Lifelong and Continual Learning

Part II – Slides for June 16, 2022

Bing Liu and **Zixuan Ke**

Department of Computer Science

University of Illinois at Chicago

A short PhD course (8 hours) given at Aalborg University on June 14 and June 16, 2022

Topics

- Lifelong or continual learning
- Early research on lifelong learning
- Continual learning based on deep neural networks
- Continual learning in the open-world
- Summary

Let’s Recall What We Learned on Tue.

- Continual Learning
  - After a task is learned, its training data (at least a large portion of it) is no longer accessible.
  - Earlier work on lifelong/continual learning mainly builds separate models for knowledge transfer.
- Deep Continual Learning:
  - We want one **single** neural model to be able to do well on all tasks
  - What will happen?
    - Catastrophic forgetting

Let’s Recall What We Learned on Tue.

- Deep Continual Learning:
  - Catastrophic forgetting
    - Intuitively
      - The 2D plane example
    - Mathematically
      - We cannot directly estimate the expected loss for previous tasks because their data is not accessible

Let's Recall What We Learned on Tue.

- Deep Continual Learning:
  - Evaluation
    - Possible scenarios
      - Catastrophic forgetting 😔
      - No Forgetting ☺️
      - Problematic Learning 😔
      - Forward Transfer ☺️
      - Backward Transfer ☺️
    - Quantity Metrics
      - Meta Table

Let’s Recall What We Learned on Tue.

- Deep Continual Learning:

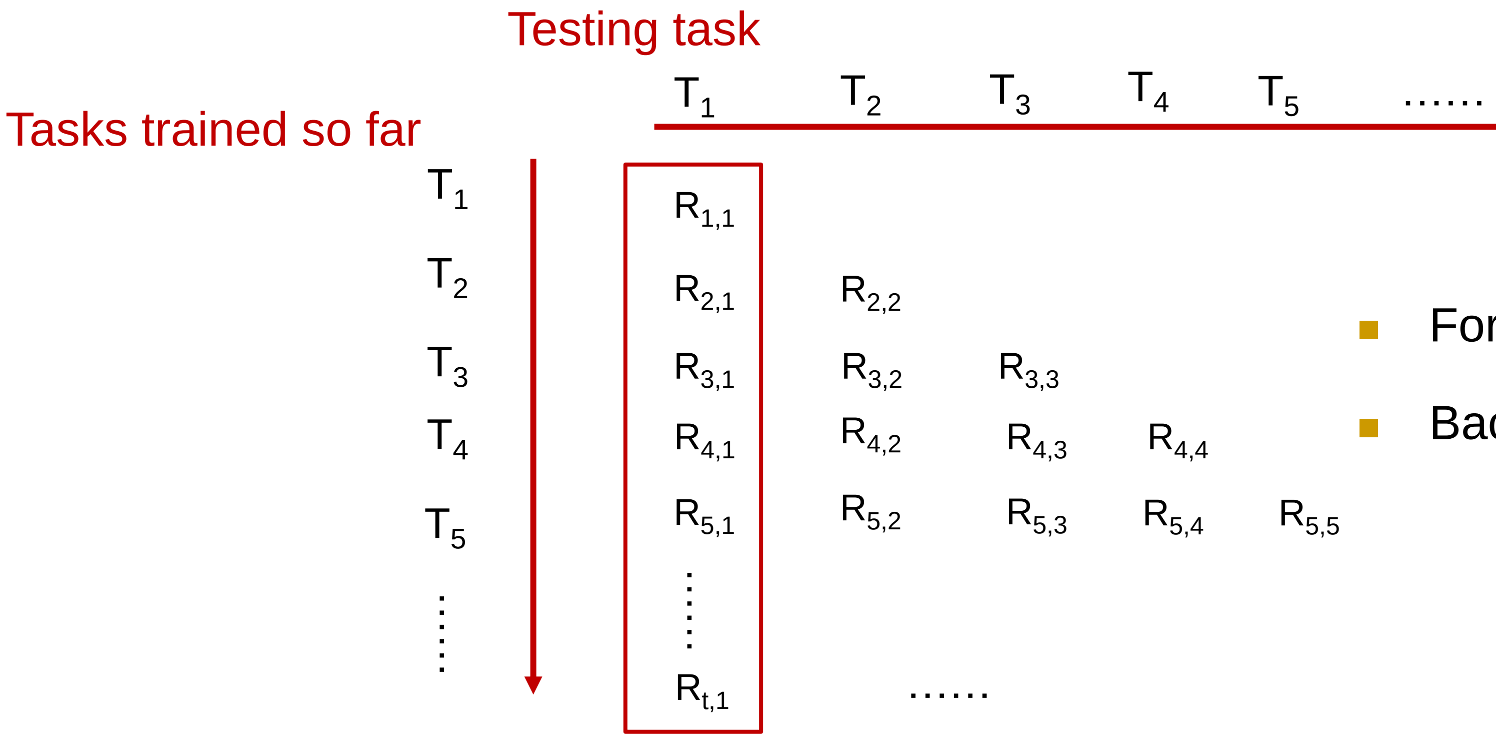
$R_{m,n}$: The performance of the model on task T_n, after continually training *till* task T_m

Testing task

Tasks trained so far

	T_1	T_2	T_3	T_4	T_5	……
T_1	$R_{1,1}$					
T_2	$R_{2,1}$	$R_{2,2}$				
T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
T_5	$R_{5,1}$	$R_{5,2}$	$R_{5,3}$	$R_{5,4}$	$R_{5,5}$	
……						

Let’s Recall What We Learned on Tue.



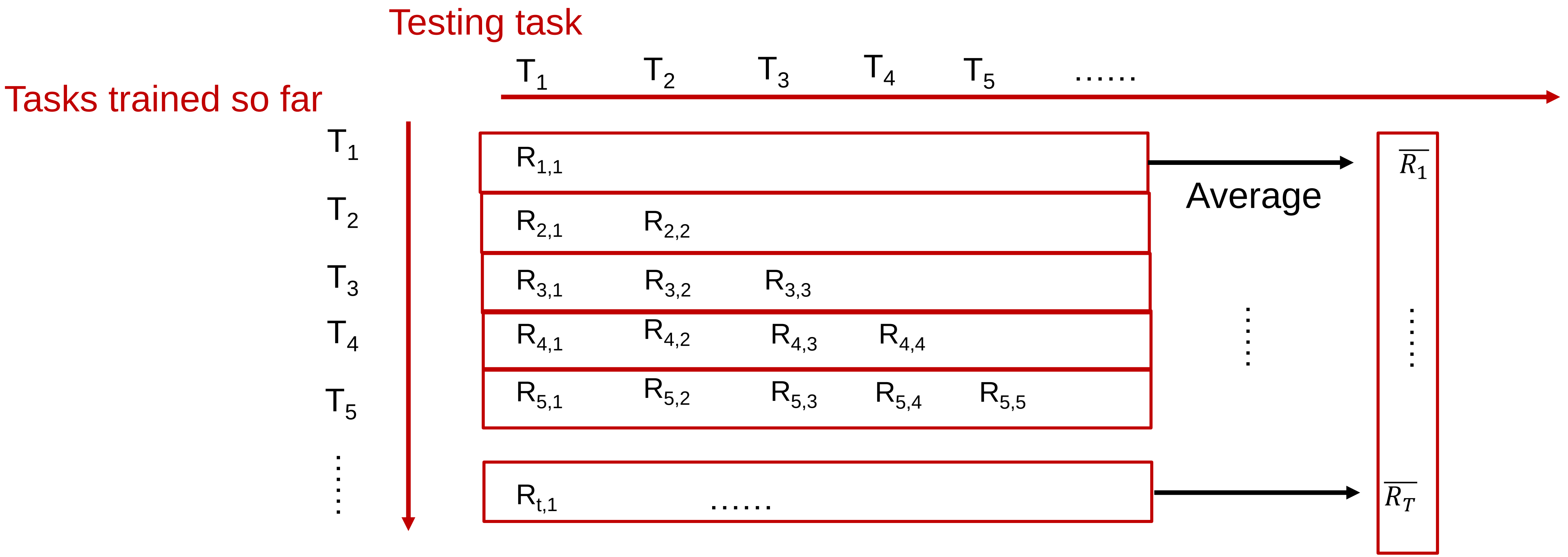
Tasks trained so far \ Testing task	T_1	T_2	T_3	T_4	T_5	……
T_1	$R_{1,1}$					
T_2	$R_{2,1}$	$R_{2,2}$				
T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
T_5	$R_{5,1}$	$R_{5,2}$	$R_{5,3}$	$R_{5,4}$	$R_{5,5}$	
⋮	⋮					
	$R_{t,1}$	……				

- Forgetting rate (FR): $\frac{1}{T-1}\sum_{i=1}^{T-1} R_{i,i} - R_{t,i}$
- Backward Transfer (BWT): $\frac{1}{T-1}\sum_{i=1}^{T-1} R_{t,i} - R_{i,i}$

Y. Liu et al,. Mnemonics training: Multi-class incremental learning without forgetting. CVPR, 2020

Lopez-Paz and Ranzato, Gradient Episodic Memory for Continual Learning, NIPS 2017

Let’s Recall What We Learned on Tue.



Testing task

Tasks trained so far

	T_1	T_2	T_3	T_4	T_5	……	Average
T_1	$R_{1,1}$						$\overline{R_1}$
T_2	$R_{2,1}$	$R_{2,2}$					
T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$				
T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$			⋮
T_5	$R_{5,1}$	$R_{5,2}$	$R_{5,3}$	$R_{5,4}$	$R_{5,5}$		
⋮	$R_{t,1}$	……					$\overline{R_T}$

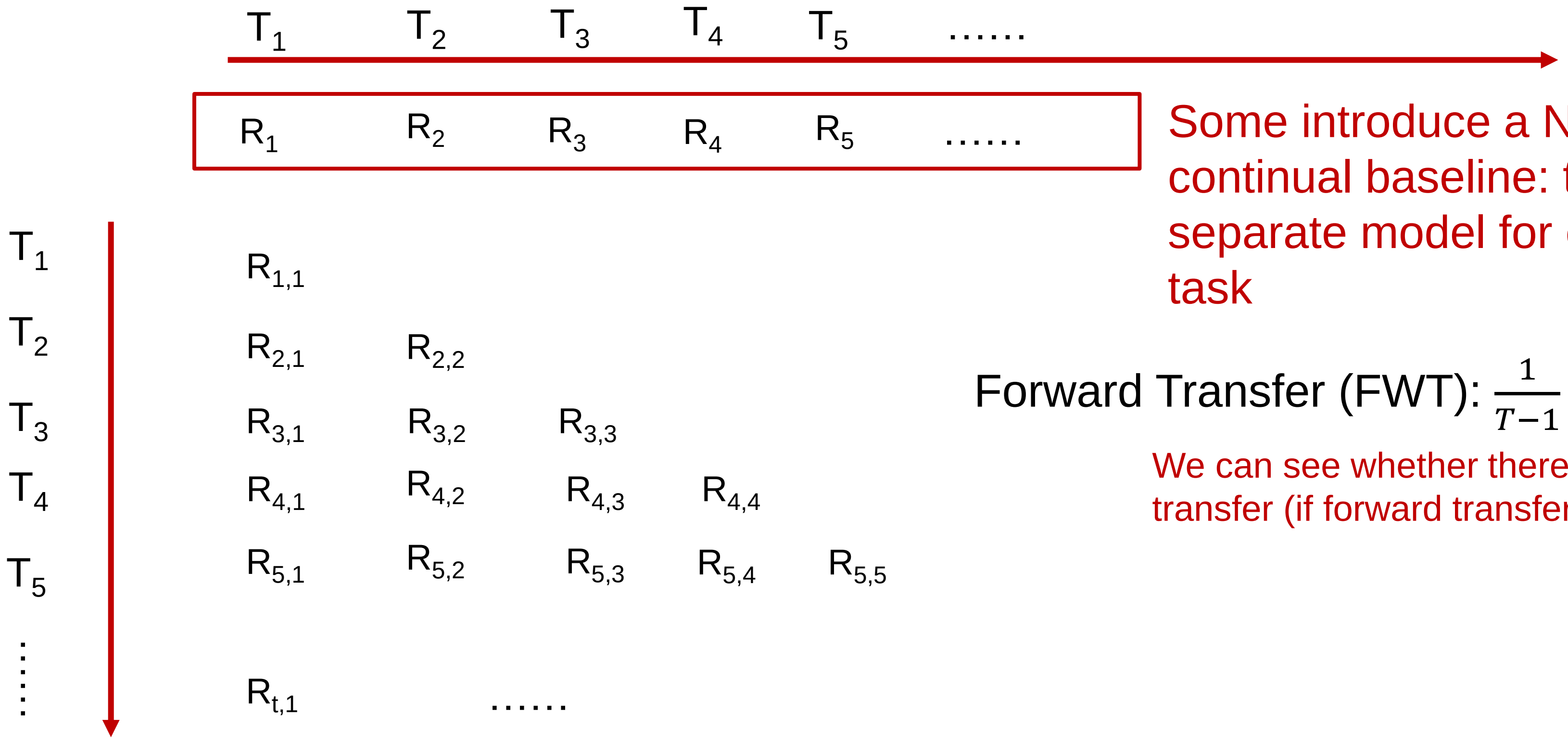
Instead of drawing a curve for one single task, Some use the progressive accumulated average

Rebuffi et al,. iCaRL: Incremental classifier and representation learning. CVPR 2017

Let’s Recall What We Learned on Tue.

Testing task

Tasks trained so far



	T_1	T_2	T_3	T_4	T_5	……
	R_1	R_2	R_3	R_4	R_5	……
T_1	$R_{1,1}$					
T_2	$R_{2,1}$	$R_{2,2}$				
T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
T_5	$R_{5,1}$	$R_{5,2}$	$R_{5,3}$	$R_{5,4}$	$R_{5,5}$	
……	$R_{t,1}$	……				

Some introduce a Non-continual baseline: train separate model for each task

Forward Transfer (FWT): $\frac{1}{T-1}\sum_{i=1}^{T-1} R_{i,i} - R_i$

We can see whether there is forward transfer (if forward transfer, FWT > 0)

Lopez-Paz and Ranzato, Gradient Episodic Memory for Continual Learning, NIPS 2017

Let’s Recall What We Learned on Tue.

Testing task

Tasks trained so far	T_1	T_2	T_3	T_4	T_5	……
T_1	$R_{1,1}$					
T_2	$R_{2,1}$	$R_{2,2}$				
T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
T_5	$R_{5,1}$	$R_{5,2}$	$R_{5,3}$	$R_{5,4}$	$R_{5,5}$	
……	$R_{t,1}$		……			

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

After continual training, average over all seen tasks (the last row) can give us a high-level evaluation of the model. This is a popular number to report the results.

Let’s Recall What We Learned on Tue.

- Deep Continual Learning:
  - Evaluation
    - Possible scenarios
    - Quantity Metrics
    - Popular non-continual learning baselines
      - Multi-task Learning (MTL)
        - Usually regarded as upper bound
      - Individual task Learning (ONE)
        - Train a separate model for each task (no forgetting/transfer)
      - Naïve continual learning (NCL)
        - Train tasks sequentially, without taking care of forgetting (catastrophic forgetting) prevention

Let’s Recall What We Learned on Tue.

- Deep Continual Learning:
  - Evaluation
  - Goals
    - ❑ Prevent forgetting
    - ❑ Encourage forward and backward transfer

Let’s Recall What We Learned on Tue.

- Deep Continual Learning:
  - Evaluation
  - Goals
  - Approaches

Let’s Recall What We Learned on Tue.

- Replayed-based
  - Use an explicit memory to maintain a subset of training samples, or
  - Learn a data generator
- Regularization-based
  - Add a regularization term to loss function
- Architecture-based
  - Each task dedicates a different sub-network

Let’s Recall What We Learned on Tue.

- Replayed-based
  - GEM
    - Store raw samples
    - Optimize the training by constraining the previous loss not to increase
  - LAMOL
    - Background: Language model
    - Finetune the language model as data generator and task solver *at the same time*
      - Use a unified format (QA) for all tasks
    - When a new task arrives, the LM first generates pseudo-samples, and then combines both the generated and new samples for learning.

Let's Recall What We Learned on Tue.

- Play with the model
  - https://github.com/ZixuanKe/PyContinual

- Regularization-based
  - Add a regularization term to loss function

Let’s Recall What We Learned on Tue.

- Regularization-based
  - **Regularization = loss + penalty term**
  - DER++
    - Distill previous knowledge (network response) to current model by adding a distillation penalty term
  - EWC
    - Probably the most well-known continual learning system (not the best performing though)
    - Compute the prior based on parameter importance (fisher matrix, base on gradient) to constrain the update

Approaches

- Replayed-based
  - Use an explicit memory to maintain a subset of training samples, or
  - Learn a data generator
- Regularization-based
  - Add a regularization term to loss function
- *Architecture-based*
  - *Each task dedicates a different sub-network*

Approaches

- Both replayed-based and regularization-based have forgetting
  - Can we have some methods that guarantee no forgetting at all?
  - Yes! Architecture-based
  - **NOTE**
    - No forgetting does not mean we can solve the CL problem
      - It does *largely* solve the Task-incremental problem (TIL), because it can totally avoid forgetting
        - we will see that transferring knowledge is still not easy
      - It does not solve class/domain-incremental learning (CIL/DIL) problem
        - Architecture-based cannot be directly used for task agnostic scenarios
        - We will see why

Architecture-based

- What is a sub-network?
  - Subnetwork = mask * network
  - The mask can be applied at different locations
    - Parameters
    - Outputs
  - We will see both

Traditionally:

$$h = f(\mathbf{x}, W)$$

At parameter level

$$f(\mathbf{x}, M \odot W)$$

At output level

$$\mathbf{h}_l' = \mathbf{a}_l^t \odot \mathbf{h}_l$$

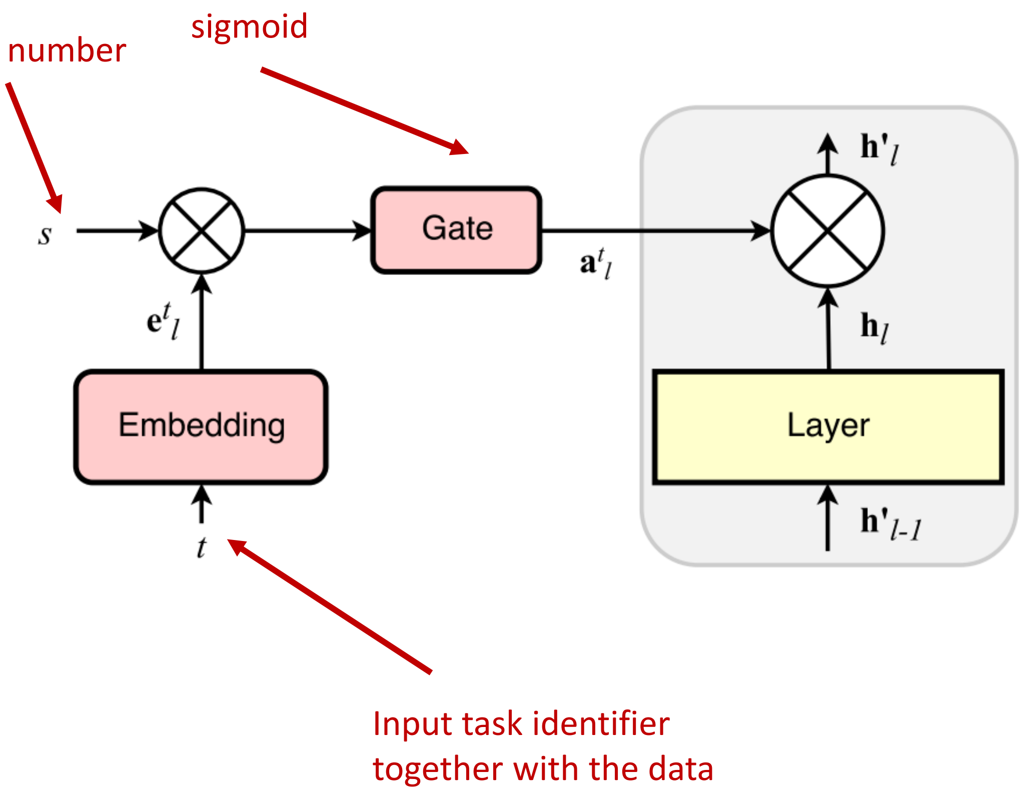
Architecture-based: output-based

- HAT[5]
    - Idea
        - Given a *static* network, if one can restrict the training to a sub-network, and make sure this sub-network is not changed by any subsequent tasks. Forgetting can be prevented.
    - How does it work?
        - How to find the sub-network?
        - How to make sure it is not changed?

[5]: Serra et al., Overcoming catastrophic forgetting with hard attention to the task, ICML 2018

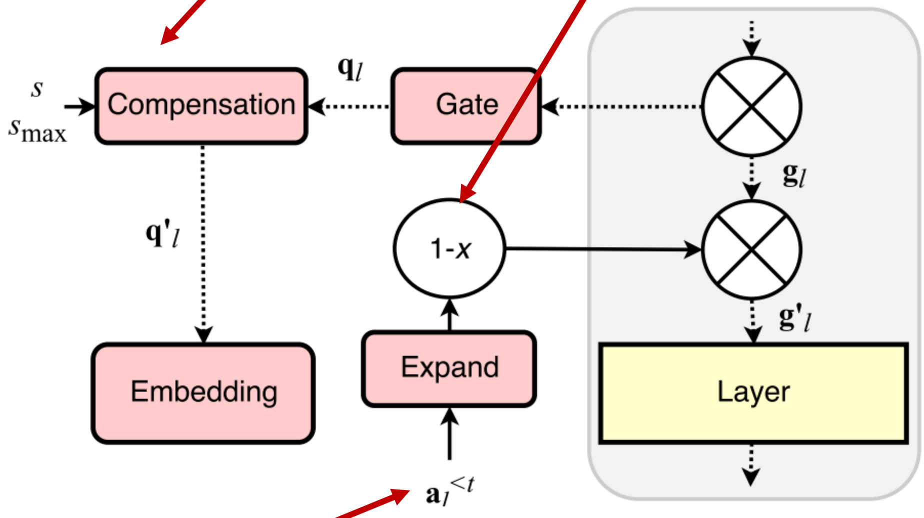
Architecture-based: output-based

- HAT
  - Forward: find the sub-network
    - Input the task identifier *t,* together with the data
    - The *t* is used to train a *task embedding.*
    - A pseudo-gate function (sigmoid) is applied on the task embedding, so that a gate is masked on the output of *each* layer
    - The mask gives us a sub-network for the task *t*
    - After training task *t,* the mask is stored



Architecture-based: output-based

- HAT
  - Backward: make sure previous sub-networks are not changed
    - When training the new task, we block the previous tasks' sub-networks
      - By setting their gradient to 0



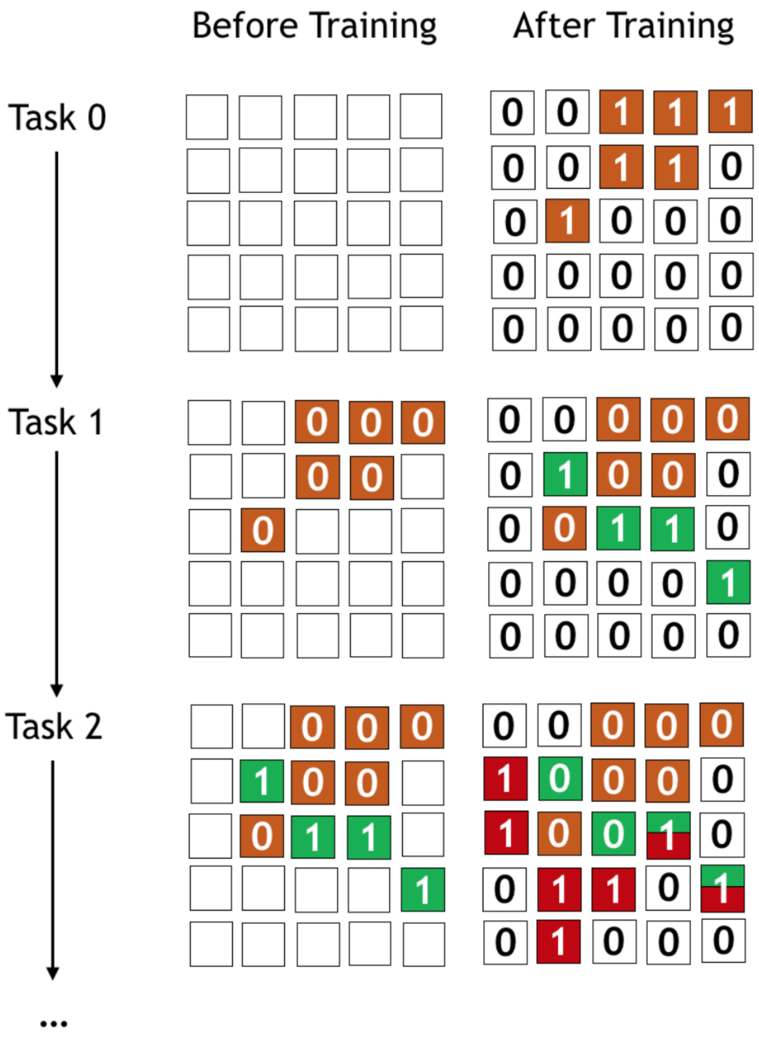
Additional training trick

Since it is binary, 1-x means to set it to 0

Accumulated sub-network

Architecture-based: output-based

- HAT
  - A dynamic illustration
    - Task 0 is the first task. After learning it, we obtain its useful units/neurons, marked in orange with a 1 in each unit, which serves as a mask for future tasks.
    - In learning task 1, we found that task 1 is not similar to task 0. Those useful units for task 0 is masked (with 0 in those orange units or cells in the matrix on the left). The process also learns the useful units for task 1



Architecture-based: output-based

- HAT
  - How is it performing
    - Datasets
      - CIFAR10, CIFAR100, MNIST, SVNH
    - Metrics
      - Forgetting Rate

Architecture-based: output-based

- HAT
  - How does it perform

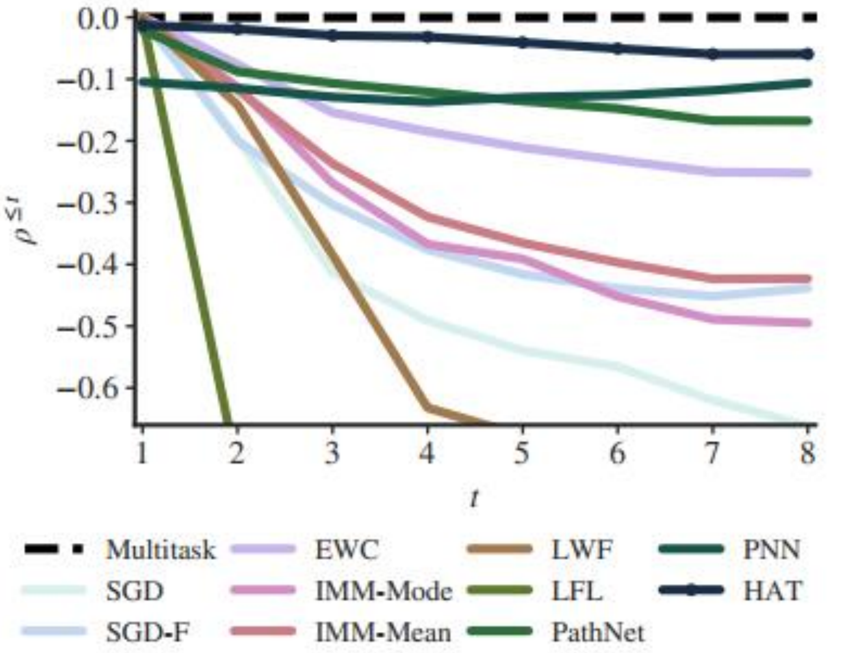


Figure 3. Average forgetting ratio $\rho^{\le t}$ for the considered approaches (10 runs).

Architecture-based: output-based

- HAT
  - How about pros and cons?
  - Pros
    - Almost no forgetting
      - Because different tasks are using different sub-networks
  - Cons
    - Task information (task-id) is needed in testing
      - Can only be used for Task-incremental Learning
    - The network capacity can easily run out
    - Only very weak knowledge transfer can happen
      - Tasks are separated into different subnetworks (though they may share some parameters, but used parameters cannot change)

Now we know why the architecture-based model solves the Task-incremental learning problem, but not class/domain-incremental problem

Architecture-based: output-based

- HAT
  - Cons
    - Task information is needed in testing
      - Can only be used in Task-incremental Learning
    - The network capacity can easily run out
    - Only very weak knowledge transfer can happen
      - Tasks are separated into different subnetworks (though they may share some parameters, but used parameters cannot change)
  - Can we address the “capacity quickly ran out” issue?
    - Yes. Let’s change a bit of our mindset
      - If you want your model to perform well, what do you do?

Architecture-based: parameter-based

- Supsup[6]
  - Idea:
    - If you want your model to perform well, what do you do?
      - Train the weights?
    - Supsup provides an alternative view
      - Keep the weight random
      - Train the mask to select sub-networks
      - This is called "***sup***ermask"
    - Different tasks use different masks
      - Forgetting is thus prevented
      - This is called "***sup***erposition"

Traditionally:

$$f(\mathbf{x}, W)$$

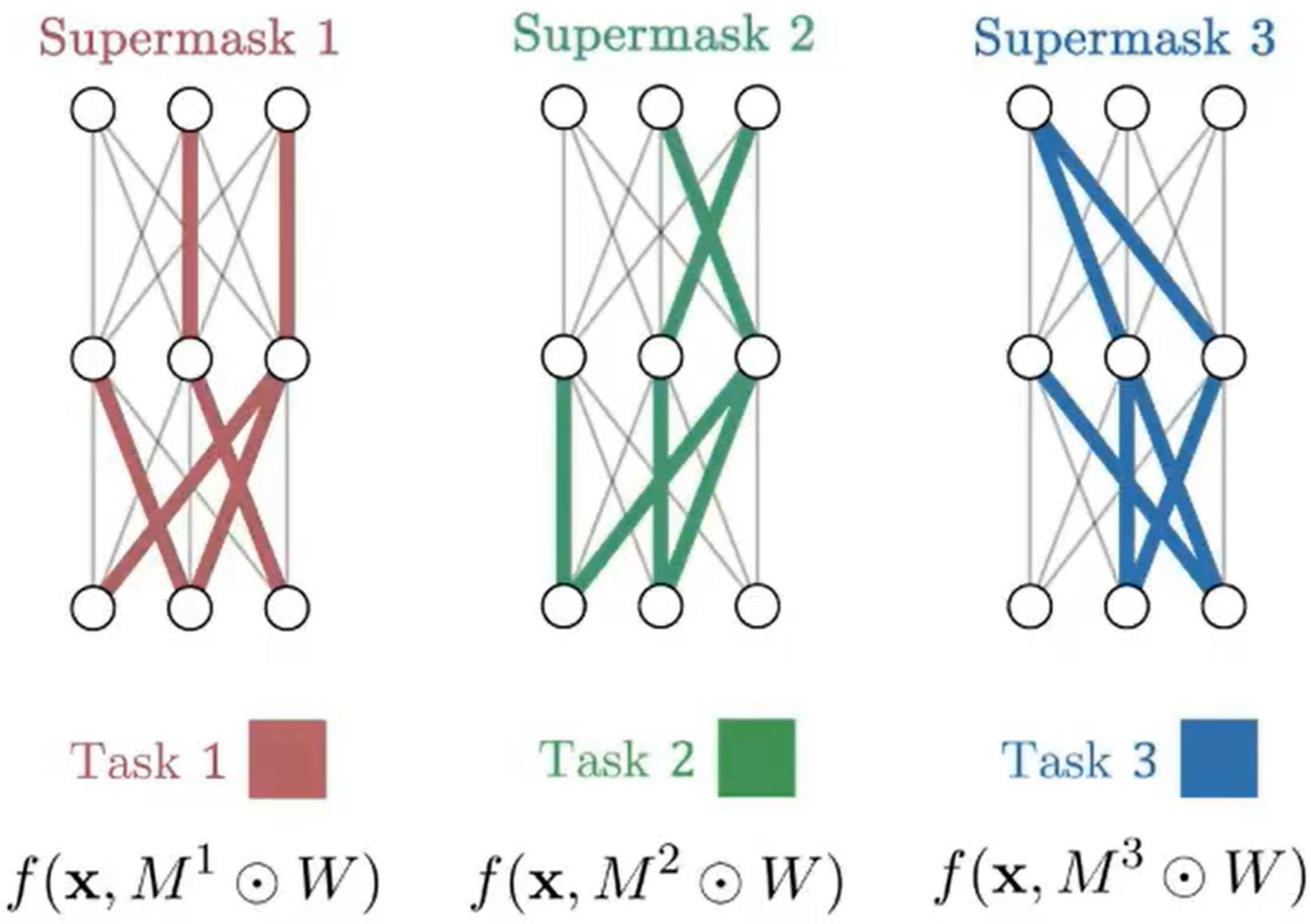
Supermasks:

$$f(\mathbf{x}, M \odot W)$$

[6]: Wortsman et al., Supermasks in Superposition, NeurIPS 2020

Architecture-based: parameter-based

- Supsup
  - How does it work?
    - The weights are not trained.
    - But masks are trained to choose the network
  - Masks are simply the indices, which can be efficiently saved


Supermask 1
Supermask 2
Supermask 3
Task 1
Task 2
Task 3
$f(\mathbf{x}, M^1 \odot W)$
$f(\mathbf{x}, M^2 \odot W)$
$f(\mathbf{x}, M^3 \odot W)$

Architecture-based: parameter-based

- Supsup
  - How does it perform
    - Datasets
      - MNIST, CIFAR100
    - Metrics
      - Progressive Results

Architecture-based: parameter-based

- Supsup
  - How does it perform

Extremely large number of tasks

Why this is possible?

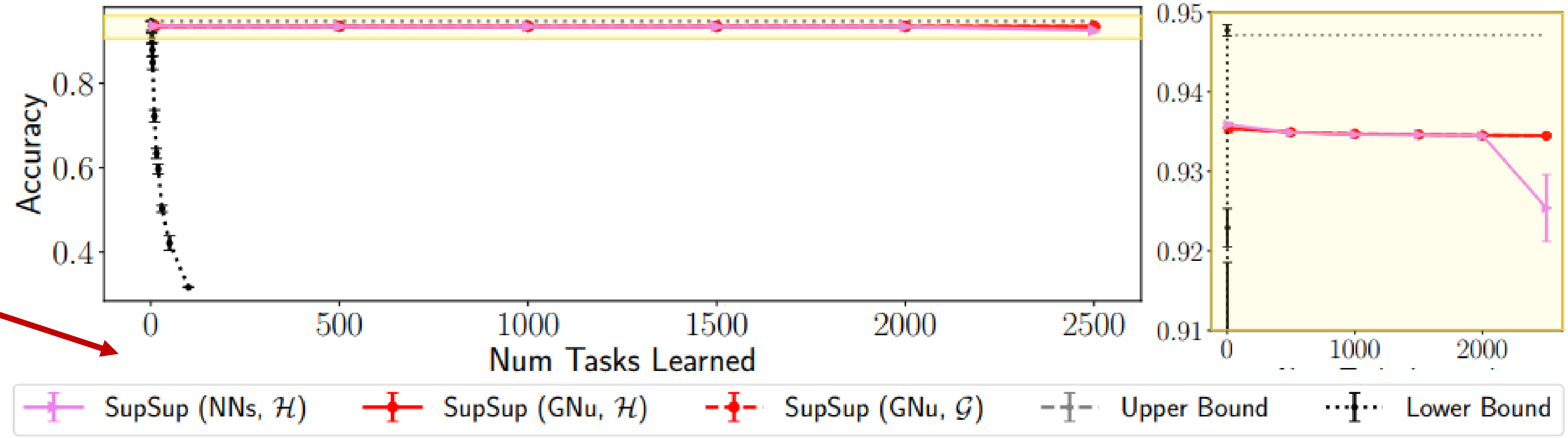


Figure 4: Learning 2500 tasks and inferring task identity using the **One-Shot** algorithm. Results for both the GNu and NNs scenarios with the LeNet 300-100 model using output size 500.

Architecture-based: parameter-based

- Supsup
  - Pros and Cons?
  - Pros:
    - The capacity is much better than HAT, since there can be almost infinite different number of masks in a network
  - Cons:
    - Need additional task identification technique to deal with class-incremental or domain-incremental
    - For network that is not randomly initialized, i.e., pre-trained network, this method will not be working
    - *Still*, Only very weak knowledge transfer can happen

Architecture-based: parameter-based

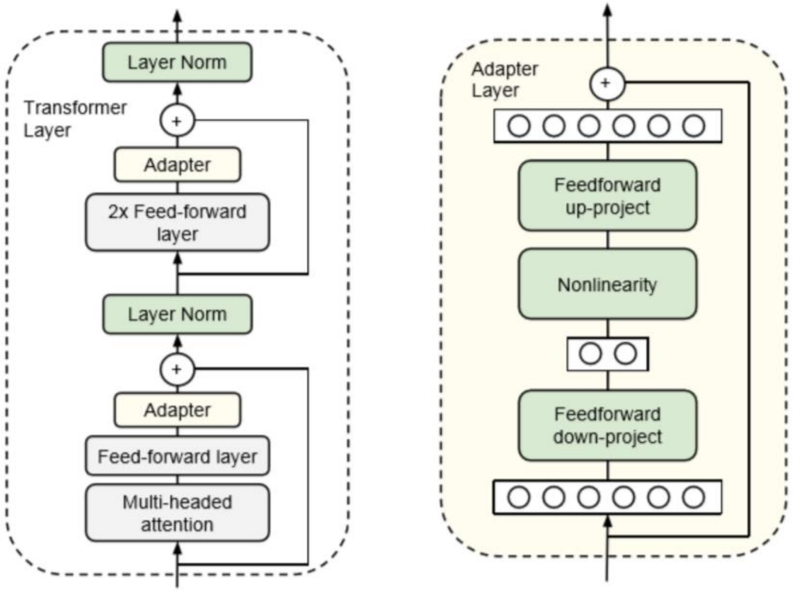
- Since both HAT and Supsup can solve the forgetting problem and they are mainly for task-incremental learning,
  - Knowledge transfer is highly desired
- But one common issue of both HAT and SupSup is
  - There is only very weak knowledge transfer
    - for Supsup, no transfer at all
  - We want our CL learner to achieve both
    - Forgetting prevention, **and** Knowledge transfer
  - Can we achieve these?

Architecture-based: beyond forgetting prevention

- A couple of models can achieve both
  - We will introduce some of them
  - BCL
  - CLASSIC
  - CAT
- These are mainly NLP models, so we need a bit more background of NLP

Architecture-based: beyond forgetting prevention

- More about NLP
  - Background
    - Pre-trained Language Model
      - RoBERT, GPT, T5…
      - How to use
        - Fine-tuning
        - *Parameter-efficient tunning*
          - Fix the language model, train additional added module (e.g., insert randomly initialized fully-connected networks to each Transformer layers)
            - The added module is called the **adapter**.
          - In this way, one only needs to train a tiny portion of the parameters to achieve similar results as fine-tuning



Architecture-based: beyond forgetting prevention

- The next few approaches are based on Adapter
  - Work together with a large-scale pre-trained model (typically, Transformer)
  - Belong to the line of parameter-efficient fine-tuning
    - Adapter
    - Prompt
    - Prefix
    - ……

Architecture-based: beyond forgetting prevention

- Adapter-based parameter-efficient fine-tuning
  - Attractive to continual learning because
    - The cost of such parameter-efficient modules is small
      - E.g., a prompt may contain only 7k parameters (Transformer-based model can easily reach 150M+ parameters)
    - We can cleverly avoid training/updating the language model (LM)
      - Forgetting thus will not happen in the LM
      - We only need to focus on the tiny simple adapter
        - Many CL methods (e.g., HAT, Supsup) can be applied to the adapter

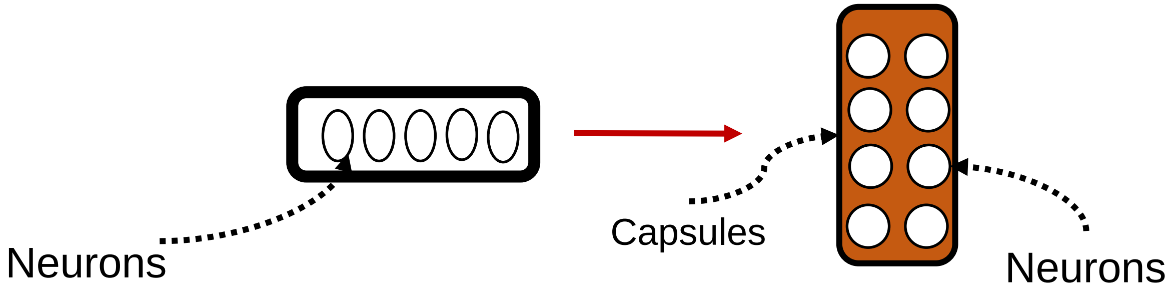
Architecture-based: beyond forgetting prevention

- So far, we know
  - by using the adapter and CL methods, we can avoid forgetting.
- But how do we enable knowledge transfer?
- We need a bit more background in order to understand the models that we will study.

Sabour et al., Dynamic routing between capsules. In NIPS, 2017

Architecture-based: beyond forgetting prevention

- Background
  - Capsule Network
    - Some materials explain this in very complicated ways. Let’s make it simple
    - It is a network type, just the same as a standard fully-connected network, except
      - From Neurons to Capsules

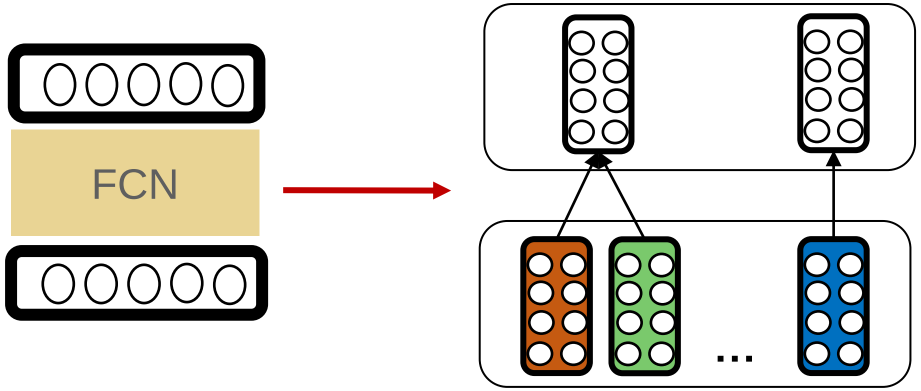


- In a typical network, a layer consists of a set of neurons. In a capsule network, a layer consists of capsules (by reshaping)
- Why? So that a layer can encode more information

Sabour et al., Dynamic routing between capsules. In NIPS, 2017

Architecture-based: beyond forgetting prevention

- Background
  - Capsule Network
    - Some materials explain this in very complicated ways. Let’s make it simple
    - It is a network type, just the same as a standard fully-connected network, except
      - From Neurons to Capsules
      - From weight connection to Routing

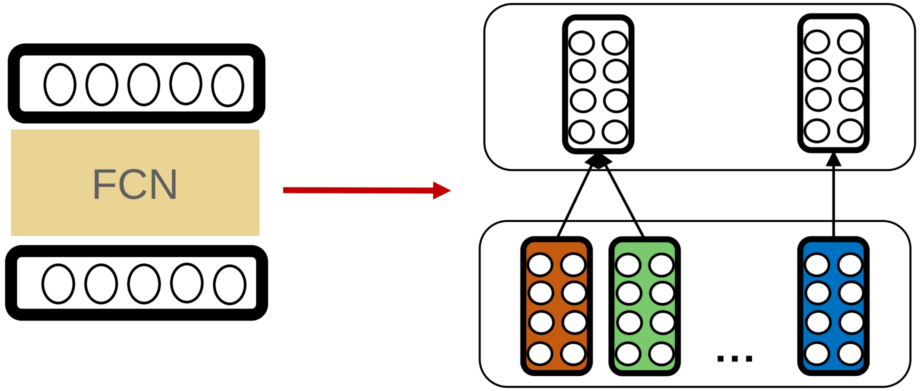


Dynamic Routing routes/groups similar capsules to next layer. It is like an unsupervised *clustering* process

Sabour et al., Dynamic routing between capsules. In NIPS, 2017

Architecture-based: beyond forgetting prevention

- Background
  - Capsule Network
    - Why is this attractive to continual learning?
      - If we regard each capsule as the feature from each task
      - The capsule network can help us group similar tasks together!
        - Similar tasks means there are transferrable knowledge among them

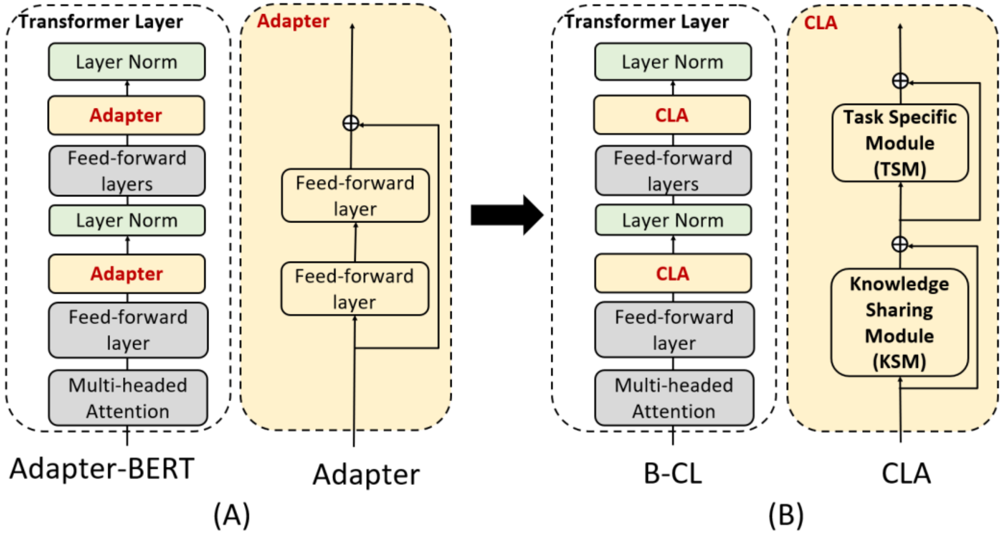


Dynamic Routing routes/groups similar capsules to next layer. It is like an unsupervised *clustering* process

Sabour et al., Dynamic routing between capsules. In NIPS, 2017

Architecture-based: beyond forgetting prevention

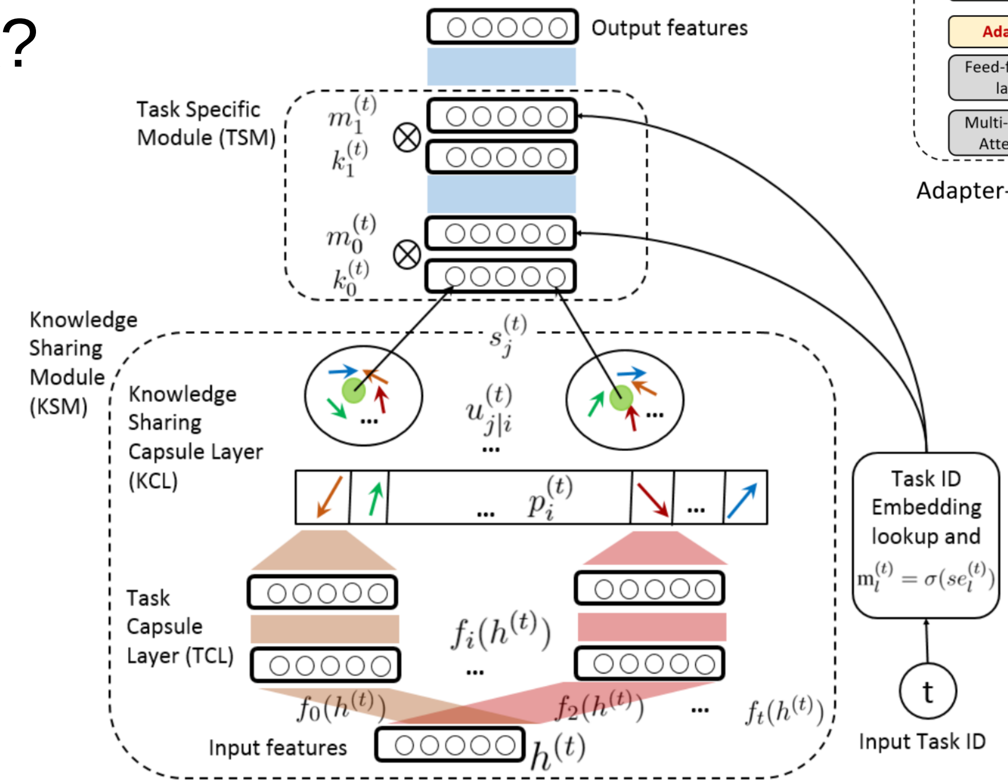
- BCL[7]
  - Idea
    - Based on Adapter, HAT and capsule network
    - Adapter avoid forgetting in the LM
    - HAT avoids forgetting in the adapter
    - Capsule network enables knowledge transfer
  - How does it work?

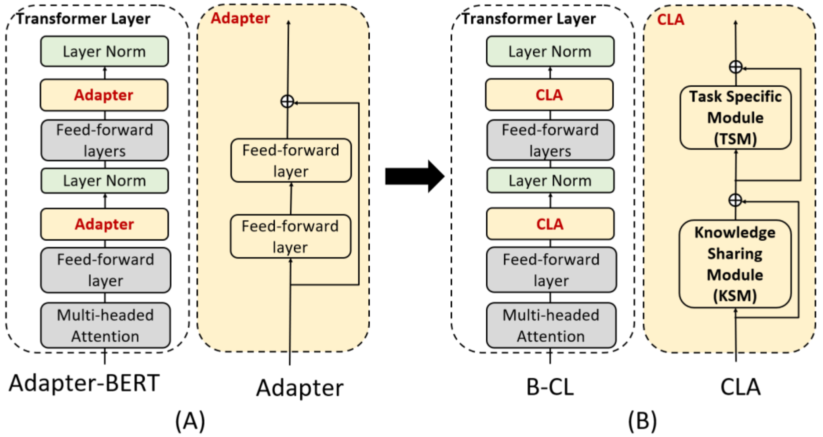


[7]: Ke et al., Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks, NAACL 2021

Architecture-based: beyond forgetting prevention

- BCL[7]
  - How does it work?
    - TSM is HAT
    - KSM is Capsule network





[7]: Ke et al., Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks, NAACL 2021

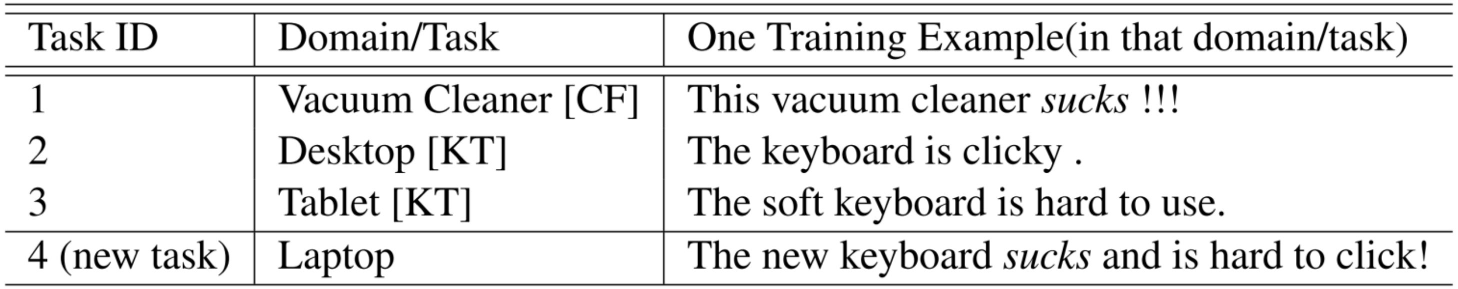
Architecture-based: beyond forgetting prevention

- BCL[7]
  - How does it perform?
    - It is used to solve an NLP problem
      - *Aspect Sentiment Classification*.

[7]: Ke et al., Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks, NAACL 2021

Architecture-based: beyond forgetting prevention

- Background
  - Aspect Sentiment Classification



Task ID	Domain/Task	One Training Example(in that domain/task)
1	Vacuum Cleaner [CF]	This vacuum cleaner *sucks* !!!
2	Desktop [KT]	The keyboard is clicky .
3	Tablet [KT]	The soft keyboard is hard to use.
4 (new task)	Laptop	The new keyboard *sucks* and is hard to click!

We can see both forgetting (CF) prevention and knowledge transfer (KT) are needed

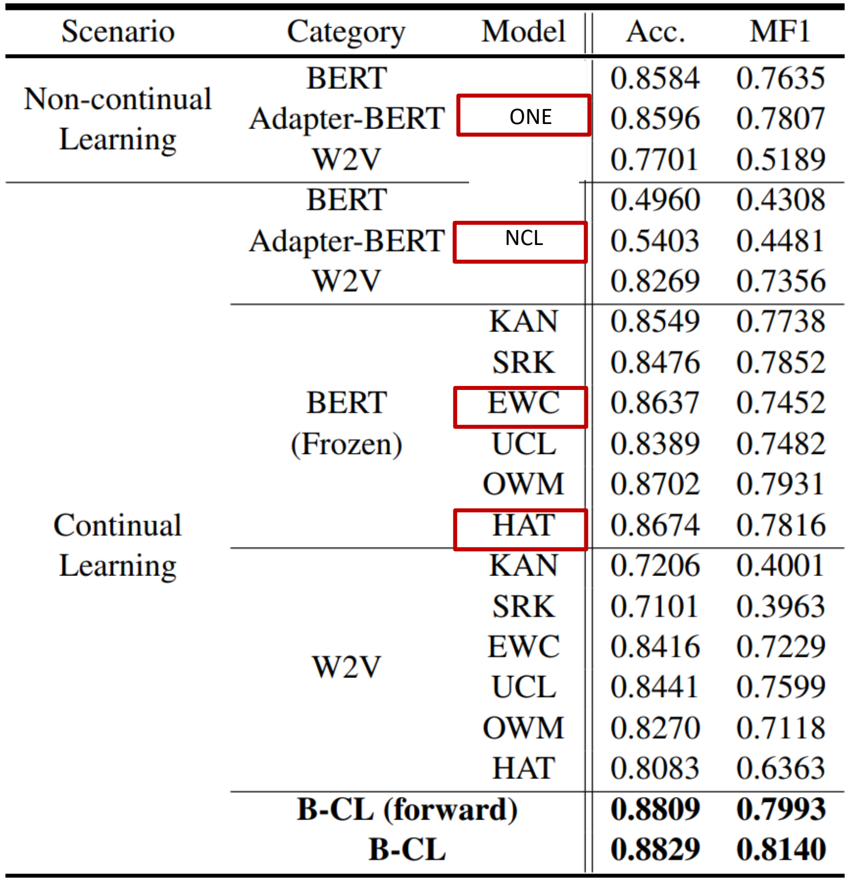
[7]: Ke et al., Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks, NAACL 2021

Architecture-based: beyond forgetting prevention

- BCL
  - How does it perform
    - Datasets
      - Aspect Sentiment Classification
    - Metrics
      - FWT, BWT

Architecture-based: beyond forgetting prevention

- BCL
  - How does it perform
    - BWT
    - There are many baselines that we have already seen (see red boxes)



Scenario	Category	Model	Acc.	MF1
Non-continual Learning	BERT	ONE	0.8584	0.7635
	Adapter-BERT		0.8596	0.7807
	W2V		0.7701	0.5189
Continual Learning	BERT	NCL	0.4960	0.4308
	Adapter-BERT		0.5403	0.4481
	W2V		0.8269	0.7356
	BERT (Frozen)	KAN	0.8549	0.7738
		SRK	0.8476	0.7852
		EWC	0.8637	0.7452
		UCL	0.8389	0.7482
		OWM	0.8702	0.7931
		HAT	0.8674	0.7816
	W2V	KAN	0.7206	0.4001
		SRK	0.7101	0.3963
		EWC	0.8416	0.7229
		UCL	0.8441	0.7599
		OWM	0.8270	0.7118
		HAT	0.8083	0.6363
	B-CL (forward)		**0.8809**	**0.7993**
	B-CL		**0.8829**	**0.8140**

Architecture-based: beyond forgetting prevention

- BCL
  - Pros and Cons?
  - Pros:
    - Deal with both knowledge transfer and forgetting
  - Cons:
    - Only for task incremental learning (TIL)
    - Parameter-efficient fine-tuning comes with cost: the adapter is randomly initialized
      - Sometimes can be hard

Architecture-based: beyond forgetting prevention

- Is this enough?
  - We have seen in ASC that the tasks in the task sequence can be similar and dissimilar
    - However, most of them are still similar (e.g., many sentiment words are shared across domains)
  - In practice, there can be a *mixed* sequence of tasks
    - Mix of *similar* tasks and dissimilar tasks
    - The CL model is supposed to learn well even under this challenging setting
    - Can we address this?
      - Avoid forgetting for dissimilar tasks
      - Enable transfer for similar tasks

Architecture-based: beyond forgetting prevention

- CAT[8]
  - Idea
    - Based on task similarity, the model has different focuses
      - For similar tasks, CAT focuses on knowledge transfer
      - For dissimilar tasks, CAT focuses on forgetting prevention
  - How does it work?
    - Focus on forgetting prevention is intuitive, we can use HAT
    - Focus on knowledge transfer becomes intuitive as well
      - Because we know what tasks are similar, we can simply share their parameters
    - But how do we detect task similarity?

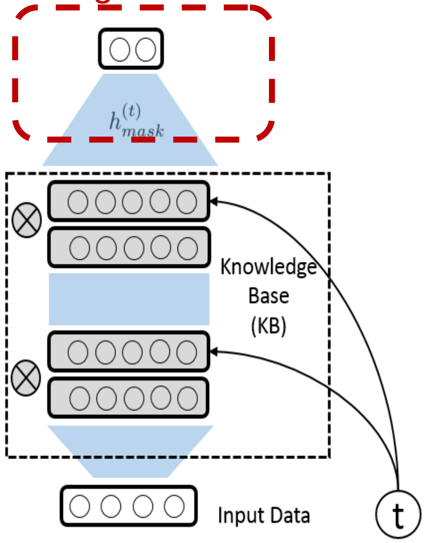
[8]: Ke et al., Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks, NeurIPS 2020

Architecture-based: beyond forgetting prevention

- CAT[8]
  - But how do we detect task similarity?

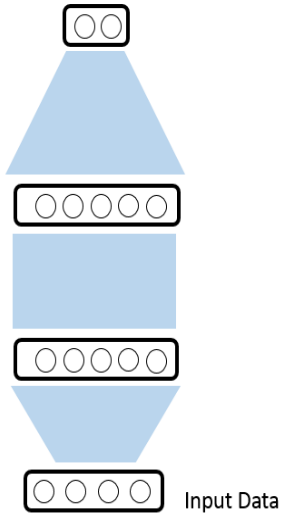
Only a small readout function is trainable

Check whether each task *i's* knowledge is transferable to current task *t*



Transfer Network

A similar independent network trained from scratch



Reference Network

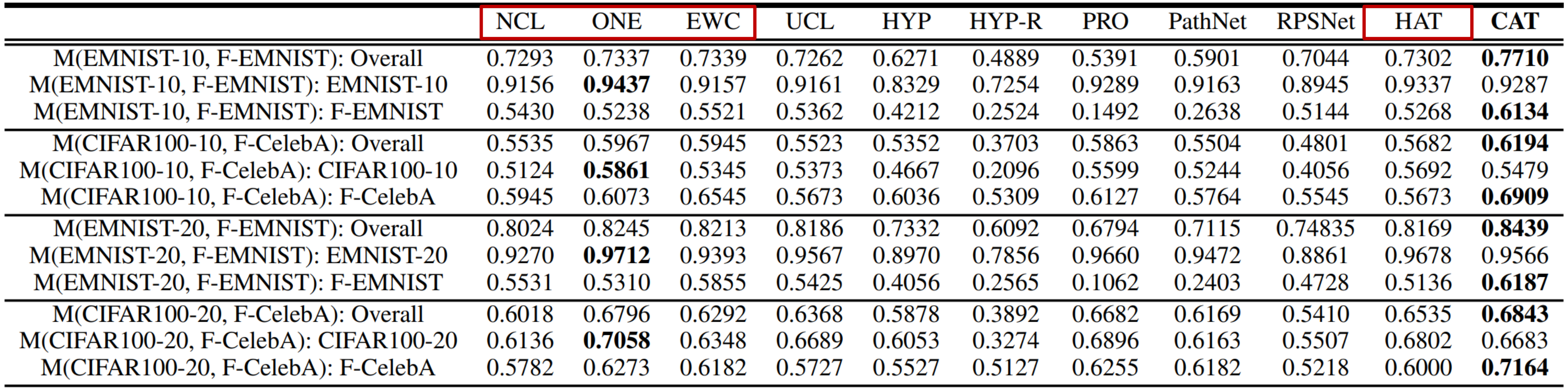
[8]: Ke et al., Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks, NeurIPS 2020

Architecture-based: beyond forgetting prevention

- CAT
  - How does it perform
    - Datasets
      - Similar datasets
        - From Federated learning, F-CelebA, F-EMNIST
      - Dissimilar datasets
        - EMNIST, CIFAR100
    - Metrics
      - FWT, BWT

Architecture-based: beyond forgetting prevention

- CAT
  - How does it perform
    - BWT
    - We have seen many of these baselines before (see red boxes)



	NCL	ONE	EWC	UCL	HYP	HYP-R	PRO	PathNet	RPSNet	HAT	CAT
M(EMNIST-10, F-EMNIST): Overall	0.7293	0.7337	0.7339	0.7262	0.6271	0.4889	0.5391	0.5901	0.7044	0.7302	**0.7710**
M(EMNIST-10, F-EMNIST): EMNIST-10	0.9156	**0.9437**	0.9157	0.9161	0.8329	0.7254	0.9289	0.9163	0.8945	0.9337	0.9287
M(EMNIST-10, F-EMNIST): F-EMNIST	0.5430	0.5238	0.5521	0.5362	0.4212	0.2524	0.1492	0.2638	0.5144	0.5268	**0.6134**
M(CIFAR100-10, F-CelebA): Overall	0.5535	0.5967	0.5945	0.5523	0.5352	0.3703	0.5863	0.5504	0.4801	0.5682	**0.6194**
M(CIFAR100-10, F-CelebA): CIFAR100-10	0.5124	**0.5861**	0.5345	0.5373	0.4667	0.2096	0.5599	0.5244	0.4056	0.5692	0.5479
M(CIFAR100-10, F-CelebA): F-CelebA	0.5945	0.6073	0.6545	0.5673	0.6036	0.5309	0.6127	0.5764	0.5545	0.5673	**0.6909**
M(EMNIST-20, F-EMNIST): Overall	0.8024	0.8245	0.8213	0.8186	0.7332	0.6092	0.6794	0.7115	0.74835	0.8169	**0.8439**
M(EMNIST-20, F-EMNIST): EMNIST-20	0.9270	**0.9712**	0.9393	0.9567	0.8970	0.7856	0.9660	0.9472	0.8861	0.9678	0.9566
M(EMNIST-20, F-EMNIST): F-EMNIST	0.5531	0.5310	0.5855	0.5425	0.4056	0.2565	0.1062	0.2403	0.4728	0.5136	**0.6187**
M(CIFAR100-20, F-CelebA): Overall	0.6018	0.6796	0.6292	0.6368	0.5878	0.3892	0.6682	0.6169	0.5410	0.6535	**0.6843**
M(CIFAR100-20, F-CelebA): CIFAR100-20	0.6136	**0.7058**	0.6348	0.6689	0.6053	0.3274	0.6896	0.6163	0.5507	0.6802	0.6683
M(CIFAR100-20, F-CelebA): F-CelebA	0.5782	0.6273	0.6182	0.5727	0.5527	0.5127	0.6255	0.6182	0.5218	0.6000	**0.7164**

Architecture-based: beyond forgetting prevention

- CAT
  - Pros and Cons?
  - Pros:
    - Handle both transfer and forgetting, even in a mixed sequence scenario
  - Cons:
    - Only for TIL
    - Need to train an additional reference network
      - This is not possible when it comes to pre-trained network

Architecture-based: beyond TIL

- Can we go beyond TIL?
  - What if I don't have task information in testing?
    - Can we still use architecture-based methods?
    - Yes! But some adaptation is needed.
    - CLASSIC[9] is an example
      - More accurately, we will see this is for domain-incremental learning

[9]: Ke et al., Ke et al.,CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks. In EMNLP, 2021, NeurIPS 2020

Architecture-based: beyond TIL

- Background
  - Contrastive loss
    - Become very popular recently. You probably already know that
    - Intuitively
      - Unsupervised learning
      - Push dissimilar representations away
      - Pull similar representations together
      - What is similar/dissimilar (positive/negative) is decided by you

[5]: Ke et al., CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks, EMNLP 2021

Architecture-based: beyond TIL

- Background
  - Contrastive loss
    - Become very popular recently. You probably already know that
    - Mathematically
      - Maximize the mutual information
      - *Alignment*: The model can learn the invariant (shared) knowledge among similar (positive) pairs
      - *Uniformity*: Preserve maximal information via challenging negative samples

[5]: Ke et al., CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks, EMNLP 2021

Architecture-based: beyond TIL

- Background
  - Contrastive loss
    - Why is it attractive to continual learning?
      - We want to pull together similar knowledge from similar tasks while push away dissimilar tasks

[5]: Ke et al., CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks, EMNLP 2021

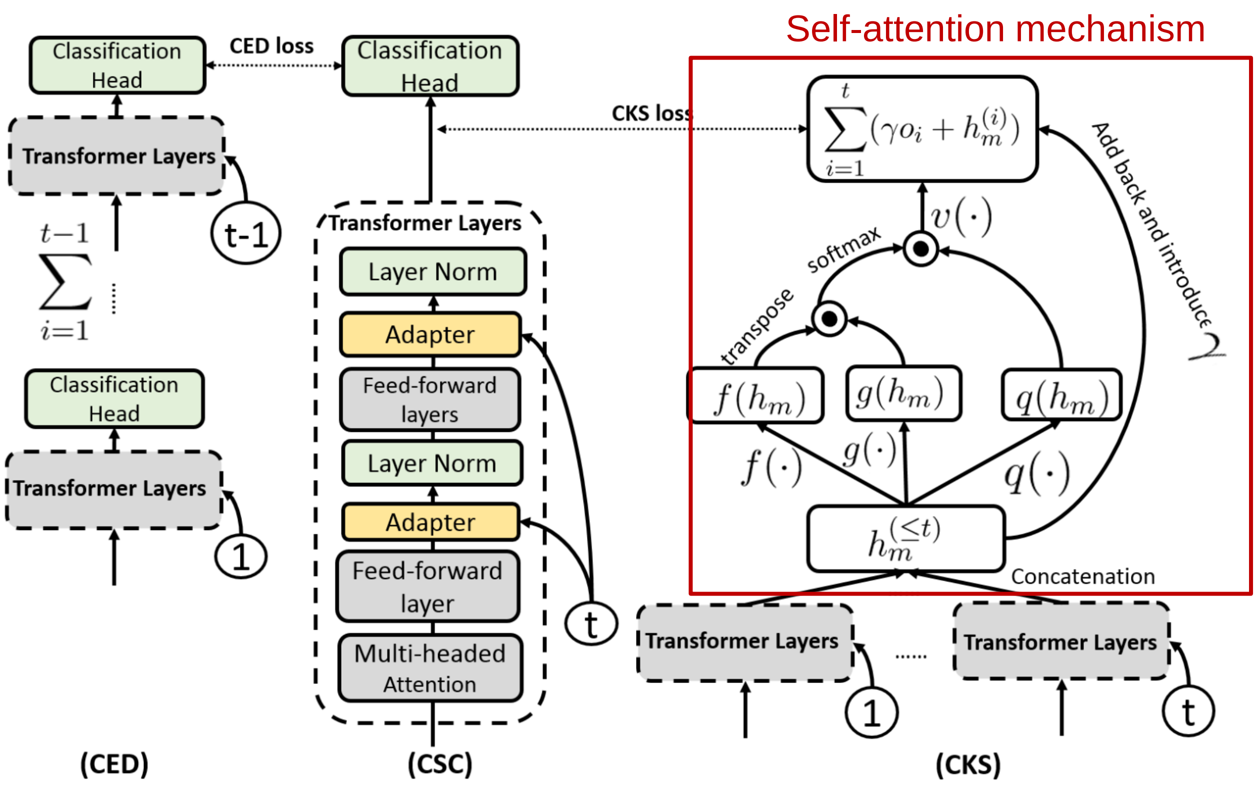
Architecture-based: beyond TIL

- CLASSIC[9]
  - Idea
    - Contrastive loss to deal with similar and dissimilar tasks
    - Knowledge distillation to accumulate **all** previous knowledge into the final/last model, so that we do not need task information in testing
  - How does it work?

[5]: Ke et al., CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks, EMNLP 2021

Architecture-based: beyond TIL

- CLASSIC[9]
  - How does it work?
    - CED as distillation
    - Train a self-attention to extract similar knowledge from different tasks. This serves as positive samples



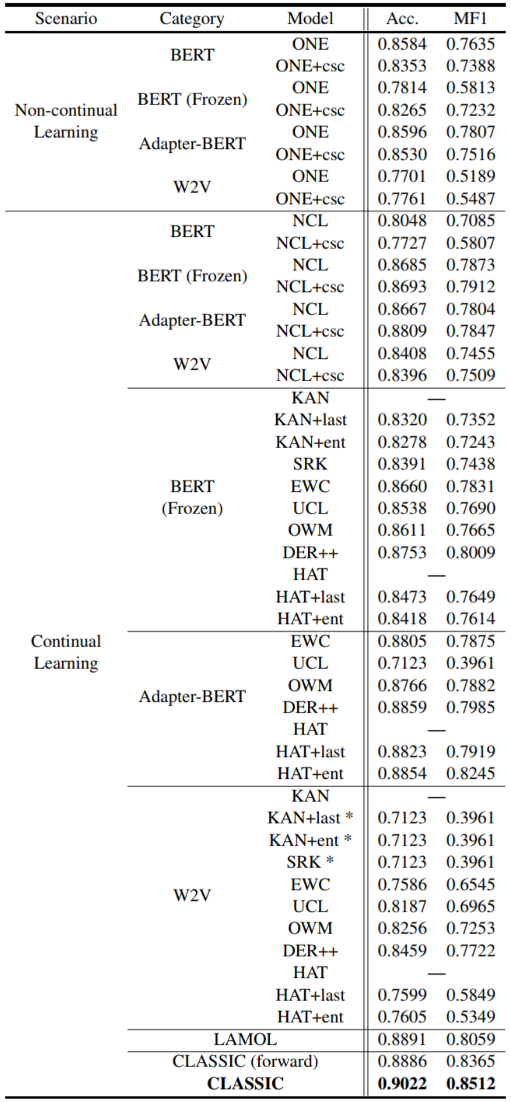
[5]: Ke et al., CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks, EMNLP 2021

Architecture-based: beyond TIL

- CLASSIC
  - How does it perform
    - Datasets
      - Aspect Sentiment Classification
    - Metrics
      - FWT, BWT,

Architecture-based: beyond TIL

- CLASSIC
  - How does it perform
    - Task information is unknown in testing
    - Label is shared across tasks
    - A.k.a. Domain-incremental learning



Scenario	Category	Model	Acc.	MF1
Non-continual Learning	BERT	ONE	0.8584	0.7635
		ONE+csc	0.8353	0.7388
	BERT (Frozen)	ONE	0.7814	0.5813
		ONE+csc	0.8265	0.7232
	Adapter-BERT	ONE	0.8596	0.7807
		ONE+csc	0.8530	0.7516
	W2V	ONE	0.7701	0.5189
		ONE+csc	0.7761	0.5487
Continual Learning	BERT	NCL	0.8048	0.7085
		NCL+csc	0.7727	0.5807
	BERT (Frozen)	NCL	0.8685	0.7873
		NCL+csc	0.8693	0.7912
	Adapter-BERT	NCL	0.8667	0.7804
		NCL+csc	0.8809	0.7847
	W2V	NCL	0.8408	0.7455
		NCL+csc	0.8396	0.7509
	BERT (Frozen)	KAN	—	
		KAN+last	0.8320	0.7352
		KAN+ent	0.8278	0.7243
		SRK	0.8391	0.7438
		EWC	0.8660	0.7831
		UCL	0.8538	0.7690
		OWM	0.8611	0.7665
		DER++	0.8753	0.8009
		HAT	—	
		HAT+last	0.8473	0.7649
		HAT+ent	0.8418	0.7614
	Adapter-BERT	EWC	0.8805	0.7875
		UCL	0.7123	0.3961
		OWM	0.8766	0.7882
		DER++	0.8859	0.7985
		HAT	—	
		HAT+last	0.8823	0.7919
		HAT+ent	0.8854	0.8245
	W2V	KAN	—	
		KAN+last *	0.7123	0.3961
		KAN+ent *	0.7123	0.3961
		SRK *	0.7123	0.3961
		EWC	0.7586	0.6545
		UCL	0.8187	0.6965
		OWM	0.8256	0.7253
		DER++	0.8459	0.7722
		HAT	—	
		HAT+last	0.7599	0.5849
		HAT+ent	0.7605	0.5349
	LAMOL		0.8891	0.8059
	CLASSIC (forward)		0.8886	0.8365
	CLASSIC		**0.9022**	**0.8512**

Architecture-based: beyond TIL

- CLASSIC
  - Pros and Cons?
  - Pros:
    - Beyond TIL, first technique for DIL
    - Have mechanisms to deal with both similar and dissimilar tasks
  - Cons:
    - Cannot deal with unshared label scenario (CIL)
    - It relies on fixed LM.
      - In some cases, this is useful (e.g., fine-tuning)
      - However, in some other cases, we do need to update the LM (e.g., pre-training)

What's next? More recent work

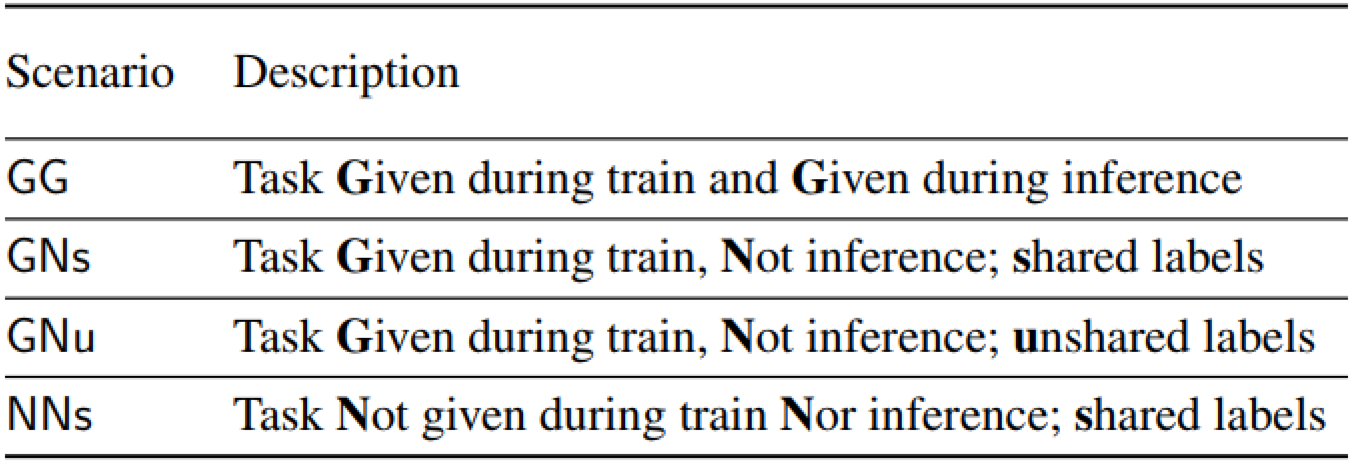
- So far, we have learned 9 different CL models
  - Including their ideas, implementations, evaluations, pros and cons
  - Replayed-based
    - GEM, LAMOL
  - Regularization-based
    - DER++
    - EWC
  - Architecture-based
    - HAT, BCL, CAT, CLASSIC
    - Supsup

What’s next? More recent work

- So far, we have learned 9 different CL models
  - Want to have some hands-on experience and write some codes?
    - Go to
      - https://github.com/ZixuanKe/PyContinual
- What is next?
  - Let us to see some more recent directions/ideas very briefly
    - Beyond Supervised Learning
    - Online Continual Learning (Prof. Liu)

What’s next? More recent work

- Beyond Supervised Learning
  - We already know
    - Task-incremental, Class-incremental and Domain incremental
  - There is a more general categorization



Scenario	Description
GG	Task **G**iven during train and **G**iven during inference
GNs	Task **G**iven during train, **N**ot inference; **s**hared labels
GNu	Task **G**iven during train, **N**ot inference; **u**nshared labels
NNs	Task **N**ot given during train **N**or inference; **s**hared labels

More recent work: Beyond Supervised Learning

- For example, *continual pre-training*
  - We know that the LM have become the basic component of NLP applications
  - However, as time goes by, we need to update the LM
    - This is naturally a continual learning scenario
  - How to achieve continual pre-training?
    - GN (it should be useful for any end-tasks)
    - Unsupervised/self-supervised loss: Masked Language Model
    - A different goal
      - Instead of preserving the knowledge for a specific task
      - We need to preserve the transferrable knowledge

More recent work: Beyond Supervised Learning

- For example, continual pre-training
  - Still in infancy
    - Some recent work on
      - Dataset[10,11]
      - Analysis
        - Comparing different CL models[12]
        - A model relies on a very large memory and Non-SoTA language model[13]

[10]: Jang et al., Towards Continual Knowledge Learning of Language Models, ICLR 2022

[11]: Lazaridou et al., Mind the Gap: Assessing Temporal Generalization in Neural Language Models, NeurIPS 2021 (spotlight)

[12]: Jin et al., Lifelong Pretraining: Continually Adapting Language Models to Emerging Corpora, Arkiv, 2022

[13]: Qin et al., ELLE: Efficient Lifelong Pre-training for Emerging Data, Findings of ACL 2022

That’s it

- Thank you
- Q&A

Topics

- Lifelong or continual learning
- Early research on lifelong learning
- Continual learning based on deep neural networks
- Continual learning in the open-world
- Summary

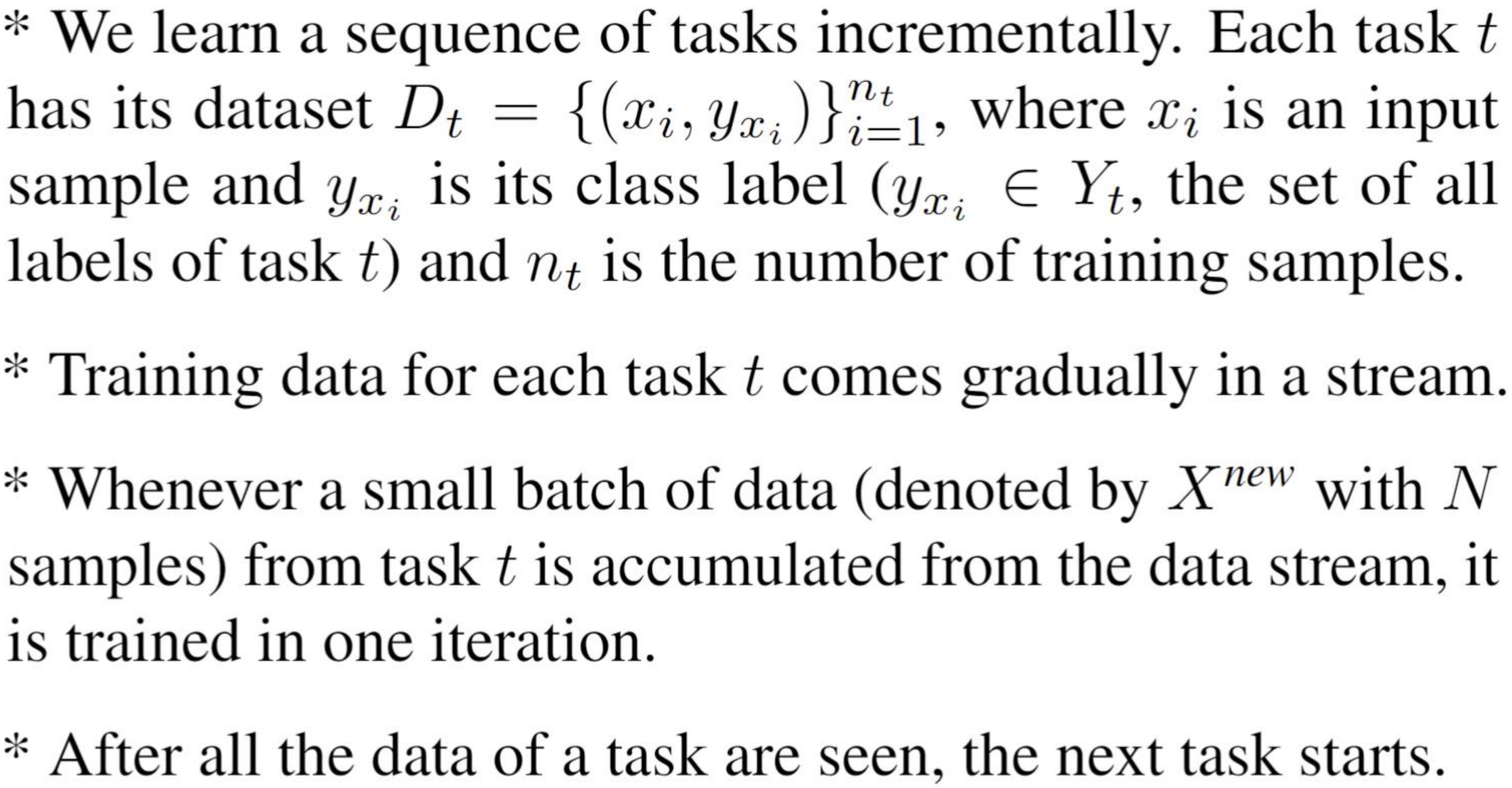
Sub-topics

- Continual learning based on deep neural networks
    - Batch continual learning
    - Online continual learning

Recall: Batch and online continual learning

- Batch continual learning
  - When a new task arrives, all its training data are available
  - Training can use any number of epochs
- Online continual learning
  - The data comes in a data stream.
  - The data for each task comes in gradually. When a small batch of data is accumulated, it is learned in one iteration.
  - Training is effectively done in one epoch.

Online CL: Problem statement



* We learn a sequence of tasks incrementally. Each task t has its dataset $D_t = \{(x_i, y_{x_i})\}_{i=1}^{n_t}$, where x_i is an input sample and y_{x_i} is its class label ($y_{x_i} \in Y_t$, the set of all labels of task t) and n_t is the number of training samples.

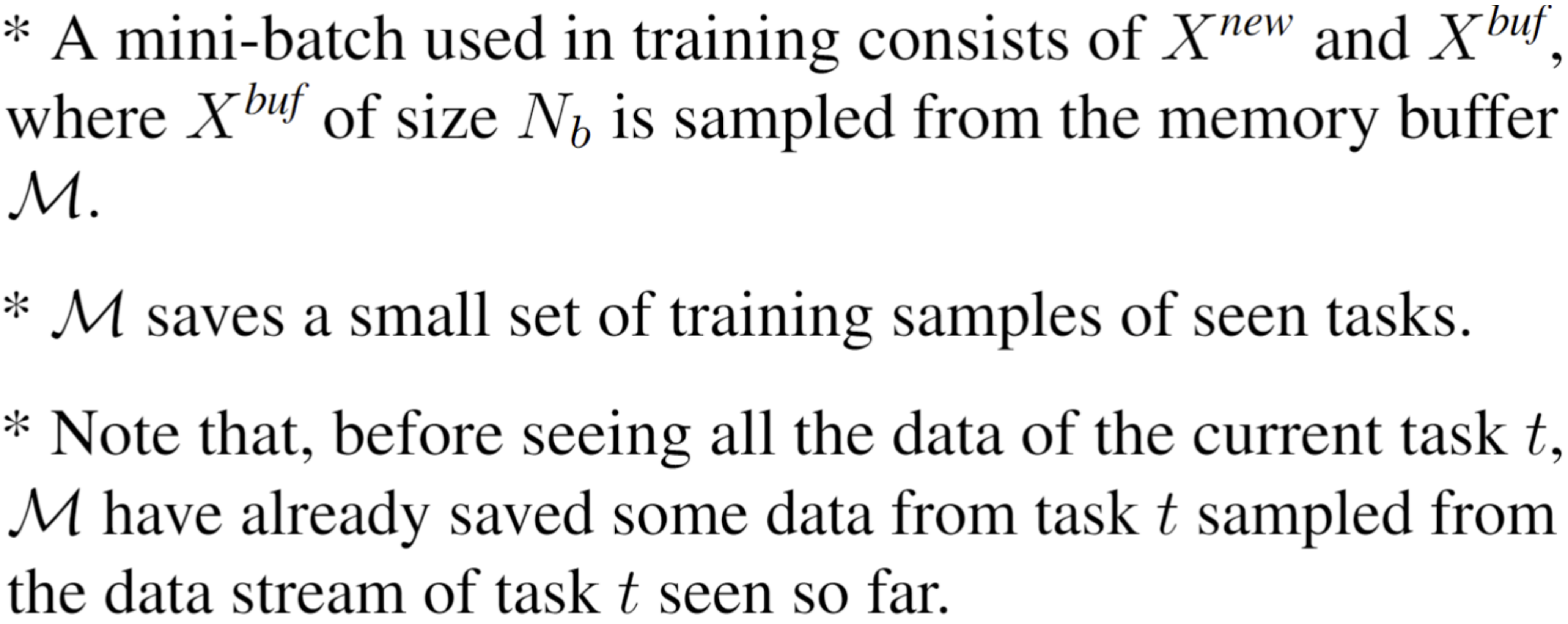
* Training data for each task t comes gradually in a stream.

* Whenever a small batch of data (denoted by X^{new} with N samples) from task t is accumulated from the data stream, it is trained in one iteration.

* After all the data of a task are seen, the next task starts.

Training is done in one epoch

Online CL: Replay approach (most popular)



* A mini-batch used in training consists of X^{new} and X^{buf}, where X^{buf} of size N_b is sampled from the memory buffer $\mathcal{M}$.

* $\mathcal{M}$ saves a small set of training samples of seen tasks.

* Note that, before seeing all the data of the current task t, $\mathcal{M}$ have already saved some data from task t sampled from the data stream of task t seen so far.

Difference of replay: batch CL and online CL

- Inter-task CF and Intra-task CF

1. Batch CL only have inter-task CF, but online CL have both
2. Online CL samples replay data continuously from both previous tasks and the current task. Batch CL only saves previous data.
3. In batch CL, when a task arrives, all its training data is available, but for online CL, the data comes gradually.
    - ❑ In batch CL, all the saved replay data can be used in training the new task in any number of epochs.
    - ❑ But in online CL, only X^{buf} is used in one iteration.

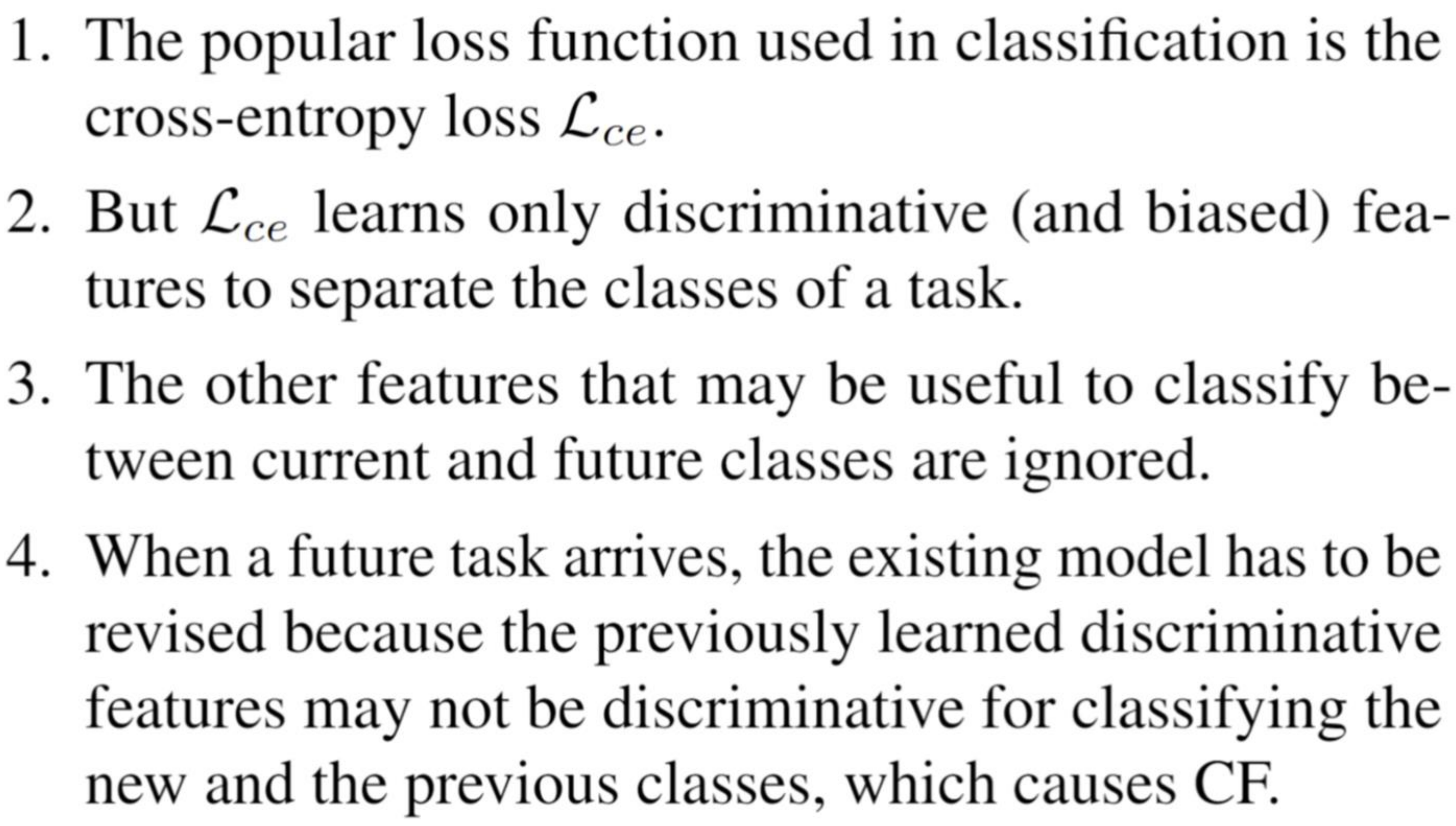
Guo, Liu, and Zhao. Online Continual Learning through Mutual Information Maximization. ICML-2022

Online CL: Mutual information (MI) maximization

- **OCM**: **O**nline **C**ontinual learning based on **M**utual information Maximization.
  - OCM uses the replay approach.
- Objective: dealing with CF in the CIL setting using MI maximization
  - Preventing information/feature loss in representation learning
  - Preserving previously learned knowledge
- OCM has a new training strategy and a new data augmentation method called ***local rotation***.

Guo, Liu, and Zhao. Online Continual Learning through Mutual Information Maximization. ICML-2022

Problem with cross entropy (or other) loss



1. The popular loss function used in classification is the cross-entropy loss $\mathcal{L}_{ce}$.
2. But $\mathcal{L}_{ce}$ learns only discriminative (and biased) features to separate the classes of a task.
3. The other features that may be useful to classify between current and future classes are ignored.
4. When a future task arrives, the existing model has to be revised because the previously learned discriminative features may not be discriminative for classifying the new and the previous classes, which causes CF.

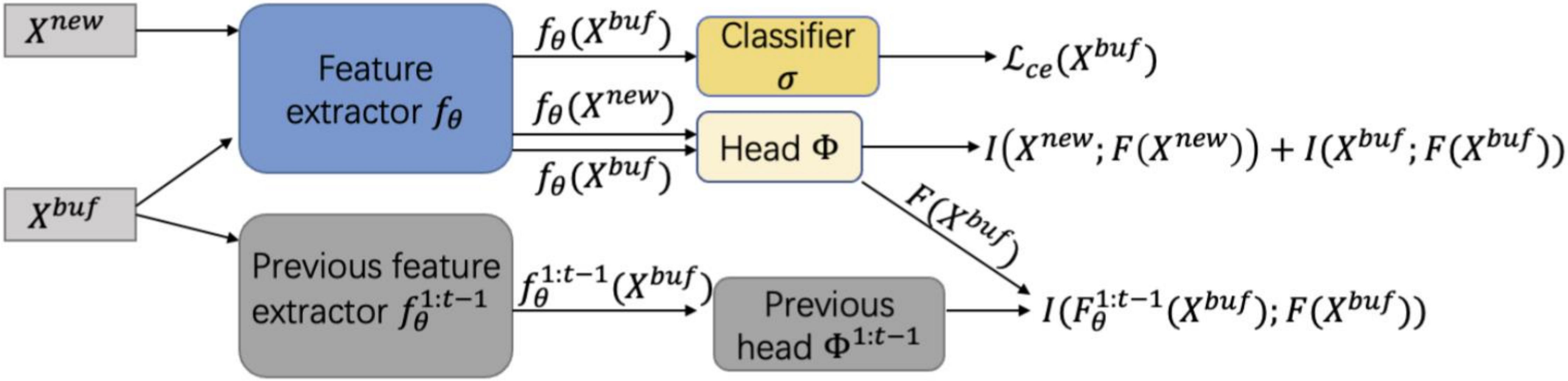
Proposition 1. Minimal cross-entropy does not imply that all possible features are learned.

Proposition 2. Features not learned may cause CF in continual learning.

Remark: We need to learn & use as many features as possible, i.e., learning holistic representations.

Guo, Liu, and Zhao. Online Continual Learning through Mutual Information Maximization. ICML-2022

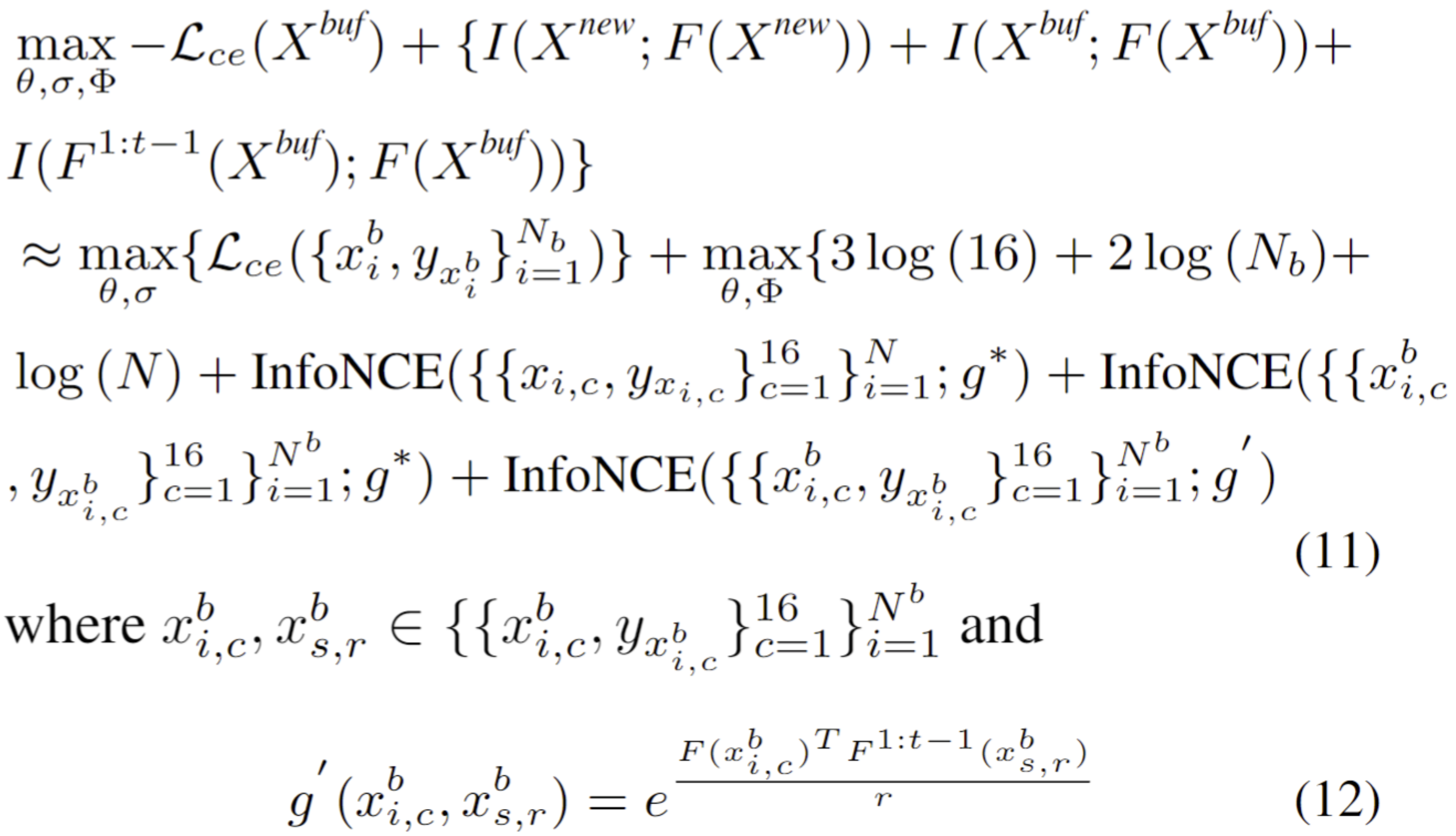
OCM architecture



X^{new}
X^{buf}
Feature extractor f_θ
Previous feature extractor $f_\theta^{1:t-1}$
$f_\theta(X^{buf})$
$f_\theta(X^{new})$
$f_\theta(X^{buf})$
$f_\theta^{1:t-1}(X^{buf})$
Classifier σ
Head Φ
Previous head $\Phi^{1:t-1}$
$\mathcal{L}_{ce}(X^{buf})$
$I(X^{new}; F(X^{new})) + I(X^{buf}; F(X^{buf}))$
$F(X^{buf})$
$I(F_\theta^{1:t-1}(X^{buf}); F(X^{buf}))$

Figure 1. Architecture of OCM. $\mathcal{L}_{ce}$: cross-entropy loss.

The final objective



$$\max_{\theta,\sigma,\Phi} -\mathcal{L}_{ce}(X^{buf}) + \{I(X^{new}; F(X^{new})) + I(X^{buf}; F(X^{buf}))+$$

$$I(F^{1:t-1}(X^{buf}); F(X^{buf}))\}$$

$$\approx \max_{\theta,\sigma}\{\mathcal{L}_{ce}(\{x_i^b, y_{x_i^b}\}_{i=1}^{N_b})\} + \max_{\theta,\Phi}\{3\log(16) + 2\log(N_b)+$$

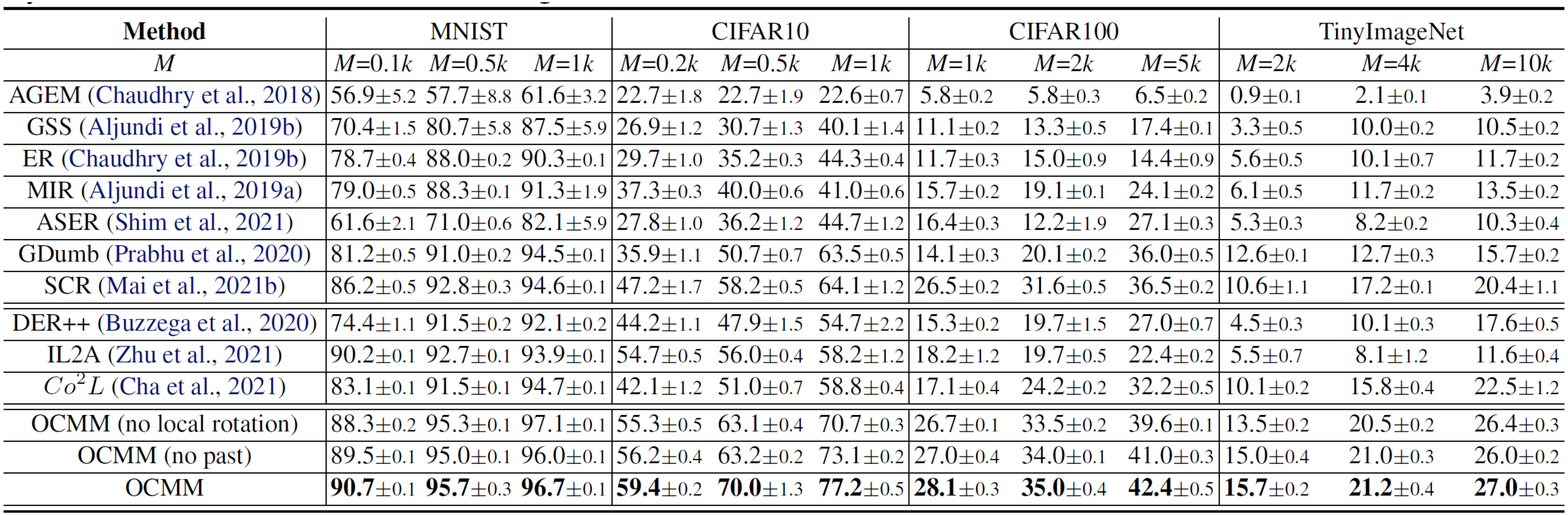
$$\log(N) + \text{InfoNCE}(\{\{x_{i,c}, y_{x_{i,c}}\}_{c=1}^{16}\}_{i=1}^{N}; g^*) + \text{InfoNCE}(\{\{x_{i,c}^b$$

$$, y_{x_{i,c}^b}\}_{c=1}^{16}\}_{i=1}^{N^b}; g^*) + \text{InfoNCE}(\{\{x_{i,c}^b, y_{x_{i,c}^b}\}_{c=1}^{16}\}_{i=1}^{N^b}; g^{'}) \tag{11}$$

where $x_{i,c}^b, x_{s,r}^b \in \{\{x_{i,c}^b, y_{x_{i,c}^b}\}_{c=1}^{16}\}_{i=1}^{N^b}$ and

$$g^{'}(x_{i,c}^b, x_{s,r}^b) = e^{\frac{F(x_{i,c}^b)^T F^{1:t-1}(x_{s,r}^b)}{r}} \tag{12}$$

Experiment results



Method	MNIST			CIFAR10			CIFAR100			TinyImageNet		
M	M=0.1k	M=0.5k	M=1k	M=0.2k	M=0.5k	M=1k	M=1k	M=2k	M=5k	M=2k	M=4k	M=10k
AGEM (Chaudhry et al., 2018)	56.9±5.2	57.7±8.8	61.6±3.2	22.7±1.8	22.7±1.9	22.6±0.7	5.8±0.2	5.8±0.3	6.5±0.2	0.9±0.1	2.1±0.1	3.9±0.2
GSS (Aljundi et al., 2019b)	70.4±1.5	80.7±5.8	87.5±5.9	26.9±1.2	30.7±1.3	40.1±1.4	11.1±0.2	13.3±0.5	17.4±0.1	3.3±0.5	10.0±0.2	10.5±0.2
ER (Chaudhry et al., 2019b)	78.7±0.4	88.0±0.2	90.3±0.1	29.7±1.0	35.2±0.3	44.3±0.4	11.7±0.3	15.0±0.9	14.4±0.9	5.6±0.5	10.1±0.7	11.7±0.2
MIR (Aljundi et al., 2019a)	79.0±0.5	88.3±0.1	91.3±1.9	37.3±0.3	40.0±0.6	41.0±0.6	15.7±0.2	19.1±0.1	24.1±0.2	6.1±0.5	11.7±0.2	13.5±0.2
ASER (Shim et al., 2021)	61.6±2.1	71.0±0.6	82.1±5.9	27.8±1.0	36.2±1.2	44.7±1.2	16.4±0.3	12.2±1.9	27.1±0.3	5.3±0.3	8.2±0.2	10.3±0.4
GDumb (Prabhu et al., 2020)	81.2±0.5	91.0±0.2	94.5±0.1	35.9±1.1	50.7±0.7	63.5±0.5	14.1±0.3	20.1±0.2	36.0±0.5	12.6±0.1	12.7±0.3	15.7±0.2
SCR (Mai et al., 2021b)	86.2±0.5	92.8±0.3	94.6±0.1	47.2±1.7	58.2±0.5	64.1±1.2	26.5±0.2	31.6±0.5	36.5±0.2	10.6±1.1	17.2±0.1	20.4±1.1
DER++ (Buzzega et al., 2020)	74.4±1.1	91.5±0.2	92.1±0.2	44.2±1.1	47.9±1.5	54.7±2.2	15.3±0.2	19.7±1.5	27.0±0.7	4.5±0.3	10.1±0.3	17.6±0.5
IL2A (Zhu et al., 2021)	90.2±0.1	92.7±0.1	93.9±0.1	54.7±0.5	56.0±0.4	58.2±1.2	18.2±1.2	19.7±0.5	22.4±0.2	5.5±0.7	8.1±1.2	11.6±0.4
Co^2L (Cha et al., 2021)	83.1±0.1	91.5±0.1	94.7±0.1	42.1±1.2	51.0±0.7	58.8±0.4	17.1±0.4	24.2±0.2	32.2±0.5	10.1±0.2	15.8±0.4	22.5±1.2
OCMM (no local rotation)	88.3±0.2	95.3±0.1	97.1±0.1	55.3±0.5	63.1±0.4	70.7±0.3	26.7±0.1	33.5±0.2	39.6±0.1	13.5±0.2	20.5±0.2	26.4±0.3
OCMM (no past)	89.5±0.1	95.0±0.1	96.0±0.1	56.2±0.4	63.2±0.2	73.1±0.2	27.0±0.4	34.0±0.1	41.0±0.3	15.0±0.4	21.0±0.3	26.0±0.2
OCMM	**90.7**±0.1	**95.7**±0.3	**96.7**±0.1	**59.4**±0.2	**70.0**±1.3	**77.2**±0.5	**28.1**±0.3	**35.0**±0.4	**42.4**±0.5	**15.7**±0.2	**21.2**±0.4	**27.0**±0.3

Topics

- Lifelong or continual learning
- Early research on lifelong learning
- Continual learning based on deep neural networks
- Continual learning in the open-world
- Summary

Sub-topics

- Continual learning in the open-world
  - Closed world and open world
  - Open world continual learning framework
  - Novelty or out-of-distribution detection
  - Two open world continual learning systems: CML and CILK

What do we want to achieve?

- In almost existing continual learning research, both the tasks and training data are given by the user or the engineers.

- Can the system discover new tasks and also acquire the ground-truth training data by itself.
  - Then the system can become autonomous

Closed-world assumption and open-world learning

(Fei et al, 2016; Shu et al., 2017, Hu et al., 2020)

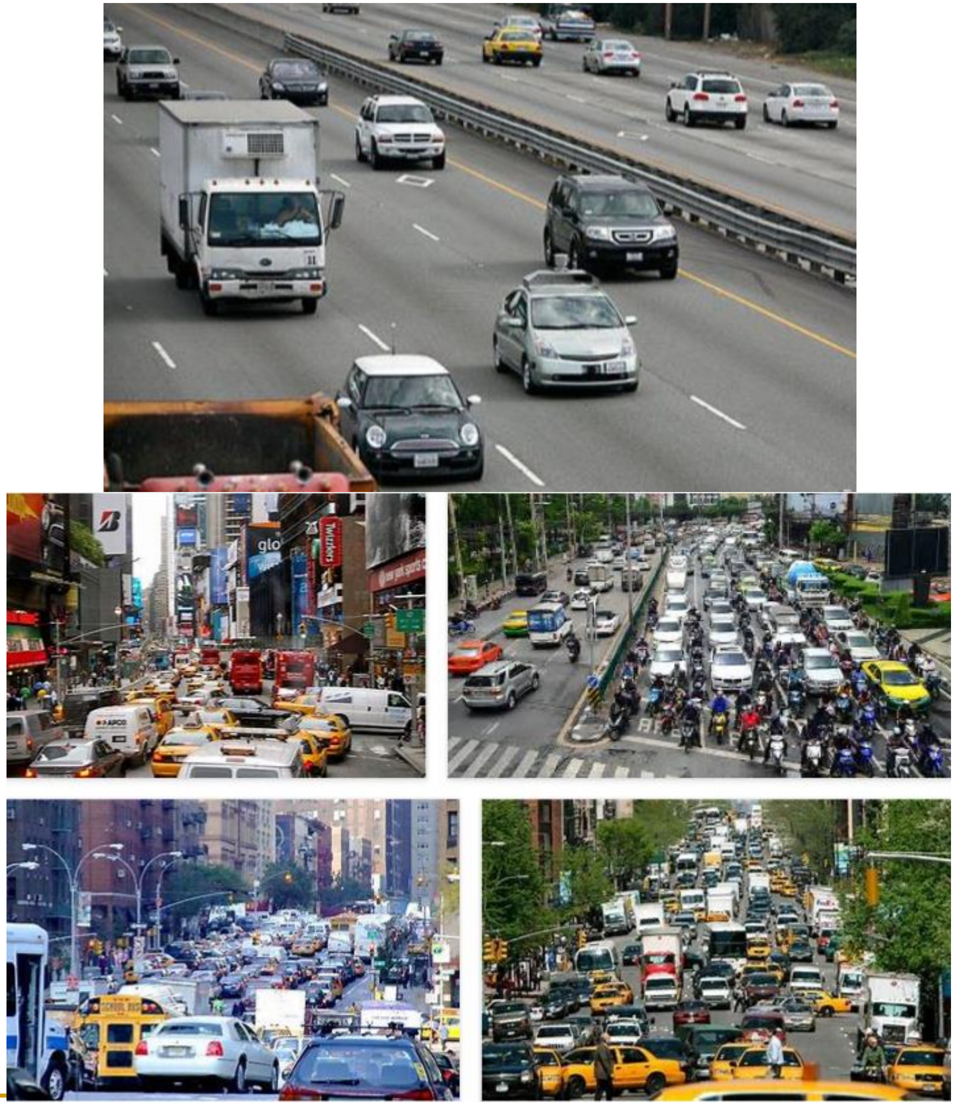
- Traditional machine learning:
  - Training data: $D^{train} = \{D_1, D_2, \ldots, D_t\}$ of classes $Y^{train} = \{l_1, l_2, \ldots, l_t\}$.
  - Test data: $D^{test}, Y^{test} \in \{l_1, l_2, \ldots, l_t\}$
- **Closed-world assumption**: $Y^{test} \subseteq Y^{train}$
  - Classes appeared in testing must have been seen in training, **nothing new.**
  - A system that is **unable to identify anything new**, it cannot learn by itself.
- **Open-world learning**: $Y^{test} - Y^{train} \neq \phi$
  - Training data: $D^{train} = \{D_1, D_2, \ldots, D_t\}$, $Y^{train} = \{l_1, l_2, \ldots, l_t\}$.
  - Test data: $D^{test}, Y^{test} \in \{l_1, l_2, \ldots, l_t, L_0\}$

Fei, and Liu. Breaking the Closed World Assumption in Text Classification. *NAACL-HLT 2016*

Hu, et al. HRN: A Holistic Approach to One Class Learning. NeurIPS-2020.

Self-driving cars need to learn continuously too

- Self-driving cars cannot reach human-level of driving with only rules and off-line training.
  - Impossible to cover all corner cases
  - Real-world is full of unknowns.
- Have to learn & adapt continuously in its interaction with humans and the environment by itself.
  - in the open world (changes & unknowns).



Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2018

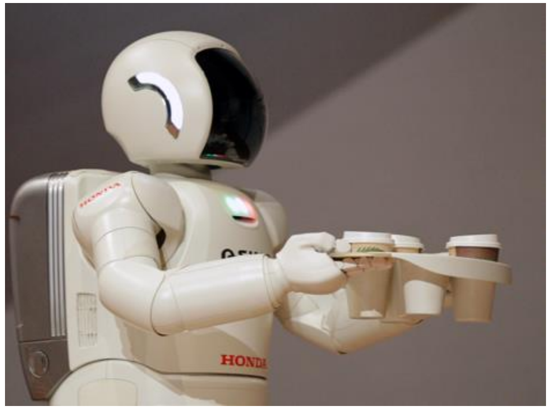
A personal experience with a self-driving car

- I consulted for a self-driving car company for a year.
- Once we took a self-driving car for a field test on the road.
    - At a T-junction, the car suddenly stopped and refused to move.
        - Every direction was clear, and nothing that we can see was on the road.
- We had to take over manually and drove the car to the lab.
    - Debugging found that a sensor detected a pebble on the road.
  - If the car should have said ”*I detected an unknown object on the road. What should I do?*” we would have said “*It is safe. Please go ahead*.”
    - The car can then learn the new object so that it will have no issue next time.
        - That is, **learning on the fly** or on the job.

Chatbots should learn continually after deployment

(Chen & Liu, 2018, Liu, 2020)

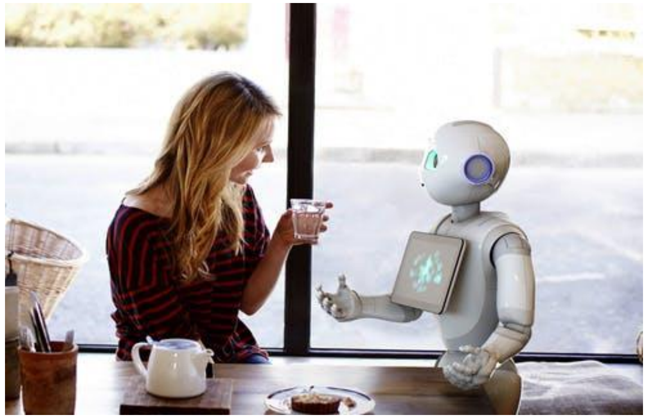
- Chatbot: human users may say things a chatbot does not understand.
  - It must learn new knowledge and new language expressions **during chatting**.
    - E.g., asking the current or other users.
  - Humans learn a great deal in our daily conversations
- Chatbots **should not** solely rely on offline training initiated by engineers.



Liu. Learning on the Job: Online Lifelong and Continual Learning. AAAI-2020

Lifelong learning chatbots: Scopes

- **Learning from external sources**
  - ✓ Learning by reading web corpus, web tables or past conversation [information extraction], e.g., NELL.



- **Interactive learning**
  - ✓ Learning through interactive multi-turn dialogue [**our focus**]

Liu. *Learning on the Job: Online Lifelong and Continual Learning*. AAAI - 2020

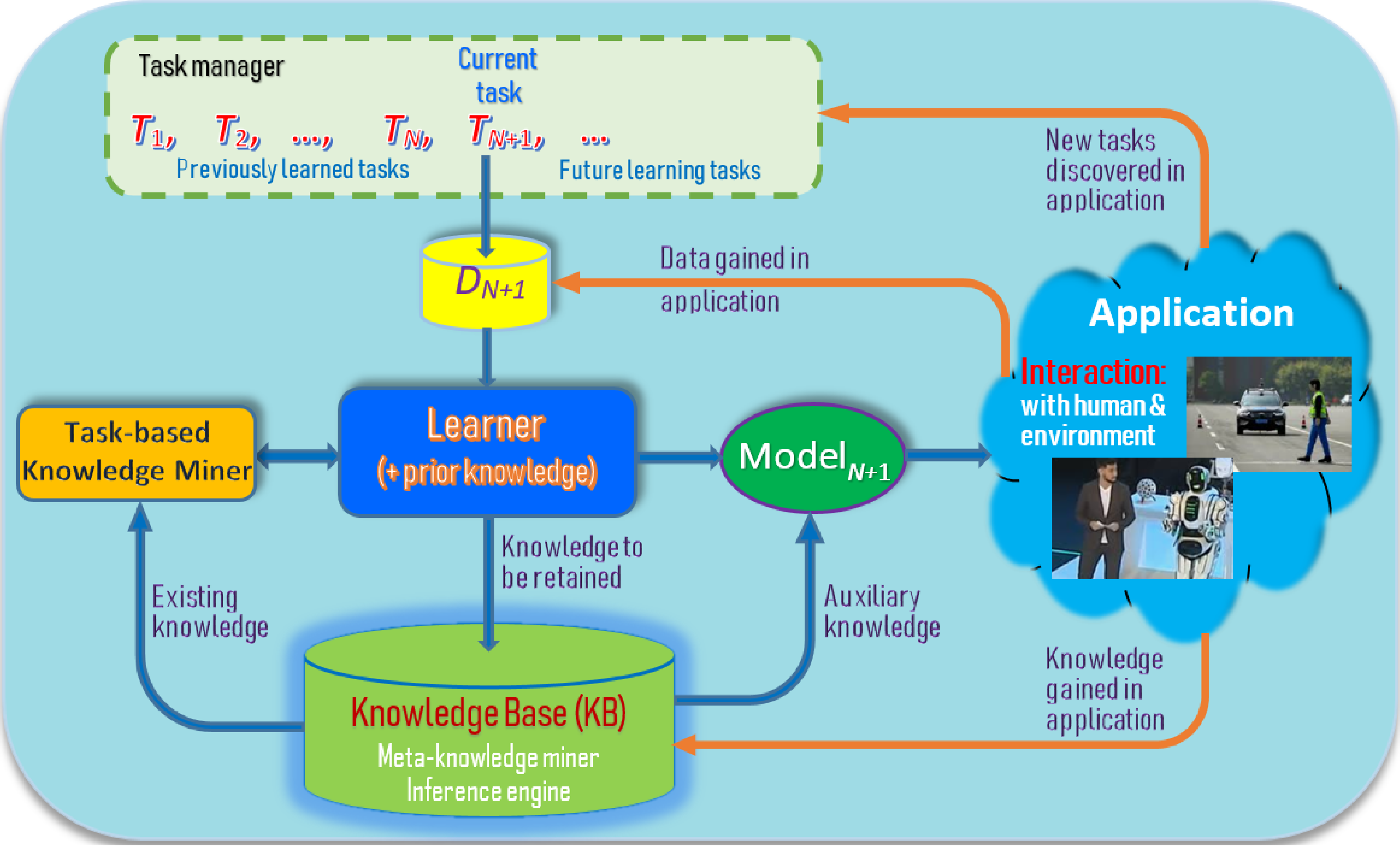
Liu and Mazumder. *Lifelong and Continual Learning Dialogue Systems: Learning during Conversation.* AAAI-2021

Sub-topics

- Continual learning in the open-world
  - Closed world and open world
  - Open world continual learning framework
  - Novelty or out-of-distribution detection
  - Two open world continual learning systems: CML and CILK

Open-world continual learning (AI Autonomy)

(Fei et al 2016, Shu et al 2017a, 2017, Chen & Liu, 2018, Liu, 2020)



Orange lines:

Learning after model deployment

- Learning on the job

Liu. Learning on the Job: Online Lifelong and Continual Learning. AAAI-2020

Characteristics of continual learning

(Chen and Liu, 2018, Liu, 2020)

- **Continuous and incremental learning process** (no forgetting)
- **Knowledge accumulation in KB** (long-term memory)
- **Knowledge transfer/adaptation** (across tasks)

- **Learning after deployment (on the job)**. *Self-supervision* using the *accumulated knowledge* and *interaction* with **humans** & **environment.**

  Main steps:

  - ❑ Identify new tasks to learn (open-world learning or OOD detection)
  - ❑ Acquire ground-truth training data (collecting training data interactively)
  - ❑ Learn the tasks incrementally (continual learning)

Liu. Learning on the Job: Online Lifelong and Continual Learning. AAAI-2020

Ke, Liu, and Huang. Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks. NeurIPS-2020

Learning on the job (after model deployment)

(Liu, 2020, Chen and Liu, 2018)

- It is estimated that about 70% of our human knowledge comes from ‘on-the-job’ learning.
  - Only about 10% through formal training
  - The rest 20% through observation of others
- An AI agent should learn on the job too as
  - The world is too complex and constantly changing.
    - Have to learn continually and adapt
  - Without this capability, an AI agent is not truly intelligent.

Liu. Learning on the Job: Online Lifelong and Continual Learning. AAAI-2020

Example 1 – a chatbot system

(Liu and Mei, 2020; Liu and Mazumder, 2021)

- Session 1
  - User-1: Hey, I visited Stockholm last week. The place is awesome!
  - Chatbot: Where is Stockholm?
  - User-1: Stockholm is the capital of Sweden.

- Session 2
  - User-2: I am planning a tour to Europe next month.
  - Chatbot: Are you visiting Stockholm? I heard it is a nice place.

Liu and Mazumder. Lifelong and Continual Learning Dialogue Systems: Learning during Conversation. AAAI-2021

Example 2 - a greeting bot in a hotel

(Chen and Liu, 2018)

- See an existing guest.
  - Bot: “Hello John, how are you today?”
- See a new guest - recognize he/she is new **(OOD and create a new task)**
  - Bot: “Welcome to our hotel! What is your name, sir?” **(get class label)**
  - Guest: “David” **(got class label: David)**
  - Bot learns to recognize David automatically
    - ❑ take pictures of David **(get training data)**
    - ❑ learn to recognize David **(continual learning)**
- See David next time.
  - Bot: “Hello David, how are you today?” **(use the new knowledge)**

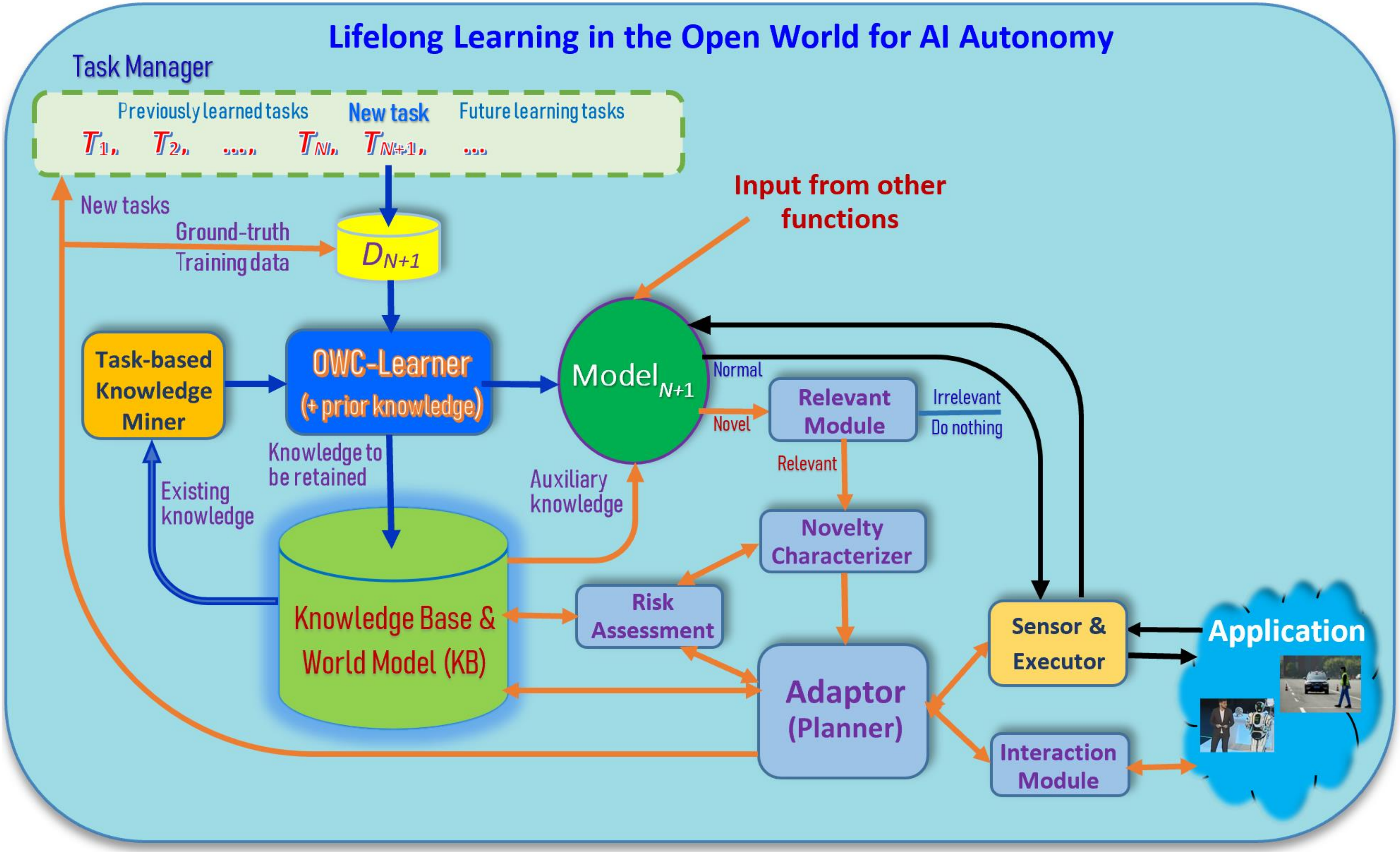
Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2015, 2018

Example (cont.)

- In a real hotel, the situation is much more complex.
  - How does the bot know that the novel object is a new guest?
    - Is the object a person, an animal, or a piece of furniture?
      - needs to use existing knowledge to characterize the novel object!
  - Different **characterizations** require different responses (**adaptation** or **accommodation** strategies)? E.g.,
    - If it looks like an animal, report to a hotel staff.
    - If it looks like a policeman, do nothing
    - If it looks like a hotel guest (with luggage), ask for his/her name: "*Welcome to our hotel! What is your name, sir?*" and learn to recognize him/her

Liu, Robertson, Grigsby, and Mazumder. Self-Initiated Open World Learning for Autonomous AI Agents. AAAI Spring Symposium, 2022
Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2015, 2018.

Lifelong/continual learning in the open world



- This paradigm is for

  AI Autonomy

Self-initiated **O**pen-world Continual **L**earning and **A**daptation (**SOLA**)

- An agent is represented as ***a pair*** (*T*, *S*),
  - *T* is the primary task-performer (e.g., dialogue system of the greeting bot)
  - *S* is a set of supporting functions (e.g., vision system and speech system) that supports the primary task-performer.
- *T* or each $S_i \in S$ has its own seven sub-systems (*L*, *R*, *C*, *A*, *K*, *I*, *E*),
  - ***L***: **open-world continual learner** (classification, novelty detection & continual learning)
  - ***R***: ***relevance* module** that decides if a detected novelty is relevant to agent's task
  - ***C***: **novelty characterizer** that characterizes the novelty
  - ***A***: **adaptor** that adapts to novelty – characterizing novelty and formulating response
  - ***K***: **risk assessment** and management module
  - ***I*** : **interactive module** to communicate with humans or other agents
  - ***E***: action **executor**

Liu, Robertson, Grigsby, and Mazumder. Self-Initiated Open World Learning for Autonomous AI Agents. AAAI Spring Symposium, 2022.

Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2015, 2018.

Novelty characterization and adaptation

- Characterization: a description of the novel object based on the agent's existing knowledge.
  - Characterization at different levels of details => more or less precise responses.
    - Often done based on ***similarity***:
      - E.g., *it looks like an animal* (general), or *it looks like a dog* (more specific).
    - **Attributes/properties**: e.g., a moving object, speed and direction of moving.
- Adapting to novelty: a pair (*Characterization*, *Response*)
  - Response: According to characterization, formulate a specific course of actions to respond to the novelty, e.g.,
    - If novel object looks like an animal (characterization), report to hotel staff (response).
    - If ***cannot characterize***, take ***default action*** (e.g., do nothing)
  - Enable continual learning (see next slide)
- Risk assessment: each decision carries risks

Liu, Robertson, Grigsby, and Mazumder. Self-Initiated Open World Learning for Autonomous AI Agents. AAAI Spring Symposium, 2022.

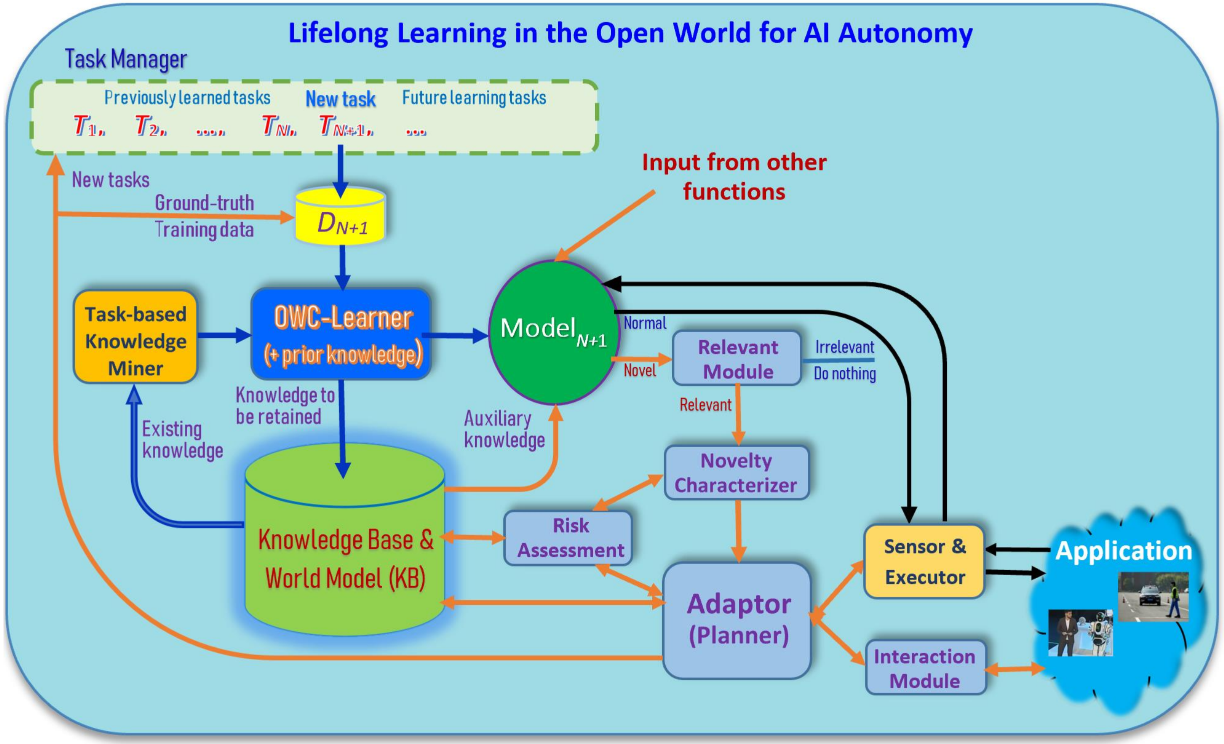
Adaptation enabled continual learning

- **Ask a human and learn**
  - E.g., for the greeting bot, ask the human using the interactive module ***I*** (in natural language) to get ground-truth data and incrementally learn.
- **Imitation learning**.
  - E.g., on seeing a novel object by a self-driving car, if the car in front drives through it with no issue, the car may choose the same course of action as well and learn it for future use.
- **Perform limited reinforcement learning**.
  - By interacting with the environment through trial-and-error exploration, the agent learns a good response policy for future use.

Liu, Robertson, Grigsby, and Mazumder. Self-Initiated Open World Learning for Autonomous AI Agents. AAAI Spring Symposium, 2022.

(L, R, C, A, K, I, E)

- Every module potentially has the lifelong learning capability
  - Learning everything
  - Learning everywhere
- In a particular system,
  - Not all modules are necessary
  - some modules may be shared:
    - E.g., in a chatbot, Interaction module and Executor may be the same



Liu, Robertson, Grigsby, and Mazumder. Self-Initiated Open World Learning for Autonomous AI Agents. AAAI Spring Symposium, 2022.

Sub-topics

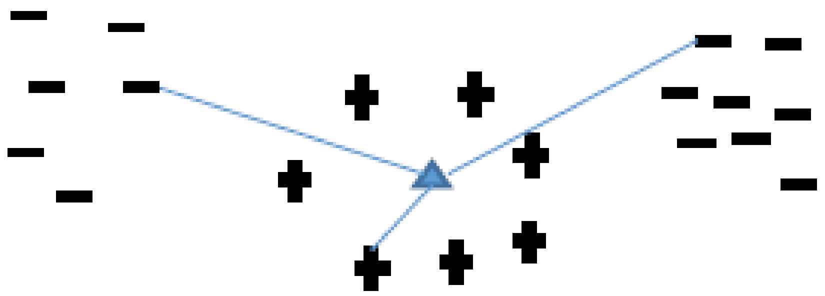
- Continual learning in the open-world
  - Closed world and open world
  - Open world continual learning framework
  - Novelty or out-of-distribution detection
  - Two open world continual learning systems: CML and CILK

One-class OOD detection:

- To detect unseen classes, Fei and Liu (2016) proposed CBS learning:
  - Center-based similarity (CBS) space learning.
- It performs space transformation
  - Each document vector *d* is transformed to a CBS space vector
    1. Compute centers c_i for the positive class
    2. Compute the similarities of each document to c_i.
    - This gives us a new data set (in CSB space).

Fei, Wang, and Liu. Learning Cumulatively to Become More Knowledgeable. KDD-2016

Space Transformation and Learning

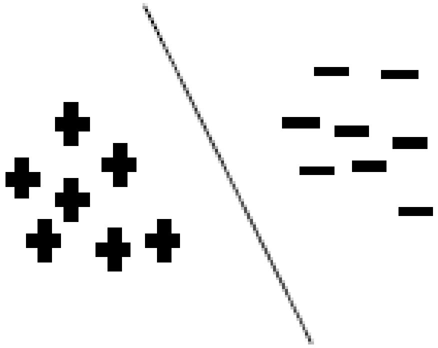


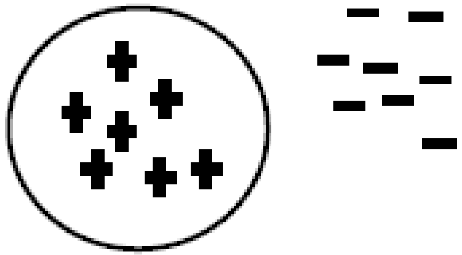
- We can use many similarity measures.
- After space transformation, we can run SVM to build a classification in the CBS space
    - CBS learning basically finds a ball for each class

Traditional learning vs. CBS learning

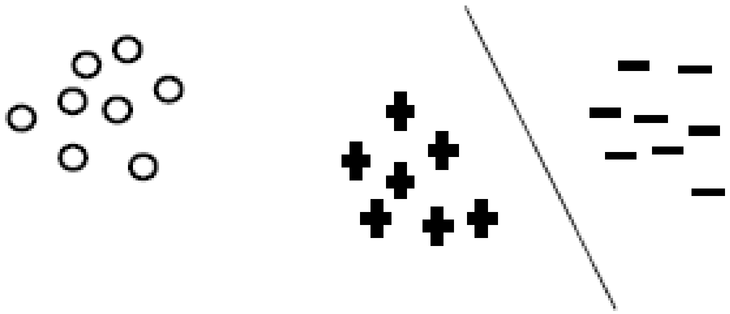
- Traditional learning (using SVM)

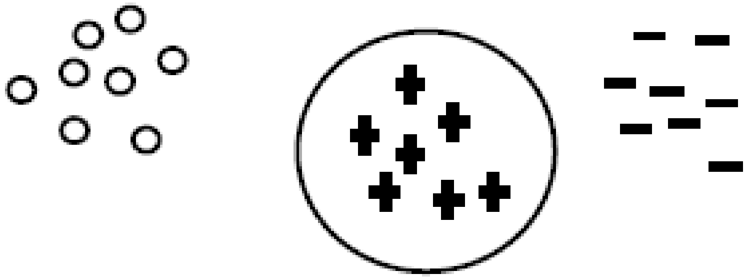
CBS learning





- Classification (testing)





One-class learning: a holistic approach

- It aims to use as many features are possible (holistic representation).
- A one-class loss is proposed.
  - Negative Log Likelihood (NLL) for one-class
  - Holistic regularization (H-regularization)

$$\mathcal{L} = \underbrace{\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}_{\mathbf{x}}} [-\log(\mathrm{Sigmoid}(f(\mathbf{x})))]}_{\text{NLL}} + \lambda \cdot \underbrace{\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}_{\mathbf{x}}} \|\nabla_{\mathbf{x}} f(\mathbf{x})\|_2^n}_{\text{H-regularization}}$$

Hu, Wang, Qin, JMa, and Liu. HRN: A Holistic Approach to One Class Learning. NeurIPS-2020

Results in AUC

- For OOD detection, the common evaluation measure is *Area under the ROC curve* (AUC).

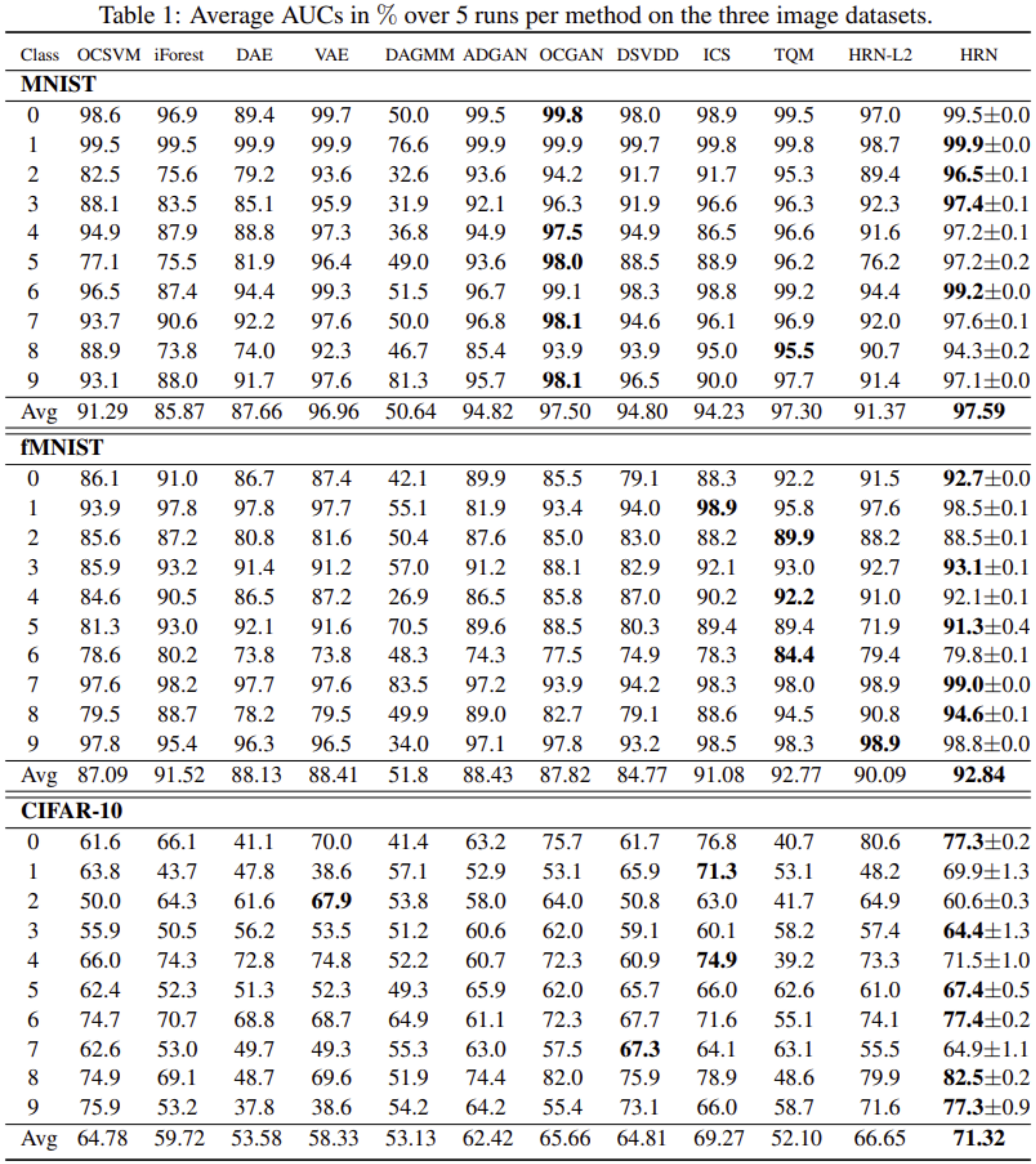


Table 1: Average AUCs in % over 5 runs per method on the three image datasets.

Class	OCSVM	iForest	DAE	VAE	DAGMM	ADGAN	OCGAN	DSVDD	ICS	TQM	HRN-L2	HRN
MNIST												
0	98.6	96.9	89.4	99.7	50.0	99.5	**99.8**	98.0	98.9	99.5	97.0	99.5±0.0
1	99.5	99.5	99.9	99.9	76.6	99.9	99.9	99.7	99.8	99.8	98.7	**99.9**±0.0
2	82.5	75.6	79.2	93.6	32.6	93.6	94.2	91.7	91.7	95.3	89.4	**96.5**±0.1
3	88.1	83.5	85.1	95.9	31.9	92.1	96.3	91.9	96.6	96.3	92.3	**97.4**±0.1
4	94.9	87.9	88.8	97.3	36.8	94.9	**97.5**	94.9	86.5	96.6	91.6	97.2±0.1
5	77.1	75.5	81.9	96.4	49.0	93.6	**98.0**	88.5	88.9	96.2	76.2	97.2±0.2
6	96.5	87.4	94.4	99.3	51.5	96.7	99.1	98.3	98.8	99.2	94.4	**99.2**±0.0
7	93.7	90.6	92.2	97.6	50.0	96.8	**98.1**	94.6	96.1	96.9	92.0	97.6±0.1
8	88.9	73.8	74.0	92.3	46.7	85.4	93.9	93.9	95.0	**95.5**	90.7	94.3±0.2
9	93.1	88.0	91.7	97.6	81.3	95.7	**98.1**	96.5	90.0	97.7	91.4	97.1±0.0
Avg	91.29	85.87	87.66	96.96	50.64	94.82	97.50	94.80	94.23	97.30	91.37	**97.59**
fMNIST												
0	86.1	91.0	86.7	87.4	42.1	89.9	85.5	79.1	88.3	92.2	91.5	**92.7**±0.0
1	93.9	97.8	97.8	97.7	55.1	81.9	93.4	94.0	**98.9**	95.8	97.6	98.5±0.1
2	85.6	87.2	80.8	81.6	50.4	87.6	85.0	83.0	88.2	**89.9**	88.2	88.5±0.1
3	85.9	93.2	91.4	91.2	57.0	91.2	88.1	82.9	92.1	93.0	92.7	**93.1**±0.1
4	84.6	90.5	86.5	87.2	26.9	86.5	85.8	87.0	90.2	**92.2**	91.0	92.1±0.1
5	81.3	93.0	92.1	91.6	70.5	89.6	88.5	80.3	89.4	89.4	71.9	**91.3**±0.4
6	78.6	80.2	73.8	73.8	48.3	74.3	77.5	74.9	78.3	**84.4**	79.4	79.8±0.1
7	97.6	98.2	97.7	97.6	83.5	97.2	93.9	94.2	98.3	98.0	98.9	**99.0**±0.0
8	79.5	88.7	78.2	79.5	49.9	89.0	82.7	79.1	88.6	94.5	90.8	**94.6**±0.1
9	97.8	95.4	96.3	96.5	34.0	97.1	97.8	93.2	98.5	98.3	**98.9**	98.8±0.0
Avg	87.09	91.52	88.13	88.41	51.8	88.43	87.82	84.77	91.08	92.77	90.09	**92.84**
CIFAR-10												
0	61.6	66.1	41.1	70.0	41.4	63.2	75.7	61.7	76.8	40.7	80.6	**77.3**±0.2
1	63.8	43.7	47.8	38.6	57.1	52.9	53.1	65.9	**71.3**	53.1	48.2	69.9±1.3
2	50.0	64.3	61.6	**67.9**	53.8	58.0	64.0	50.8	63.0	41.7	64.9	60.6±0.3
3	55.9	50.5	56.2	53.5	51.2	60.6	62.0	59.1	60.1	58.2	57.4	**64.4**±1.3
4	66.0	74.3	72.8	74.8	52.2	60.7	72.3	60.9	**74.9**	39.2	73.3	71.5±1.0
5	62.4	52.3	51.3	52.3	49.3	65.9	62.0	65.7	66.0	62.6	61.0	**67.4**±0.5
6	74.7	70.7	68.8	68.7	64.9	61.1	72.3	67.7	71.6	55.1	74.1	**77.4**±0.2
7	62.6	53.0	49.7	49.3	55.3	63.0	57.5	**67.3**	64.1	63.1	55.5	64.9±1.1
8	74.9	69.1	48.7	69.6	51.9	74.4	82.0	75.9	78.9	48.6	79.9	**82.5**±0.2
9	75.9	53.2	37.8	38.6	54.2	64.2	55.4	73.1	66.0	58.7	71.6	**77.3**±0.9
Avg	64.78	59.72	53.58	58.33	53.13	62.42	65.66	64.81	69.27	52.10	66.65	**71.32**

DOC: Deep Open Classification

(Shu et al. 2017)

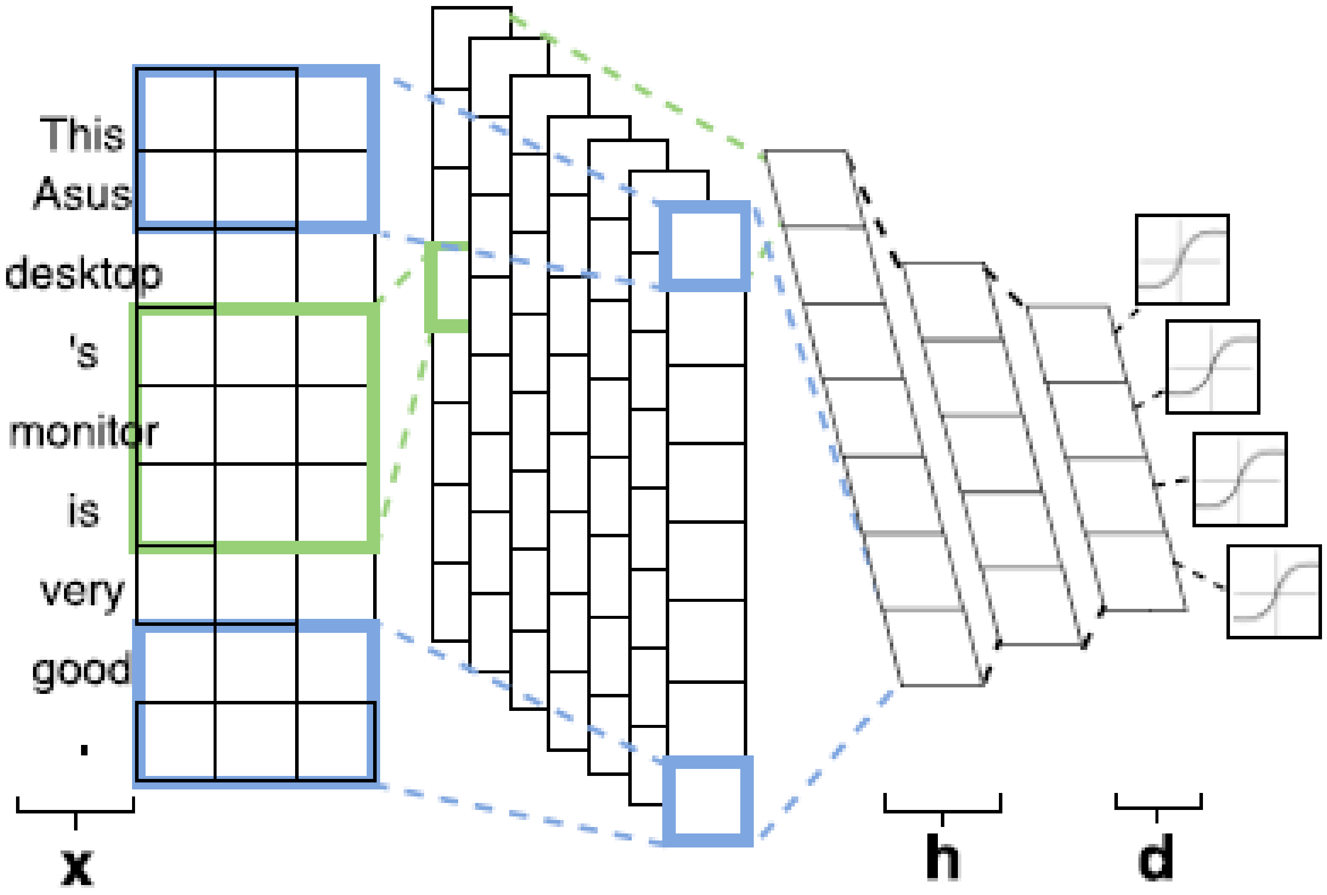


Figure 1: Overall Network of DOC

Given a classification problem with n classes, called, *in-distribution* (IND) classes. In DOC,

- Each head is a sigmoid unit as a binary classifier,
- One for each class

Shu, Xu, Liu. DOC: Deep Open Classification of Text Documents. EMNLP-2017

Finding the rejection threshold

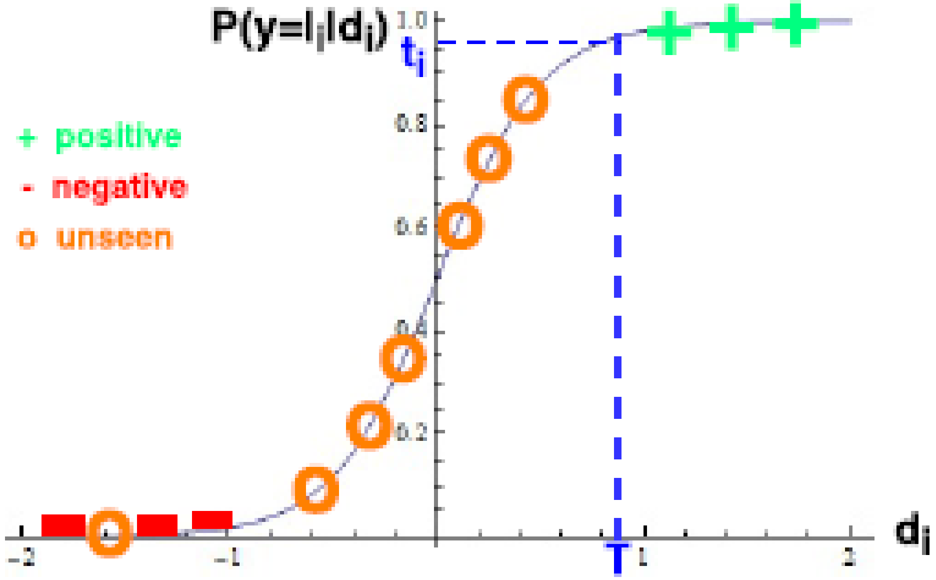


Figure 2: Open space risk of sigmoid function and desired decision boundary $d_i = T$ and probability threshold t_i.

CSI: OOD detection via contrastive learning

- CSI consists of two main ideas (step 1)
  - It does data augmentation and also class augmentation
    - For an image,
      - It does: crop, horizontal flip, and color changes to create additional images. These are treated as positive data.
      - It does rotations of 90, 180 and 270 degrees and treat them as different classes and thus negative data.
  - It uses supervised contrastive learning for feature learning.

$$\mathcal{L}_c = \frac{1}{8N} \sum_{x \in \tilde{\mathcal{B}}} \frac{-1}{|P(x)|} \sum_{p \in P(x)} \log \frac{\exp(z_x \cdot z_p / \tau)}{\sum_{x' \in \tilde{\mathcal{B}}(x)} \exp(z_x \cdot z_{x'} / \tau)}$$

Tack, Mo, Jeong, Shin. CSI: Novelty detection via contrastive learning on distributionally shifted instances. NeurIPS-2020.

CSI: OOD detection via contrastive learning (cont.)

- Step 2 of CSI
  - Given the feature extractor *h* trained with the loss in the previous slide, *h* is frozen, and
  - only fine-tune the linear classifier *f*, which is trained to predict the classes of task *t* and the augmented rotation classes.
- At inference or testing,
  - Ensemble class prediction.

$$f(h(x))_j = \frac{1}{4} \sum_{\text{deg}} f(h(x_{\text{deg}}))_{j,\text{deg}}$$

CMD: OOD detection via data generation

- CSI is very accurate in AUC but extremely slow due to its data and class augmentation and contrastive learning.
- CMD is much faster, at least 10 times faster, with on par AUC
  - ❑ CMD learns a CVAE generator to generate pseudo-OOD data.

$$\mathcal{L}(\mathbf{x};\theta,\phi) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z},c)] + KL(q_\phi(\mathbf{z}|\mathbf{x},c)||p_\theta(\mathbf{z}|c))$$

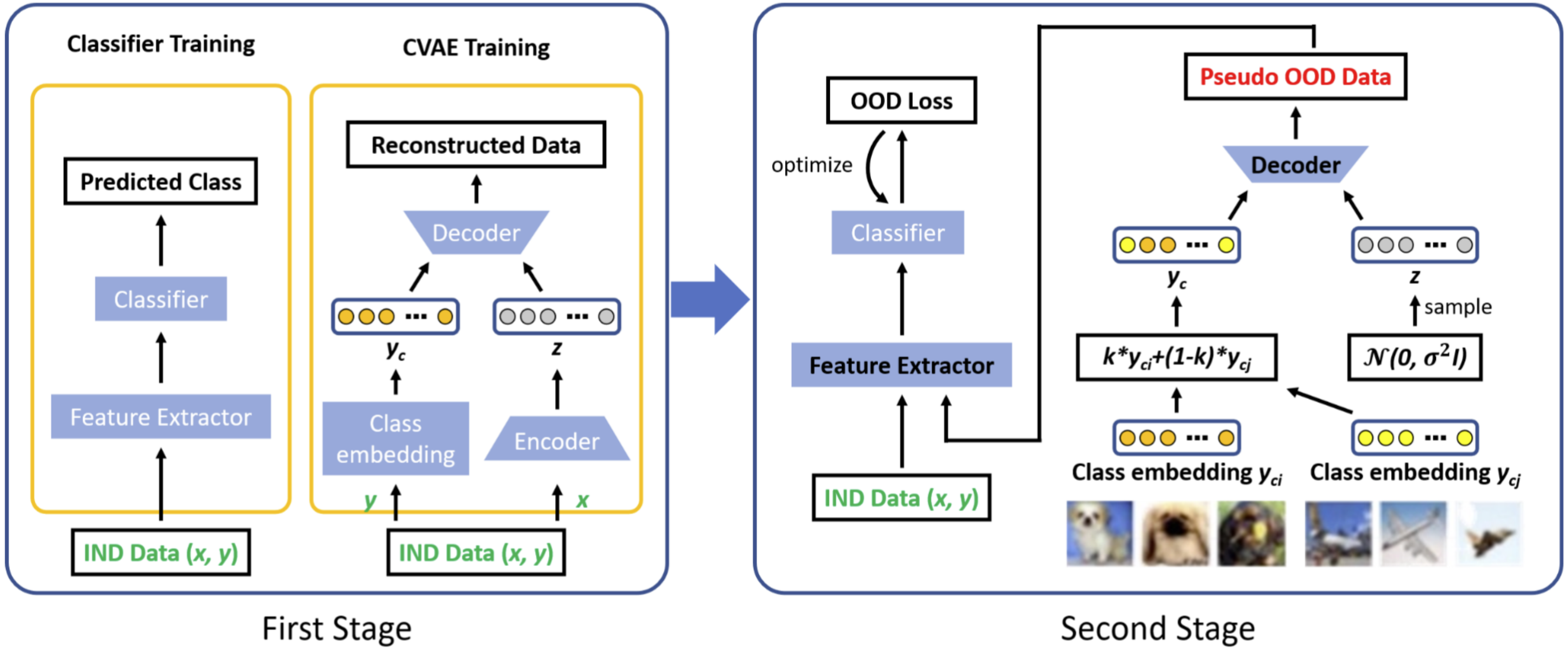
  - ❑ Key idea: combine embeddings of different IND classes to generate abnormal conditions for CVAE to generate pseudo-OOD data.

$$p_\theta(\mathbf{x}|\mathbf{z},\mathbf{k},c_i,c_j) = p_\theta(\mathbf{x}|[\mathbf{k}*\mathbf{y}_{c_i} + (1-\mathbf{k})*\mathbf{y}_{c_j},\mathbf{z}])$$

    - Such data are similar but also different from the data of any existing class.

Wang, Shao, Li, Hu and Liu. CMG: A Class-Mixed Generation Approach to Out-Of-Distribution Detection. ECML-2022.

CMG framework and training process



Note that:

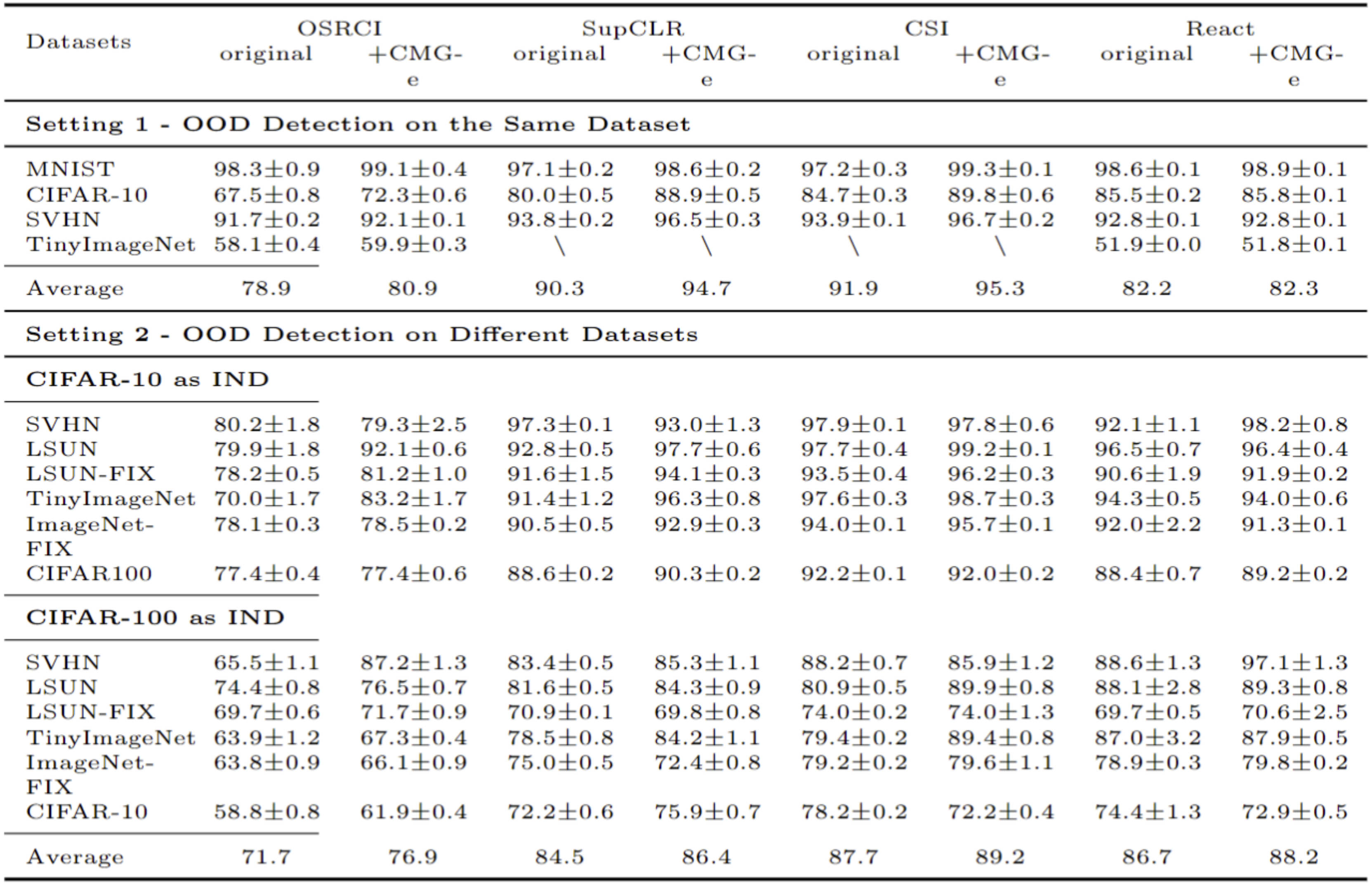
(1) Stage 2 only fine-tunes, no feature learning.

(2) The classifier is not specified. Then CMG can improve existing OOD detectors.

Fig. 1. CMG framework and its training process. The OOD loss can be cross entropy in CMG-softmax, cross-entropy+energy in CMG-energy, or other possible losses. Although we put Classifier Training and CVAE Training in First Stage, they are independent.

Performance comparison

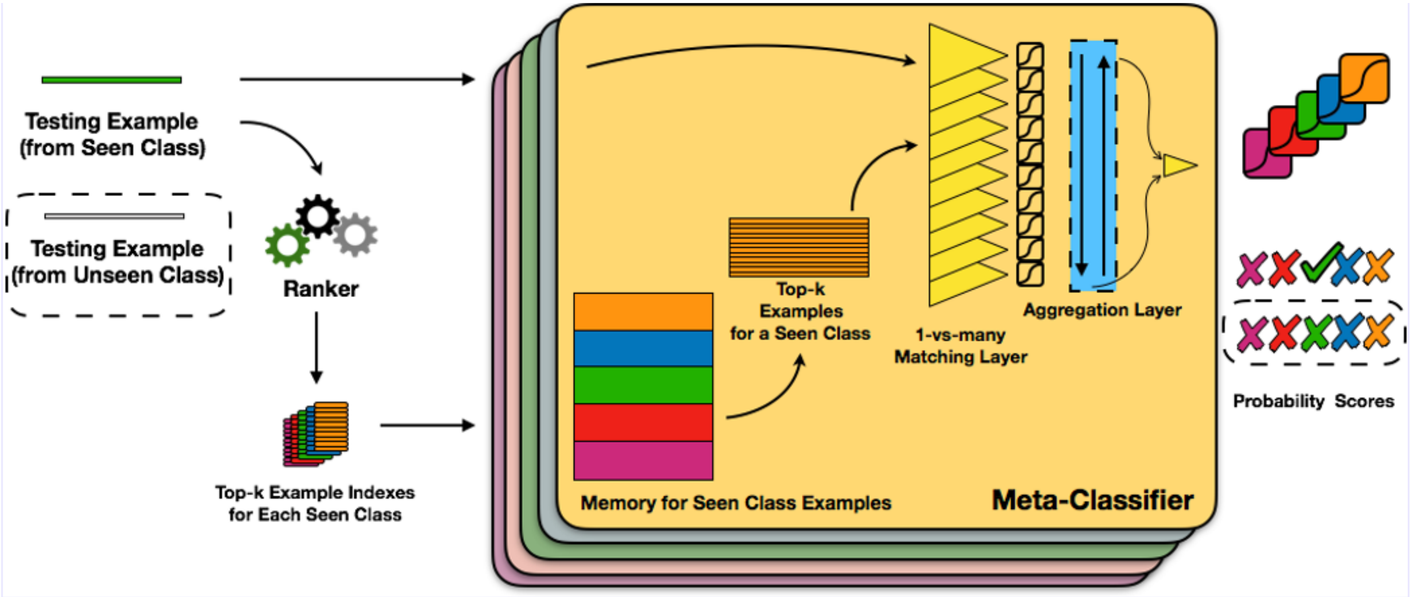
- CMG-e adds an energy-based loss to cross entropy as the OOD loss to fine-tune an OOD detector.
- CMG runs at least 10 times faster than CSI.



Datasets	OSRCI		SupCLR		CSI		React	
	original	+CMG-e	original	+CMG-e	original	+CMG-e	original	+CMG-e
Setting 1 - OOD Detection on the Same Dataset								
MNIST	98.3±0.9	99.1±0.4	97.1±0.2	98.6±0.2	97.2±0.3	99.3±0.1	98.6±0.1	98.9±0.1
CIFAR-10	67.5±0.8	72.3±0.6	80.0±0.5	88.9±0.5	84.7±0.3	89.8±0.6	85.5±0.2	85.8±0.1
SVHN	91.7±0.2	92.1±0.1	93.8±0.2	96.5±0.3	93.9±0.1	96.7±0.2	92.8±0.1	92.8±0.1
TinyImageNet	58.1±0.4	59.9±0.3	\	\	\	\	51.9±0.0	51.8±0.1
Average	78.9	80.9	90.3	94.7	91.9	95.3	82.2	82.3
Setting 2 - OOD Detection on Different Datasets								
CIFAR-10 as IND								
SVHN	80.2±1.8	79.3±2.5	97.3±0.1	93.0±1.3	97.9±0.1	97.8±0.6	92.1±1.1	98.2±0.8
LSUN	79.9±1.8	92.1±0.6	92.8±0.5	97.7±0.6	97.7±0.4	99.2±0.1	96.5±0.7	96.4±0.4
LSUN-FIX	78.2±0.5	81.2±1.0	91.6±1.5	94.1±0.3	93.5±0.4	96.2±0.3	90.6±1.9	91.9±0.2
TinyImageNet	70.0±1.7	83.2±1.7	91.4±1.2	96.3±0.8	97.6±0.3	98.7±0.3	94.3±0.5	94.0±0.6
ImageNet-FIX	78.1±0.3	78.5±0.2	90.5±0.5	92.9±0.3	94.0±0.1	95.7±0.1	92.0±2.2	91.3±0.1
CIFAR100	77.4±0.4	77.4±0.6	88.6±0.2	90.3±0.2	92.2±0.1	92.0±0.2	88.4±0.7	89.2±0.2
CIFAR-100 as IND								
SVHN	65.5±1.1	87.2±1.3	83.4±0.5	85.3±1.1	88.2±0.7	85.9±1.2	88.6±1.3	97.1±1.3
LSUN	74.4±0.8	76.5±0.7	81.6±0.5	84.3±0.9	80.9±0.5	89.9±0.8	88.1±2.8	89.3±0.8
LSUN-FIX	69.7±0.6	71.7±0.9	70.9±0.1	69.8±0.8	74.0±0.2	74.0±1.3	69.7±0.5	70.6±2.5
TinyImageNet	63.9±1.2	67.3±0.4	78.5±0.8	84.2±1.1	79.4±0.2	89.4±0.8	87.0±3.2	87.9±0.5
ImageNet-FIX	63.8±0.9	66.1±0.9	75.0±0.5	72.4±0.8	79.2±0.2	79.6±1.1	78.9±0.3	79.8±0.2
CIFAR-10	58.8±0.8	61.9±0.4	72.2±0.6	75.9±0.7	78.2±0.2	72.2±0.4	74.4±1.3	72.9±0.5
Average	71.7	76.9	84.5	86.4	87.7	89.2	86.7	88.2

Open-world continual learning via meta-learning

- **L2AC perform:**
  - OOD detection
  - Continual learning
- **L2AC–meta-learning**



  - It maintains a dynamic set S of seen classes that allows new classes to be added or deleted without re-training.
    - Each class is represented by a small set of training examples.
  - In testing, the meta-classifier uses only the examples of the seen classes on-the-fly for classification and rejection (novel)

Xu, Liu, Shu and Yu. Open-world Learning and Application to Product Classification. WWW-2019

L2AC framework

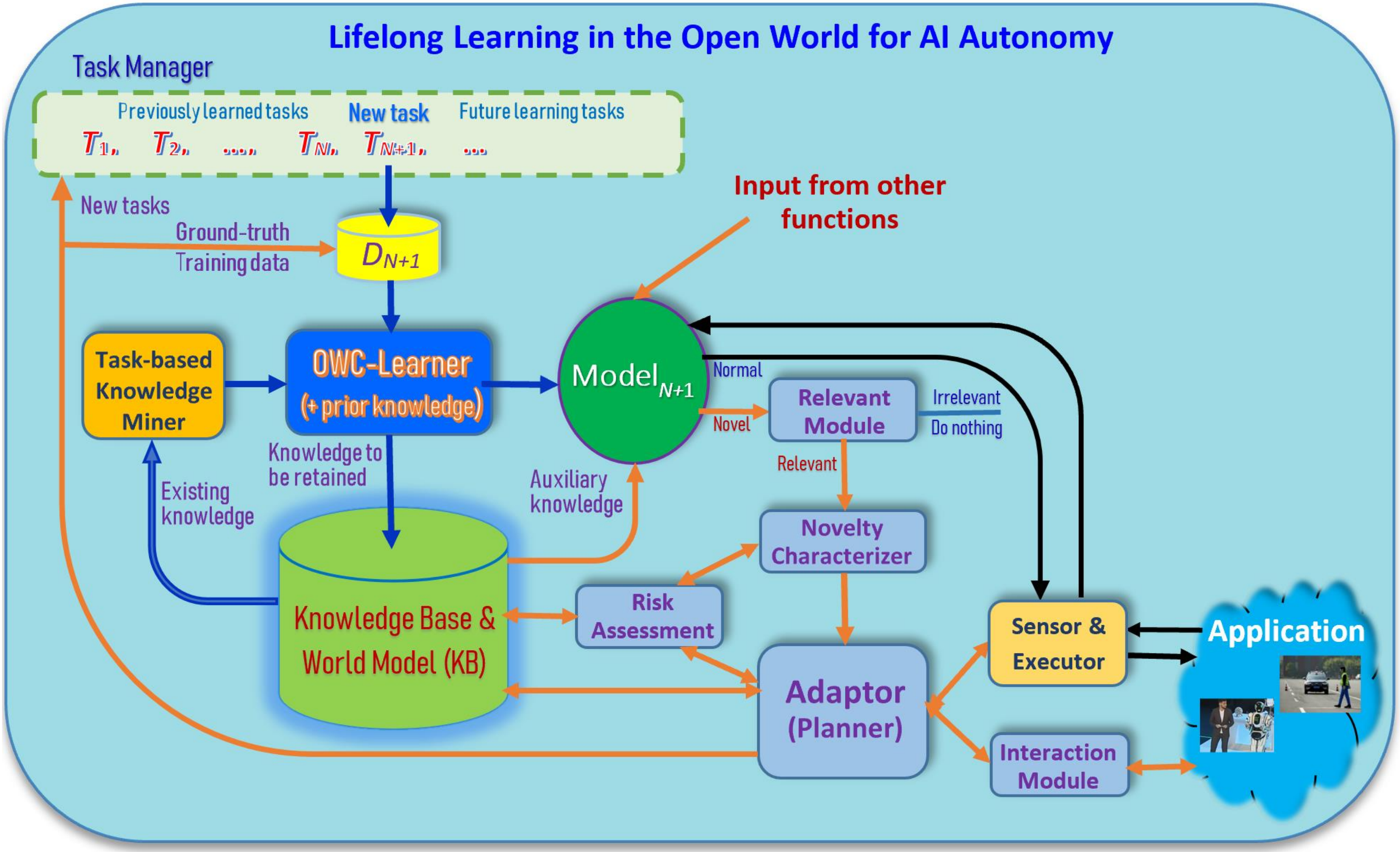
- L2AC has two major components:
  - **a ranker**: retrieve some examples from a seen class that are similar/near to the test example.
    - Given a test example x that may come from either a seen or an unseen class, the ranker finds a list of top-k nearest examples to x from each seen class c.
  - **a meta-classifier**. perform classification after it reads the retrieved examples from the seen classes.
    - The meta-classifier produces the probability that the test x belongs to class c
    - A threshold is used to determine reject (OOD).
- Continual learning: give some examples of each new class.

Sub-topics

- Continual learning in the open-world
  - Closed world and open world
  - Open world continual learning framework
  - Novelty or out-of-distribution detection
  - Two open world continual learning systems: CML and CILK

Recall open world continual learning

- **SOLA: S**elf-initiated **O**pen-world Continual **L**earning and **A**daptation.
- AI autonomy



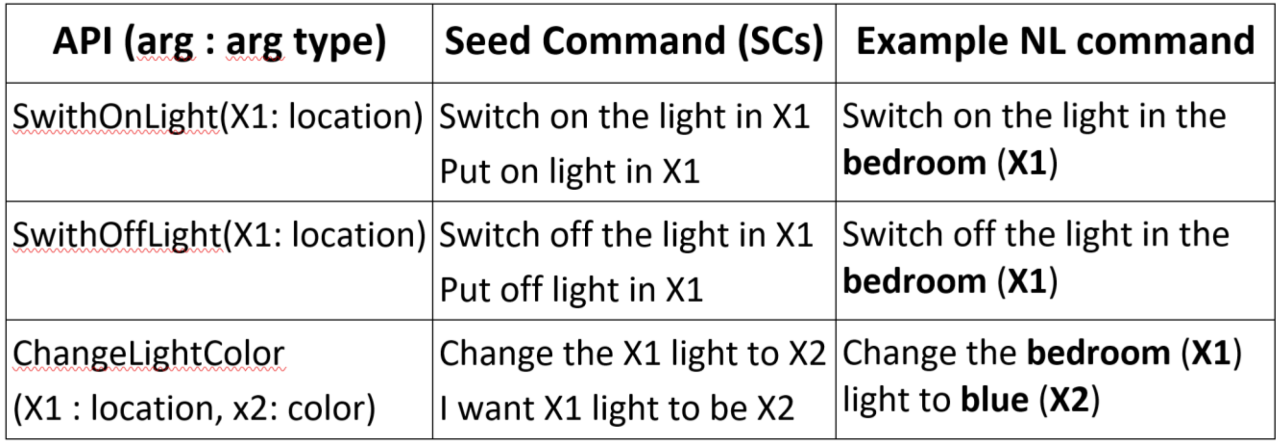
Example SOLA: natural language interface (NLI)

- Performance task: user asks the system (CML, like **Siri** and **Alexa**) to perform a task in NL, the system does it via API actions
  - Approach: *natural language to natural language matching* (**NL2NL**)
- **CML** builds NLIs for API-driven applications semi-automatically.
- **To build a new NLI** (or add a new skill to an existing NLI),
  - Application developer writes a set S_i of seed commands (SCs) in NL to represent each API action *i*.
    - ❑ SCs in S_i are just like paraphrased NL commands from users to invoke *i*, but the objects to be acted upon in each SC are replaced with variables, the arguments of *i*.
  - When the system does not understand a command (**novelty**), it **adapts** and **learns** new (paraphrased) SCs from users interactively and continually.

Mazumder, Liu, Wang, and Esmaeilpour. An Application-Independent Approach to Building Task-Oriented Chatbots with Interactive Continual Learning. *NeurIPS-2020 Workshop on Human in the Loop Dialogue Systems (HLDS-2020)*

An example – Smart home

- SmartHome: API action: *SwitchOnLight*(X1)
  - Switching on a light at a given place X1



API (arg : arg type)	Seed Command (SCs)	Example NL command
SwithOnLight(X1: location)	Switch on the light in X1 Put on light in X1	Switch on the light in the **bedroom** (**X1**)
SwithOffLight(X1: location)	Switch off the light in X1 Put off light in X1	Switch off the light in the **bedroom** (**X1**)
ChangeLightColor (X1 : location, x2: color)	Change the X1 light to X2 I want X1 light to be X2	Change the **bedroom** (**X1**) light to **blue** (**X2**)

- Let an SC be “put on light in X1" for this API,
  - where X1 is a variable representing the argument of the API.
- User command: “power on the light in the bedroom”
  - It can be matched or grounded to this SC, where the grounded API arguments are {X1 = ‘bedroom’}.

CML has three components

- SC (seed command) specification
  - to enable application developer to specify a set of SCs for each of their APIs
- Command grounding module
  - ground a user command *C* to an action SC by matching C with the correct SC (whose associated action API is then executed)
- Interactive continual learner
  - It interacts with end-users to learn new SCs and paraphrases of API argument values.

Command grounding module (CGM)

- Rephraser and Tagger (*R*):
  - Given the user command C, *R* repharses C and tags each word or phrase in the rephrased C with either ‘O’ (i.e., not an argument type) or one of the possible argument types of the action SCs.
- SC Matcher (*M*):
  - Given the rephrased and tagged command C and the set T of (action or utility) SCs, Matcher *M* computes a match score *f*(t, C) for each *t* in T and returns the top ranked SC.
  - This work uses an information retrieval (IR) based unsupervised matching model for *M*

Novelty detection, characterization, adaptation

- Novelty detection: when CML cannot ground a user command, e.g., it cannot understand "*turn off the light in the kitchen*"
- Novelty characterization: which part of the command it understands and which part it has difficulty based on similarity. E.g., it cannot ground "*turn off*"
- Adaptation (or accommodation):
  - Response: ask the user by providing some options (to collect ground-truth data)

    Bot: Sorry, I didn't get you. Do you mean to:

    option-1. switch off the light in the kitchen,
    option-2. switch on the light in the kitchen, or
    option-3. change the color of the light in the kitchen?
  - Continual learning: learn a **new SC.** No issue with related commands in future.

Risk consideration

- CML manages **risk** in two ways
  - Do not ask user too many questions in order not to annoy the user.
    - Learning can be done to assess each user's tolerance.
  - When characterization is not confident, take the default action, i.e.,
    - Ask the user to **say his/her command in another way**
      - rather than providing a list of random options for user to choose from
        - which can be annoying or make the user lose confidence in the system!

Continuous knowledge learning in dialogues

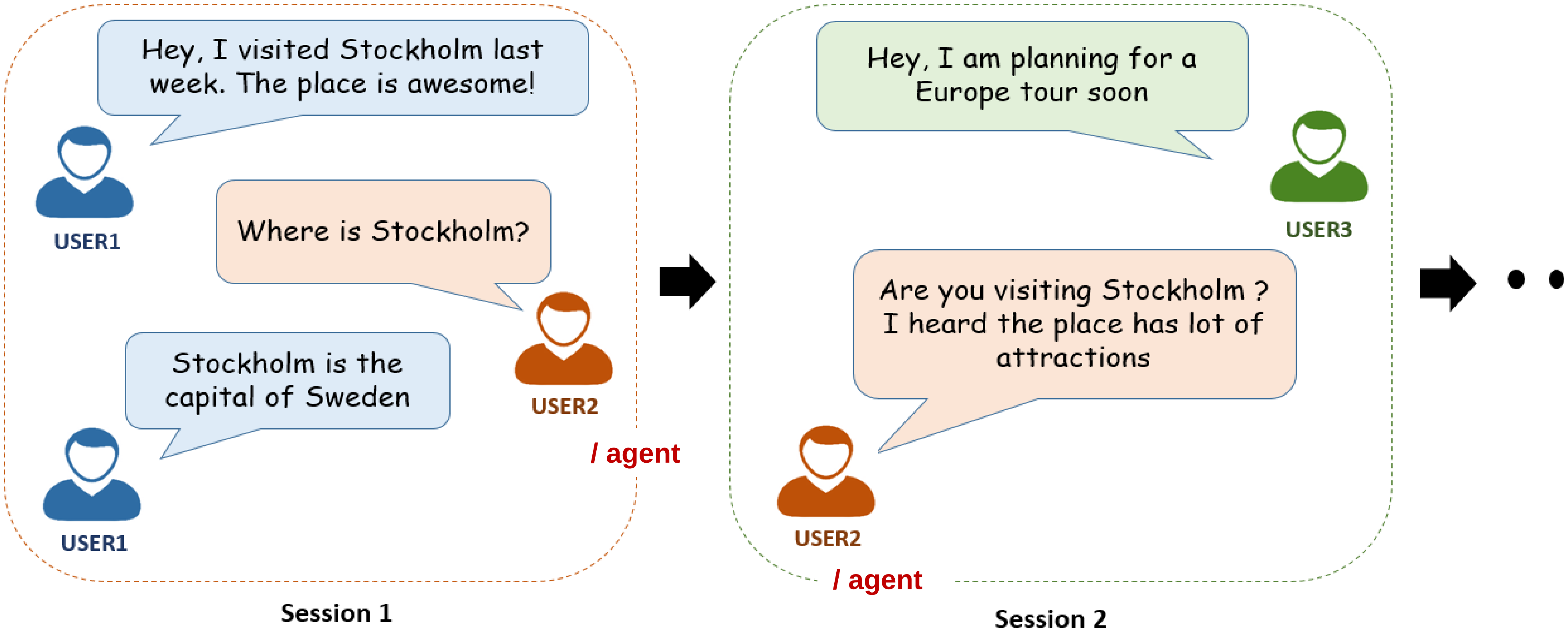
- Dialogue systems are increasingly using knowledge bases (KBs) storing factual knowledge to help generate responses.
  - KBs are inherently incomplete and remain fixed,
    - which limit dialogue systems’ conversation capability
- CILK: ***C****ontinuous and* ***I****nteractive* ***L****earning of* ***K****nowledge*
  - to continuously and interactively learn and infer new knowledge during conversations

Mazumder, Ma, and Liu. Towards a Continuous Knowledge Learning Engine for Chatbots. arXiv:1802.06024 [cs.CL], 16 Feb. 2018

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019

Knowledge learning in conversation

Humans Learn and Leverage Knowledge in Lifelong Manner!



Hey, I visited Stockholm last week. The place is awesome!

Where is Stockholm?

Stockholm is the capital of Sweden

Hey, I am planning for a Europe tour soon

Are you visiting Stockholm ? I heard the place has lot of attractions

Knowledge learning happens in a multi-user environment

Opportunities to learn in conversations

1. Extracting knowledge directly from user utterances. E.g.,
   - User: Obama was born in Hawaii.
   - Agent extracts: (Obama, BornIn, Hawaii) – expressed in triples **(h, r, t)**
2. Asking user questions & expecting correct answers, e.g.,
   - Agent: Where was Obama born?
   - User: Hawaii => (Obama, BornIn, Hawaii)
3. When the agent cannot answer user questions, it asks the user for some supporting facts and then infers the answers.
   - **We focus on this setting** (which covers 1 and 2)

Liu and Mazumder. Lifelong and Continual Learning Dialogue Systems: Learning during Conversation. AAAI-2021

Two types of queries or questions

- Wh-question
  - E.g., Where was Obama born?
  - (Obama, bornIn, s?)

- Fact verification question
  - Was Obama born in Hawaii?
  - (Obama, bornIn? Hawaii)

Mazumder, Ma, and Liu. Towards a Continuous Knowledge Learning Engine for Chatbots. arXiv:1802.06024 [cs.CL], 16 Feb. 2018

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019

Assumptions

- Focus on developing the core interactive knowledge learning
  - **Do not build all peripheral components** (like fact or relation extraction, entity linking, etc.) which are assumed to be available for use.
- Assume that the user has good intentions
  - **User answers questions with 100% conformity** about the veracity of his/her facts
    - Cross-verification can be used to deal with wrong knowledge
- **Do not assume user can answer all questions**
  - as opposed to the teacher-student setup - the teacher is assumed to know everything.

Mazumder, Ma, and Liu. Towards a Continuous Knowledge Learning Engine for Chatbots. arXiv:1802.06024 [cs.CL], 16 Feb. 2018

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019

Components for knowledge learning

Knowledge Base

$\mathcal{K}$

Stores acquired Facts (Triples)

KB: Collection of Triples

$$\mathcal{T} = \{\ (h, r, t) \mid h, t \in E, r \in R\ \}$$

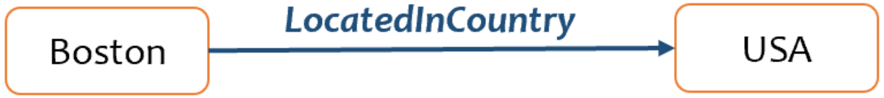
Triple *Entity Set* *Relation Set*

<u>**Triple Store**</u>

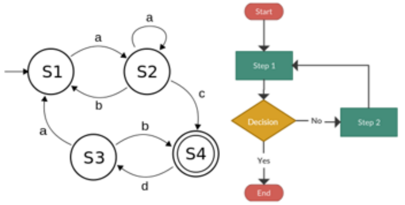
(Boston, *LocatedInCountry*, USA)

head *relation* *tail*

or



<u>**Knowledge Graph**</u>

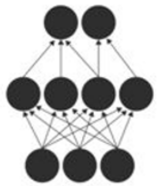


Interaction Module

$\mathcal{I}$

Interacts with user to acquire Facts

- decides whether to ask or not, and formulates questions to ask the user for supporting facts



Inference Module

$\mathcal{M}$

Infers new Knowledge to answer user's query

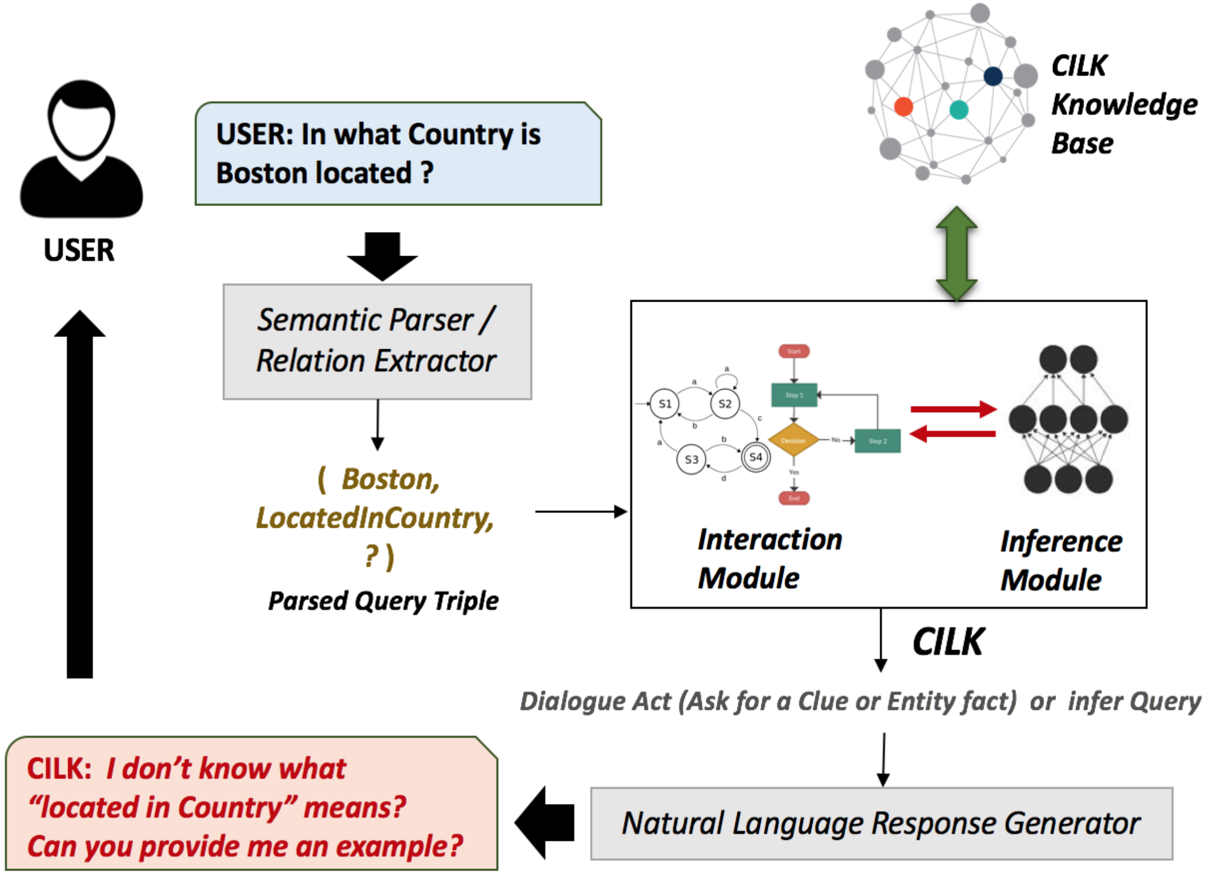
- Performs inference over the acquired Facts and existing KB

Mazumder, Ma, and Liu. Towards a Continuous Knowledge Learning Engine for Chatbots. arXiv:1802.06024 [cs.CL], 16 Feb. 2018

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019

Interactive knowledge learning in dialogue: example

(Mazumder et al. 2019)



USER

USER: In what Country is Boston located ?

Semantic Parser / Relation Extractor

(Boston, LocatedInCountry, ?)

Parsed Query Triple

CILK Knowledge Base

Interaction Module

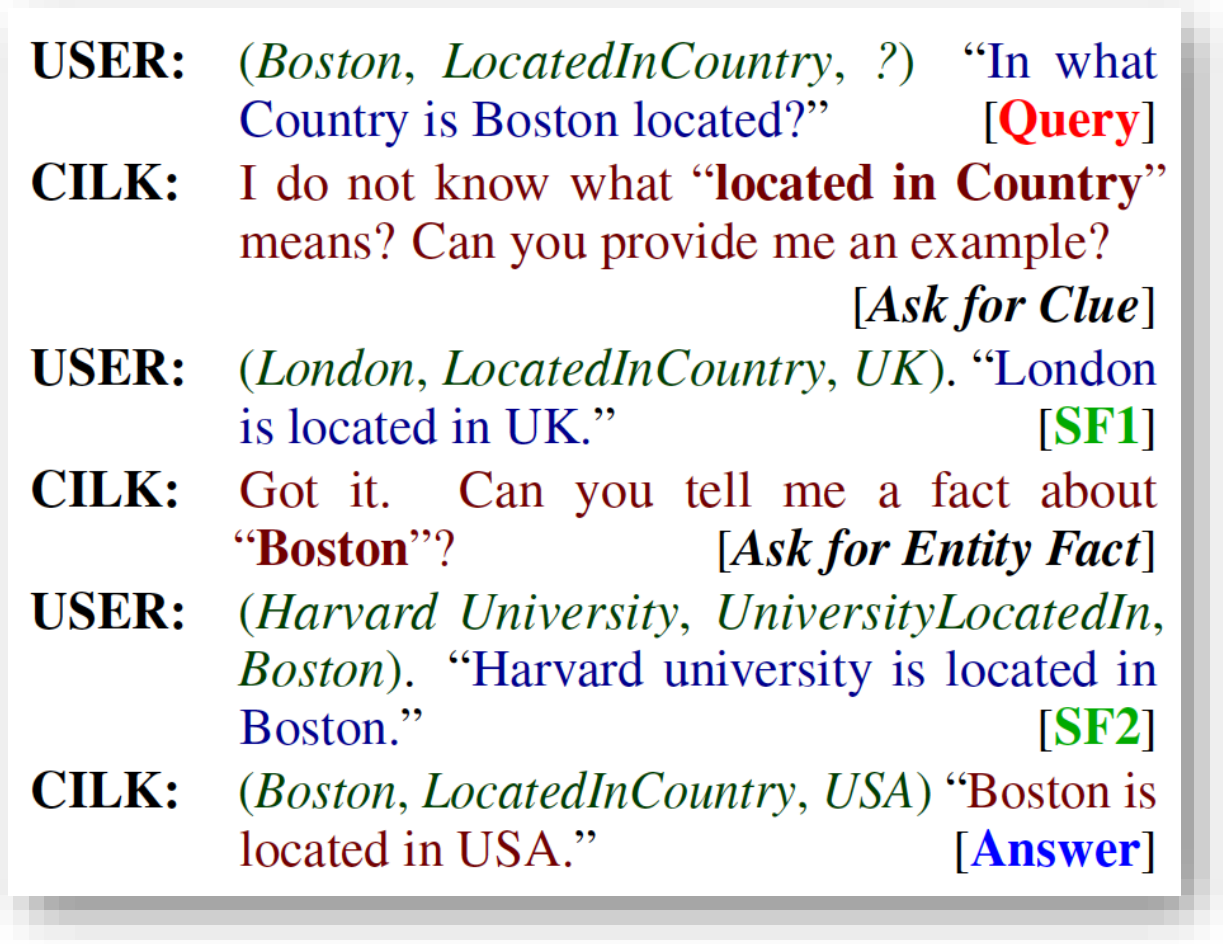
Inference Module

CILK

Dialogue Act (Ask for a Clue or Entity fact) or infer Query

Natural Language Response Generator

CILK: *I don't know what "located in Country" means? Can you provide me an example?*



USER: (*Boston*, *LocatedInCountry*, *?*) "In what Country is Boston located?" [**Query**]

CILK: I do not know what "**located in Country**" means? Can you provide me an example? [*Ask for Clue*]

USER: (*London*, *LocatedInCountry*, *UK*). "London is located in UK." [**SF1**]

CILK: Got it. Can you tell me a fact about "**Boston**"? [*Ask for Entity Fact*]

USER: (*Harvard University*, *UniversityLocatedIn*, *Boston*). "Harvard university is located in Boston." [**SF2**]

CILK: (*Boston*, *LocatedInCountry*, *USA*) "Boston is located in USA." [**Answer**]

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019

Topics

- Lifelong or continual learning
- Early research on lifelong learning
- Continual learning based on deep neural networks
- Continual learning in the open-world
- Summary

Summary

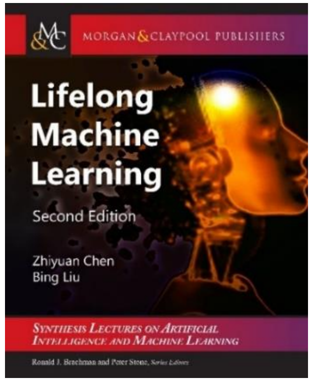
- Classical ML: isolated, closed-world, single-task learning
- This short course studied the following
  - Definition of lifelong/continual learning
  - Early research of lifelong learning
  - Lifelong/continual learning using deep neural networks
    - Batch continual learning
    - Online continual learning
  - Continual learning in the open world
    - Try to achieve AI autonomy

Summary: challenges

- Catastrophic forgetting
  - Still a major challenge, especially for class incremental learning (CIL)
- Knowledge transfer – limited work has been done.
  - Correctness and applicability of knowledge
  - What to transfer and how to transfer forward and backward
- Open world continual learning is an even bigger challenge
  - Novelty, characterization, adaptation, and incremental learning
- Goal: **AI autonomy -** the next generation AI.
  - Autonomous learning agents will be built in restricted environments
    - ❑ Interacting with people, other agents, and the environment

Thank You!

Q&A



MORGAN&CLAYPOOL PUBLISHERS

Lifelong Machine Learning

Second Edition

Zhiyuan Chen
Bing Liu

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Ronald J. Brachman and Peter Stone, Series Editors