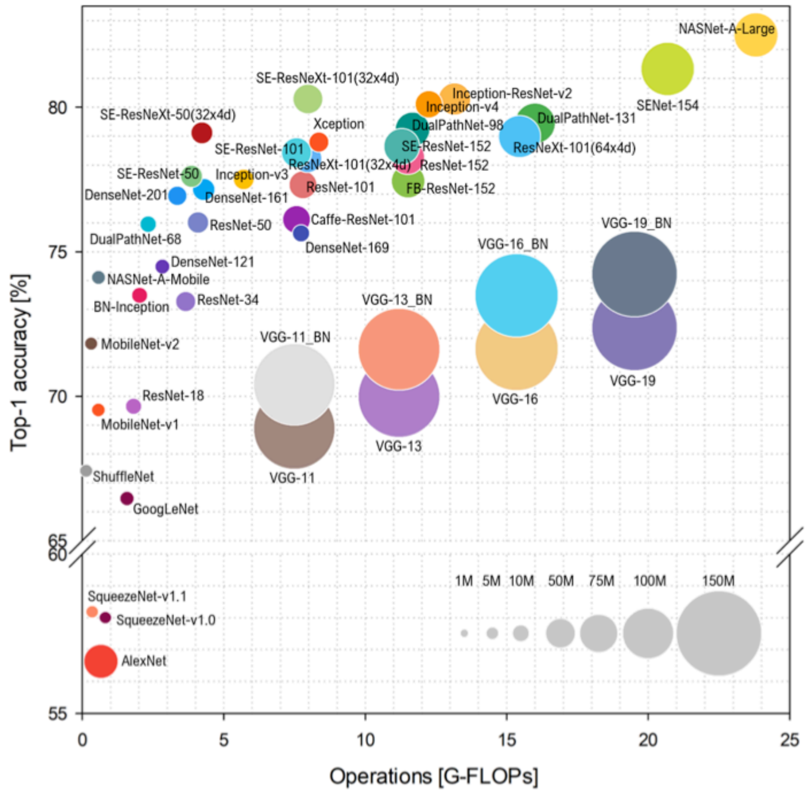
“Deep Reinforcement Learning that Matters”, Henderson et al, AAAI 2018

Universität Bremen

Reproducibility crisis: ~~continual~~ ML

Recall: even in "standard" ML with static models & i.i.d. datasets:

- Many aspects of variation/interest
- Fair comparisons, statistical significance, over-emphasis on singular metrics vs. exhaustive reporting



Bianco et al, "Benchmark Analysis of Representative Deep Neural Network Architectures", IEEE Access, 2018

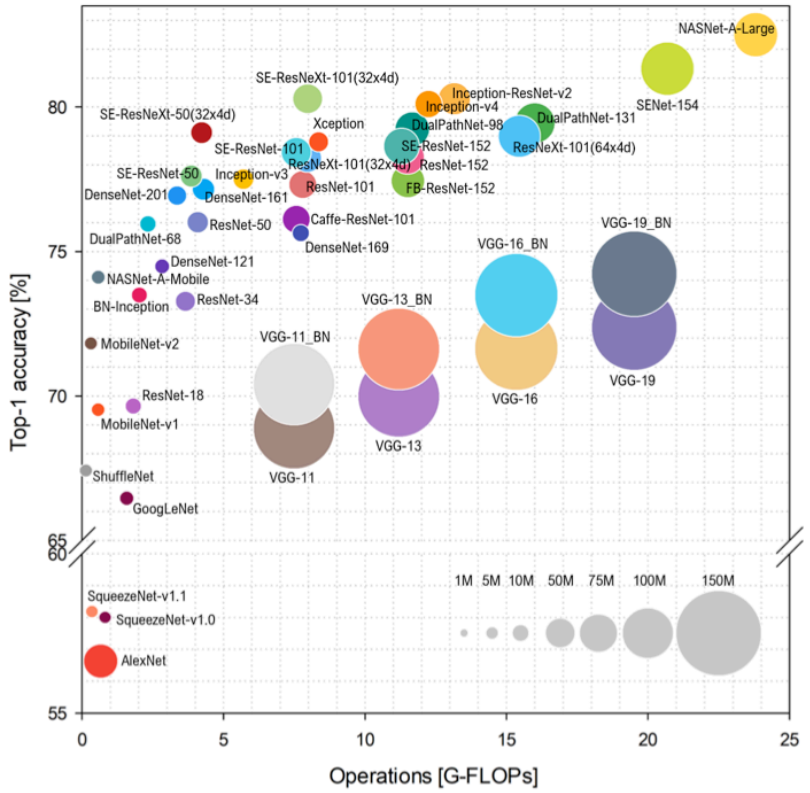
Universität Bremen

Reproducibility crisis: ~~continual~~ ML

Recall: even in "standard" ML with static models & i.i.d. datasets:

- Many aspects of variation/interest
- Fair comparisons, statistical significance, over-emphasis on singular metrics vs. exhaustive reporting

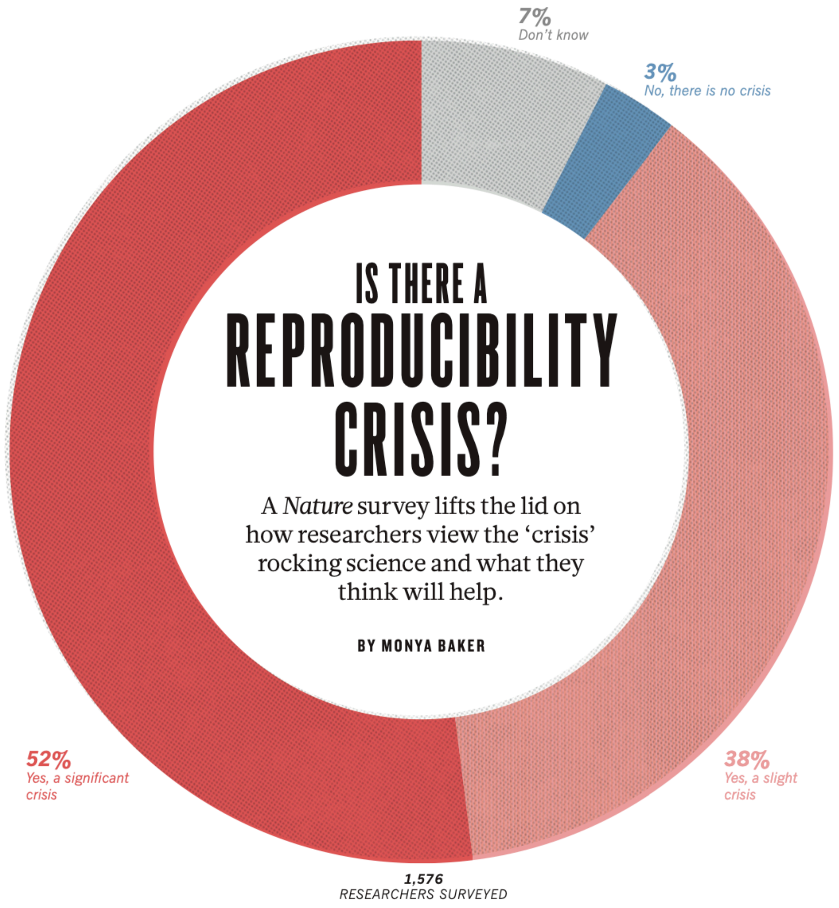
- (Misaligned?) research incentives: "publish or perish"
- Code, data assets, accessibility …



Bianco et al, "Benchmark Analysis of Representative Deep Neural Network Architectures", IEEE Access, 2018

Universität Bremen

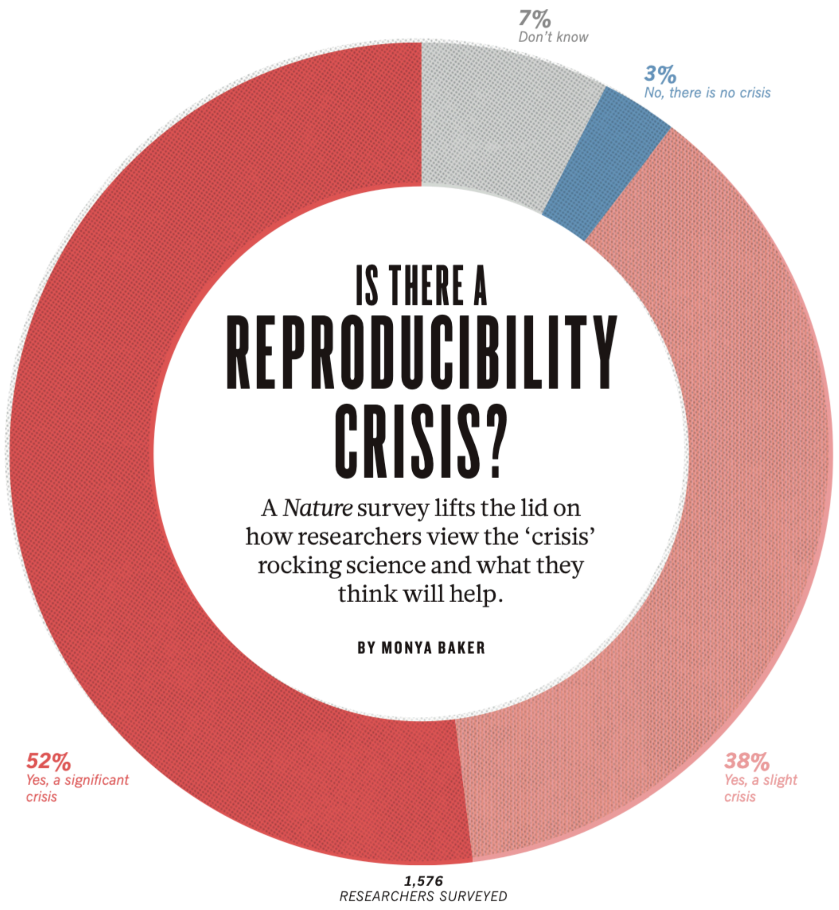
Reproducibility crisis: ~~continual ML~~ science

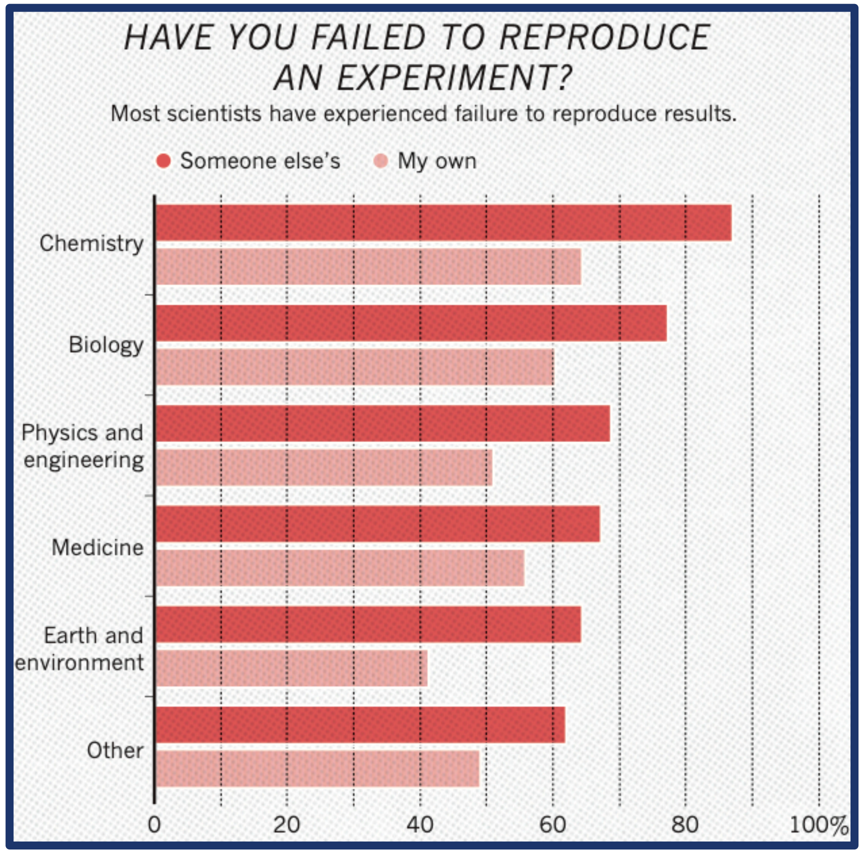


"1500 scientists lift the lid on reproducibility", Baker, Nature 533, 2016

Universität Bremen

Reproducibility crisis: ~~continual ML~~ science

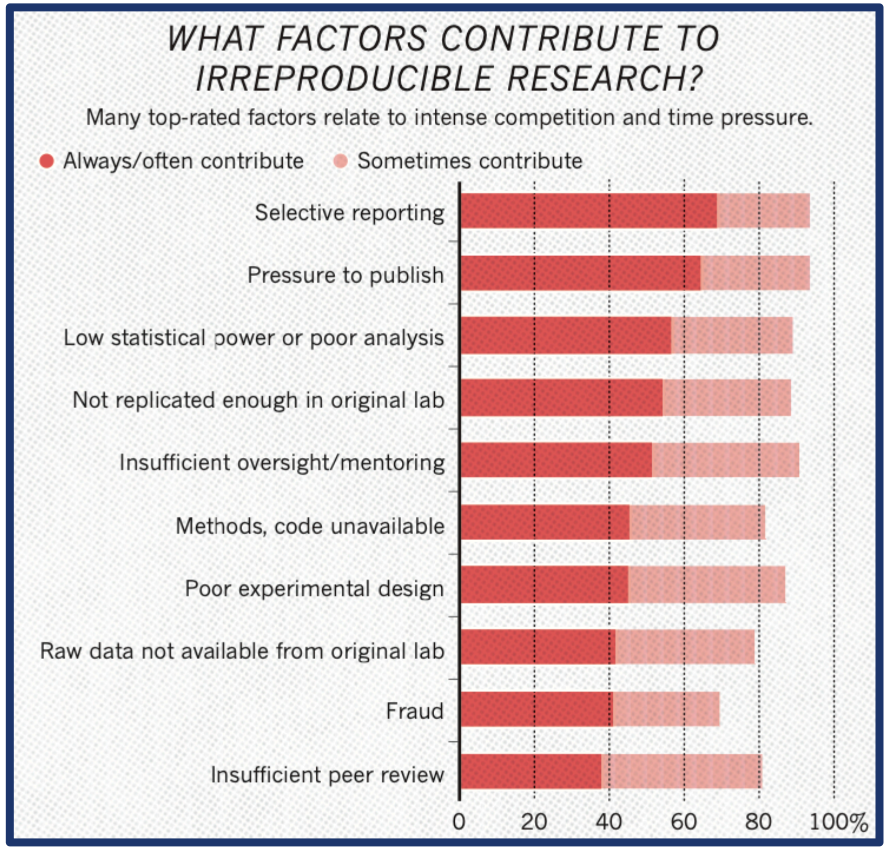




"1500 scientists lift the lid on reproducibility", Baker, Nature 533, 2016

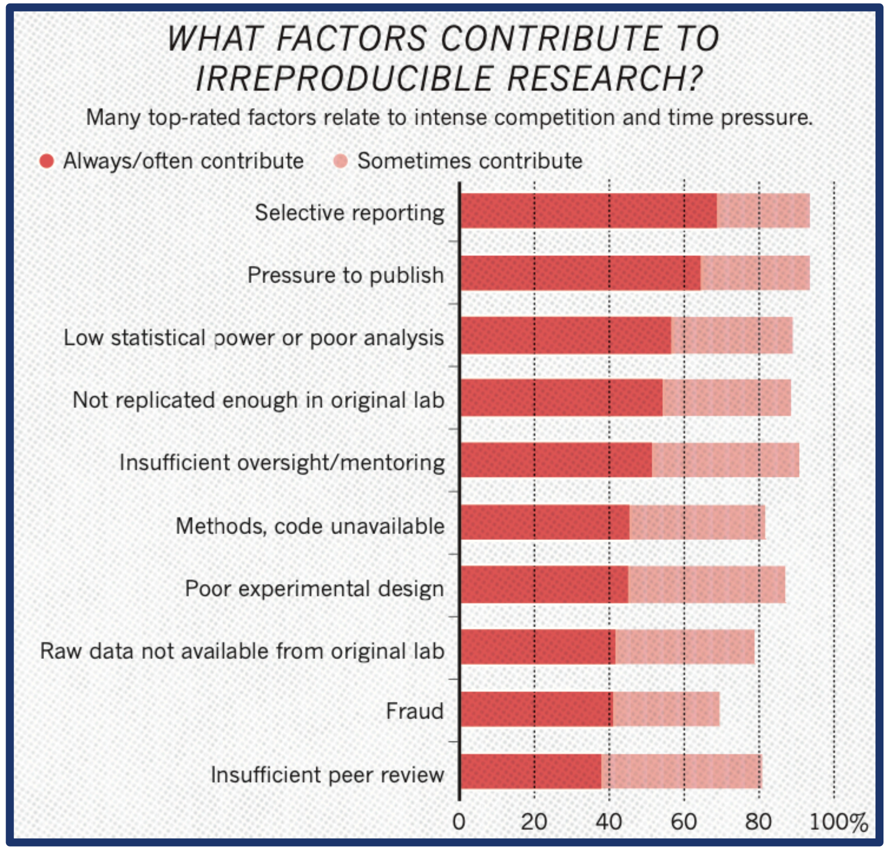
Universität Bremen

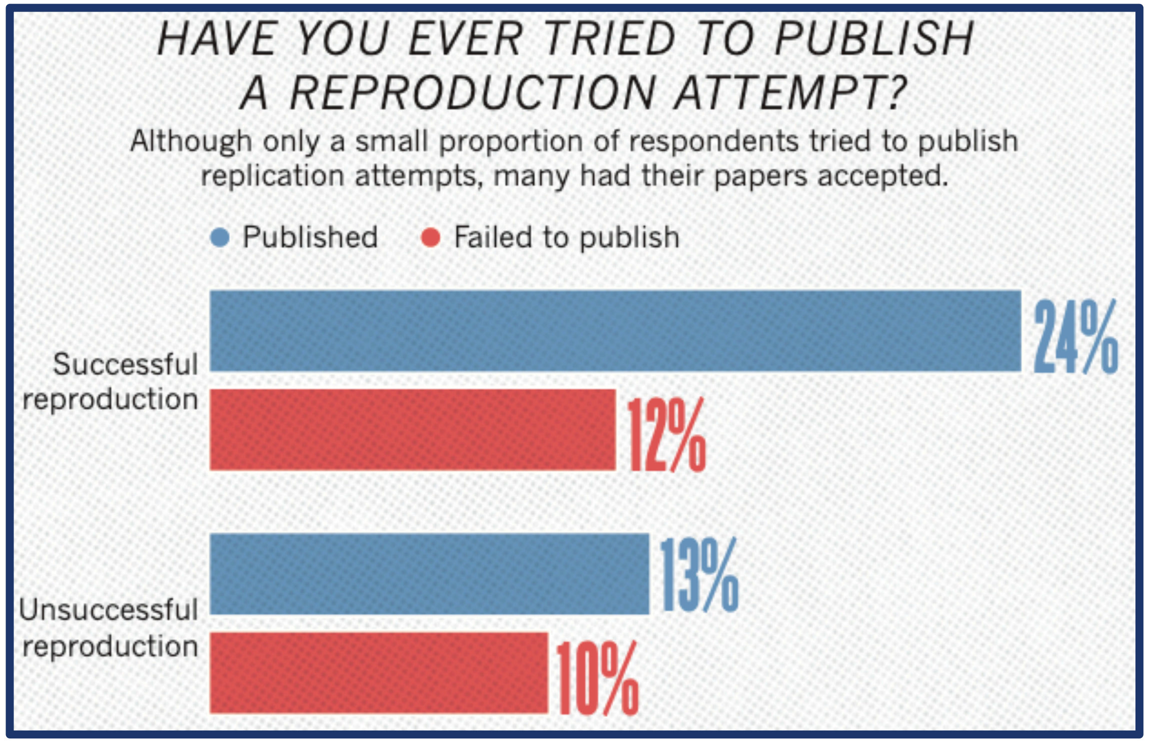
Reproducibility crisis: ~~continual ML~~ science



“1500 scientists lift the lid on reproducibility”, Baker, Nature 533, 2016

Reproducibility crisis: ~~continual ML~~ science





"1500 scientists lift the lid on reproducibility", Baker, Nature 533, 2016

Question time

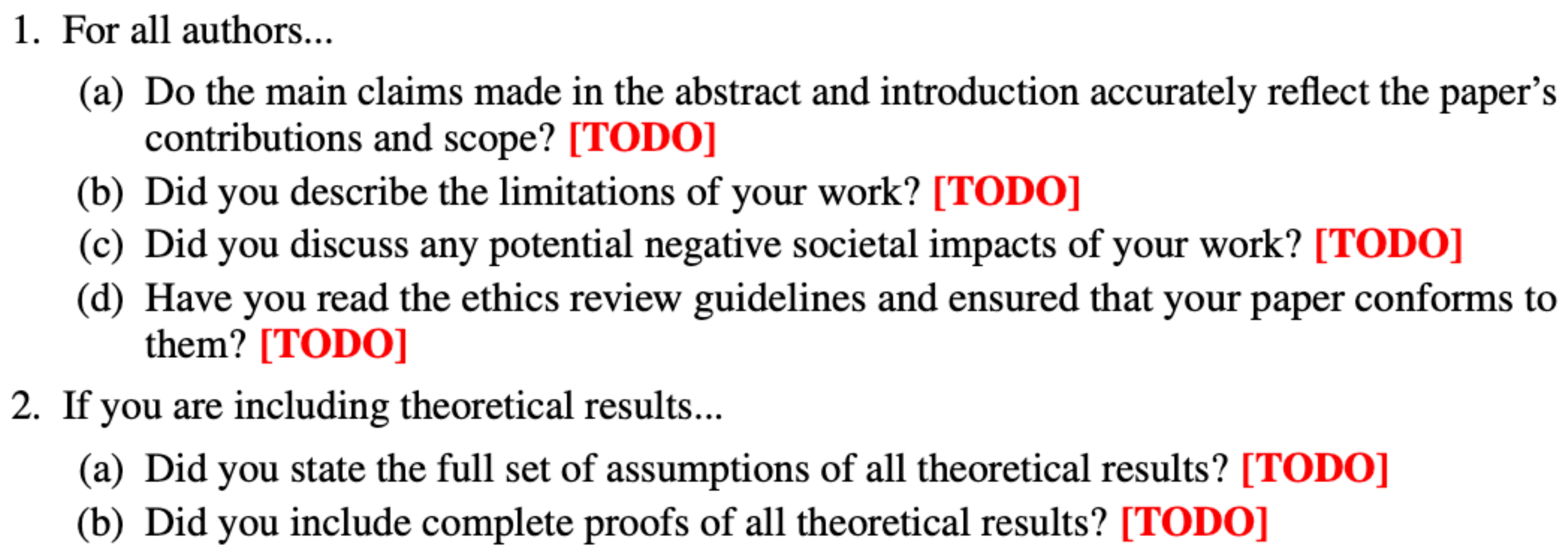
There are plenty of other interesting aspects to measure - perhaps you can think of some.
But importantly, how should we report & compare?

This is the one point in the lecture, where I/we don't have answers yet. But we can look at current practice: guidelines, checklists & more transparency

Universität Bremen

Assumptions matter: incentivize transparency

The era of ML checklists: from NeurIPS to multiple publication venues

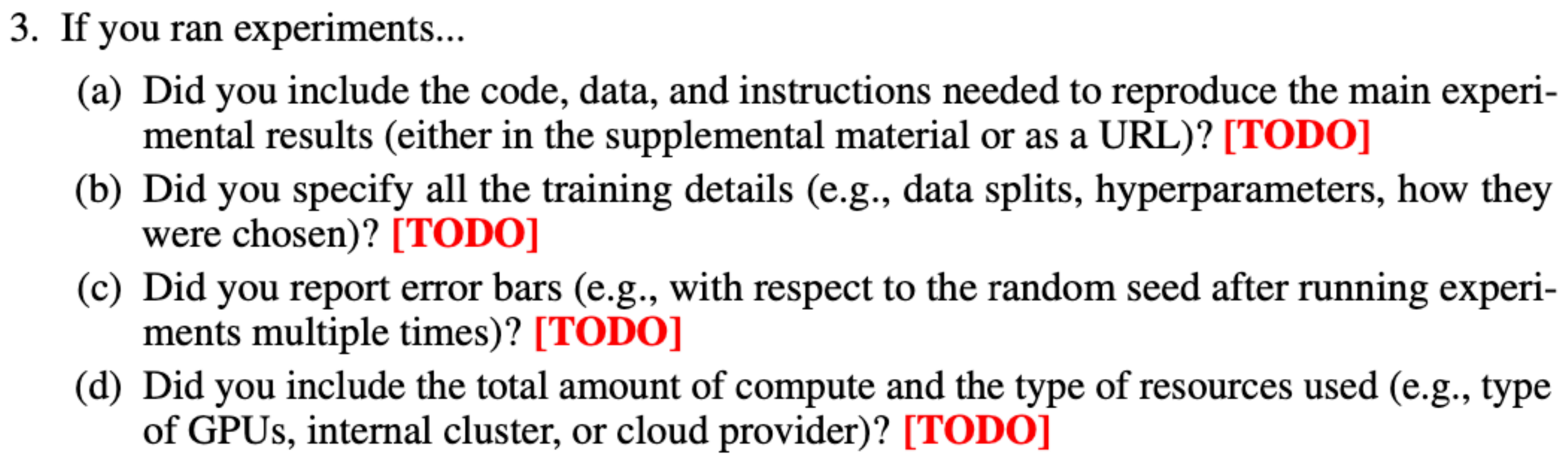


1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [TODO]
   (b) Did you describe the limitations of your work? [TODO]
   (c) Did you discuss any potential negative societal impacts of your work? [TODO]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [TODO]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [TODO]
   (b) Did you include complete proofs of all theoretical results? [TODO]

Checklist blog: https://neuripsconf.medium.com/introducing-the-neurips-2021-paper-checklist-3220d6df500b , checklist taken from formatting instructions

Assumptions matter: incentivize transparency

The era of ML checklists: from NeurIPS to multiple publication venues



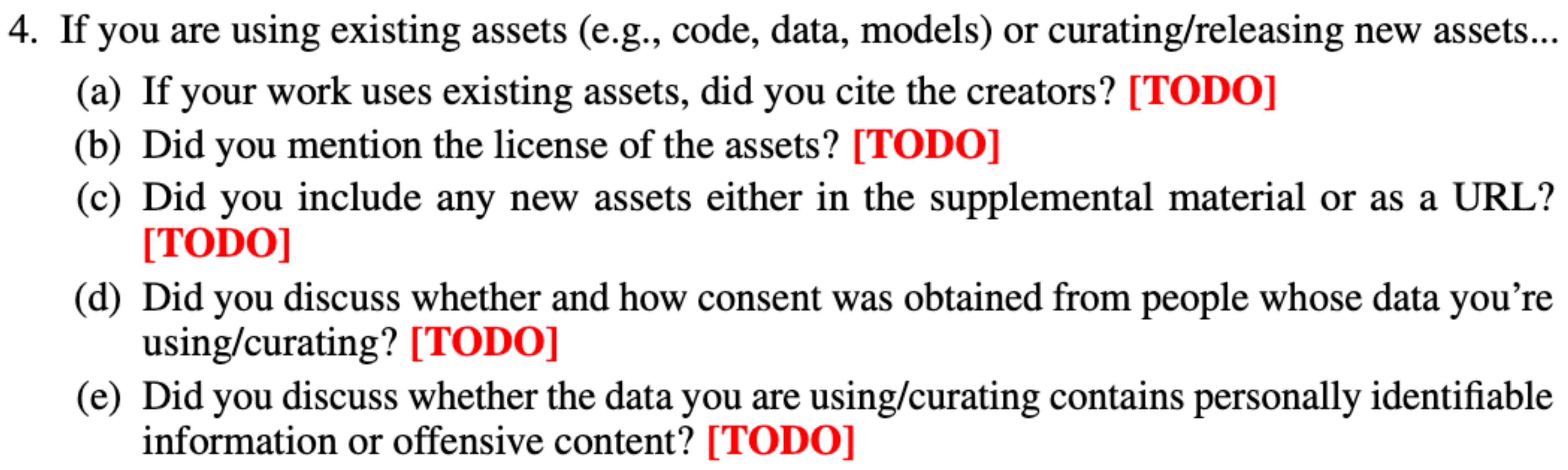
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [TODO]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [TODO]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [TODO]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [TODO]

Checklist blog: https://neuripsconf.medium.com/introducing-the-neurips-2021-paper-checklist-3220d6df500b , checklist taken from formatting instructions

Universität Bremen

Assumptions matter: incentivize transparency

The era of ML checklists: from NeurIPS to multiple publication venues



4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**

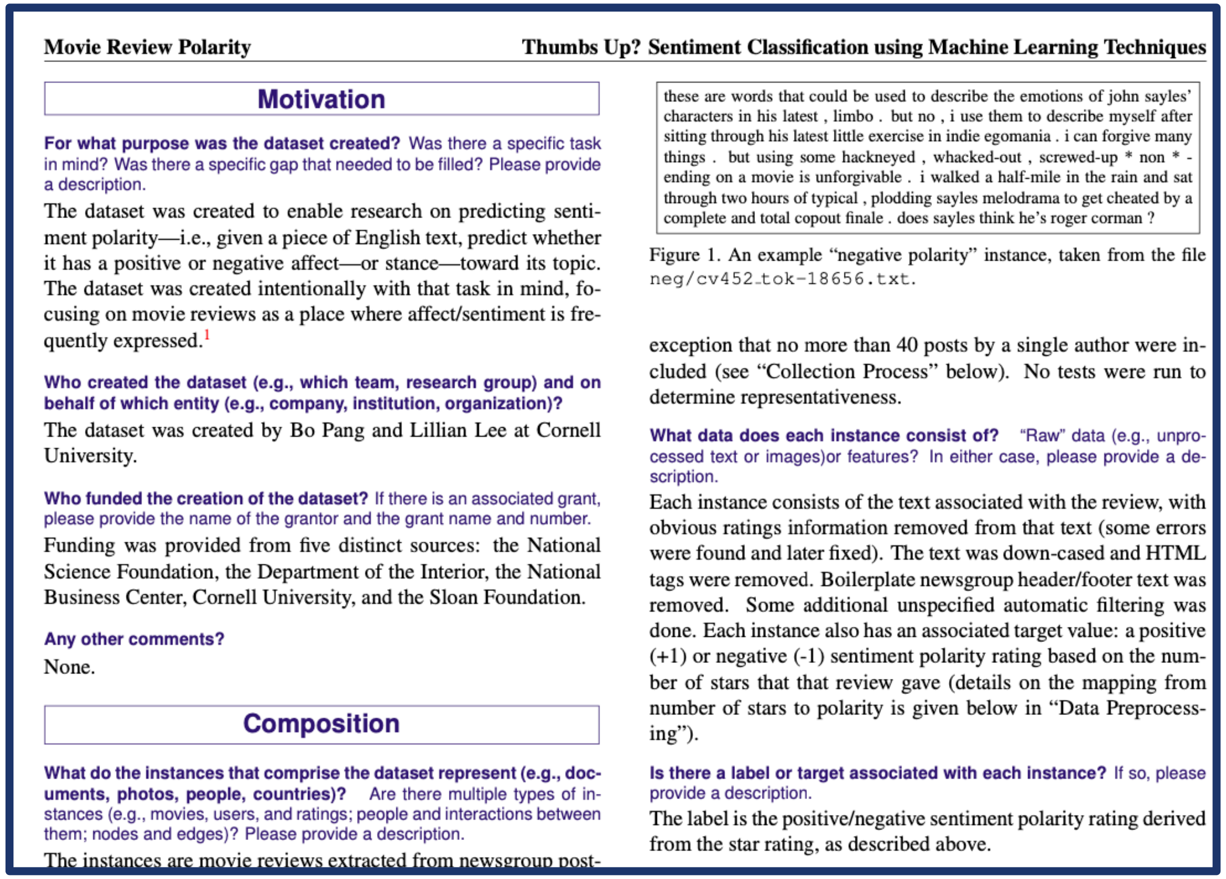
Checklist blog: https://neuripsconf.medium.com/introducing-the-neurips-2021-paper-checklist-3220d6df500b , checklist taken from formatting instructions

Universität Bremen

Intent matters: specify intended data use

The era of ML checklists:

- from algorithms to data: motivation, composition, collection, processing, maintenance, ethical considerations …



Movie Review Polarity — Thumbs Up? Sentiment Classification using Machine Learning Techniques

Motivation

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description.

The dataset was created to enable research on predicting sentiment polarity—i.e., given a piece of English text, predict whether it has a positive or negative affect—or stance—toward its topic. The dataset was created intentionally with that task in mind, focusing on movie reviews as a place where affect/sentiment is frequently expressed.[1]

Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?

The dataset was created by Bo Pang and Lillian Lee at Cornell University.

Who funded the creation of the dataset? If there is an associated grant, please provide the name of the grantor and the grant name and number.

Funding was provided from five distinct sources: the National Science Foundation, the Department of the Interior, the National Business Center, Cornell University, and the Sloan Foundation.

Any other comments?

None.

Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description.

The instances are movie reviews extracted from newsgroup post-

these are words that could be used to describe the emotions of john sayles' characters in his latest , limbo . but no , i use them to describe myself after sitting through his latest little exercise in indie egomania . i can forgive many things . but using some hackneyed , whacked-out , screwed-up * non * - ending on a movie is unforgivable . i walked a half-mile in the rain and sat through two hours of typical , plodding sayles melodrama to get cheated by a complete and total copout finale . does sayles think he's roger corman ?

Figure 1. An example "negative polarity" instance, taken from the file `neg/cv452_tok-18656.txt`.

exception that no more than 40 posts by a single author were included (see "Collection Process" below). No tests were run to determine representativeness.

What data does each instance consist of? "Raw" data (e.g., unprocessed text or images)or features? In either case, please provide a description.

Each instance consists of the text associated with the review, with obvious ratings information removed from that text (some errors were found and later fixed). The text was down-cased and HTML tags were removed. Boilerplate newsgroup header/footer text was removed. Some additional unspecified automatic filtering was done. Each instance also has an associated target value: a positive (+1) or negative (-1) sentiment polarity rating based on the number of stars that that review gave (details on the mapping from number of stars to polarity is given below in "Data Preprocessing").

Is there a label or target associated with each instance? If so, please provide a description.

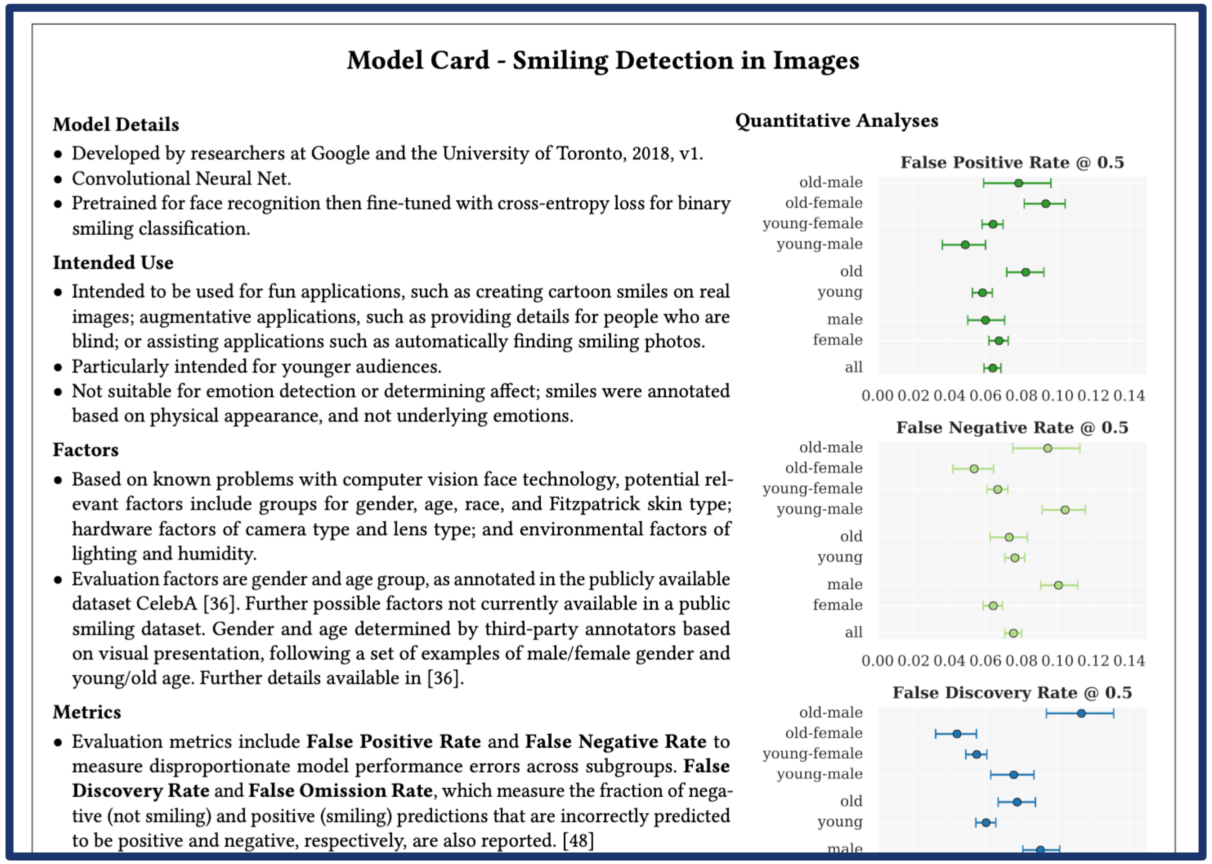
The label is the positive/negative sentiment polarity rating derived from the star rating, as described above.

"Datasheets for Datasets", Gebru et al, CACM 2021

Universität Bremen

Intent matters: specify intended model use

The era of ML checklists:
- from algorithms to data: motivation, composition, collection, processing, maintenance, ethical considerations …
- to models: application intent, development, factors, metrics



Model Card - Smiling Detection in Images

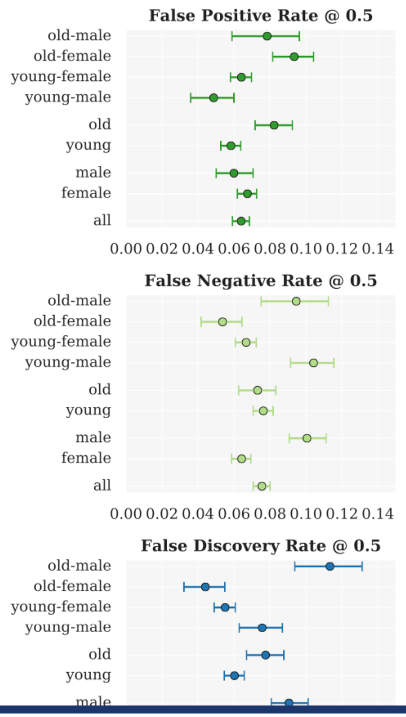
Model Details
- Developed by researchers at Google and the University of Toronto, 2018, v1.
- Convolutional Neural Net.
- Pretrained for face recognition then fine-tuned with cross-entropy loss for binary smiling classification.

Intended Use
- Intended to be used for fun applications, such as creating cartoon smiles on real images; augmentative applications, such as providing details for people who are blind; or assisting applications such as automatically finding smiling photos.
- Particularly intended for younger audiences.
- Not suitable for emotion detection or determining affect; smiles were annotated based on physical appearance, and not underlying emotions.

Factors
- Based on known problems with computer vision face technology, potential relevant factors include groups for gender, age, race, and Fitzpatrick skin type; hardware factors of camera type and lens type; and environmental factors of lighting and humidity.
- Evaluation factors are gender and age group, as annotated in the publicly available dataset CelebA [36]. Further possible factors not currently available in a public smiling dataset. Gender and age determined by third-party annotators based on visual presentation, following a set of examples of male/female gender and young/old age. Further details available in [36].

Metrics
- Evaluation metrics include **False Positive Rate** and **False Negative Rate** to measure disproportionate model performance errors across subgroups. **False Discovery Rate** and **False Omission Rate**, which measure the fraction of negative (not smiling) and positive (smiling) predictions that are incorrectly predicted to be positive and negative, respectively, are also reported. [48]

Quantitative Analyses

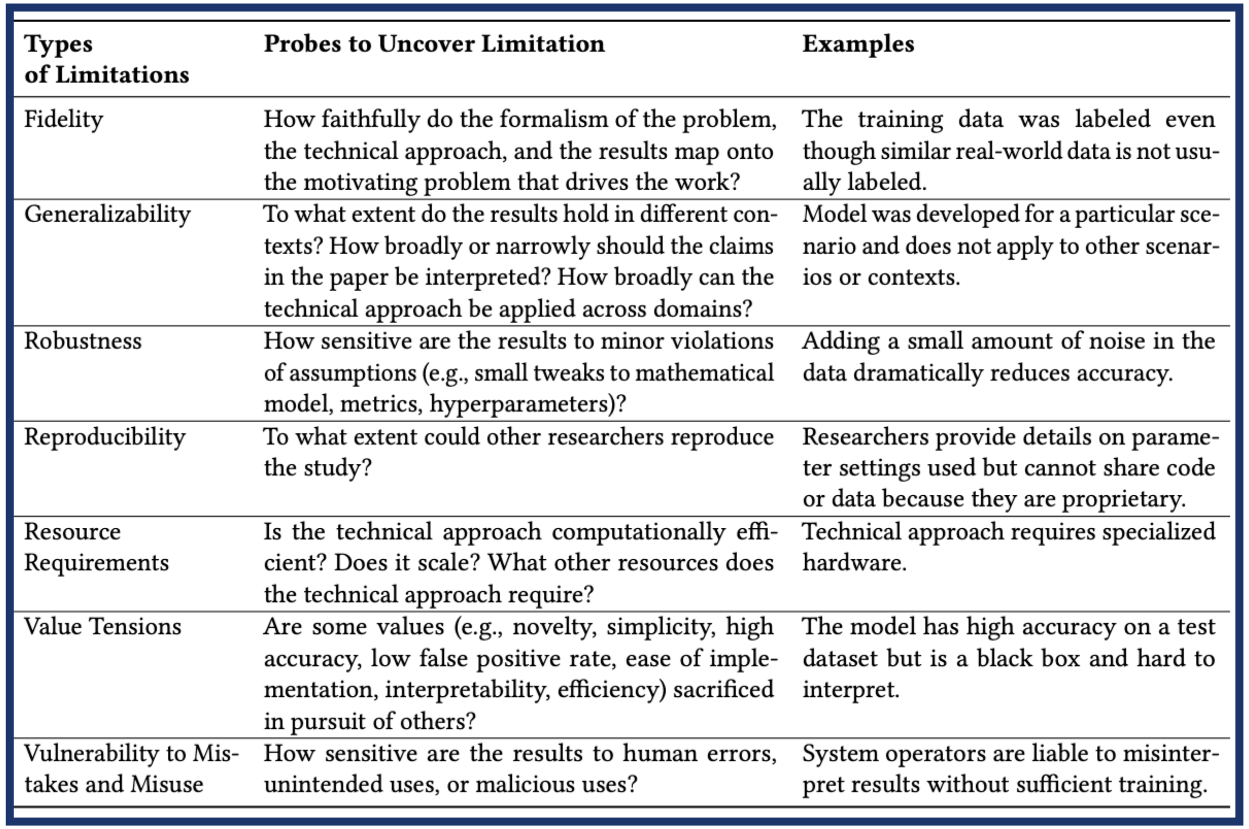


“Model Cards for Model Reporting”, Mitchell et al, FAccT 2019

Intent matters: specify intended system use

The era of ML checklists:

- from algorithms to data: motivation, composition, collection, processing, maintenance, ethical considerations …
- to models: application intent, development, factors, metrics
- to known limitations



Types of Limitations	Probes to Uncover Limitation	Examples
Fidelity	How faithfully do the formalism of the problem, the technical approach, and the results map onto the motivating problem that drives the work?	The training data was labeled even though similar real-world data is not usually labeled.
Generalizability	To what extent do the results hold in different contexts? How broadly or narrowly should the claims in the paper be interpreted? How broadly can the technical approach be applied across domains?	Model was developed for a particular scenario and does not apply to other scenarios or contexts.
Robustness	How sensitive are the results to minor violations of assumptions (e.g., small tweaks to mathematical model, metrics, hyperparameters)?	Adding a small amount of noise in the data dramatically reduces accuracy.
Reproducibility	To what extent could other researchers reproduce the study?	Researchers provide details on parameter settings used but cannot share code or data because they are proprietary.
Resource Requirements	Is the technical approach computationally efficient? Does it scale? What other resources does the technical approach require?	Technical approach requires specialized hardware.
Value Tensions	Are some values (e.g., novelty, simplicity, high accuracy, low false positive rate, ease of implementation, interpretability, efficiency) sacrificed in pursuit of others?	The model has high accuracy on a test dataset but is a black box and hard to interpret.
Vulnerability to Mistakes and Misuse	How sensitive are the results to human errors, unintended uses, or malicious uses?	System operators are liable to misinterpret results without sufficient training.

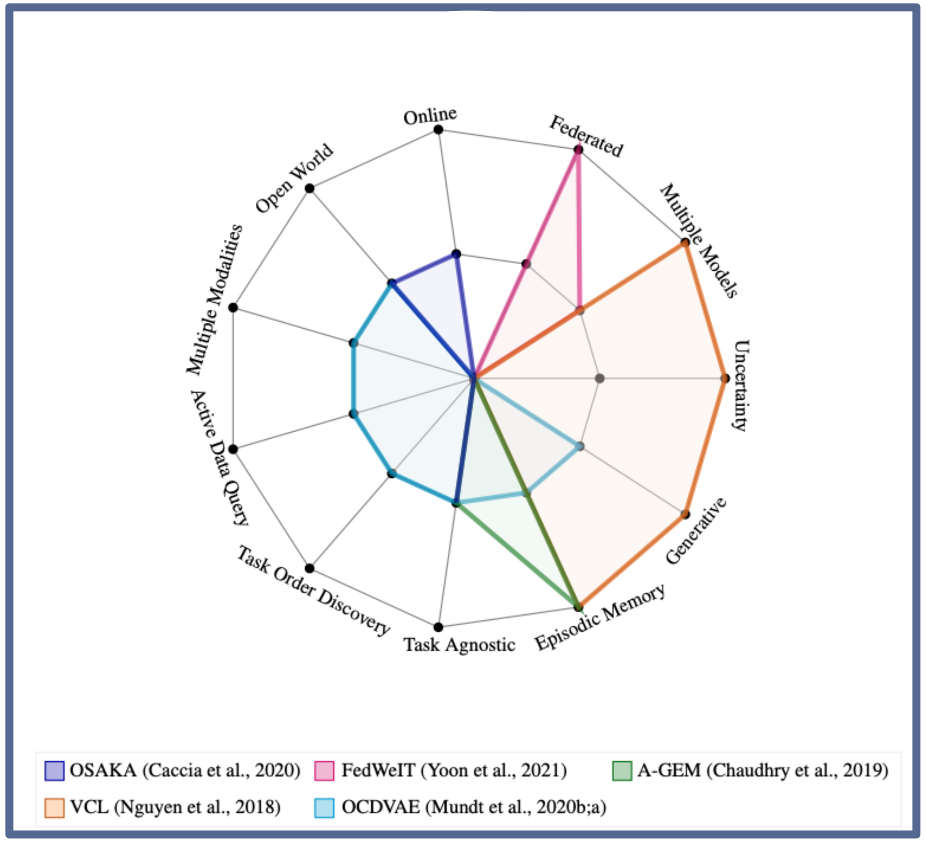
Smith et al, “REAL ML: Recognizing, Exploring, and Articulating Limitations of Machine Learning Research”, FAccT 2022

Universität Bremen

Assumptions & intent matters in continual ML

CLEVA-Compass for continual ML

Inner compass (star plot):
indicates paradigm inspiration & continual setting assumptions



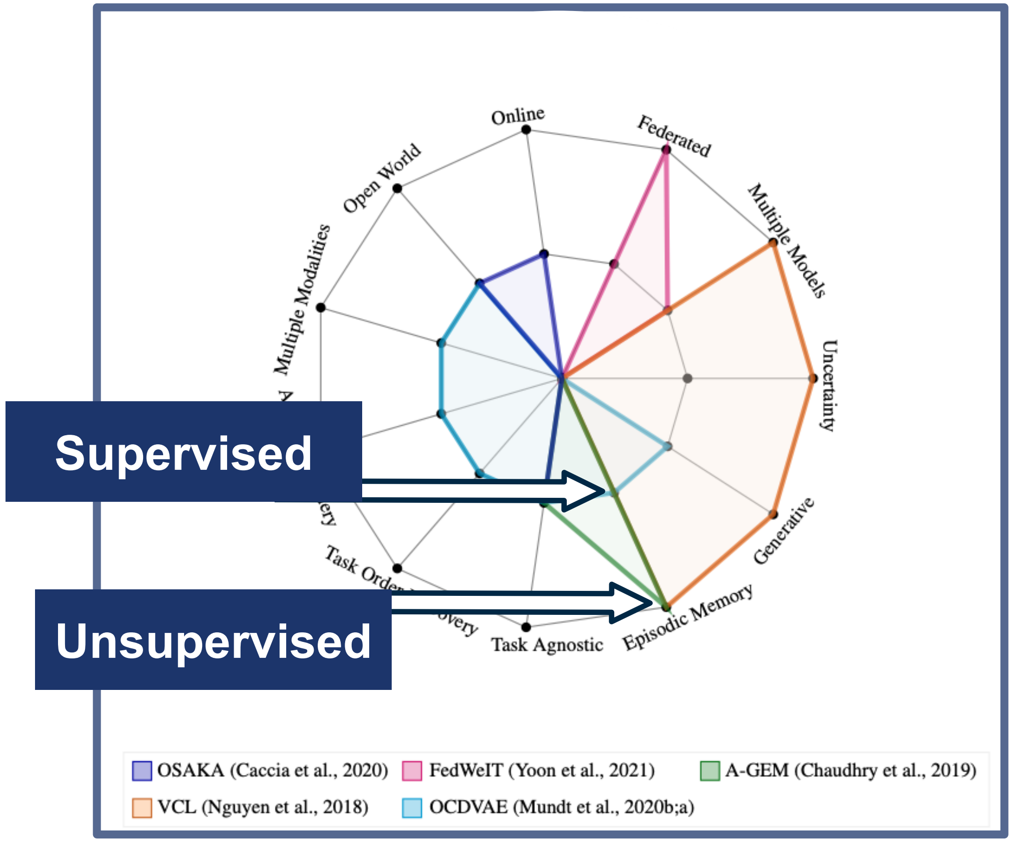
Mundt et al, "CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability", ICLR 2022

Assumptions & intent matters in continual ML

CLEVA-Compass for continual ML

Inner compass (star plot): indicates paradigm inspiration & continual setting assumptions

Inner levels: indicates "supervision"



Mundt et al, "CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability", ICLR 2022

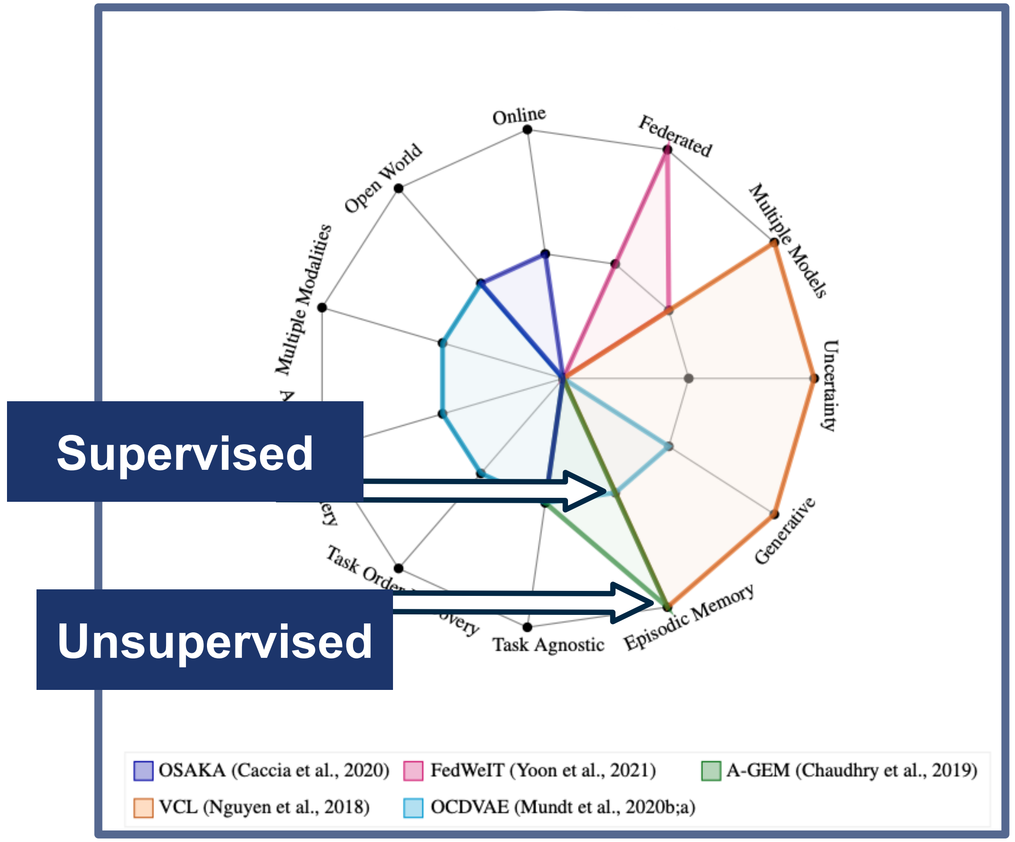
Assumptions & intent matters in continual ML

CLEVA-Compass for continual ML

Inner compass (star plot): indicates paradigm inspiration & continual setting assumptions

Inner levels: indicates "supervision"

What does it mean for (data) memory to be "un-/supervised"?



Mundt et al, "CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability", ICLR 2022

Universität Bremen

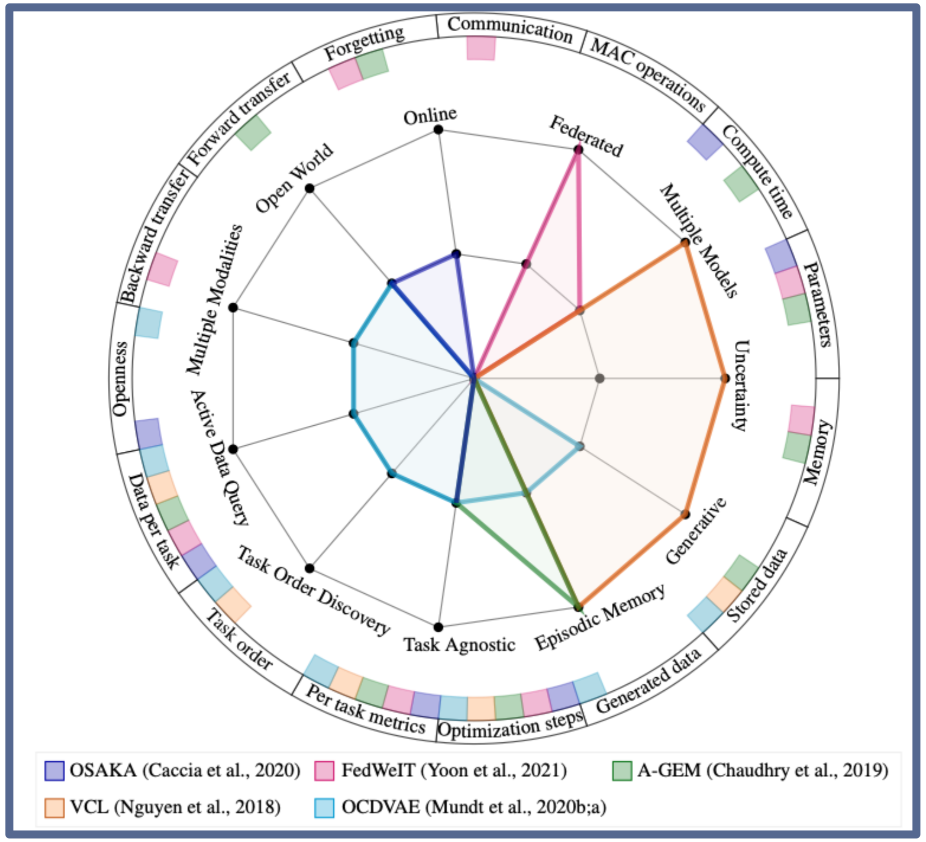
Assumptions & intent matters in continual ML

CLEVA-Compass for continual ML

Inner compass (star plot): indicates paradigm inspiration & continual setting assumptions

Inner levels: indicates "supervision"

Outer compass level (ring): indicates practically reported set of important evaluation metrics



Mundt et al, "CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability", ICLR 2022

Question time

The five algorithms in the “CLEVA-Compass” illustration all try to solve “continual CIFAR-100” image classification, but they appear differently.
Is this bad necessarily?

Universität Bremen

Assumptions can be necessary for applications, but approach comparison should be meaningful

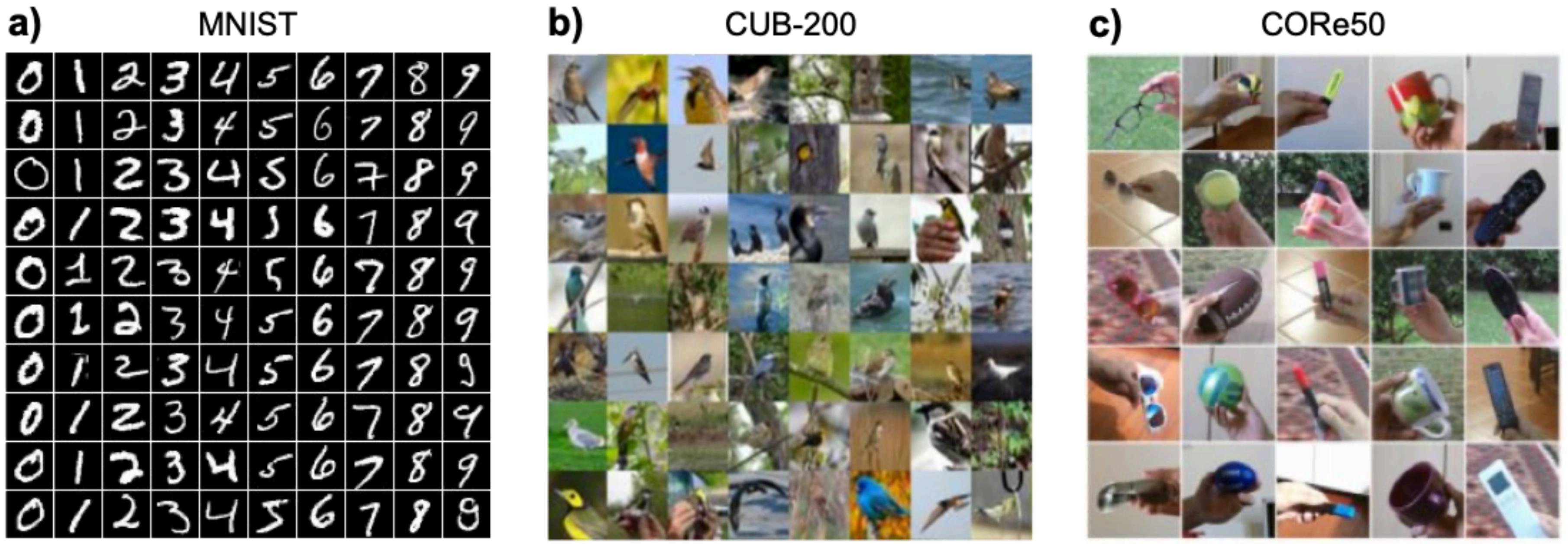
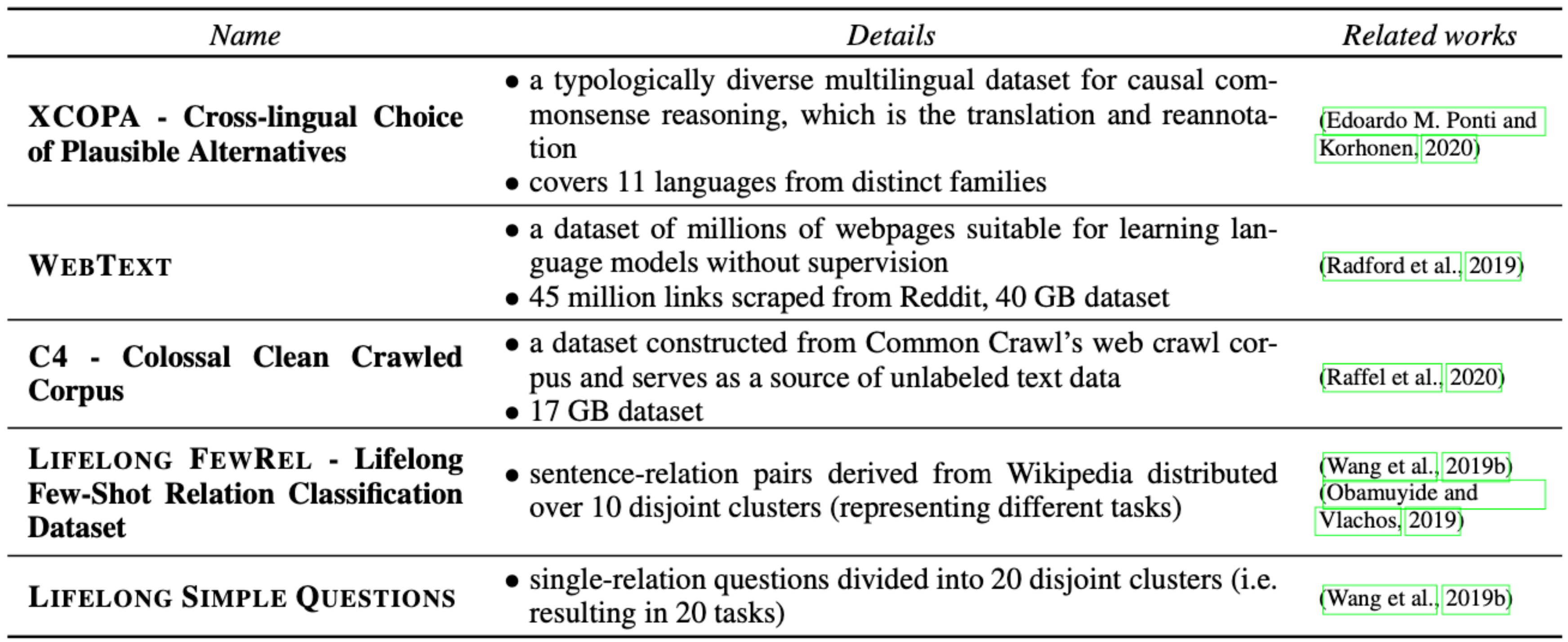


Figure 3: Example images from benchmark datasets used for the evaluation of lifelong learning

Parisi et al, “Continual Lifelong Learning with Neural Networks: A Review”, Neural Networks 2019

Universität
Bremen

Assumptions can be necessary for applications, but approach comparison should be meaningful



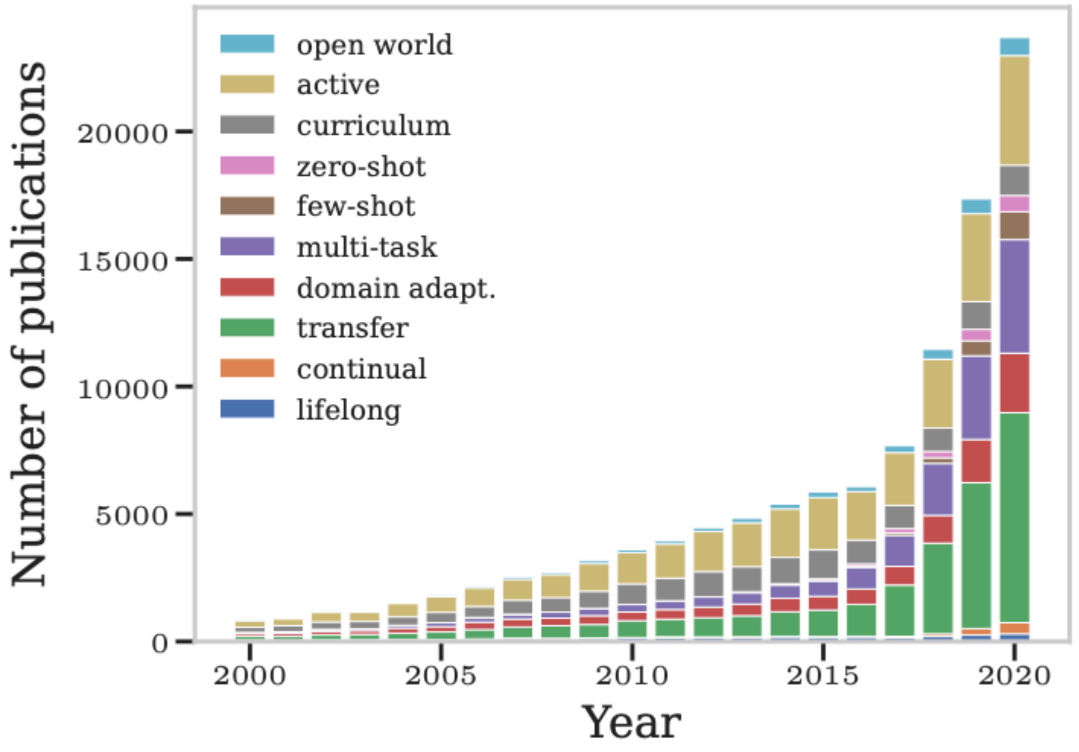
Name	*Details*	*Related works*
XCOPA - Cross-lingual Choice of Plausible Alternatives	• a typologically diverse multilingual dataset for causal commonsense reasoning, which is the translation and reannotation • covers 11 languages from distinct families	(Edoardo M. Ponti and Korhonen, 2020)
WebText	• a dataset of millions of webpages suitable for learning language models without supervision • 45 million links scraped from Reddit, 40 GB dataset	(Radford et al., 2019)
C4 - Colossal Clean Crawled Corpus	• a dataset constructed from Common Crawl's web crawl corpus and serves as a source of unlabeled text data • 17 GB dataset	(Raffel et al., 2020)
Lifelong FewRel - Lifelong Few-Shot Relation Classification Dataset	• sentence-relation pairs derived from Wikipedia distributed over 10 disjoint clusters (representing different tasks)	(Wang et al., 2019b) (Obamuyide and Vlachos, 2019)
Lifelong Simple Questions	• single-relation questions divided into 20 disjoint clusters (i.e. resulting in 20 tasks)	(Wang et al., 2019b)

Biesialska et al, "Continual Learning in Natural Language Processing: A Survey", COLING 2020

Universität Bremen

Setting nuances can be a gift & a curse

Let's briefly remind ourselves of some "paradigm assumptions"



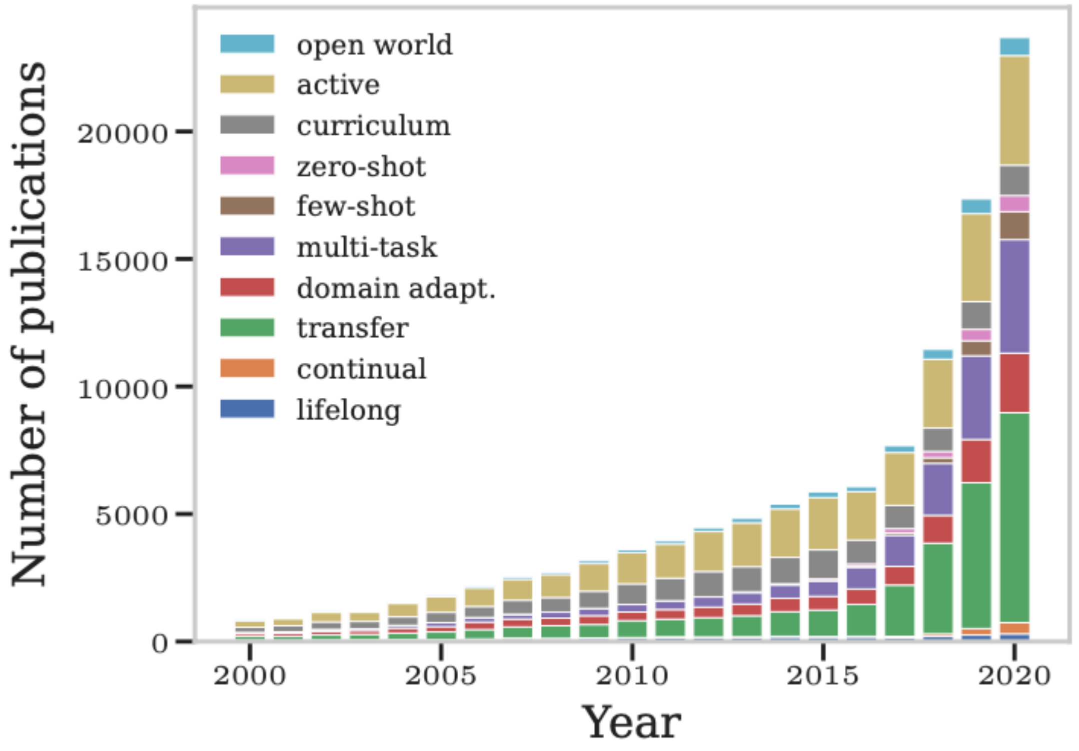
Mundt et al, "CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability", ICLR 2022

Setting nuances can be a gift & a curse

Let's briefly remind ourselves of some "paradigm assumptions"

We will circle back to summarize connections & implications at the end of the course.

First, let's look at a last frontier: "dealing with future data"



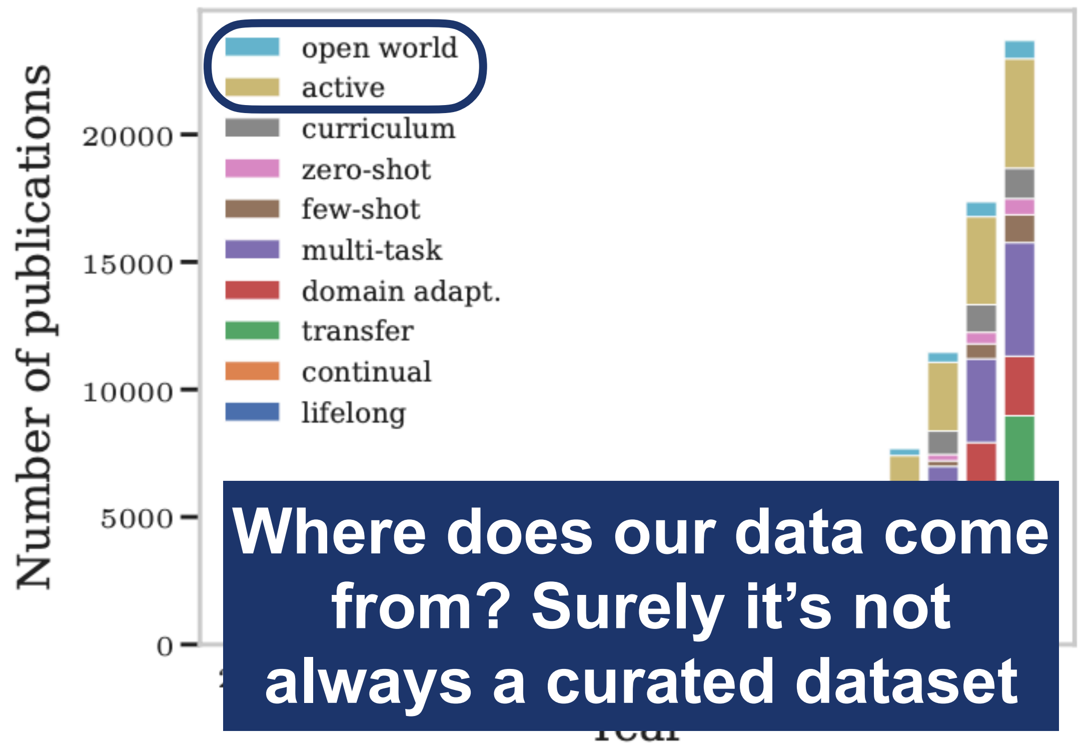
Mundt et al, "CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability", ICLR 2022

Setting nuances can be a gift & a curse

Let’s briefly remind ourselves of some “paradigm assumptions”

We will circle back to summarize connections & implications at the end of the course.

First, let’s look at a last frontier: “dealing with future data”

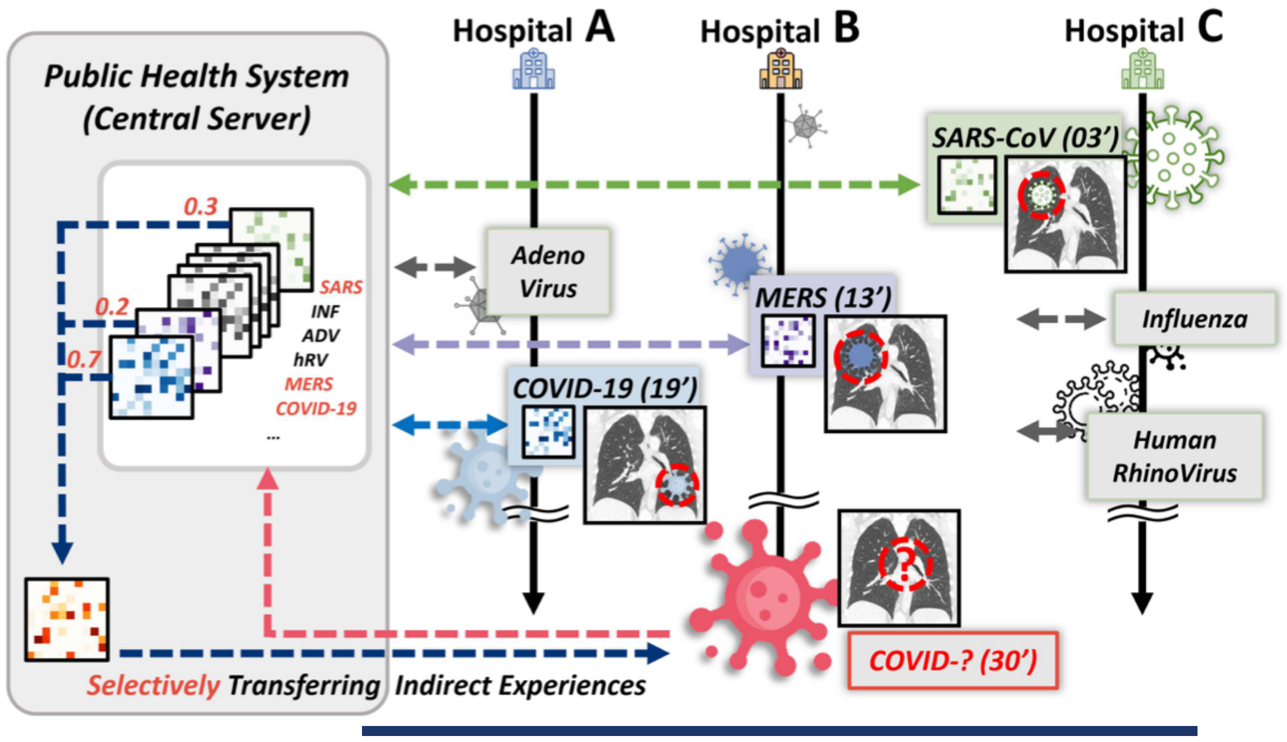


Where does our data come from? Surely it’s not always a curated dataset

Mundt et al, “CLEVA-Compass: A Continual Learning Evaluation Assessment Compass to Promote Research Transparency and Comparability”, ICLR 2022

Universität Bremen

Our earlier example as motivation

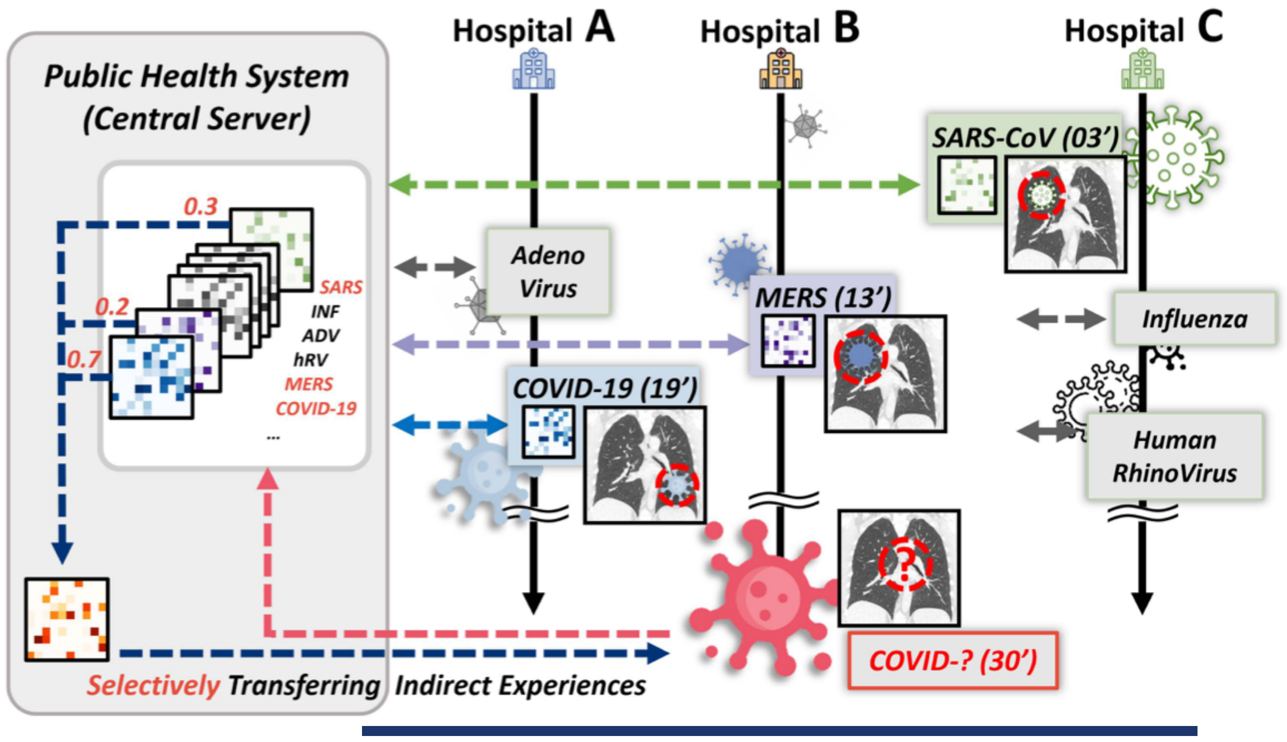


Yoon et al, "Federated Continual Learning with Weighted Inter-client Transfer", ICML 2021

Our earlier example as motivation

Questions to still answer:

- Can we determine if new data is informative?
- Can we spot if data is related to our task(s)?
- Can we actively seek/ acquire new data?
- Can we predict robustly on unknown data items?
- ….



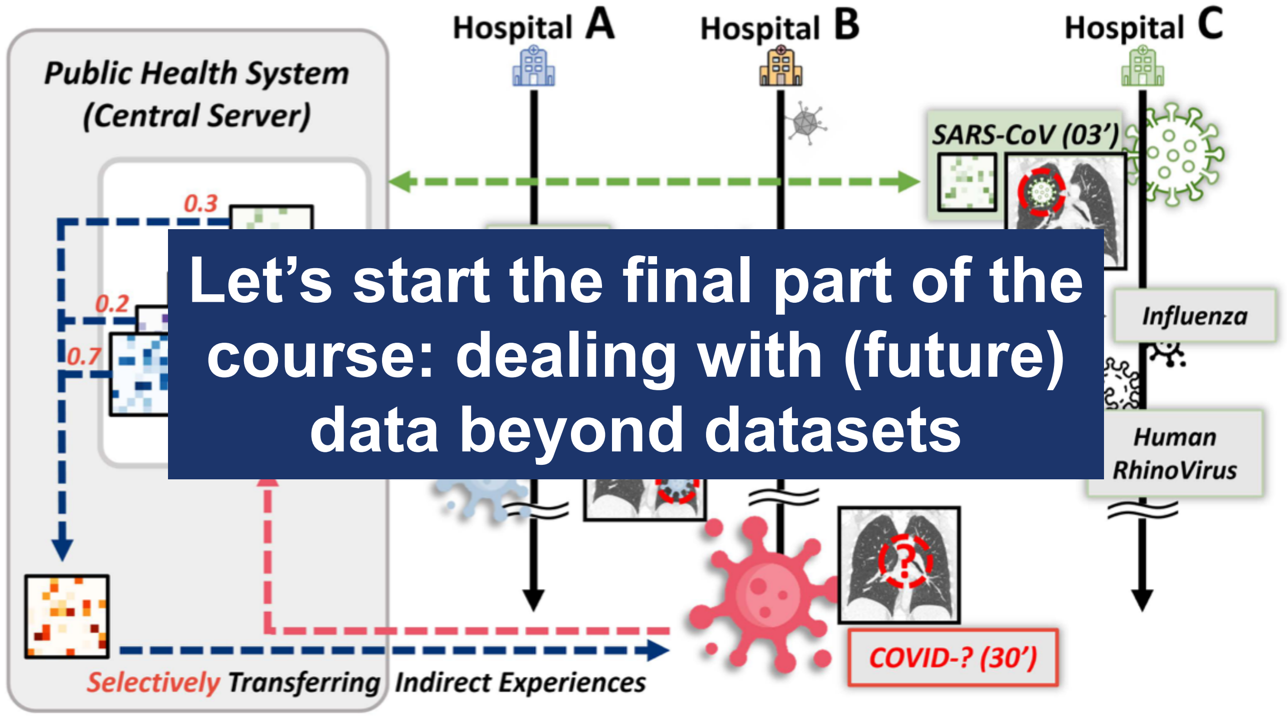
Yoon et al, "Federated Continual Learning with Weighted Inter-client Transfer", ICML 2021

Universität Bremen

Our earlier example as motivation

Questions to still answer:

- Can we determine if new data is informative?
- Can we spot if data is related to our task(s)?
- Can we actively seek/ acquire new data?
- Can we predict robustly on unknown data items?
- ….



Let's start the final part of the course: dealing with (future) data beyond datasets

Yoon et al, "Federated Continual Learning with Weighted Inter-client Transfer", ICML 2021