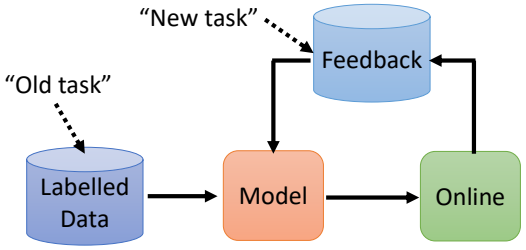
Life Long Learning (終身學習)

Apply new users' data on trained models after deployed

Life Long Learning (LLL), Continuous Learning, Never Ending Learning, Incremental Learning

After models are online, deployed, model parameters can be updated by new feedbacks.



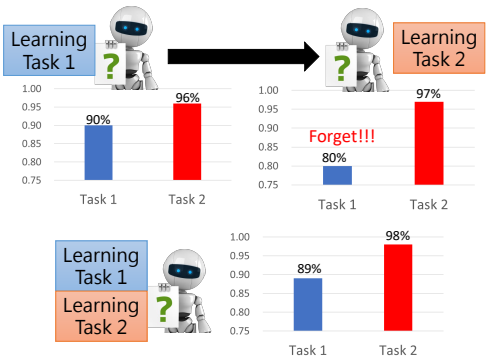
Difficulties

Catastrophic Forgetting

Models forget what they learned last time

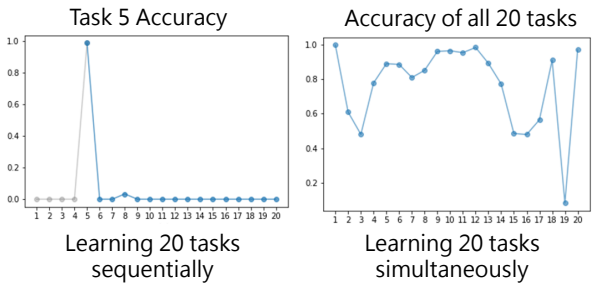
E.g. Image recognition

Current researches are on different **tasks**, or should we say different **domains.**
Models will forget task 1 after learning task2. But if we put task1 and task2 together and learn together, the model will just be fine.



E.g. QA mission: bAbi

Facebook trained QA model through 20 models and found machine forgot what it learned before.
It is Not because machine are not able to do it, but machine just didn't do it.
Left figure shows when the machine are trained by the 20 tasks **sequentially**. Right figure indicates when the machine are trained by all the 20 tasks **once**.

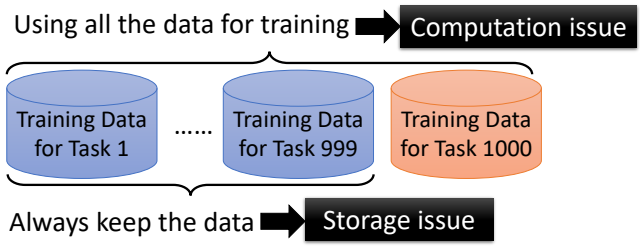


Can multi-task training solve the problem?

There will be a storage issue because the machine need to storage all the data it has been seen and the computation cost is also an issue.

Multi-task training can be considered as the upper bound of LLL.

- Multi-task training can solve the problem!

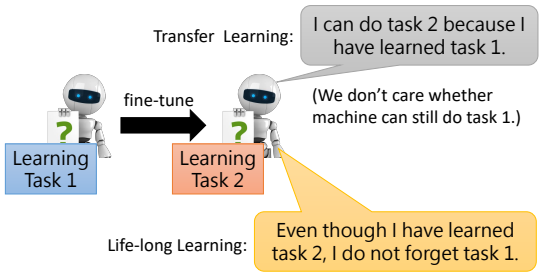


Can we just train a model for each task?

Eventually we cannot store all the models.
Knowledge cannot transfer across different tasks

Compare with Transfer Learning

It's different from transfer learning.
Transfer learning focuses on model performance of the **new task** and Life Long Learning focuses on the **previous tasks.**



How to Evaluation the technique of Life long learning

We need a sequence of tasks.
We use specific rules to modified the original tasks.
Methods:
1. Permutation, change angle etc…
2. Or we can divide the original task to different subset tasks.

$R_{i,j}$: after training task i, performance on task j.
If $i > j$, after training task I, does task j be forgot.
If $i < j$, can we transfer the skill of task I to task j.
There are two applicable scores to evaluate the level of **forgotten**.

Level of Forgotten

$$Accuracy = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$$

$$Backward\ Transfer = \frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i}$$

Usually Backward Transfer is negative.

		Test on			
		Task 1	Task 2	.....	Task T
Rand Init.		$R_{0,1}$	$R_{0,2}$		$R_{0,T}$
After Training	Task 1	$R_{1,1}$	$R_{1,2}$		$R_{1,T}$
	Task 2	$R_{2,1}$	$R_{2,2}$		$R_{2,T}$
	⋮				
	Task T-1	$R_{T-1,1}$	$R_{T-1,2}$		$R_{T-1,T}$
	Task T	$R_{T,1}$	$R_{T,2}$		$R_{T,T}$

New task capability

$$Forward\ Transfer = \frac{1}{T-1}\sum_{i=2}^{T} R_{i-1,i} - R_{0,i}$$

If the machine have not see task T, only see tasks t to t-1, What the machine will learn.

		Test on			
		Task 1	Task 2	.....	Task T
Rand Init.		$R_{0,1}$	$R_{0,2}$		$R_{0,T}$
After Training	Task 1	$R_{1,1}$	$R_{1,2}$		$R_{1,T}$
	Task 2	$R_{2,1}$	$R_{2,2}$		$R_{2,T}$
	⋮				
	Task T-1	$R_{T-1,1}$	$R_{T-1,2}$		$R_{T-1,T}$
	Task T	$R_{T,1}$	$R_{T,2}$		$R_{T,T}$

Research Directions of Solutions

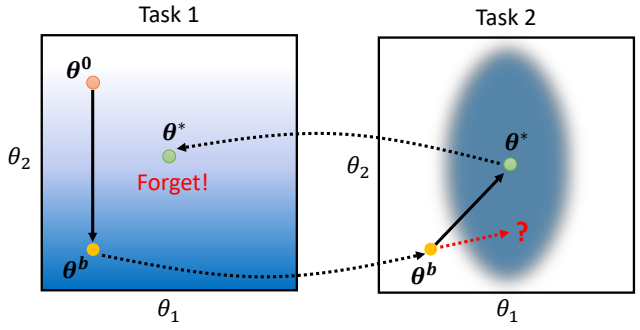
1. **Selective Synaptic Plasticity - Regularization- based Approach**
   1. **Some neurons are fixed.**
2. **Additional Neural Resource Allocation**
3. **Memory Reply**

Why Catastrophic Forgetting happens?

The right figure shows the loss functions of task1 and task2. We used gradient descent to adjust the parameters of our machine.

Task 1: $\theta^0 \xrightarrow{task\ 1} \theta^b \xrightarrow{task\ 2} \theta^*$

θ^* are the parameters after trained by t1 and t2. But when we apply θ^* on task 1, the result is bad. This is the phenomenon of forget. This is because the moving direction of θ when training task 2.



The error surfaces of tasks 1 & 2.
(darker = smaller loss)

Selective Synaptic Plasticity

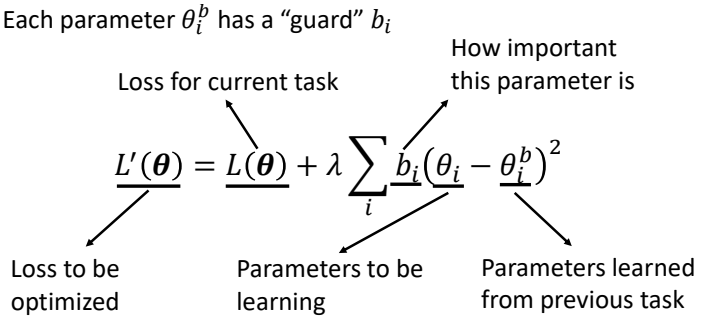
Basic Idea: Some parameters in the model are important to the previous tasks. Only change the unimportant parameters.

$\theta^b$is the model learned from the previous tasks.
To solve the forget phenomenon, each parameter θ_i^b has a "guard" b_i.

We don't minimize θ directly in order to prevent the catastrophe forgetting.

$$L'(\theta) = L(\theta) + \lambda \sum_i b_i(\theta_i - \theta_i^b)^2$$

The guard b_i represent the importance of the parameter to previous tasks. Which means **how strong we hope (θ_i, θ_i^b) to be as close as possible.**



θshould be close to θ^b in certain directions.

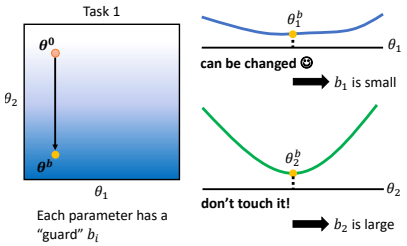
If $b_i = 0$, there is no constraint on b_i, which will lead catastrophic forgetting
If $b_i = \infty$, θ_i would always be equal to θ_i^b, which is called **Intransigence.** This means the machine lacks the ability to learn new tasks.

Until now, b_i **are manual set.** The reason is if we let machine learn b_i, the machine will try to minimize the loss. The easiest way to do so is to let b_i equal to 0.

How do we find b_i?

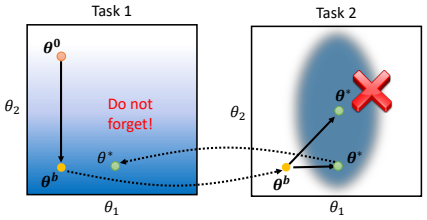
If change a parameter won't change the result of task 1, which means the parameter is not critical to task 1, the b1 can be set small.

When we change a parameter lead to a huge change of task 1, we know it is critical to task 1. Then the b will be large.



b_i can be calculated by different methods.

Elastic Weight Consolidation (EWC)
https://arxiv.org/abs/1612.00796
Synaptic Intelligence (SI)
https://arxiv.org/abs/1703.04200
Memory Aware Synapses (MAS)
https://arxiv.org/abs/1711.09601
RWalk
https://arxiv.org/abs/1801.10112
Sliced Cramer Preservation (SCP)
https://openreview.net/forum?id=BJge3TNKwH



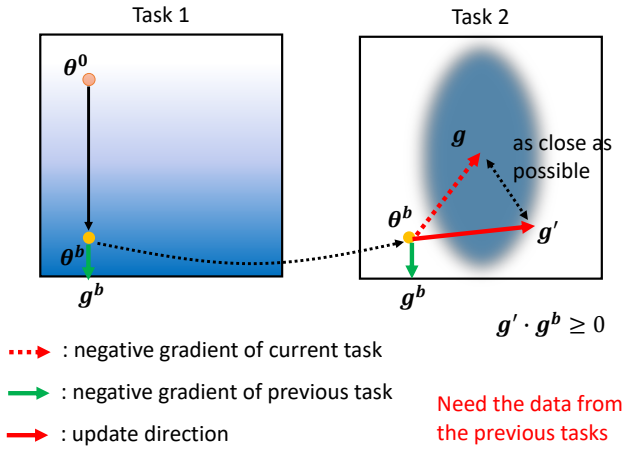
b_1 is small, while b_2 is large.
(We can modify θ_1, but do not change θ_2.)

Gradient Episodic Memory (GEM)

We can change the update direction to avoid catastrophic forgetting.

This method still need to store the data from task 1.
GEM need to store all the previous data.

But GEM only need to store the smallest data.

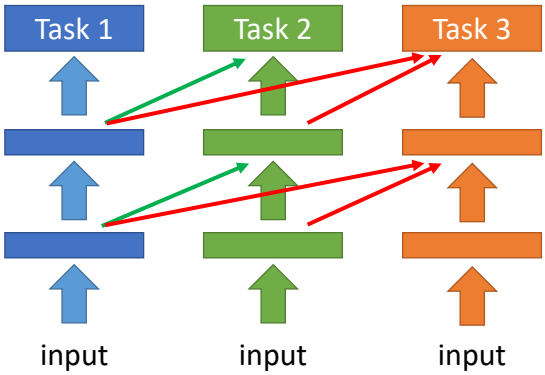


Additional Neural Resource Allocation

Progressive Neural Networks

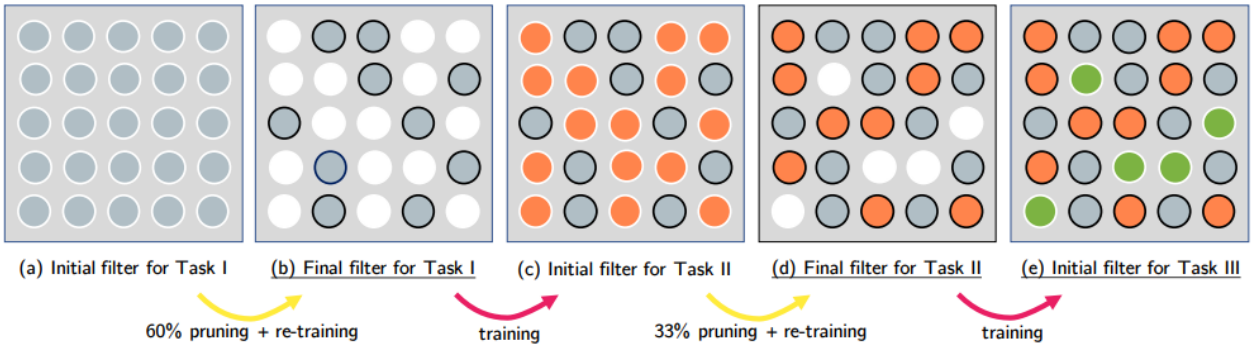
Task 2 share layer parameters from task 1 and parameters of task 1 are fixed.

When have new missions, add some new neurons.
The model can be very big and exhausts all the memory.



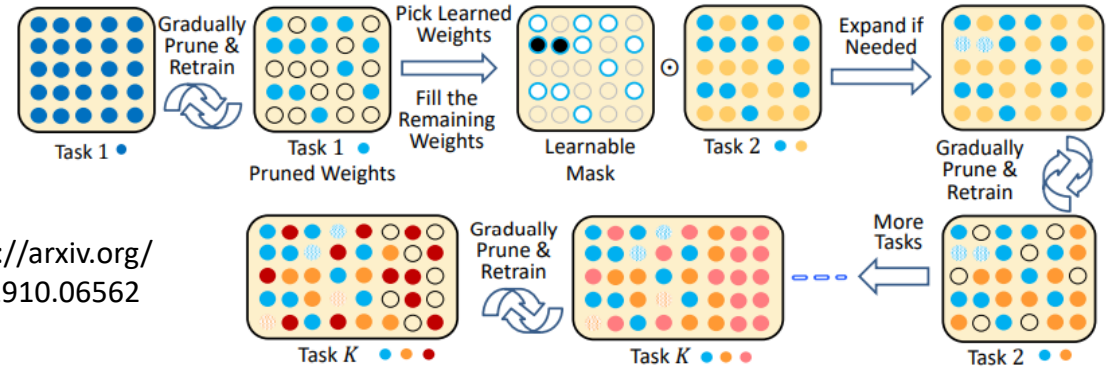
PackNet

Open a large network first and use different parts of the network for each task.



Compacting, Picking, and Growing (CPG)

Combine Progressive Neural Networks and PackNet



https://arxiv.org/abs/1910.06562

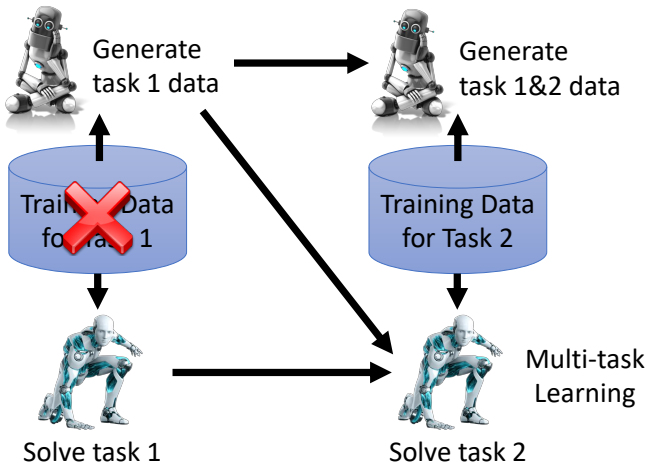
Memory Reply

Generating Data

Generating pseudo-data using generative model for previous tasks by **a generative model.**

Use $generator_{t-1}$ to generate data for task t.

Generator will occupy some memories but is useful for life long learning.



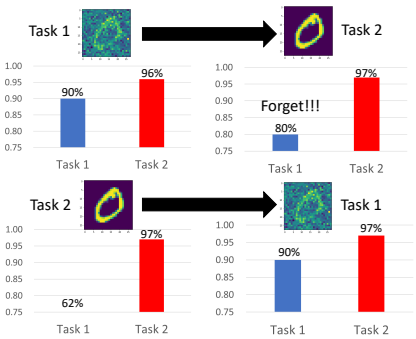
Adding new classes

Number of classes can be vary between tasks.

Learning without forgetting (LwF)

iCaRL: Incremental Classifier and Representation Learning

Three scenarios for continual learning https://arxiv.org/abs/1904.07734



Curriculum Learning

taskonomy = task + taxonomy
(分類學)

Changing order of learning tasks can change the result a lot.
Therefore we need to know which order is better for the training to avoid catastrophic forgetting.