Deep Generative View on Continual Learning

Tomasz Trzcinski

Credits
Karol Piczak
Vincenzo Lomonaco

About me

Tomasz Trzcinski, Ph.D. D.Sc.

Associate Professor at Warsaw University of Technology

Head of *Computer Vision Lab*

Associate Professor at Jagiellonian University of Cracow

Team Leader of InfoTech group in FNP-funded BioNN grant

Member of Group of Machine Learning Research (GMUM)

Chief Scientist at Tooploox

Contact details:

E-mail: tomasz.trzcinski@pw.edu.pl www: http://ii.pw.edu.pl/~ttrzcins

Research interests:

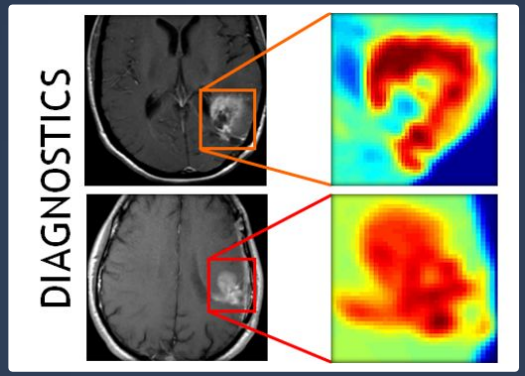
Efficient machine learning in computer vision: continual learning, representation learning, conditional computation, generative modeling

Ongoing research projects: *Deep generative view on continual learning* (OPUS-20)

Continual self-supervised representation learning (PRELUDIUM BIS-3)

National Science Centre

Machine learning eats the world



But what if the world changes?

Let's retrain from scratch, but... is it sustainable?

How does the nature solve it?

Lifelong learning of living organisms

Newborns naturally acquire knowledge without forgetting.

Short-term memory

Hippocampus as a buffer of compressed short-term memories.

Allows for fast incorporation of new knowledge.

Memory consolidation

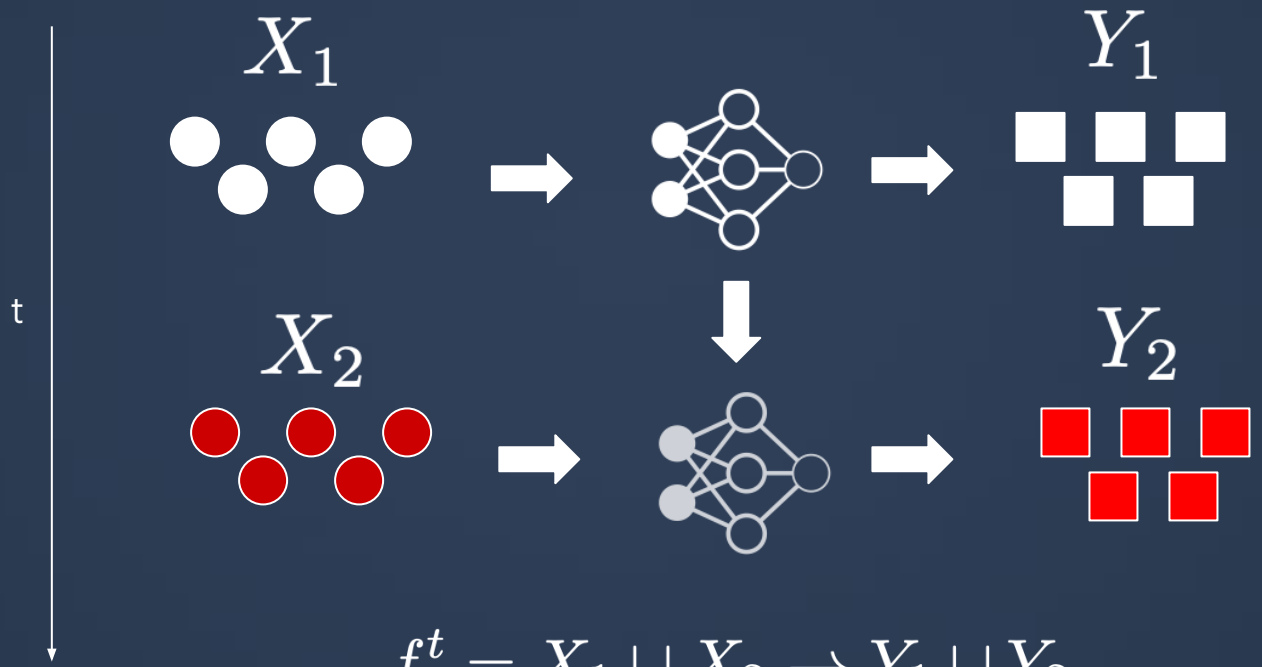
Consolidation happens in cerebral cortex.

Sleep is a critical time when the consolidation takes place.

Continual learning

$$f_\theta : X_1 \rightarrow Y_1$$

Continual learning



$$f_\theta^t = X_1 \cup X_2 \rightarrow Y_1 \cup Y_2$$

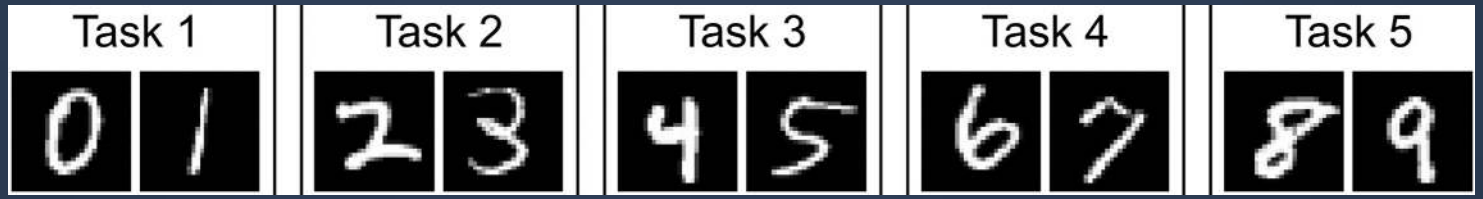
Definition

General

A machine learning paradigm that assumes model training based on **infinite stream of data** as a **sequence of experiences** in order to **acquire new knowledge for future use** (Chen & Liu, 2018).

In practice

We train a **single** model **sequentially** on tasks with ever-changing distribution of classes and input data, **without the possibility to get back to previous tasks.**



Why do we care?

Long-term goal:

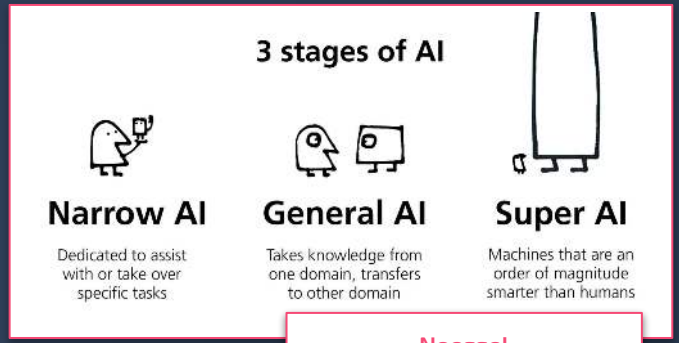
Continual learning is a cornerstone of Artificial General Intelligence (AGI).

Current applications:

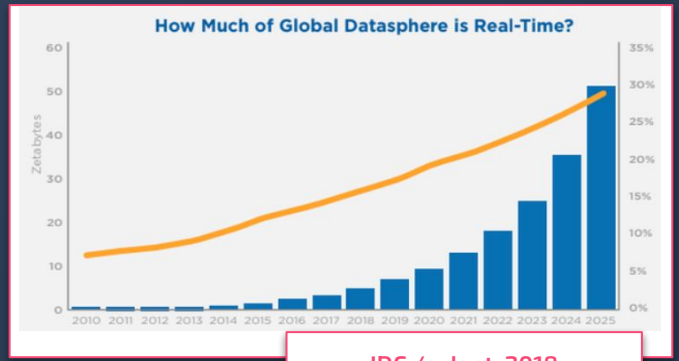
data-driven model update - we would like to include new data in the model without costly and lengthy model retraining *e.g.* detection of new virus strain,

lack of access to previously seen data - privacy, model running on the end device (robot, etc.),

real-time data supply - no time for storage, only stream processing and model training possible, *e.g.* CERN after upgrade



Noessel



IDC / zdnet, 2018

Continual learning requirements

Lack of forgetting

Learning new tasks cannot lead to performance loss on previous tasks

Constant memory and computational budget

Comparing method is fair only within a given computational constraints. Worst case, budget should scale linearly with the number of tasks.

Access to previously seen data prohibited

Worst case, limited-size buffer for previously seen samples available.

Forward transfer

Knowledge acquired on previous tasks should simplify solving future ones.

Backward transfer

Knowledge about the new tasks improves the performance on past ones.

Learning scenarios

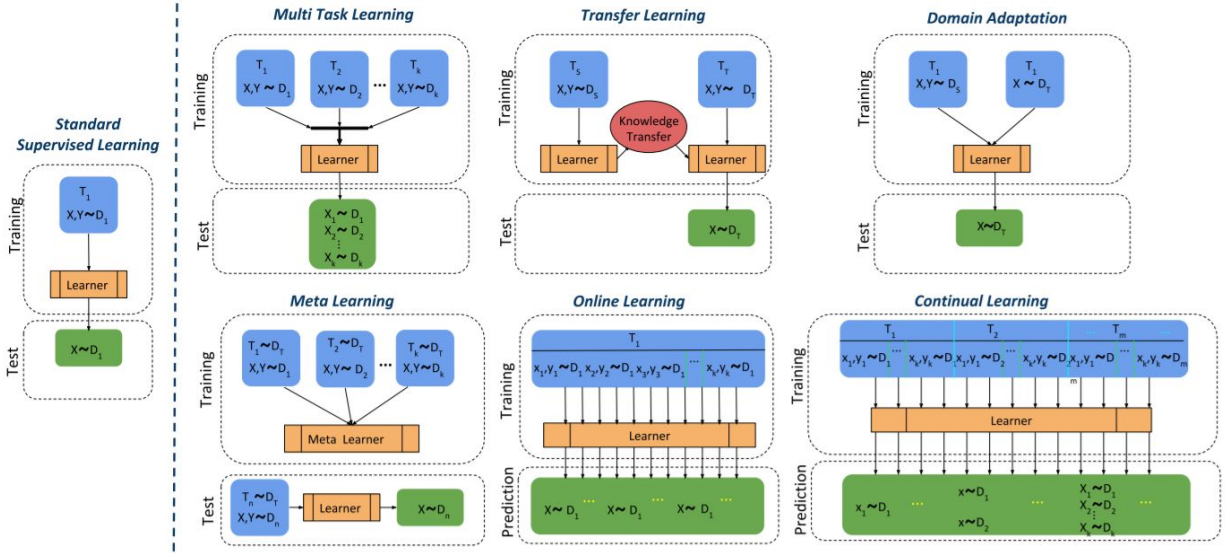


Fig. 5: The main setup of each related machine learning field, illustrating the differences with general continual learning settings.

Lange et al., 2019

Challenges of continual learning

Main challenge of continual learning

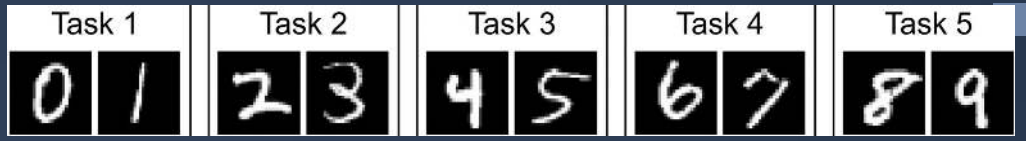
Catastrophic forgetting

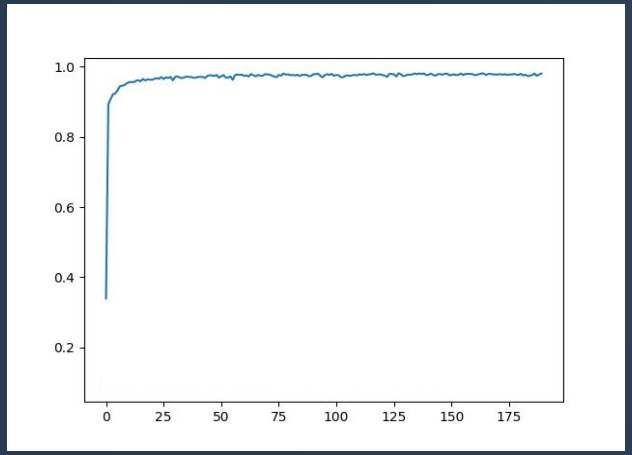
forgetting - training model for new tasks leads to the reduction of **accuracy** on previous tasks
catastrophic - forgetting is an **abrupt** (not gradual), usually very fast and wide decrease of ability to solve for previous tasks

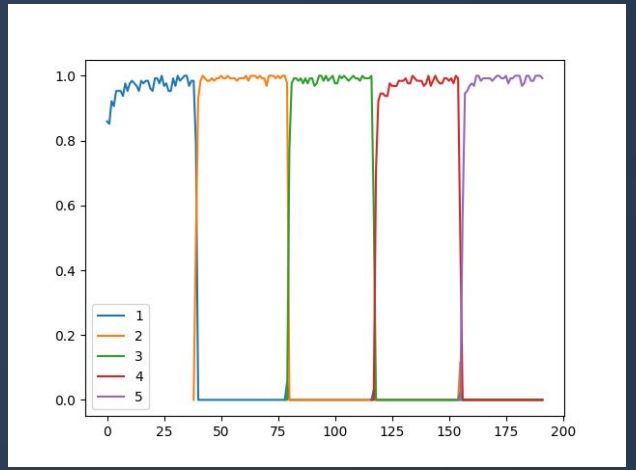
The problem occurs when the model is trained on a **stream** of data that is not stationary and cannot be randomly sampled to ensure IID (independent and identically distributed) condition.

Even for the standard learning, we can drop the random sampling to observe forgetting.

Main challenge of continual learning

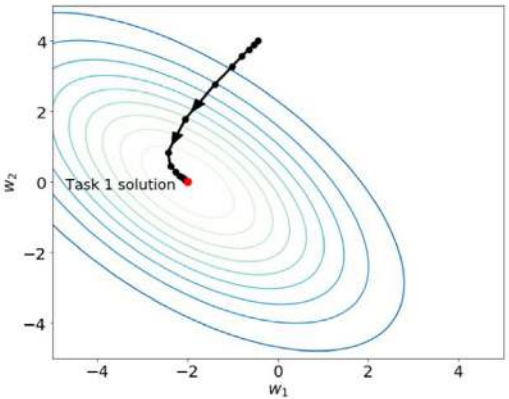


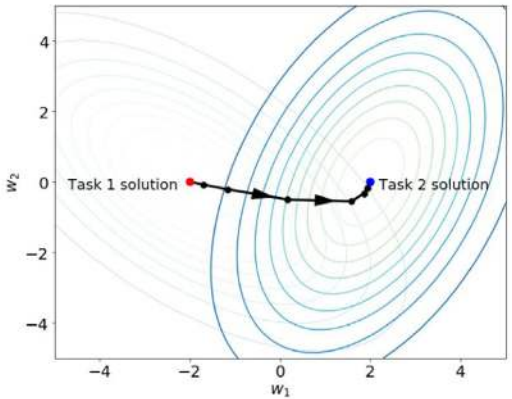


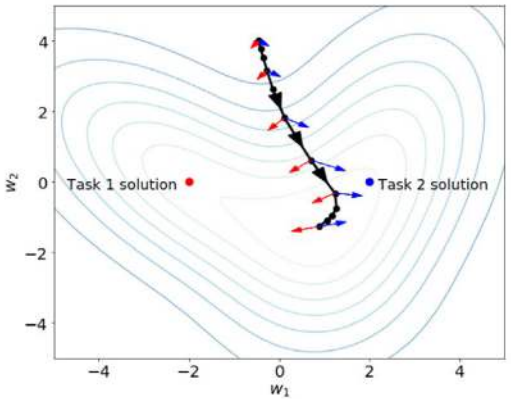


Catastrophic forgetting vs gradient descent

Forgetting is a fundamental issue linked with gradient optimisation which is driven by local changes.



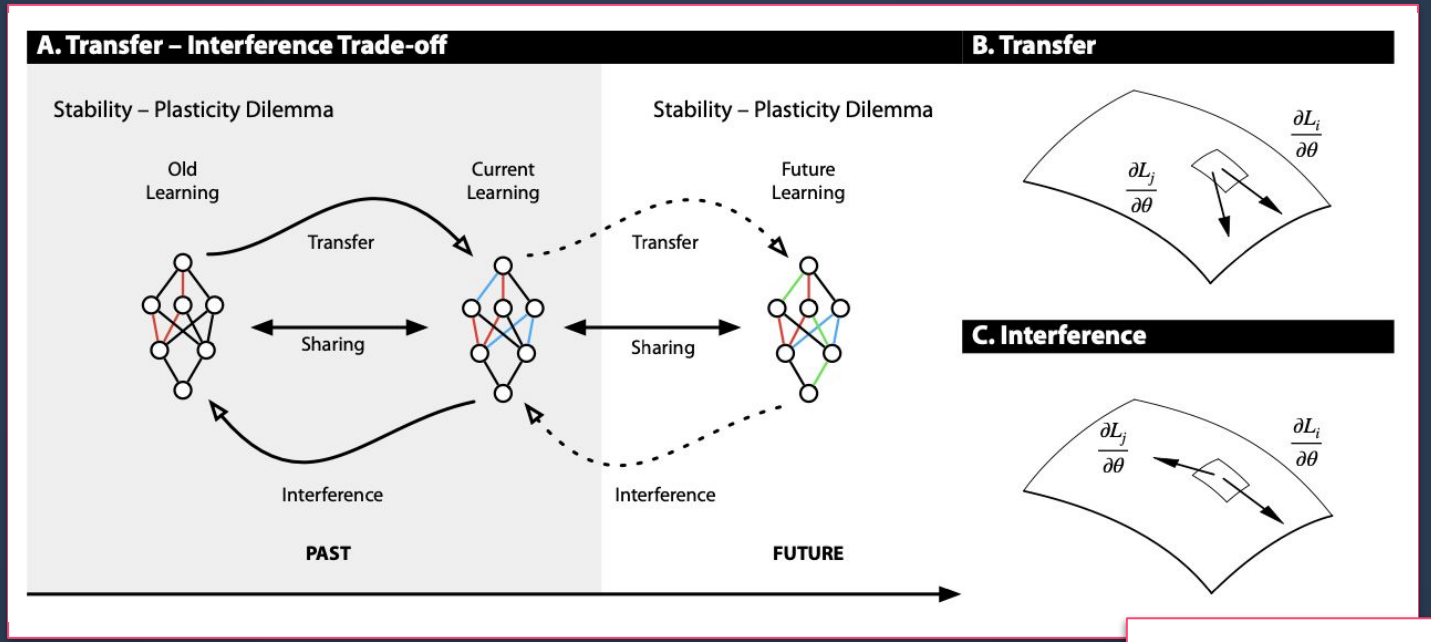




Hadsell et al., 2020

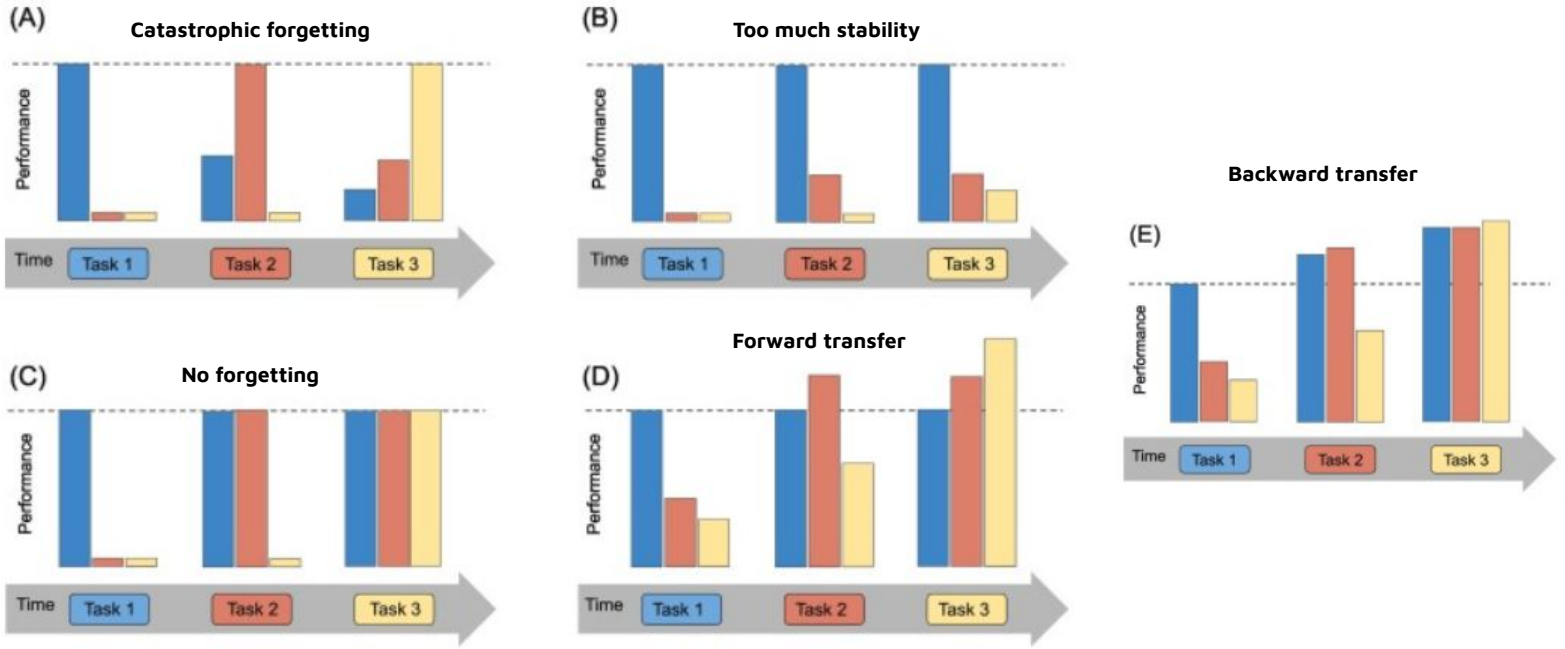
Stability-plasticity dilemma

Also known as a transfer-interference trade-off



Riemer & Cases, 2019

Possible results of continual learning



Hadsell et al., 2020

The easiest way to solve forgetting

Full *replay*

new examples are added to the dataset,
every once in a while we train on an entire dataset (fine-tuning) for a few epochs or longer,
alternatively we increase the frequency of retraining.

What are the downsides?

Are we sure the results are equivalent?

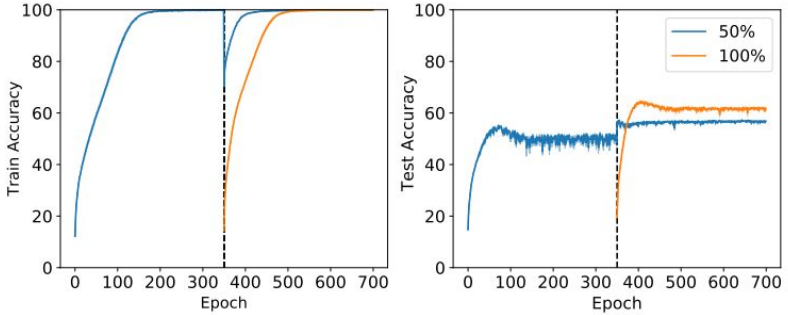
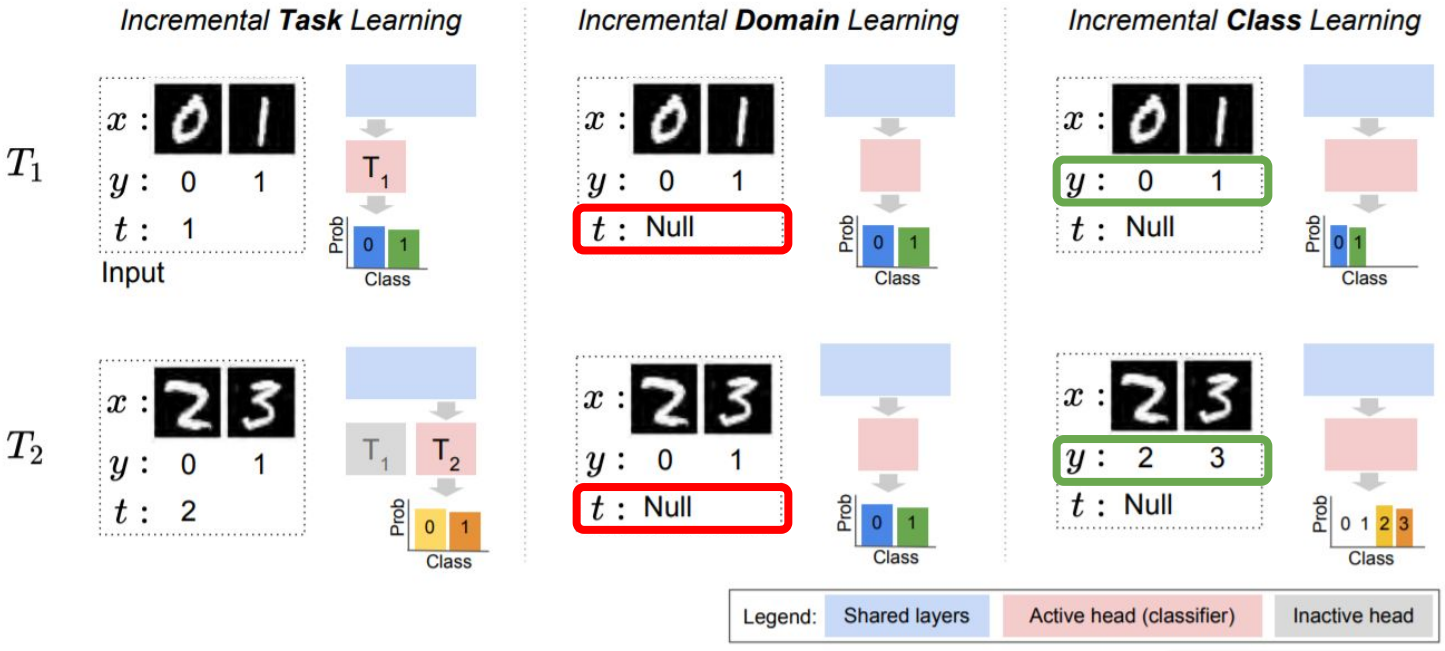


Figure 1: A comparison between ResNets trained using a warm start and a random initialization on CIFAR-10. Blue lines are models trained on 50% of CIFAR-10 for 350 epochs then trained on 100% of the data for a further 350 epochs. Orange lines are models trained on 100% of the data from the start. The two procedures produce similar training performance but differing test performance.

Ash & Adams, 2019

Continual learning scenarios

Continual learning scenarios



Hsu et al., 2018

Difficulty of *class-incremental learning*

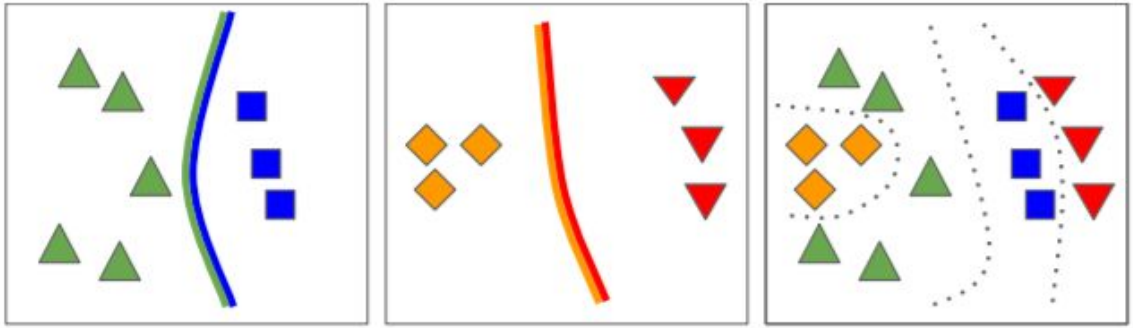


Fig. 1: A network trained continually to discriminate between task 1 (left) and task 2 (middle) is unlikely to have learned features to discriminate between the four classes (right). We call this problem *inter-task confusion*.

Masana et al., 2020

Is it only about supervised learning?

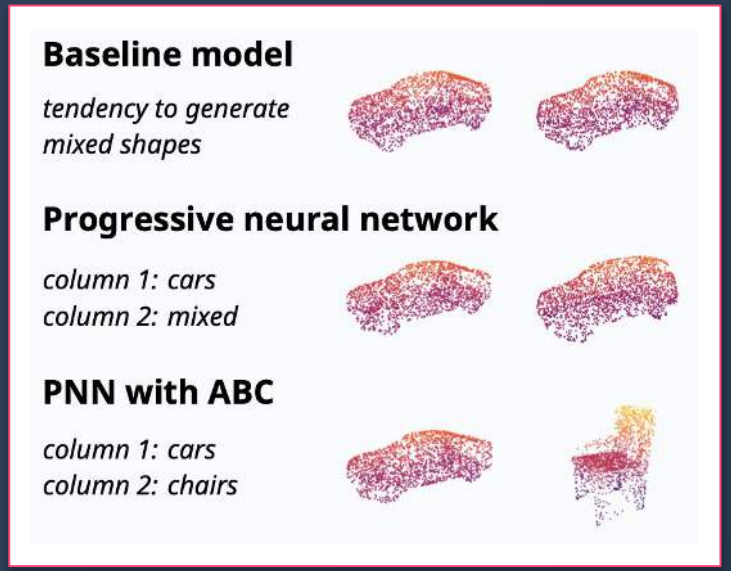
Sadowski et al.'21

Continual learning focuses on supervised learning

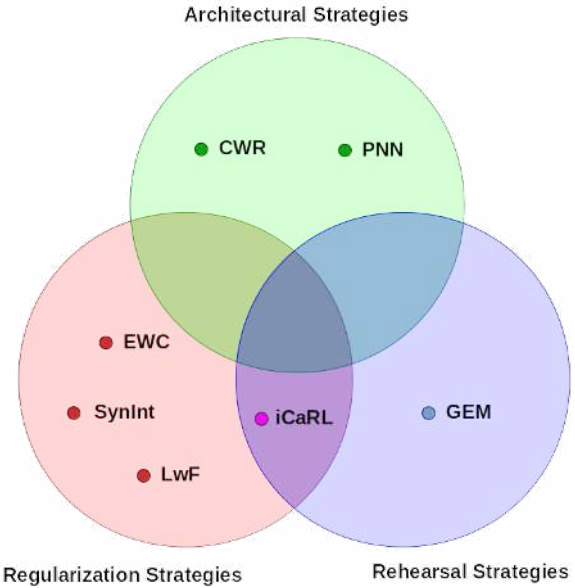
How about CL in generative modeling? Are there any works on that? **Yes** - e.g. generative models.

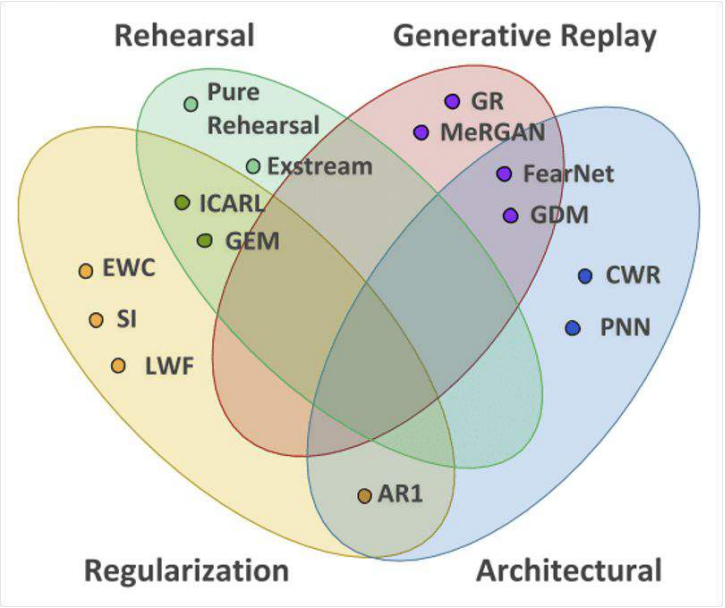
To provide diverse outputs of generative models, we **uniformly sample columns of a progressive neural network** (PNN).

We extend it with **autonomous branch construction** (ABC) - mapping new training examples to proper branches of the model based on the reconstruction loss.



Taxonomy of continual learning methods





Lomonaco

Regularization methods

L2 regularization

We train a model on the **first** task.

In the next tasks, we **add regularization** such that the weights after updates will not change dramatically with respect to those previously trained.

This mechanism is often used to keep the weights close to zero (or avoid their unnecessary increase), while here we **anchor the weights** of the updated model to the weights previously computed.

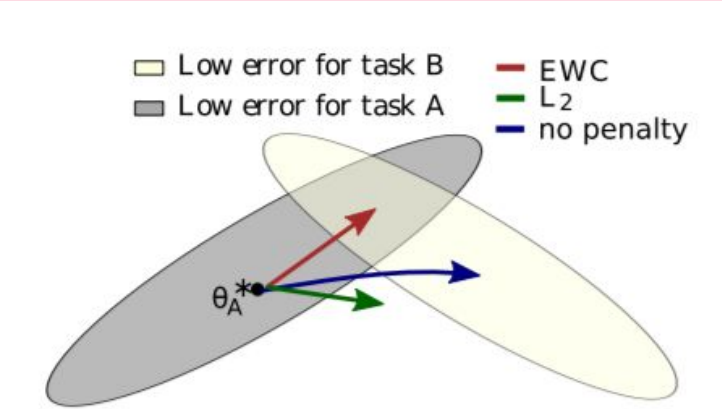
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2} \sum \left(\theta_i - \theta^*_{A,i} \right)^2$$

Each weight is considered **equally important** - how about measuring the importance of weights to a given task instead?

Elastic Weight Consolidation

Key aspects

- **Seminal work** that sparked a lot attention around deep continual learning
- Interesting connection with some neuroscientific works on **memory consolidation**
- The main intuition is that we **don't want to change weights relevant for the old tasks**
- How do we estimate that? With **Fisher Information** that measures how much information about model weights is conveyed by the data.



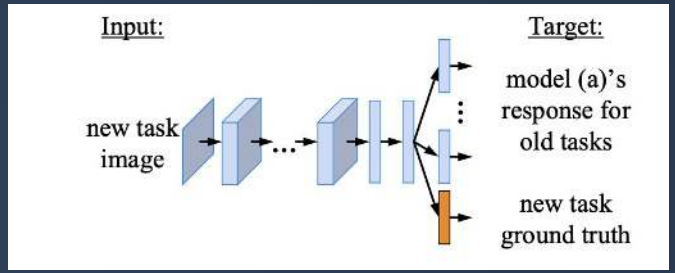
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

Kirkpatrick et al., 2016

Learning without Forgetting

Key aspects

- Regularization based on **activations**, instead of model weights
- Straightforward application of **knowledge distillation**: to the new data loss, one adds distillation loss that aims to maintain predictions of the model on **new data**
- Originally for task-incremental, but **can be extended to other CL scenarios**
- **Easy to implement**



LEARNINGWITHOUTFORGETTING:
Start with:
θ_s: shared parameters
θ_o: task specific parameters for each old task
X_n, Y_n: training data and ground truth on the new task
Initialize:
$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*
$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*
Train:
Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*
Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*
$\theta_s^*, \theta_o^*, \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\text{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$

Li & Hoiem, 2016

Continual Learning with hypernetworks

Key aspects

- Main idea: let's learn how to **generate networks weights**
- **Condition** weights on task-dependent embeddings
- Underlying hypothesis: learning in the **compressed** space is less prone to forgetting

Von Oswald'20

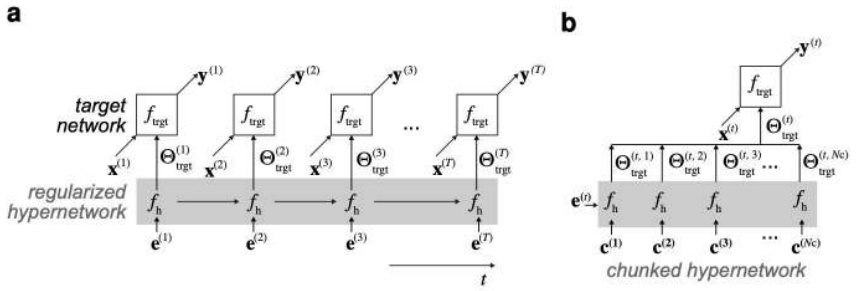


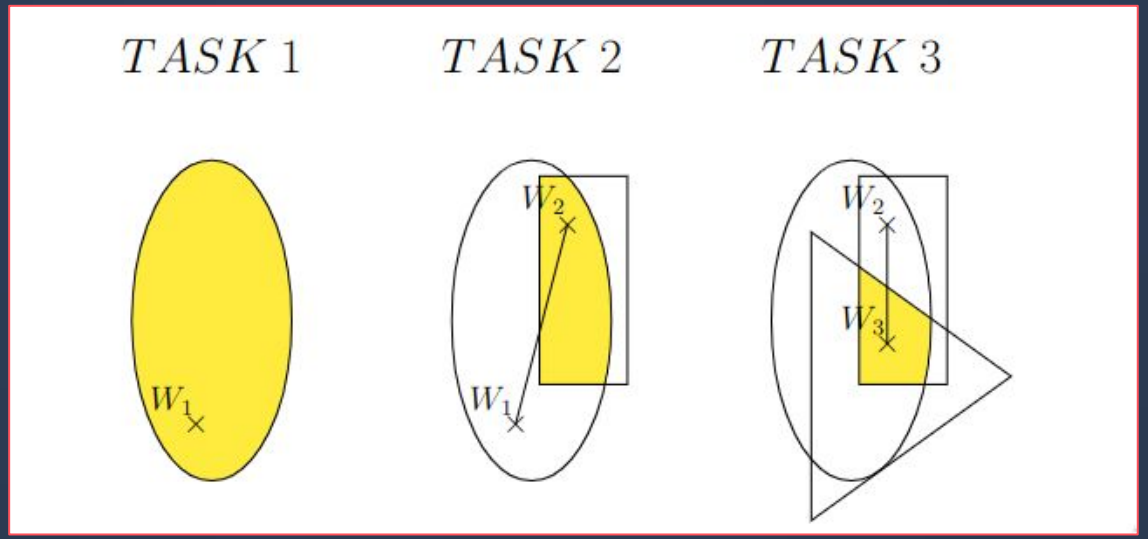
Figure 1: **Task-conditioned hypernetworks for continual learning.** **(a)** Commonly, the parameters of a neural network are directly adjusted from data to solve a task. Here, a weight generator termed *hypernetwork* is learned instead. Hypernetworks map embedding vectors to weights, which parameterize a target neural network. In a continual learning scenario, a set of task-specific embeddings is learned via backpropagation. Embedding vectors provide task-dependent context and bias the hypernetwork to particular solutions. **(b)** A smaller, chunked hypernetwork can be used iteratively, producing a chunk of target network weights at a time (e.g., one layer at a time). Chunked hypernetworks can achieve model compression: the effective number of trainable parameters can be smaller than the number of target network weights.

InterContiNet

Wołczyk et al.'22

NP-hard problem

- EWC's approach to finding overlapping parameter space regions is NP-hard as the regions get highly irregular

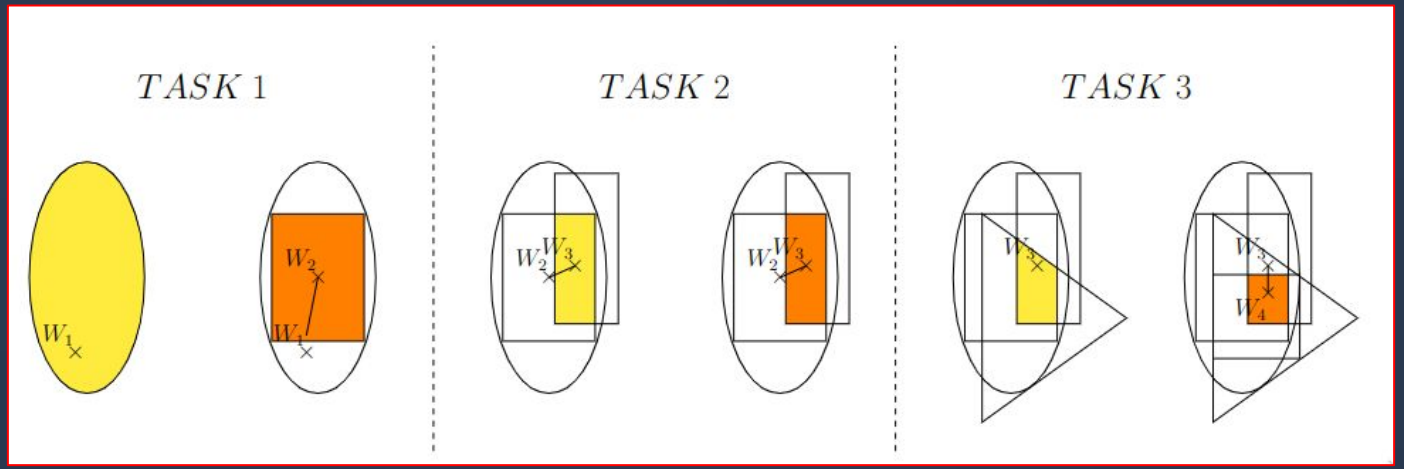


InterContiNet

Wołczyk et al.'22

Simplify setting, add assumptions

- Constrain the parameter space search to allow hyperrectangles only.
- Finding intersections between hyperrectangles can be done in polynomial time.
- We can use interval arithmetic to implement this search.



InterContiNet

Wołczyk et al.'22

Interval arithmetic

Interval arithmetic (Dahlquist & Björck, 2008)(Chapter 2.5.3) is based on the operations on segments. Let us assume A and B are numbers expressed as intervals. For all $\bar{a}, \underline{a}, \bar{b}, \underline{b} \in \mathbb{R}$ where $A = [\underline{a}, \bar{a}]$, $B = [\underline{b}, \bar{b}]$, we can define operations such as (Lee, 2004):

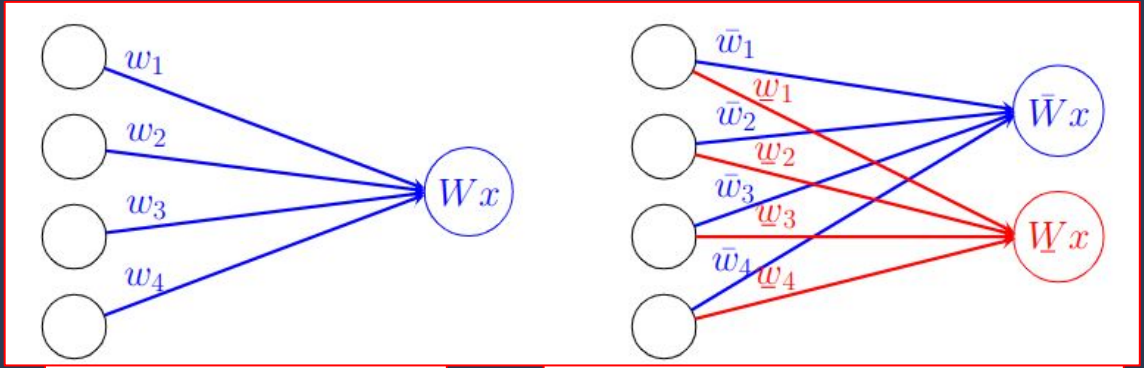
- addition: $[\underline{a}, \bar{a}] + [\underline{b}, \bar{b}] = [\underline{a} + \underline{b}, \bar{a} + \bar{b}]$
- multiplication: $[\underline{a}, \bar{a}] * [\underline{b}, \bar{b}] = [\min(\underline{a} * \underline{b}, \underline{a} * \bar{b}, \bar{a} * \underline{b}, \bar{a} * \bar{b}), \max(\underline{a} * \underline{b}, \underline{a} * \bar{b}, \bar{a} * \underline{b}, \bar{a} * \bar{b})]$

InterContiNet

Wołczyk et al.'22

Interval arithmetic

$$[\underline{W}_k, \bar{W}_k] = [W_k - \varepsilon_k, W_k + \varepsilon_k]$$



$$h(x) = Wx + b.$$

$$[\underline{h}(x), \bar{h}(x)] = [\underline{W}, \bar{W}][x, x].$$

InterContiNet - results

Wołczyk et al.'22

Table 2. The average accuracy across all five tasks of the split FashionMNIST protocol, evaluated after learning the whole sequence. Each value is the average of five runs (with standard deviations).

Method	Incremental task	Incremental domain	Incremental class
SGD	92.22 ± 3.06	82.77 ± 0.44	19.91 ± 0.01
Adam	88.13 ± 5.64	78.48 ± 0.47	19.96 ± 0.01
L2	97.36 ± 0.17	**92.65 ± 0.09**	26.91 ± 1.23
EWC	97.53 ± 0.15	92.12 ± 0.18	19.90 ± 0.01
oEWC	96.70 ± 0.57	88.83 ± 0.30	19.87 ± 0.02
SI	97.00 ± 0.25	91.45 ± 0.07	19.97 ± 0.34
MAS	97.43 ± 0.14	91.74 ± 0.19	10.00 ± 0.00
LwF	98.10 ± 0.07	88.63 ± 0.12	**39.51 ± 1.45**
InterContiNet	**98.37 ± 0.06**	**92.65 ± 0.40**	35.11 ± 0.02
Offline	97.98 ± 0.05	96.39 ± 0.06	82.54 ± 0.13

Table 3. The average accuracy across all five tasks of Split CIFAR-10, evaluated after training AlexNet on the whole sequence. Each value is the average of five runs (with standard deviations).

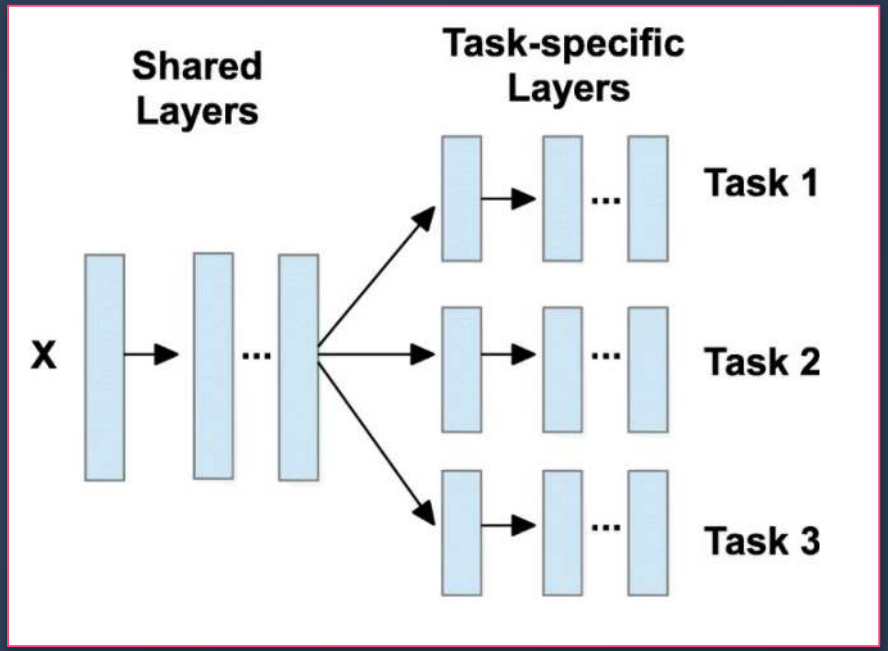
Method	Incremental task	Incremental domain	Incremental class
SGD	64.74 ± 8.53	69.22 ± 0.55	15.56 ± 5.07
EWC	67.33 ± 7.11	70.26 ± 0.66	11.83 ± 4.10
oEWC	64.59 ± 7.97	65.97 ± 8.97	15.49 ± 5.01
SI	67.26 ± 7.77	69.69 ± 0.73	17.38 ± 4.13
MAS	66.20 ± 9.29	72.54 ± 0.59	11.79 ± 4.01
LwF	**93.03** ± 0.28	**76.97** ± 0.91	13.89 ± 5.33
InterContiNet	72.64 ± 1.18	69.48 ± 1.36	**19.07** ± 0.15
Offline	93.11 ± 0.18	90.54 ± 0.32	82.72 ± 0.09

Architectural methods

Multi-head Architectures

Key elements

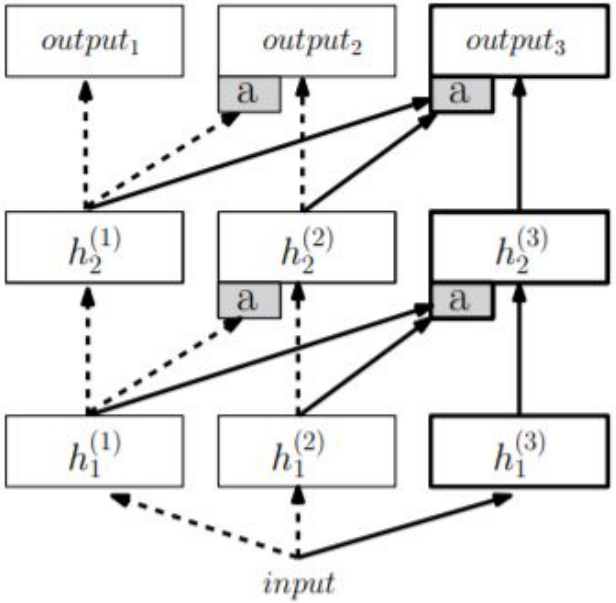
- Great to specialize behaviours **if we know the boundaries between the tasks**
- Clear split between **shared** and **task-specific parameters**
- A new head **per experience** is quite inefficient and possibly ineffective



Progressive Neural Networks

Key aspects

- Main focus on **forward transfer** and reuse of previously acquired knowledge
- Previous **"columns" (known from neuroscience) frozen** preventing backward transfer
- Quite inefficient: **significant increase in the number of parameters**, difficult to scale on longer sequences of experiences
- **Adapters** (dim. reduction, 1x1 convolutions) and **pruning** can reduce complexity
- **Task ID** must be known

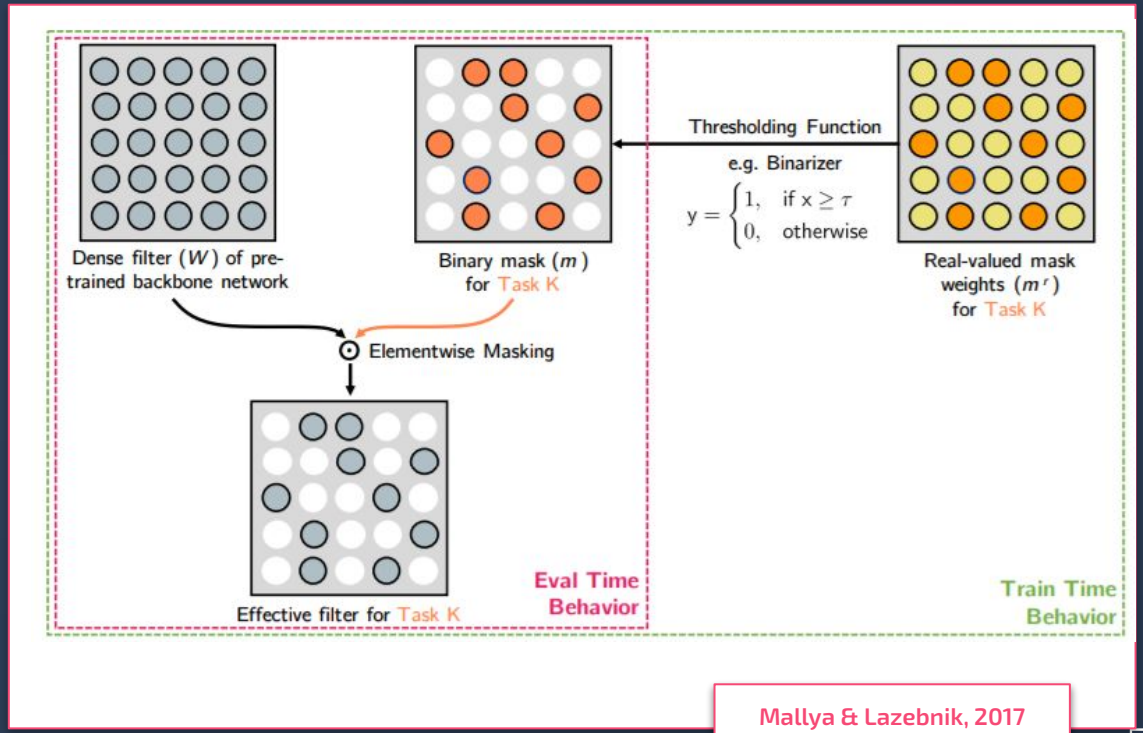


Rusu et al., 2016

Piggyback - masking weights

Key aspects

- Start from a **pre-trained model** (backbone)
- Mask all weights than train float to **binarize** which part of the network to be used.
- Zero forgetting, but **no knowledge transfer**
- **Very efficient**, but requires task labels.

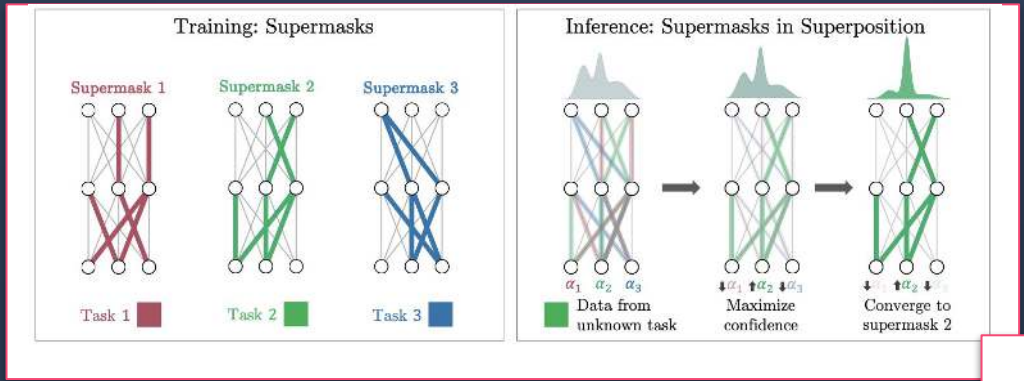


Mallya & Lazebnik, 2017

Supermasks in superposition

Key aspects

- Good binary masks are enough to obtain decent performance **even on random weights.**
- **Random weight matrices** can be created out of a random seed which leads to significant compression.
- They can be used in **superposition** to infer task ID



Wortsman et al.'20

Replay methods

Random replay

Basic approach

- **Sample randomly** from the current experience data
- Fill your fixe **Random Memory (RM)**
- **Replace examples randomly** to maintain approximately equal number of examples for experience

Algorithm 1 Pseudocode explaining how the external memory RM is populated across the training batches. Note that the amount h of patterns to add progressively decreases to maintain a nearly balanced contribution from the different training batches, but no constraints are enforced to achieve a class-balancing.

$RM = \varnothing$
RM_{size} = number of patterns to be stored in RM
for each training batch B_i:
  train the model on shuffled $B_i \cup RM$
  $h = \frac{RM_{size}}{i}$
  R_{add} = random sampling h patterns from B_i
  $R_{replace} = \begin{cases} \varnothing & \text{if } i == 1 \\ \text{random sample } h \text{ patterns from } RM & \text{otherwise} \end{cases}$
  $RM = (RM - R_{replace}) \cup R_{add}$

Pellegrini et al.'19

GDUMB: Control Baseline

Greedy Sampler and Dumb Learner

- Samples stored in a buffer greedily (to balance classes) and a learner trained **only on a buffer (!)**
- Note that there's **no knowledge transfer** in this strategy (quite dumb indeed!)
- Despite its simplicity, it was shown to work **better than some existing and more complex strategies**, questioning the utility of some benchmarks/metrics in the field
- **Conclusion:** If your cannot beat GDumb there's something wrong in your strategy or evaluation

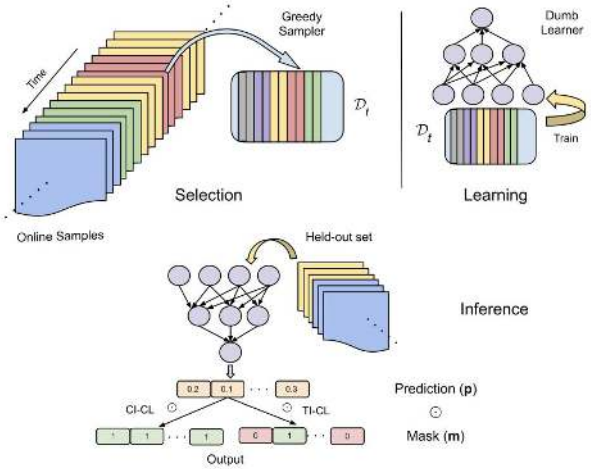


Fig. 1. Our approach (GDumb): The sampler greedily stores samples while balancing the classes. When asked, the learner trains a network from scratch on memory $\mathcal{D}_t$ provided by the sampler. If a mask $\mathbf{m}$ is given at inference, GDumb classifies on the subset of labels provided by the mask. Depending on the mask, GDumb's inference can vary between two extremes: CI (class-incremental) and TI (task-incremental) formulations.

Prabhu et al.'20

iCaRL

iCaRL: Incremental Classifier and Representation Learning

One of the first *replay methods* (Rebuffi et al., 2016).
It selects and stores representative samples (*exemplars*) closest to the average network response for a given class.

Classification based on the *nearest-mean-of-exemplars* (*prototypes* - the idea later popularized by *prototypical networks* - Snell et al., 2017).

Loss function for new classes minimized together with the distillation loss between the responses of old and new model for old classes (LwF).

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \dots, X^t$ // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \dots, s-1$ **do**
  $q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i)) + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y) \log(1-g_y(x_i)) \Big]$$

that consists of *classification* and *distillation* terms.

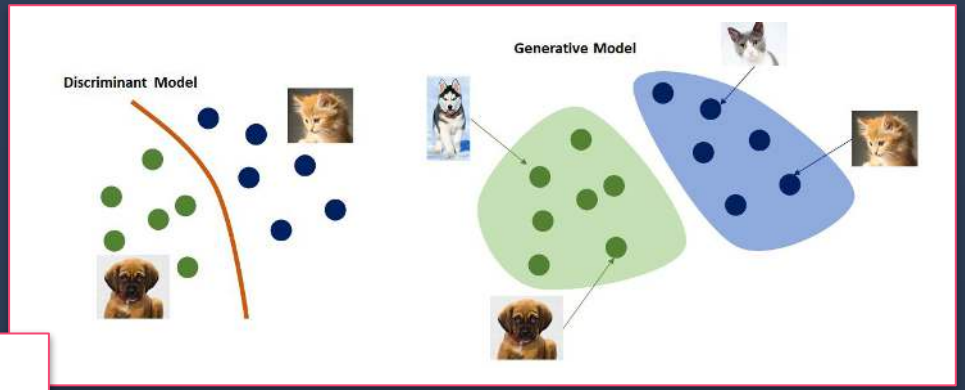
Rebuffi et al.'17

Generative models for the rescue

- Instead of a buffer, we can train a GAN/VAE
- They can generate new samples from samples coming from a predefined probability distribution
- Deep **generative look** on continual learning

- **Our research hypothesis**:

  Generative models are more robust to catastrophic forgetting since their goal is to represent the data, not to discriminate between instances.



Masarczyk et al.'21

Generative rehearsal methods

Deep generative replay

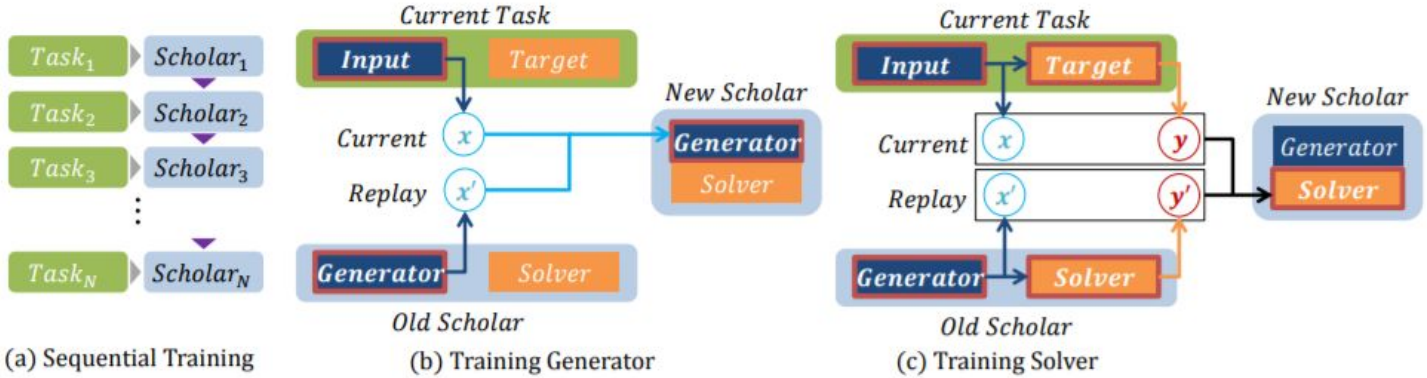


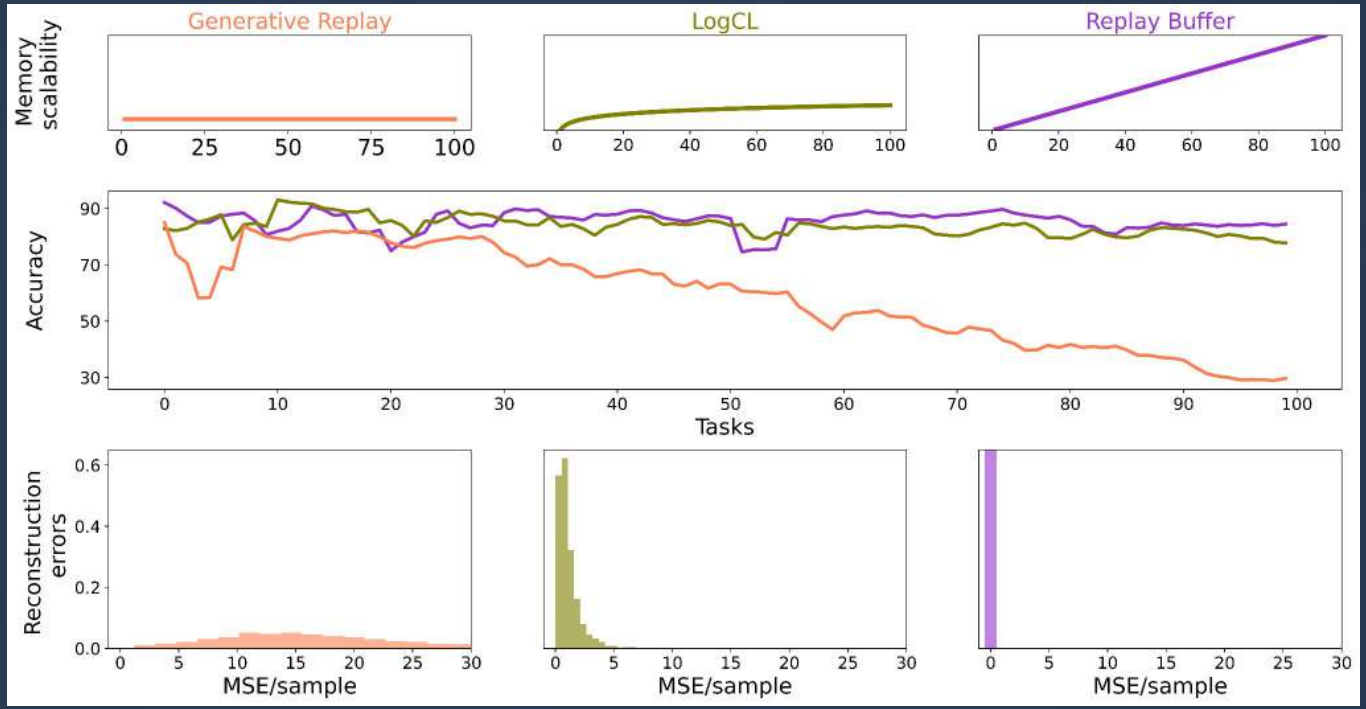
Figure 1: Sequential training of scholar models. (a) Training a sequence of scholar models is equivalent to continuous training of a single scholar while referring to its most recent copy. (b) A new generator is trained to mimic a mixed data distribution of real samples $\boldsymbol{x}$ and replayed inputs $\boldsymbol{x}'$ from previous generator. (c) A new solver learns from real input-target pairs $(\boldsymbol{x}, \boldsymbol{y})$ and replayed input-target pairs $(\boldsymbol{x}', \boldsymbol{y}')$, where replayed response $\boldsymbol{y}'$ is obtained by feeding generated inputs into previous solver.

Shin et al., 2017

Limitations of generative replay

Masarczyk et al.'22

Quality of reconstructions degrades with time (contrary to the quality of buffered samples)



Logarithmic Continual Learning

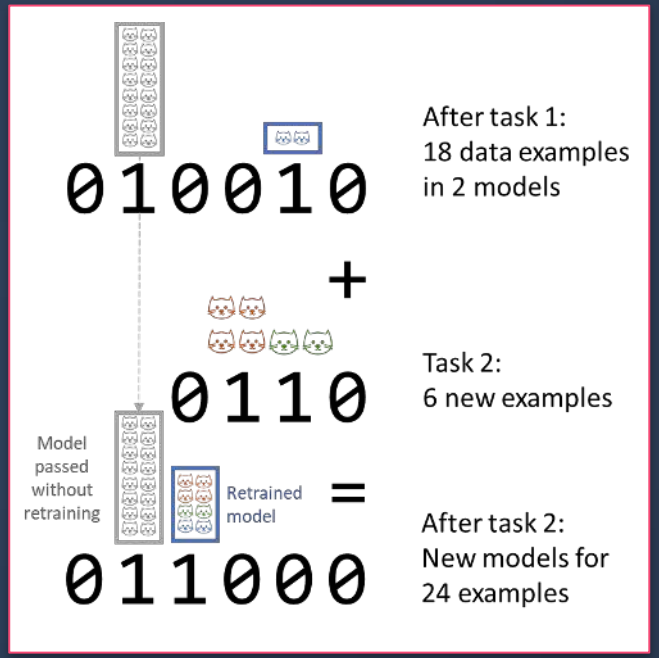
Masarczyk et al.'22

Population of **generative models created**.

Sample allocation based on **binary arithmetic.**

Bits correspond to sequentially built generative models with twice larger capacity.

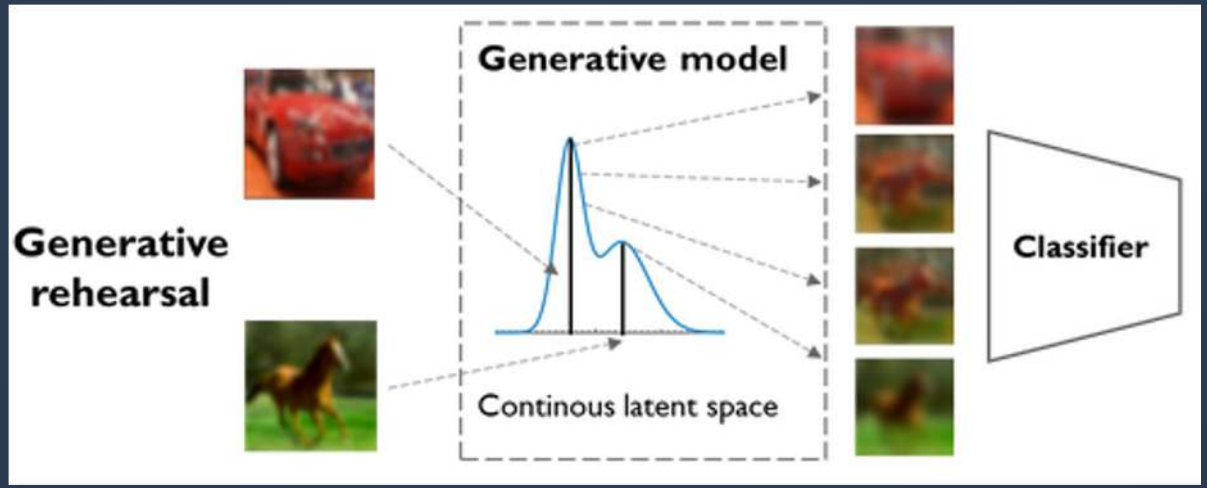
Sample re-allocation happens only when bit changes within the proposed representation.



Limitations of generative replay

Class overlap within the latent space

Given the encoding of the input images into a single continuous latent space, multiple classes from different tasks can live in **proximity in the latent space**, hence leading to the **confusion between replayed classes.**



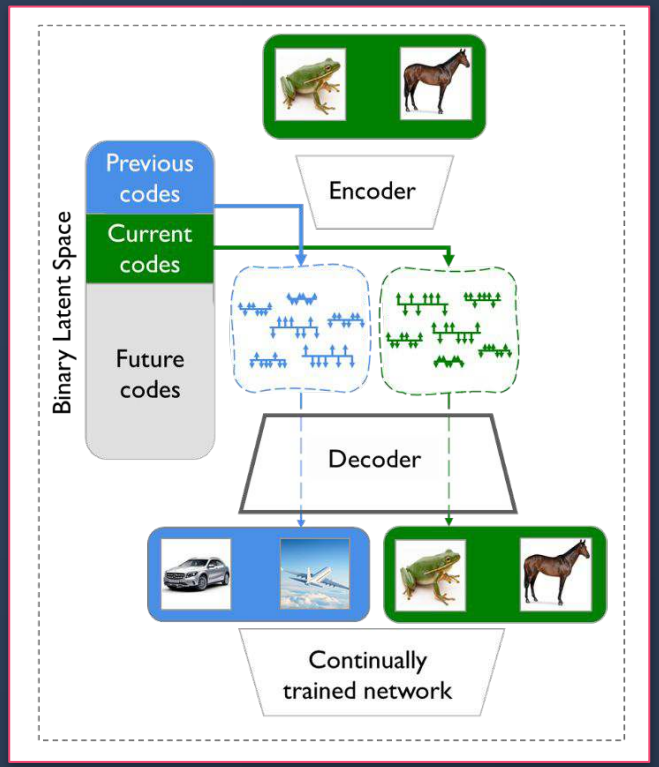
BinPlay

A **generative replay** method

Binary latent space used to **store binary codes of the rehearsed examples**

Allows for **efficient memorization** of previously seen samples and their replay.

Deja et al.'21



BinPlay

Deja et al.'21

New batch
size = 3

1. Precalculation of binary codes for **3** new objects:

1
2
3

$C(i) = i * p^{CONST} \mod 2^3$

001
100
011

2. Assigning images to codes

Encoder

001
100
011

Decoder

3. Inference for all 3 encoded images

Calculate 3 vectors

1, 2, 3

C(i)

001 100 011

Decoder

BinPlay - results

Deja et al.'21

Data storage	Model	MNIST	Fashion MNIST	CIFAR–10
Memory buffer	GEM [21]	86.3 ± 1.4	70.3 ± 0.7	17.5 ± 1.6
	iCARL [28]	71.7 ± 0.5	67.7 ± 0.4	32.4 ± 2.1
	ER [5]	84.5 ± 1.6	70.2 ± 1.7	38.5 ± 1.7
	ER–MIR [1]	87.6 ± 0.7	69.7 ± 2.7	47.6 ± 1.1
Both	AQM [3]	93.6 ± 0.7	67.4 ± 0.3	51.4 ± 2.2
Generative replay	GEN-MIR [1]	86.6 ± 0.3	52.4 ± 1.5	18.8 ± 0.9
	OCDVAE [27]	93.2 ± 3.7	69.9 ± 1.7	21.6
	GR [31]	92.5 ± 0.5	68.0 ± 0.9	27.3 ± 1.3
	GR+distill [34]	95.6 ± 0.2	78.1 ± 1.0	28.4 ± 0.3
	RTF [33]	95.1 ± 0.3	75.2 ± 0.8	28.7 ± 0.2
	BinPlay (ours)	**97.2 ± 0.6**	**81.4 ± 0.9**	**63.3 ± 1.4**

MultibandVAE

Deja et al.'22

Limitations of BinPlay

Capacity of latent space limited

Alignment of previous encodings fixed for future batches

MultibandVAE

Deja et al.'22

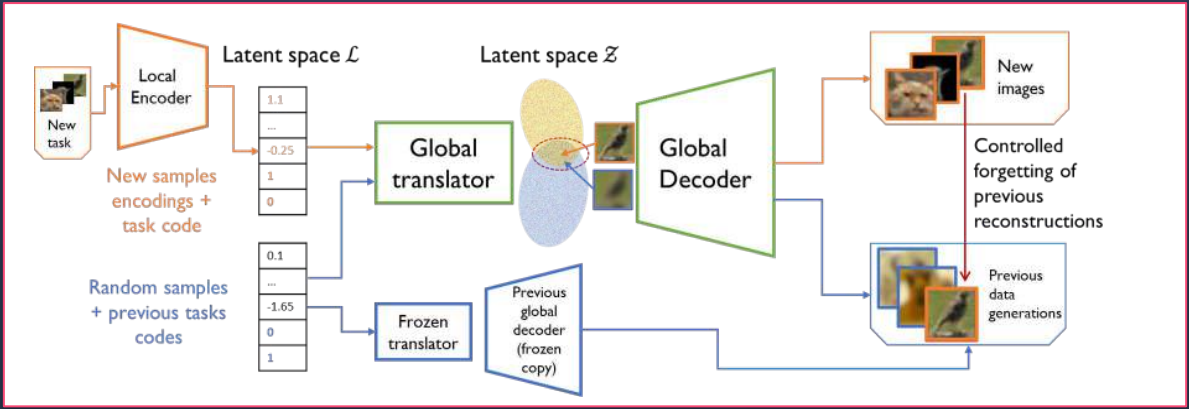
Limitations of BinPlay

- Capacity of latent space limited
- Alignment of previous encodings fixed for future batches

Continual Learning problem set as a **latent space knowledge accumulation** task

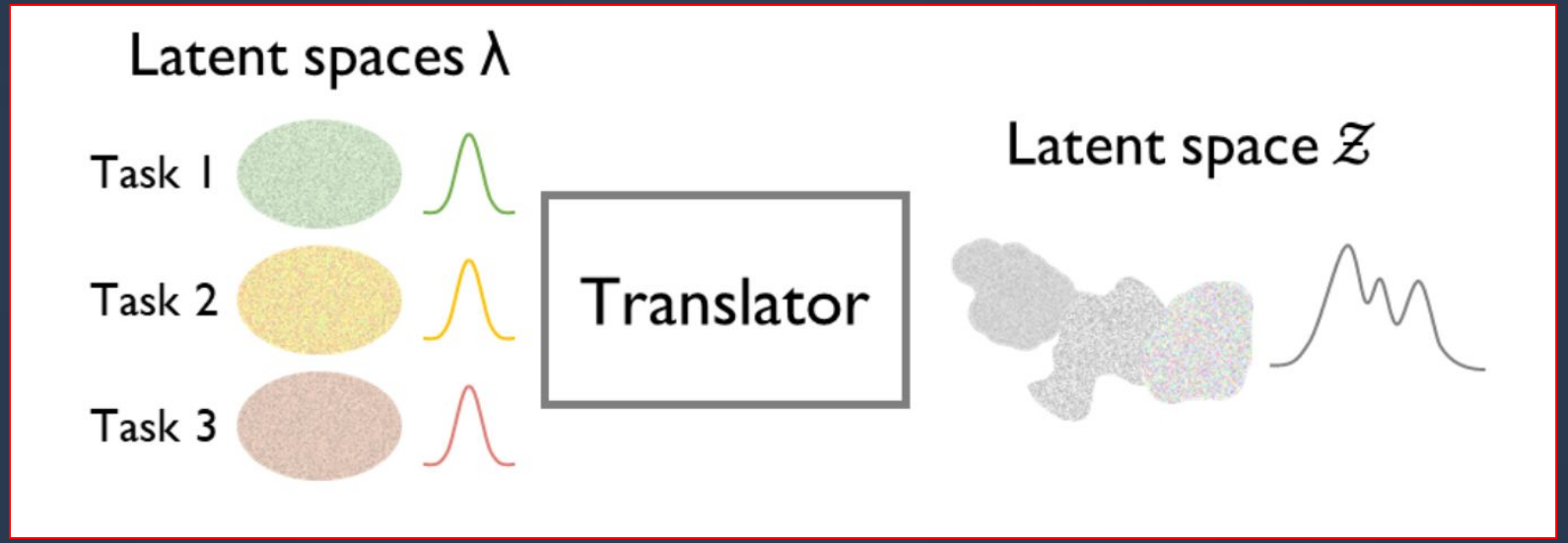
Effective **re-alignment of per-batch latent space partitions**

Similarities between classes seen across batches exploited



Latent re-alignment

Deja et al.'22



Training samples

Deja et al.'22

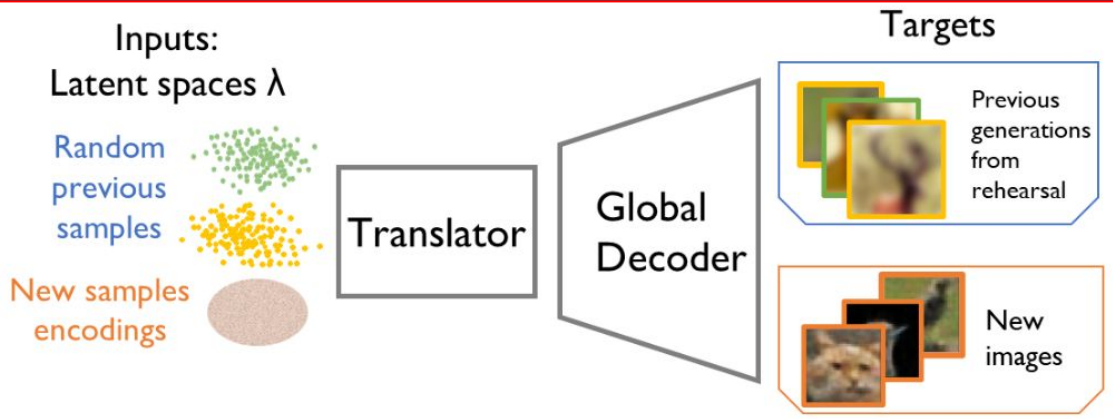
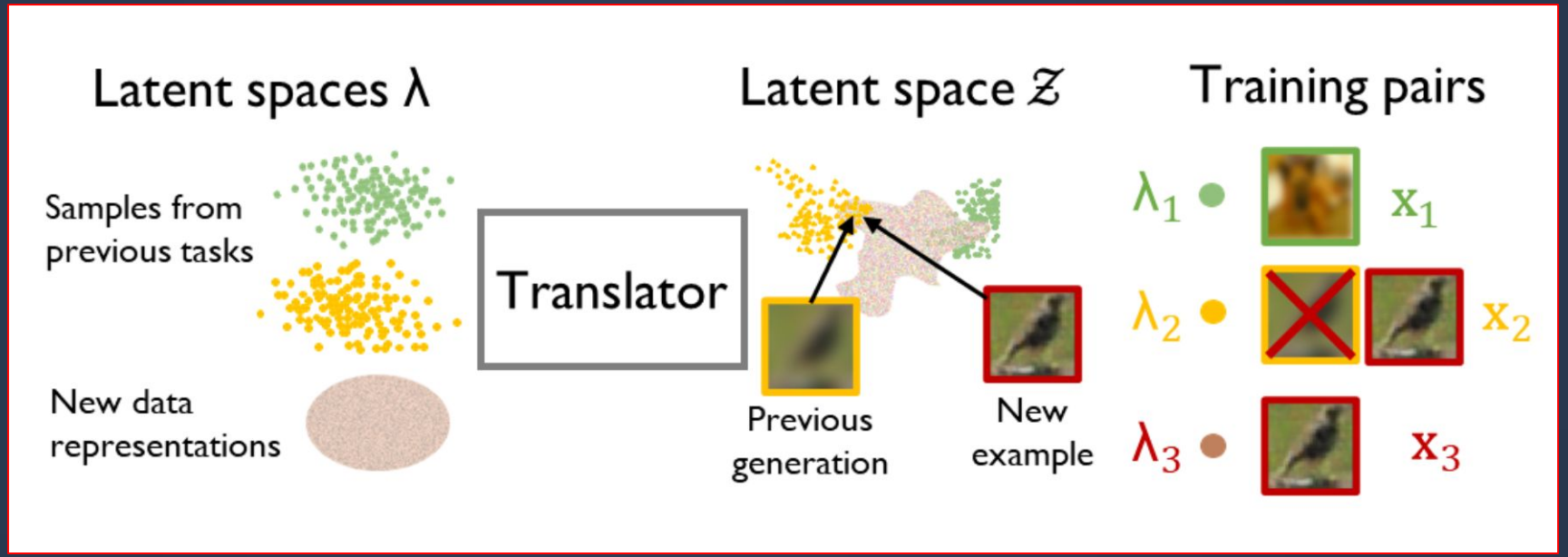


Figure 3: We train our translator and global decoder with new data encoded to latent space λ associated with original images, and samples of previous data generations generated in a standard rehearsal schema.

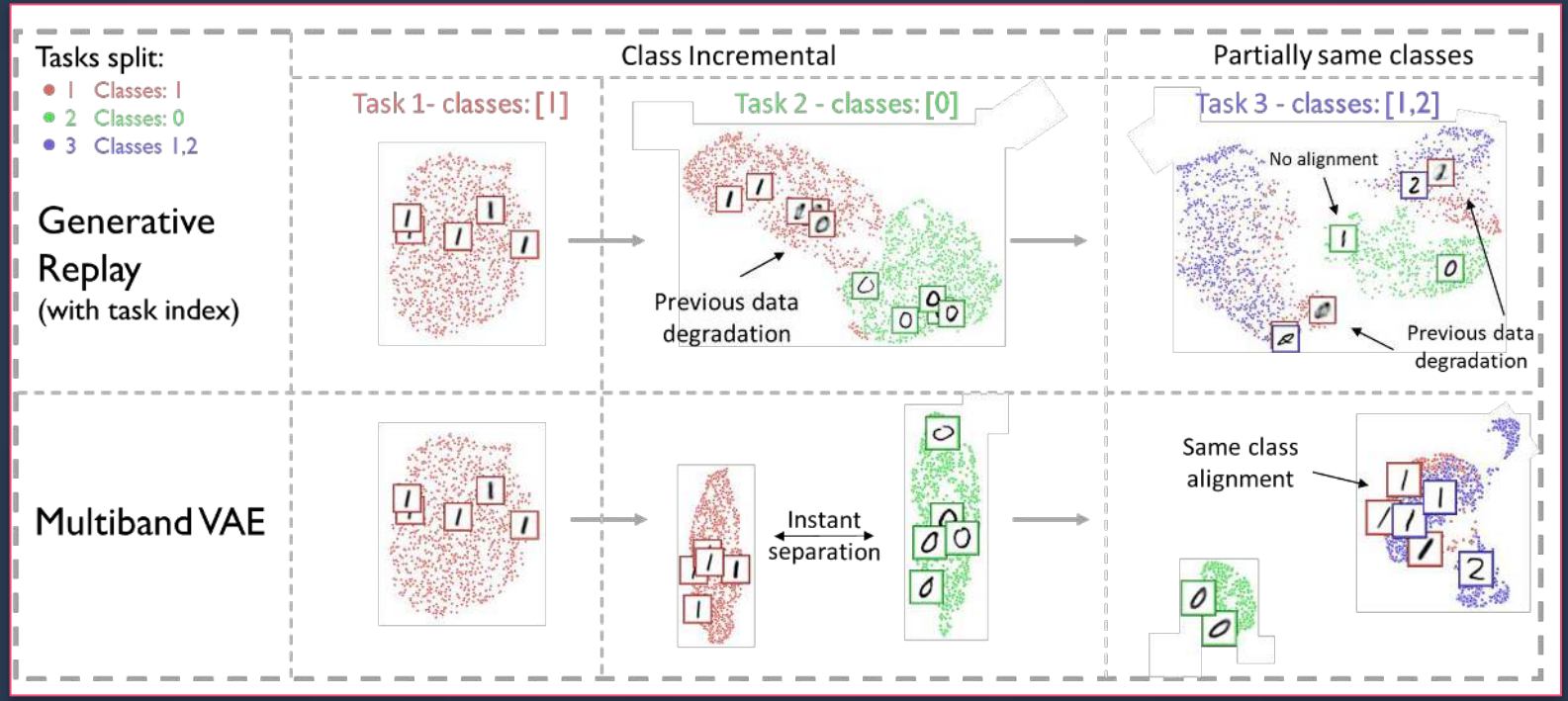
Controlled forgetting

Deja et al.'22



Knowledge consolidation - example

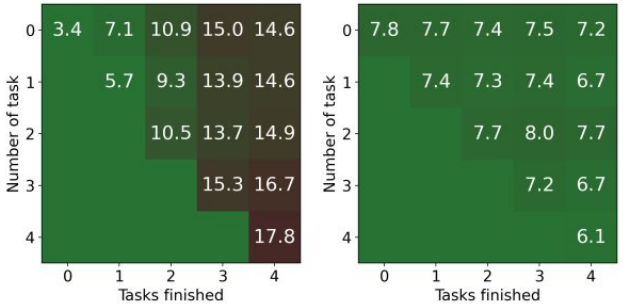
Deja et al.'22



MultibandVAE - results

Deja et al.'22

Num. tasks	Split-MNIST Class Incremental 5			MNIST Dirichlet $\alpha = 1$ 10		
Measure	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑
SI	129	77	80	153	75	76
EWC	136	73	82	120	79	83
Generative replay	120	79	87	254	70	65
VCL	68	85	94	127	78	80
HyperCL	62	91	87	148	78	75
CURL	107	95	77	181	84	74
Livelong-VAE	173	75	72	224	63	73
Livelong-VAEGAN	48	**98**	89	131	90	83
Multiband VAE	**24**	94	**97**	**41**	**92**	**96**
Multiband VAE (conv)	23	92	98	30	92	97



Generative replay Multiband VAE

Figure 7: Comparison of Wasserstein distance ↓ between original simulation channels and generations from VAE trained with standard GR and our multiband training. Multiband VAE well consolidates knowledge with forward transfer (each row starts with better score) and backward knowledge transfer (improvement for some rows when retrained with more data). At the same time standard GR struggles to retain quality of generations on old tasks.

Conclusions

Continual learning is a long-standing problem of machine learning and a cornerstone of human-level artificial intelligence.

Generative modeling perspective offers new insights into the topic of continual learning, as accumulating knowledge is natural for generative models.

Our proposed approaches (BinPlay, MultibandVAE, LogCL) prove validity of this research path and open new interdisciplinary connections across neuro- and computational sciences.

Future research includes adapting generative models to feature replay, rather than original data rehearsal, as well as investigating modern generative models (e.g. diffusion-based models) and the representations they learn in the context of continual learning.

ICML'22 Workshop on Dynamic Neural Networks

Speakers

T. Tuytelaars (KU-Leuven), F. Locatello (Amazon), G. Huang (Tsinghua), X. Huang (Fudan)

Organizers

M. Levorato (UC Irvine), S. Scardapane (Sapienza), B. McDanel (F&M College), A. Banino (DeepMind), C. Riquelme (Google Brain), T. Trzcinski (WUT, UJ)

ICML | 2022

Thirty-ninth International Conference on Machine Learning

https://dynn-icml2022.github.io/

Topics: *dynamic routing, mixture-of-experts models, early-exit methods, conditional computations, capsules and object-oriented learning, reusable components, online network growing and pruning, online neural architecture search and applications of dynamic networks (continual learning, wireless/embedded devices and similar topics).*

Collaborators

Warsaw University of Technology

TOOPLOOX

JAGIELLONIAN UNIVERSITY IN KRAKÓW

Fundusze Europejskie

Rzeczpospolita Polska

Unia Europejska
Europejskie Fundusze Strukturalne i Inwestycyjne

FNP TEAM-NET

National Science Centre Poland

Still interested in CL? 🤣

CVLab@PW seeks excellent PhD candidates interested in ML, CV, CL and other fancy abbreviations…

Some facts about us:

~14 people (4 post-doc/faculty, 10 PhD students)
~10 A* conf papers (6x NeurIPS, ICML, 2xCVPR, IJCAI)
~3 ongoing research grants (2x NCN, Microsoft Research Cambridge)
0 animals hurt during our experiments 😎

Contact us: tomasz.trzcinski@pw.edu.pl
or come visit 413/414 WEiTI PW

Thank you!

CVLab@PW

Still interested in CL? 🤣

Starting **Computer Vision Group** @ IDEAS NCBR, a newly created Polish centre of R&D in AI (German MPI-alike).

Some facts about IDEAS NCBR:
~200 researchers to be recruited (incl. 100 PhD students)
~10 groups to be created, 3 already formed
~3 teams in Computer Vision group will work on CL, conditional computations and partial information

Benefits: Competitive PhD/post-doc salary, private health insurance, innovation bonus (equity in spin-offs)

Looking for PhD students, post-docs & team leaders:
https://lnkd.in/etMV7ViK
https://lnkd.in/eWjHv3Pt
https://lnkd.in/eMiDQ9v9

Thank you