Embracing Change:
Continual Learning in Deep Neural Networks

Razvan Pascanu
razp@google.com
Research Scientist @ DeepMind

1 The world is non-stationary

Image from economist.com

The world is non-stationary.

When we build and deploy machine learning algorithms, we assume that our problem domain is fixed and that all variability is captured in a static dataset or learning environment.

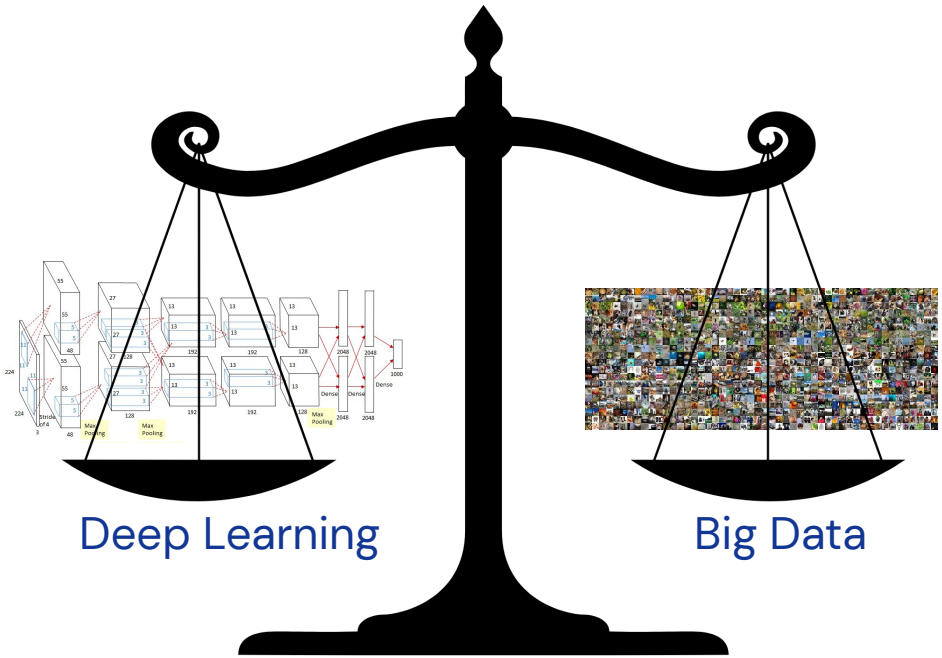
This is rarely true. For example:

- Health
- Robotics
- Language

Deep Learning is optimised for static, large-scale datasets

When we build and deploy machine learning algorithms, we assume that our problem domain is fixed and that all variability is captured in a static dataset or learning environment.

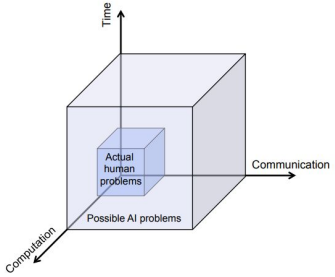
- Deep Learning is powerful because the models readily scale to fit large datasets.
- Supported by large-scale compute, and end-to-end optimization.
- Optimization is gradient-based, which assumes that the dataset is balanced, shuffled and randomly sampled during training (i.i.d. - more on this later).



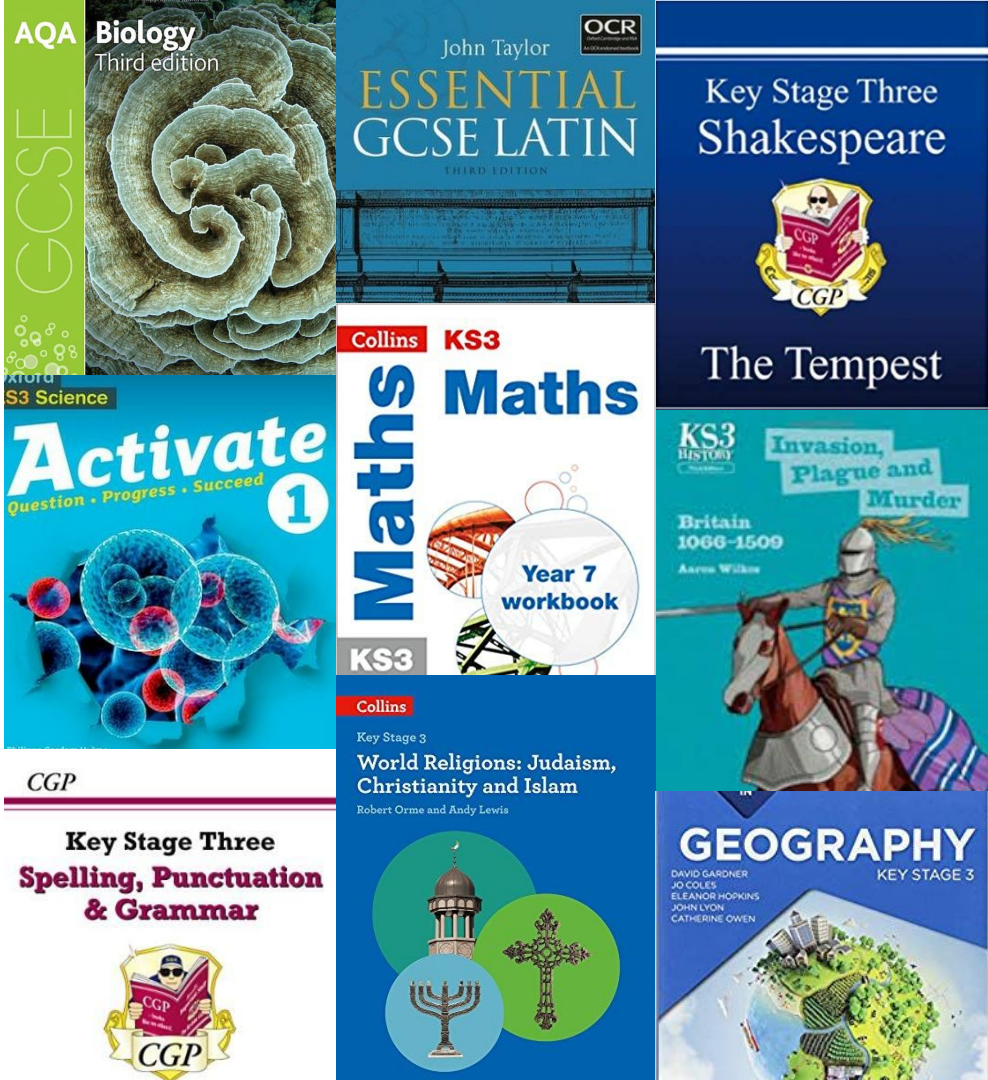
Humans don't learn well from randomly sampled data

Imagine learning all your high school subjects by sampling one page at random from every textbook.

Thomas Griffiths, *Understanding Human Intelligence through Human Limitations*



The profile of human intelligence is due to its limitations...



Defining Continual Learning / LifeLong Learning

Please see: *Hadsell et al. Embracing Change: Continual Learning in Deep Neural Networks*
https://www.cell.com/trends/cognitive-sciences/fulltext/S1364-66132030219-9

The learning environment is non-stationary, divided into a set of tasks that need to be completed sequentially.

There are many variations:

- task transitions (smooth or discrete);
- task length and repetition;
- task type (such as unsupervised, supervised, or reinforcement learning);
- or it may not have well-defined tasks

Defining Continual Learning (Solutions)

Characterizing continual learning **solutions** is more challenging: there are many **desiderata** and they are often **contradictory or competing**.

1. **Minimal access to previous tasks.** The model does not have infinite storage for previous experience and, crucially, it can not interact with previously seen tasks.
2. **Minimal increase in model capacity and computation.** The approach must be scalable: it cannot add a new model for each subsequent task.
3. **Minimizing catastrophic forgetting and interference**. Training on new tasks should not significantly reduce performance on previously learned tasks.
4. **Maintaining plasticity.** The model should be able to keep learning effectively as new tasks are observed.
5. **Maximizing forward and backward transfer.** Learning a task should improve related tasks,
   both past and future, in terms of both learning efficiency and performance

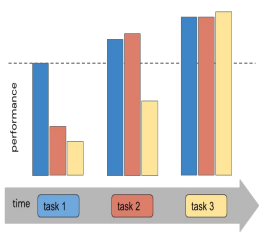


Illustration of forwards and backwards transfer.

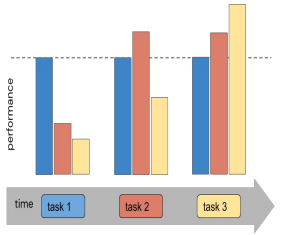


Illustration of forward transfer.

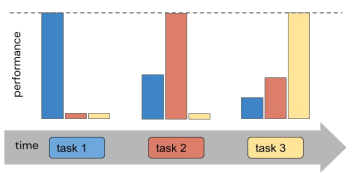


Illustration of catastrophic forgetting

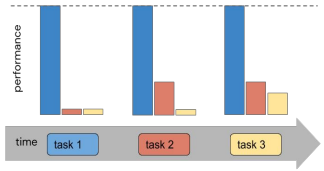
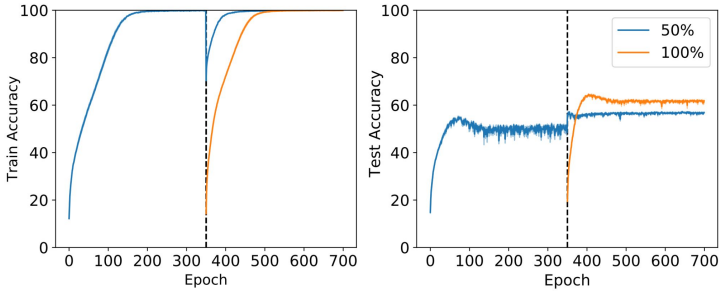
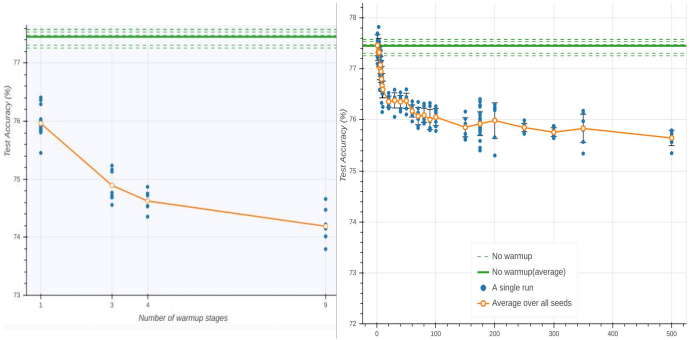


Illustration of declining plasticity

Forward/backward transfer and plasticity



Ash et al. 2020, *On warm-starting neural network training*



Berariu et al 2021, *A study on the plasticity of neural networks*

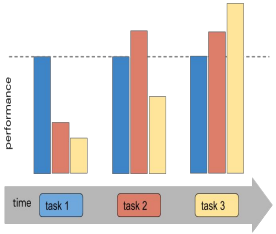


Illustration of forward transfer.

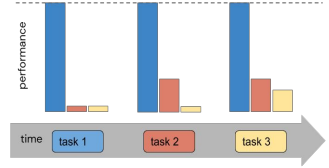


Illustration of declining plasticity

- Slower/Faster convergence (typically referred as forward transfer)
- Inability to optimize (intransigence -- Chaudhry et al. 2018)
- Inability to generalize (plasticity)

Contradictions, dilemmas, and trade-offs

Some of the aforementioned desiderata become competing objectives when optimized in a single model:

- Maintaining **perfect recall** (by forgetting nothing) in a **fixed-capacity model** is impossible given an arbitrarily long sequence of tasks.
    - This dilemma motivates **fast recovery**, which allows forgetting if previous performance levels can be recovered with a minimal amount of new experience.
- Forward (and backward) transfer contrasts with the ability to perfectly recall previous tasks.
- Any solution needs to balance competing needs. But what constitutes an optimal trade-off? How much should the model remember and how much is the model allowed to grow?

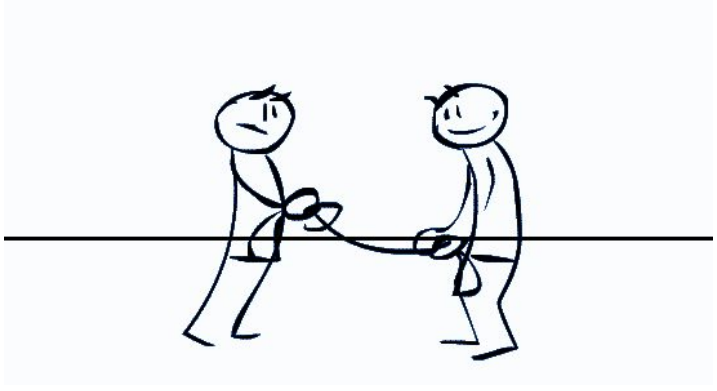
What is the goal of a Continual Learning system ?

Reasons for solving continual learning

- Real world is open-ended, we need systems that are able to continuously learn
- Humans learn in a non-stationary setting, CL takes us a step further to understand human learning
- For legal reasons or space constraints, data needs to be thrown out, but the system might still need to be aware of it
- Safety protocols should not be forgotten by a end-to-end learning system
- CL provides a structured space to understand non-stationary learning which we need to be able to deal with in domains like RL
- CL can be a tool to enhance efficiency of learning
- **A system that solves the CL problem should be able to accumulate knowledge and reuse this knowledge on future tasks in order to learn more efficiently (i.e. solving forgetting should lead to more efficient learning)**
- ...

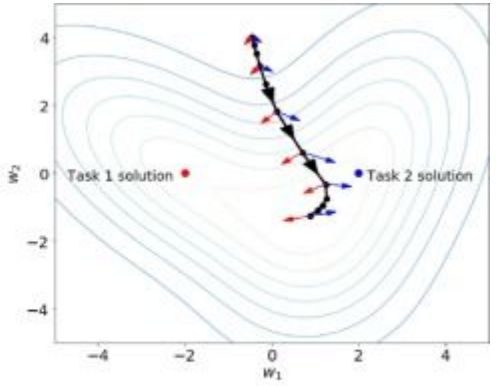
Tug-of-war learning dynamics

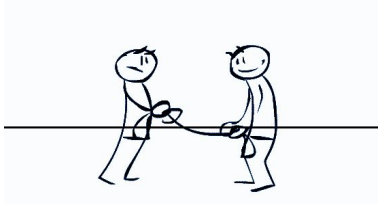
and the I.I.D. assumption



Gradient-based optimization and tug-of-war dynamics

- Continual Learning is a huge challenge for deep learning models because of **gradient-based optimization**.
- Gradient-based learning is effective and cheap, the de rigeur method for training neural networks for close to 4 decades.
- However, a close look at the learning dynamics reveals a problem.
- Each training sample produces a gradient for each parameter in the network that votes to make the parameter bigger or smaller.
- In a mini-batch, a gradient is produced by each sample in parallel and they are summed to decide the winning direction.
- The result is a **tug-of-war** over the direction of change of each parameter.





Gradient-based optimization and tug-of-war dynamics

How efficient is gradient-based optimization for stationary learning? Are all parts of the dataset learned at the same speed? The answer is No.

- Most examples in a dataset are learned fast and then multiple repetitions are needed to learn remaining examples.[1,2] However, the tug-of-war dynamics require that all examples are present, even the easier ones, which wastes computational resources.
- Recent work has shown empirically that concepts are discovered sequentially, even if they are simultaneously present in the data[3].
- Therefore, **even if tasks are equally complex and presented simultaneously, the model might still learn them sequentially**, thus losing efficiency due to the tug-of-war dynamics.

Continual learning could unleash unprecedented learning efficiency, even in stationary learning settings.

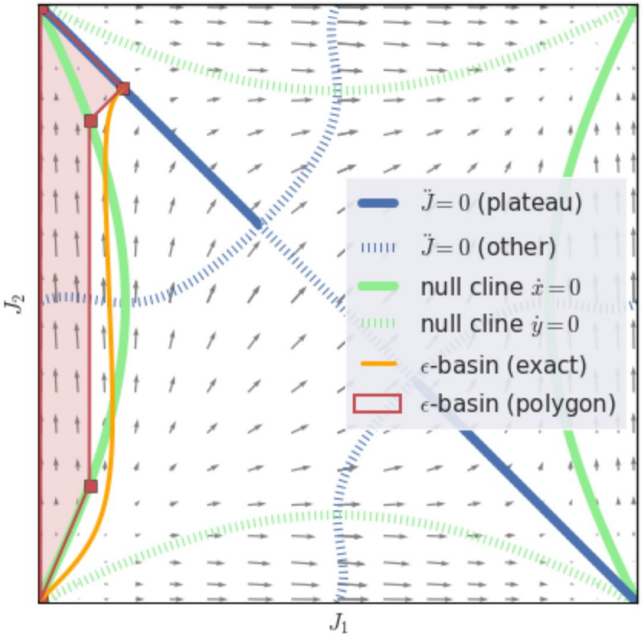
1. Devansh A. et al. A closer look at memorization in deep networks, ICML 2017;
2. Chang H.-S. et al., Active bias: training more accurate neural networks by emphasizing high variance samples, NeurIPS 2017
3. Raposo D. et al. Discovering objects and their relations from entangled scene representations. arXiv 2017
4. Schaul, T et al, Ray Interference: Source of Plateaus in Deep Reinforcement Learning, arxiv, 2019

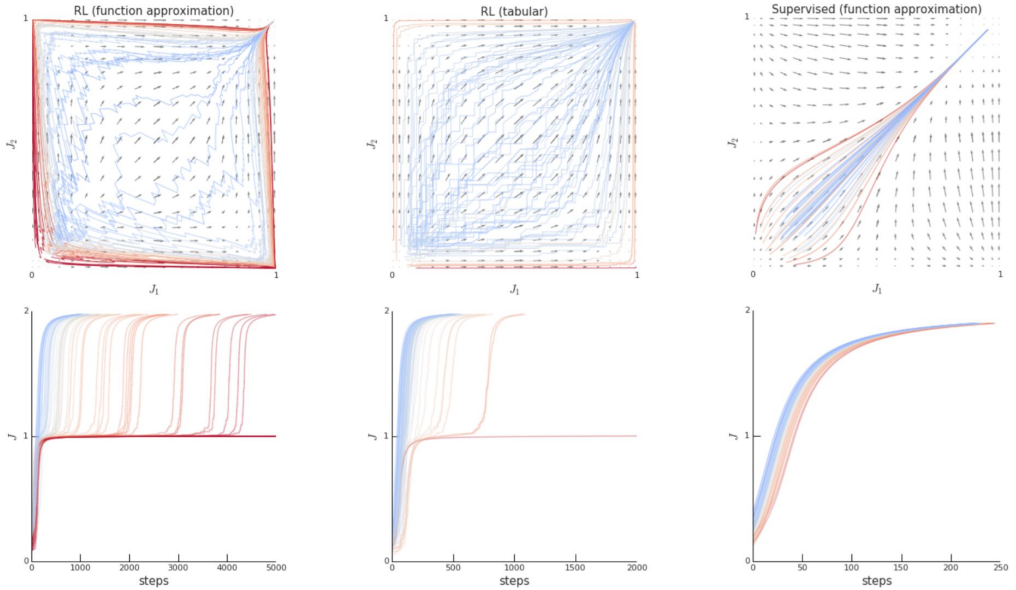
Gradient-based optimization and tug-of-war dynamics

$$J := \sum_{1 \leq k \leq K} J_k$$

Schaul et al. 2019, *Ray Interference:a source of Plateaus in Deep Reinforcement Learning*

A system suffers from plateaus if it has (a) negative interference between the components of its objective, and (b) coupled performance and learning progress





Gradient-based optimization and tug-of-war dynamics

Learning has **tug-of-war** dynamics to resolve credit assignment

- This requires data to be **I.I.D.**
- **No explicit knowledge composition**

- Failure in credit assignment leads to catastrophic forgetting

Continual Learning studies credit assignment in learning, and in particular the effects of this tug-of-war dynamics.

Resolving continual learning means finding better/different mechanism of doing credit assignments. CL is about optimization/learning.

It has big implications for our understanding of the learning process for neural networks.

4 Continual Learning landscape

Looking for solutions

Continual learning can be understood as a set of approaches to stabilize the tug-of-war learning dynamics over a sequence of tasks **without having the previous tasks available.** The landscape is relatively complex, where any family of approaches bleeds into another (and there are several other topics like task-inference or task-boundary detection).

There are also tight connections between continual learning/life-long learning and meta-learning, transfer learning, never-ending learning, domain adaptation, online learning, multi-task learning.

Regularization-based solutions

- Use regularization to change the plasticity of *some* parameters (identifying which ones, and how much to regularize, is the hard part)
  - Elastic Weight Consolidation, Kirkpatrick 2017
  - Synaptic Intelligence, Zenke 2017
- Use distillation to mitigate parameter drift
  - Learning Without Forgetting, Li 2017

$$L(\theta) = L_A(\theta) + L_B(\theta) + L_C(\theta)$$

$$\downarrow$$

$$L(\theta) = (\theta - \theta_A)\frac{\partial^2 L_A(\theta)}{\partial \theta^2}(\theta - \theta_A) + (\theta - \theta_B)\frac{\partial^2 L_B(\theta)}{\partial \theta^2}(\theta - \theta_B) + L_C(\theta)$$

Memory-based solutions for Continual Learning

- Simple but effective: Replay (saving a buffer of past experience) and Episodic memory (also doing inference on the past experience)
  - Catastrophic forgetting, rehearsal and pseudorehearsal, Robins 1995
  - Experience replay for CL, Rolnick 2019
  - Episodic memory in lifelong learning, d'Autume 2019
  - Memory-based Parameter Adaptation, Sprechmann 2018
- More scalable: use exemplars or memory vectors in a sparse memory setting
  - iCaRL, Rebuffi 2017
  - Functional Regularization, Titsias 2019
  - Using hindsight to anchor.., Chaudhry 2020
- More biologically plausible: use generative models to remove storage requirements
  - Deep generative replay, Shin 2017
  - Continual unsupervised representation learning (CURL), Rao 2019

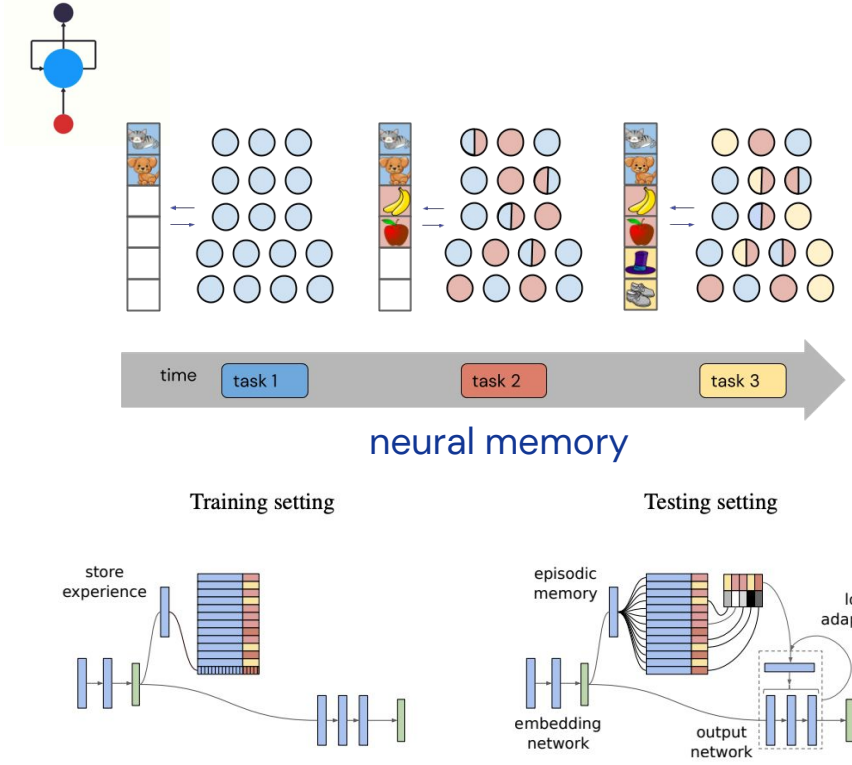


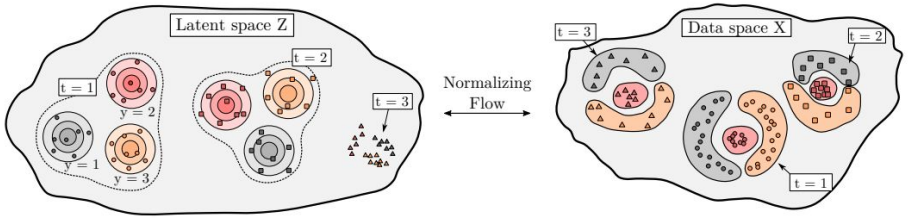
Figure 1: Architecture for the MbPA model. Left: Training usage. The parametric network is used directly and experiences are stored in the memory. Right: Testing setting. The embedding is used to query the episodic memory, the retrieved context is used to adapt the parameters of the output network.

Memory-based Parameter Adaptation (MbPA)

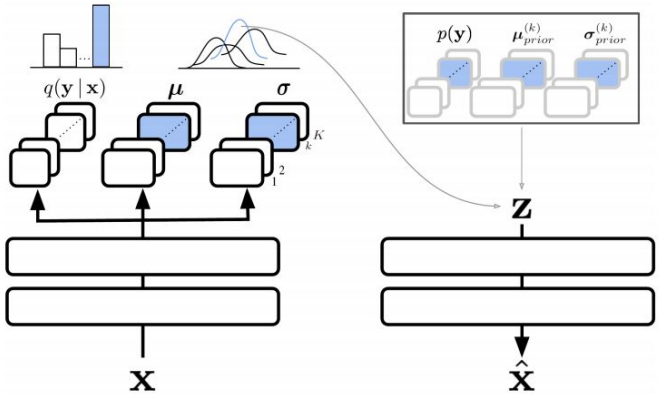
Memory-based solutions for Continual Learning

- This seems a particularly natural choice when the task itself is generative
- It also opens the door towards non-parametric architectures (components)
- Also it can become a bridge between rehearsal/memory based methods and regularization based methods
- E.g. for HPM, one can build a regularizer to check whether the flow of the new model maps points the same way as the original flow

Kirichenko et al. 2021, *Task-agnostic Continual Learning with Hybrid Probabilistic Models*



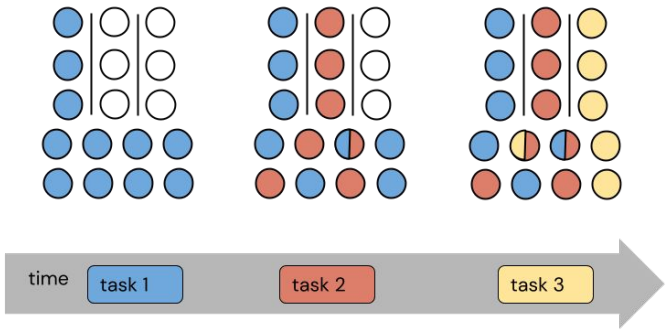
Rao et al. 2019, *Continual Unsupervised Representation Learning*



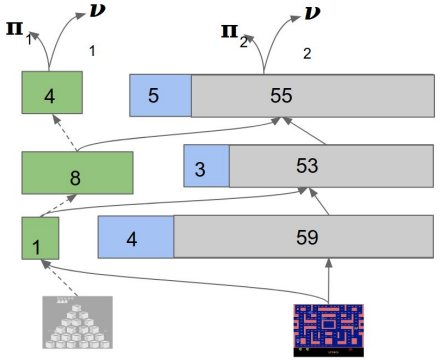
Modularity and sparsity

Modularity offers a middle ground between a monolithic architecture and an ensemble.

- Add on new capacity for new tasks (requires task boundaries)
  - Progressive Neural Nets, Rusu 2016
  - Dynamically Expandable Nets, Yoon 2018
  - Neurogenesis deep learning, Draelos 2017
  - Reinforced Continual Learning, Xu 2018
- Compress or prune to scale further
  - Progress & Compress, Schwarz 2018
  - Continual Learning via neural pruning, Golkar 2019



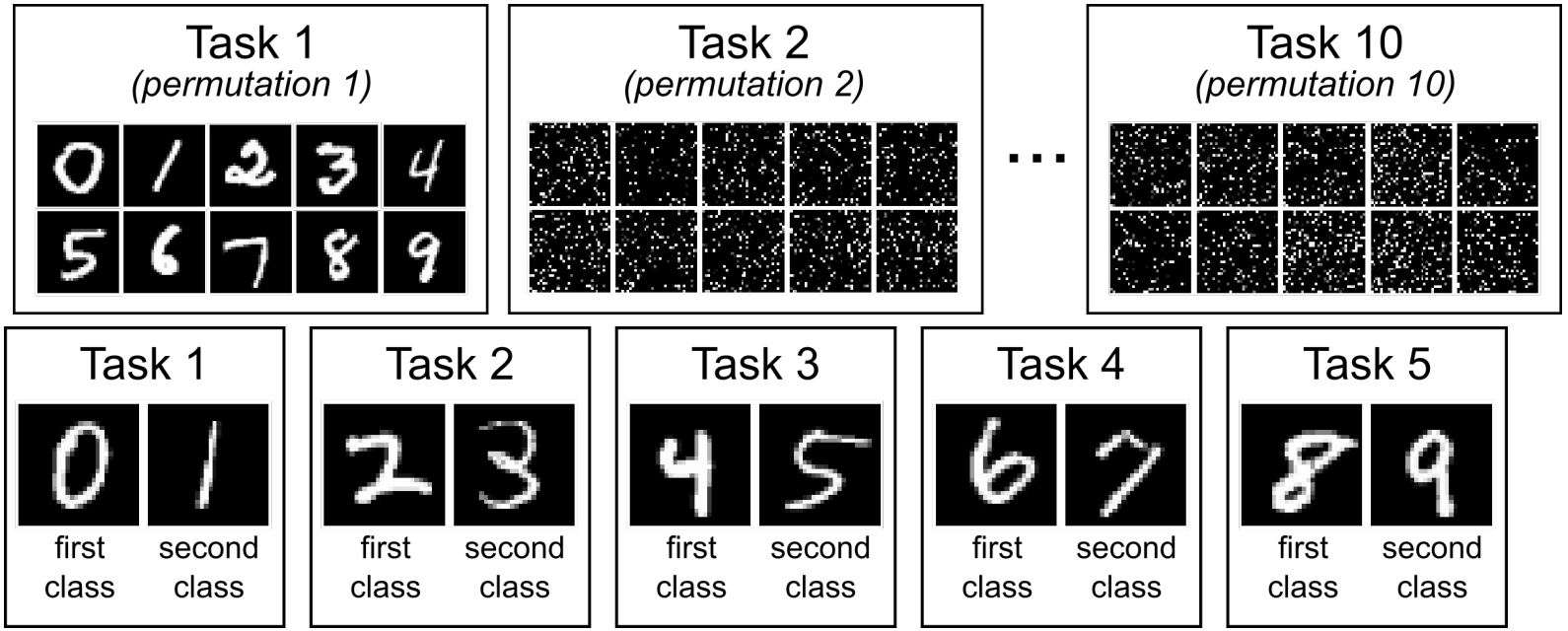
modular architectures



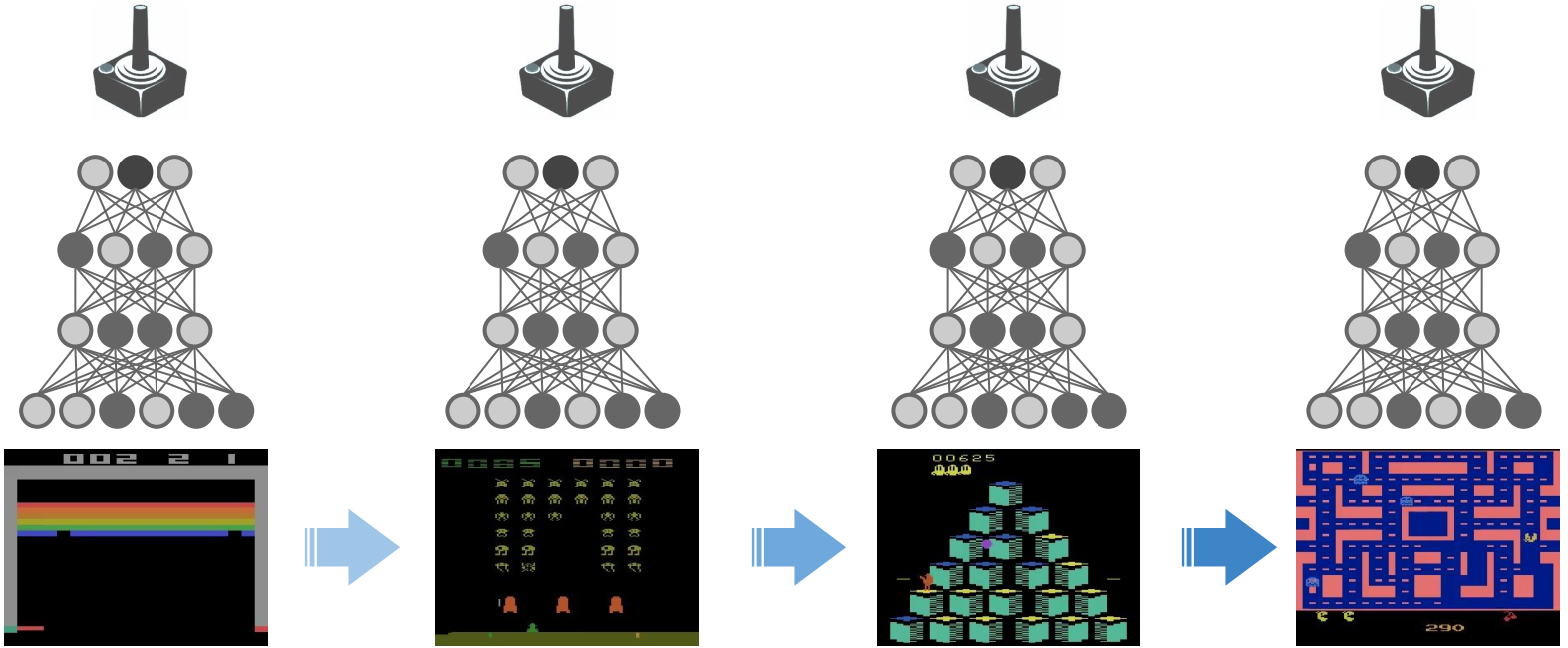
Neural Pruning

5 Benchmarks for Continual Learning

permuted-MNIST and split-{MNIST, CIFAR-10, CIFAR-100, Fashion MNIST, Omniglot, SVHN, ImageNet, miniImageNet, etc.}

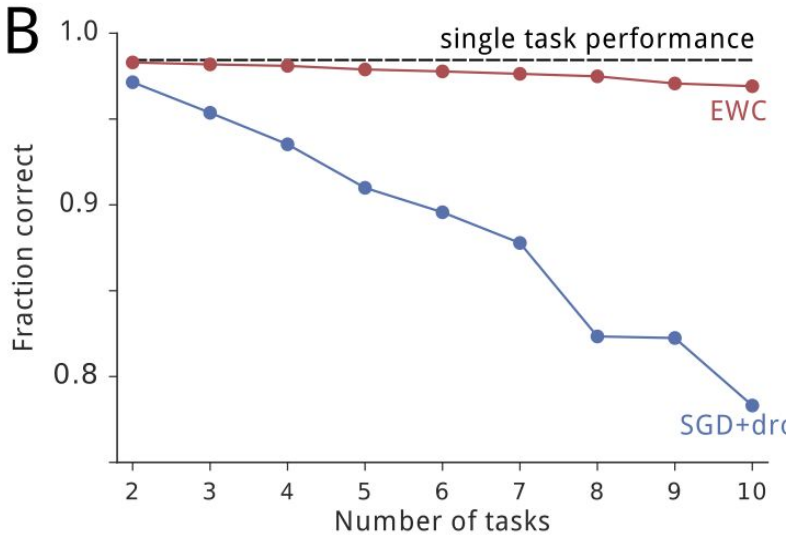


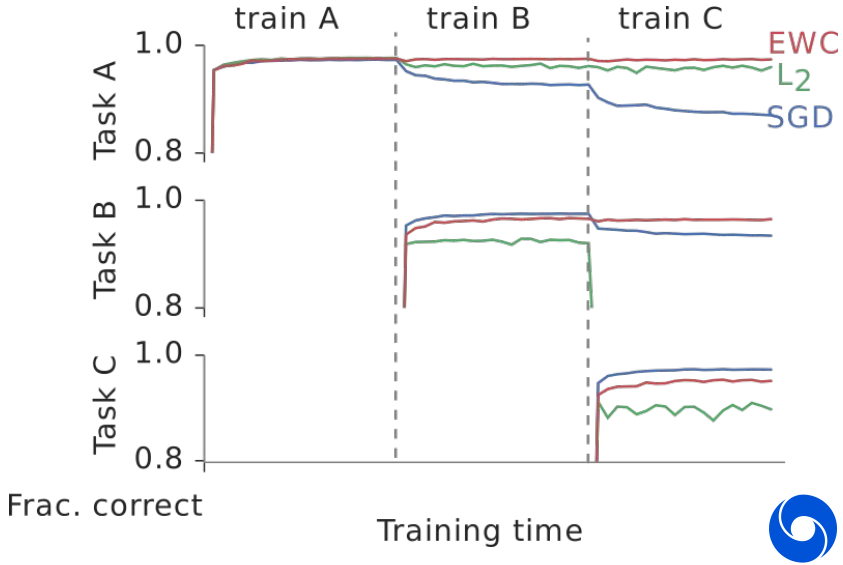
Atari



Typical benchmarks

- Provide a series of task (with a predefined order)
- Provide amount of learning in each task (e.g. 10M steps)
- User can exploit the knowledge of task switches points
- Usually no computational/memory constraint (though the user should try to avoid it)
- Other details (e.g. different output layers/shared output layers)





Methods & results

Regularisation-based

- L2
- EWC
- MAS
- VCL

Modularity

- PackNet

Replay/Memory based

- Perfect Memory
- A-GEM

method	performance	forgetting	f. transfer
Fine-tuning	0.05 [0.05, 0.06]	0.73 [0.72, 0.75]	**0.20 [0.17, 0.23]**
L2	0.43 [0.39, 0.47]	0.02 [0.00, 0.03]	-0.75 [-0.87, -0.65]
EWC	0.60 [0.56, 0.64]	0.02 [-0.00, 0.05]	-0.19 [-0.25, -0.14]
MAS	0.51 [0.49, 0.53]	0.00 [-0.01, 0.02]	-0.52 [-0.58, -0.48]
VCL	0.48 [0.46, 0.50]	0.01 [-0.01, 0.02]	-0.48 [-0.56, -0.42]
PackNet	**0.80 [0.79, 0.82]**	0.00 [-0.01, 0.01]	**0.18 [0.14, 0.21]**
Perfect Memory	0.14 [0.13, 0.16]	0.05 [0.04, 0.06]	-1.37 [-1.46, -1.30]
A-GEM	0.07 [0.07, 0.08]	0.72 [0.71, 0.74]	0.17 [0.13, 0.20]
MT	0.50 [0.47, 0.54]	—	—
MT (PopArt)	0.66 [0.62, 0.70]	—	—
RT	—	—	**0.46**

Table 1. Results on CW20, for CL methods and multi-task training. Metrics are defined in Section 4.1, RT is eq. (4). We used 20 seeds and provide 90% confidence intervals.

The distractor problem

The distractor problem

A →B →C →D→…

To appear soon on arxiv …

- Assume we have a sequence of tasks **A**, **B**, **C** where **A** *is related to* **C** but **B** *is not related to* ***C***
- **B** acts as the distractor, and typical learning will lead to *forgetting* **A** when learning **B**, and hence when we learn **C** we can not transfer from **A** anymore

Why is this setup useful ?

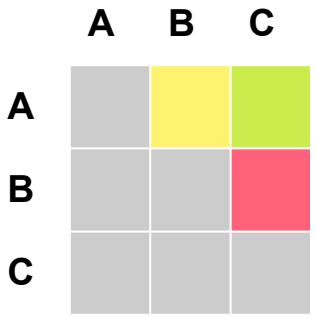
- It provides a way to reason about forward transfer and forgetting
- It can provide a target of the CL algorithm : *mitigate the interference produced by the distractor*

The distractor problem

A →B →C →D→…

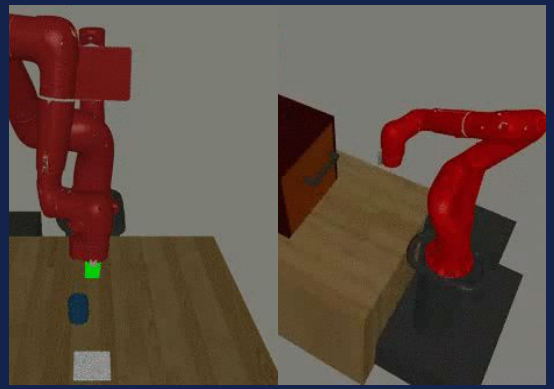
To appear soon on arxiv …

- Assume we have a sequence of tasks **A**, **B**, **C** where **A** *is related to* **C** but **B** *is not related to* ***C,*** *and hence* ***B*** *is the distractor*
- Defining *task similarity* is hard. But one can rely on a transfer matrix for this
- In general this can give you a target but not an upper bound (as there can be compositionality of knowledge), which can be maximum over the row of the matrix up to the current task
- There are other effects beside interference (e.g. reduced effective capacity of the neural network)..

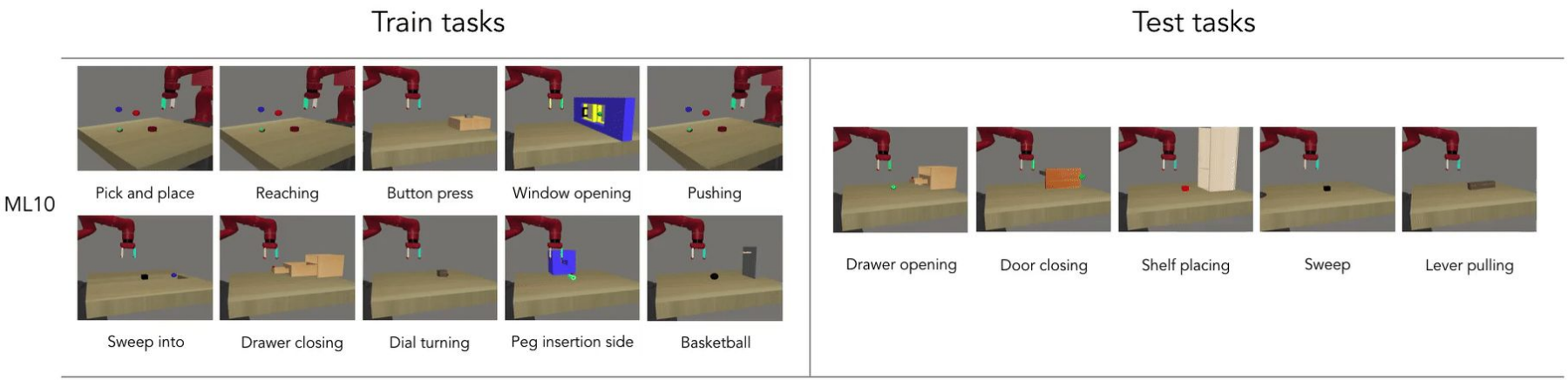


Continual Meta-World

https://arxiv.org/abs/2105.10919



Meta-World



- Robotics, hence of some pragmatic relevance
- Relationship between tasks better than traditional datasets (e.g. permuted MNIST, Atari)
- Externally defined
- One can work from states, avoiding some issues of RL learning

Metrics

Average performance

$$\mathrm{P}(t) := \frac{1}{N}\sum_{i=1}^{N} p_i(t),$$

Forward Transfer

$$\mathrm{FT}_i = (\mathrm{AUC}_i - \mathrm{AUC}_i^b)/(1 - \mathrm{AUC}_i^b),$$

$$\mathrm{AUC}_i := \frac{1}{\Delta}\int_{(i-1)\cdot\Delta}^{i\cdot\Delta} p_i(t)\mathrm{d}t$$

$$\mathrm{AUC}_i^b := \frac{1}{\Delta}\int_{0}^{\Delta} p_i^b(t)\mathrm{d}t,$$

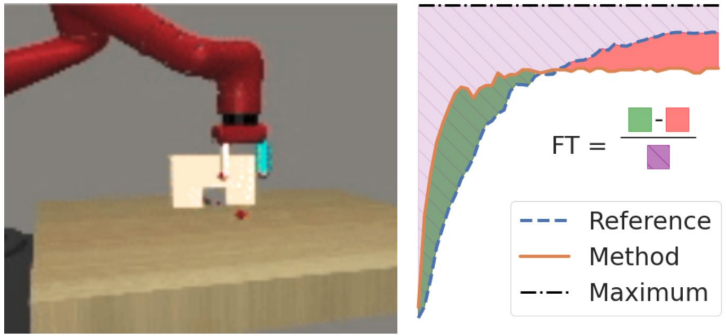


Figure 1. Left graph shows task PEG-UNPLUG-SIDE-V1 and the right graph presents forward transfer from SHELF-PLACE-V1 to PEG-UNPLUG-SIDE-V1. In this case $FT = 0.10$.

Forgetting

$$F_i = p_i(T) - p_i(i \cdot \Delta).$$

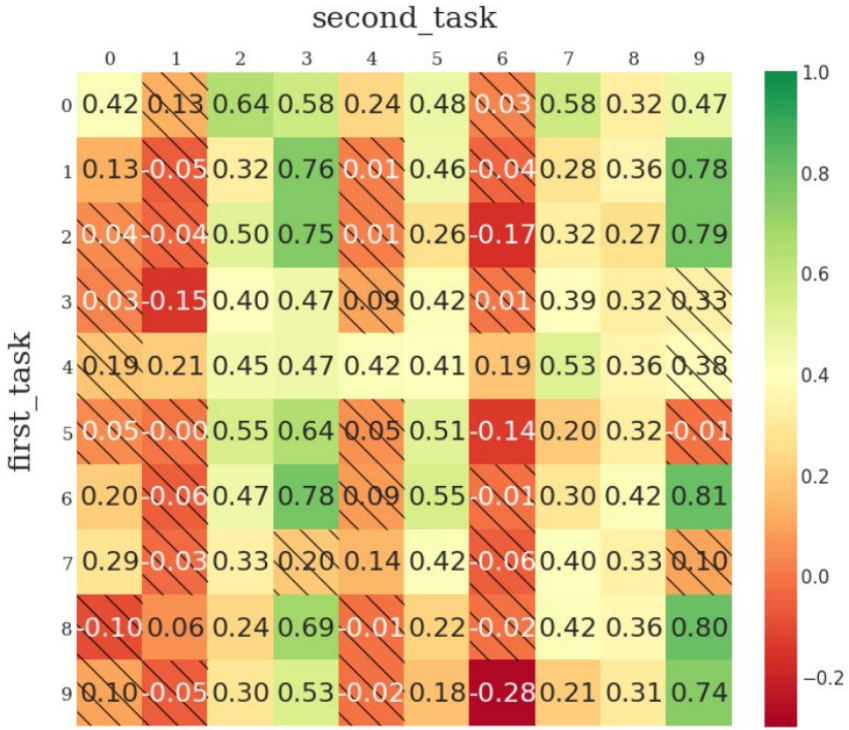
Transfer and Similarity

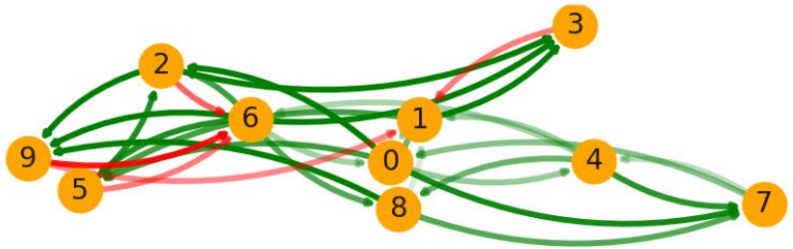
- Sequence of 20 tasks (this 10 tasks repeated twice)
- Additional pairs of 3 tasks to study interesting phenomena

Define a reference transfer as:

$$\mathrm{RT} := \frac{1}{N}\sum_{i=2}^{N} \max_{j<i} \mathrm{FT}(t_j, t_i),$$

For this dataset, the value is **0.46**





Conclusions

- Simple domains like this one could already provide a signal for seeing what works and what does not
- Existing Continual learning algorithms considered to not exhibit forward transfer.
- Some interesting RL specific observations :)

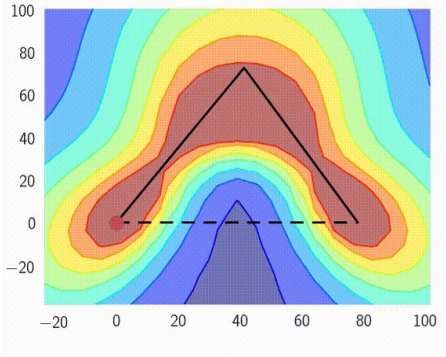
method	performance	forgetting	f. transfer
Fine-tuning	0.05 [0.05, 0.06]	0.73 [0.72, 0.75]	**0.20 [0.17, 0.23]**
L2	0.43 [0.39, 0.47]	0.02 [0.00, 0.03]	-0.75 [-0.87, -0.65]
EWC	0.60 [0.56, 0.64]	0.02 [-0.00, 0.05]	-0.19 [-0.25, -0.14]
MAS	0.51 [0.49, 0.53]	0.00 [-0.01, 0.02]	-0.52 [-0.58, -0.48]
VCL	0.48 [0.46, 0.50]	0.01 [-0.01, 0.02]	-0.48 [-0.56, -0.42]
PackNet	**0.80 [0.79, 0.82]**	0.00 [-0.01, 0.01]	**0.18 [0.14, 0.21]**
Perfect Memory	0.14 [0.13, 0.16]	0.05 [0.04, 0.06]	-1.37 [-1.46, -1.30]
A-GEM	0.07 [0.07, 0.08]	0.72 [0.71, 0.74]	0.17 [0.13, 0.20]
MT	0.50 [0.47, 0.54]	—	—
MT (PopArt)	0.66 [0.62, 0.70]	—	—
RT	—	—	**0.46**

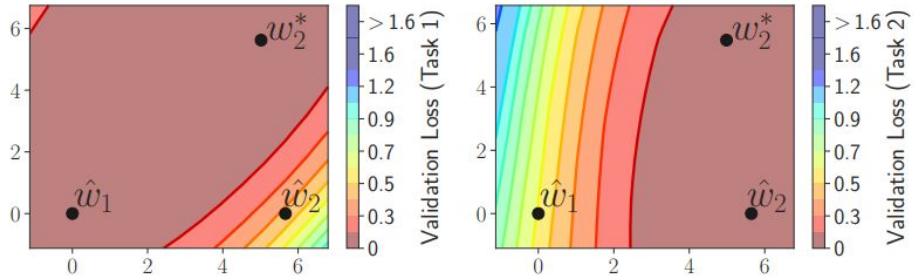
Table 1: Results on CW20, for CL methods and multi-task training. Metrics are defined in Section 4.1, RT is eq. (4). We used 20 seeds and provide 90% confidence intervals.

Weaknesses of evaluation: The gaussian assumption

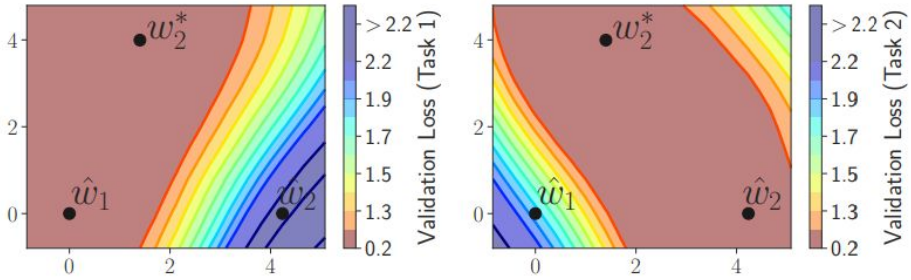
Linear mode connectivity in Multitask and Continual Learning

https://arxiv.org/abs/2010.04495

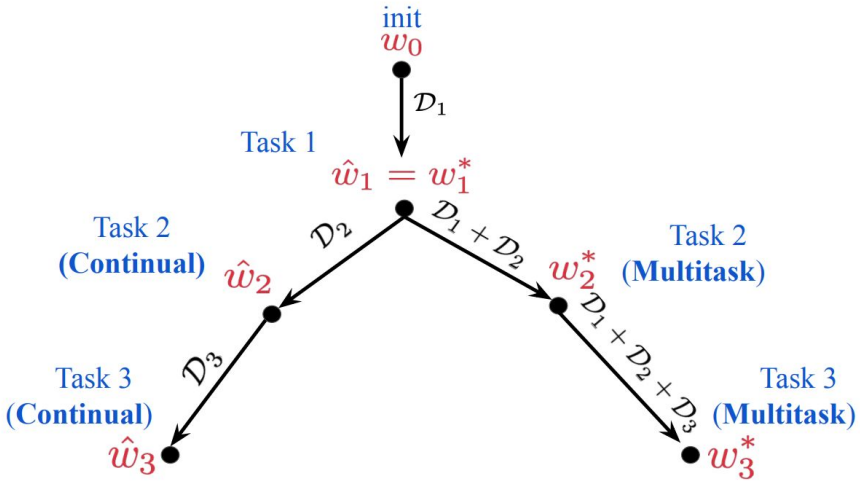


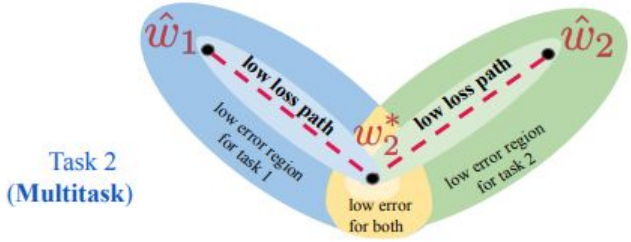


(a) MNIST Task 1 loss (b) MNIST Task 2 loss



(c) CIFAR Task 1 loss (d) CIFAR Task 2 loss

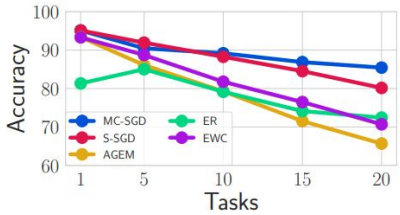




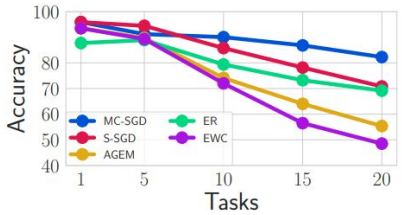
Conclusions

- For most standard vision tasks, linear mode connectivity holds
- That implies that the multitask solution when reaching task *N* is within the region where the 2nd order Taylor expansion holds
- That is a Gaussian approximation of the posterior (typical regularization based CL methods) can do as well as rehearsal based methods
- Are these tasks too simple ?!

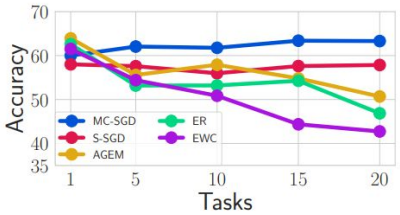
Method	Permuted MNIST		Rotated MNIST		Split CIFAR100	
	Accuracy ↑	Forgetting ↓	Accuracy ↑	Forgetting ↓	Accuracy ↑	Forgetting ↓
Naive SGD	44.4 (±2.46)	0.53 (±0.03)	46.3 (±1.37)	0.52 (±0.01)	40.4 (±2.83)	0.31 (±0.02)
EWC (Kirkpatrick et al., 2017)	70.7 (±1.74)	0.23 (±0.01)	48.5 (±1.24)	0.48 (±0.01)	42.7 (±1.89)	0.28 (±0.03)
A-GEM (Chaudhry et al., 2018b)	65.7 (±0.51)	0.29 (±0.01)	55.3 (±1.47)	0.42 (±0.01)	50.7 (±2.32)	0.19 (±0.04)
ER-Reservoir (Chaudhry et al., 2019)	72.4 (±0.42)	0.16 (±0.01)	69.2 (±1.10)	0.21 (±0.01)	46.9 (±0.76)	0.21 (±0.03)
Stable SGD (Mirzadeh et al., 2020)	80.1 (±0.51)	0.09 (±0.01)	70.8 (±0.78)	0.10 (±0.02)	59.9 (±1.81)	0.08 (± 0.03)
MC-SGD (ours)	**85.3 (±0.61)**	**0.06 (±0.01)**	**82.3 (±0.68)**	**0.08 (±0.01)**	**63.3 (±2.21)**	**0.06 (± 0.03)**
Multitask Learning	89.5 (±0.21)	0.0	89.8(±0.37)	0.0	68.8(±0.72)	0.0



(a) Permuted MNIST



(b) Rotated MNIST



(c) Split CIFAR-100

I want to give thanks to my collaborators (in random order) :

- Raia Hadsell
- Yee Whye Teh
- Andrei Rusu
- Dushyant Rao
- Alexandre Galashov
- Amal Rannen-Triki
- Christos Kaplanis
- Mehrdad Farajtabar
- Seyed Iman Mirzadeh
- Piotr Miłoś
- Lukasz Kucinski
- Maciej Wolczyk
- Michal Zajac
- Siddhant Jayakumar
- Jonathan Schwarz
- Tudor Berariu
- Florin Gogianu
- Claudia Clopath
- Doina Precup
- Nicolas Heess
- Omar Rivasplata
- Jorg Bornschein
- Peter Battaglia
- Xu Ji
- Balaji Lakshminarayanan
- Eric Elsen
- Simon Osindero
- Jack Rae
- Wojciech Czarnecki
- Soham De
- Samuel Smith
- Andrew Brock
- Stanislav Forth
- Mihaela Rosca
- Lucian Busoniu
- Caglar Gulcehre
- Ziyu Wang
- Owen He
- Jakub Sygnowski
- Thomas Paine
- Konrad Zolna
- Yutian Chen
- Matthew Hoffman
- Ilja Kuzborskij
- Sheheryar Zaidi
- Jovana Mitrovic
- Sebastien Flennerhag
- Jane Wang
- Andre Barreto
- Francesco Visin
- Petar Velickovic
- Charles Blundell
- Michal Valko
- Dilan Gorur
- Nandos de Freitas
- Dhruva Tirumala
- Matko Bosnjak
- Srivatsan Srinivasan
- Ang Li
- Huiyi Hu
- Polina Kirichenko
- Viorica Patraucean
- Ștefan Dumitrescu
- Petru Rebeja
- Beata Lorincz
- Adriana Stan
- Radu Ionescu
- **... (many others)**

Thank you !

Any questions ?