Continual Learning in Complex Environments

Raia Hadsell
Director of Robotics, DeepMind

1

The world is non-stationary

The world is non-stationary.

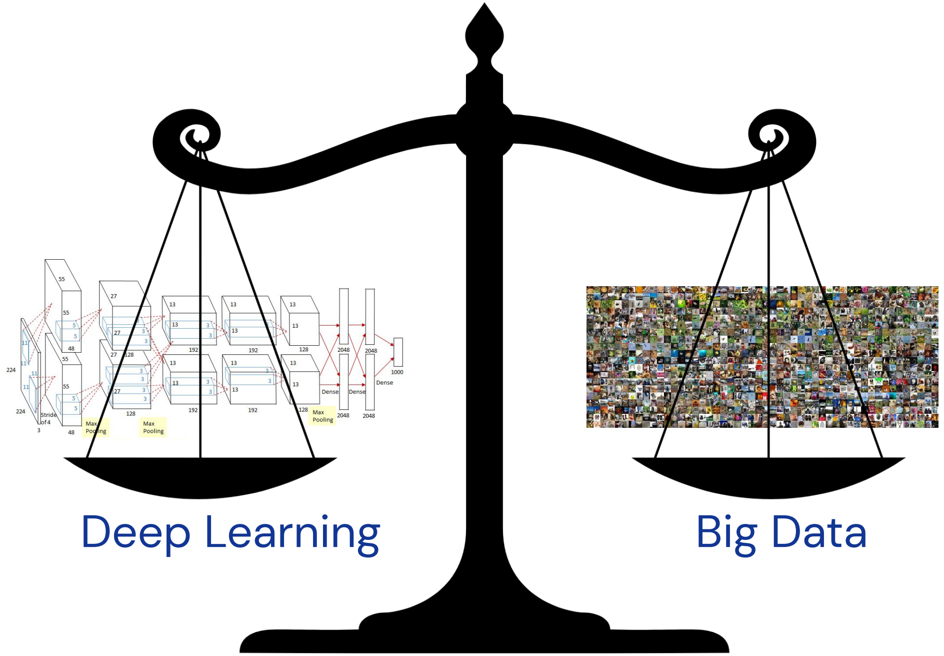
When we build and deploy machine learning algorithms, we assume that our problem domain is fixed and that all variability is captured in a static dataset or learning environment.

This is rarely true. For example:

- Health
- Robotics
- Language

Deep Learning is optimised for static, large-scale datasets

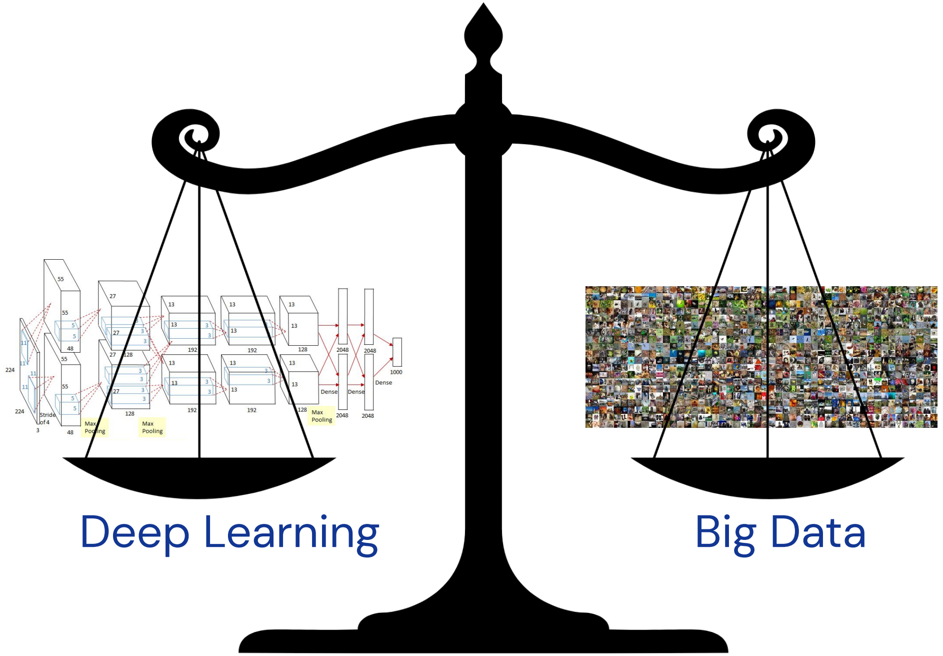
- Deep Learning is powerful because the models readily scale to fit large datasets.



Deep Learning is optimised for static, large-scale datasets

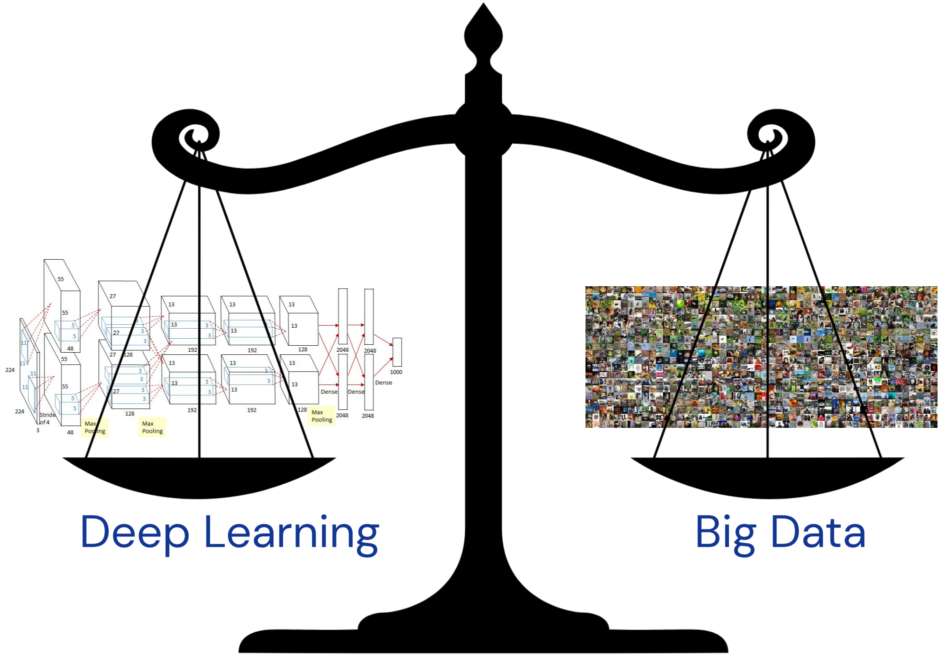
- Deep Learning is powerful because the models readily scale to fit large datasets.

➡ Supported by large-scale compute, and end-to-end optimization.



Deep Learning is optimised for static, large-scale datasets

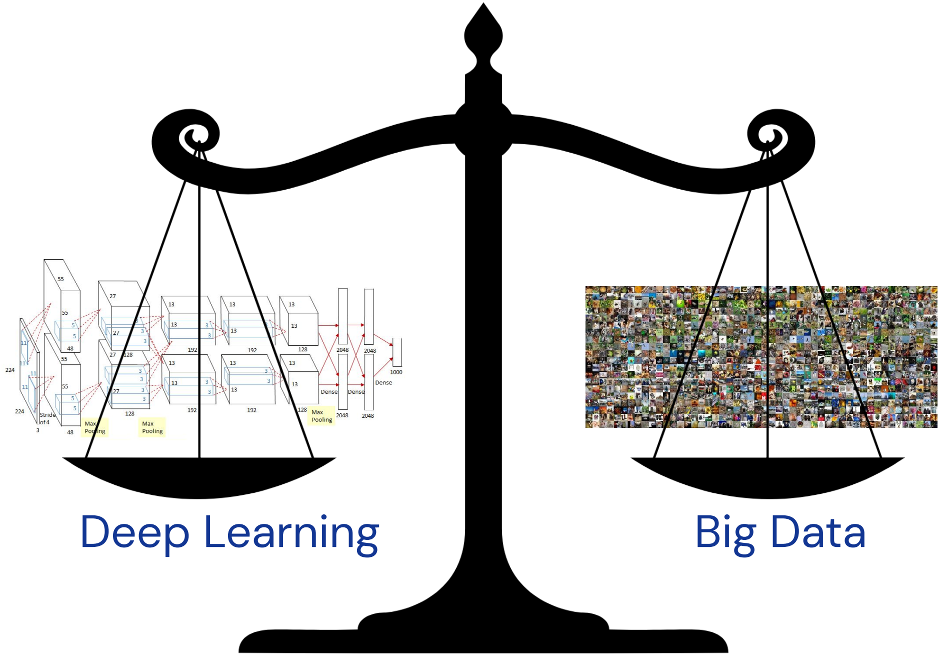
- Deep Learning is powerful because the models readily scale to fit large datasets.
- Supported by large-scale compute, and end-to-end optimization.
- Optimization is gradient-based, which assumes that the dataset is balanced, shuffled and randomly sampled during training (i.i.d. - more on this later).



Deep Learning is optimised for static, large-scale datasets

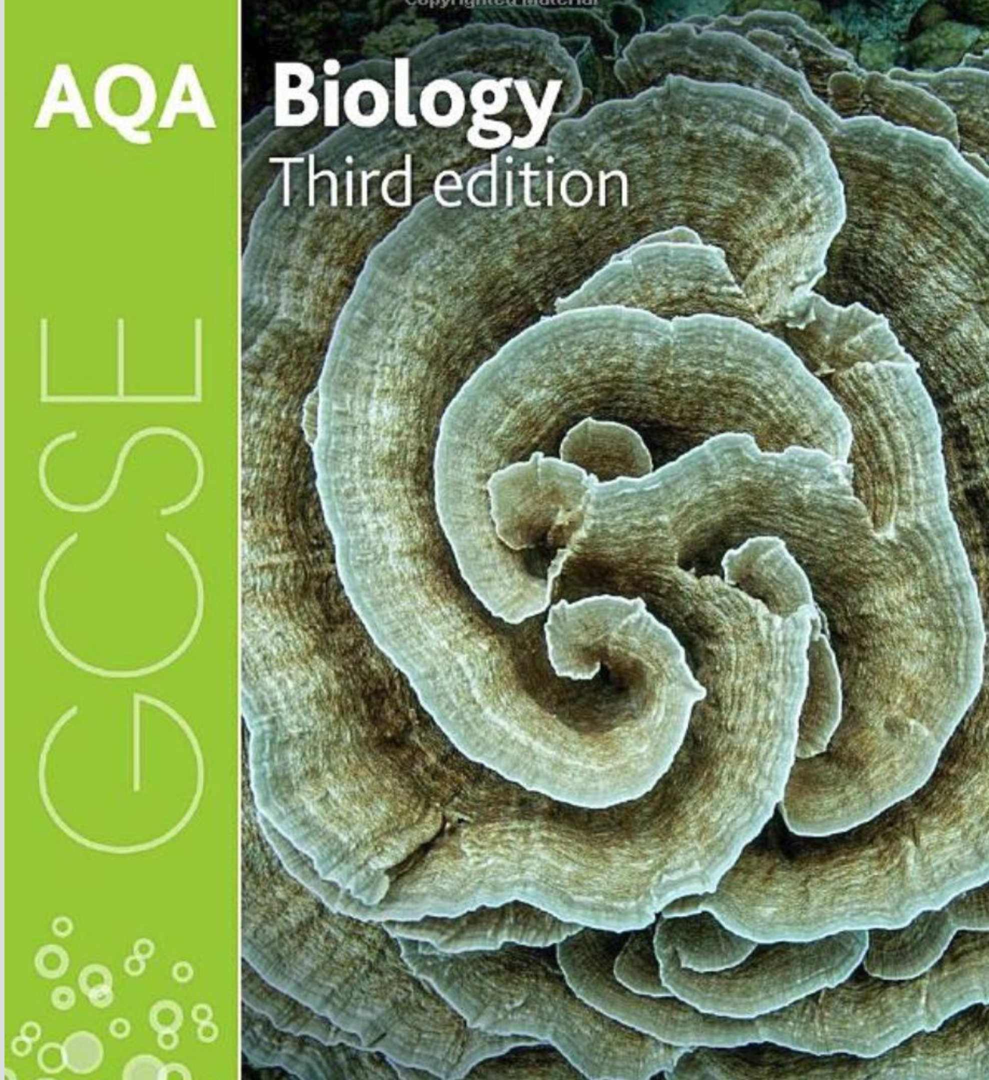
- Deep Learning is powerful because the models readily scale to fit large datasets.
- Supported by large-scale compute, and end-to-end optimization.
- Optimization is gradient-based, which assumes that the dataset is balanced, shuffled and randomly sampled during training (i.i.d. - more on this later).

➡ The result is that deep learning models are inefficient at learning, and may suffer from catastrophic forgetting, interference, and other failure modes if trained in non-stationary settings



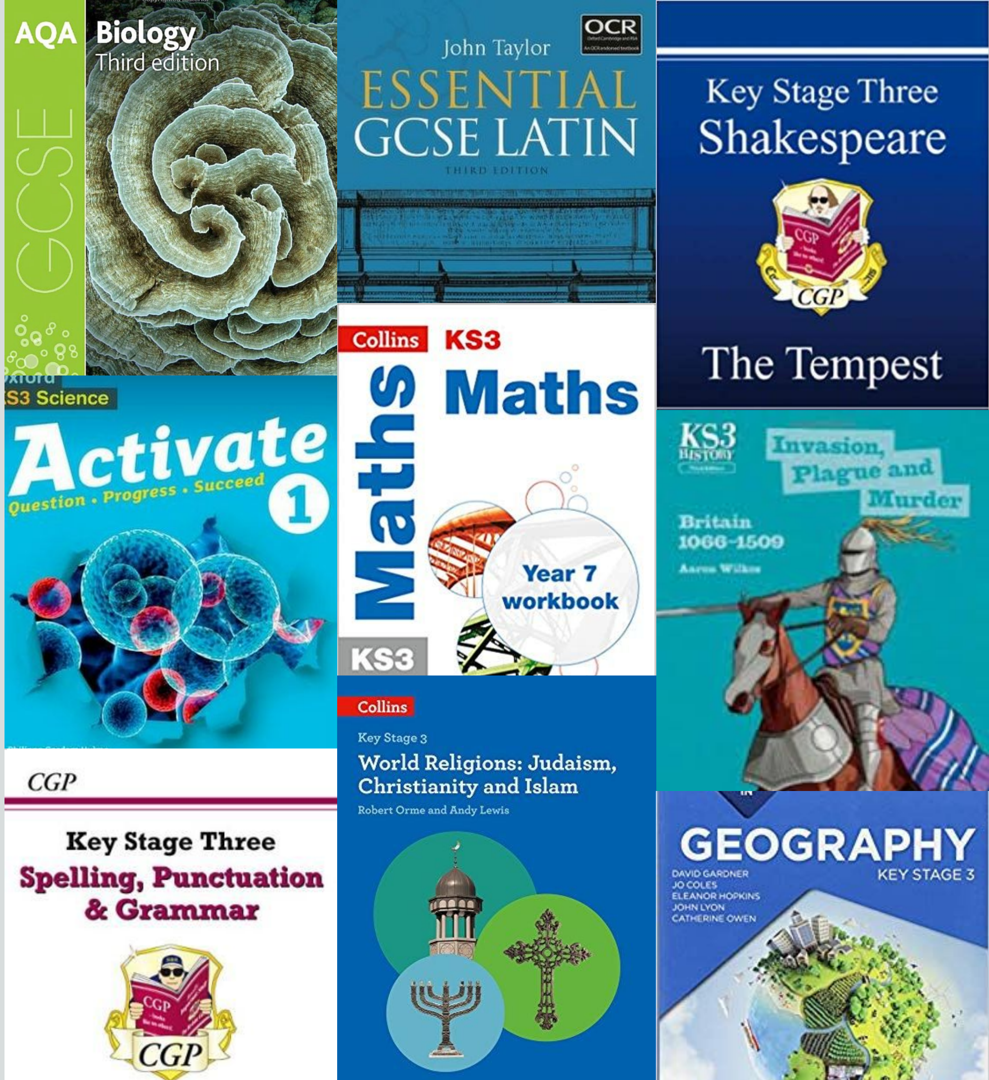
Humans don’t learn well from randomly sampled data

Imagine learning high school biology by sampling pages at random.



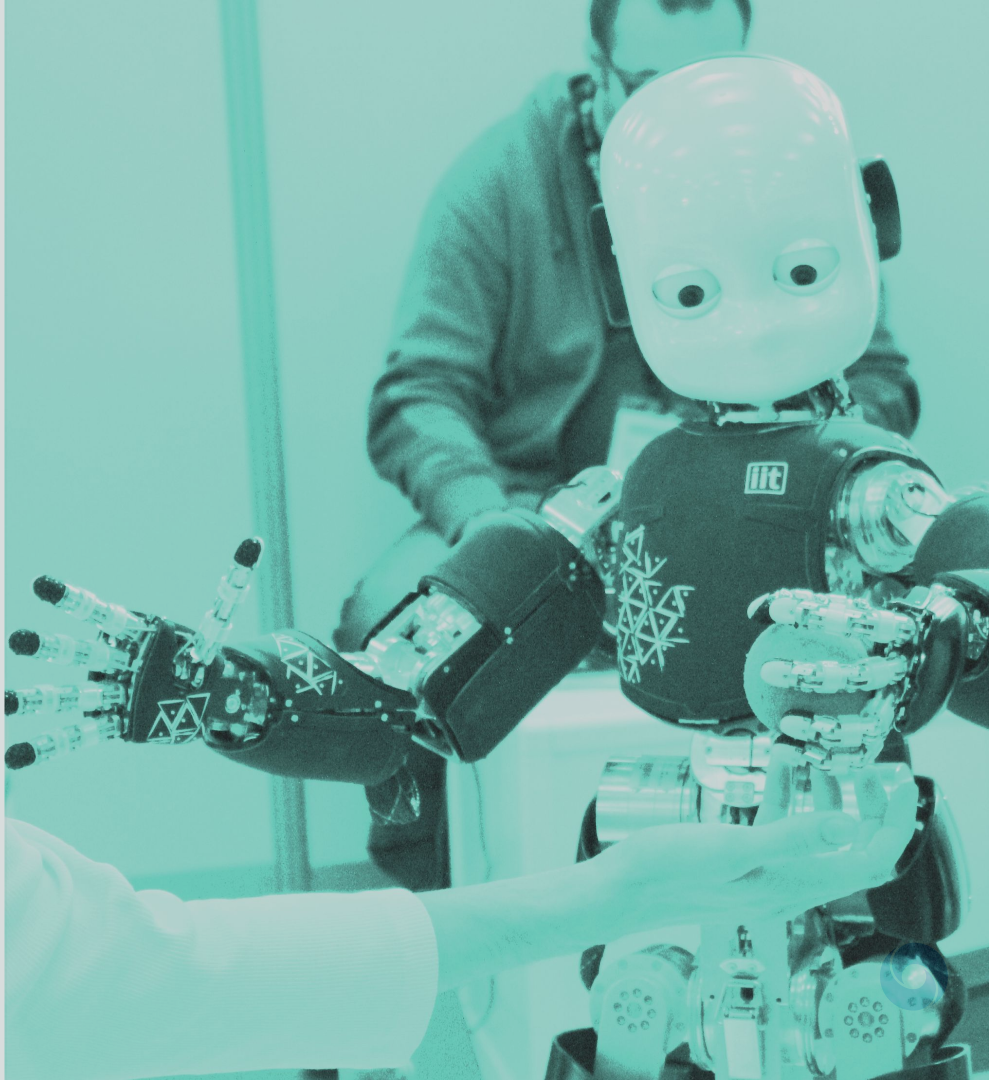
Humans don't learn well from randomly sampled data

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



What could be gained by enabling Deep Learning to learn *sequentially*?

1. Applications that continually adapt to track a changing problem, or specialize to a domain (e.g. epidemiological models, language specialization)
2. Robots that add skills over time, becoming more capable.
3. AGI - continual learning is considered to be requisite to achieve human-level intelligence[1]
4. Dramatically more efficient Deep Learning methods, even in stationary settings.



1. Hassabis D. et al., Neuroscience-inspired artificial intelligence. Neuron 2017

2 Defining Continual Learning

The Continual Learning Problem

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

This is differentiated from Curriculum Learning, in which the sequence of tasks is controlled by the learner (or a beneficent teacher).

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. **Fast adaptation and recovery.** The model should be capable of fast adaptation to novel tasks or domain shifts and of fast recovery when presented with past tasks.

Continual Learning Solutions

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

4. **Minimizing catastrophic forgetting and interference**. Training on new tasks should not significantly reduce performance on previously learned tasks.

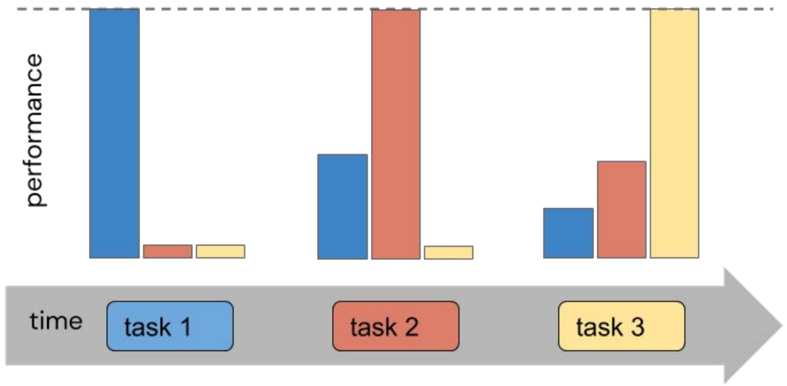


Illustration of catastrophic forgetting

Continual Learning Solutions

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

5. **Maintaining plasticity.** The model should be able to keep learning effectively as new tasks are observed.

   (Failure often occurs because of regularization or lack of model capacity.)

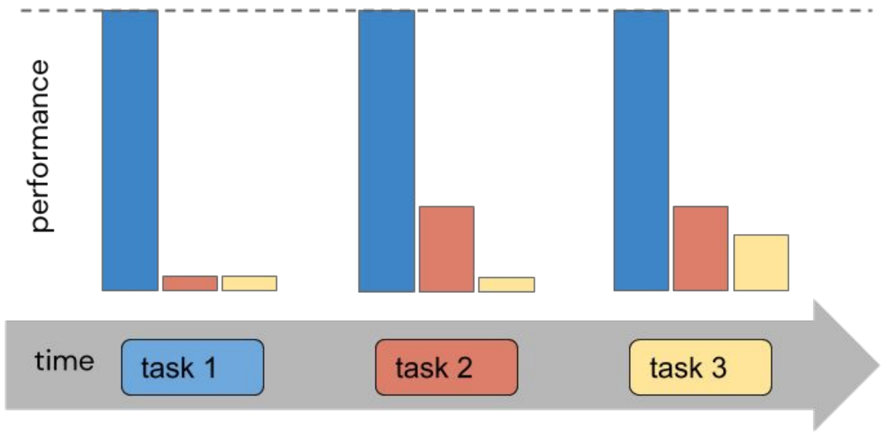


Illustration of declining plasticity, or learning intransigence.

Continual Learning Solutions

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

6. **Maximizing forward and backward transfer.** Learning a task should improve related tasks, both past and future, in terms of both learning efficiency and performance

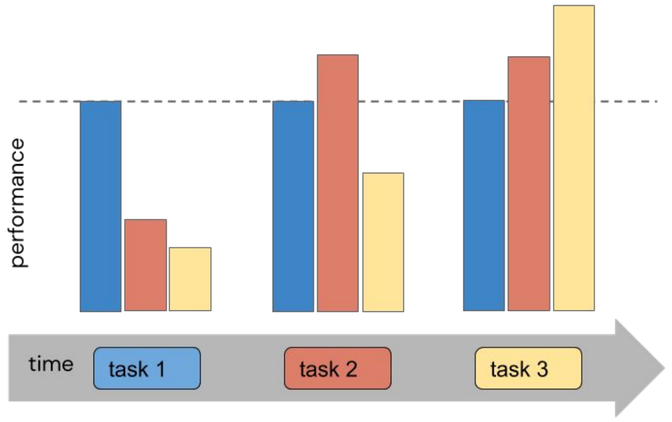


Illustration of forward transfer.

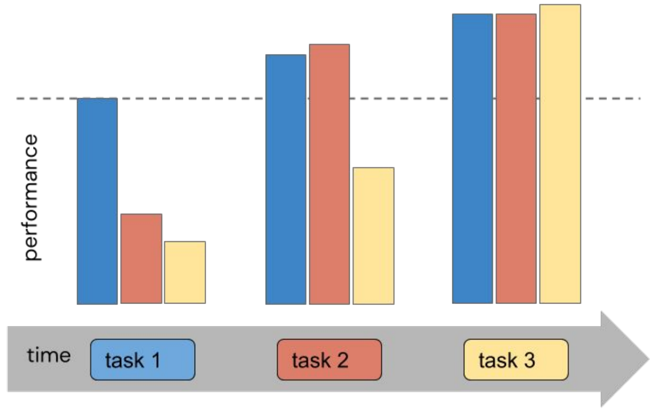


Illustration of forwards and backwards transfer.

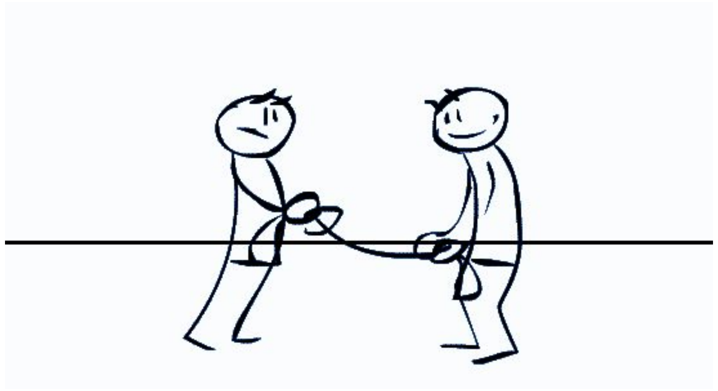
Contradictions, dilemmas, and trade-offs

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?
- Real-world problem domains resolve these dilemmas, however.

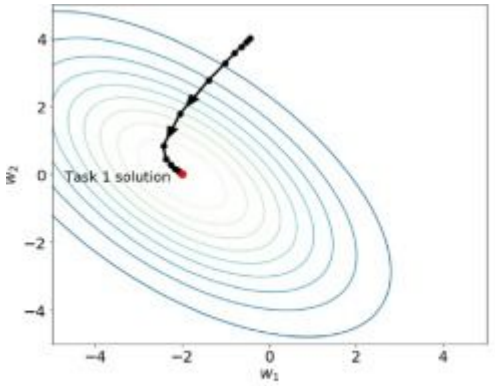
3 Tug-of-war learning dynamics

and the I.I.D. assumption



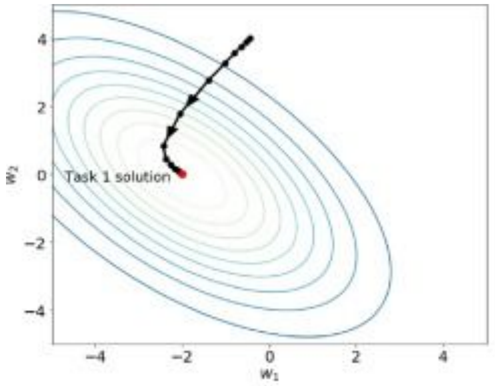
Gradient-based optimization and tug-of-war dynamics

- 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

- 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.

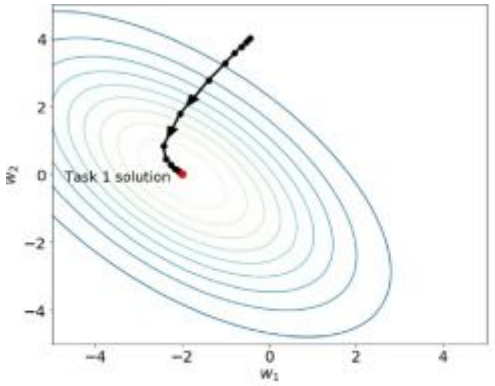


Gradient-based optimization and tug-of-war dynamics

- 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.

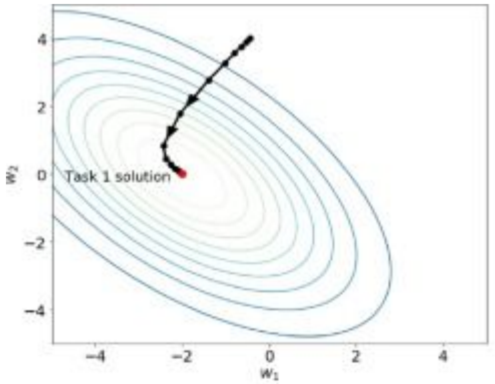


Gradient-based optimization and tug-of-war dynamics

- 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.

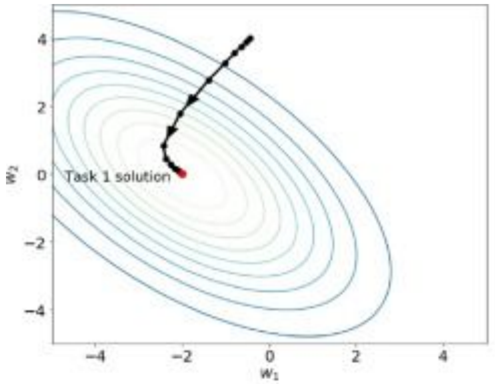
➡ Each training sample produces a gradient for each parameter in the network that votes to make the parameter bigger or smaller.

- 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

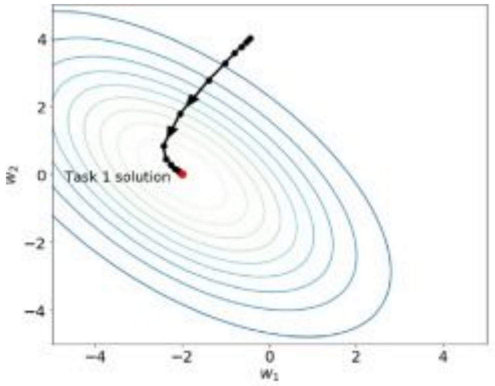
- Continual Learning is a huge challenge for deep learning models because of **gradient-based optimization**.
- Gradient-based learning is effective and cheap, the de rigeur method for training neural networks for close to 4 decades.
- However, a close look at the learning dynamics reveals a problem.
- 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.

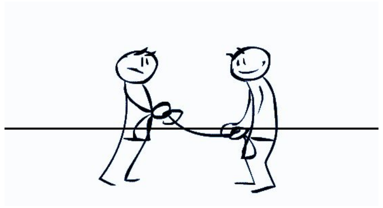


Gradient-based optimization and tug-of-war dynamics

- 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

➡ Why is this important?

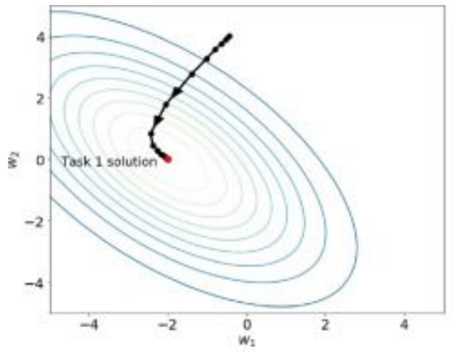
- The tug-of-war dynamics means that data must be **i.i.d. – independent and identically distributed –** for parameters to reach an equilibrium and learning to converge.
- If they are not i.i.d., for instance in continual learning settings, then catastrophic forgetting results: the unopposed gradients of Task 2 cause the parameters to rapidly change.
- Thus **all tasks** must be present in expectation for learning to progress. This is very inefficient: in an N-way tug-of-war, each parameter will still only change in one direction.

Gradient-based optimization and tug-of-war dynamics

- Why is this important?

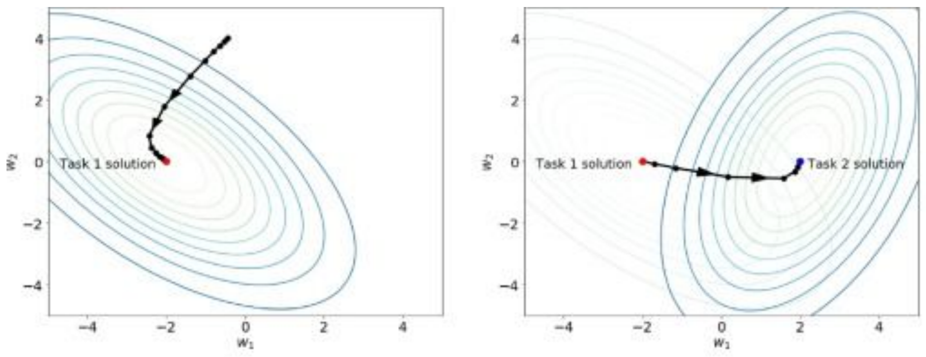
➡ The tug-of-war dynamics means that data must be **i.i.d. – independent and identically distributed –** for parameters to reach an equilibrium and learning to converge.

- If they are not i.i.d., for instance in continual learning settings, then catastrophic forgetting results: the unopposed gradients of Task 2 cause the parameters to rapidly change.
- Thus **all tasks** must be present in expectation for learning to progress. This is very inefficient: in an N-way tug-of-war, each parameter will still only change in one direction.



Gradient-based optimization and tug-of-war dynamics

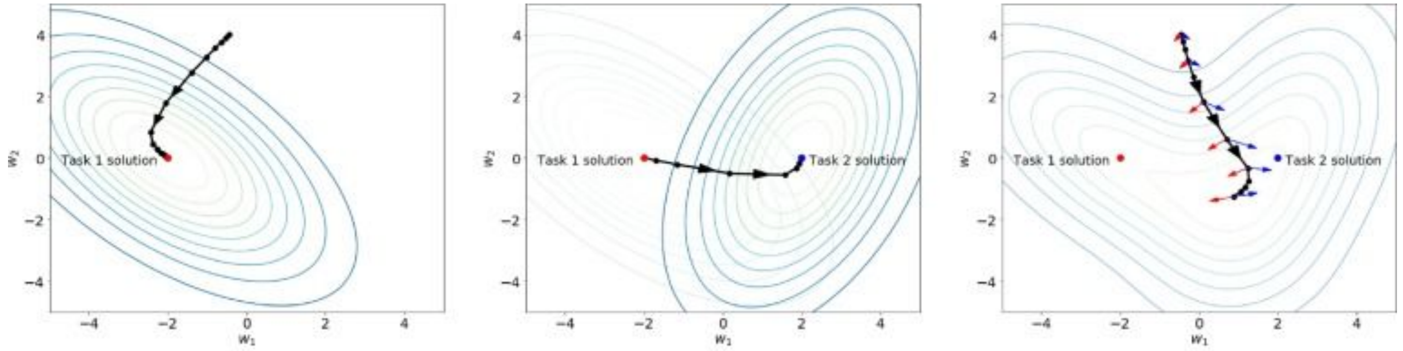
- Why is this important?
- The tug-of-war dynamics means that data must be **i.i.d. – independent and identically distributed –** for parameters to reach an equilibrium and learning to converge.
- If they are not i.i.d., for instance in continual learning settings, then catastrophic forgetting results: the unopposed gradients of Task 2 cause the parameters to rapidly change.
- Thus **all tasks** must be present in expectation for learning to progress. This is very inefficient: in an N-way tug-of-war, each parameter will still only change in one direction.



Gradient-based optimization and tug-of-war dynamics

- Why is this important?
- The tug-of-war dynamics means that data must be **i.i.d. – independent and identically distributed –** for parameters to reach an equilibrium and learning to converge.
- If they are not i.i.d., for instance in continual learning settings, then catastrophic forgetting results: the unopposed gradients of Task 2 cause the parameters to rapidly change.

Thus **all tasks** must be present in expectation for learning to progress. This is very inefficient: in an N-way tug-of-war, each parameter will still only change in one direction.



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.

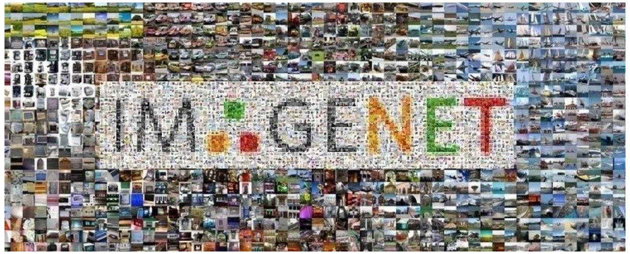
1. Devansh A. et al. A closer look at memorization in deep networks, ICML 2017;
2. Chang H.-S. et al., Active bias: training more accurate neural networks by emphasizing high variance samples, NeurIPS 2017
3. Raposo D. et al. Discovering objects and their relations from entangled scene representations. arXiv 2017

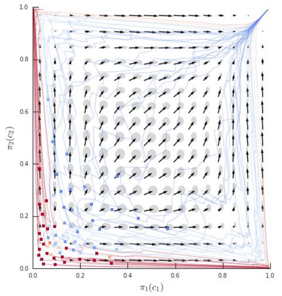
Gradient-based optimization and tug-of-war dynamics

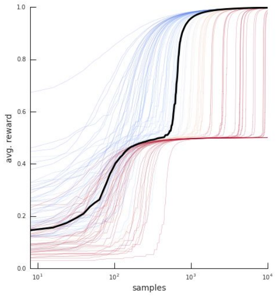
How efficient is gradient-based optimization for stationary learning? Are all parts of the dataset learned at the same speed? The answer is No.

- 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[3,4].







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

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[3,4].

➡ 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.

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

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[3,4].
- 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

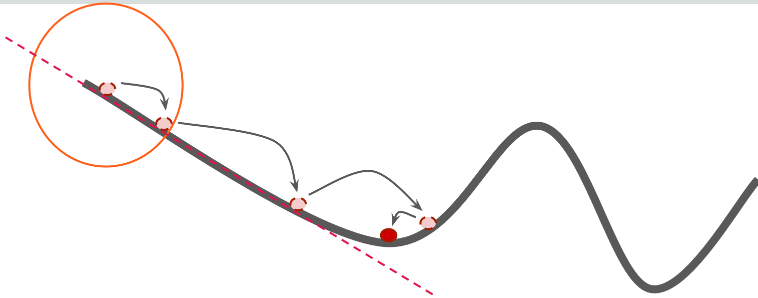
4

Maintaining plasticity

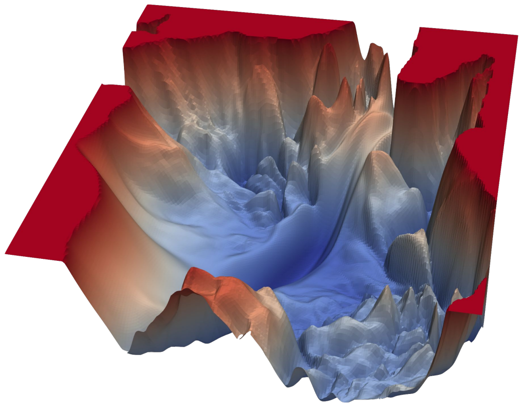
Learning dynamics: neural networks may lose plasticity at steady-state.

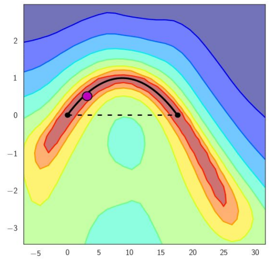
Learning for neural networks seems to be well behaved ...

- Gradient based learning is local, hence susceptible to converge to ***bad local minima***
- This does not happen in practice, furthermore solutions found by learning seem to generalise
- Leaving the illusion of a robust black box machinery that is able to extract information from data

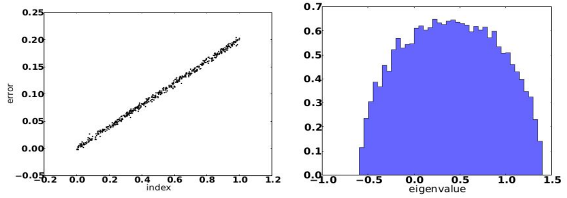


Misbehaved learning expected from nonlinear system





▸ Statistical physics (on random gaussian fields) [Bray and Dean, 2007, Fyodorov and Williams, 2007]



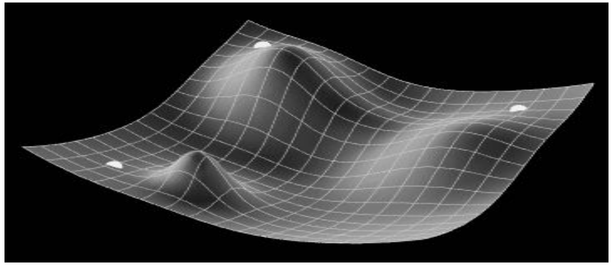
… but signs are that this is dependent on initial conditions and assumptions of the training regime

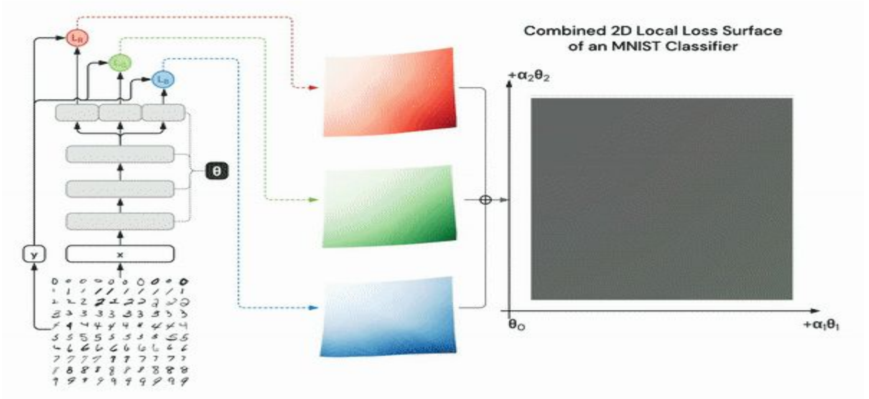
- One big step to get neural networks to train was *better initialisation, larger models* and better activation functions and architectures.

$$\pm\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}$$

- Bad local minima do exist (e.g. Swirzcz et al. 2017) as well as strangely shaped loss surfaces. (e.g. Czarnecki et al. 2019)

- Initialisation is vital to convergence, and to enable the system to learn.
  (e.g. De & Smith 2020)

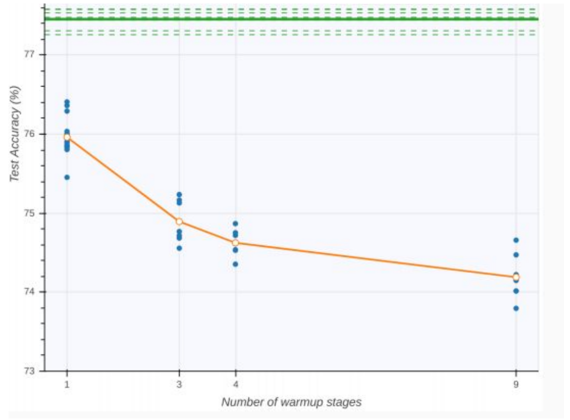


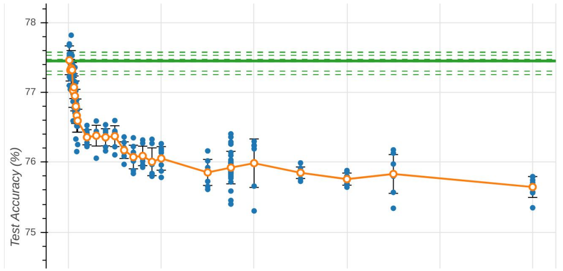


Learning dynamics: neural networks may *lose plasticity* at steady-state.

What happens in a lifelong learning system?

- One can not control initialisation anymore
- Early signs (e.g. Ash et al 2020) suggest that this can bias the learning process towards memorisation rather than learning
- In an extreme case, it can harm optimisation, where the system does not suffer from catastrophic forgetting, but rather inability to learn





4 Continual Learning Solutions

Looking for solutions

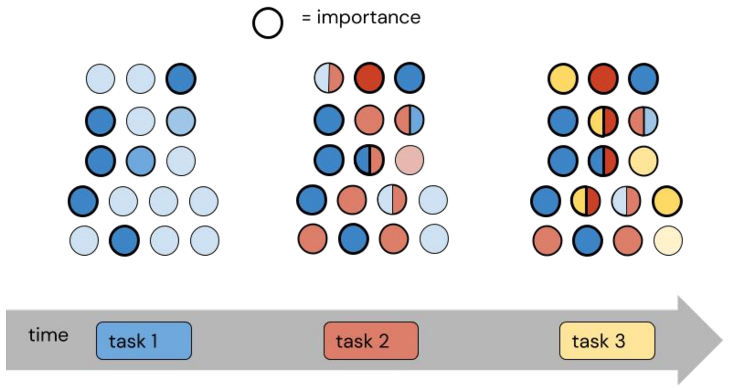
Continual learning can be understood as a set of approaches to stabilize the tug-of-war learning dynamics over a sequence of tasks **without having the previous tasks available.** Interestingly, most CL approaches are biologically inspired.

Four solution spaces:

1. **Gradient-based approaches:** change the gradients or use regularization to directly manage the tug-of-war.
2. **Modular approaches:** Use architecture design or sparsity to allow for specialized task parameters without impacting others.
3. **Memory approaches:** Use memory (replay, episodic, generative) to create proxy samples of previous tasks.
4. **Meta-Learning:** Instead of designing one, **learn** an inductive bias from data to solve continual learning.

Gradient-based solutions

a.k.a., fix the problem at the source

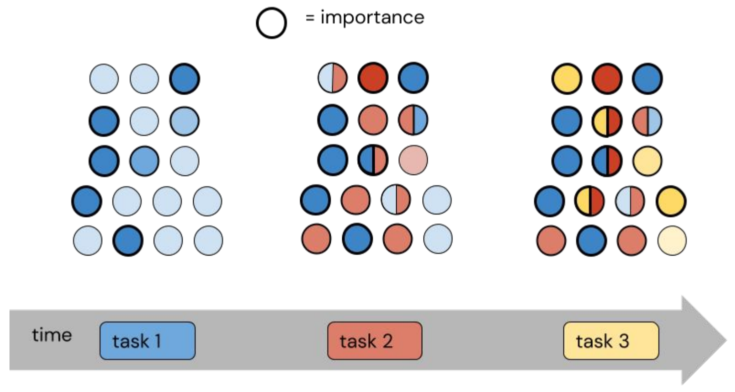


protecting important gradients

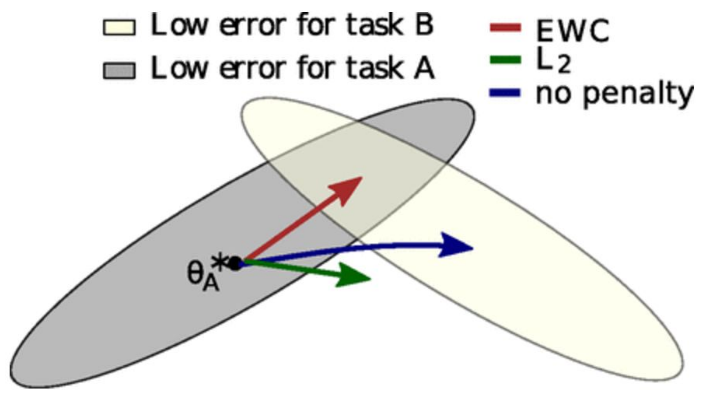
Gradient-based solutions

a.k.a., fix the problem at the source

- Align new gradients with old tasks
  - Gradient Episodic Memory, Lopez-Paz 2017
- 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



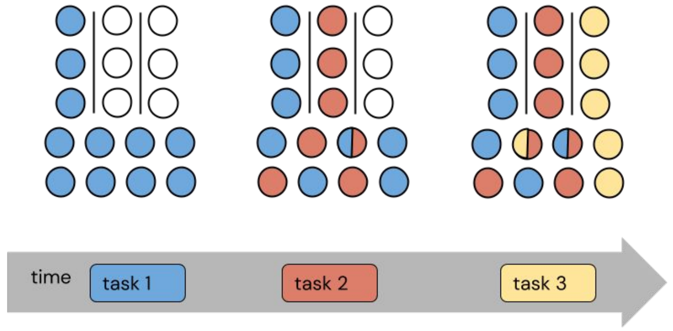
protecting important gradients



EWC: Elastic Weight Consolidation

Modularity and sparsity

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

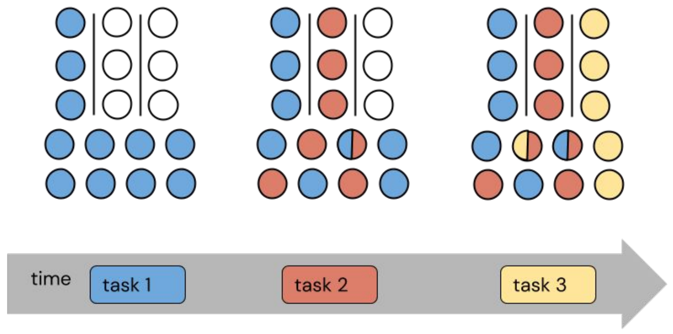


modular architectures

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

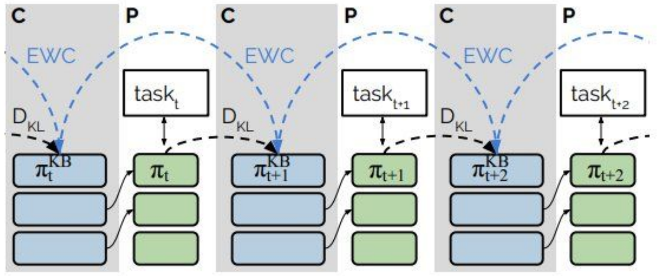
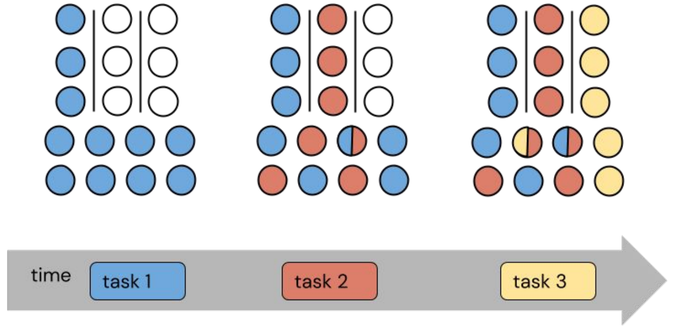


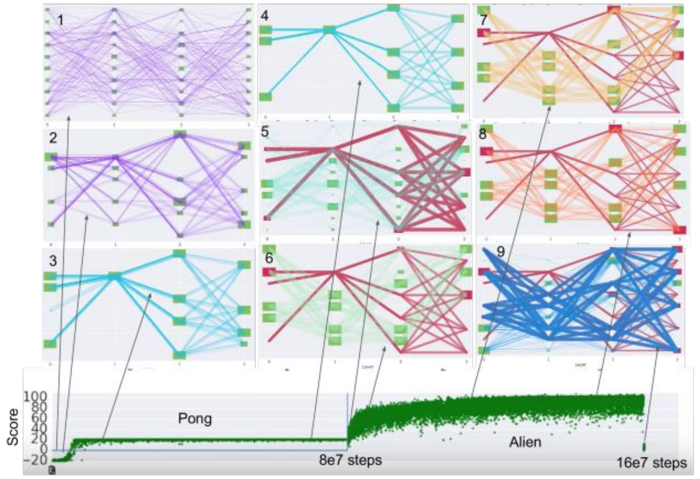
Figure 1. Illustration of the Progress & Compress learning process. In the compress phases (C), the policy learnt most recently by the active column (green) is distilled to the knowledge base (blue) while protecting previous contents with EWC (Elastic Weight Consolidation). In the progress phases (P), new tasks are learnt by the active column while reusing features from the knowledge base via lateral, layerwise connections.

Modularity and sparsity

- Begin with a very large network and partition it by channeling new task gradients to unused parts
  - PathNet, Fernando 2017
  - Conceptors, He 2017
  - Random Path Selection, Rajasegaran 2019
- Use sparsity (in activations or gradients) to limit the extent and impact of learning updates.
  - Selfless Sequential learning, Aljundi 2018

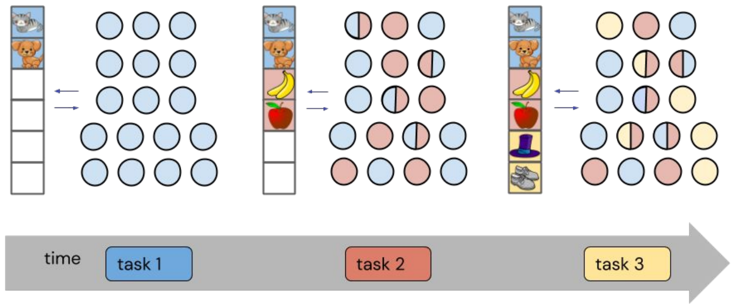


modular architectures



PathNet module reuse

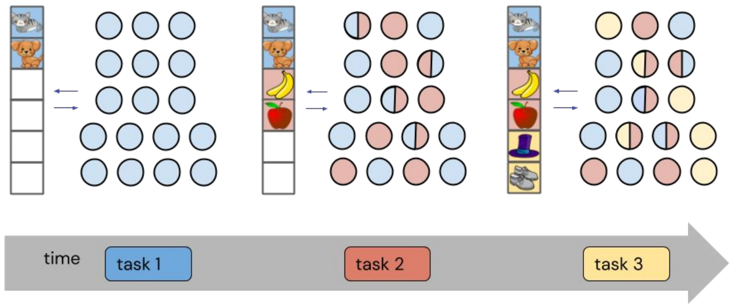
Memory-based solutions for Continual Learning



neural memory

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
  - 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



neural memory

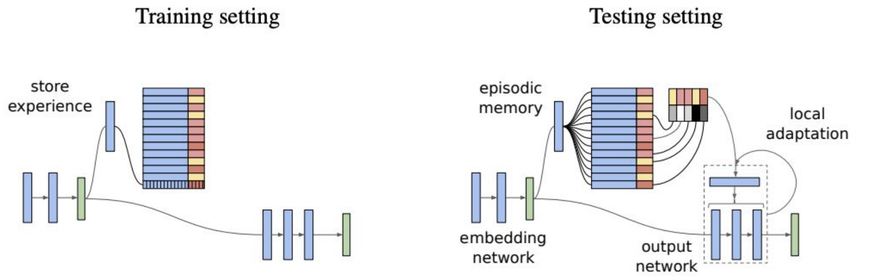


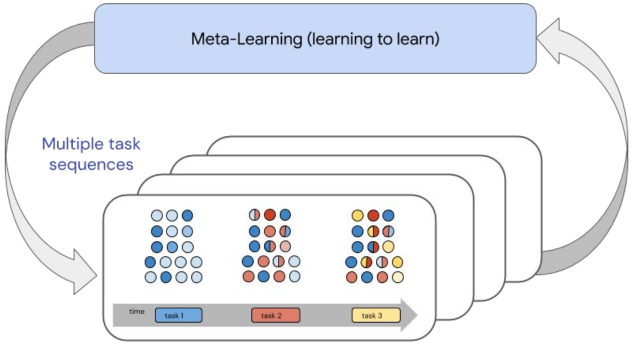
Figure 1: Architecture for the MbPA model. Left: Training usage. The parametric network is used directly and experiences are stored in the memory. Right: Testing setting. The embedding is used to query the episodic memory, the retrieved context is used to adapt the parameters of the output network.

Memory-based Parameter Adaptation (MbPA)

Meta-Learning
Discovering inductive biases for CL

Perhaps rather than hand-engineering the architecture and update rule, we can learn a good inductive bias for continual learning.

- Basic idea:
  - 'inner loop' optimizes for specific tasks,
  - 'outer loop' optimizes over a *set* of tasks

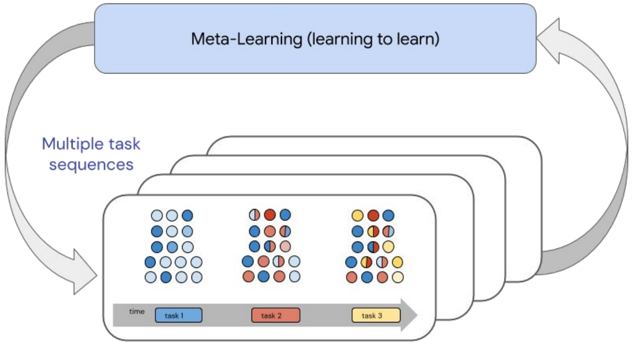


Meta-Learning

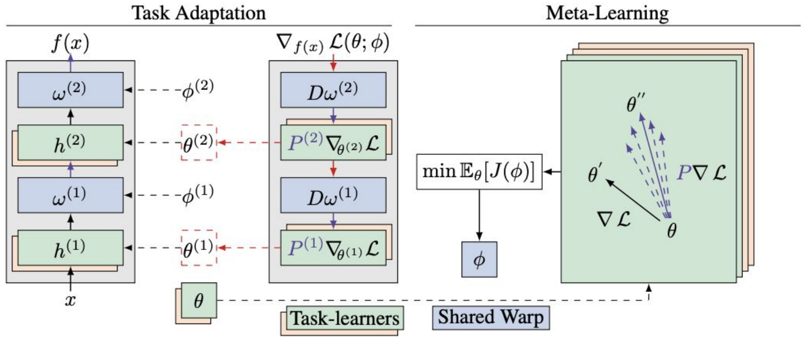
Meta-Learning
Discovering inductive biases for CL

Perhaps rather than hand-engineering the architecture and update rule, we can learn a good inductive bias for continual learning.

- Basic idea:
  - 'inner loop' optimizes for specific tasks,
  - 'outer loop' optimizes over a *set* of tasks
- For CL, the outer loop optimizes for performance or knowledge retention in non-stationary settings
  - Warped Gradient Descent, Flennerhag 2020
  - Deep online learning via meta-learning, Nagabandi 2019
  - Learning to Continually Learn, Beaulieu 2020
- No free lunch however - Meta-learning requires careful design task distribution and is computationally demanding.



Meta-Learning



Warped Gradient Descent

Summary

- Solving continual learning is important: for adaptable applications, for human-level AGI, and for efficient deep learning.
- An overlooked challenge is in the i.i.d. assumption that results from tug-of-war learning dynamics in gradient-based optimization.
- Another under-researched area is the behaviour of neural nets at steady state, rather than at initialisation or convergence.
- There are many possible research directions, from optimization to modularity and sparsity, to memory and meta-learning.

Thank you!