Google DeepMind

Understanding plasticity in neural networks

Clare Lyle
Google DeepMind

Deep Learning Classics & Trends

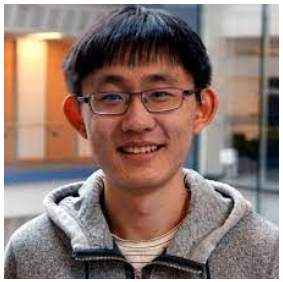
Credits

Talk mostly focuses on results from paper

“Understanding Plasticity in Neural Networks” presented @ ICML this summer

Thanks go to several fantastic collaborators!

Evgenii Nikishin



Zeyu Zheng

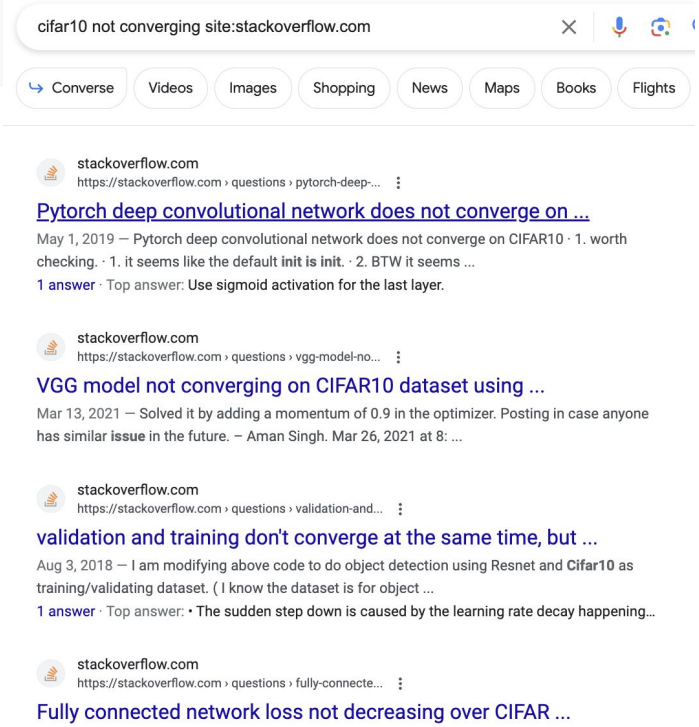
Will Dabney

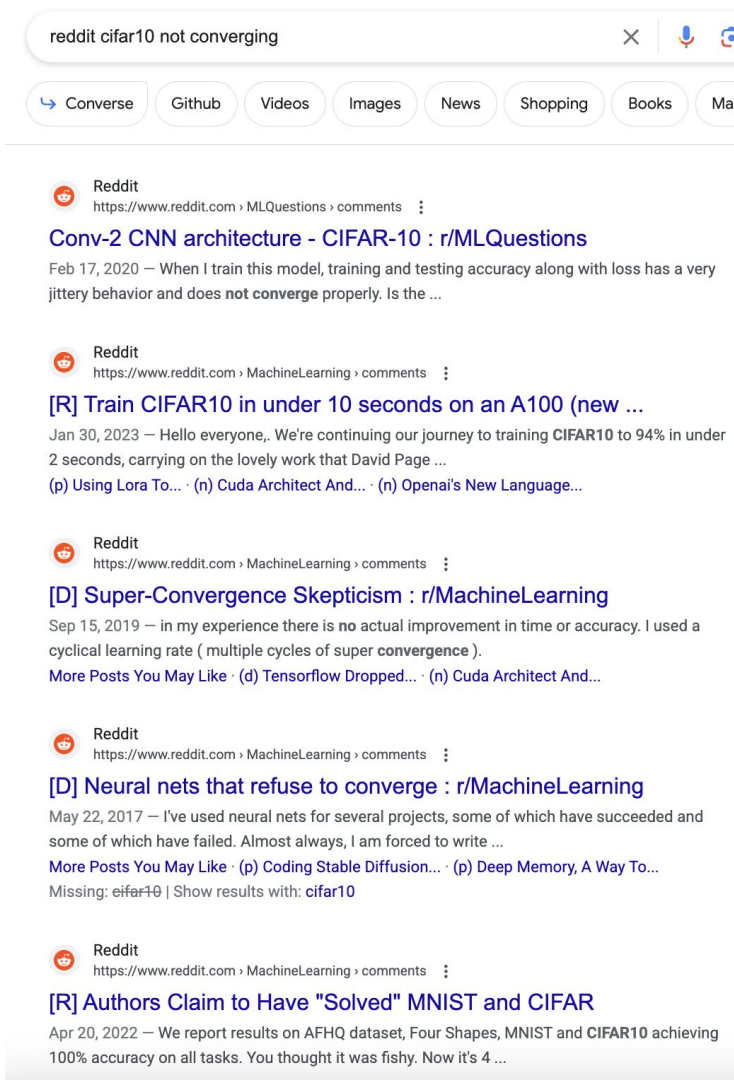
Bernardo Avila Pires

Razvan Pascanu

Stationary learning is easy*

```
# forward + backward + optimize
outputs = net(inputs)
loss = criterion(outputs, labels)
loss.backward()
optimizer.step()
```





Learning under non-stationarity is hard

22 votes

2 answers

17k views

DQN - Q-Loss not converging

The Q-values are **converging**, too (see figure 1). However, for all different settings of hyperparameter the Q-loss is **not converging** (see figure 2). ... Do you have ideas why the Q-loss is **not**...

tensorflow deep-learning reinforcement-learning q-learning

user8861893 **229** asked Oct 31, 2017 at 13:07

Stack Overflow
https://stackoverflow.com › questions › why-is-my-dq...

Why is my DQN (Deep Q Network) not learning?

Jun 29, 2021 — I am training a **DQN** (Deep Q Network) on a CartPole **problem** from OpenAI's gym, but when I start the training, the total score from an episode ...

DQN not working Properly - **Stack Overflow**	Dec 4, 2017
Why is my Deep Q Network **not** learning to play a simple game?	Apr 20, 2020
My neural network does **not** appear to learn **DQN**	Nov 27, 2022
DQN not converging - **Stack Overflow**	Oct 10, 2022

More results from stackoverflow.com

Reddit
https://www.reddit.com › comments › dqn_agent_doe...

DQN agent doesn't learn : r/reinforcementlearning

Sep 17, 2022 — I think the problem could be **your dqn_update_time**. It is defined inside the function and is not global. It is only decremented once and will ...

Reddit
https://www.reddit.com › comments › issues_with_the...

Issues with the training process of DQN

Aug 16, 2023 — Hello everyone, I NEED YOUR HELP ! Currently I´m **working** on a **DQN** agent with: - 42 state features that vary between 0 and 1, - 49 actions

Reddit
https://www.reddit.com › comments › dqn_from_scrat...

DQN from scratch not able to learn any environment, no ...

Nov 13, 2022 — I've attempted to implement a **DQN** from scratch, without importing the neural network from a library, and have attempted to get it to **work** with ...

Reddit
https://www.reddit.com › comments › wfqbgs › why_i...

Why is my DQN cartpole not learning?

Aug 4, 2022 — I coded in a DQN (without any target network). For some reason, **the algorithm fails to learn any meaningful policy**. Here's my code.

Reddit
https://www.reddit.com › comments › dqn_not_learning

DQN not learning : r/reinforcementlearning

Aug 5, 2022 — **DQN not** learning ... I posted a prior version of my code before and got some good suggestions. I've modified my code and would highly appreciate ...

Why is non-stationarity such a challenge?

1. Features learned for early tasks might not be helpful on later tasks – worst case: might bake in spurious correlations

2. Non-stationarity often caused by **bootstrapping:** network uses its own outputs to construct the task, introducing feedback loops that can drive instability

3. As training progresses, network accumulates pathologies that make it harder to optimize

"Plasticity loss"

Progress on early tasks can interfere with later tasks

Features that are helpful in one situation might be detrimental to learning in another

E.g. network learns to depend on spurious correlations which are only revealed to be spurious later in training (Ash & Adams, 2020)

Can be viewed as a form of plasticity loss

UK government paints instructions on road to mitigate poor adaptation of tourists to distribution shift of British traffic patterns.

Many non-stationary objectives are fundamentally unstable

Optimization path for supervised classification

Optimization path for Deep Q-Network

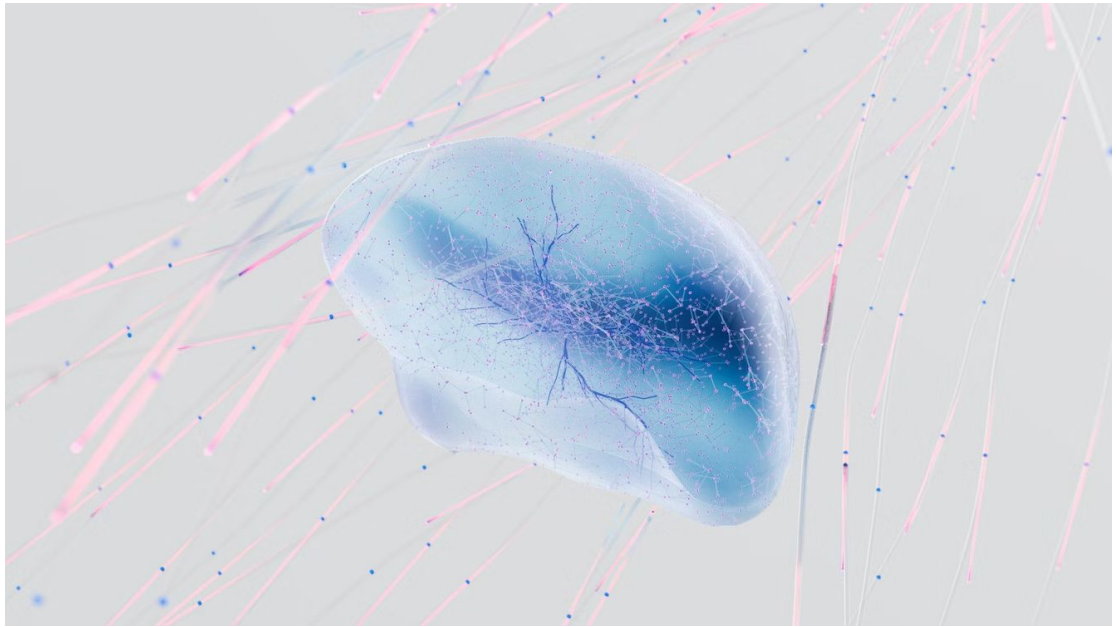
Emergence of pathologies in the network

Freshly initialized parameters are easy to train

BUT no reason to expect optimization to preserve this property

Example: networks with fully-connected layers often accumulate **dead units** as they train, and networks with attention layers often exhibit **logit norm growth** that can interfere with gradient propagation.

The emergence of optimization pathologies is the main form of plasticity loss that we will focus on in this talk.



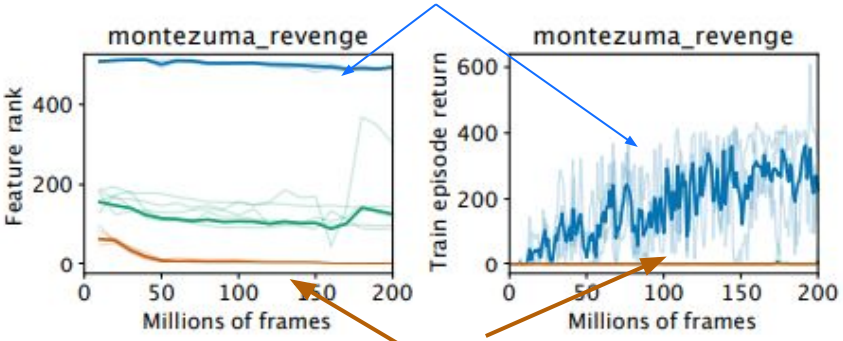
Why focus on plasticity?

Plasticity is necessary for learning

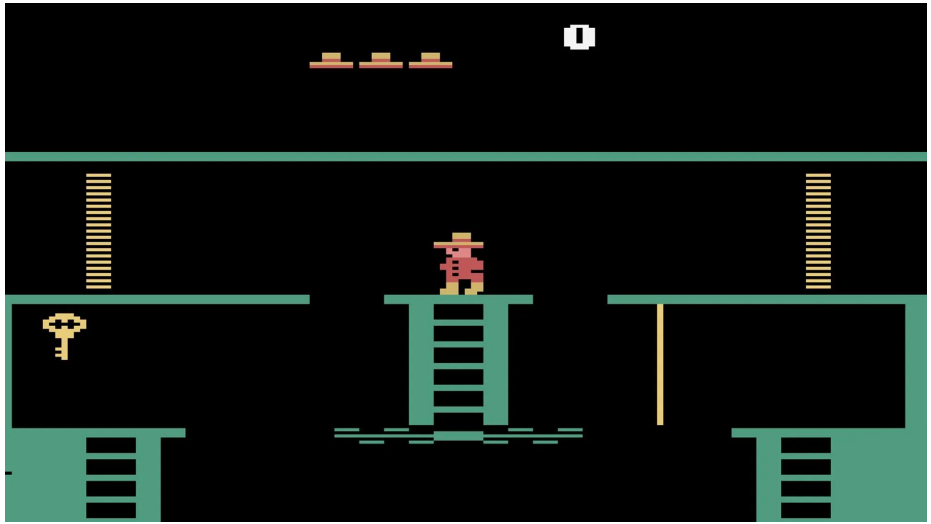
In order to **learn**, need to be able to **change predictions**

Neural networks which have lost plasticity due to e.g. saturating all hidden units of a layer are not able to respond to reward signal quickly enough to improve their policy even if they do randomly stumble on a reward.

Agent trained with regularization to avoid feature collapse, takes advantage of sparse reward



Agent trained without regularization saturates all ReLUs, never recovers



Loss of plasticity is pervasive

$$Z_t(\phi) = \int_0^{v_r} e^{j2\pi tv}\phi(t,v)dv$$

$$Z_f(\phi) = \int_0^{\tau_r} e^{-j2\pi tv}\phi(\tau,f)d\tau$$

???

Talk outline

1. Define and characterize plasticity in neural networks
2. Dig into the mechanisms of plasticity loss
3. Compare some (partial) solutions
4. Discuss open questions

Google DeepMind

1 Characterizing plasticity in neural networks

Formalizing plasticity

“

The less a science has advanced, the more its terminology tends to rest on an uncritical assumption of mutual understanding.

Quine

Plasticity

Noun

1. The state of being plastic.
2. The capacity for continuous alteration of the neural pathways and synapses of the living brain and nervous system in response to experience or injury.
3. The ability of a learning system to adapt to changes in its environment or objective function.

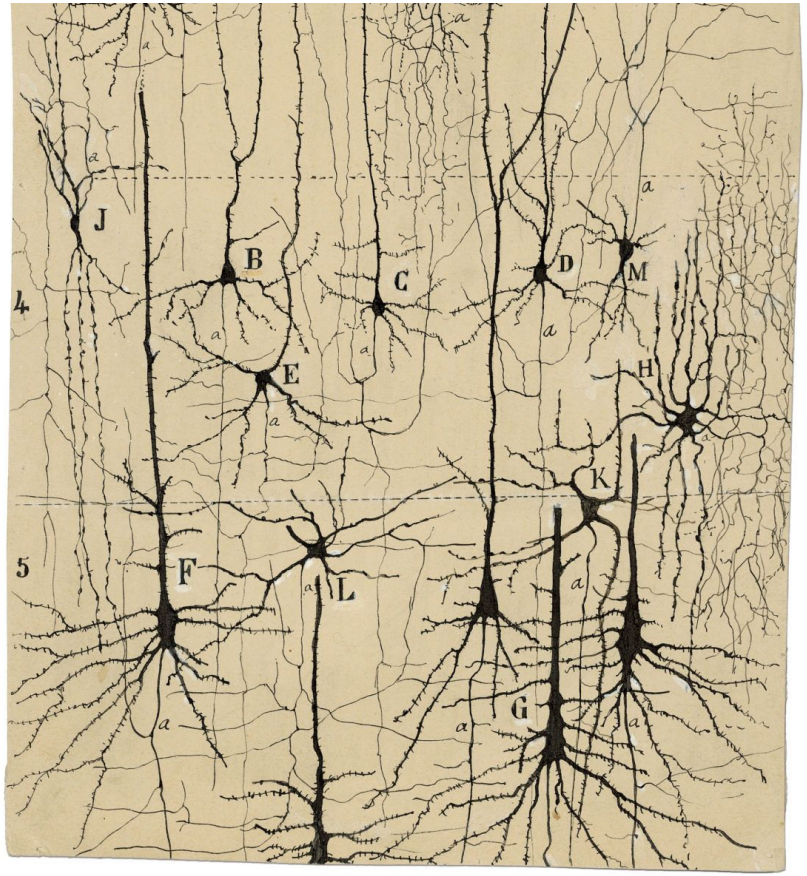
Neural plasticity in the wild

Long-term research problem in neuroscience: how does the brain learn?

Depends on formation of *new connections* between neurons

Would imply that artificial neural networks with fixed architectures never gain or lose plasticity

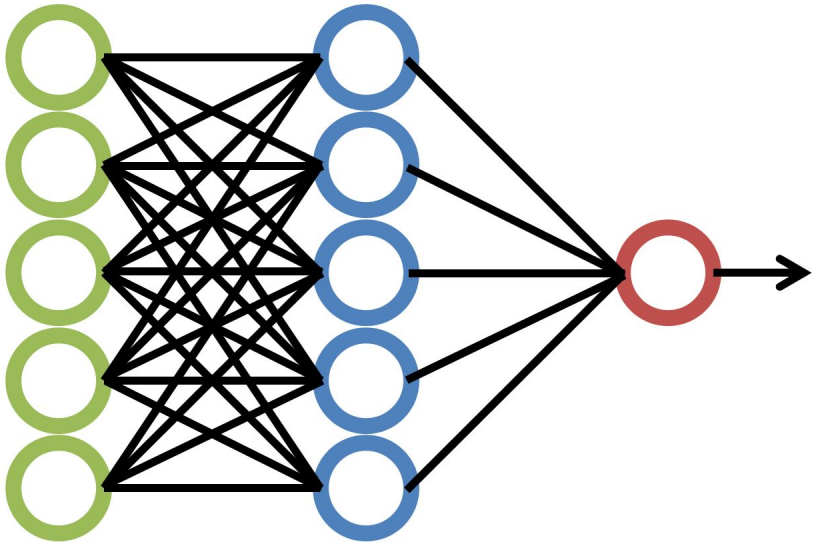
So we want a different definition



Defining plasticity in artificial neural networks

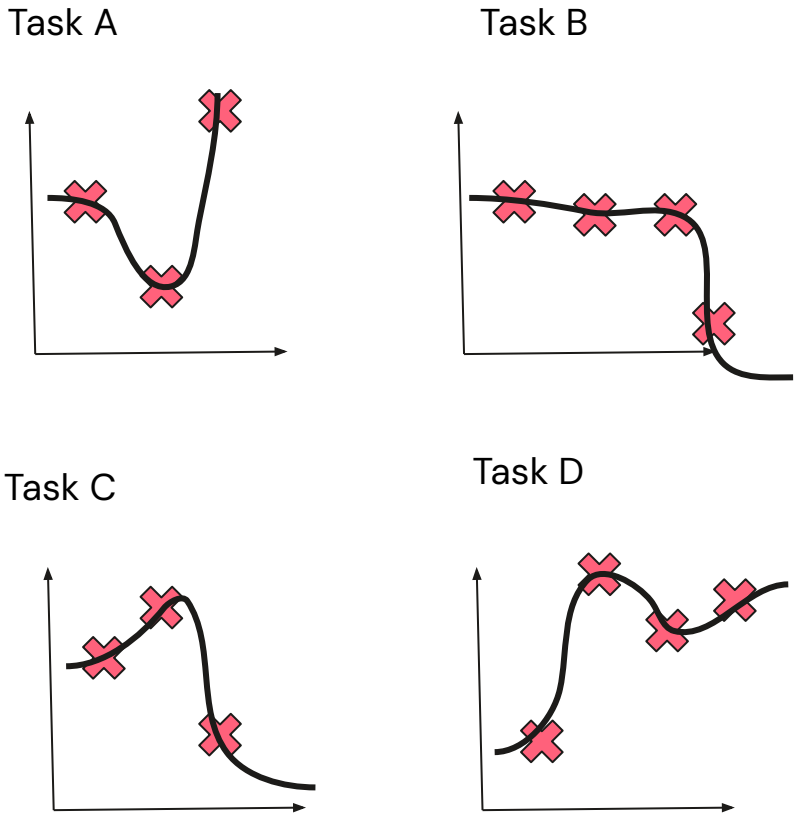
Want a quantity which:

- Depends on the optimization algorithm, architecture, and parameters we start from
- Is higher when the network can quickly learn new things, and lower when it cannot
- Looks at changes in the outputs of the network, not its internal structure
- Depends on the class of new tasks we want the network to be able to learn



Measuring plasticity

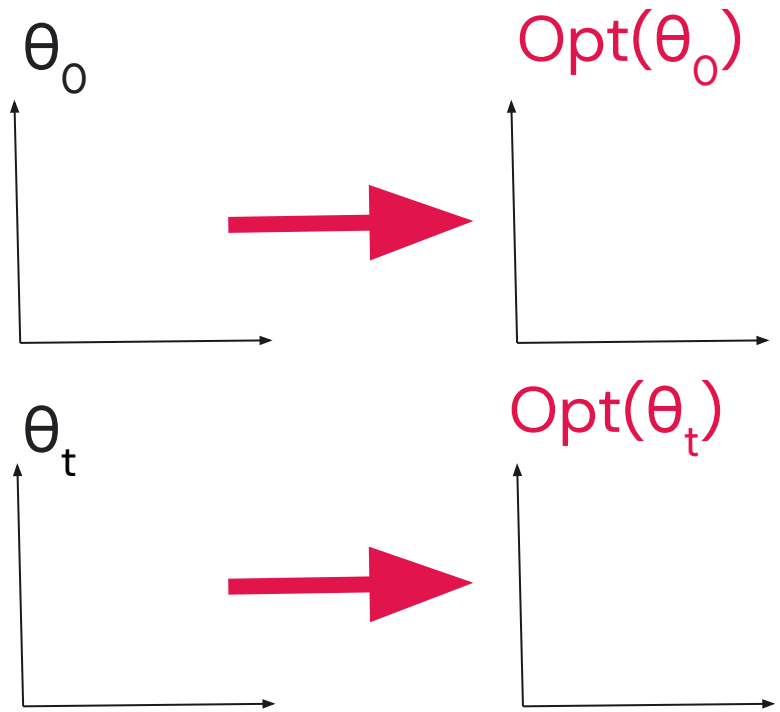
Idea: measure plasticity by testing the network's ability to quickly solve **new tasks.**



Measuring plasticity

Idea: measure plasticity by testing the network's ability to quickly solve new tasks.

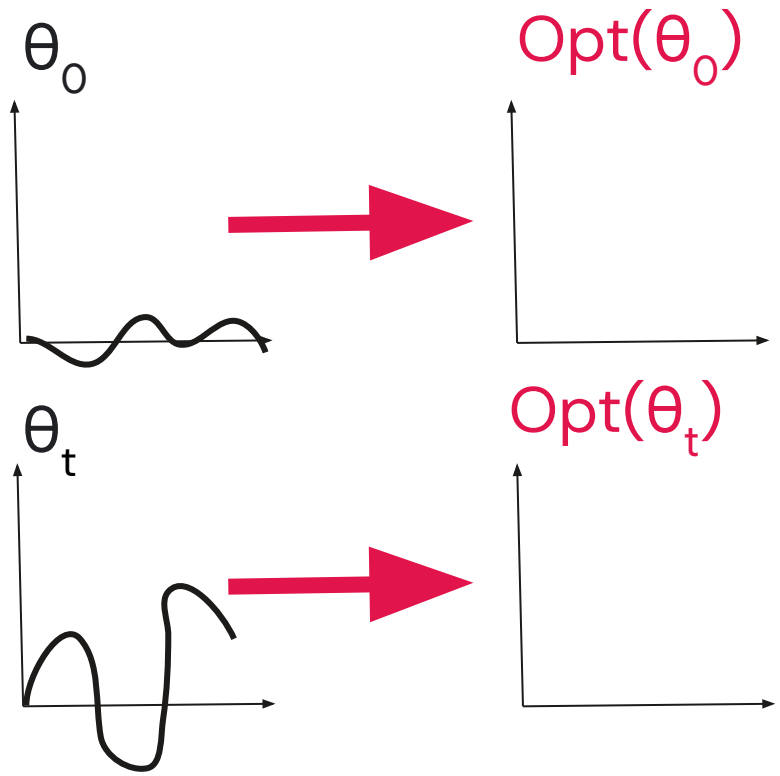
- Use a fixed optimizer and training budget to get optimization protocol *O*



Measuring plasticity

Idea: measure plasticity by testing the network's ability to quickly solve new tasks.

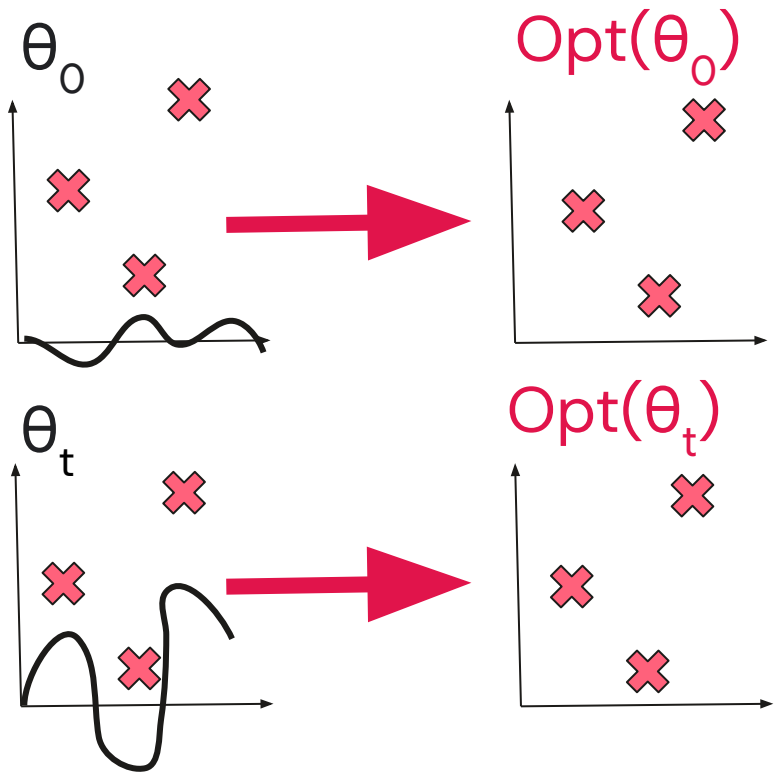
- Use a fixed optimizer and training budget to get optimization protocol *O*
- Start from the parameters whose plasticity we want to evaluate



Measuring plasticity

Idea: measure plasticity by testing the network's ability to quickly solve new tasks.

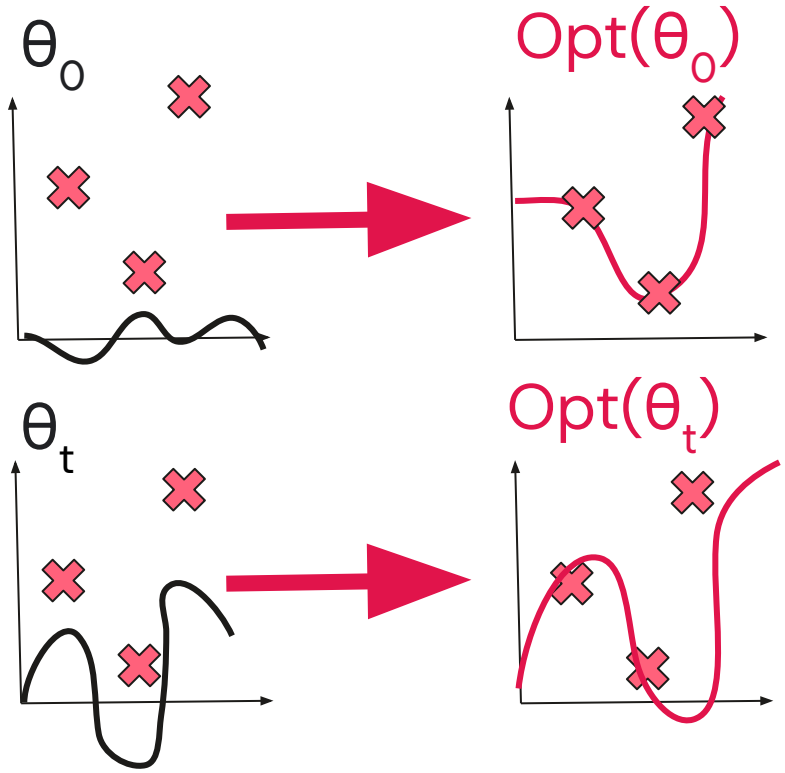
- Use a fixed optimizer and training budget to get optimization protocol *O*
- Start from the parameters whose plasticity we want to evaluate
- Randomly sample a "probe" learning task from some distribution.



Measuring plasticity

Idea: measure plasticity by testing the network's ability to quickly solve new tasks.

- Use a fixed optimizer and training budget to get optimization protocol *O*
- Start from the parameters whose plasticity we want to evaluate
- Randomly sample a "probe" learning task from some distribution.
- Run the optimization protocol from starting parameters and evaluate loss at end of optimization.



Measuring plasticity

For some probe task g_ω define probe task loss

$$\ell_{f,\mathbf{x}}(\theta) = \mathbb{E}_{x\sim\mathbf{x}}[(f(\theta,\mathbf{x}) - g_\omega(\mathbf{x}))^2]$$

Set b to be some baseline (e.g. the average loss obtained by predicting the mean on the set of probe tasks), and let l ~ L denote sampling a loss from a distribution of probe tasks g_ω. Then plasticity can be defined as:

$$\mathcal{P}(\theta_t) = b - \mathbb{E}_{\ell\sim\mathcal{L}}[\ell(\theta_t^*)] \text{ where } \theta_t^* = \mathcal{O}(\theta_t, \ell)$$

Desiderata checklist

- Depends on the **optimization algorithm** ✅
- Is higher when the network can quickly learn new things, and lower when it cannot
- Looks at changes in the outputs of the network, not its internal structure
- Depends on the class of new tasks we want the network to be able to learn

$$\mathcal{P}(\theta_t) = b - \mathbb{E}_{\ell \sim \mathcal{L}}[\ell(\theta_t^*)] \text{ where } \theta_t^* = \mathcal{O}(\theta_t, \ell)$$

Desiderata checklist

- Depends on the optimization algorithm ✅
- Is **higher when the network can quickly learn new things**, and lower when it cannot ✅
- Looks at changes in the outputs of the network, not its internal structure
- Depends on the class of new tasks we want the network to be able to learn

$$\mathcal{P}(\theta_t) = b - \mathbb{E}_{\ell \sim \mathcal{L}}[\ell(\theta_t^*)] \text{ where } \theta_t^* = \mathcal{O}(\theta_t, \ell)$$

Desiderata checklist

- Depends on the optimization algorithm ✅
- Is higher when the network can quickly learn new things, and lower when it cannot ✅
- Looks at changes in the **outputs of the network**, not its internal structure ✅
- Depends on the class of new tasks we want the network to be able to learn

$$\mathcal{P}(\theta_t) = b - \mathbb{E}_{\ell \sim \mathcal{L}}[\ell(\theta_t^*)] \text{ where } \theta_t^* = \mathcal{O}(\theta_t, \ell)$$

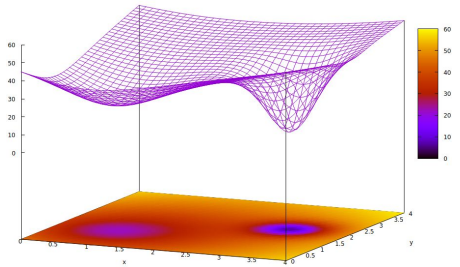
Desiderata checklist

- Depends on the optimization algorithm ✅
- Is higher when the network can quickly learn new things, and lower when it cannot ✅
- Looks at changes in the **outputs of the network**, not its internal structure ✅
- **Depends on the class of new tasks** we want the network to be able to learn ✅

$$\mathcal{P}(\theta_t) = b - \mathbb{E}_{\ell \sim \mathcal{L}}[\ell(\theta_t^*)] \text{ where } \theta_t^* = \mathcal{O}(\theta_t, \ell)$$

Expected vs empirical risk minimization

$$\ell_{f,\mathbf{X}}(\theta) = \mathbb{E}_{x \sim \mathbf{X}}[(f(\theta, \mathbf{x}) - g_\omega(\mathbf{x}))^2]$$

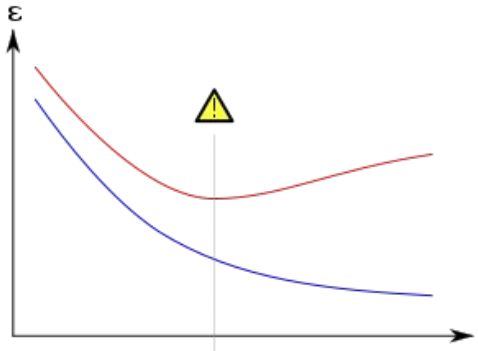


What distribution **X** do we want to draw from?

Want to ensure that the network can minimize the loss on its **training data**.

Also want the network to generalize well to **new inputs**.

This talk will focus on convergence on training data.



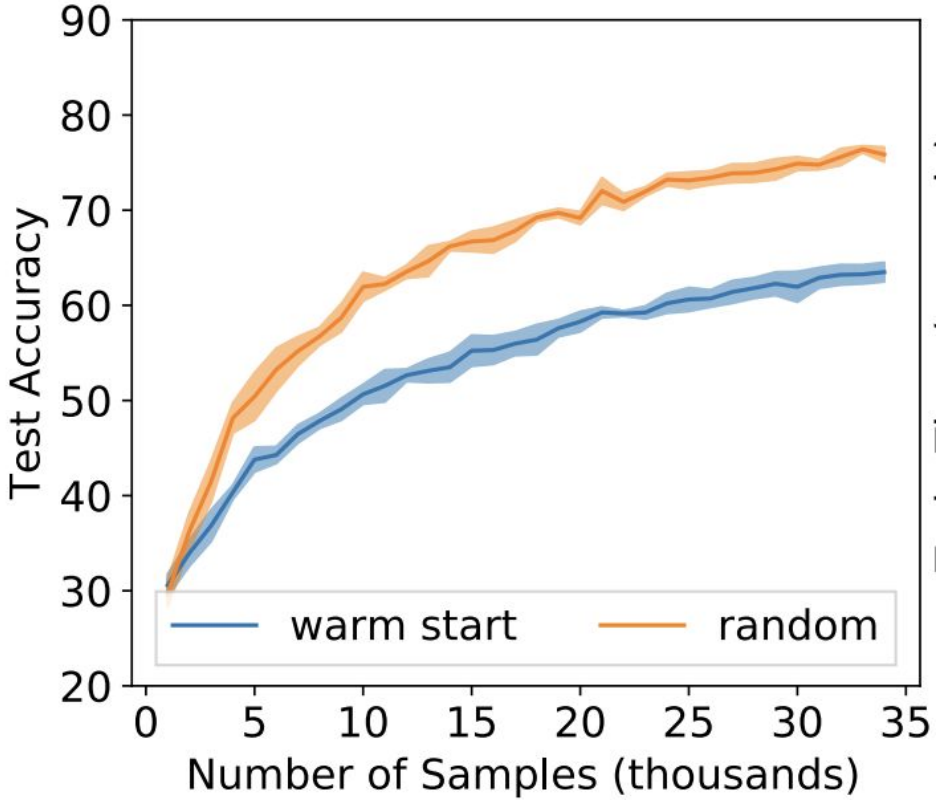
Faces of plasticity loss

Warm starting

Observed in 2019 in the context of "warm-starting" neural network training.

Pre-training on half of CIFAR-10 for several epochs results in worse generalization than training on the whole dataset from the start

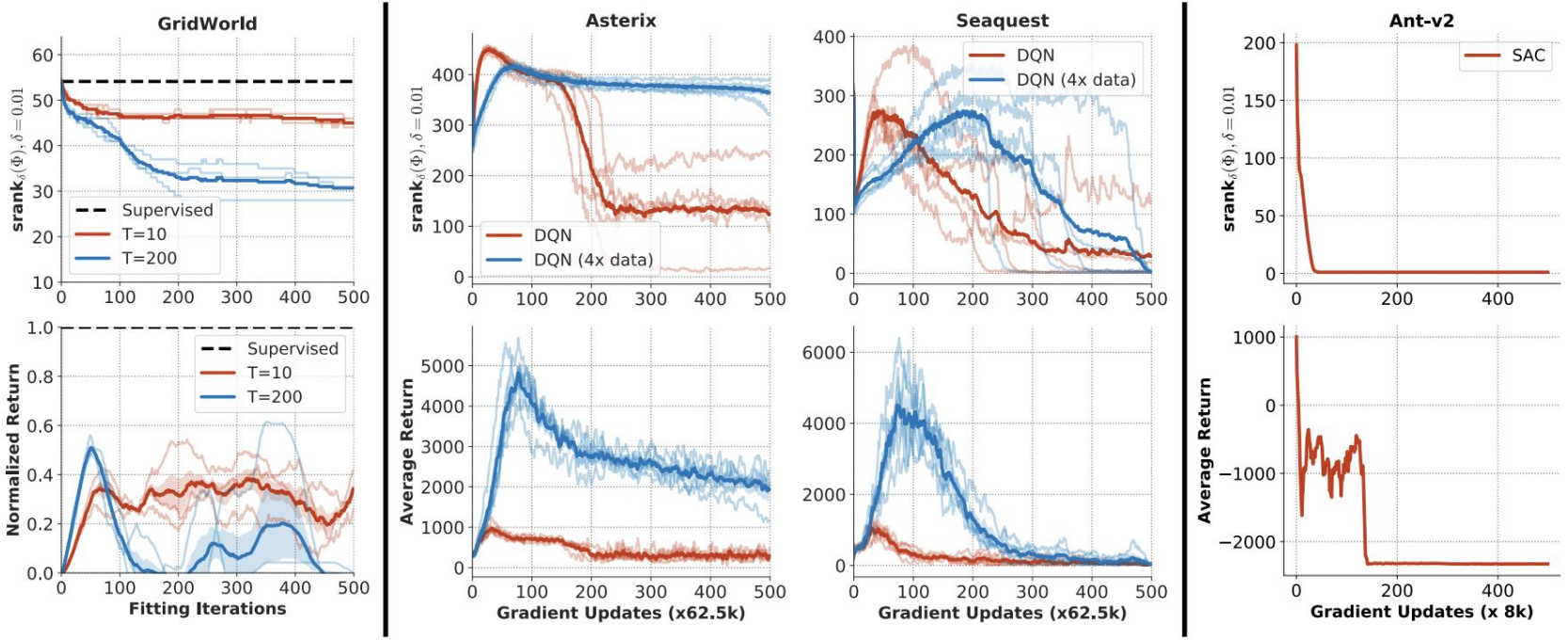
Ash & Adams, 2019



Implicit Under-parameterization

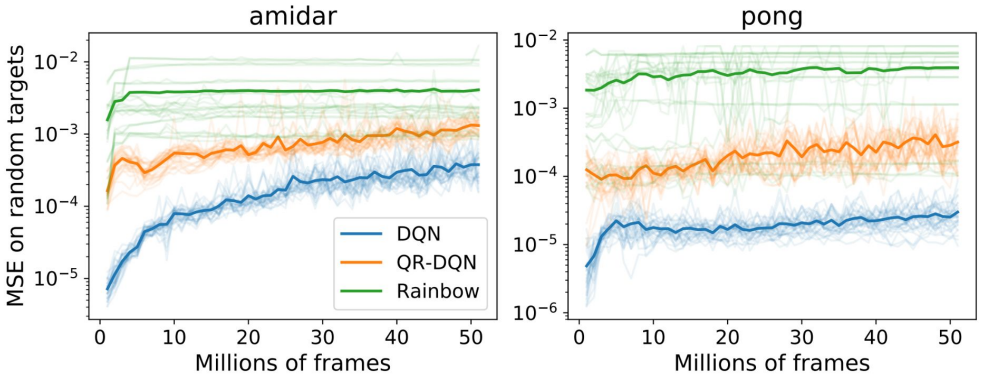
Observation that the effective rank of the feature embeddings tends to decline over time, corresponding to performance and feature collapse.

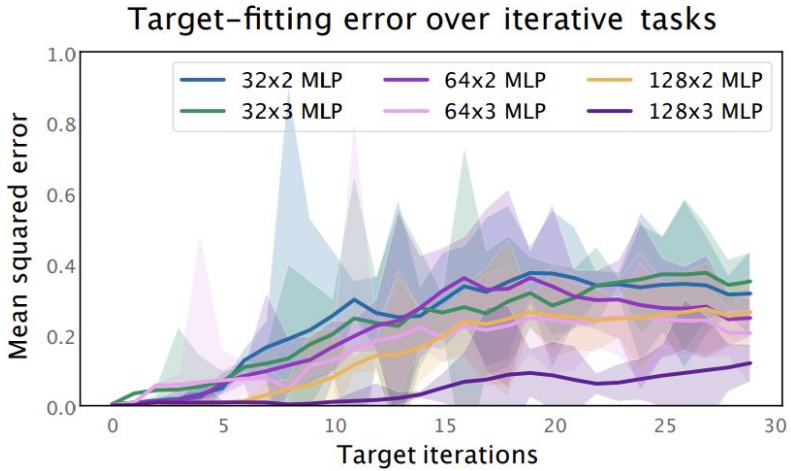
Agarwal et al., 2020



Capacity Loss

Observation that in some cases, training a neural network on a sequence of tasks makes it harder to learn on new tasks – even if the new tasks are drawn from the same distribution as what the network was trained on.





Lyle et al., 2021

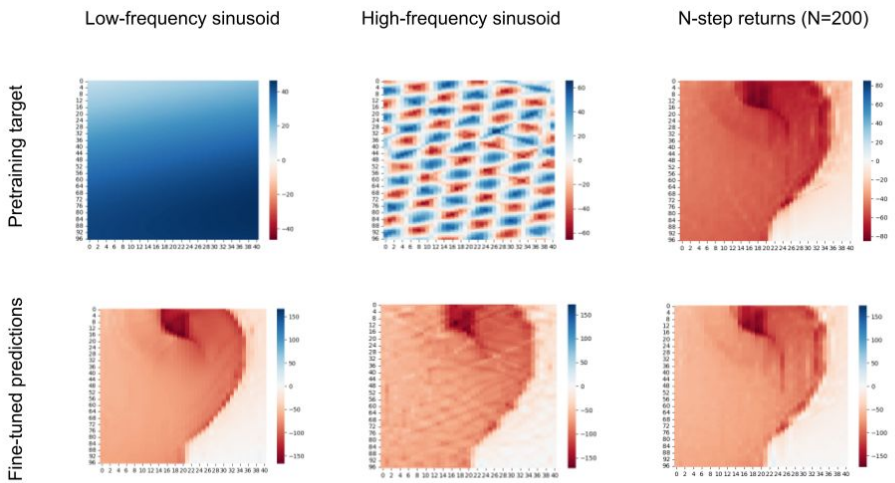
The Primacy Bias

Early training data has outsized impact on generalization and inductive biases

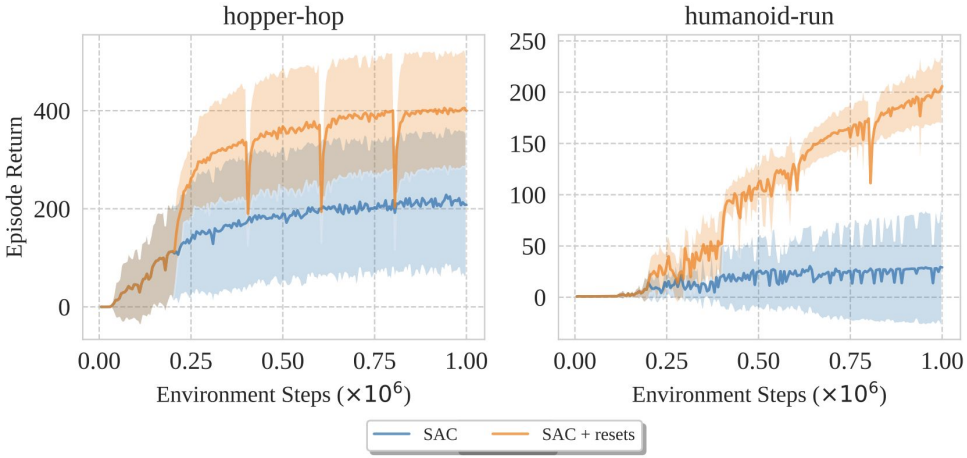
Network resets in deep RL can boost performance

Related to “critical learning periods” in visual system

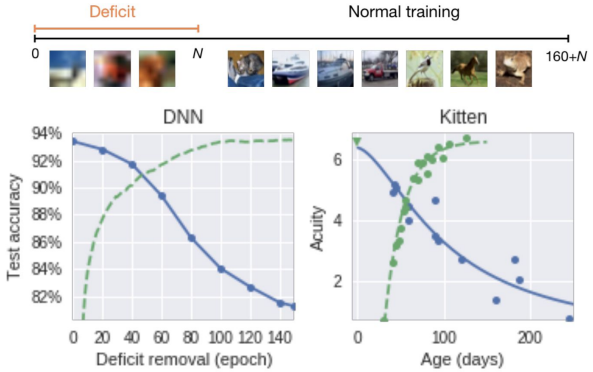
Lyle et al., 2022



Nikishin et al., 2022



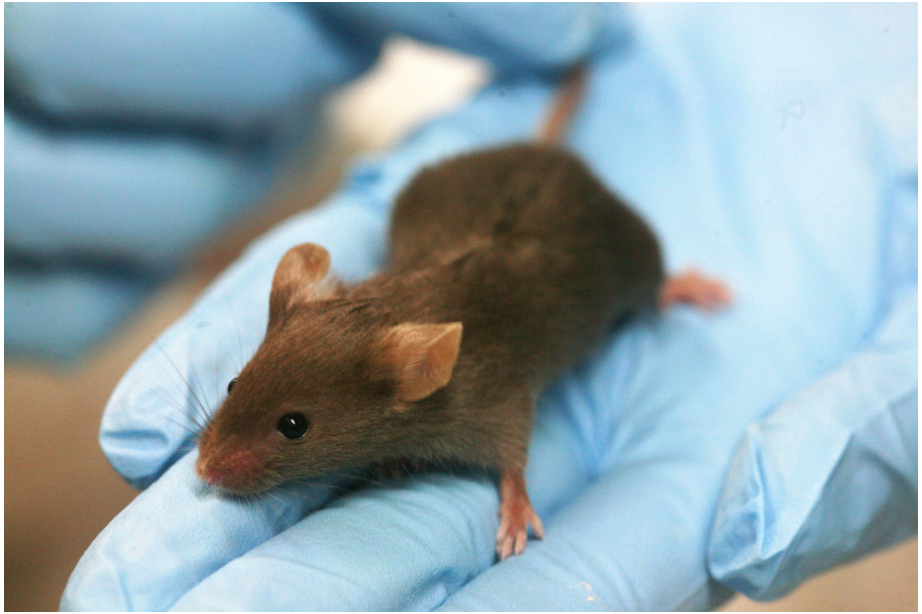
Achille et al., 2017



Google DeepMind

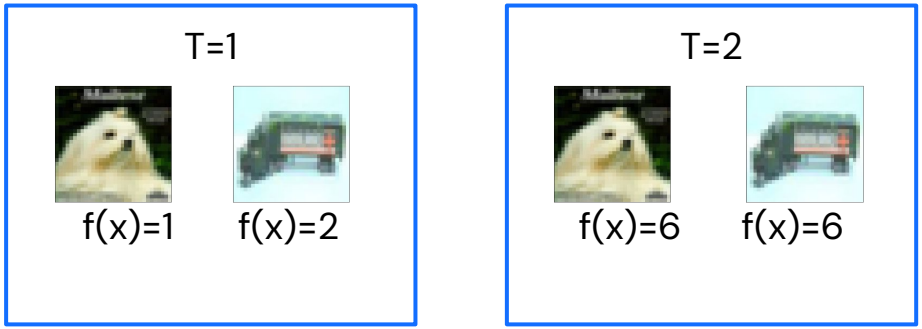
2 Mechanisms of plasticity loss

Model organisms of plasticity loss

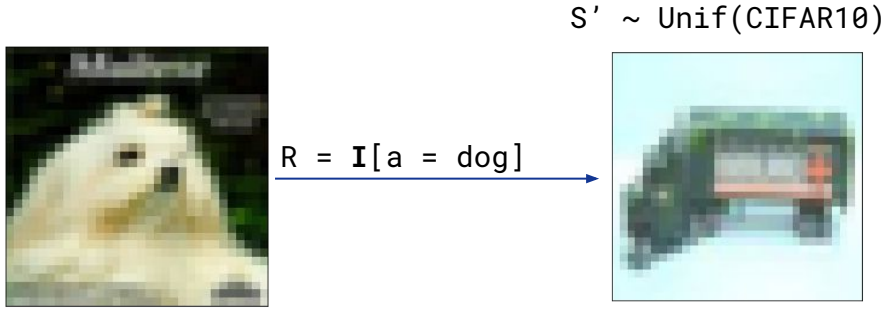


Two models of non-stationary learning

1. (Re-)randomized label memorization
   a. Models sudden, drastic changes in the task

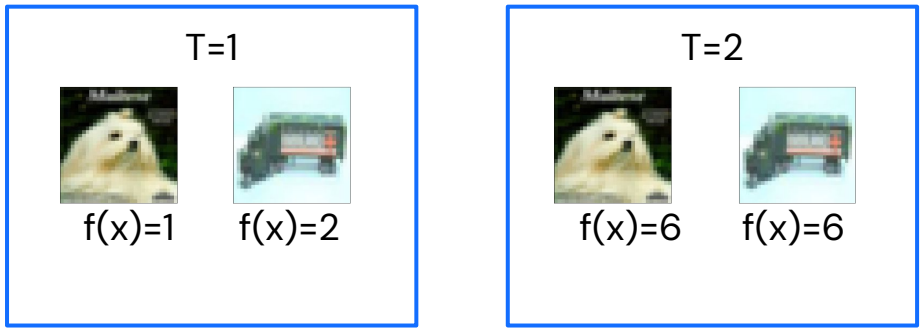


2. Q-learning on an image classification "MDP"
   a. Non-stationarity implicitly induced by target network updates

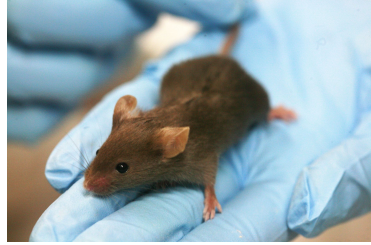


Two models of non-stationary learning

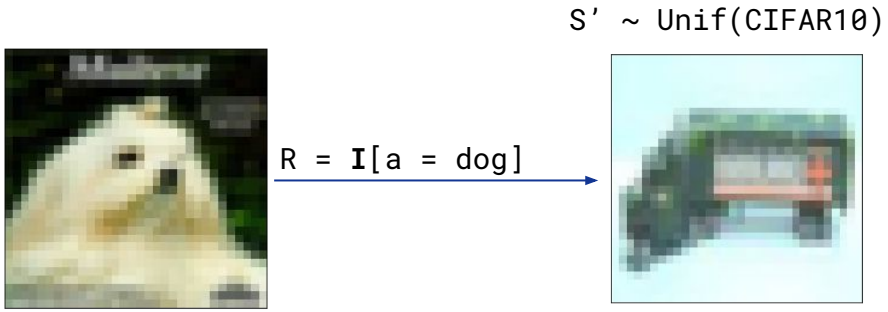
1. (Re-)randomized label memorization
   a. Models sudden, drastic changes in the task
   b. Simulates noisy or hard-to-learn relationships in the real world that network may have to quickly adapt to.
   c. Amenable to both classification and regression losses



Classification as an MDP



2. Q-learning on an image classification “MDP”
    a. Non-stationarity implicitly induced by target network updates
    b. **Very** dense-reward
    c. Can consider variations where e.g. rewards correspond to random labels (so must be memorized), or where rewards are only given for a subset of classes
    d. No exploration confounding – can even collect data using hand-coded optimal policy
    e. More natural form of nonstationarity

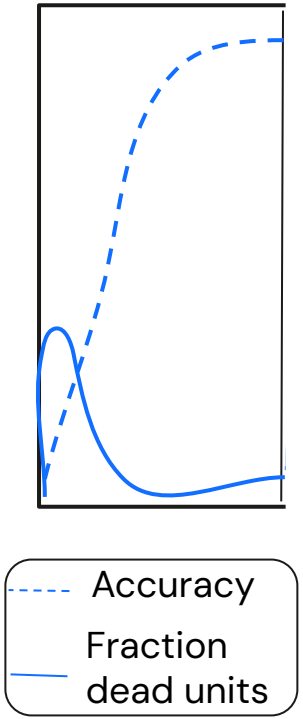


Problem 1: accumulating saturated units

Networks tend to accumulate dead units

Can be hard to undo because no gradients

Have to hope that the preceding layer features eventually accidentally wander back into a regime with positive dot product

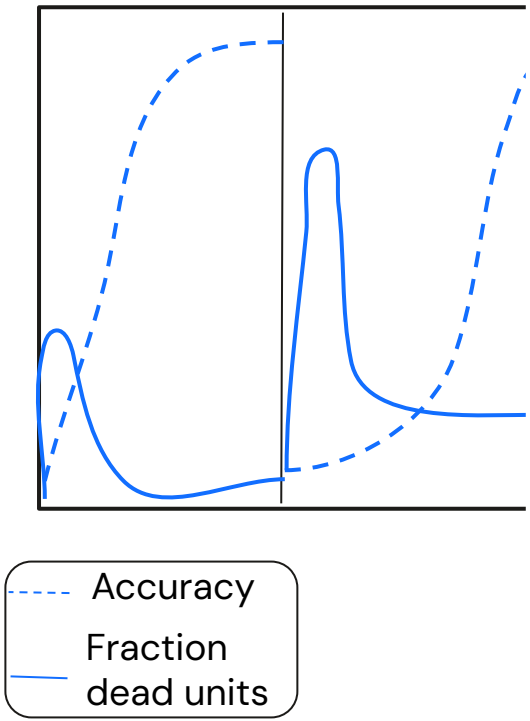


Problem 1: accumulating saturated units

Networks tend to accumulate dead units

Can be hard to undo because no gradients

Have to hope that the preceding layer features eventually accidentally wander back into a regime with positive dot product

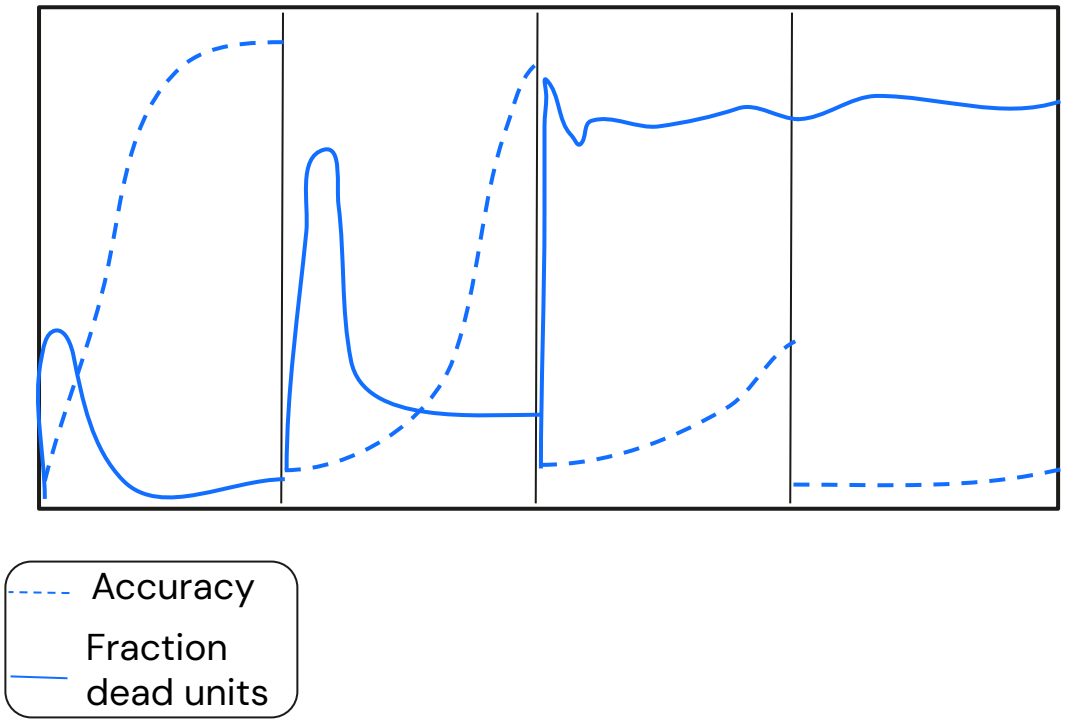


Problem 1: accumulating saturated units

Networks tend to accumulate dead units

Can be hard to undo because no gradients

Have to hope that the preceding layer features eventually accidentally wander back into a regime with positive dot product



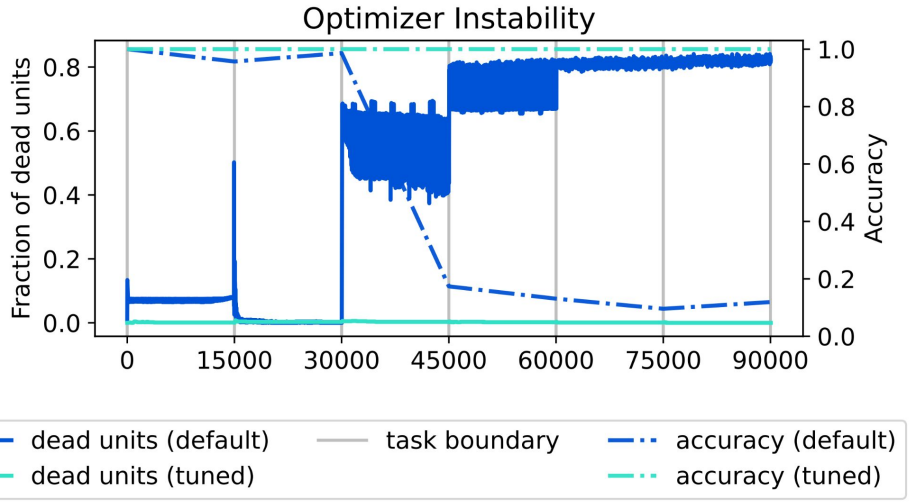
Problem 1: accumulating saturated units

Not a huge problem in single-task setting

BUT a big problem when the correlations between features and targets change.

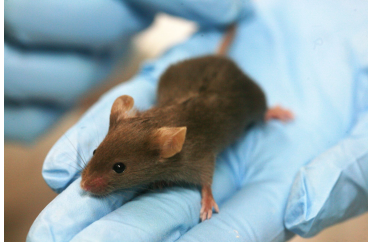
Network can accidentally go overboard and kill off activations

Exacerbated by adaptive optimizer step size

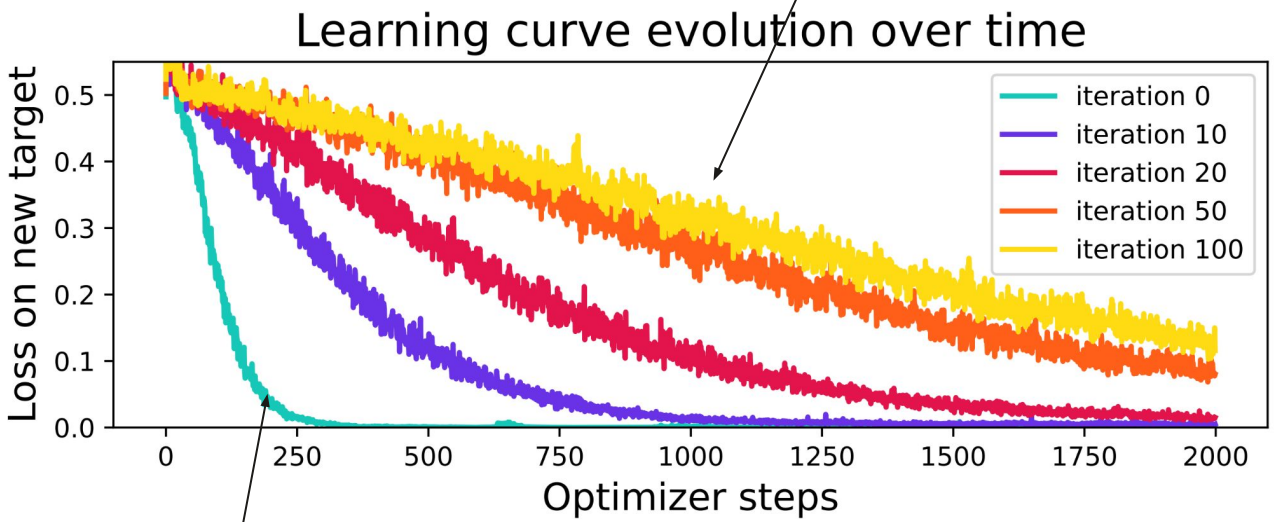


$$u_t = \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \bar{\epsilon} + \epsilon}$$

Problem 2: difficulty navigating loss landscape

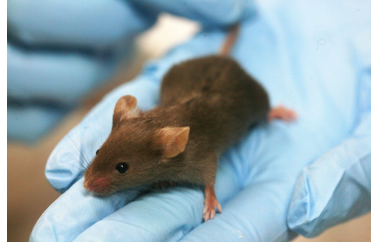


Learning curve on new probe task
after 100 iterations of RL training



Learning curve on new probe task of
random initialization

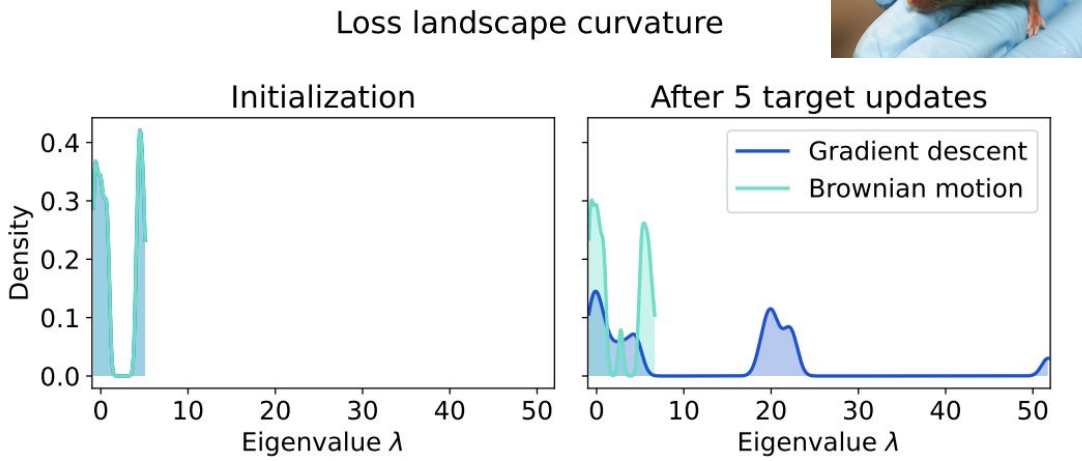
Problem 2: difficulty navigating loss landscape

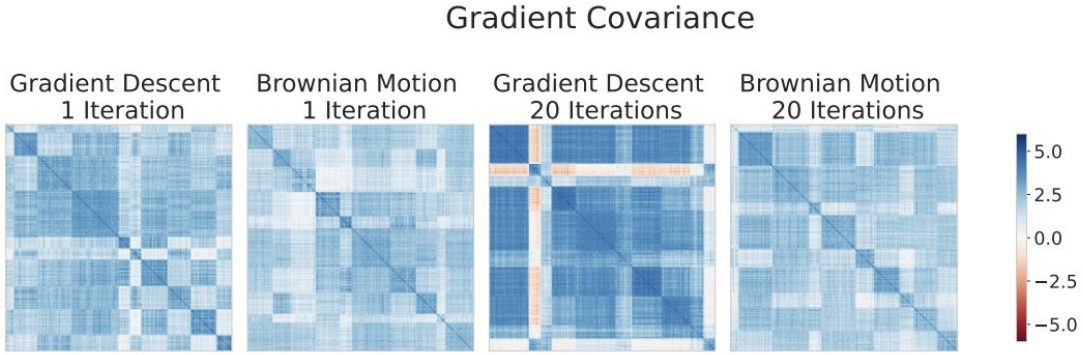


But why is the loss landscape getting harder to navigate?

Partially due to nature of learning problem: regression on large target is hard for neural networks

But seems to also be driven by something more fundamental – see similar issues even in classification tasks





Correlates of plasticity

Are there easy-to-measure properties of a network that correlate with plasticity, and which we can *intervene* on to improve the network's adaptability to new tasks?

A good explanation of plasticity should be **consistent** under different experimental settings.

Correlates of plasticity

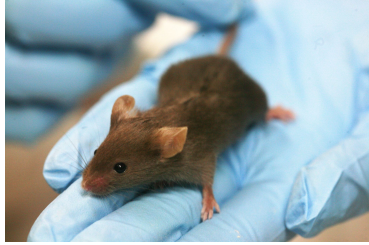
Are there easy-to-measure properties of a network that correlate with plasticity, and which we can *intervene* on to improve the network's adaptability to new tasks?

A good explanation of plasticity should be **consistent** under different experimental settings.

Consider several candidate causal factors in plasticity loss:

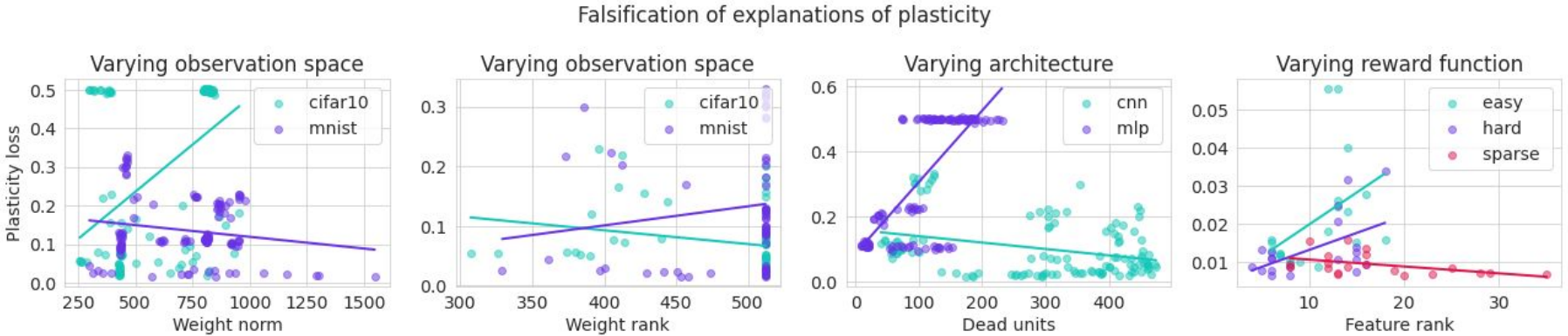
1. Weight norm
2. Weight rank
3. Number of dead units
4. Numerical rank of feature embeddings.

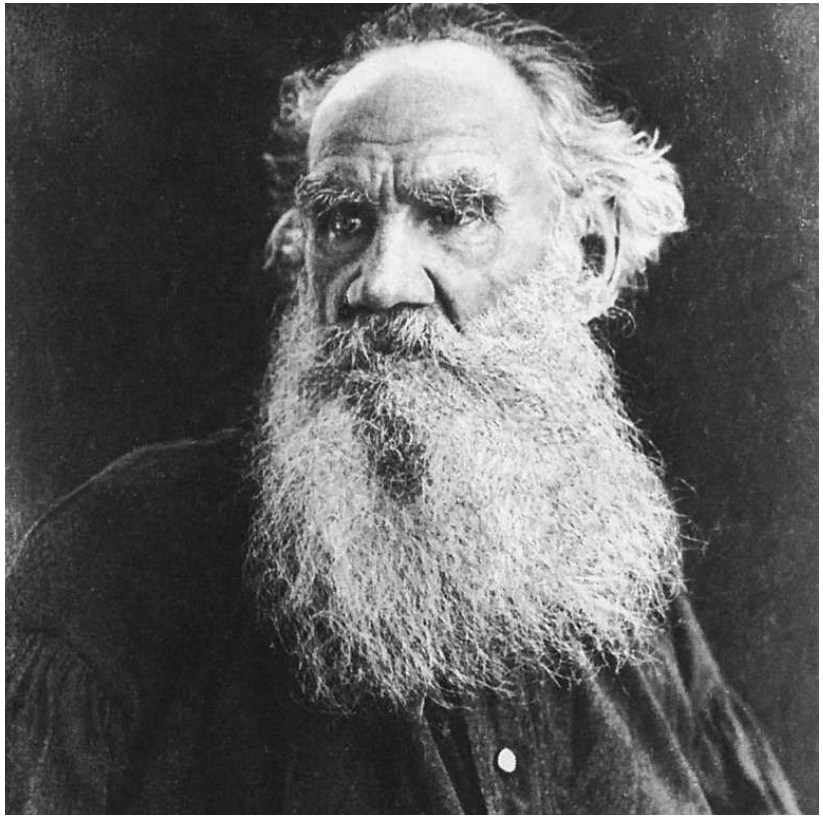
Correlates of plasticity



Test several candidate causal factors in plasticity loss:

1. Weight norm ❌
2. Weight rank ❌
3. Number of dead units ❌
4. Numerical rank of feature embeddings. ❌





“

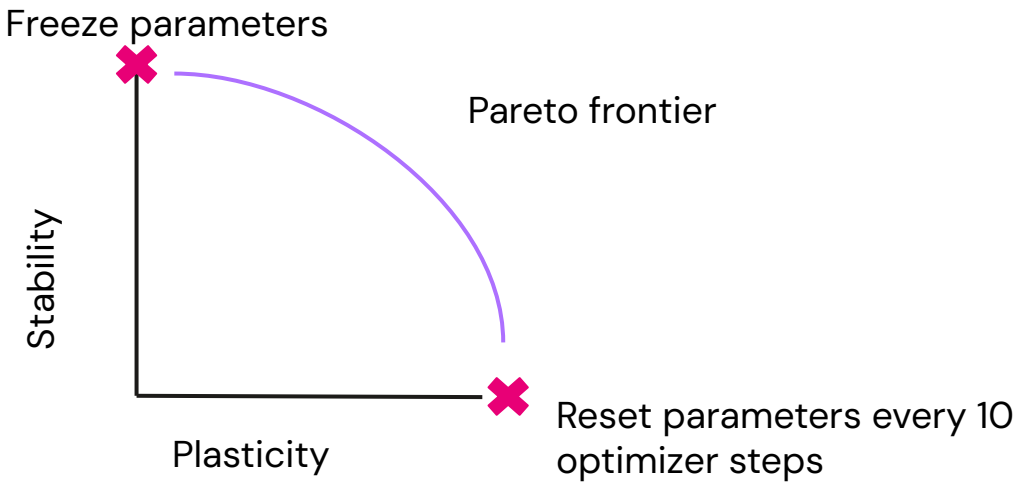
Trainable networks are all alike; every untrainable network is untrainable in its own way.

Google DeepMind

3 How to maintain plasticity

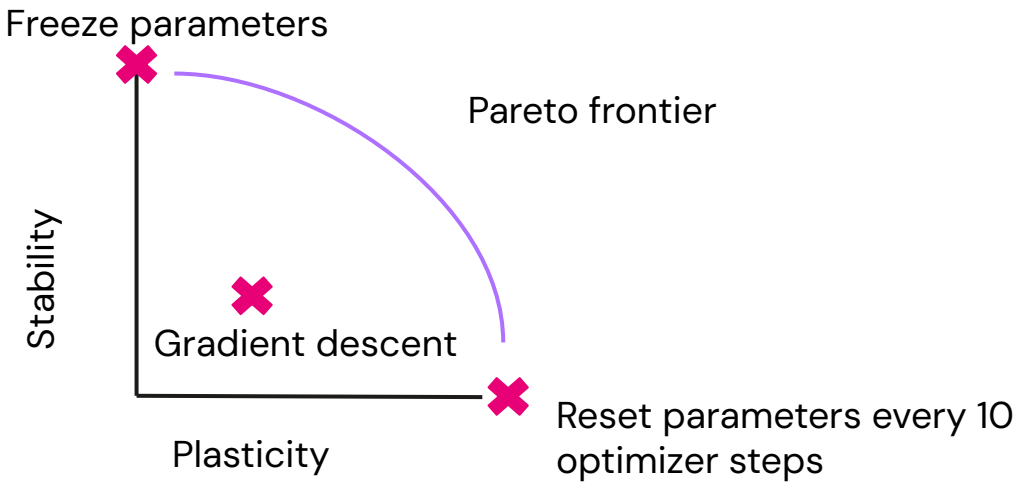
Caveat: trade-offs

- Learning systems face a “stability-plasticity trade-off”
- To maintain plasticity, could frequently re-initialize the entire network
- However, also want to be able to take advantage of things the network has learned so far
- Plasticity is also likely in tension with catastrophic forgetting (out of scope of this talk)



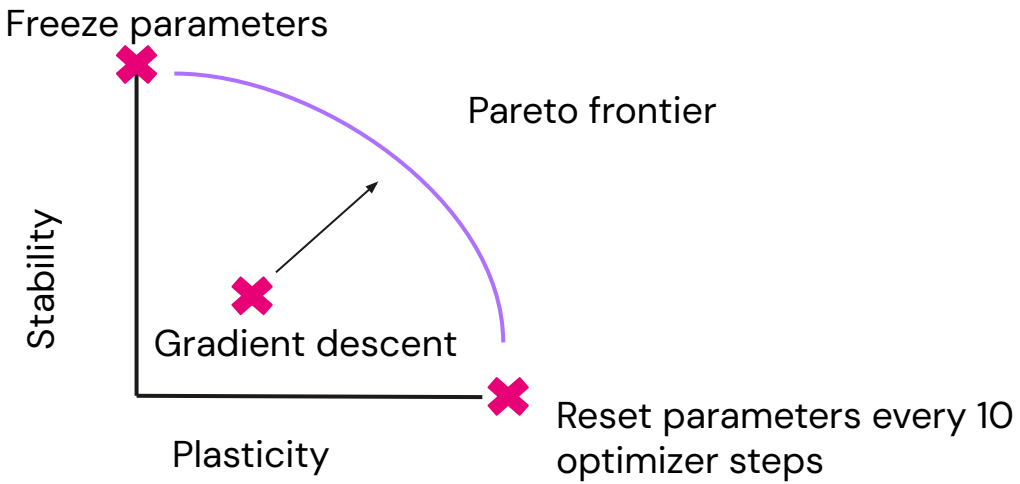
Caveat: trade-offs

- Learning systems face a “stability-plasticity trade-off”
- To maintain plasticity, could frequently re-initialize the entire network
- However, also want to be able to take advantage of things the network has learned so far
- Plasticity is also likely in tension with catastrophic forgetting (out of scope of this talk)

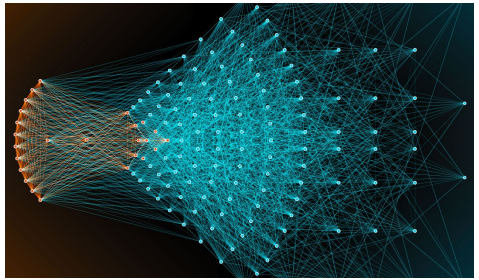


Caveat: trade-offs

- Learning systems face a “stability-plasticity trade-off”
- To maintain plasticity, could frequently re-initialize the entire network
- However, also want to be able to take advantage of things the network has learned so far
- Plasticity is also likely in tension with catastrophic forgetting (out of scope of this talk)



Three simple fixes



Bigger networks

Scale is beneficial for a variety of reasons – easier to smoothly interpolate, more expressive in general

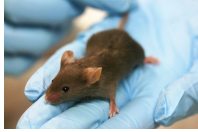
Stable architectures

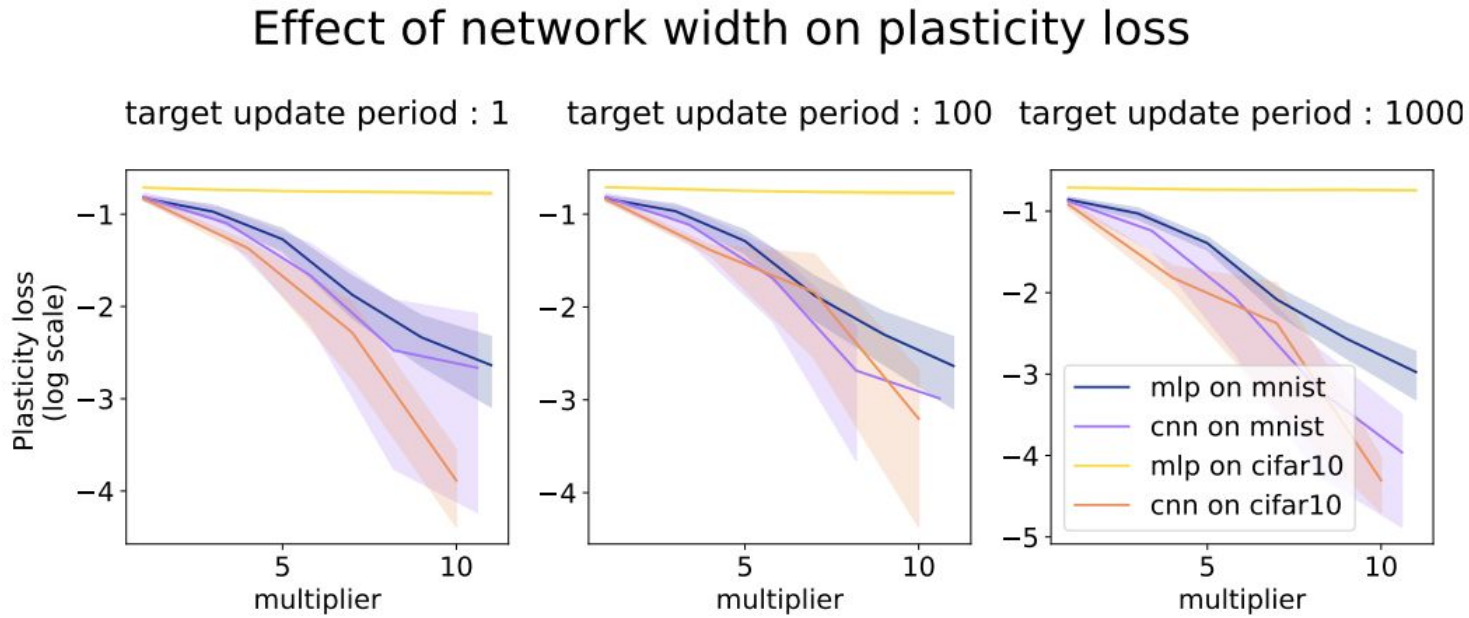
Use architectural tricks (normalization, residual connections, etc.) known to make optimizers behave better.

Nicer parameterizations

Choose parameterizations of the learning problem that are scale-invariant and induce nice loss landscapes

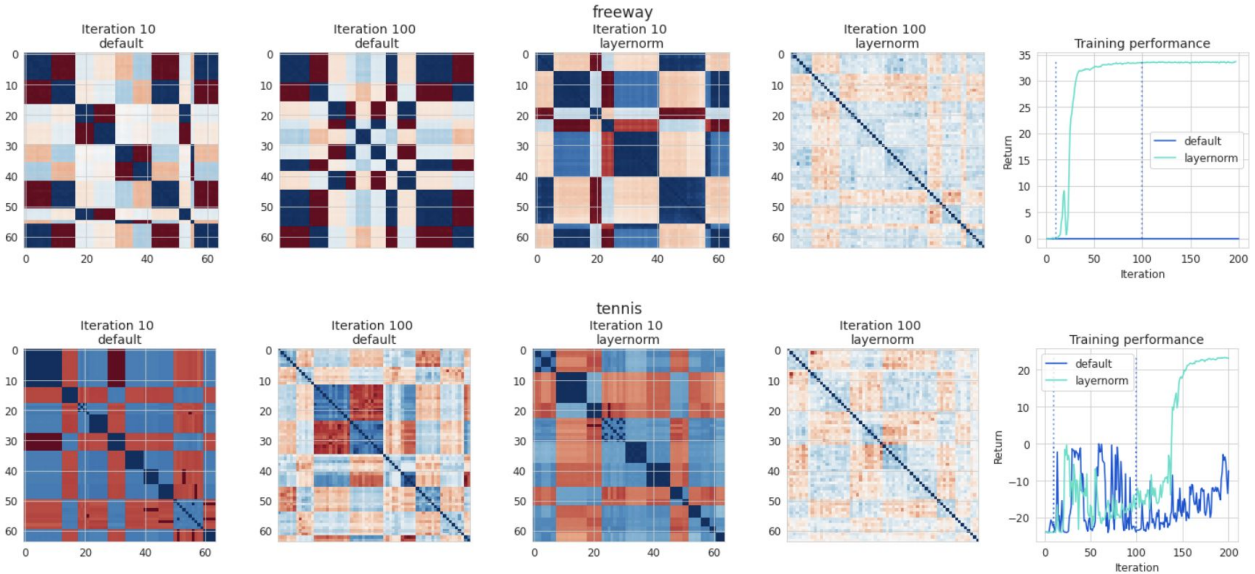
Solution 1: get a bigger network

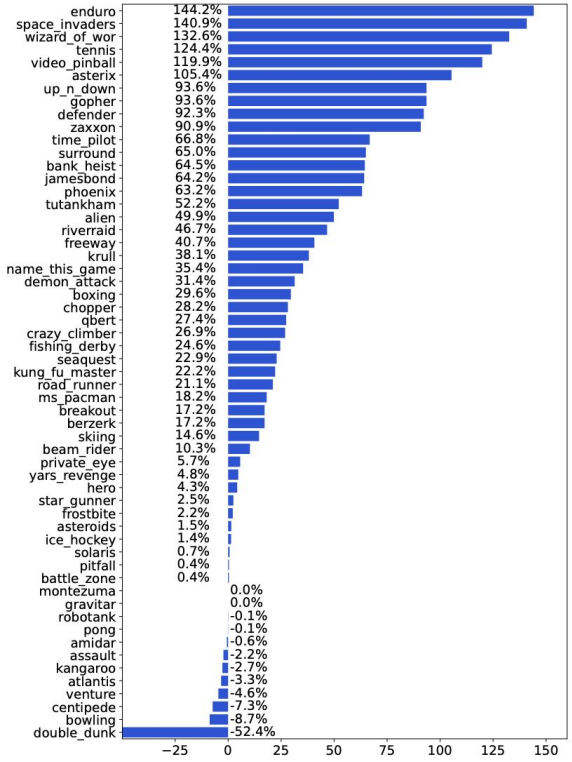


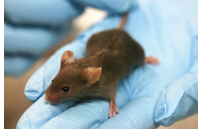


Larger networks lose plasticity less. However, need **extremely** overparameterized networks for this to work, and size of network needed scales with problem complexity.

Solution 2: use a more stable architecture





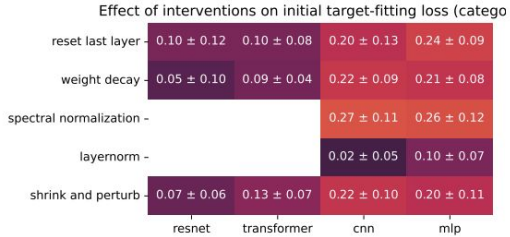


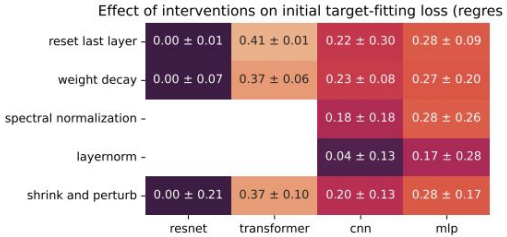
Solution 3: re-parameterize the learning objective

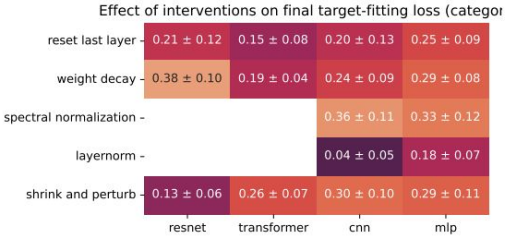
Softmax cross-entropy has nice bounded gradients (w.r.t. logits), and is translation-invariant w.r.t. the scale of the support of the distribution

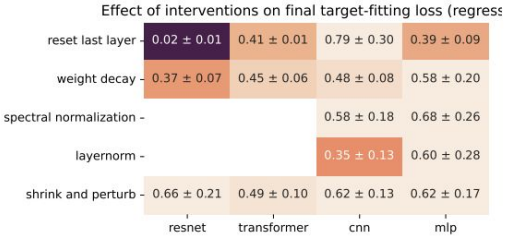
$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Re-parameterizing network output this way resolves a lot of optimization issues that come up from having increasing targets in RL.









Google DeepMind

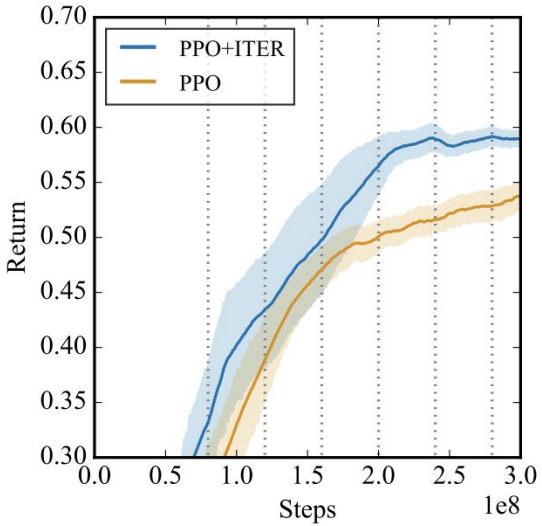
Fun solutions

“Trivial” solution

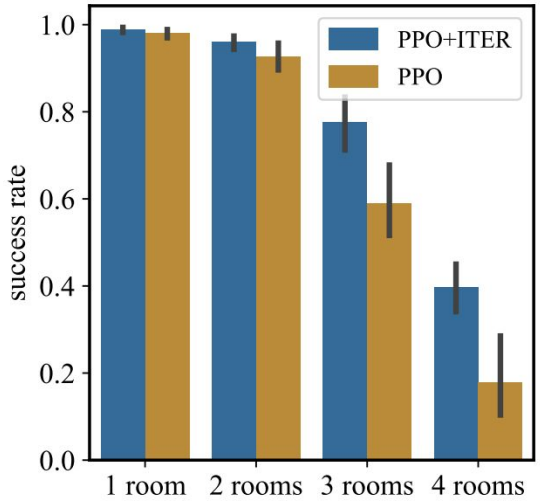
(Igl, Farquhuar, Whiteson; 2023)

Reset the entire network!

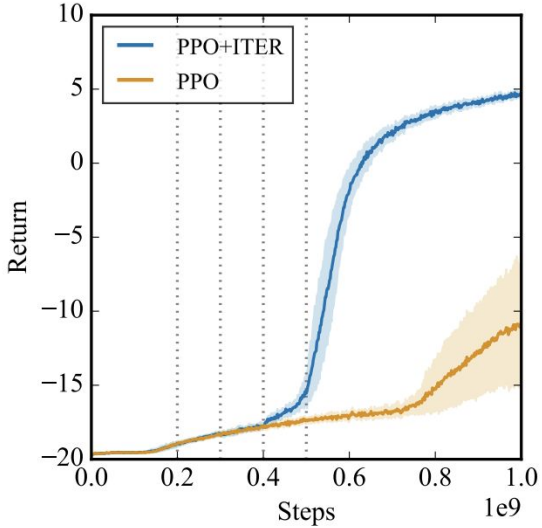
Not actually trivial: have to figure out how to get the re-initialized parameters back up to speed quickly.



(a) Multiroom



(b) Multiroom, by layout



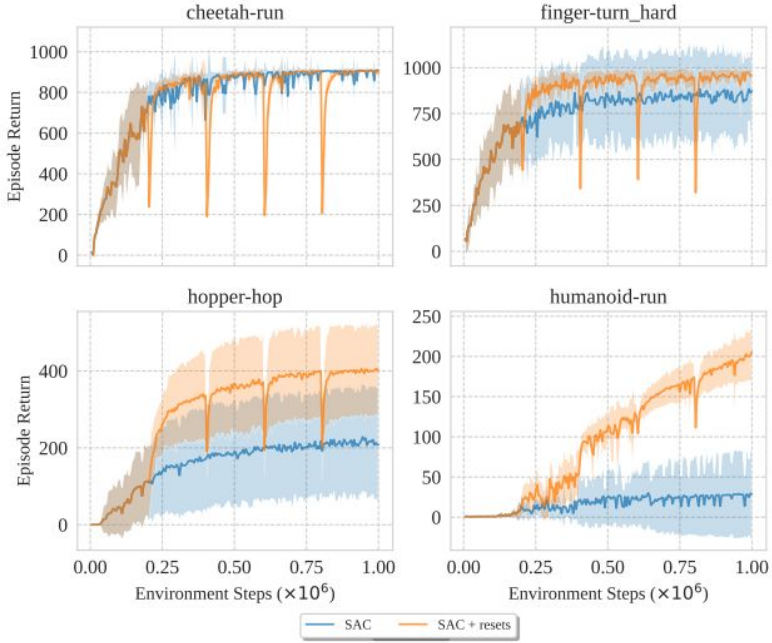
(c) Boxoban

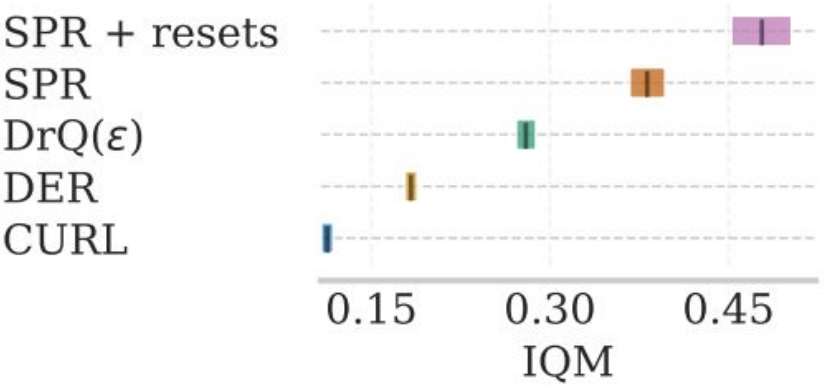
Resetting a single layer

(Nikishin*, D'Oro*, Schwarzer*, et al; 2023)

Lazier version: just reset individual layers (usually the last one).

Not as crazy as it sounds – c.f. "Are all layers created equal?" (Zhang et al., JMLR 2022).





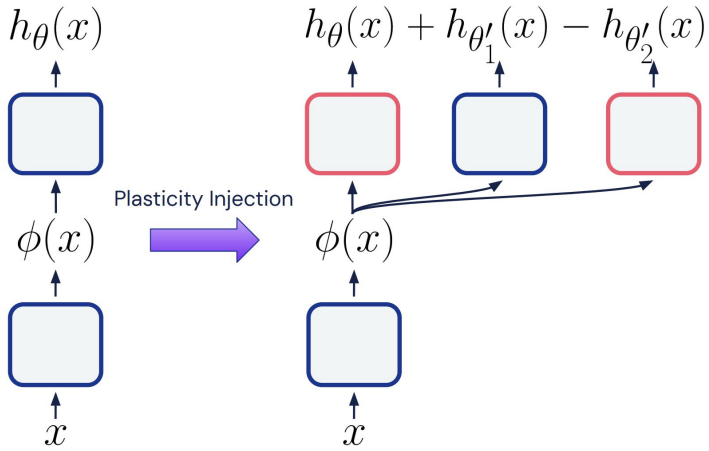
Adding another network

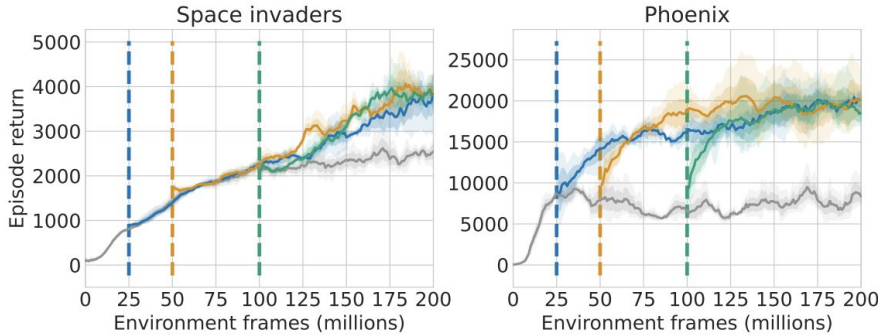
Nikishin, Oh, Ostrovski, L, Pascanu, Dabney, Barreto, 2023

Rather than resetting the network and starting from scratch, freeze network and sum the outputs of frozen + freshly initialized networks.

Number of trainable parameters stays constant, but effective capacity is increased.

Can allow DQN agents to break through plateaus in atari.

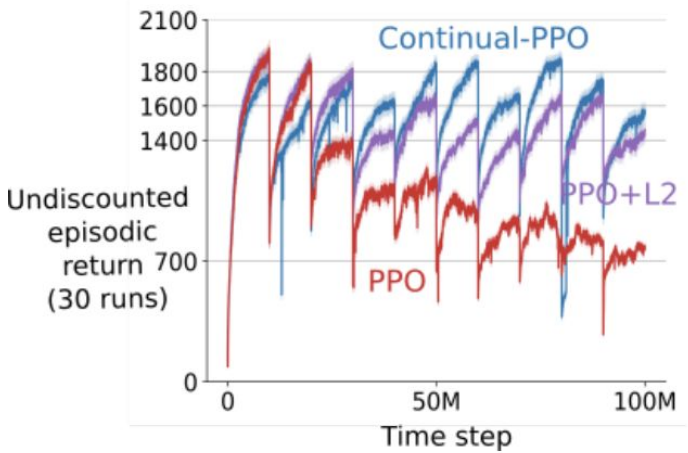




Resetting units

(Dohare, Sutton, & Mahmood; 2022)

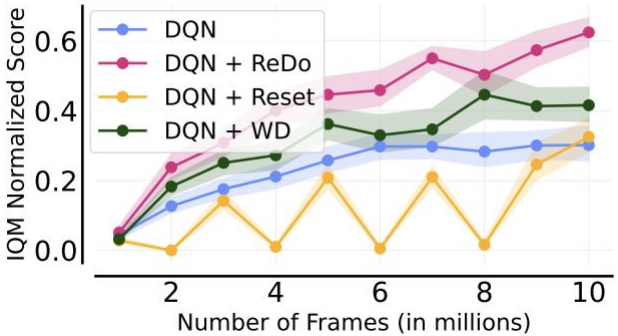
Resetting units which aren't "useful" (for some notion of utility) can improve robustness of RL algorithms in nonstationary tasks.



Can use an even simpler heuristic: reset a ReLU unit if it is zero on all inputs.

Leads to significant performance improvements in online, value-based RL agents trained on Atari domain.

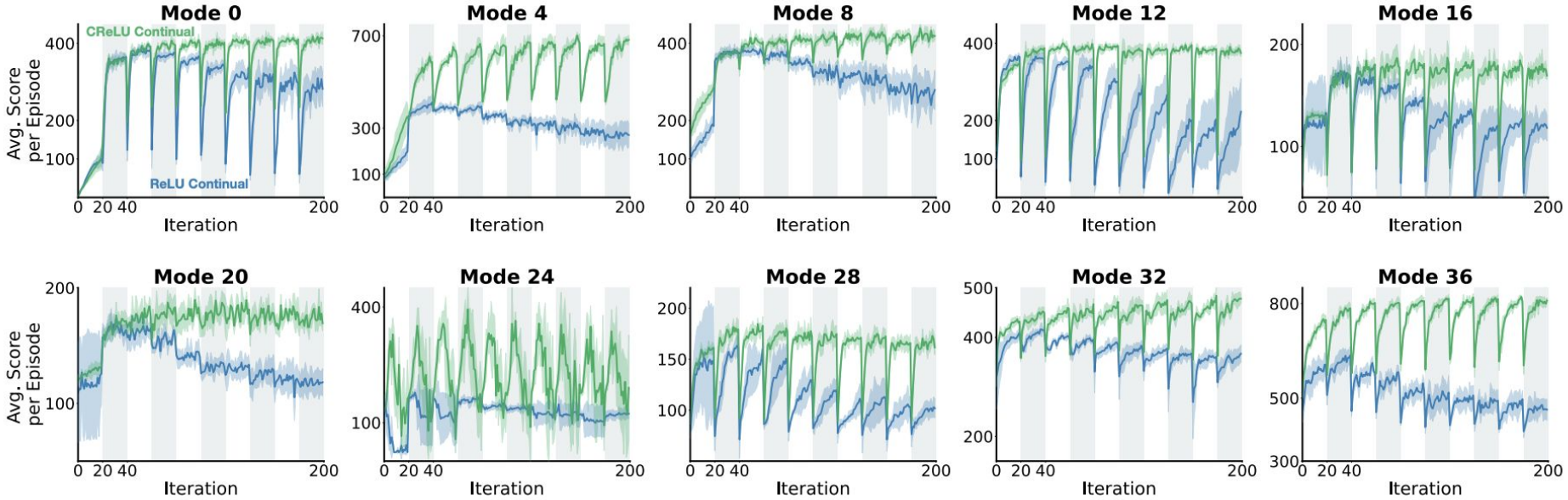
(Sokar, Agarwal, Castro, Evci, 2023)



Picking better activation functions

Abbas, Zhao, Modayil, White, & Machado. 2023

Using concatenated ReLU rather than ReLU activations improves robustness in RL agent trained on a sequence of different Atari games and levels.

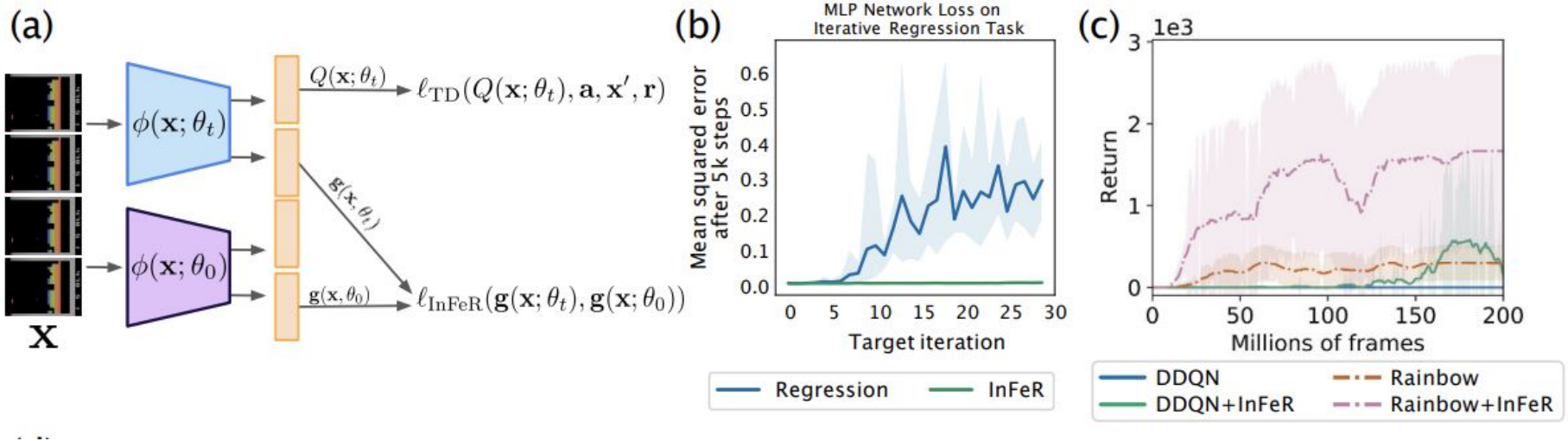


Feature regularization

L, Rowland, Dabney, 2022

Regress random projections of features back to the value they had at initialization.

Avoids representation collapse, but may limit degrees of freedom during optimization.

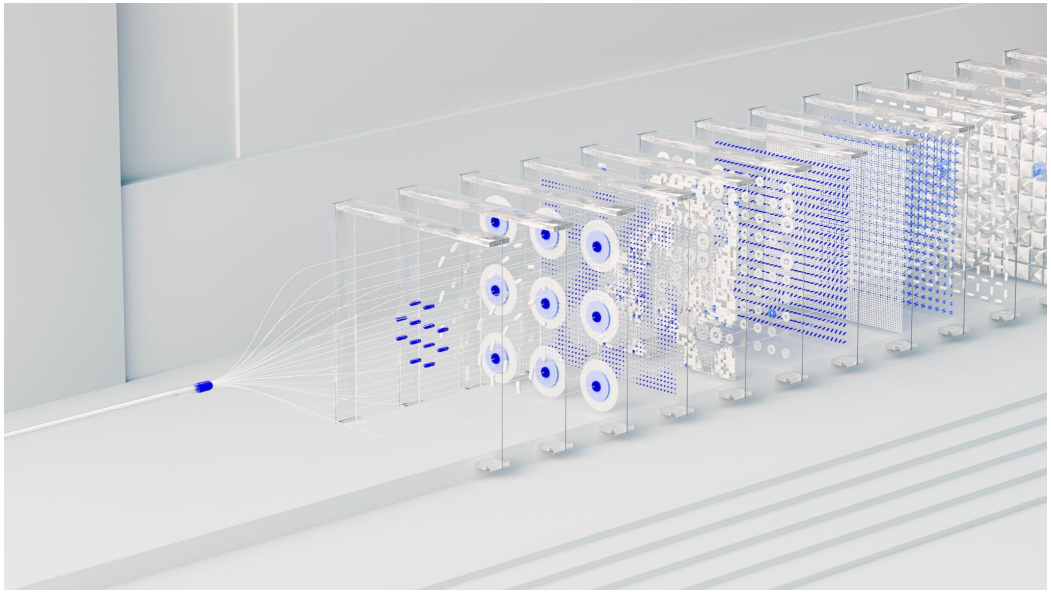


Google DeepMind

4 What's next

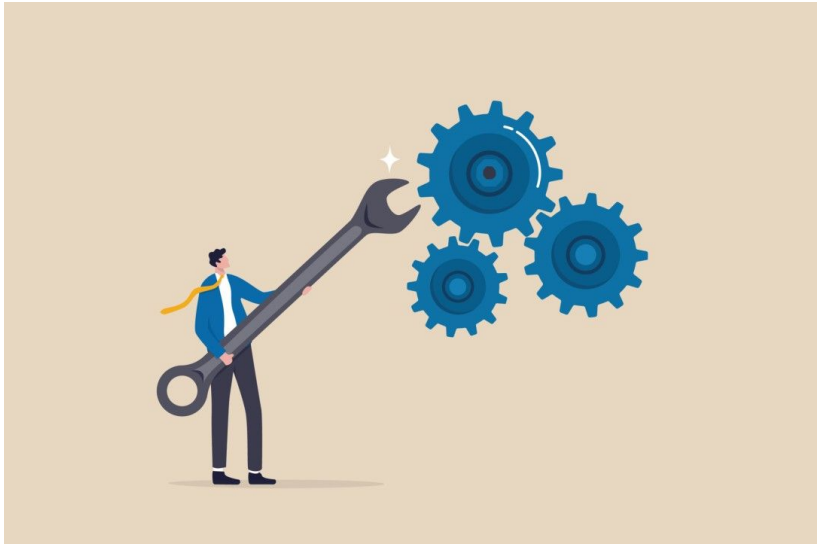
Implications for large models

- Bigger models exhibit less plasticity loss
- Harder tasks induce more plasticity loss
- Unknown whether large language models are big enough to not exhibit plasticity loss on all economically relevant tasks that might require continual learning



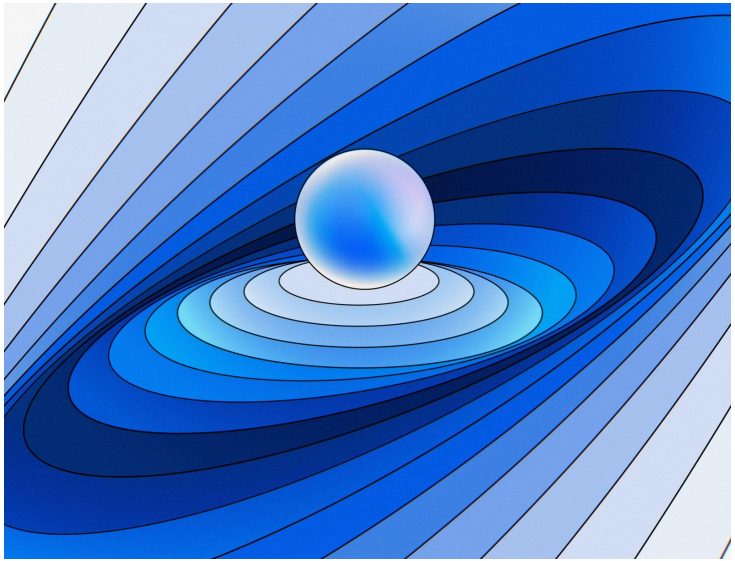
Implications *of* pre-training, and *for* fine-tuning

- Some plasticity loss is probably desirable if it improves domain-relevant performance
- Pre-training on sufficiently diverse data unlikely to reduce plasticity on the training domain
- However, fine-tuning on a sequence of many datasets might present a problem if the network loses plasticity over time



Theoretical understanding

- Still don’t have a great theoretical model of plasticity or trainability
- Signal propagation perspectives from neural network initialization are data-agnostic, don’t reflect data dependence we see in practice
- Possible connection to other findings on learning dynamics in supervised learning?



Characterizing trade-offs

Don’t have a fine-grained model of which tasks require plasticity loss and which don’t

Plasticity loss that arises from increased compression may be necessary for generalization

Suggests there may be fundamental tradeoffs between plasticity and efficiency/generalization

Conclusions

So what have we learned?

1. Plasticity loss is observed in a variety of contexts where **non-stationary training dynamics** occur
2. At least in extreme cases, this can **prevent agents from improving their performance**
3. A variety of **promising approaches** exist to **reduce** the loss of plasticity
4. But we don’t have a way of avoiding the need to eventually reset the network, i.e. we **can’t avoid plasticity loss entirely** yet.

Thanks!