Continual Learning

VISMAC 23

Simone Calderara   Angelo Porrello

AImageLab
University of Modena and Reggio Emilia, Italy

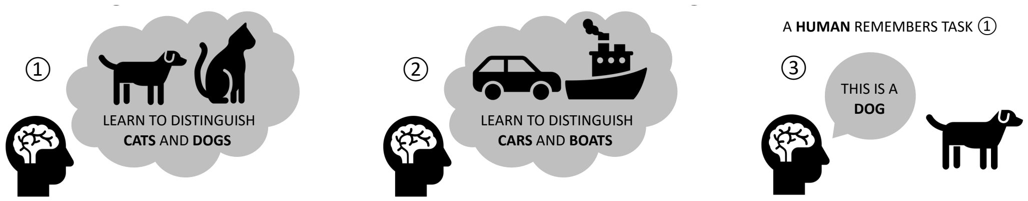
Agenda

AImage Lab
University of Modena and Reggio Emilia

Introduction

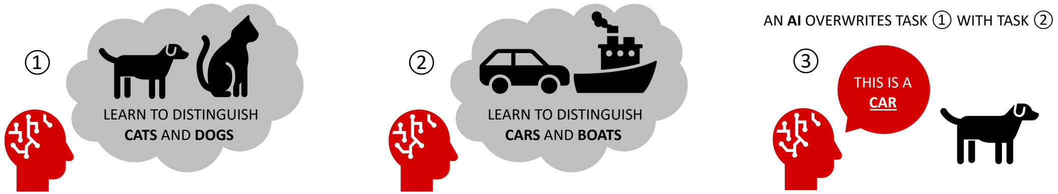
Humans and memories

Human intelligence. Human beings can learn new tasks and, at the same time, they can remember what learned before.



Neural Networks and memories

Catastrophic Forgetting. Instead, Neural Networks, when trained on a sequence of tasks, forget old tasks: *i.e.*, they may perform well on the examples of the current task, but their capabilities on old tasks drop.



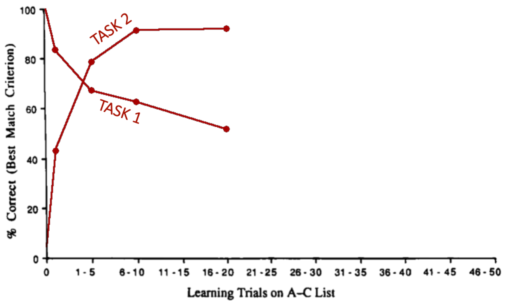
①
LEARN TO DISTINGUISH CATS AND DOGS
②
LEARN TO DISTINGUISH CARS AND BOATS
AN AI OVERWRITES TASK ① WITH TASK ②
③
THIS IS A CAR

Humans vs Neural Networks

Barnes & Underwood first highlight the dynamics of forgetting in **1959**.

- **Task 1**: subjects learn a series of word pairs (*e.g.*, home-star, field-sky, ...);
- **Task 2**: right-hand items of pairs change (*e.g.*, home-field, field-car, ...).

Result. Human subjects forget about pairs from Task 1 as more pairs from Task 2 are observed...



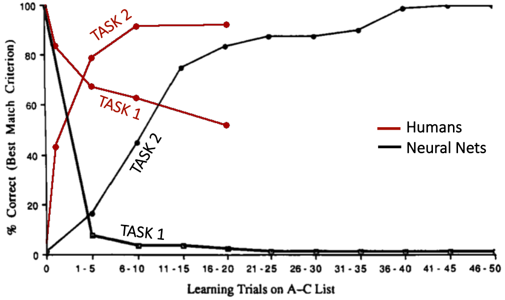
Barnes & Underwood. "Fate" of first-list associations in transfer theory. J Exp. Psychol, 1959.

Humans vs Neural Networks (2)

In 1989, McCloskey & Cohen repeat the same experiment with Artificial Neural Networks.

They find that degradation is much more severe in NNs: a **few learning steps** are enough to drive accuracy to **zero**.

This is an intrinsic property of NNs, which they call ***Catastrophic Forgetting***.



McCloskey & Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Psychol Learn Motiv, 1989

Catastrophic forgetting: why?

So, Neural Networks forget.

Why? How?

Catastrophic forgetting: why?

So, Neural Networks forget.

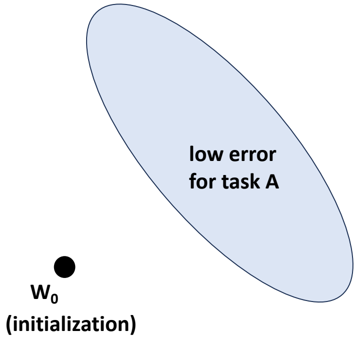
Why? How?

It is principally due to the optimization algorithm (*i.e.*, gradient descent or its stochastic variants – SGD).

Catastrophic forgetting: why? (2)

A NN has its set of **parameters** W (*i.e.*, a set of weights and biases).

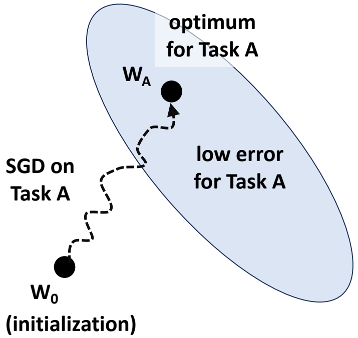
Optimization starts from a random point W_0 in parameter space and reaches a *local minimum*, within a region where the error is **small** on the task being learned.



Catastrophic forgetting: why? (3)

Given a task A:

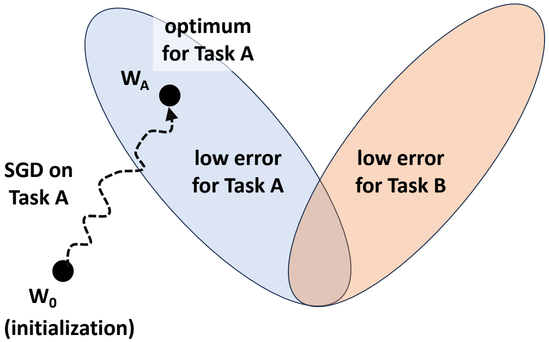
Optimization starts from a random point W_0 in parameter space and reaches a *local minimum* W_A, within a region where the error is **small** on the task being learned.



Catastrophic forgetting: why? (4)

Considering a second incoming task B:

The task B has its own loss landscape that the network is going to optimize, irrespective of the error of any other task.



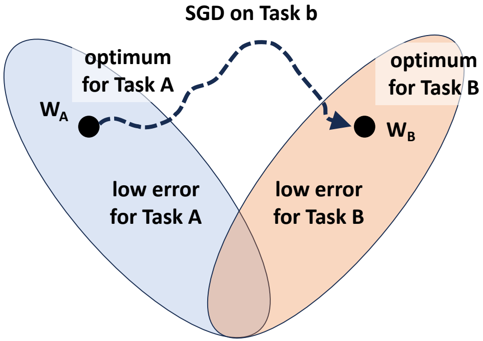
Catastrophic forgetting: why? (5)

To change its behavior or learn how to perform the second task, the network must **change** its parameters.

SGD **moves without constraints** throughout a new region in parameter space, until it reaches a new local optimum $\mathrm{W_B}$.

$\rightarrow$ Catastrophic forgetting.

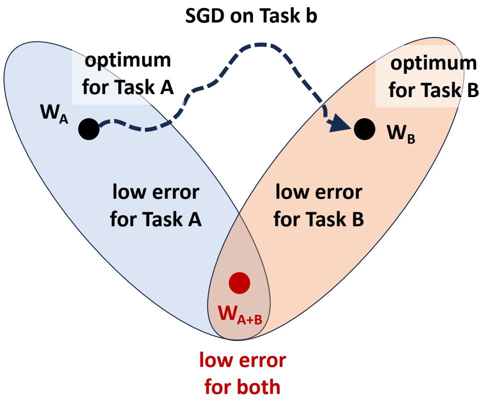
- Loss for Task B: low :)
- Loss for Task A: high :(



Catastrophic forgetting: why? (5)

Theoretically, NNs are **over-parametrized**.

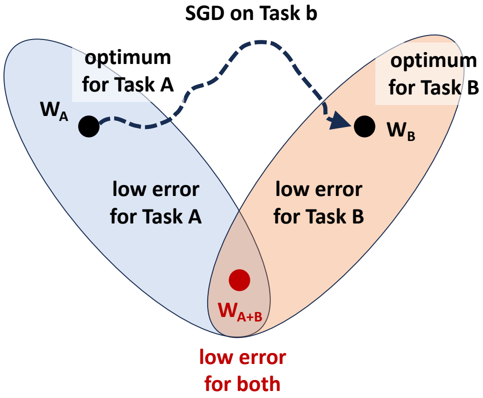
Over-parameterization makes it likely that there is a solution for Task B, $\mathrm{W}_{\mathrm{A+B}}$, that is close to the previously found solution for Task A, W_{A}.



Catastrophic forgetting: why? (6)

Continual Learning techniques aim at protecting the performance in previous tasks by constraining the subsequent learning (*e.g.*, to stay in a region of low error for Task A).

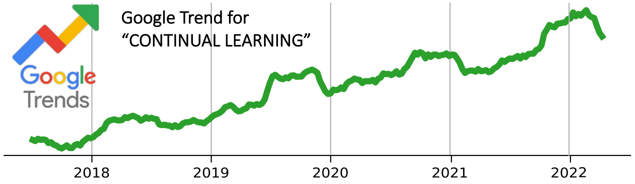
Challenge. Do it with no access to the examples of Task A.

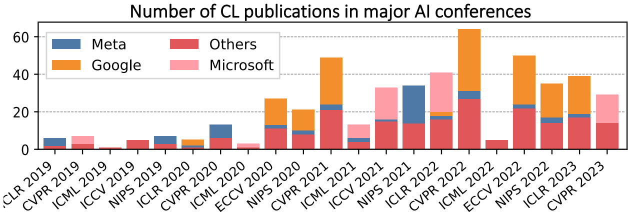


Continual Learning (CL)

Continual Learning (CL) studies how to train a ML system from a stream of changing data.

It's a rapidly growing new trend in ML research.





Continual Learning (CL)

AImage Lab
University of Modena and Reggio Emilia

Beyond academia: cutting-edge startups and ML-driven corps are pioneering CL solutions.

TESLA

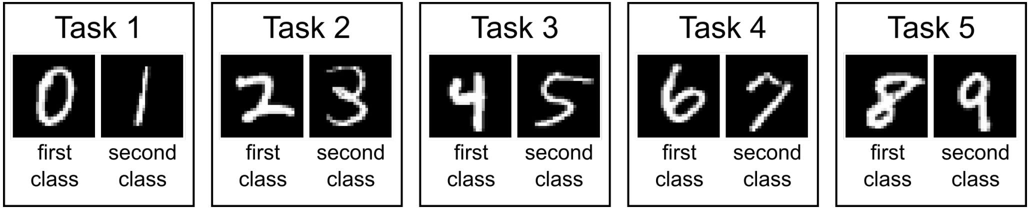
data engine

Google
TensorFlow
Extended

neurala
Lifelong-DNN™

A common CL scenario

A CL classification problem is split in T tasks; during each task $t \in \{1, ..., T\}$ input samples x and their corresponding ground truth labels y are drawn from an i.i.d. distribution D_t.



A function f, with parameters θ, has to be optimized on one task at a time, in a sequential manner.

Image from ”Three scenarios for continual learning” by Gido M. van de Ven et. al. (2019).

Common objective

GOAL: to correctly classify, at any given point in training, examples from any of the observed tasks up to the current one $t \in \{1, \dots, t_c\}$:

$$\underset{\theta}{\text{argmin}} \sum_{t=1}^{t_c} \mathcal{L}_t, \quad \text{where} \quad \mathcal{L}_t \triangleq \mathbb{E}_{(x,y)\sim D_t}\big[\ell(y, f_\theta(x))\big]. \tag{1}$$

Challenge. Data from previous tasks are assumed to be unavailable, meaning that the best configuration of θ w.r.t. $\mathcal{L}_{1\dots t_c}$ must be sought without D_t for $t \in \{1, \dots, t_c - 1\}$.

A simple approach: Elastic Weight Consolidation (1)

Elastic Weigth Consolidation (EWC). Introduced by Kirkpatrick et. al. in 2017, it is one of first approaches to deal with *catastrophic forgetting*.

Biologically inspired. EWC takes inspiration from brains.

- **Plasticity.** The ability of the nervous system to change its activity in response to intrinsic or extrinsic stimuli by reorganizing its structure, functions, or connections.
- **Synaptic consolidation** enables continual learning by reducing the plasticity of synapses that are vital to previously learned tasks.

Elastic Weight Consolidation (2)

Synaptic consolidation enables continual learning by reducing the plasticity of synapses that are vital to previously learned tasks.

Similarly, EWC constraints **important** parameters **to stay close** to their **old values** through a tailored regularization loss (a **quadratic penalty**).

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

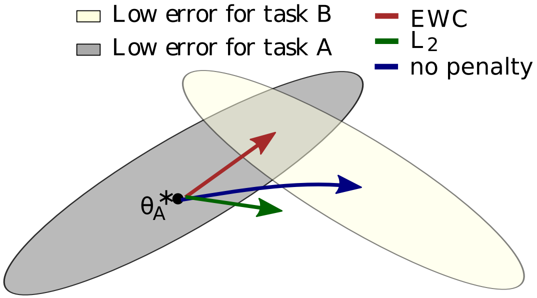


Image from "Overcoming catastrophic forgetting in neural networks" by Kirkpatrick et. al. (2017).

Elastic Weight Consolidation (3)

Similarly, EWC constraints **important** parameters **to stay close** to their **old values** through a tailored regularization loss (a **quadratic penalty**).

$$\underbrace{\mathcal{L}(\theta)}_{\text{Total loss to be optimized}} = \underbrace{\mathcal{L}_{\mathrm{B}}(\theta)}_{\text{Loss for Task B only e.g. the Cross-Entropy}} + \underbrace{\sum_{\mathrm{i}} \frac{\lambda}{2} \mathrm{F}_{\mathrm{i}} (\theta_{\mathrm{i}} - \theta^*_{\mathrm{A,i}})^2}_{\text{EWC regularization objective}}$$

where:

- θ^*_{A} is the set of optimum parameters for the previous task A
- F_{i} is an estimate of how much each parameter $\theta^*_{\mathrm{A,i}}$ is important for preserving the performance in task A
- λ is an hyper-parameter.

Elastic Weight Consolidation (4)

Similarly, EWC constraints **important** parameters **to stay close** to their **old values** through a tailored regularization loss (a **quadratic penalty**).

$$\underbrace{\mathcal{L}(\theta)}_{\substack{\text{Total loss}\\ \text{to be optimized}}} = \underbrace{\mathcal{L}_{\mathrm{B}}(\theta)}_{\substack{\text{Loss on Task B}\\ \text{e.g. the Cross-Entropy}}} + \underbrace{\sum_{\mathrm{i}} \frac{\lambda}{2} \mathrm{F}_{\mathrm{i}}(\theta_{\mathrm{i}} - \theta^{*}_{\mathrm{A,i}})^2}_{\substack{\text{EWC regularization}\\ \text{objective}}}$$

It can be imagined as a spring **anchoring** the parameters to the previous solution, hence the name **elastic**.

Importantly, the stiffness of this spring should not be the same for all parameters; rather, **it should be greater** for those parameters that **matter most** to the performance during task A.

How to estimate per-parameter importance?

The formula introduced in the previous slides requires to determine which parameters are **important**.

How? Many heuristics could be envisioned; EWC uses the diagonal of the empirical **Fisher Information Matrix** computed at the optimum of the first task θ^*_A.

$$I_{\theta^*_A} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\nabla_\theta \log p(x^{(i)}_A|\theta^*_A)}_{\text{gradients of the loss function}} \nabla_\theta \log p(x^{(i)}_A|\theta^*_A)^T$$

It can be computed from first-order derivatives (**easy to calculate**) by a simple moving average of **squared gradients**.

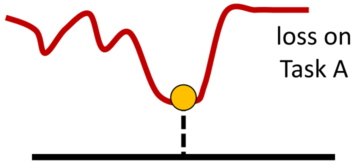
Check the PyTorch implementation here or here.

Fisher Information Matrix

There is a theoretical relation between the empirical Fisher Information Matrix (FIM) and the **second derivative** of the loss near a minimum.

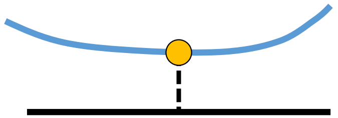
As such, the FIM captures the **curvature** of the log likelihood function: a high Fisher information indicates that the log likelihood is **sharply** peaked there.

For such a reason, it would be inconvenient to modify the corresponding weight.



High 2nd derivative

→ high importance, high penalty



Small 2nd derivative

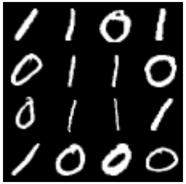
→ low importance, low penalty

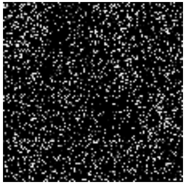
Continual Learning

Continual Learning Benchmarks

Continual Learning approaches are predominantly evaluated on abstract image classification settings:

- **Sequential MNIST**: Task 1: classify 0 and 1; Task 2: classify 2 and 3, etc.
- **Sequential CIFAR**: Task 1: airplane vs car; Task 2: bird vs cat, etc.
- **Rotated MNIST**: classify digits with rotation changing.
- **Permuted MNIST**: classify digits with a different pixel permutation for each task.





Long and complex benchmarks are regarded as more realistic, but can be quite demanding in terms of total compute.

Evaluation protocol

Train/test splits. Each task is split into training and test sets.

- **Training phase:** the model learns the tasks in a sequential manner.
- **Testing phase:** the model is evaluated all on the test sets jointly (*order does not matter*).

Multiple epochs are usually allowed on each task, so that the model can fit the current task well.

Hyperparameter tuning and cross-validation Unfortunately, no established and common practices already exist.

Metrics

Metrics. The **average accuracy** is the most common one.

Formally. Given a test set for each of the T tasks, and indicating with $\mathrm{R}_{\mathrm{i,j}}$ the test classification accuracy on task $\mathrm{t_j}$ after observing the last sample from task $\mathrm{t_i}$, we have:

$$\text{Average Accuracy:} \quad \mathsf{ACC} = \frac{1}{\mathrm{T}} \sum_{\mathrm{i}=1}^{\mathrm{T}} \mathrm{R}_{\mathrm{T,i}}. \tag{2}$$

It can be measured either at any given point within the tasks’ sequence, or after the end of the last task (as in the formula).

Other metrics could be used, such as **Backward and Forward Transfer** (BWT and FWT), introduced by David Lopez-Paz et. al. in ”Gradient Episodic Memory for Continual Learning”.

Evaluation and metrics

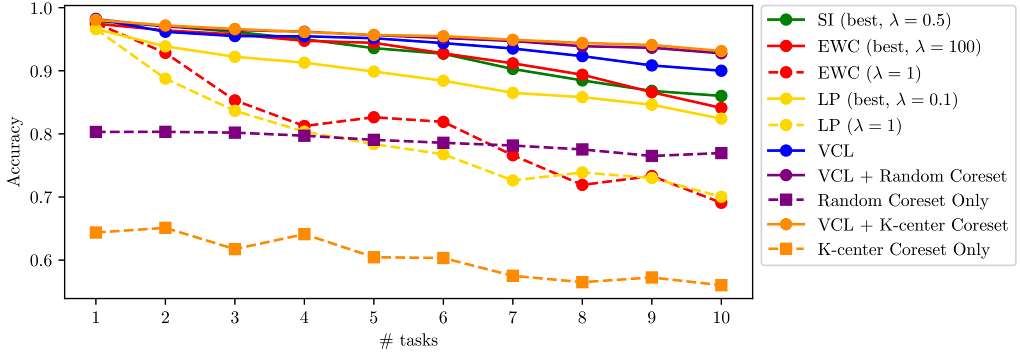


Figure 2: Average test set accuracy on all observed tasks in the Permuted MNIST experiment.

Image from ”Variational Continual Learning” by Nguyen et. al. (2017).

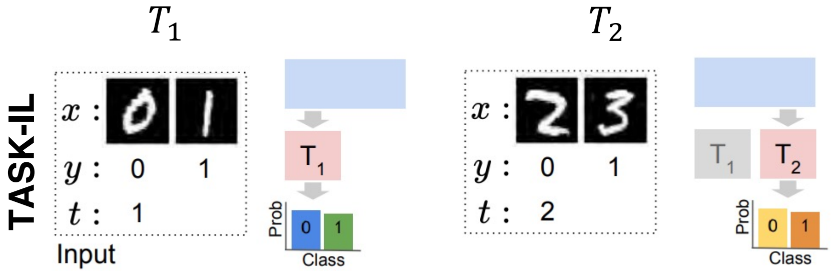
Continual Learning: three scenarios

Three main evaluation scenarios:

- Task-Incremental Learning (Task-IL)
- Class-Incremental Learning (Class-IL)
- Domain-Incremental Learning (Domain-IL)

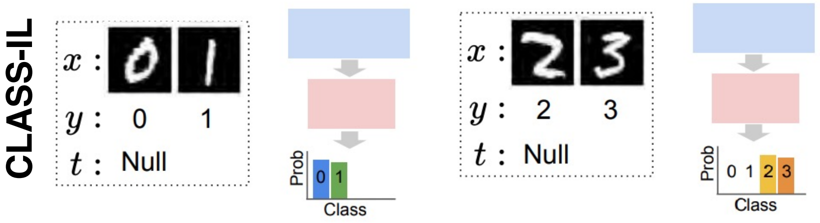
Task-Incremental Learning (Task-IL)

- During training, each task is learned separately;
- At test time, each example to be tested is coupled with its **task identity**, indicating the index of the task the example belongs to.
- This way, the model can select the right classification head.



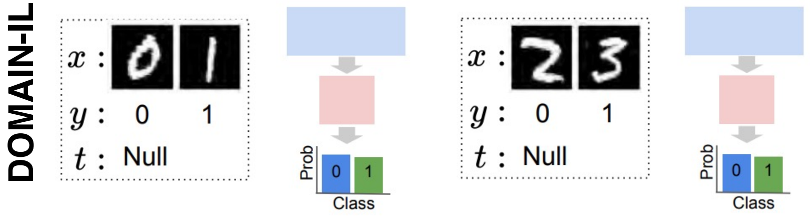
Class-Incremental Learning (Class-IL)

- As in Task-IL, each task is learned separately during training;
- Differently from Task-IL, task identities are **not provided** at test time;
- Class-IL is more **challenging**, as the model has to guess the correct class among all the classes seen so far.



Domain-Incremental Learning (Domain-IL)

- Differently from both Task-IL and Class-IL, the set of classes remains **stable** across the tasks and **does not grow**,
- Each task introduces a **domain shift**, a change in the underlying probability distribution the input are sampled from,
- It may be simulated through data transformation (*e.g.* rotations, permutations),

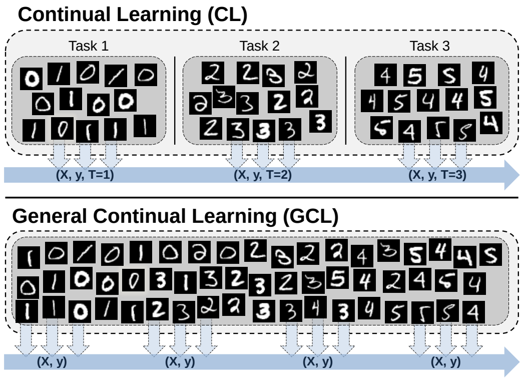


Other scenarios in literature

AImage Lab
University of Modena and Reggio Emilia

General Continual Learning: no clear boundaries between subsequent tasks; the data distribution is supposed to change smoothly over time.

Online Class-IL/Task-IL: only a single pass (*epoch*) is allowed on each task.



Other scenarios in literature (2)

Continual semi-supervised/self-supervised learning: examples may be either partly labeled, or completely unlabeled.

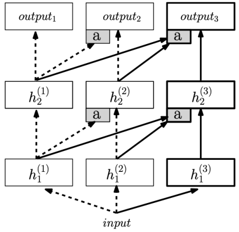
Data-Incremental Learning (Data-IL): the set of classes is stable across tasks; the tasks derive from different partitions of the original training set.

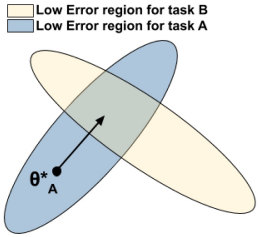
N.B. In this lecture, we will focus on Class-IL, which is by far the most studied.

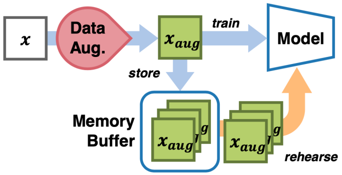
A taxonomy of CL approaches

CL approaches are typically categorized into three categories:

- **Architectural methods**: distinct sub-models for distinct tasks.
- **Regularization methods**: specific loss terms (weight importance, knowledge distillation) to prevent the model from changing.
- **Replay methods**: store previously seen examples in a memory buffer and use them in later iterations.







Architectural approaches

Weight sharing

Observation. NNs forget as the updates introduced in later tasks overwrite the optimal parametrization attained for past tasks.

- Learning all incremental tasks with a **shared** set of parameters (i.e., a single model as well as one parameter space) is a major cause of the **inter-task interference**.

How can we mitigate it?

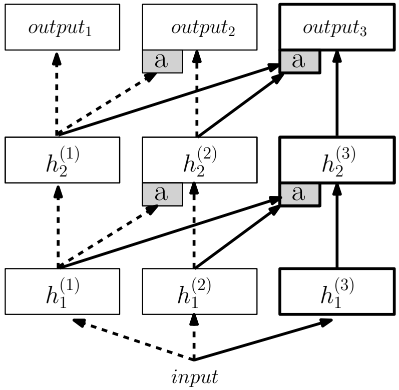
Architectural approaches

Architectural methods devote **distinct** sub-models/task-specific parameters for **distinct** tasks.

Main idea. adding new parameters, tailored for the new tasks.

These approaches **isolate** parameters for specific tasks and can guarantee maximal stability by **fixing** the parameter subsets of previous tasks.

Image from "Progressive Neural Networks" (PNN) by Rusu et. al. (2016).

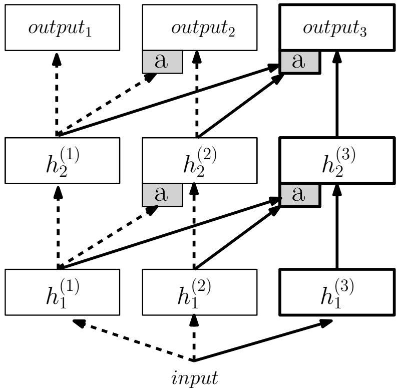


Progressive Neural Networks

An example: **Progressive Neural Networks** (PNN) by Rusu et. al..

In PNN, *Catastrophic forgetting* is prevented by instantiating a new neural network (a **column**) for each task being solved, while **transfer** is enabled via **lateral connections** to features of previously learned columns.

Image from ”Progressive Neural Networks” (PNN) by Rusu et. al. (2016).



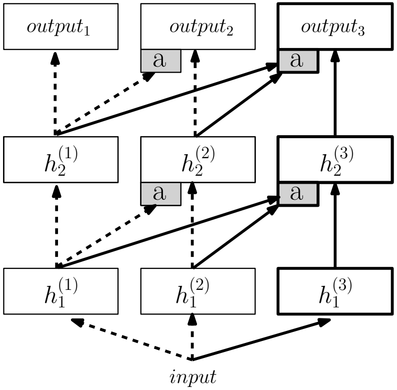
Progressive Neural Networks: advantages

Accuracy. Progressive networks are **immune** to catastrophic forgetting **by design**.

Knowledge reuse. The addition of new capacity alongside pretrained networks gives these models the flexibility to both **reuse old computations** and learn new ones.

... what about its **disadvantages**?

Image from "Progressive Neural Networks" (PNN) by Rusu et. al. (2016).

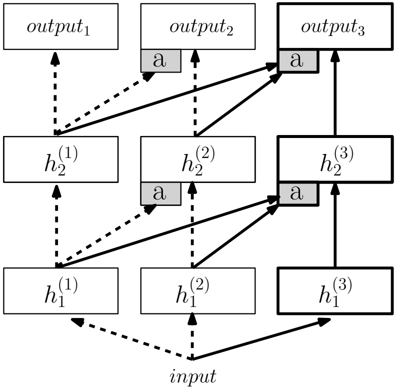


Progressive Neural Networks: disadvantages (1)

Not scalable. The number of parameters grow with the number of tasks.

- Huge memory requirements :(
- There is an intrinsic bound on the number of tasks that can be learned :(

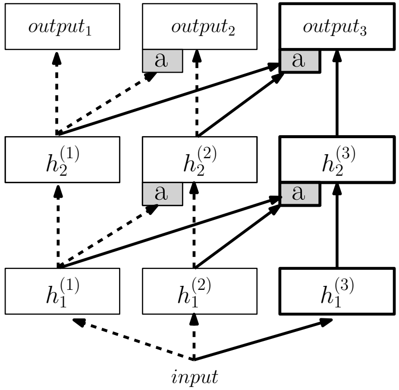
Image from "Progressive Neural Networks" (PNN) by Rusu et. al. (2016).



Progressive Neural Networks: disadvantages (2)

Not suitable for Class-IL. Choosing which column to use for inference requires knowledge of the task label, which is allowed only in the Task-IL scenario.

Image from ”Progressive Neural Networks” (PNN) by Rusu et. al. (2016).



Solutions

Addressing the scalability issue. The authors of PNN observed that only a fraction of the new capacity is actually utilized, and that this trend increases with more columns.

Countermeasures. Adding fewer layers or less capacity, by pruning, or by online compression during learning. Some examples:

- **PackNet** (next slide)
- PathNet
- Piggyback
- HAT (Hard Attention to the Task)

PackNet

PackNet by Mallya et. al. exploit redundancies in large deep networks to **free up** parameters that can then be employed to learn new tasks.

Iterative pruning and **Network re-training**. PackNet sequentially **packs** multiple tasks into a single network with minimal drop in performance and storage overhead.

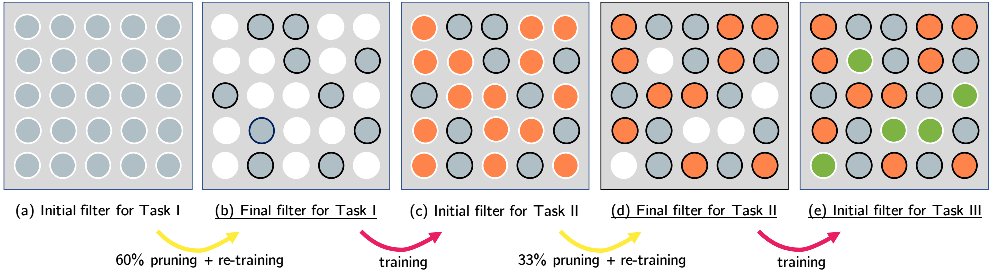

(a) Initial filter for Task I
(b) Final filter for Task I
(c) Initial filter for Task II
(d) Final filter for Task II
(e) Initial filter for Task III
60% pruning + re-training
training
33% pruning + re-training
training

Image from "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning" by Mallya et. al.

Regularization approaches

Regularization approaches

Regularization approaches add **explicit regularization terms** in the loss function to balance the old and new tasks.

They apply **weight sharing** across tasks and do not instantiate additional parameters.

- Reduced memory footprint :)
- An example from previous slides: EWC

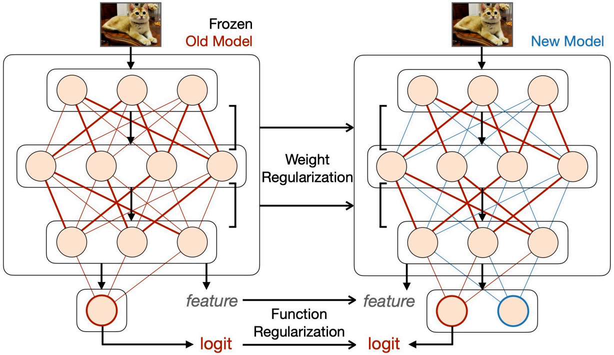
However, the auxiliary regularization objective usually requires to store a **frozen** copy of the old model for **reference**.

Regularization approaches (2)

Depending on the target of regularization, such methods can be divided into two sub-directions.

- **Weight regularization**
- **Function regularization**

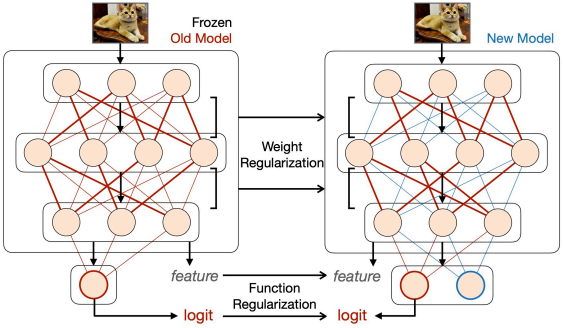
Image from ”A Comprehensive Survey of Continual Learning: Theory, Method and Application” by Wang et. al. (2023).



Weight regularization

Weight regularization approaches selectively regularize the variation of network parameters.

e.g., through a quadratic **penalty** that penalizes the variation of each parameter depending on its ***importance*** to performing the old tasks.

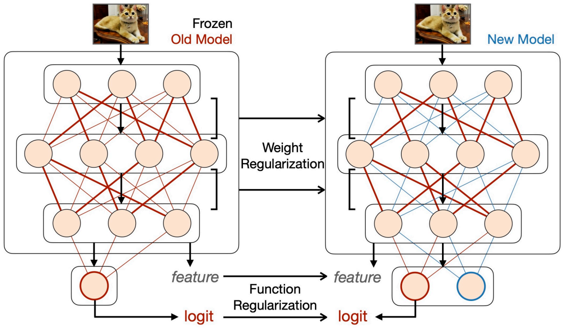


Some examples: EWC, Synaptic Intelligence (SI), MAS, RWalk, etc.

Function regularization

Function regularization approaches target the intermediate or final output of the prediction function.

Teacher-student paradigm. They typically employ the previously-learned model as the teacher and the currently-trained model as the student.



Some examples: Learning without Forgetting (LwF, next slides).

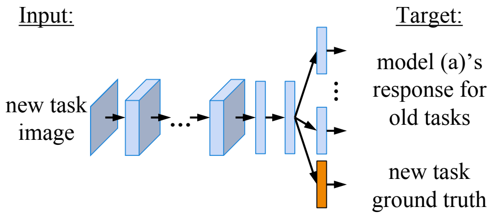
Learning without Forgetting (LwF)

Goal. To learn a network that can perform well on both old tasks and new tasks when only new-task data is present.

LwF in few words

Applying **Knowledge Distillation (KD)** between the teacher (the old network, with frozen parameters) and the student (the current one, learning the current task).

(e) Learning without Forgetting



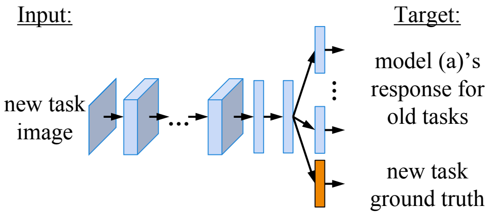
Learning without Forgetting (LwF)

Step 1: Given a new-task image, compute a forward pass through the network appointed at the end of the previous task (the *teacher*).

This way, you have an output (probability), which can be treated as an additional "*pseudo label*" for the new-task data.

Step 2: Train a new network (the *student*) only on the examples of the new task, using both the true labels and the generated pseudo labels.

(e) Learning without Forgetting



Regularization approaches: a summary

Advantages.

- Bounded memory cost
- They can further distills knowledge on the large stream of unlabeled data that may be available in the wild

Disadvantages.

- Not very effective on long tasks
- Vulnerable to domain shift between tasks

Solution. Provide a few training samples from old tasks (**replaying approaches** $\rightarrow$).

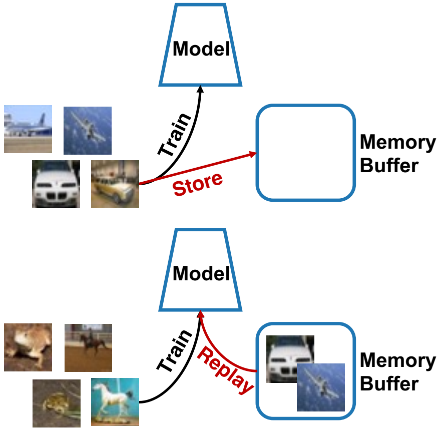
Rehearsal approaches

Rehearsal approaches

Rehearsal approaches store previously seen examples in a **memory buffer** and use them in later iterations.

Experience replay (ER). An old, simple, yet surprisingly effective baseline for CL.

- Simple and straightforward :)
- Performance proportional to memory size :(
- It may clash with privacy constraints :(



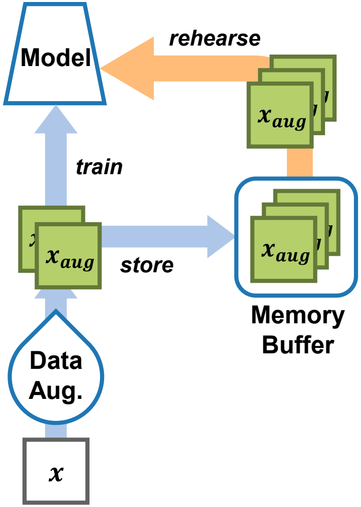
Experience replay (ER)

Experience replay (ER) stores a few old training samples within a small memory buffer $\mathcal{B}$, with fixed memory capacity.

- *e.g.*, it can contain only 500 examples of past tasks.

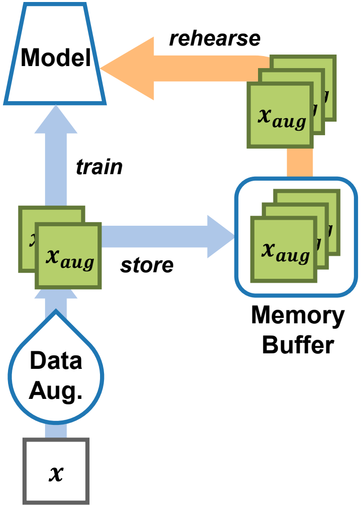
Loss function. Indicating with ℓ the cross-entropy loss between the output f_θ and the true labels y, we have that:

$$\mathcal{L}_{\text{ER}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\left[\ell(y, f_\theta(x))\right] + \mathbb{E}_{(x,y)\sim\mathcal{B}}\left[\ell(y, f_\theta(x))\right].$$



Experience replay (ER)

$$\mathcal{L}_{\text{ER}} = \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_t}\left[\ell(y, f_\theta(x))\right]}_{\substack{\textbf{Current task} \\ \text{loss on a batch from} \\ \text{the current task}}} + \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{B}}\left[\ell(y, f_\theta(x))\right]}_{\substack{\textbf{Memory buffer} \\ \text{loss on a batch sampled} \\ \text{from the memory buffer } \mathcal{B}}}$$



Experience Replay (ER): pseudocode

Algorithm 1 Experience Replay (ER) with Reservoir Sampling

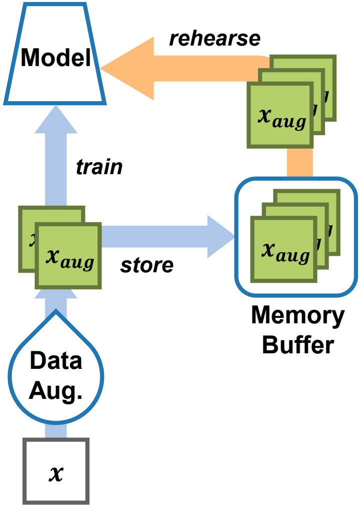
```
procedure Train($D, \theta, \alpha, k$)
    $M \leftarrow \{\}$
    for $t = 1, ..., T$ do
        for $(x, y)$ in $D_t$ do
            // Draw batch from buffer:
            $B \leftarrow \text{sample}(x, y, k, M)$
            // Update parameters with mini-batch SGD:
            $\theta \leftarrow \text{SGD}(B, \theta, \alpha)$
            // Memory buffer update:
            $M \leftarrow M \cup \{(x, y)\}$
        end for
    end for
    return $\theta, M$
end procedure
```

Challenges

How to build the memory buffer?

Challenges. Due to the extremely limited storage space, the key challenges consist of how to construct and how to exploit the memory buffer.

As for construction, the preserved old training samples should be carefully selected, compressed, augmented, and updated, in order to recover the past information in an adaptive manner.



Reservoir sampling

Reservoir sampling provides an **online strategy** to construct the memory buffer.

In particular, it solves the problem of keeping some limited number M of N total items seen before with **equal probability** $\frac{M}{N}$ when you don't know what number N will be in advance.

Algorithm 2 Reservoir Sampling

```
procedure Reservoir($M, N, x, y$)
    if $M > N$ then
        $M[N] \leftarrow (x, y)$
    else
        $j =$ randint($\min = 0, \max = N$)

        if $j < M$ then
            $M[j] \leftarrow (x, y)$
        end if
    end if
    return M
end procedure
```

Other sampling strategies

Other common sampling strategies.

- **Ring Buffer**, which further ensures an equal number of old training samples per class,
- **Weighted** reservoir sampling, retain difficult examples with higher probability,
- **Mean-of-Feature** selects an equal number of old training samples that are closest to the feature mean of each class,
- Gradient-based or optimizable strategies as **GSS**, which maximizes the **sample diversity**.
- Other fixed principles, such as the **k-means**.

Experience Replay: issues

Due to its simplicity, ER is an ideal starting point to develop a strong CL method.

However, it is affected by some key issues:

- ER repeatedly optimizes a relatively small buffer: possible overfitting problem
- Incrementally learning a sequence of classes implicitly biases the network towards newer tasks.

Solution. Again, Knowledge Distillation (next slide).

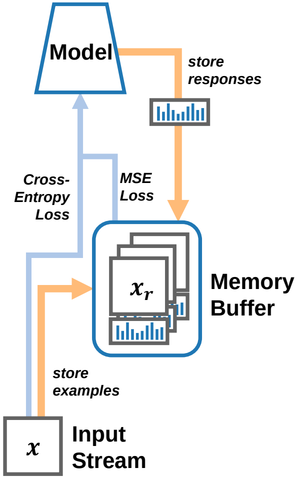
Dark Experience Replay (DER)

Dark Experience Replay, DER: an approach relying on *Dark Knowledge* for retaining past experiences.

As ER, it maintains a subset of past network responses in a buffer $\mathcal{B}$; then, in addition to the loss of the current task $\mathcal{L}_{t_c}$, DER minimizes the L^2 distances between past and current outputs for buffer datapoints:

$$\mathcal{L}_{t_c} + \alpha\, \mathbb{E}_{(x,z)\sim\mathcal{B}}\left[\, \|z - h_\theta(x)\|_2^2 \,\right].$$

DER++ is a variant of DER that also asks the learner to predict the ground truth labels for past examples.



Dark Experience Replay (DER)

$$\mathcal{L}_{t_c} \underbrace{+ \alpha\, \mathbb{E}_{(x,z)\sim\mathcal{B}}\left[\, \|z - h_\theta(x)\|_2^2 \,\right]}_{\text{Function Regularization}} .$$

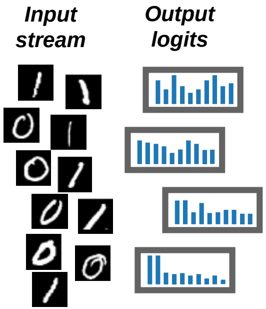
Function Regularization

MSE current responses vs
those stored in the memory buffer.

The logits stored into the memory buffer are not just proxies for the ground-truth labels, but have a **deeper meaning**

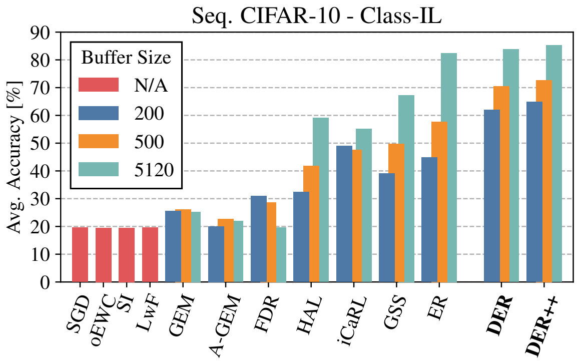
Secondary information: logits are more informative than labels, as they encode visual similarities and the relations between classes.

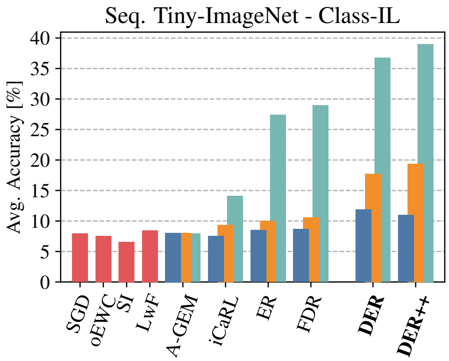
- Therefore, they carry out a more insightful signal regarding past tasks.



Experimental results

Compared to existing approaches, ER, DER and DER++ achieve strong performance, despite their simplicity.





Other rehearsal approaches

Other common rehearsal approaches.

- Gradient Episodic Memory (GEM) and its lightweight variant Average-GEM (A-GEM)
- Meta Experience Replay (MER)
- Incremental Classifier and Representation Learning (iCaRL)
- Learning a Unified Classifier Incrementally via Rebalancing (LUCIR)

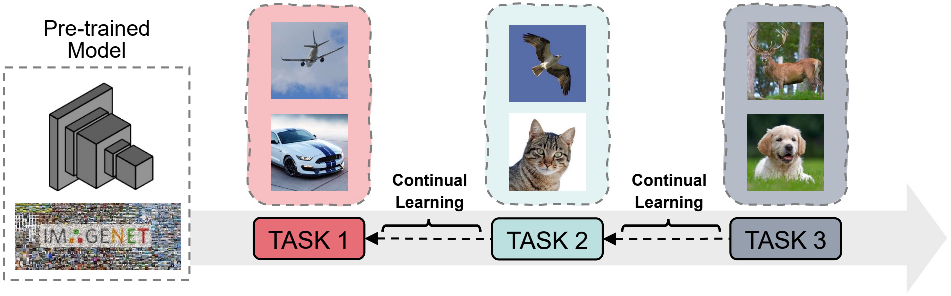
Approaches tailored to Vision Transformers (VIT):

- Learning-to-prompt (L2P)

Continual with Pre-trained models

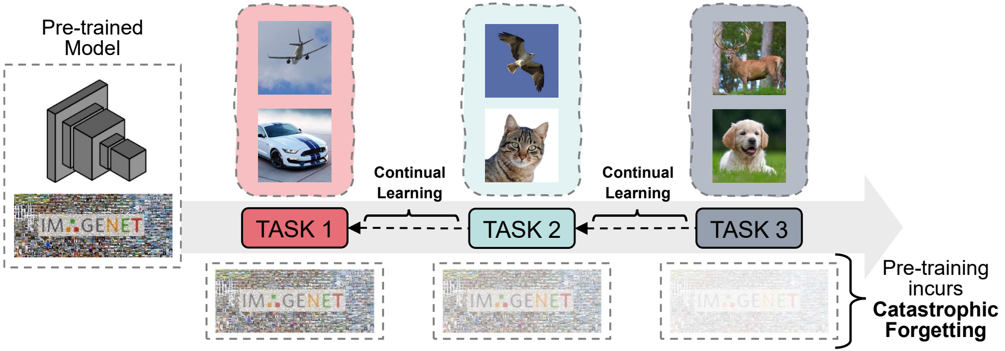
Why using Pre-train?

Pre-Training is commonly employed in the vast majority of scenarios.

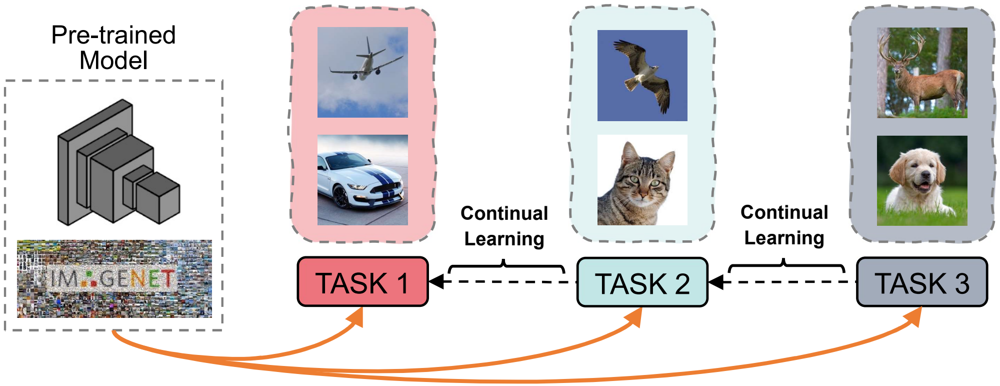


Pre-training & Catastrophic Forgetting

- … due to forgetting it is effective ***only*** in the **first task**!
- CL does not protect pre-training from Forgetting



Pre-training & Catastrophic Forgetting



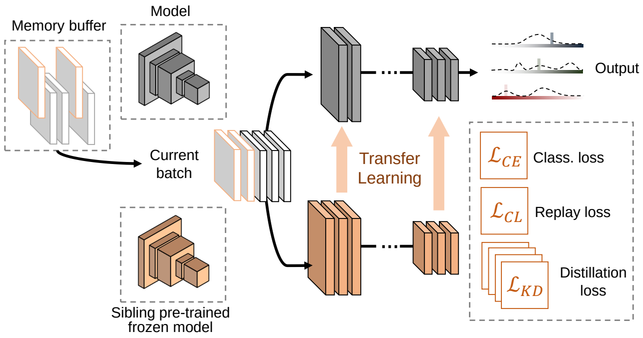
- The objective is to use (*transfer*) its knowledge to the following tasks
- No need to maintain the ability to solve the pre-training task

The Pre-train is not an additional task

Transfer without Forgetting

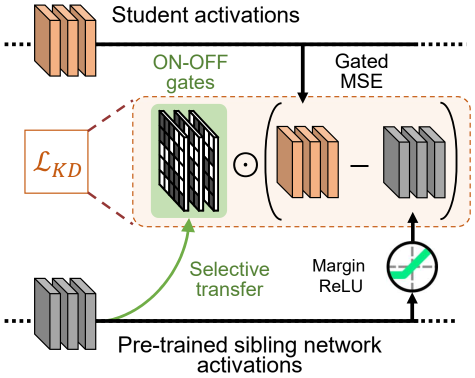
Idea: distill knowledge from a **frozen** pre-trained sibling network.

Multilevel Knowledge Distillation: features from the sibling are *adapted* by small learned modules.



Memory buffer
Model
Current batch
Output
Transfer Learning
$\mathcal{L}_{CE}$ Class. loss
$\mathcal{L}_{CL}$ Replay loss
$\mathcal{L}_{KD}$ Distillation loss
Sibling pre-trained frozen model

Transfer without Forgetting



- **Feature Distillation**: match in-training and pre-trained activations.

$$\sum_{l=1}^{L} ||\mathbb{M}(\hat{h}^{(l)}; t) \odot (f^{(l)} - \mathsf{ReLU}_m(\hat{h}^{(l)}))||_2^2$$

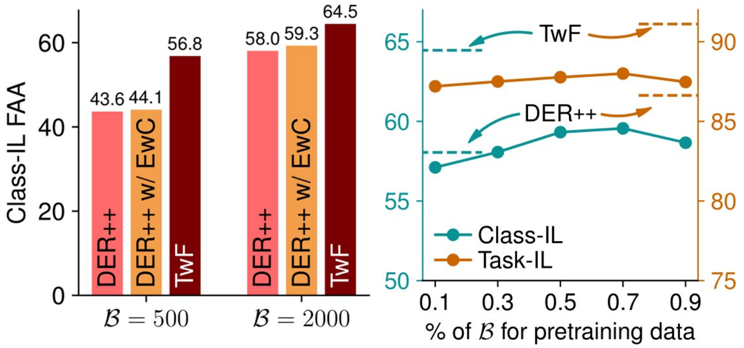
- Attention *gates* (spatial- and channel-wise) **modulate** the transfer.

$$\mathbb{M}(\hat{h}^{(l)}) \triangleq \mathsf{gumbel}(\mathbb{M}_{\mathsf{Ch}}(\hat{h}^{(l)}) + \mathbb{M}_{\mathsf{Sp}}(\hat{h}^{(l)}))$$

Attention maps are also **replayed** during CL.

Comparison with EwC and Replay

- Using **EwC** on pre-training weights (left)
- **DER++** also replaying pre-training data (right)

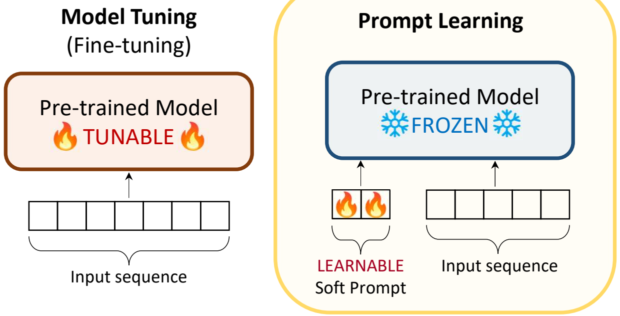


Both effective, but less so than **TwF**.

Other transfer learning strategies – Transformers & CL

Transformer-based architectures require **extensive pre-train** to achieve SOTA

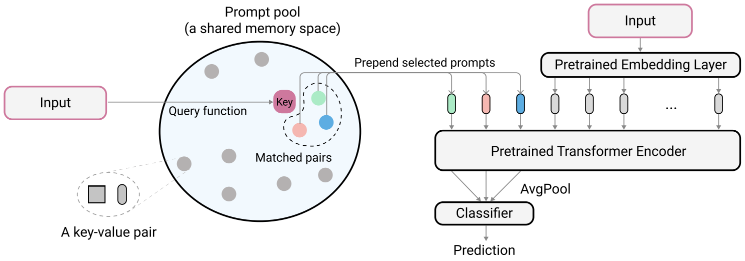
- **Prompting**: a transfer learning technique originated in the field of Natural Language Processing
- With **prompting** we leave the pre-trained knowledge intact and only train a few additional parameters



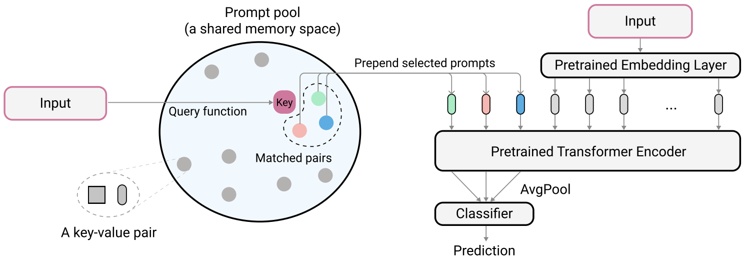
Learning to Prompt for Continual Learning

… training all the prompts in all tasks would incur forgetting!

- **Learning to Prompt**: Select a subset of prompts depending on the current input
- Ideally, inputs that *share information* should also *share prompts*



Learning to Prompt for Continual Learning



- Each prompt P_i is associated to a learnable **key** k_i: $\{(P_i, k_i)\}_{i=1}^{N}$
- Introduce a **query** function $q(x)$ and select the M prompts whose key best matches the query: $K_x = \text{argmin}_{\{s_i\}_{i=1}^{N} \subseteq [1,M]} \sum_{i=1}^{N} \text{diff}(q(x), k_{s_i})$
- Directly use the pre-trained frozen model to get the query: $q(x) = f_0(x)$
- *Replay free!*

Learning to Prompt for Continual Learning

More elaborate prompting mechanisms are possible

- DualPrompt, CODA-Prompt, S-Prompt to name a few.
- We can adapt even Large Pretrained Models (*e.g.*, CLIP) – see Prompt-Fusion, AttriCLIP

… *but is it the best we can do?*

- Some preliminary works suggest that fine-tuning the whole model can achieve higher performance *even in continual*

Credits

Credits i

These slides heavily borrow from a number of awesome sources. I'm really grateful to all the people who take the time to share their knowledge on this subject with others.

In particular:

- De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., ... & Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence, 44(7), 3366-3385.

Credits ii

- Wang, L., Zhang, X., Su, H., & Zhu, J. (2023). A comprehensive survey of continual learning: Theory, method and application. arXiv preprint arXiv:2302.00487.

References i

- Barnes, J. M., & Underwood, B. J. (1959). ” Fate” of first-list associations in transfer theory. Journal of experimental psychology, 58(2), 97.
- McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation (Vol. 24, pp. 109-165). Academic Press.
- Van de Ven, G. M., & Tolias, A. S. (2019). Three scenarios for continual learning. arXiv preprint arXiv:1904.07734.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13), 3521-3526.

References ii

- Lopez-Paz, D., & Ranzato, M. A. (2017). Gradient episodic memory for continual learning. Advances in neural information processing systems, 30.
- Nguyen, C. V., Li, Y., Bui, T. D., & Turner, R. E. Variational Continual Learning. In International Conference on Learning Representations.
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., ... & Hadsell, R. (2016). Progressive neural networks. arXiv preprint arXiv:1606.04671.
- Mallya, A., & Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 7765-7773).

References iii

- Serra, J., Suris, D., Miron, M., & Karatzoglou, A. (2018, July). Overcoming catastrophic forgetting with hard attention to the task. In International Conference on Machine Learning (pp. 4548-4557). PMLR.
- Mallya, A., Davis, D., & Lazebnik, S. (2018). Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 67-82).
- Li, Z., & Hoiem, D. (2017). Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence, 40(12), 2935-2947.
- Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., & Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. arXiv preprint arXiv:1810.11910.

References iv

- Buzzega, P., Boschini, M., Porrello, A., Abati, D., & Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems, 33, 15920-15930.
- Rebuffi, S. A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 2001-2010).
- Aljundi, R., Lin, M., Goujaud, B., & Bengio, Y. (2019). Gradient based sample selection for online continual learning. Advances in neural information processing systems, 32.

References v

- Hou, S., Pan, X., Loy, C. C., Wang, Z., & Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (pp. 831-839).
- Wang, Z., Zhang, Z., Lee, C. Y., Zhang, H., Sun, R., Ren, X., ... & Pfister, T. (2022). Learning to prompt for continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 139-149).
- Boschini, M., Bonicelli, L., Porrello, A., Bellitto, G., Pennisi, M., Palazzo, S., ... & Calderara, S. (2022, October). Transfer without forgetting. In European Conference on Computer Vision (pp. 692-709).

References vi

- Z., Gengwei, W., Liyuan, K., Guoliang, C., Ling, & W., Yunchao. SLCA: Slow Learner with Classifier Alignment for continual learning on a pre-trained model. In Proceedings of the IEEE/CVF International Conference on Computer Vision.
- Smith, J. S., Tian, J., Halbe, S., Hsu, Y. C., & Kira, Z. (2023). A closer look at rehearsal-free continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2409-2419).