DEIM-2023 Tutorial

Lifelong and Continual Learning

Bing Liu,[1] Tatsuya Konishi,[2] Gyuhak Kim,[1] and Zixuan Ke[1]

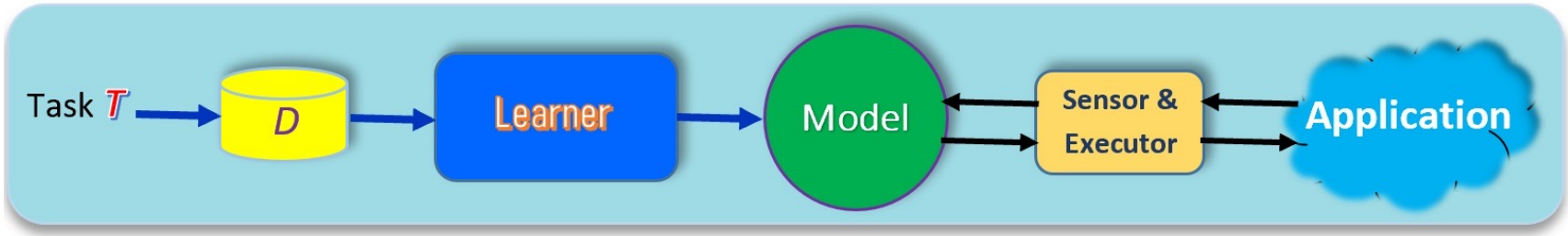
(The order of the presenters is based on the order of their presentations)

[1] University of Illinois Chicago (UIC), USA

[2] KDDI Research, Japan (currently visiting UIC)

Introduction

- Classical machine learning: **Isolated single-task learning**



- **A Key Weakness**
  - **No continual learning,** no knowledge accumulation or transfer
- **Humans** learn continually and accumulate knowledge over time.
  - We never stop learning
- **ChatGPT** can only answer questions based on its training data going up until 2021. It needs to be continually updated or adapted.

Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2016, 2018
Liu. Learning on the Job: Online Lifelong and Continual Learning. AAAI-2020

Early definition of lifelong learning

(Thrun 1996, Silver et al 2013; Ruvolo and Eaton, 2013; Chen and Liu, 2014, Chen and Liu, 2016)

- The learner has performed learning on a sequence of tasks from 1 to *N*.
- When faced with the new or (*N*+1)th task, it uses the relevant knowledge learned from the earlier tasks stored in the *knowledge base* (KB) to help learn the (*N*+1)th task.
  - After learning (*N*+1)th task, KB is updated with the learned results from the (*N*+1)th task.
- **Goal**: knowledge transfer

Thrun. Is learning the n-th thing any easier than learning the first? NIPS-1996.

New definition of lifelong/**continual learning**

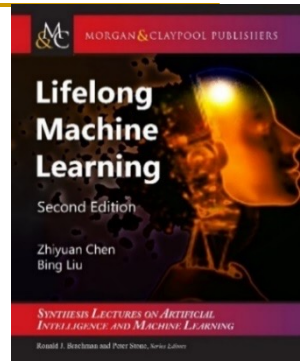
(Chen and Liu, 2014, 2018)

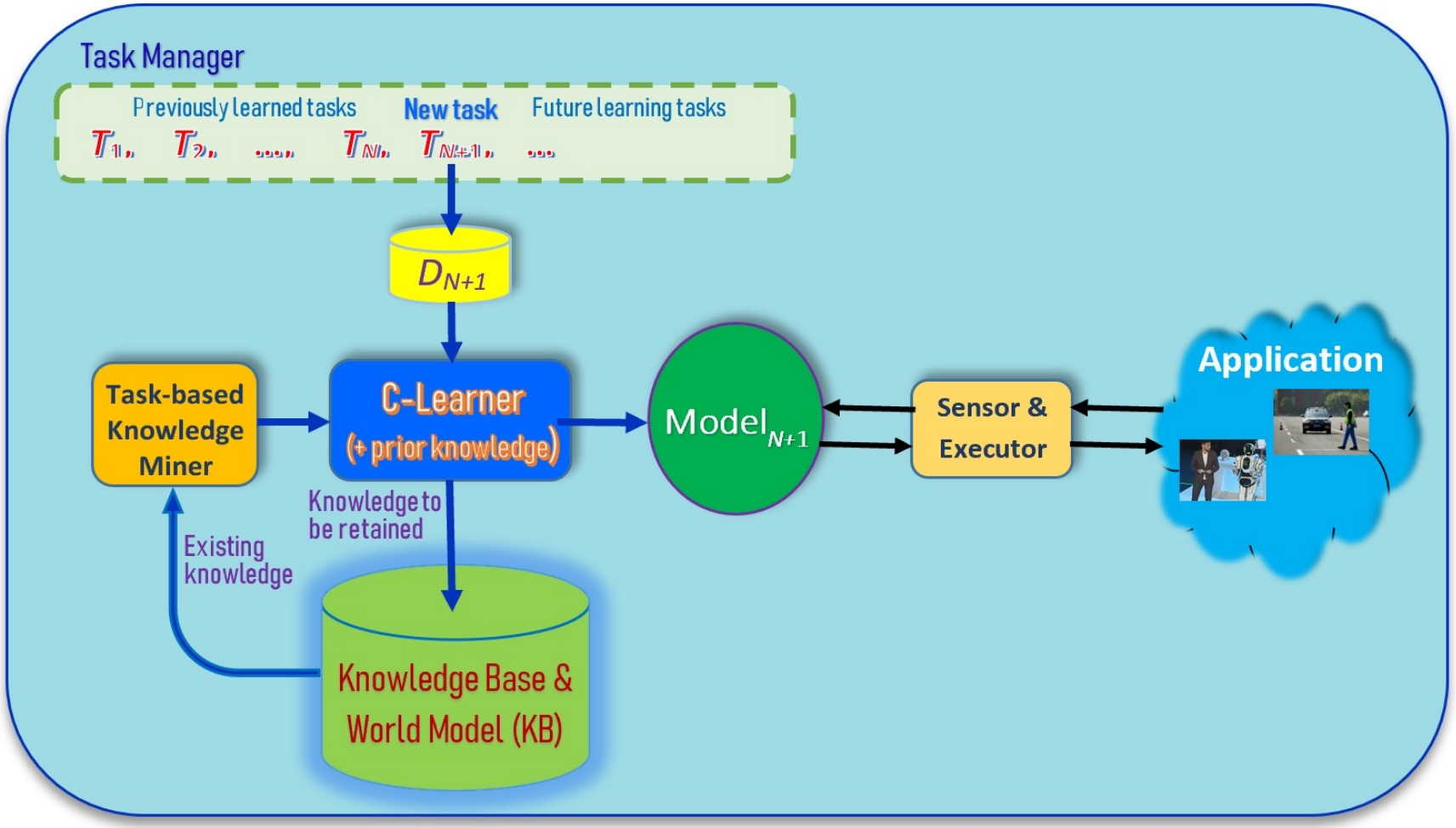
- Learn a sequence of tasks, T_1, T_2, …, T_N, … incrementally. Each task *t* has a training dataset $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.
- Goal: Learning all tasks in a single neural network
  1. **without catastrophic forgetting:** Learning of new task T_{N+1} should not result in degradation of accuracy for the previous *N* tasks.
  2. with knowledge transfer: leveraging the knowledge learned from the previous tasks to learn the new task T_{N+1} better.
- Assumption: Once a task is learned, its data is no longer accessible, at least a majority of it, and both the task T_{N+1}.

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

Lifelong/continual learning

(Thrun 1996, Silver et al 2013; Ruvolo and Eaton, 2013; Chen and Liu, 2014, 2018)





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

Different continual learning settings

- Task incremental learning (TIL)
  - train a “separate” model for each task
  - task-id is provided in testing
- Class incremental learning (CIL)
  - produce a single model from all tasks
  - classify all classes in testing
- Domain incremental learning (DIL)
  - All tasks have the same set of classes
  - Task-id is not provided in testing

Topics

- Early Research on Lifelong Learning (Bing Liu)
- Task Incremental Learning (Tatsuya Konishi)
- Class Incremental Learning (Gyuhak Kim)
- Domain Incremental Learning & Continual Learning in NLP (Zixuan Ke)
- Open-World Continual Learning (Bing Liu)
- Summary (Bing Liu)

Early Research on Lifelong Learning

Bing Liu

University of Illinois Chicago

https://www.cs.uic.edu/~liub/

Early work of lifelong learning (LL)
(Thrun, 1996)

- Concept learning tasks: The functions are learned over the lifetime of the learner, $f_1, f_2, f_3, \ldots \in F$.
- Each task: learn the function $f: I \rightarrow \{0, 1\}$. $f(x)=1$ means x belongs to a particular concept (or class).
  - For example, $f_{dog}(x)=1$ means x is a dog.
- For *n*th task, we have its training data X and also
  - the training data X_k of k =1 , 2, …, n-1 tasks.
- Most early LL is about *task incremental learning* (TIL)
  - In learning each new task, it may still use all previous task data.

Thrun. Is learning the n-th thing any easier than learning the first? NIPS-1996.

Approaches in (Thrun, 1996)

- The paper proposed a few approaches based on two learning algorithms,
  - Memory-based, e.g., kNN
  - Neural networks,
- **Goal:** when we learn $f_{dog}(x)$, we can leverage functions or knowledge learned from previous tasks, such as $f_{cat}(x)$, $f_{bird}(x)$, $f_{tree}(x)$, etc.
  - Data for $f_{cat}(X)$, $f_{bird}(X)$, $f_{tree}(X)$… are called support sets.

Thrun. Is learning the n-th thing any easier than learning the first? NIPS-1996.

KNN based "lifelong learning"

- Use the support sets of $\{f_1, f_2, \ldots, f_k, \ldots, f_{n-1}\}$ to learn a new representation, or function

  g: $I \rightarrow I'$

  - which maps input vectors to a new space. The new space is the input space for the final *k*NN.
  - Adjust *g* to minimize the energy function.

Adjusting *g* to minimize *E* forces the distance between pairs of examples of the same class to be small, and the distance between examples of different classes to be large

$$E := \sum_{k=1}^{n-1} \sum_{\langle x, y=1\rangle \in X_k} \left(\sum_{\langle x', y'=1\rangle \in X_k} ||g(x)-g(x')|| - \sum_{\langle x', y'=0\rangle \in X_k} ||g(x)-g(x')|| \right)$$

  - g is a neural network, trained with Back-Prop. kNN is then applied for the *n*th (new) task. *g* basically learns a feature extractor.

ELLA: Efficient Lifelong Learning Algorithm

(Ruvolo & Eaton, 2013)

- ELLA is based on GO-MTL (Kumar et al., 2012)
  - GO-MTL is a batch multitask learning method
- ELLA is an online multitask learning method
  - ELLA is more efficient and can handle a large number of tasks
  - Becomes a lifelong learning method
    - The model for a new task can be added efficiently.
    - The model for each past task can be updated rapidly.

Ruvolo and Eaton. ELLA: An efficient lifelong learning algorithm. ICML-2013.

Notations

- N tasks in total
- k (< N) latent basis model components
- Each basis task is represented by a $\boldsymbol{l}$ (a vector of size d)
- For all latent tasks, $\boldsymbol{L} = (\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_k)$
- $\boldsymbol{L}$ is learned from N individual tasks.
  - E.g., weights/parameters of logistic regression or linear regression

The Approach

- $\mathbf{s}^t$ is a linear weight vector and is assumed to be sparse.

$$\boldsymbol{\theta}^t = \mathbf{L}\mathbf{s}^t$$

- Stacking $\mathbf{s}^t$ for all tasks, we get $\mathbf{S}$. $\mathbf{S}$ captures the task grouping structure.

$$\underset{d\times N}{\boldsymbol{\theta}} = \underset{d\times k}{\mathbf{L}} \times \underset{k\times N}{\mathbf{S}}$$

Initial Objective Function of ELLA

- Objective Function

$$e_T(\mathbf{L}) = \frac{1}{T}\sum_{t=1}^{T}\min_{\mathbf{s}^{(t)}}\left\{\frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}\left(f\left(\mathbf{x}_i^{(t)};\mathbf{L}\mathbf{s}^{(t)}\right),y_i^{(t)}\right) + \mu\|\mathbf{s}^{(t)}\|_1\right\} + \lambda\|\mathbf{L}\|_{\mathsf{F}}^2 \ , \qquad (1)$$

Approximate Equation (1)

- Eliminate the dependence on all of the past training data through inner summation
  - By using the second-order Taylor expansion around $\theta = \theta^{(t)}$ where
  - $\theta^{(t)}$ is an optimal predictor learned on only the training data for task *t*.

Taylor Expansion

- One variable function

$$g(x) \approx g(a) + g'(a)(x-a) + \frac{1}{2}g''(a)(x-a)^2$$

- Multivariate function

$$g(\mathbf{x}) \approx g(\mathbf{a}) + \nabla g(\mathbf{a})(\mathbf{x}-\mathbf{a}) + \frac{1}{2}\|(\mathbf{x}-\mathbf{a})\|^2_{\boldsymbol{H}(\mathbf{a})}$$

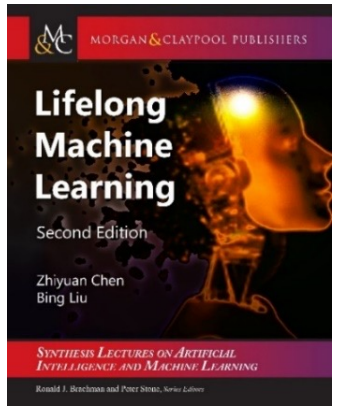
Removing inner summation

Plugging the second-order Taylor expansion into Eq. (1) yields
(see (Chen and Liu, 2018) for the derivation. The original paper didn't give the derivation)

$$\frac{1}{T}\sum_{t=1}^{T}\min_{\mathbf{s}^t}\left\{\|\hat{\boldsymbol{\theta}}^t-\mathbf{L}\mathbf{s}^t\|^2_{\boldsymbol{H}^t}+\mu\|\mathbf{s}^t\|_1\right\}+\lambda\|\mathbf{L}\|_F^2$$

$$\boldsymbol{H}^t=\frac{1}{2}\nabla^2_{\boldsymbol{\theta}^t,\boldsymbol{\theta}^t}\frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\boldsymbol{x}_i^t;\boldsymbol{\theta}^t),y_i^t\right)\Big|_{\boldsymbol{\theta}^t=\hat{\boldsymbol{\theta}}^t}$$

$$\hat{\boldsymbol{\theta}}^t=\operatorname*{argmin}_{\boldsymbol{\theta}^t}\frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\boldsymbol{x}_i^t;\boldsymbol{\theta}^t),y_i^t\right)$$



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

Simplify optimization with an assumption

- After $s^{(t)}$ is computed given the training data for task t, it will not be updated when training on other tasks. Only $\boldsymbol{L}$ changes.

ELLA Accuracy Result

- ELLA vs. GO-MTL

Dataset	Problem Type	Batch MTL Accuracy	ELLA Relative Accuracy
Land Mine	Classification	0.7802 ± 0.013 (AUC)	$99.73 \pm 0.7\%$
Facial Expr.	Classification	0.6577 ± 0.021 (AUC)	$99.37 \pm 3.1\%$
Syn. Data	Regression	-1.084 ± 0.006 (-rMSE)	$97.74 \pm 2.7\%$
London Sch.	Regression	-10.10 ± 0.066 (-rMSE)	$98.90 \pm 1.5\%$

ELLA was based on the multitask learning method in GO-MTL

LTM: Lifelong Topic Modeling

- Topic modeling (Blei et al 2003) finds topics from a collection of documents.
  - A document is a distribution over topics
  - A topic is a distribution over words, e.g., {*price*, *cost*, *cheap*, *expensive*, …}
- **Problem**: Learn a sequence of aspect extraction tasks in sentiment analysis with review data D_1, D_2, …,D_N.
  - Learned topics from each D_i is S_i. $S = \cup_i S_i$ is the *topic base so far.*
- Goal: Given a new task with its review data D^t, learn from D^t with the help of S.

Chen and Liu. Topic Modeling using Topics from Many Domains, Lifelong Learning and Big Data. ICML-2014.

Sentiment Analysis (SA) Context

- "*The size is great, but pictures are poor*."
  - Aspects/topics (product features): size, picture
- Why lifelong learning can help SA? A fair amount of aspect overlapping across reviews of different products or domains
  - Every product review domain has the aspect *price*,
  - Most electronic products share the aspect *battery*
  - Many also share the aspect of *screen*.
- This sharing of concepts / knowledge across domains is true in general, not just for SA.

What knowledge?

- Should be in the same aspect/topic

  => Must-Links

  e.g., {picture, photo}

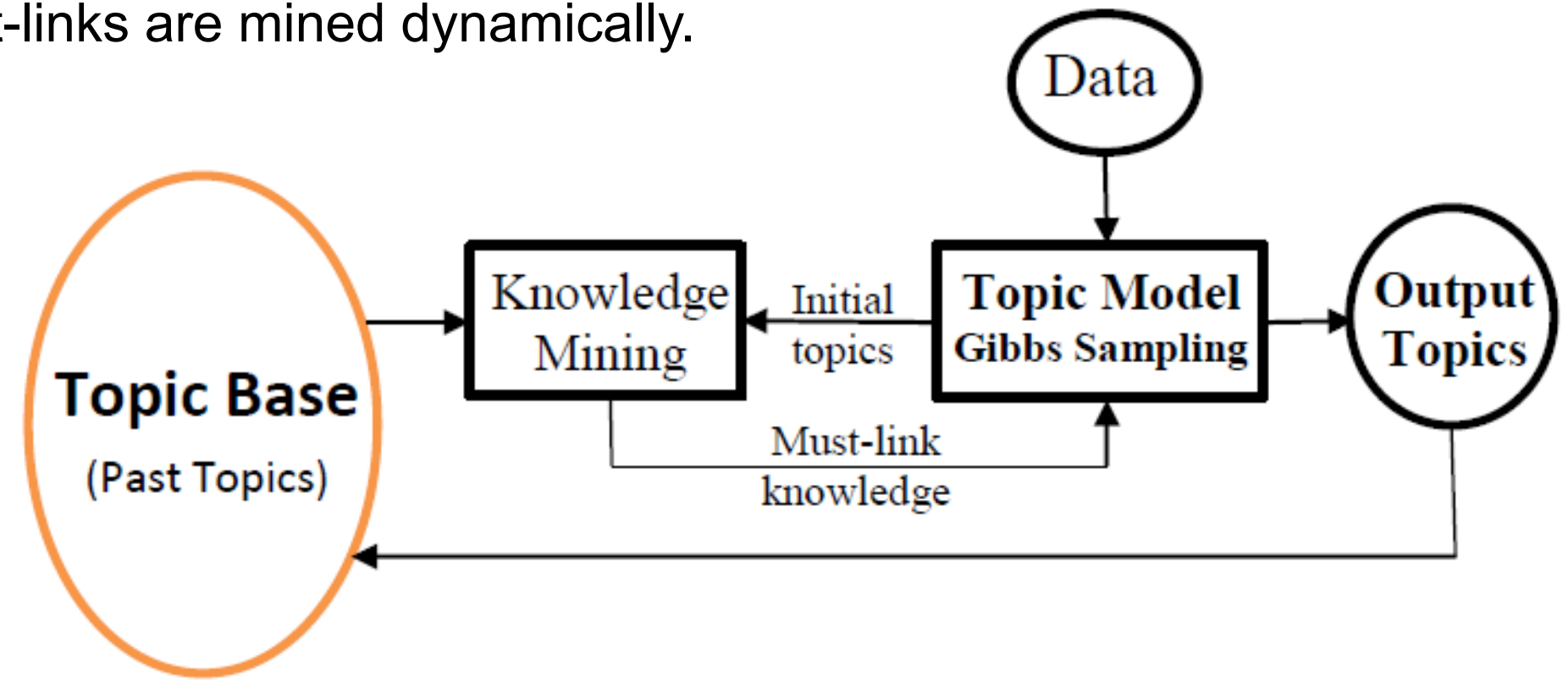
- Should not be in the same aspect/topic

  => Cannot-Links

  e.g., {battery, picture}

Lifelong Topic Modeling (LTM)

- Must-links are mined dynamically.



Chen and Liu. Topic Modeling using Topics from Many Domains, Lifelong Learning and Big Data. ICML-2014.

LTM Model

- Step 1: Run topic modeling (e.g., LDA) on each task/domain D_i to produce a set of topics S_i called Topic Base
- Step 2: Mine prior knowledge (must-links) and use knowledge to guide modeling.

Algorithm 2 LTM(D^t, S)

1: $A^t \leftarrow$ GibbsSampling(D^t, $\emptyset$, N); // Run N Gibbs iterations with no knowledge (equivalent to LDA).
2: **for** $i = 1$ **to** N **do**
3: $K^t \leftarrow$ KnowledgeMining(A^t, S);
4: $A^t \leftarrow$ GibbsSampling(D^t, K^t, 1); // Run with knowledge K^t.
5: **end for**

Knowledge Mining Function

- Topic matching: find similar topics from the topic base for each topic in the new domain
- Pattern mining: find frequent itemsets from the matched topics

Algorithm 3 KnowledgeMining(A^t, S)

1: **for** each p-topic $s_k \in S$ **do**
2: $j^* = \min_j$ KL-Divergence(a_j, s_k) for $a_j \in A^t$;
3: **if** KL-Divergence(a_{j^*}, s_k) $\leq \pi$ **then**
4: $M^t_{j^*} \leftarrow M^t_{j^*} \cup s_k$;
5: **end if**
6: **end for**
7: $K^t \leftarrow \cup_{j^*}$ FIM($M^t_{j^*}$); // Frequent Itermset Mining.

An Example

- Given a newly discovered topic:

  {*price*, *book*, *cost*, *seller*, *money*}

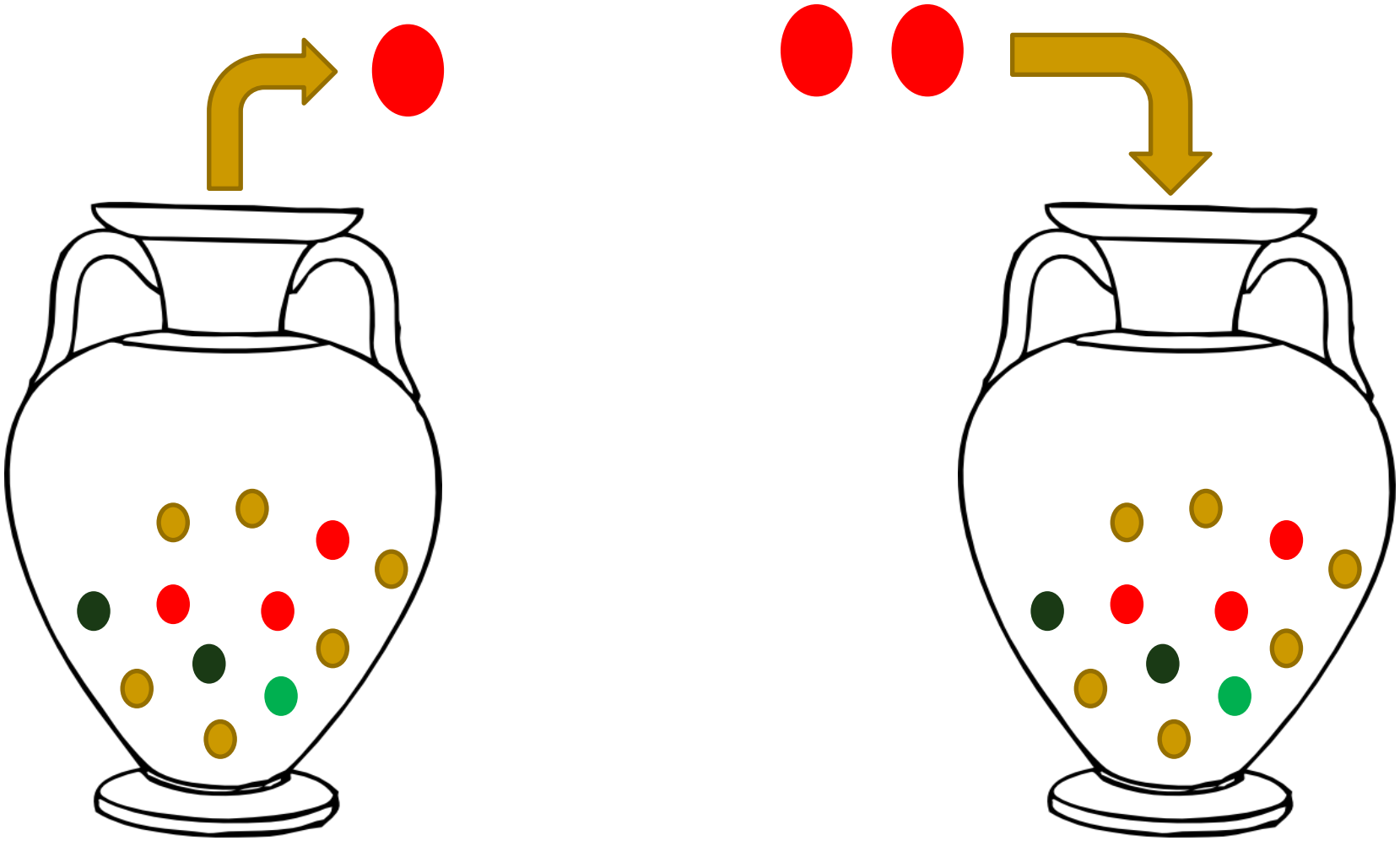
  - We find matching topics from topic base *S*
    - Domain 1: {*price*, *color*, *cost*, *life*, *picture*}
    - Domain 2: {*cost*, *screen*, *price*, *expensive*, *voice*}
    - Domain 3: {*price*, *money*, *customer*, *expensive*}
- If we require words to appear in at least two domains, we get two must-links (knowledge):
  - {*price*, *cost*} and {*price*, *expensive*}.
  - Each set is likely to belong to the same aspect/topic.

Model Inference: Gibbs Sampling

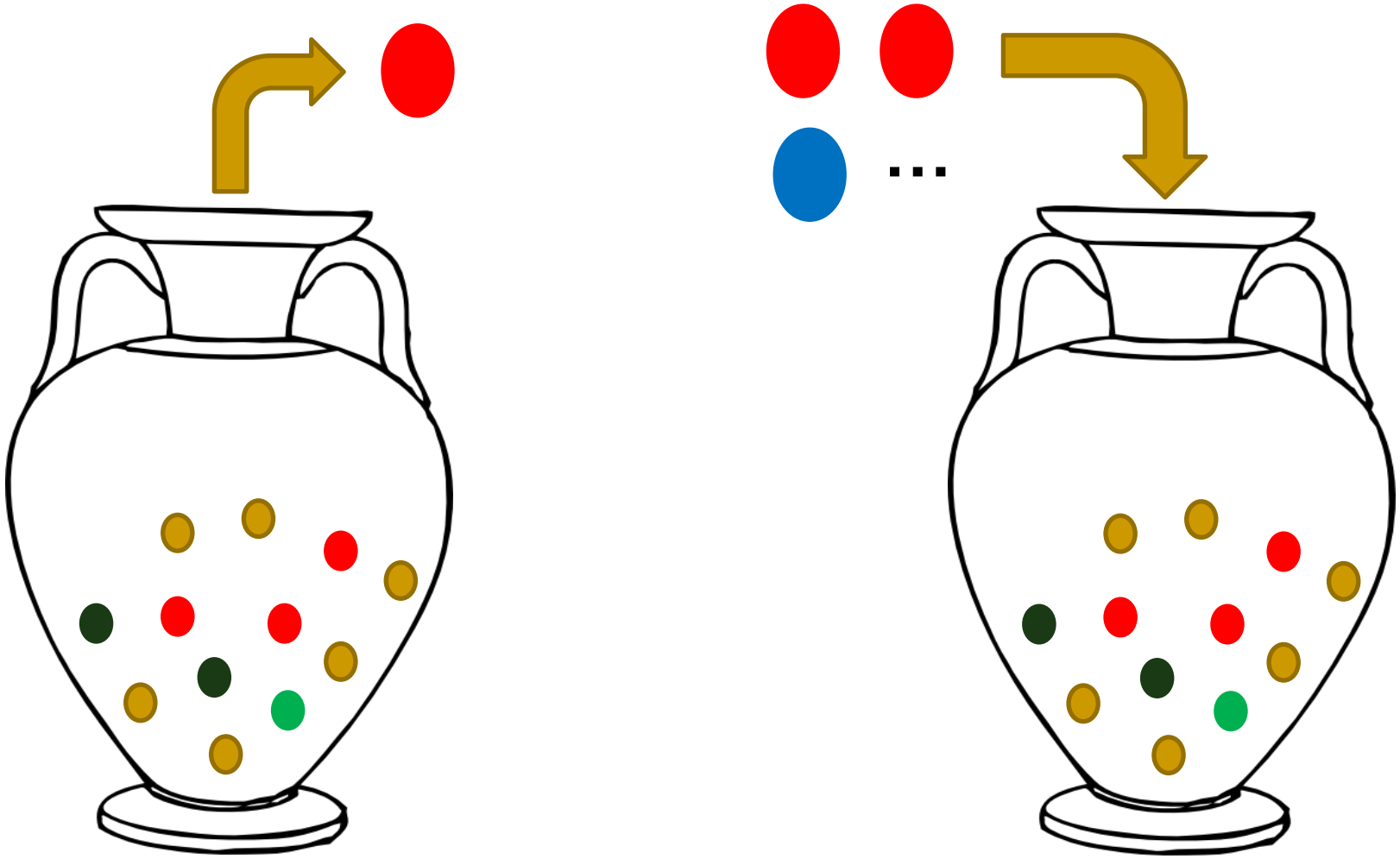
- How to use the *must-links* knowledge?
  - e.g., {*price*, *cost*} & {*price*, *expensive*}

- Graphical model: same as LDA (Latent Dirichlet allocation)
- But the model inference is very different
  - Generalized Pólya Urn Model (GPU)
- Idea: When assigning a topic *t* to a word *w*, also assign *a fraction of t* to words in must-links sharing with *w*.

Simple Pólya Urn model (SPU)



Generalized Pólya Urn model (GPU)



Experiment Results

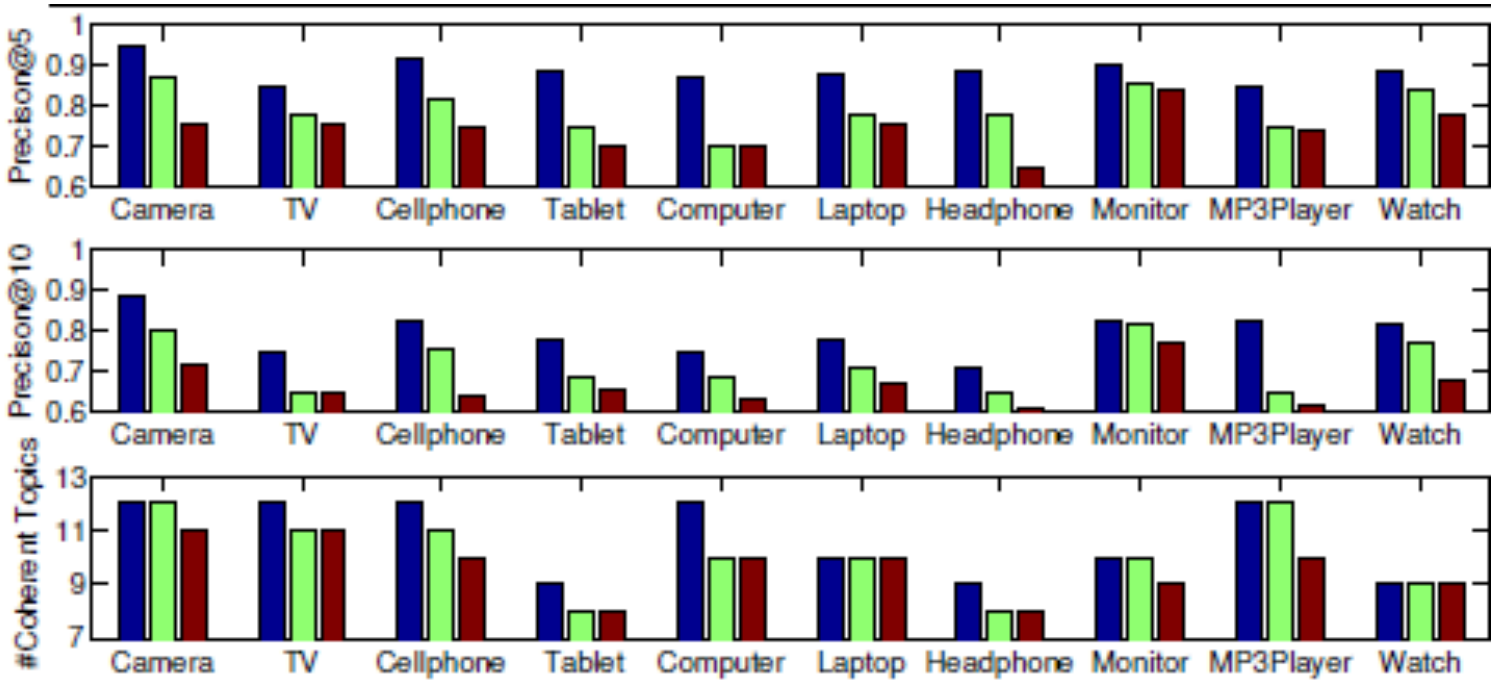


Figure 2. Top & Middle: Topical words *Precision*@5 & *Presicion*@10 of coherent topics of each model respectively; Bottom: number of coherent (#Coherent) topics discovered by each model. The bars from left to right in each group are for LTM, LDA, and DF-LDA. On average, for *Precision*@5 and

Early work vs. more recent work

- Early research of lifelong/continual learning focused on the task incremental learning (TIL) setting.
  - A separate model is built for each task.
    - No catastrophic forgetting
  - Knowledge transfer across tasks is the main goal.
- Recent deep learning based continual learning
  - Build all task models within a single neural network
  - Deal with both catastrophic forgetting and knowledge transfer
  - Investigate all three settings (TIL, CIL and DIL) of continual learning

Task Incremental Learning

Tatsuya Konishi

KDDI Research, Inc. / Visiting scholar at UIC

Goals of this talk

- To be helpful for those who're interested in continual learning (CL) and want to apply CL techniques to your own applications/data.
  - TIL is the simplest scenario of CIL, but it handles many real world aplications.
    - Humans can give instructions to models in the form of task ID.
      e.g., classification over fine-grained labels of animals (CV), sentiment analysis/text summarization (NLP).
  - You can step in the world of CL through learning of TIL.
    - Next two talks by Gyuhak Kim and Zixuan Ke will explain CIL and DIL, respectively, which are more challenging scenarios!
- This talk gives …
  - The standard TIL setup.
  - Simple introduction of representative CL methods which are helpful for you all to know the trend of ideas on CL.

Agenda

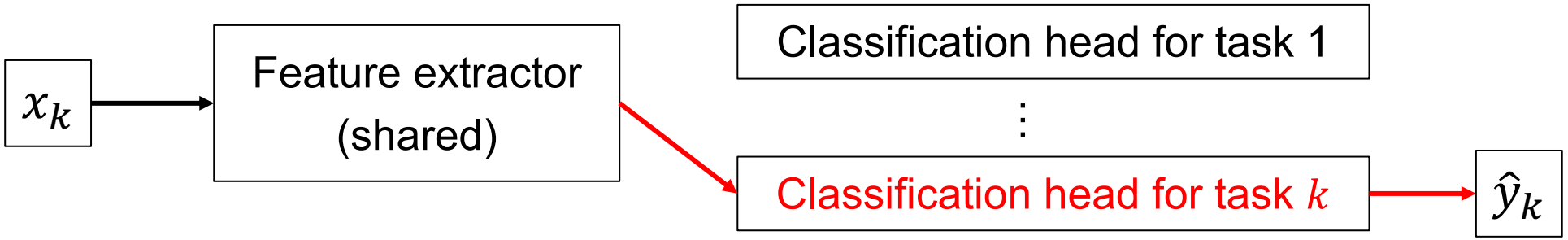
1. **What is TIL?**

2. Categorization in CL
   & Deep dive into representative methods

3. Future directions

What is TIL?

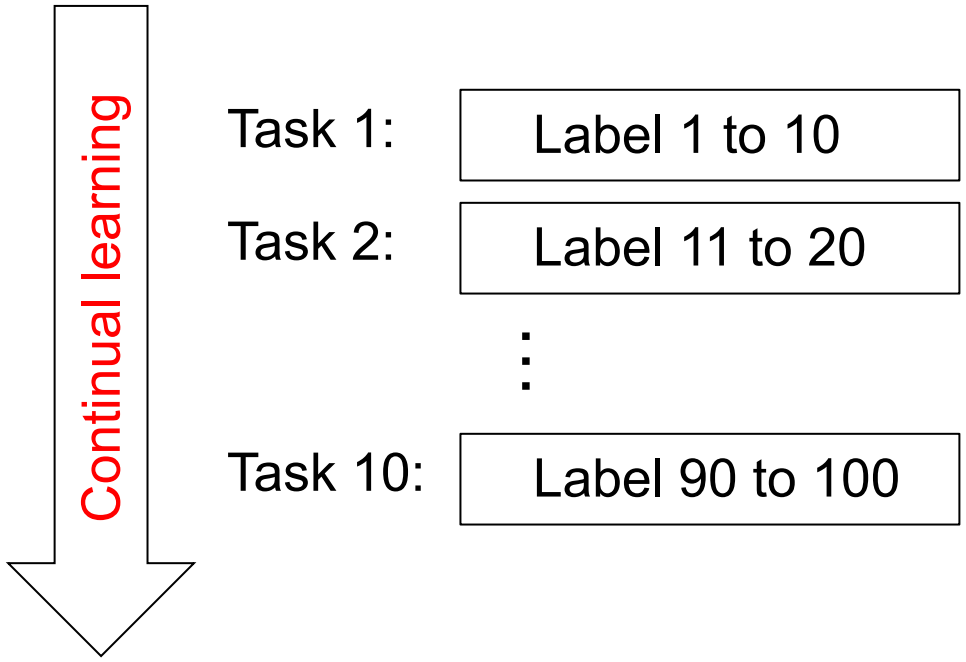
- TIL is the simplest scenario in CL.
  - TIL learns a sequence of tasks, $1,2,\ldots,T$. Each task $k \in \mathbf{T} = \{1,2,\ldots,T\}$ has a set of labels, $\mathbf{Y}_k$, and its corresponding data, $\mathbf{X}_k$.
    The goal is to build a model $f: \mathbf{X} \times \mathbf{T} \rightarrow \mathbf{Y}$ to identify the class labels $y_k \in \mathbf{Y}_k$ for a test instance x_k from task k.
  - Both in training and testing, a model requires task ID to identity a correct classification head.



- Objective: Plasticity-stability dilemma
  - (Stability) Preventing forgetting and (Plasticity) Knowledge transfer
  - Some TIL approaches have achieved no fogetting, but balancing stability-plasticity still remains an interesting topic.

How to evaluate TIL systems?

- Common datasets
  - Split CIFAR100 (into T tasks, e.g., $T = 10$)



- Metrics

T is the total number of tasks.
$R_{i,j}$ is the test accuracy on task j after learning task i.

  - Average accuracy

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

  - Forgetting ratio (negative values mean forgetting)

$$\frac{1}{T-1}\sum_{i=1}^{T-1}(R_{i,i} - R_{i,T})$$

After learning all tasks

Just after learning task i

Agenda

1. What is TIL?

2. **Categorization in CL**
   & Deep dive into representative methods

3. Future directions

Categories of CL methods

- Regularization
  - Concept
    - Penalize changes to important parameters
  - Approaches
    - Function regularization
      - LwF
    - Weight regularization
      - EWC
      - UCL

- Replay
  - Concept
    - Store some previous samples to be learned together with the learning of a new current task
  - Approaches
    - Experience replay
      - GEM
    - Generative replay
      - DGR

- Architecture
  - Concept
    - Let each task monopolize a own sub-network inside the whole network
  - Approaches
    - Model decomposition
      - CCLL
    - Modular network
      - Progressive Networks
      - PathNet
    - Parameter isolation
      - HAT
      - SupSup
      - CAT
      - PRM

- Optimization
  - Concept
    - Control the loss such that forgetting can be alleviated
  - Approaches
    - Gradient projection
      - CUBER
    - Loss landscape
      - Stable-SGD

Regularization>Function regularization | LwF

- LwF (*L*earning *w*ithout *F*orgetting) is one of the earliest work, which shows that preserving outputs between old and new models is an effective strategy to mitigate forgetting.
  - LwF uses the Knowledge Distillation loss to ensure the output probabilities are close to the recorded output probabilities of the previous tasks' network.

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

Not to change output of the previous model during learning of the new task.

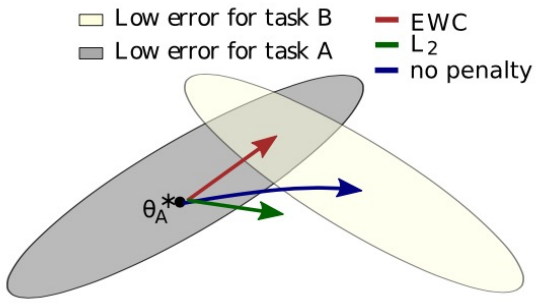
LwF, https://arxiv.org/abs/1606.09282, Learning without Forgetting. Li and Hoiem, In Proc. of ECCV, 2016.

Regularization>Weight regularization | EWC

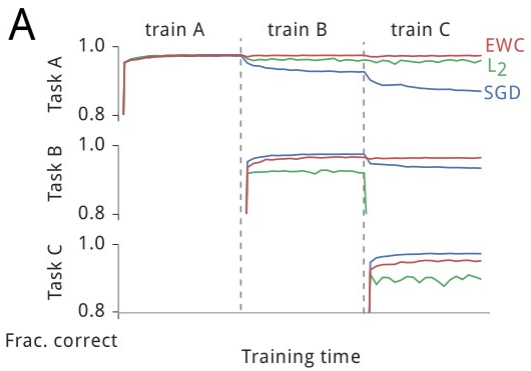
- EWC (*E*lastic *W*eight *C*onsolidation) is an early work in CL.
- EWC selectively slows down learning on the weights important for previous tasks.
  - The posterior distribution, $p(\theta|D_A)$, is regarded as the importance for task A.
  - Since the true posterior probability is intractable, the Laplace approximation is applied to approximate it as a Gaussian distribution with a diagonal precision given by the diagonal of the Fisher information matrix, F.
  - The Fisher information matrix works as a regularization term to suppress parameter changes.

Regularization

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



EWC ensures task A is remembered while training on task B.



Results on permuted MNIST tasks.

EWC, http://arxiv.org/abs/1801.01423, Overcoming catastrophic forgetting in neural networks. Kirkpatrick et al., PNAS, 2017.

Regularization>Weight regularization | UCL

- UCL (*U*ncertainty-regularized *C*ontinual *L*earning) proposes node-wise uncertainty based on Bayesian learning framework, with two additional regularization terms.

  Weights with small uncertainty should be treated as important.

  → Weights with more uncertainty are going to be updated more easily.

$$\frac{1}{2}\sum_{l=1}^{L}\Bigg[\underbrace{\left\|\frac{\boldsymbol{\mu}_t^{(l)}-\boldsymbol{\mu}_{t-1}^{(l)}}{\boldsymbol{\sigma}_{t-1}^{(l)}}\right\|_2^2}_{(a)}+\underbrace{\mathbf{1}^\top\left\{\left(\frac{\boldsymbol{\sigma}_t^{(l)}}{\boldsymbol{\sigma}_{t-1}^{(l)}}\right)^2-\log\left(\frac{\boldsymbol{\sigma}_t^{(l)}}{\boldsymbol{\sigma}_{t-1}^{(l)}}\right)^2\right\}}_{(b)}\Bigg],$$

  Weights indentified as important in previous tasks should be kept as important for future tasks, which is helpful to prevent forgetting.

- Drawbacks
  - Though UCL can effectively balance plasticity and stability, it still suffers from forgetting of old tasks.

UCL, https://arxiv.org/abs/1905.11614, Uncertainty-based Continual Learning with Adaptive Regularization. Ahn et al., In Proc. of NeurIPS, 2019.

Categories of CL methods

- Regularization
  - Concept
    - Penalize changes to important parameters
  - Approaches
    - Function regularization
      - LwF
    - Weight regularization
      - EWC
      - UCL

Replay methods are most popular in the CIL setup, which are also discussed in the next talk by Gyuhak Kim.

- Replay
  - Concept
    - Store some previous samples to be learned together with the learning of a new current task
  - Approaches
    - Experience replay
      - GEM
    - Generative replay
      - DGR

- Architecture
  - Concept
    - Let each task monopolize a own sub-network inside the whole network
  - Approaches
    - Model decomposition
      - CCLL
    - Modular network
      - Progressive Networks
      - PathNet
    - Parameter isolation
      - HAT
      - SupSup
      - CAT
      - PRM

- Optimization
  - Concept
    - Control the loss such that forgetting can be alleviated
  - Approaches
    - Gradient projection
      - CUBER
    - Loss landscape
      - Stable-SGD

Replay>Experience replay | GEM

- GEM (*G*radient *E*pisodic *M*emory) is one of the most popular replay-based methods. It stores a few samples for previous tasks that are used together in learning of a new task.
  - GEM simply collects last m examples from each task, then optimizes the model such that the loss for previous tasks with the examples decreases (i.e., getting improved), by exploiting gradients over examples.

- This paper is also known for proposing some new evaluation metrics in addition to accuracy:
  - Backward transfer: $\frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i}$

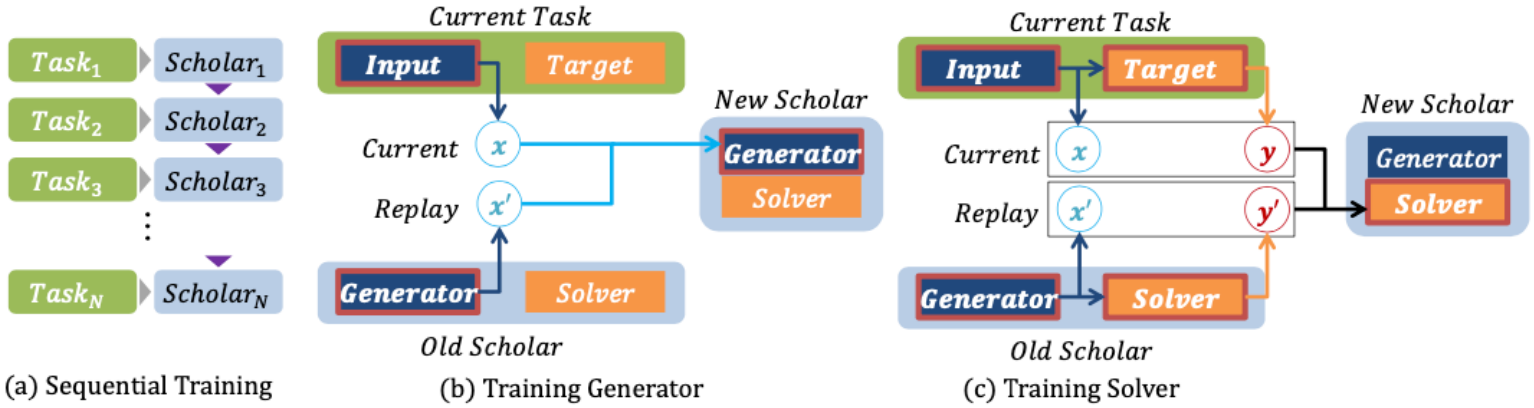
  - Forward transfer: $\frac{1}{T-1}\sum_{2}^{T} R_{i-1,i} - \bar{b}_i$

where $R_{i,j}$ is the test accuracy of the model on task j after observing the last sample from task i, and $\bar{b}_i$ is the test accuracy for task i at random initialization.

GEM, https://arxiv.org/abs/1706.08840, Gradient Episodic Memory for Continual Learning. Lopez-Paz and Ranzato, In Proc. of NeurIPS, 2017.

Replay>Generative replay | DGR

- Simply replaying all previous data alleviate the forgetting, but it requires a large memory.
- DGR (*D*eep *G*enerative *R*eplay) is the first work that employs an additional generative model to replay generated data. Learning each task is accomplished with replaying generated data sampled from the old generative model.
- In training a new task, two independent procedures are involved:
  - "generator", a deep generative model, learns to reconstruct the cumulateive input space.
  - "solver", a task solving model, is trained to couple the inputs and targets.



DGR, https://arxiv.org/abs/1705.08690, Continual Learning with Deep Generative Replay, Shin et al., In Proc. of NeurIPS, 2017.

Categories of CL methods

These methods are probably most effective for TIL, as they can achieve learning without forgetting.

- Regularization
  - Concept
    - Penalize changes to important parameters
  - Approaches
    - Function regularization
      - LwF
    - Weight regularization
      - EWC
      - UCL

- Replay
  - Concept
    - Store some previous samples to be learned together with the learning of a new current task
  - Approaches
    - Experience replay
      - GEM
    - Generative replay
      - DGR

- Architecture
  - Concept
    - Let each task monopolize a own sub-network inside the whole network
  - Approaches
    - Model decomposition
      - CCLL
    - Modular network
      - Progressive Networks
      - PathNet
    - Parameter isolation
      - HAT
      - SupSup
      - CAT
      - PRM

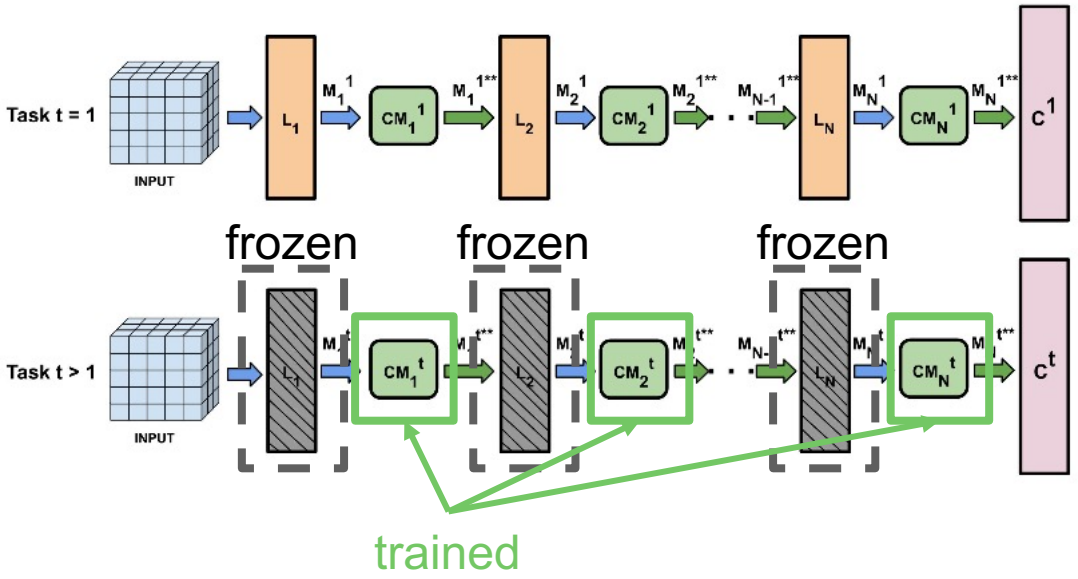
- Optimization
  - Concept
    - Control the loss such that forgetting can be alleviated
  - Approaches
    - Gradient projection
      - CUBER
    - Loss landscape
      - Stable-SGD

Architecture>Model decomposition | CCLL

- CCLL (*C*alibrating *C*NNs for *L*ifelong *L*earning) effectively re-uses the features trained on the initial task, and efficiently (re)calibrates them, using a very small number of calibration parameters.

- CCLL is trained as follows:
  - For the first task ($t = 1$), all modules are trained.
  - For all the subsequent tasks ($t > 1$), the base modules, colored in gray, are frozen. Only calibration modules, colored in green, are trained for each task.

- Drawbacks
  - The performance highly depends on the first task.
  - Backward transfer is impossible.



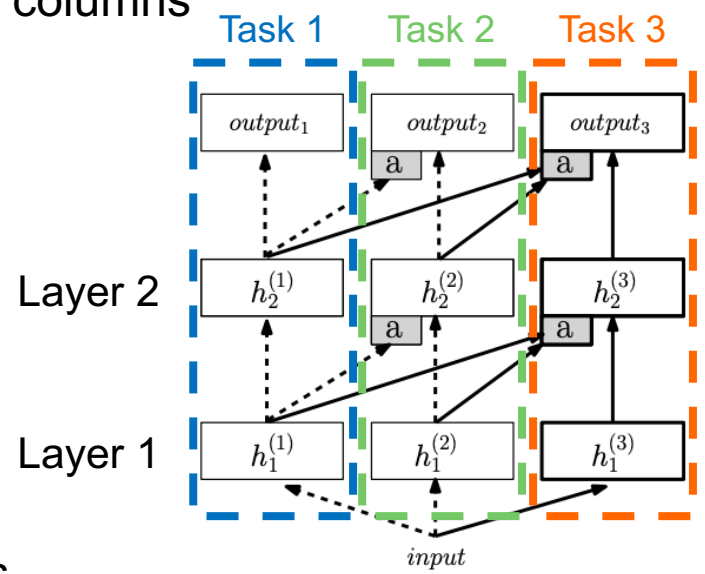
CCLL, https://dl.acm.org/doi/10.5555/3495724.3497031, Calibrating CNNs for Lifelong Learning. Singh et al., In Proc. of NeurIPS, 2020.

Architecture>Modular network | Progressive Networks

- PGN (*P*rog*g*ressive *n*etworks) tries both to
  - prevent forgetting by instantiating a new network (a column) for each task
  - enable transfer via lateral connections to features of previous learned columns

- Since modules used for previous tasks are not updated, PGN does not suffer from forgetting.

- Drawbacks
  - A new column may be added, which may induce another issue, e.g., the growth in the number of parameters with the number of tasks.
  - Backward transfer is impossible.



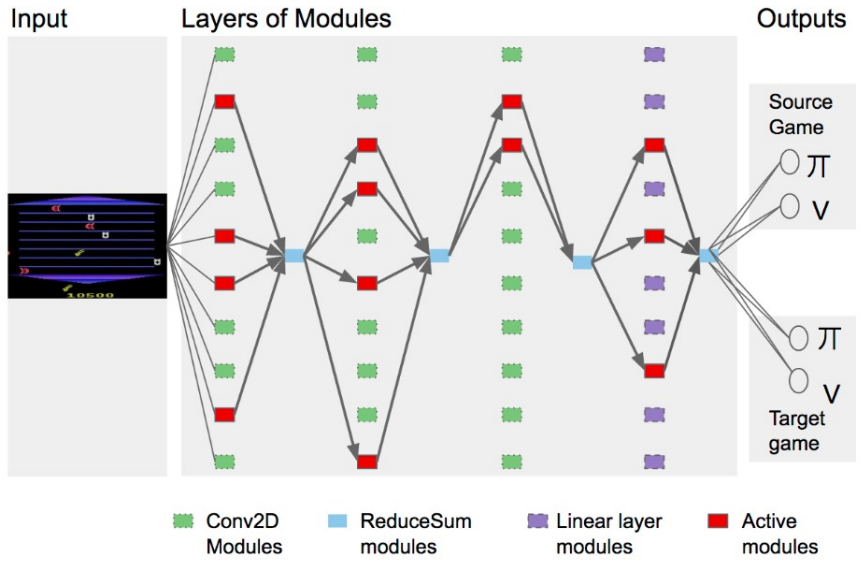
Progressive Networks, https://arxiv.org/abs/1606.04671, Progressive Neural Networks. Rusu et al., 2016.

Architecture>Modular network | PathNet

- PathNet has L layers with each layer consisting of M modules. Each module is a neural network, e.g., convolution or linear.
- Then, learning decides which modules are assigned to each task based on a tounament selection genetic algorithm.

- Since the model remembers which modules are used for a task and uses them in testing, no forgetting happens.

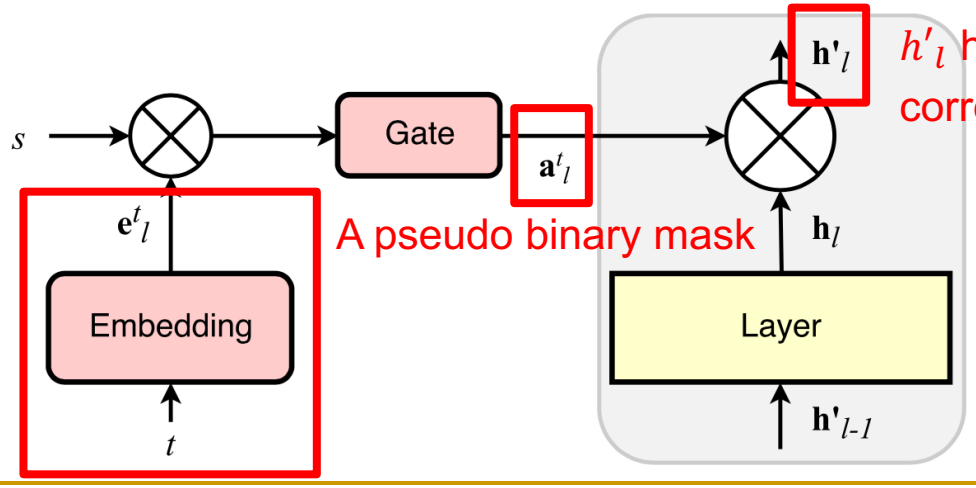
- Drawbacks
  - Later tasks will have limited numer of parameters to use.
  - Backward transfer is impossible.



PathNet, https://arxiv.org/abs/1701.08734, PathNet: Evolution Channels Gradient Descent in Super Neural Networks. Fernando et al., 2017.

Architecture>Parameter isolation | HAT (1/2)

- HAT (*H*ard *A*ttention to the *T*ask) is the first work that dynamically allocates each task a subnetwork by using masks of neurons and modifies parameters' gradients to prevent forgetting.
- In the forward pass of learning task t:
  - HAT optimizes not only model parameters but also task embeddings, e_l^t, for each layer l.
    Through a sigmoid function, a mask, $a_l^t = \sigma(se_l^t)$, is computed. As s is a large number, a_l^t is expected to behave as a binary function mask, taking 0 or 1.
  - Based on the mask, neurons of the layer are either activated (by multiplying 1) or deactivated (by multiplying 0), being passed to the next layer.
  - That is, a_l^t can be regarded as importance of neurons for that task.



h'_l has non-zero values only when its corresponding mask values of a_l^t are 1.

Task embedding is also trained.

HAT, https://arxiv.org/abs/1801.01423, Overcoming Catastrophic Forgetting with Hard Attention to the Task, Serrà et al., In Proc. of ICML, 2018.

Architecture>Parameter isolation | HAT (2/2)

- In the backward pass of learning task t:
  - The importance of neurons, a_l^t, has been accumulated in CL from task 1 to $t-1$.

An accumulated mask: $a_l^{\leq t-1} = \max\left(a_l^{t-1}, a_l^{\leq t-2}\right)$

If a neuron has importance 1 for any of previous task 1 to $t-1$, $a_l^{\leq t-1}$ has importance 1 for the neuron.

  - The gradients of parameters in layer l, $g_{l,ij}$, are modified such that parameters feeding to the important neurons are not going to be changed. The modified ones, $g'_{l,ij}$, are acutally used in the optimization.

$$g'_{l,ij} = \left[1 - \min\left(a_{l,i}^{\leq t-1}, a_{l-1,j}^{\leq t-1}\right)\right] g_{l,ij}$$

Parameters connecting important neuron j of layer $l-1$ (with 1 for $a_{l-1,j}^{\leq t-1}$) and neuron i of layer l (with 1 of $a_{l,i}^{\leq t-1}$) is not updated.

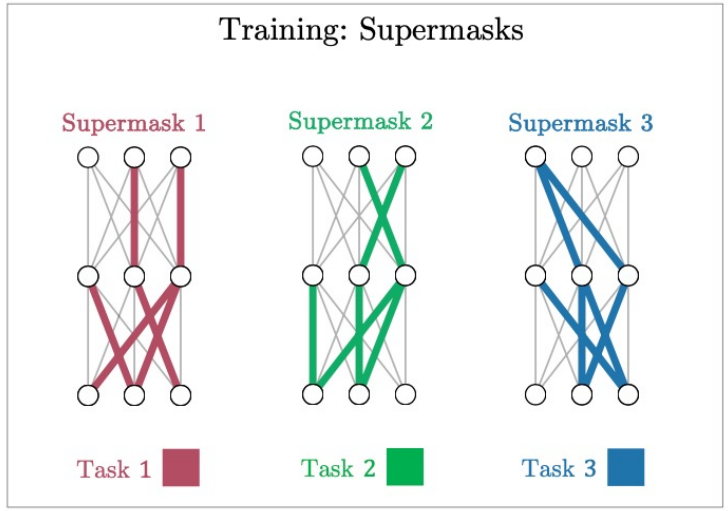
- Results
  - In the forward pass (in testing), binary masks from task embeddings are used to select neurons by which the model can recall the sub-network dedicated to the task without forgetting.

- Drawbacks
  - Later tasks will have limited number of "alive" parameters.
  - Backward transfer is impossible.

Architecture>Parameter isolation | SupSup (1/2)

- SupSup (*Sup*ermasks in *Sup*erposition) applies the lottery ticket hypothesis to CL.
  - What is the lottery ticket hypothesis (which was proposed in ICLR2019)?
    - The number of possible subnetworks is combinatiorial in the number of parameters.
    - It has been observed that the number of combinations is large enough that even within randomly weighted neural networks.
    - There exist *supermasks* that create corresponding subnetworks for tasks to achieve good performance.

- For each task t, the model trains only a binary mask M^t and does not train parameters in the network W itself.
  - Although the binary mask M^t needs to be stored per task, since it's binary and sparse, it requires only a small storage.
  - Then, output for task t, $f(x, W \odot M^t)$, can always retrieve the correct subnetwork, without forgetting.



SupSup, https://arxiv.org/abs/2006.14769, Supermasks in Superposition, Wortsman et al., In Proc. of NeurIPS, 2020.

Architecture>Parameter isolation | SupSup (2/2)

- Supermask training with Edge-Popup $y = f\left(x, W \odot M^t\right)$
  - This technique was proposed by another work.
  - The Edge-Popup alg. learns a score matrix S, which has the same size as of W or M, and computes the mask via $M = h(S)$, where the function h sets the top k% of entries in S to 1 and the rest to 0.
  - Simply, it keeps some (top k%) edges important to performance while pruning the remaining ones.

- Drawbacks
  - Neither forward nor backward transfer is possible, because each task has a completely separate subnetwork.

Architecture>Parameter isolation | CAT

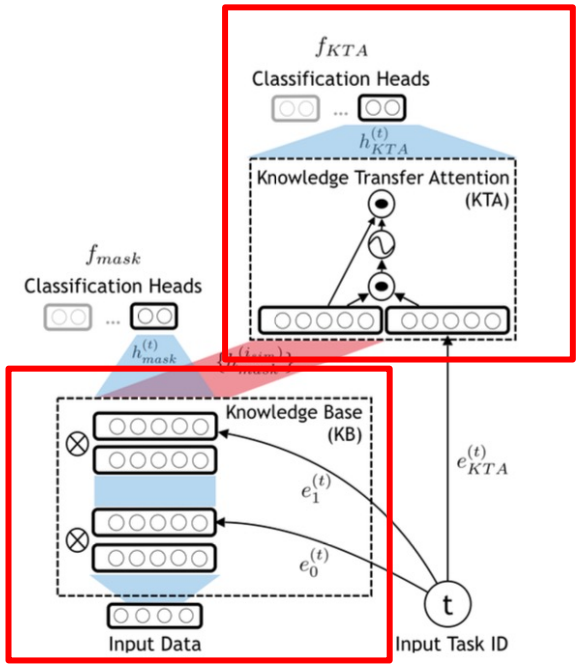
- CAT (*C*ontinual learning with forgetting *A*voidance and knowledge *T*ransfer) is the first work that enables transfer among similar tasks by extending HAT.
  - Evaluated with not only dissimilar tasks (e.g., CIFAR100 with 10 tasks), but also mixed sequences that consist both dissimilar and similar tasks.

- To detect similar tasks, two models are compared.
  - Transfer model actually tries to transfer each of previous task to a new task.
  - Reference model does NOT try to transfer any previous task.

  If the transfer model outperforms the reference model, the task is regarded as similar.

- Those detected similar tasks are then transferred via the attention to the new task, which leads to better knowledge transfer.
  Since dissimilar tasks are supposed to non-similar,
  the HAT-like masking still prevents forgetting for them.

Only similar tasks are transferred



Same as HAT

CAT, https://arxiv.org/abs/2112.10017, Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks. Ke et al., In Proc. of NeurIPS, 2020.

Architecture>Parameter isolation | PRM

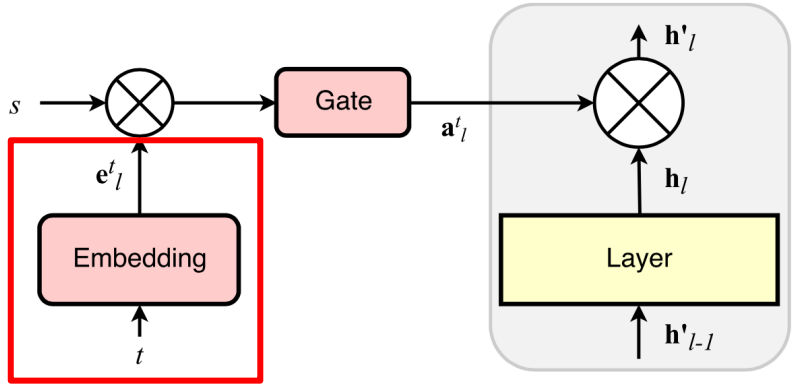
- PRM (*P*artially *R*elaxed *M*asks) exploits task similarity in a different way than CAT, using HAT's scheme as well.
- PRM finds that task similarity can be computed via clustering of task embeddings that control the binary masks.

- When training, basically following HAT's algorithm, PRM accumulates only masks of dissimilar tasks, which promotes knowledge transfer among similar tasks.

HAT: $$a_l^{\leq t-1} = \max\left(a_l^{t-1}, a_l^{\leq t-2}\right)$$

PRM: $$a_l^{\leq t-1} = \max\left(a_l^i | i \in \mathcal{D}_l^{t-1}, i \leq t-1\right)$$

where $\mathcal{D}_l^{t-1}$ is the set of dissimilar tasks for task $t-1$ in layer l.



Clustering of task embeddings tells task similarity.

→ (HAT & PRM) Parameter update: $$g'_{l,ij} = \left[1 - \min\left(a_{l,i}^{\leq t-1}, a_{l-1,j}^{\leq t-1}\right)\right] g_{l,ij}$$

PRM, https://dl.acm.org/doi/abs/10.1007/978-3-031-05933-9_29, Partially Relaxed Masks for Knowledge Transfer Without Forgetting in Continual Learning, Konishi et al., In Proc. of PAKDD, 2022.

Categories of CL methods

- Regularization
  - Concept
    - Penalize changes to important parameters
  - Approaches
    - Function regularization
      - LwF
    - Weight regularization
      - EWC
      - UCL

- Replay
  - Concept
    - Store some previous samples to be learned together with the learning of a new current task
  - Approaches
    - Experience replay
      - GEM
    - Generative replay
      - DGR

- Architecture
  - Concept
    - Let each task monopolize a own sub-network inside the whole network
  - Approaches
    - Model decomposition
      - CCLL
    - Modular network
      - Progressive Networks
      - PathNet
    - Parameter isolation
      - HAT
      - SupSup
      - CAT
      - PRM

- Optimization
  - Concept
    - Control the loss such that forgetting can be alleviated
  - Approaches
    - Gradient projection
      - CUBER
    - Loss landscape
      - Stable-SGD

Optimization>Gradient projection | CUBER

- CUBER (*C*ontin*U*al learning method with *B*ackward knowl*E*dge t*R*ansfer) theoretically analyzes the conditions under which updating the learned model of old tasks could lead to backward transfer.

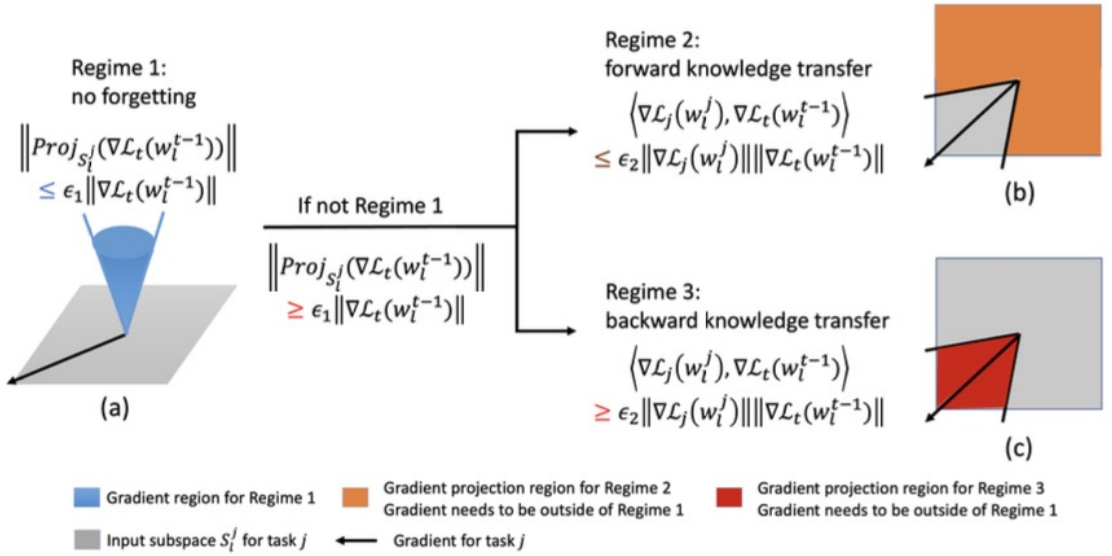
- Specifically, each layer characterizes the task correlation. It detects a regime based on their orthogonality of parameter gradients.
  - Regime 1) There is little knowledge transfer or interference
  - Regime 2) Forward transfer is possible
  - Regime 3) Backward transfer is possible.

In all regimes, the model for a new task is updated based on orthogonal projection for the knowledge protection.

e.g., For regime 1,

$$\nabla\mathcal{L}_t(w) \leftarrow \nabla\mathcal{L}_t(w) - \mathrm{Proj}_{S^j}(\nabla\mathcal{L}_t(w)),$$

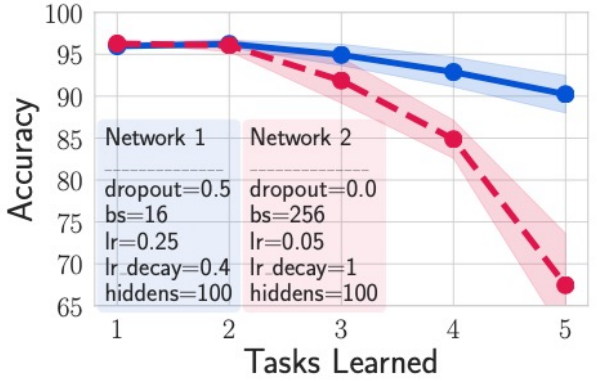
Proj_{S^j} defines the projection on the input subspace S^j of task j, i.e., $\mathrm{Proj}_{S^j}(A) = AB^j(B^j)'$ where B^j is the bases for S^j.



CUBER, https://arxiv.org/abs/2211.00789, Beyond Not-Forgetting: Continual Learning with Backward Knowledge Transfer, Lin et al., In Proc. of NeurIPS, 2022.

Optimization>Loss landscape | Stable-SGD

- Rather than dedicating an algorithm, Stable-SGD enables SGD to find a flat local minimum by adapting the factors in training regime, such as dropout, learning rate, and batch size.



- It hypothesizes that *the wider the convergent points are, the less forgetting happens*. The following are suggested.
  - Dropout reguarization that widens the local minimum (better)
  - Large initial learning rate with exponential decay schedule
  - Small batch size

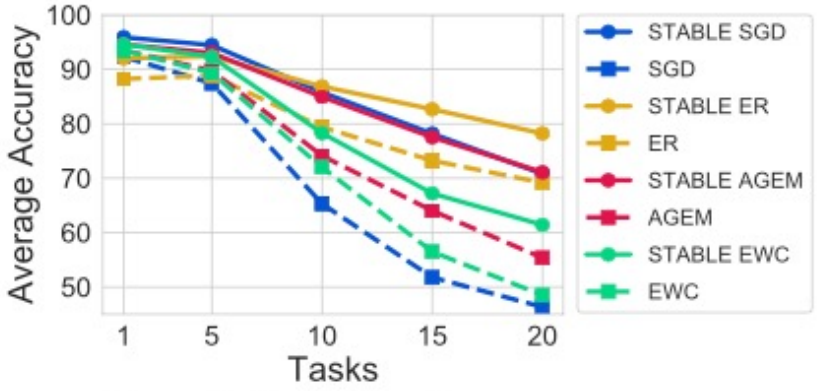


Figure 9: Evolution of average accuracy

Stable-SGD, https://arxiv.org/abs/2006.06958, Understanding the Role of Training Regimes in Continual Learning. Mirzadeh et al., In Proc. of NeurIPS, 2020.

Agenda

1. What is TIL?

2. Categorization in CL
   & Deep dive into representative methods

3. **Future directions**

Review of current situation on TIL

- In TIL, most effective approaches are "architecture"-based ones, such as PathNet, HAT, and SupSup in terms of preventing forgetting.
  - As they enable each task to monopolize its own sub-network, they achieve no-forgetting learning.
  - At the same time, however, this training paradigm limits knowledge transfer among tasks.
  - Recent studies (e.g., CAT and PRM) have tackled TIL problem in which not only preventing forgetting but also promoting transfer are pursued.

What is next for TIL?

- In TIL, the forgetting problem itself has already been resolved by HAT or SupSup.
  But, those approaches' paradigms may also induce another problem.

  For example,
  - Limited knowledge transfer caused by less shared parameters
  - Reduced learnining capacity caused by using only allocated parameters

  Limited research has been done to balance these.

- Connecting TIL with other ML research areas (e.g., self-supervised learning, meta-learning, use of pre-trained models, or federated learning) can be another way to go.

Takeaway

- Not limited to TIL, continual learning methods are grouped into several types.
  In this talk, 4 categories were introduced.
  - Regularization – suppresses the change of important parameters.
  - Replay – store some examples of previous tasks to be used in learning the new current task.
  - Architecture – let each task monopolize a sub-network.
  - Optimization – controls the loss such that forgetting can be alleviated.

- In particular, architecture methods are typically most effective for TIL, which can achieve learning with no forgetting. Representative methods are HAT and SupSup.

- However, those approaches have induced other problems, e.g., limited knowledge transfer or reduced learning capacity for new tasks. Developing TIL methods that can address such issues while preventing forgetting still remains an interesting topic for TIL.

Class Incremental Learning

Gyuhak Kim
University of Illinois Chicago
https://k-gyuhak.github.io

Overview

- Motivation, application, definition
- Challenges
- Problem setups - online, blurry task, offline
- Proposed methods
- Evaluation protocols and datasets
- A theoretical guidance on solving CIL
- CIL using pre-trained models
- Future directions

Class-Incremental Learning (CIL)

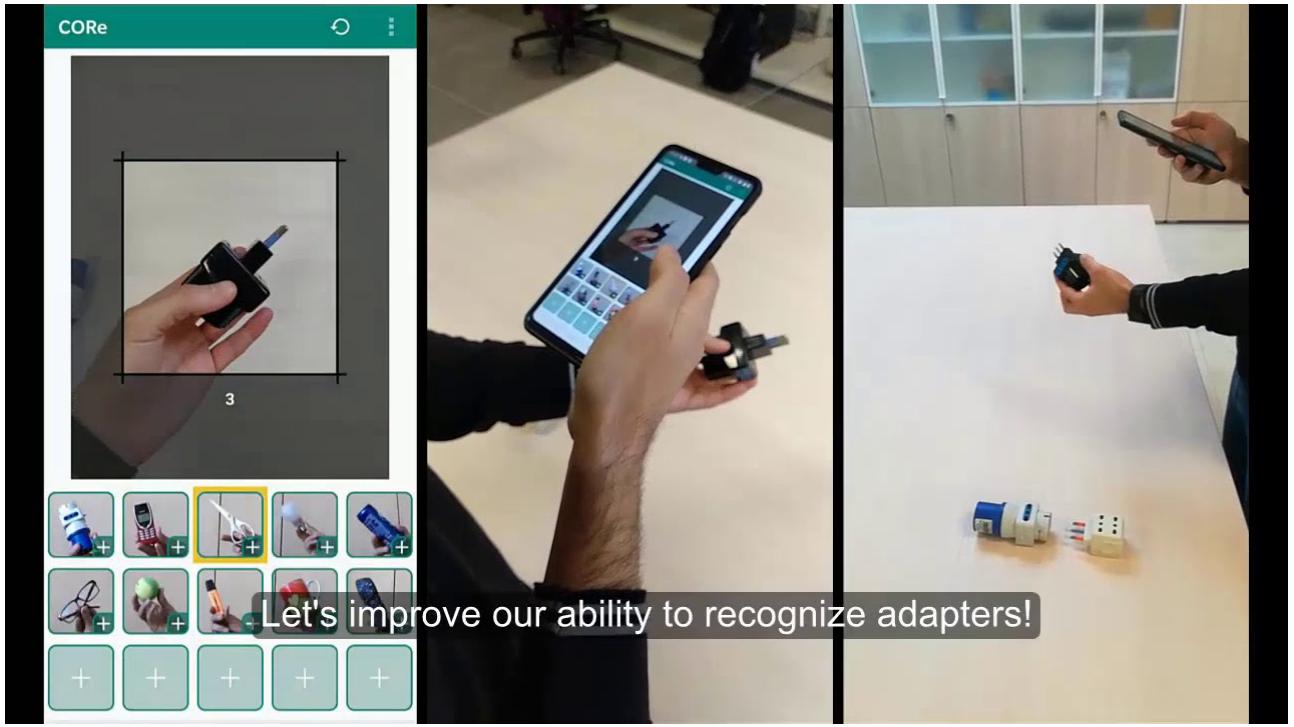
- Different learning problems in continual learning (CL):
  - Task incremental learning (TIL)
  - Class incremental learning (CIL)
  - Domain incremental learning (DIL)
  - etc.

Motivation

- Learning new tasks without forgetting the previous knowledge
  - Limited resource

Application

- Continual object recognition
  - https://www.youtube.com/watch?v=Bs3tSjwbHa4



Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replayfor real-time continual learning. 2019

Definition

Class incremental learning (CIL). CIL learns a sequence of tasks, $1, 2, ..., T$. Each task k has a training dataset $\mathcal{D}_k = \{(x_k^i, y_k^i)_{i=1}^{n_k}\}$, where n_k is the number of data samples in task k, and $x_k^i \in \mathbf{X}$ is an input sample and $y_k^i \in \mathbf{Y}_k$ (the set of all classes of task k) is its class label. All $\mathbf{Y}_k$'s are disjoint ($\mathbf{Y}_k \cap \mathbf{Y}_{k'} = \emptyset$, $\forall k \neq k'$) and $\bigcup_{k=1}^{T} \mathbf{Y}_k = \mathbf{Y}$. The goal of CIL is to construct a single predictive function or classifier $f : \mathbf{X} \rightarrow \mathbf{Y}$ that can identify the class label y of each given test instance x.

- Given test sample x, what is its class?

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, Bing Liu. A Theoretical Study on Solving Continual Learning. In NeurIPS, 2022.

Types of CIL Problems - Online

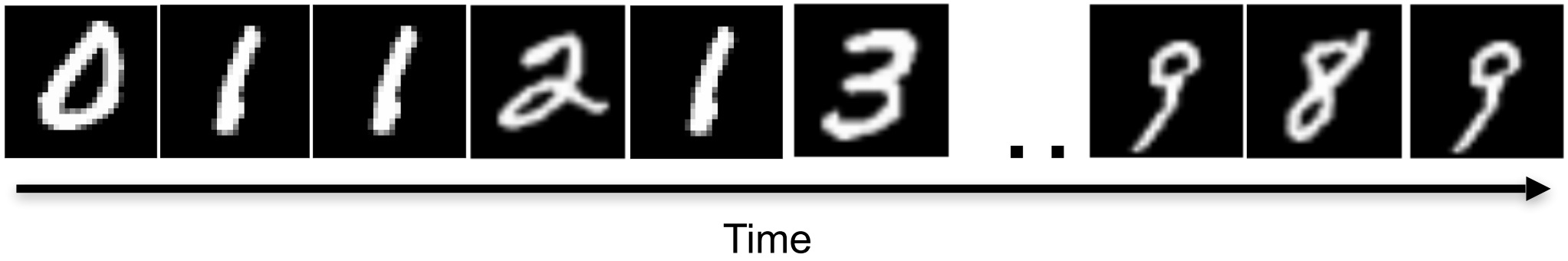
- The system can observe the training samples only once
  - e.g., single epoch

  - Task transition vs. stream

Tyler L. Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In CVPR Workshops, June 2020

Types of CIL Problems - Online (blurry task)

- There is no clear task transition
  - e.g., stream of data. Single sample per batch



- The CIL definition is still valid

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In CVPR, 2021.

Types of CIL Problems - Offline

- The system can observe the training samples multiple times
  - e.g., multiple epochs during training v.s. single epoch
  - Enough time to train the system and collect data

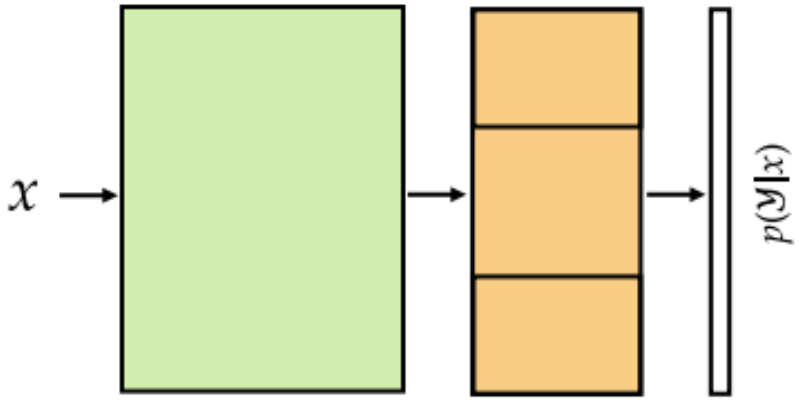
- Some variants (e.g., blurry task) makes less sense

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, Bing Liu. A Theoretical Study on Solving Continual Learning. In NeurIPS, 2022.

Proposed Methods - 4 Categories

- Regularization methods
- Replay-based methods
- Pseudo-replay methods
- Parameter-isolation methods

Proposed Methods - Regularization Methods



- Used for CIL (but any CIL can be used for TIL)
- Exemplar-free
- For task k, minimize

$$\mathcal{L} = -\frac{1}{|D^k|} \sum_{(x,y)\in D^k} \log p(y|x) + \mathcal{R}$$

Proposed Methods - Regularization Methods

- Find important parameters in performing the previous tasks

$$\mathcal{R} = \Omega_k ||\theta - \theta^*_{k-1}||^2$$

- Preserve the input-output mapping (e.g., knowledge distillation)

$$\mathcal{R} = L(f(x;\theta_k), f(x;\theta^*_{k-1}))$$

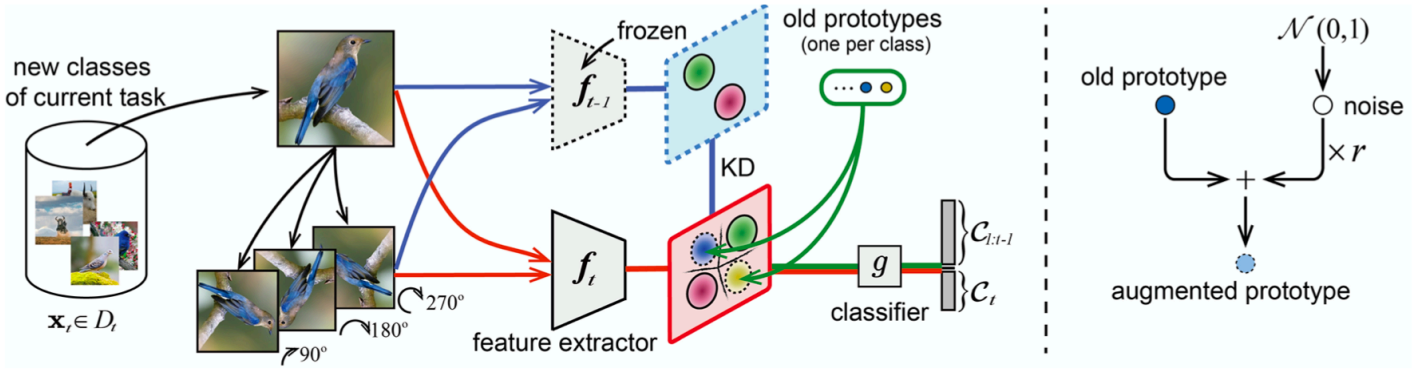
  - Examples of the loss function L are KL-divergence and MSE

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, and Others. Overcoming catastrophic forgetting in neural networks. In PNAS, 2017.
Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In ICML, pp. 3987–3995, 2017.
Zhizhong Li and Derek Hoiem. Learning Without Forgetting. In ECCV, pages 614–629. Springer, 2016.

Proposed Methods - Regularization Methods

- Regularize the features
- Denote the feature extractor by h. Then regularize the features as

$$\mathcal{R} = L(h(x;\theta), h(x;\theta^*_{k-1}))$$



Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In CVPR, 2021.

Proposed Methods - Replay-Based Methods

- Used for CIL (but any CIL can be used for TIL)
- Memory buffer $\mathcal{M}$
- For task k, minimize

$$\mathcal{L} = -\left(\frac{1}{|D^k|} \sum_{(x,y)\in D^k} \log p(y|x) + \frac{1}{|\mathcal{M}|} \sum_{(x,y)\in \mathcal{M}} \log p(y|x) \right) + \mathcal{R}$$

- The regularization R can be one of the regularization methods discussed in the previous slide

Proposed Methods - Replay-Based Methods

- Since the memory buffer is very small, it's important how to leverage it more effectively
  - Which samples to replay?
  - Which samples to save?

Proposed Methods - Replay-Based Methods

- Which samples to replay?
  - For a batch of training samples, select samples that give the maximum interference

Algorithm 1: Experience MIR (ER-MIR)

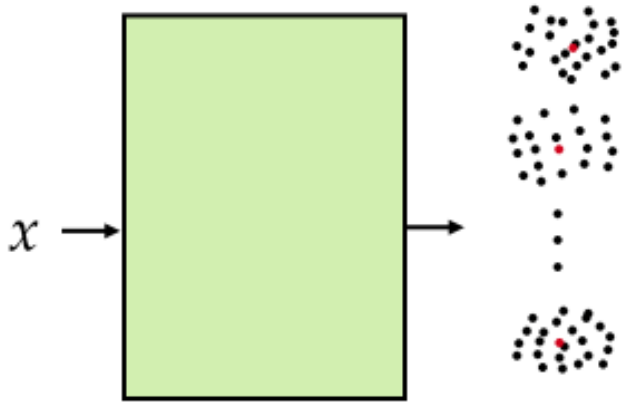
Input: Learning rate α, Subset size C; Budget $\mathcal{B}$

Initialize: Memory $\mathcal{M}$; θ
for $t \in 1..T$ **do**
  for $B_n \sim D_t$ **do**
    %%Virtual Update
    $\theta^v \leftarrow \texttt{SGD}(B_n, \alpha)$
    %Select C samples
    $B_C \sim \mathcal{M}$
    %Select based on score
    $S \leftarrow sort(s_{MI}(B_C))$
    $B_{\mathcal{M}_C} \leftarrow \{S_i\}_{i=1}^{\mathcal{B}}$
    $\theta \leftarrow SGD(B_n \cup B_{\mathcal{M}_C}, \alpha)$
    %Add samples to memory
    $\mathcal{M} \leftarrow UpdateMemory(B_n)$;
  end
end

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, and Others. Online continual learning with maximal interfered retrieval. In NeurIPS. 2019.

Proposed Methods - Replay-Based Methods

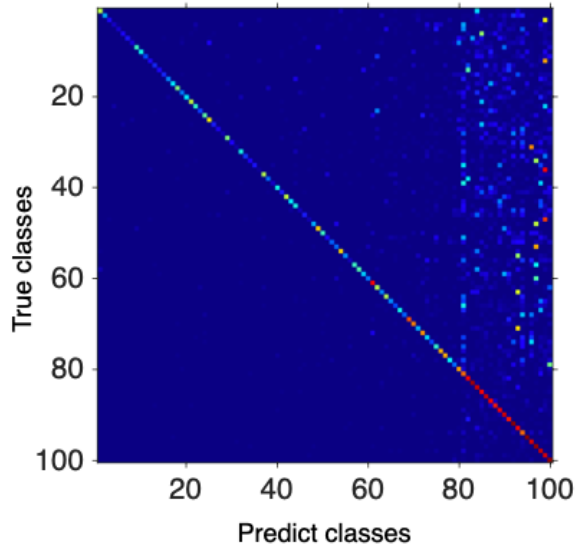
- Which samples to save?
  - Reservoir sampling [Jeffery et al., 1985]. Select random samples from stream
  - Randomly select m samples per class [Chaudhry et al., 2019]
  - Nearest-mean classifier. Select the m training samples that best approximate the mean of training samples [Rebuffi et al., 2017]



Jeffrey S Vitter. Random sampling with a reservoir. ACM Transactions on Mathematical Software (TOMS), 1985
Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, and Others. Continual learning with tiny episodic memories. 2019.
Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In CVPR, 2017.

Proposed Methods - Replay-Based Methods

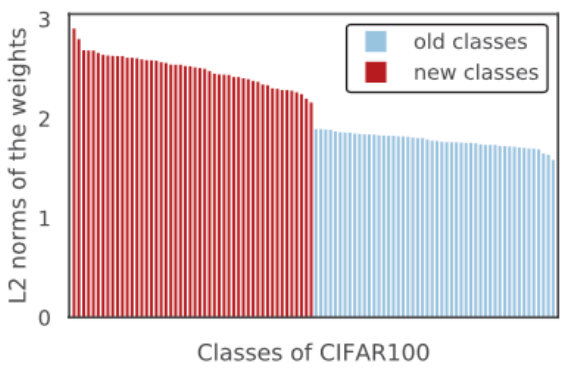
- Due to sample imbalance between memory buffer and current training data, the system is biased towards recent tasks
  - Need to correct the bias
- Bias toward later tasks
  - Samples of earlier tasks are predicted as classes in later tasks

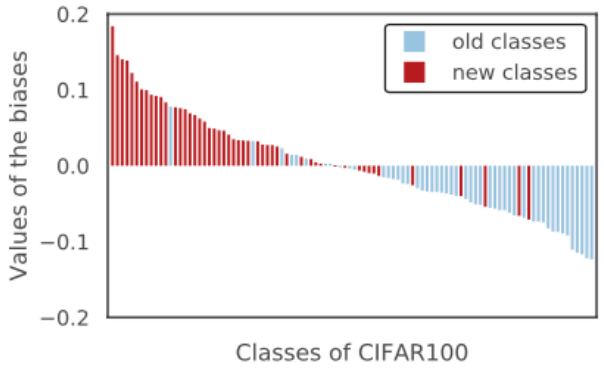


Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In CVPR, 2019.

Proposed Methods - Replay-Based Methods

- The weights w and biases b for old and new classes in the last layer:





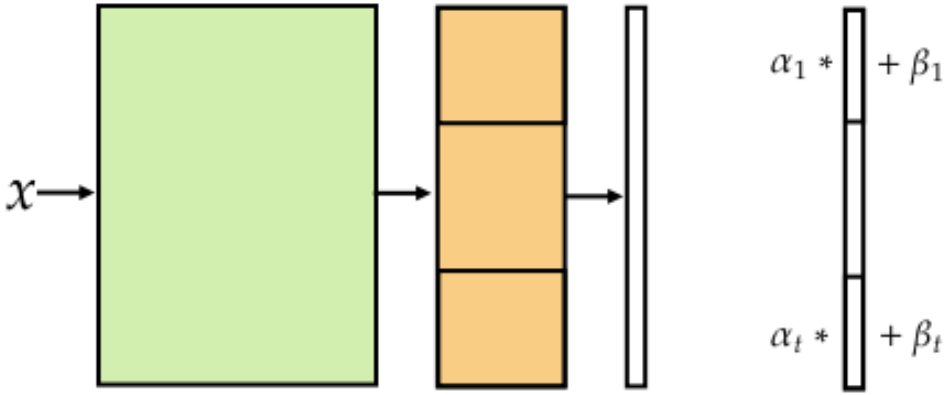
- Correct it by cosine normalization and knowledge distillation

$$p_i(x) = \frac{\exp(\eta\langle\bar{\theta}_i, \bar{f}(x)\rangle)}{\sum_j \exp(\eta\langle\bar{\theta}_j, \bar{f}(x)\rangle)}, \qquad L_{\text{dis}}^{\text{C}}(x) = -\sum_{i=1}^{|\mathcal{C}_o|} \|\langle\bar{\theta}_i, \bar{f}(x)\rangle - \langle\bar{\theta}_i^*, \bar{f}^*(x)\rangle\|,$$

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In CVPR, 2019.

Proposed Methods - Replay-Based Methods

- Correct the bias in output values
  - Find scaling α_k and shifting β_k parameters for the output of each task k to correct the bias

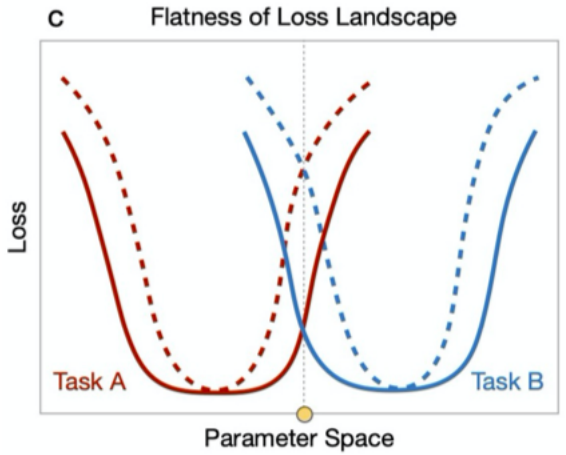


  - Find the parameters via optimization using the held-out samples in the memory buffer

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In CVPR, 2019.

Proposed Methods - Replay-Based Methods

- Find robust parameters to perturbation (i.e., flat minima)
  - One can achieve a better generalization via flat local minima as flat minima is more robust to shift (or perturbation)



- Regularize on sub-optimal output (perturbation) [Buzzega et al., 2020]

Algorithm 2 - Dark Experience Replay ++

Input: dataset D, parameters θ, scalars α and β, learning rate λ

$\mathcal{M} \leftarrow \{\}$

for (x, y) **in** D **do**

$(x', z', y') \leftarrow sample(\mathcal{M})$

$(x'', z'', y'') \leftarrow sample(\mathcal{M})$

$x_t \leftarrow augment(x)$

$x'_t,\ x''_t \leftarrow augment(x'),\ augment(x'')$

$z \leftarrow h_\theta(x_t)$

$reg \leftarrow \alpha \|z' - h_\theta(x'_t)\|_2^2 + \beta\, \ell(y'', f_\theta(x''_t))$

$\theta \leftarrow \theta + \lambda \cdot \nabla_\theta[\ell(y, f_\theta(x_t)) + reg]$

$\mathcal{M} \leftarrow reservoir(\mathcal{M}, (x, z, y))$

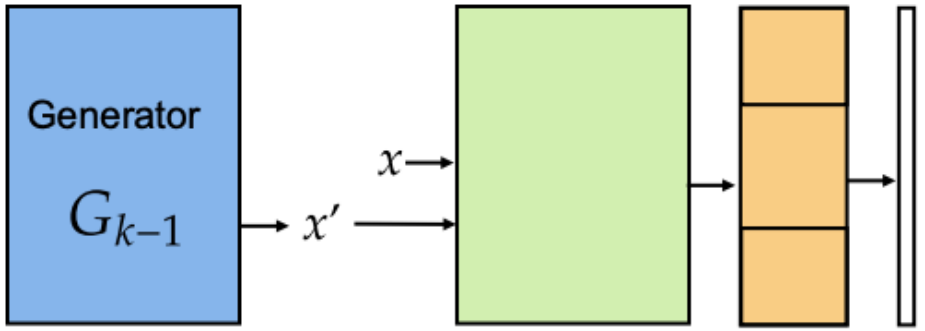
end for

z' is the logit of replay sample x' obtained via parameter sub-optimal parameter

Liyuan Wang, Xingxing Zhang, Hang Su, Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. Preprint, 2023

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In NeurIPS, 2020.

Proposed Methods - Pseudo-Replay Methods



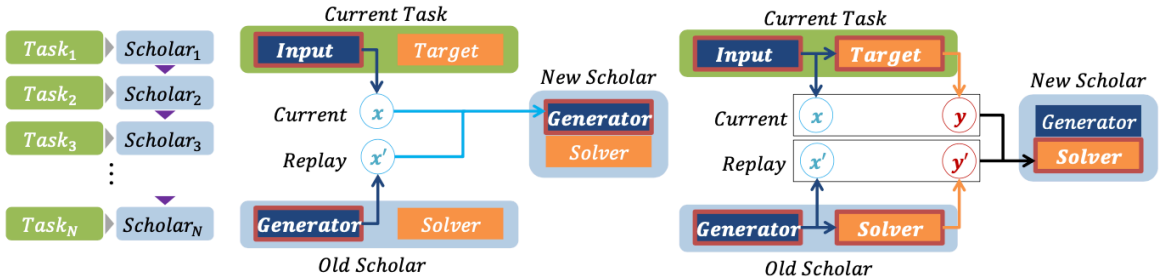
- Use a classifier and a generator
- The generator replaces the memory buffer

$$\mathcal{L} = -\left(\frac{1}{|D^k|} \sum_{(x,y)\in D^k} \log p(y|x) + \frac{1}{|\mathcal{M}|} \sum_{(x,y)\in \mathcal{M}} \log p(y|x) \right) + \mathcal{R}$$

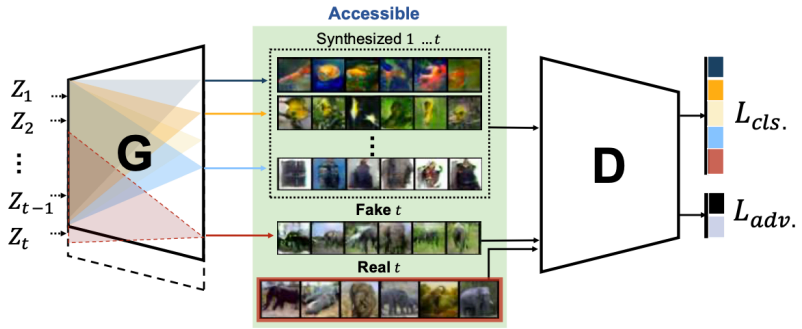
where $\mathcal{M} \sim G_{k-1}$

Proposed Methods - Pseudo-Replay Methods

- How to prevent forgetting in the generator?
  - DGR [Shin et al., 2017]

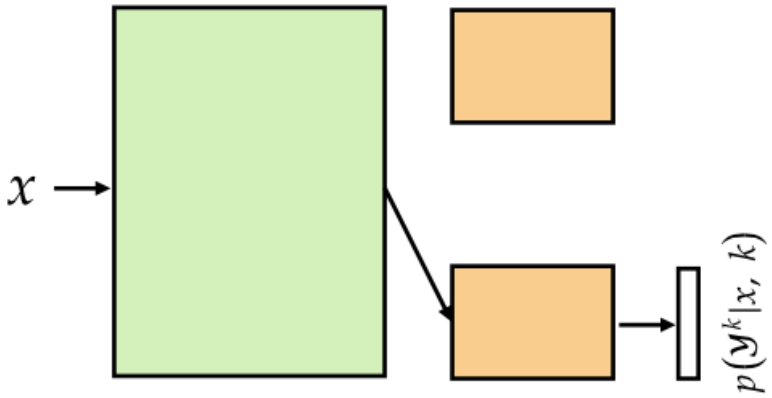


  - DGM [Ostapenko et al., 2019]. Protect via parameter-isolation method (e.g., HAT)



Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In NeurIPS, pp. 2994–3003, 2017.
Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In CVPR, 2019.

Proposed Methods - Parameter-Isolation Methods

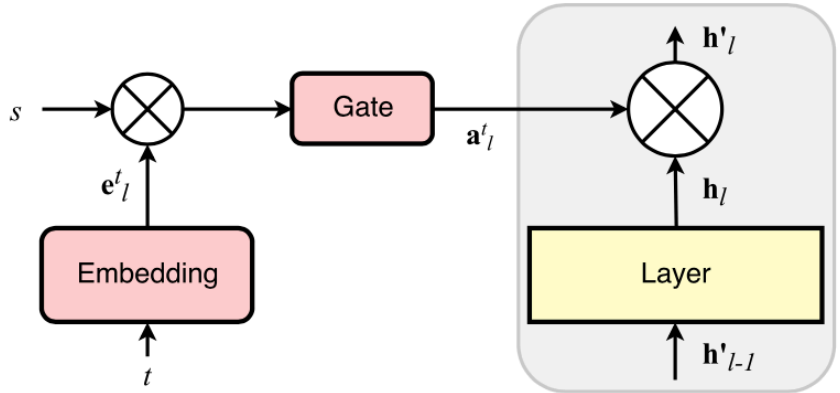


- Originally used for TIL, but can be used for CIL as well
- Use task-specific parameters
- For task k, minimize

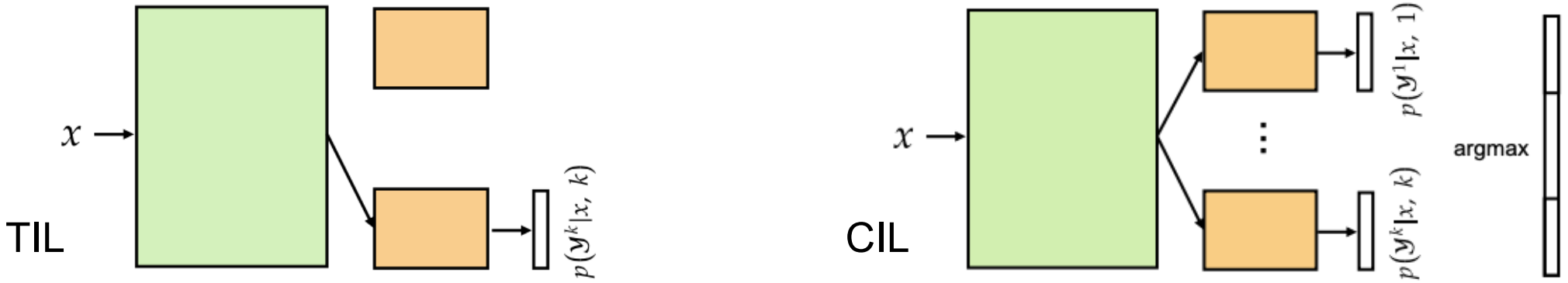
$$\mathcal{L} = -\frac{1}{|D^k|} \sum_{(x,y)\in D^k} \log p(y|x,k) + \mathcal{R}$$

Proposed Methods - Parameter-Isolation Methods

- Hard attention to the task



- CIL prediction using TIL models. Simply concatenate the outputs



Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In ICML, 2018.
Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, Bing Liu. A Theoretical Study on Solving Continual Learning. In NeurIPS, 2022.

Evaluation

- Evaluation metrics
  - Average accuracy

$$\mathcal{A}_k = \sum_{j=1}^{k} a_{k,j}/k$$

  - Average incremental accuracy

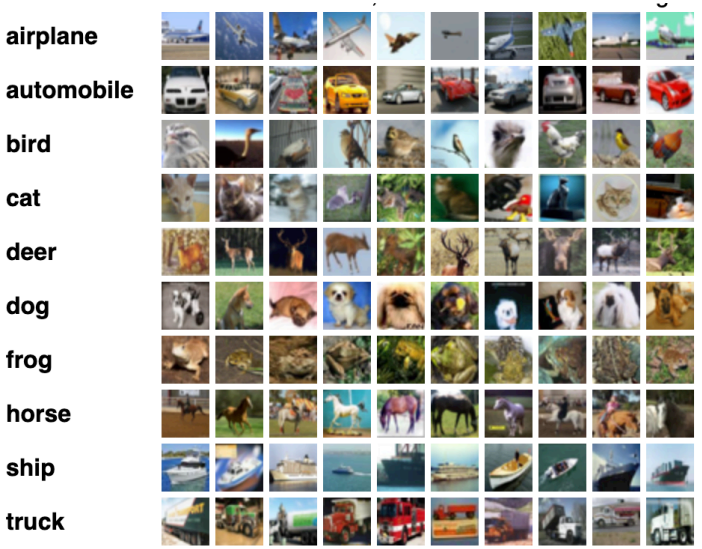
$$\mathcal{A} = \sum_{k=1}^{t} \mathcal{A}_k/t$$

  - Forgetting rate (i.e., backward-transfer)

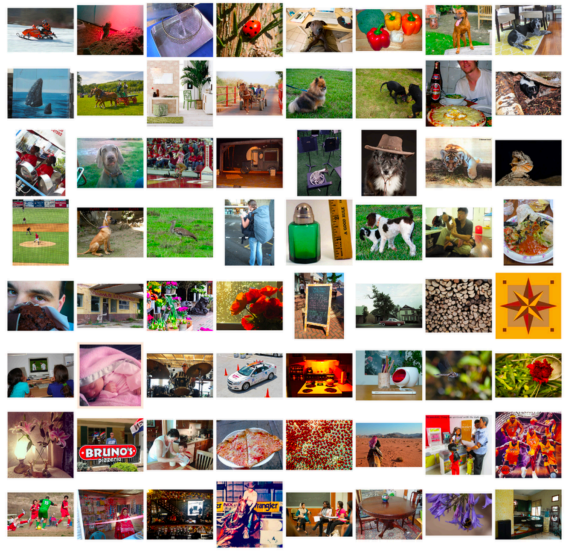
$$\mathcal{F}_t = \sum_{k=1}^{t-1} (a_{k,k} - a_{t,k})/(t-1)$$

Evaluation

- Popular datasets
  - MNIST
  - CIFAR-10
  - CIFAR-100
  - Tiny-ImageNet



CIFAR-10



ImageNet

A Theoretical Guidance on Solving CIL

- CIL is difficult because of inter-task class separation
  - The system needs to establish decision boundaries between the classes of the new task and the classes of the previous tasks without previous task data

- At inference, if the system knows which task the test instance belongs to, CIL problem is reduced to TIL

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, Bing Liu. A Theoretical Study on Solving Continual Learning. In NeurIPS, 2022.

A Theoretical Guidance on Solving CIL

- Out-of-Distribution (OOD) detection
  - Assign a class if an instance belongs to one of the classes used in the training data (IND)
  - Reject if an instance does not belong to any of the n IND training classes
- For task 1, OOD classes are

- For task 2, OOD classes are

A Theoretical Guidance on Solving CIL

- CIL problem can be decomposed into two subproblems: within-task prediction (WP) and task-id prediction (TP)

$$\mathbf{P}(x \in \mathbf{X}_{k_0,j_0}|D) = \sum_{k=1,\ldots,n} \mathbf{P}(x \in \mathbf{X}_{k,j_0}|x \in \mathbf{X}_k, D)\mathbf{P}(x \in \mathbf{X}_k|D)$$
$$= \mathbf{P}(x \in \mathbf{X}_{k_0,j_0}|x \in \mathbf{X}_{k_0}, D)\mathbf{P}(x \in \mathbf{X}_{k_0}|D)$$

WP (i.e., TIL) TP

- The loss of CIL is bounded by that of WP (TIL) and OOD (TP)

Theorem 3. *If* $H_{OOD,k}(x) \leq \delta_k,\ k = 1, \ldots, T$ *and* $H_{WP}(x) \leq \epsilon$, *we have*

$$H_{CIL}(x) \leq \epsilon + (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k}),$$

where $\mathbf{1}_{x \in \mathbf{X}_k}$ *is an indicator function.*

- A good performance of WP (TIL) or TP (or OOD) are necessary and sufficient for a good CIL

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, Bing Liu. A Theoretical Study on Solving Continual Learning. In NeurIPS, 2022.

Evaluation

- Average accuracy

Method	M-5T	C10-5T	C100-10T	C100-20T	T-5T	T-10T
OWM	95.8±0.13	51.8±0.05	28.9±0.60	24.1±0.26	10.0±0.55	8.6±0.42
MUC	74.9±0.46	52.9±1.03	30.4±1.18	14.2±0.30	33.6±0.19	17.4±0.17
PASS†	76.6±1.67	47.3±0.98	33.0±0.58	25.0±0.69	28.4±0.51	19.1±0.46
LwF	85.5±3.11	54.7±1.18	45.3±0.75	44.3±0.46	32.2±0.50	24.3±0.26
iCaRL*	96.0±0.43	63.4±1.11	51.4±0.99	47.8±0.48	37.0±0.41	28.3±0.18
Mnemonics†*	96.3±0.36	64.1±1.47	51.0±0.34	47.6±0.74	37.1±0.46	28.5±0.72
BiC	94.1±0.65	61.4±1.74	52.9±0.64	48.9±0.54	41.7±0.74	33.8±0.40
DER++	95.3±0.69	66.0±1.20	53.7±1.30	46.6±1.44	35.8±0.77	30.5±0.47
Co^{2}L		65.6				
CCG	97.3	70.1				
HAT	81.9±3.74	62.7±1.45	41.1±0.93	25.6±0.51	38.5±1.85	29.8±0.65
HyperNet	56.6±4.85	53.4±2.19	30.2±1.54	18.7±1.10	7.9±0.69	5.3±0.50
Sup	70.1±1.51	62.4±1.45	44.6±0.44	34.7±0.30	41.8±1.50	36.5±0.36
PR-Ent	74.1	61.9	45.2			
HAT+CSI	94.4±0.26	87.8±0.71	63.3±1.00	54.6±0.92	45.7±0.26	47.1±0.18
Sup+CSI	80.7±2.71	86.0±0.41	65.1±0.39	60.2±0.51	48.9±0.25	45.7±0.76
HAT+CSI+c	96.9±0.30	88.0±0.48	65.2±0.71	58.0±0.45	51.7±0.37	47.6±0.32
Sup+CSI+c	81.0±2.30	87.3±0.37	65.2±0.37	60.5±0.64	49.2±0.28	46.2±0.53

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, Bing Liu. A Theoretical Study on Solving Continual Learning. In NeurIPS, 2022.

CIL Using Pre-Trained Models

- Strong pre-trained models have been proposed, but most CIL methods are not leveraging the strong pre-trained models (except the ones in NLP)
- Using pre-trained models, the CIL performances are much better
- Need different methods when using pre-trained models
- Make sure that CL datasets are different from the pre-training dataset to ensure there is no information leakage from pre-training to CL training
- Fine-tune the entire network, adapters, or prompts

CIL Using Pre-Trained Models

- Make sure that CL datasets are different from the pre-training dataset to ensure there is no information leakage from pre-training to CL training
  - We use CIFAR-10, CIFAR-100, and Tiny-ImageNet for CL experiment
  - We removed 389 classes from the original 1000 classes in ImageNet that are similar/identical to the classes in CIFAR-10, CIFAR-100, or Tiny-ImageNet.
  - We pre-train a network with the remaining subset of 611 classes of ImageNet

- Failing to remove such classes seriously affects the results
  - The accuracy of L2P [Wang et al., 2022] method on C100-10T drops from 84 to 62

Gyuhak Kim, Zixuan Ke, Bing Liu. A Multi-Head Model for Continual Learning via Out-of-Distribution Replay. In CoLLAs, 2022.
Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, and Others. Learning to prompt for continual learning. In CVPR, 2022

Future Directions

- A lot of CIL techniques are proposed, but they are mostly empirical. Need more theoretical understanding. We made an attempt in (Kim et al., NeurIPS-2022)
- Need more rigorous study on using pre-trained models in CL
- Can we do backward transfer for parameter-isolation methods?

Domain Incremental Learning and Continual Learning in NLP

Zixuan Ke

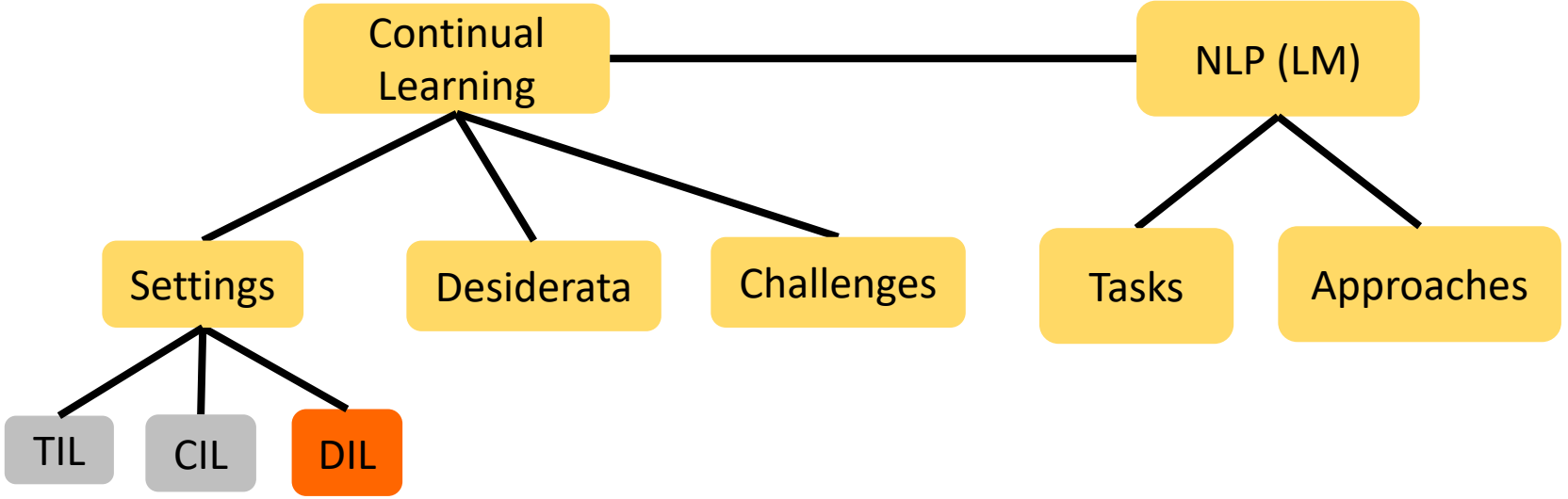
University of Illinois Chicago

https://vincent950129.github.io/

Plan

- A quick summary of TIL and CIL
- Another setting: Domain-incremental Learning
- What we have learned so far
- Continual learning of NLP Tasks
- Conclusion and Future work

Roadmap



*LM: Language Model

Settings

- We have known that

TIL

- **Assumption:** task-IDs are available in both training and testing

$$\mathcal{Y}_t \subseteq \mathcal{Y}$$

- **Goals:**

$$f : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{Y}$$

CIL

- **Assumption:** task-IDs are available only in training. In testing, a test instance from any class may be presented

$$\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \varnothing, \forall\, t \neq t'$$

$$\mathcal{Y} = \bigcup_{t=1}^{T} \mathcal{Y}_t$$

- **Goals:** $f : \mathcal{X} \rightarrow \mathcal{Y}$

Notations:

- T is the last task learned and $\mathcal{T} = \{1...T\}$
- $\mathcal{Y}_t$ is the label space of task t
- For any given tasks, t and t',

$$p(\mathcal{X}_t) \neq p(\mathcal{X}_{t'}), \forall\, t \neq t'$$

Plan

- A quick summary of TIL and CIL
- Another setting: Domain-incremental Learning
- What we have learned so far
- Continual learning of NLP Tasks
- Conclusion and Future work

Domain-incremental Learning

- Assumption
  - Class labels are assumed to be the same for all tasks
- E.g.,
  - A sequence of **sentiment classification tasks** (all tasks have the same labels {*positive, negative, neutral*})
  - **Generation tasks** in NLP (all tasks depend on the language model head which has the same number of vocabulary tokens)

Domain-incremental Learning

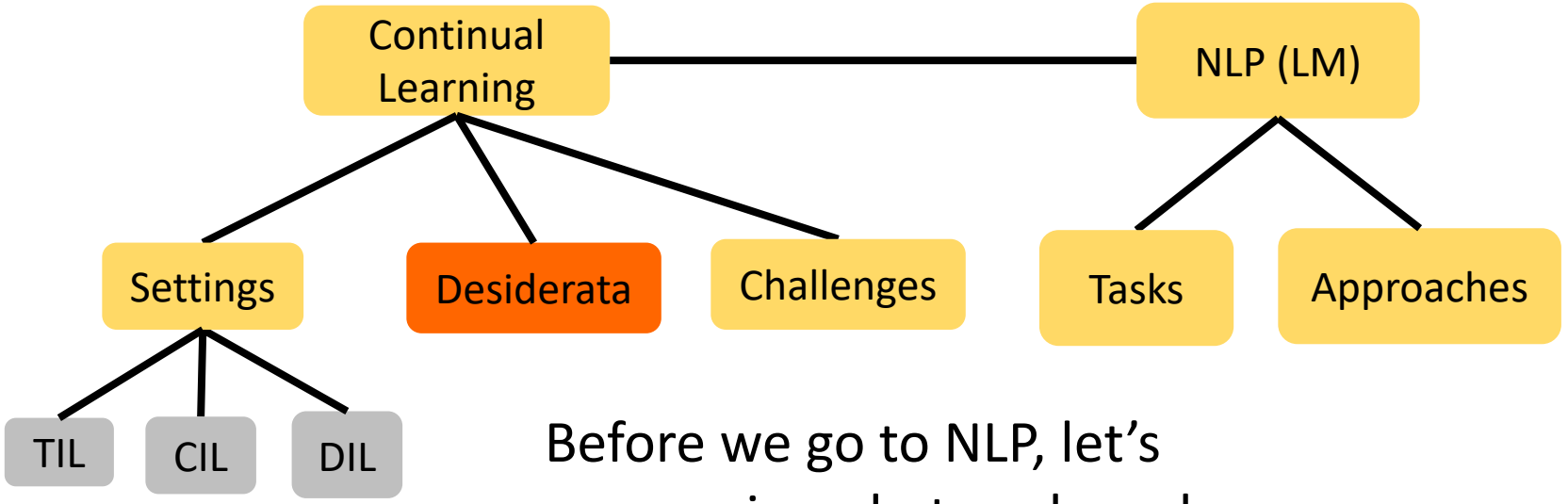
- Goal
  - Learn a function $f : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{Y}$

- Yes, it is **a special case of TIL**
  - A DIL problem can always be solved with a TIL method
- However, you may see some existing DIL systems do not use task-ID in testing,
  - This is because the tasks are very similar (task-ID does not matter) or very dissimilar (task-ID is easily predicted using the test data)

Domain-incremental Learning

- Now we know all the 3 settings, why are they important?
  - Together with TIL and CIL, they constitute **the three fundamental settings** of continual learning.
  - When attempting to leverage continual learning, **the initial step** is to establish the appropriate setting.
  - Different settings are for different applications and lead to **different specific challenges and approaches**

Roadmap



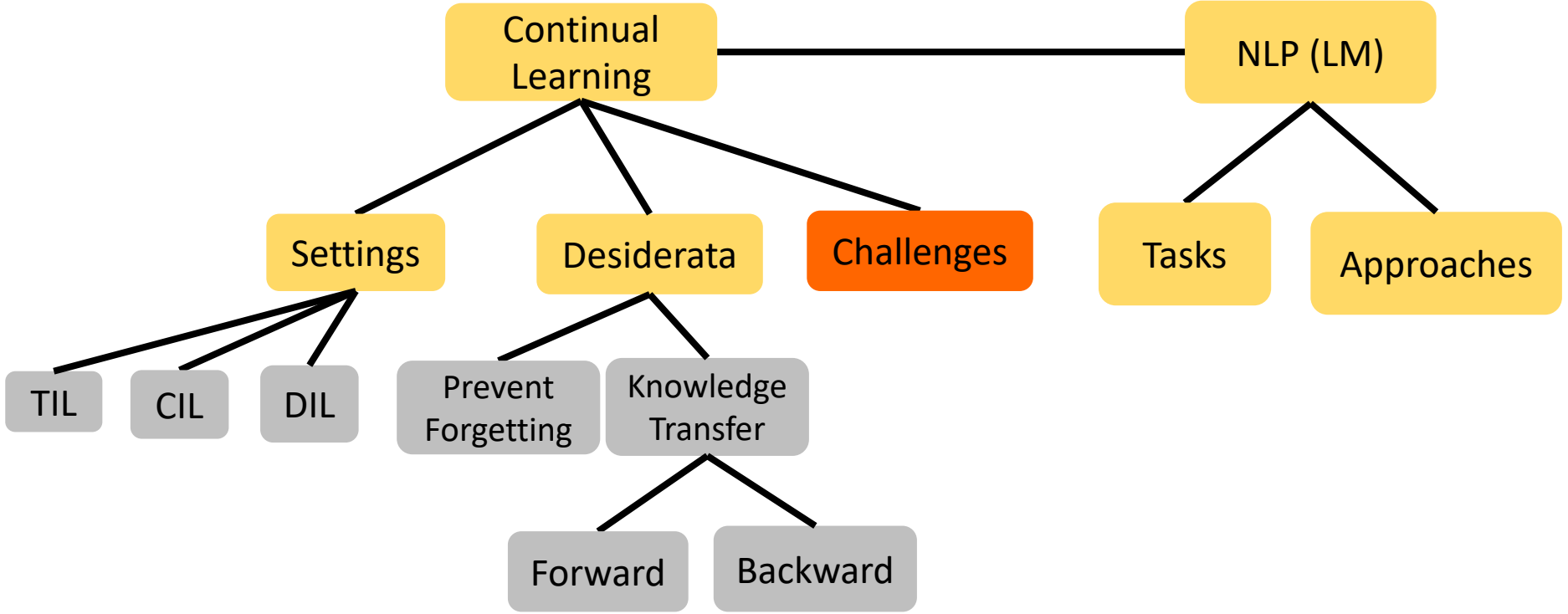
Before we go to NLP, let's summarize what we have known about continual learning **so far**

*LM: Language Model

Continual learning

- General Desiderata (TIL, DIL, CIL)
  - Not suffer from **catastrophic forgetting** (CF)
    - i.e., perform reasonably well on what has been learned
  - Achieve **positive forward knowledge transfer** (forward KT)
    - i.e., old knowledge helps new task
  - Achieve **positive backward knowledge transfer** (backward KT)
    - i.e., relevant new task helps old tasks

Roadmap



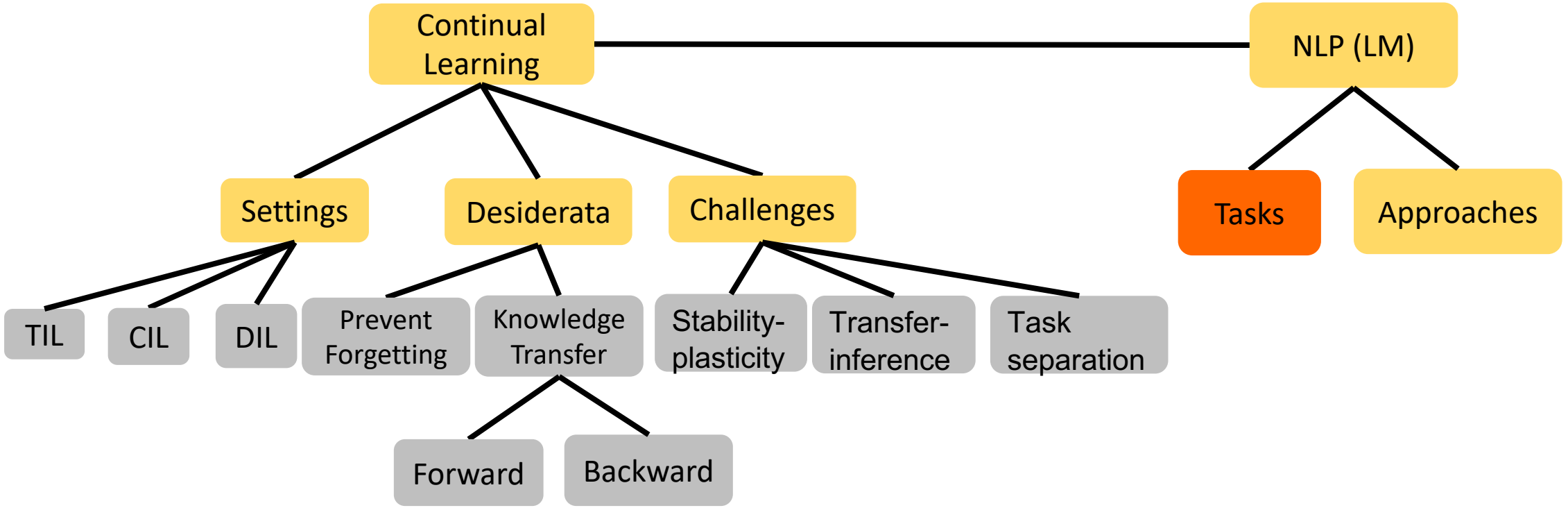
*LM: Language Model

Continual learning

- Challenges
  - **Stability-plasticity** (TIL, CIL, DIL)
    - Preserving the learned knowledge v.s. accurately learning new experiences
  - **Transfer-interference** (mostly in TIL and DIL)
    - Knowledge transfer vs. knowledge interference
    - (increase parameter-sharing vs. reduce parameter-sharing)
  - **Task separation** (mostly in CIL, and DIL w/o ID)
    - Establish decision boundaries between the current task and previous tasks
      - In learning the current task, the learner does not see previous or future task data

*Task separation a.k.a. inter-task class separation in CIL

Roadmap



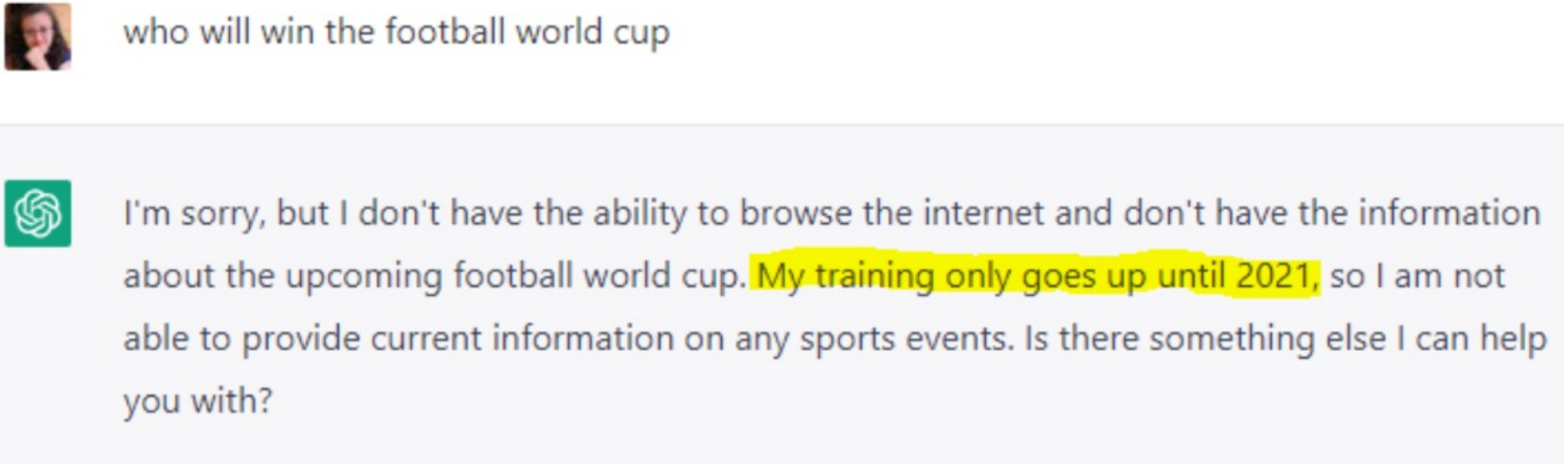
*LM: Language Model

Plan

- A quick summary of TIL and CIL
- Another setting: Domain-incremental Learning
- What we have learned so far
- Continual learning of NLP Tasks
- Conclusion and Future work

Continual Learning in NLP

- **The world is dynamic**, and time is irreversible, new emerging *{classes/domains/tasks/topics/events}* keep coming



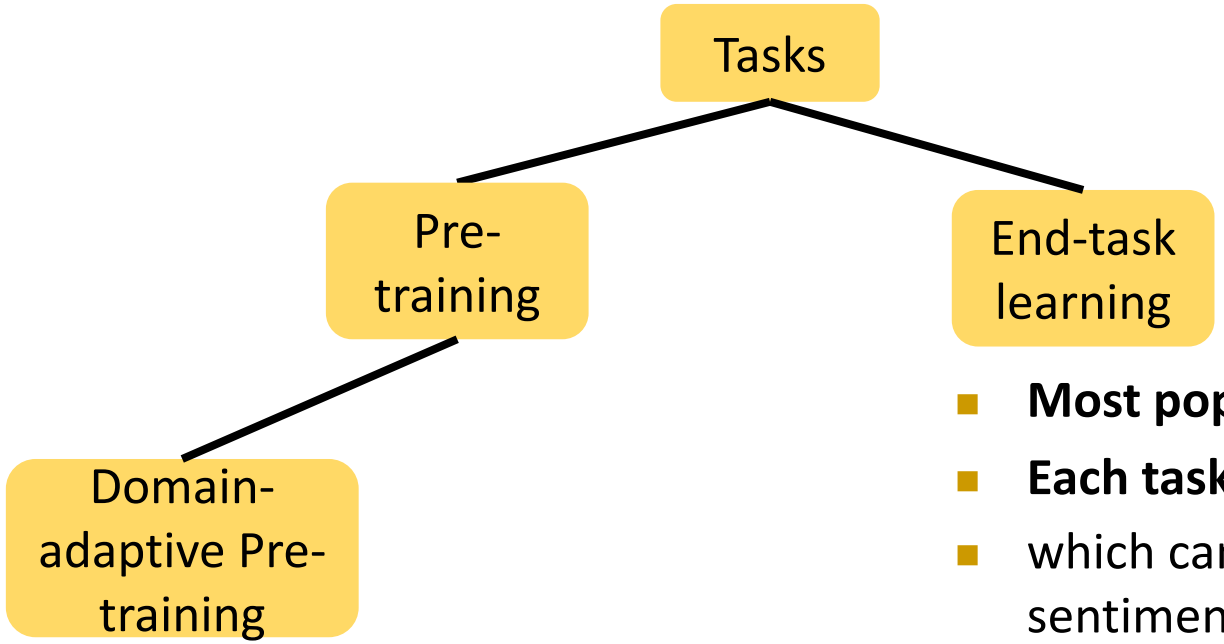
Continual Learning in NLP

- **The world is dynamic**, and time is irreversible, new emerging *{classes/domains/tasks}* keep coming
- We do not want to **re-train** for many reasons
- We do not want to train **separate** models for many reasons
- We want the model **accumulates knowledge** in its lifetime without forgetting what it has learned

Continual Learning in NLP

- "Task"
  - In terms of NLP, what is a task?
- Formulation
  - How are some NLP tasks learned continually?
- Approaches
  - What are some popular approaches in CL for NLP?

Continual Learning in NLP



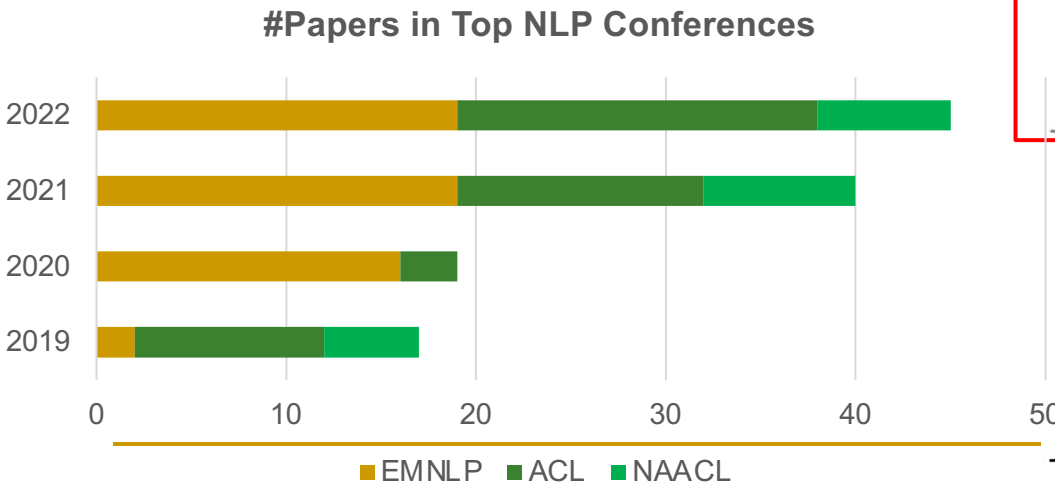
- **Each task uses a domain corpus** to adapt an LM to a domain/topic

- **Most popular**
- **Each task is an end-task**
- which can be any NLP problem (e.g., sentiment classification, NER, summarization)

*Domain-adaptive pre-training (DAPT) a.k.a. post-training or pre-finetuning

Formulation of NLP Problems

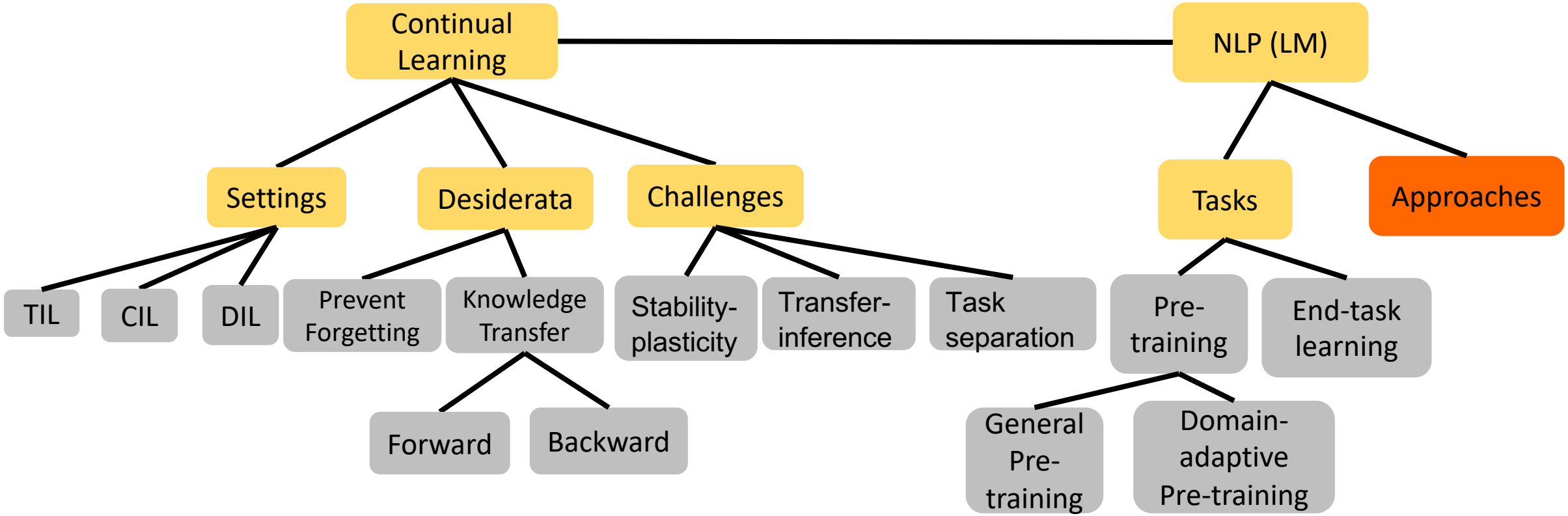
- Large #NLP problems belong to **DIL**
  - Many problems are converted to generation tasks in NLP via prompting
- Continual learning for NLP has been growing **rapidly**, this table keeps expanding.



CL Settings	Type	NLP Problems	CL Papers
TIL	End-task	Aspect sentiment classification	B-CL (Ke et al., 2021c)
			CTR (Ke et al., 2021a)
		Intent classification	MeLL (Wang et al., 2021a)
			PCLL (Zhao et al., 2022)
		Slot filling	PCLL (Zhao et al., 2022)
		Topic classification	CTR (Ke et al., 2021a)
		Mixed diverse tasks	CLIF (Jin et al., 2021)
DIL (w/o ID)	End-task	Mixed 5 classification tasks	LFPT5 (Qin and Joty, 2022)
		Named-entity recognition	LFPT5 (Qin and Joty, 2022)
		Summerization	LFPT5 (Qin and Joty, 2022)
		Paraphrase	RMR-DSE (Li et al., 2022a)
		Dialogue response generation	RMR-DSE (Li et al., 2022a)
			AdapterCL (Madotto et al., 2020)
		Dialogue state tracking	AdapterCL (Madotto et al., 2020)
		Dialogue end2end	AdapterCL (Madotto et al., 2020)
		Aspect sentiment classification	CLASSIC (Ke et al., 2021b)
		Question answering	MBPA++ (de Masson d'Autume et al., 2019)
			Meta-MBPA++ (Wang et al., 2020)
	DAPT	5 pre-training domains	ELLE (Qin et al., 2022)
		8 pre-training domains	DEMIX (Gururangan et al., 2021)
		10 pre-training domains	Continual-T0 (Scialom et al., 2022)
DIL (w/ ID)	End-task	Mixed 5 classification tasks	LAMOL (Sun et al., 2020)
		Mixed classification and labeling tasks	LAMOL (Sun et al., 2020)
		Dialogue state tracking	C-PT (Zhu et al., 2022)
		Dialogue response generation	TPEM (Geng et al., 2021)
		Mixed classification and generation	ConTinTin (Yin et al., 2022)
		Mixed 4 generation tasks	ACM (Zhang et al., 2022)
	DAPT	5 pre-training domains	CPT (Ke et al., 2022)
CIL	End-task	Named-entity recognition	ExtendNER (Monaikul et al., 2021)
		Intent classification	CID (Liu et al., 2021)
			PAGeR (Varshney et al., 2022)
		Mixed 5 classification tasks	IDBR (Huang et al., 2021)
		Slot filling	ProgM (Shen et al., 2019)
		Sentence representation	SRC (Liu et al., 2019a)
		Mixed 5 classification tasks	MBPA++ (de Masson d'Autume et al., 2019)
			Meta-MBPA++ (Wang et al., 2020)
	Few-shot	Intent classification	ENTAILMENT (Xia et al., 2021)
			CFID (Li et al., 2022b)

*DAPT: Domain-adaptive pre-training

Roadmap



*LM: Language Model

Approaches for Forgetting Prevention

- We have known the three popular families in continual learning to **prevent forgetting**

- **Add penalty** when training new task

Regularization-based

- **Memorize a subset** of old samples and train together with the new task

Replay-based

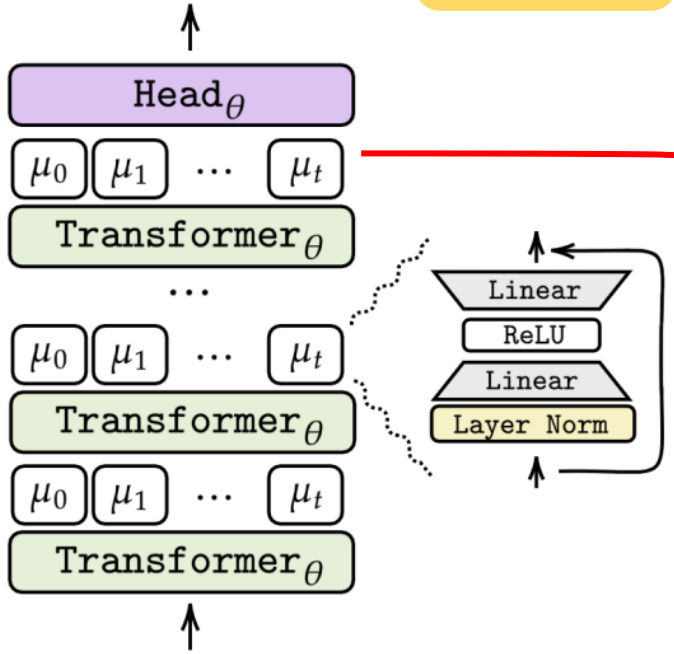
- **Allocate specific parameters** to specific tasks

Parameter-isolation

Approaches for Forgetting Prevention

- A simple example in NLP: Adapter-CL

DIL | End-task learning | Parameter -isolation



μ_t refers to the adapter for task t

Adapter-CL **uses separate adapters for different tasks,** which is a naïve way to prevent forgetting.

It can do well in **forgetting prevention**, but cannot achieve the other desideratum (**knowledge transfer**)

***adapter**: Adding a small number of parameters to the pre-trained LM. In fine-tuning, only the adapters are trainable.

Madotto et al., Continual learning in task-oriented dialogue systems, EMNLP 2020

Approaches for Knowledge Transfer

- Approaches for preventing forgetting mainly try to **reduce parameter-sharing**
- This is not enough for **knowledge transfer**, which needs to allow some used parameters to be updated or shared.

- Compute task similarity and share the parameters for similar tasks — Similarity-based[1]
- Learn the transferrable knowledge using replay data (e.g., meta-learning) — Replay-based[2]
- Soft-masking the backward pass, but keep the forward pass untouched — Soft-mask-based

*Same name, but with different goals

[1]: Ke et al., Achieving forgetting prevention and knowledge transfer in continual learning, NeuIPS 2021

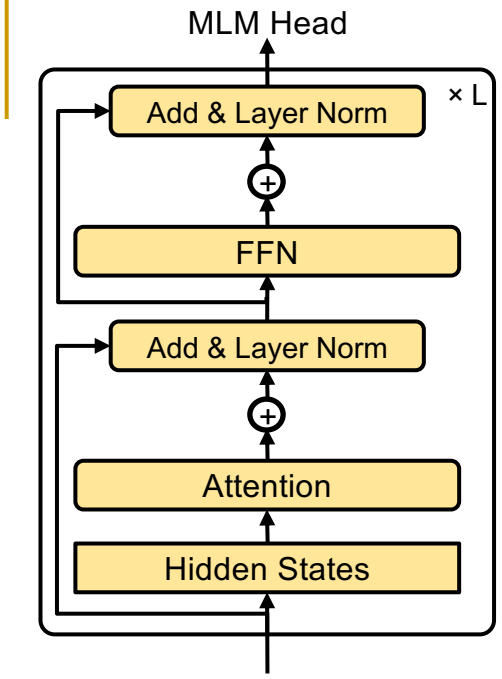
[2]: d'Autume et al., Episodic Memory in Lifelong Language Learning, NeurIPS 2019

Continual Domain-adaptive Pre-training

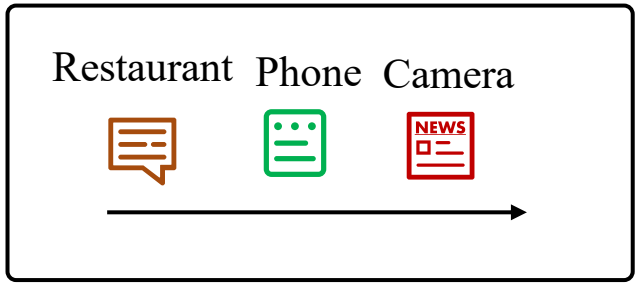
DIL | Domain-adaptive Pre-training | Soft-mask-based

- Setting
  - Continual domain-adaptive pre-training of a sequence of domains **without** accessing the data that was used in original **pre-training or previously learned domains**
  - End-task doesn't know its domain belonging
- Goals
  - CF prevention
  - KT (backward and forward)
- Approach
  - **DAS**

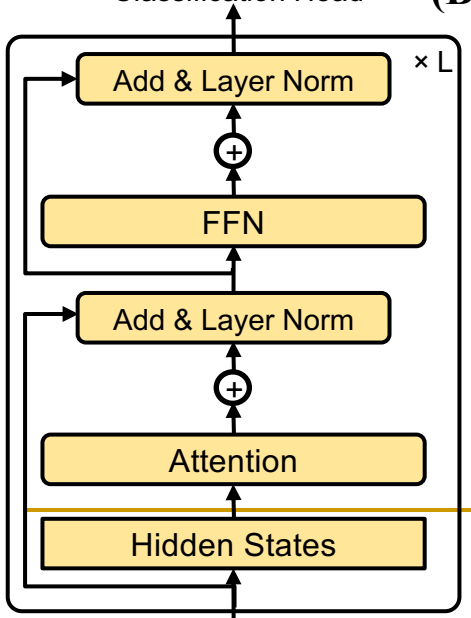
Ke et al., Continual learning of language models, ICLR 2023



(A) Continual Domain-adaptive pre-training



Given a pre-trained LM, continually domain-adaptive pre-training **a sequence of domains**



(B) Individual Fine-tuning

End-tasks

ASC-Restaurant

ASC-Phone

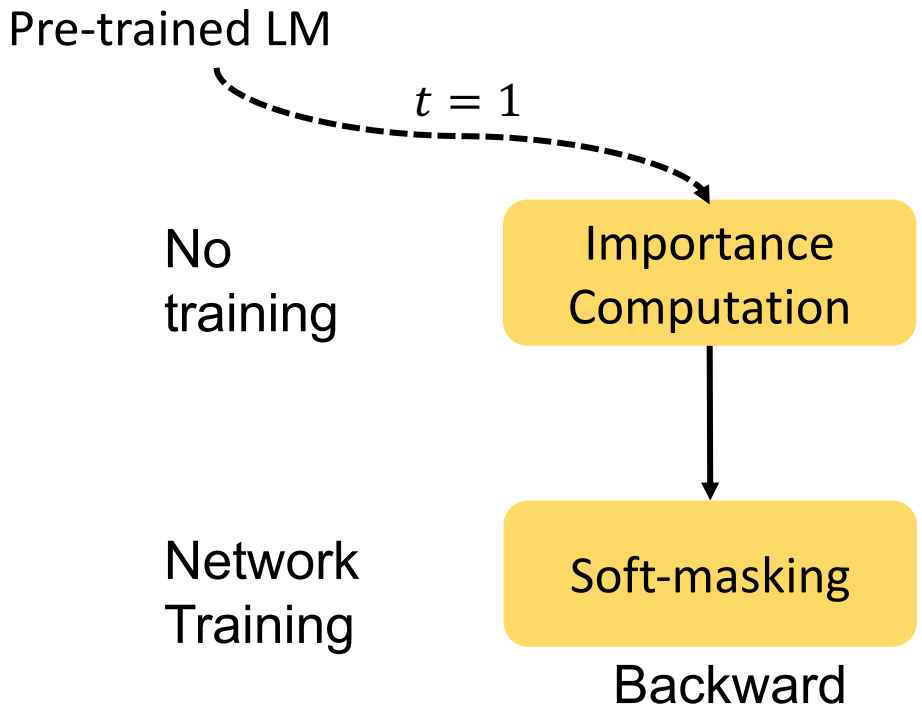
ASC-Camera

After continual learning, the domain-adaptive pre-training performance is **evaluated** by end-tasks

Each end-task **corresponding** to one domain and has its **own** training and testing set. It is trained individually and **will not** affect the domain-adaptive pre-training

ASC: Aspect Sentiment Classification

Continual Domain-adaptive Pre-training



Key ideas:

1) Detect importance of units for general and domain knowledge

2) Soft-mask the important units when training new tasks/domains

3) This can prevent forgetting and allow knowledge transfer

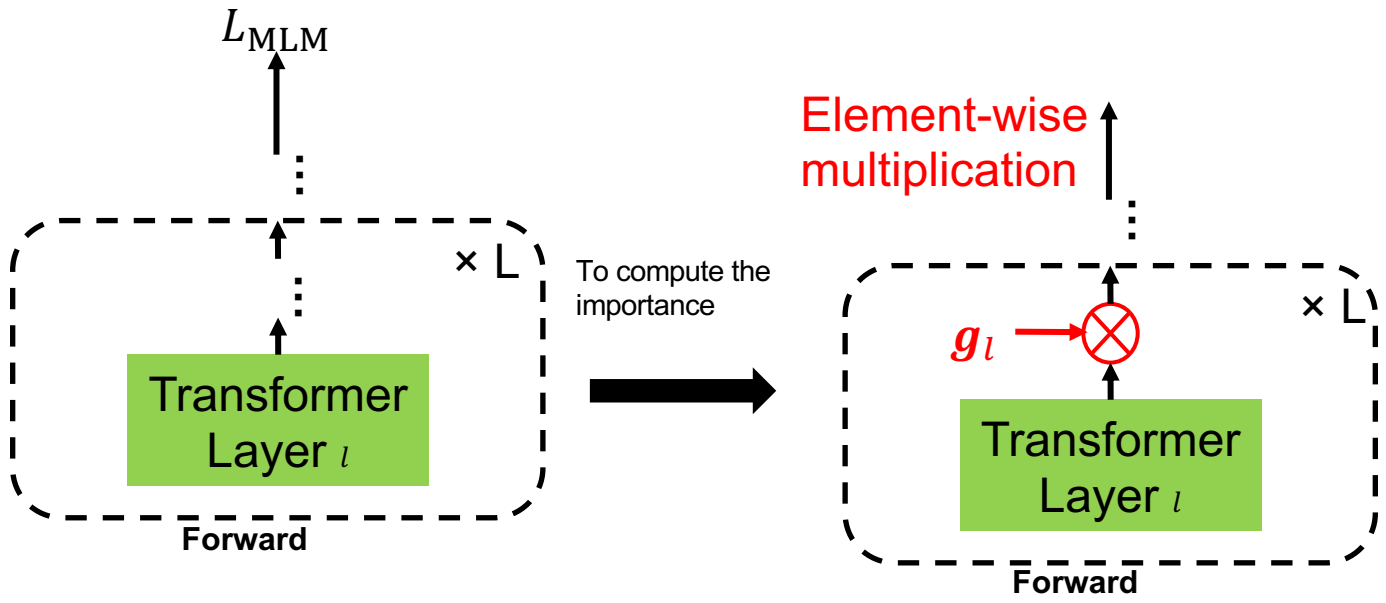
Key challenges:

1) How to detect importance for the two types of knowledge

2) How to soft-mask

Ke et al., Continual learning of language models, ICLR 2023

Importance Computation

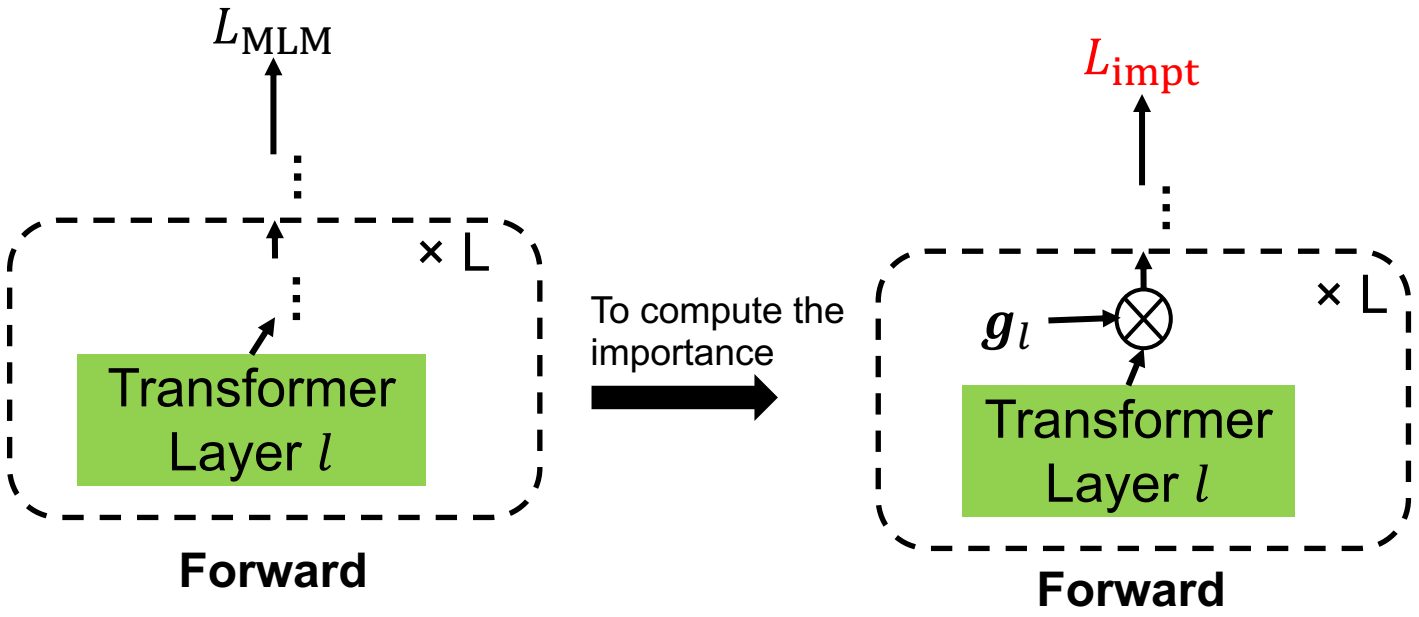


$\boldsymbol{g}_l$ is the **virtual parameters.** Each virtual parameter $g_{l,i}$ in $\boldsymbol{g}_l$ corresponding to an attention head or neurons (units)

It is **initialized as all 1's**, and has its gradient but will **never change**.

Why? We only use its gradient to compute importance

Importance Computation



For **domain knowledge,**

$L_{\text{impt}} = L_{\text{MLM}}$

$$\nabla_{g_l}^m = \frac{\partial L_{\text{impt}}(x_m^{(t)}, y_m^{(t)})}{\partial_{g_l}}$$

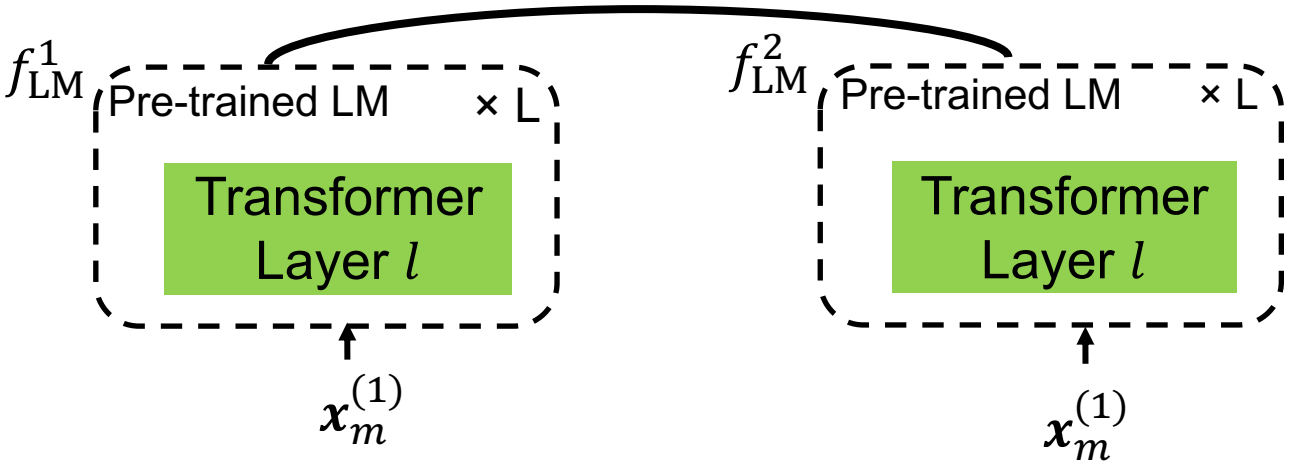
$$I_l^{(t)} = \frac{1}{M}\sum_M |\nabla_{g_l}^m|$$

Use **absolute gradient** to indicate importance[1]

[1] Michel et al. Are sixteen heads really better than one? NeurIPS, 2019.

Importance Computation

$L_{\text{impt}} = \text{KL}(f^1_{\text{LM}}(\boldsymbol{x}_m^{(1)}), f^2_{\text{LM}}(\boldsymbol{x}_m^{(1)}))$



KL: How different are the two representations?

$\boldsymbol{f^1_{LM}}$/ $\boldsymbol{f^2_{LM}}$: Transformer with different dropouts

$\boldsymbol{x}_m^{(1)}$: We only use **the first domain** data because we want to keep the pre-trained general knowledge

With the new L_{impt}, we can use the absolute gradient to indicate the importance (same as in domain knowledge)

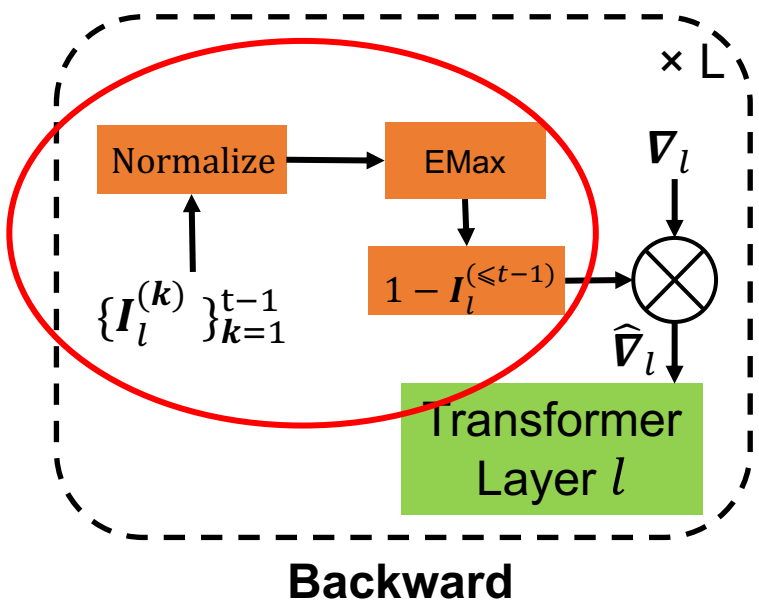
For **general knowledge**, we leverage the **random dropout** in standard Transformer

Random dropout introduces **random noise**. Given the **same input**, the difference between the representations with different random noise indicates the **robustness**.

The units that are important to the robustness is likely to be important to the **general/pre-trained knowledge** because its change will **cause the pre-trained LM** change a great deal

Ke et al., Continual learning of language models, ICLR 2023

Soft-masking



First, we normalize the importance so that they are comparable

$$I_l^{(k)} = \mathrm{Tanh}(\mathrm{Norm}(I_l^{(k)}))$$

Second, we accumulate the importance

$$I_l^{(\leqslant t-1)} = \mathrm{EMax}(\{I_l^{(t-1)}, I_l^{(t-2)}))$$

Third, we soft-mask the gradient (only in backward pass)

$$\nabla'_l = \left(1 - I_l^{(\leqslant t-1)}\right) \otimes \nabla_l$$

Ke et al., Continual learning of language models, ICLR 2023

Results

Overall end-task performance (final performance)

No DAPT

DAPT

NCL DAPT

SoTA DAPT

Category	Domain / Model	Restaurant MF1	Restaurant Acc	ACL MF1	ACL Acc	AI MF1	AI Acc	Phone MF1	Phone Acc	PubMed MF1	Camera MF1	Camera Acc	Average MF1	Average Acc	Forget R. MF1	Forget R. Acc
Non-CL	RoBERTa	79.81	87.00	66.11	71.26	60.98	71.85	83.75	86.08	72.38	78.82	87.03	73.64	79.27	—	
	DAPT (RoBERTa)	80.84	**87.68**	68.75	73.44	68.97	75.95	82.59	85.50	72.84	84.39	89.90	76.40	80.89	—	
	DAPT (Adapter)	80.19	87.14	68.87	72.92	60.55	71.38	82.71	85.35	71.68	83.62	89.23	74.60	79.62	—	
	DAPT (Prompt)	79.00	86.45	66.66	71.35	61.47	72.36	84.17	86.53	**73.09**	85.52	90.38	74.98	80.03	—	
CL Post-train	NCL	79.52	86.54	68.39	72.87	67.94	75.71	84.10	86.33	72.49	85.71	90.70	76.36	80.77	1.14	1.05
	NCL (Adapter)	80.13	87.05	67.39	72.30	57.71	69.87	83.32	85.86	72.07	83.70	89.71	74.05	79.48	0.15	-0.02
	DEMIX	79.99	87.12	68.46	72.73	63.35	72.86	78.07	82.42	71.73	86.59	91.12	74.70	79.66	0.74	0.36
	BCL	78.97	86.52	**70.71**	**74.58**	66.26	74.55	81.70	84.63	71.99	85.06	90.51	75.78	80.46	-0.06	-0.19
	CLASSIC	79.89	87.05	67.30	72.11	59.84	71.08	84.02	86.22	69.83	86.93	91.25	74.63	79.59	0.44	0.25
	KD	78.05	85.59	69.17	73.73	67.49	75.09	82.12	84.99	72.28	81.91	88.69	75.17	80.06	-0.07	0.01
	EWC	**80.98**	87.64	65.94	71.17	65.04	73.58	82.32	85.13	71.43	83.35	89.14	74.84	79.68	0.02	-0.01
	DER++	79.00	86.46	67.20	72.16	63.96	73.54	83.22	85.61	72.58	87.10	91.47	75.51	80.30	2.36	1.53
	HAT	76.42	85.16	60.70	68.79	47.37	65.69	72.33	79.13	69.97	74.04	85.14	66.80	75.65	-0.13	-0.29
	HAT-All	74.94	83.93	52.08	63.94	34.16	56.07	64.71	74.43	68.14	65.54	81.44	59.93	71.33	3.23	1.83
	HAT (Adapter)	79.29	86.70	68.25	72.87	64.84	73.67	81.44	84.56	71.61	82.37	89.27	74.63	79.78	-0.23	-0.18
	DAS	80.34	87.16	69.36	74.01	**70.93**	**77.46**	**85.99**	**87.70**	72.80	**88.16**	**92.30**	**77.93**	**81.91**	**-1.09**	**-0.60**

- w/o DAPT < DAPT < DAS
- +forgetting rate in NCL: it does suffer from forgetting
- Regularization-based methods (KD, EWC) and replay-based method (DER++) are all worse: focus on CF prevention is not enough
- Parameter-isolation method (HAT) preforms much worse: the full LM is needed for domain-adaptive pre-training
- Methods that tries to perform both KT and CF (DEMIX, BCL, CLASSIC): all weaker than DAS

Ke et al., Continual learning of language models, ICLR 2023

*Naïve continual learning (NCL):
continual learning without any specific technique

Approaches for Task Separation

- Besides forgetting and knowledge transfer, another challenge in CIL and DIL w/o ID is the task separation

  - Use heuristic methods like entropy/perplexity to detect task ID at test time

    Post-prediction[1,2]

  - Use replay data to establish the boundary

    Replay-based

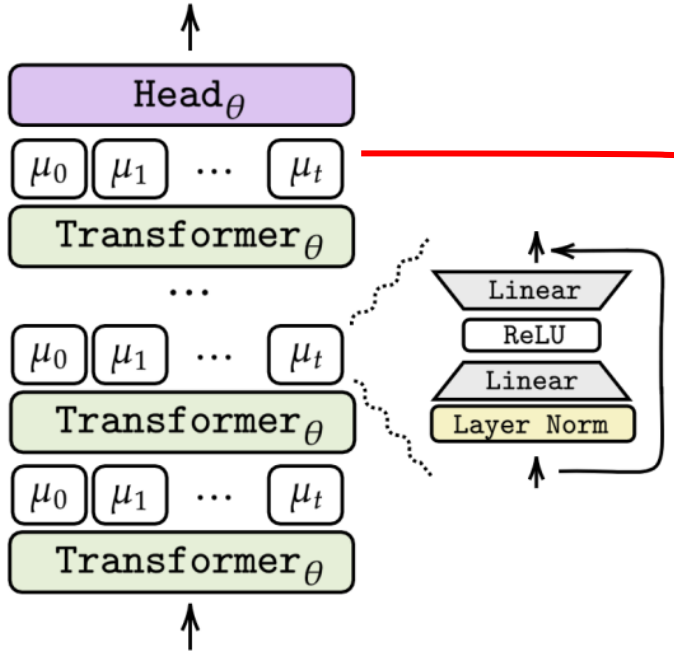
[1]: Gururangan et al., DEMix Layers: Disentangling Domains for Modular Language Modeling, NAACL 2022

[2]: Li et al., Overcoming Catastrophic Forgetting During Domain Adaptation of Seq2seq Language Generation, NAACL 2022

Approaches for Task Separation

DIL | End-task learning | Post-prediction

- A simple example in NLP: Adapter-CL



$\boldsymbol{\mu_t}$ refers to the adapter for task t

Adapter-CL uses separate adapters for different tasks, but how to know which one to use in testing?

Perplexity

$$\alpha_t = \mathrm{PPL}_{\mu_t}(X)\ \forall t \in 1, \cdots, N,$$

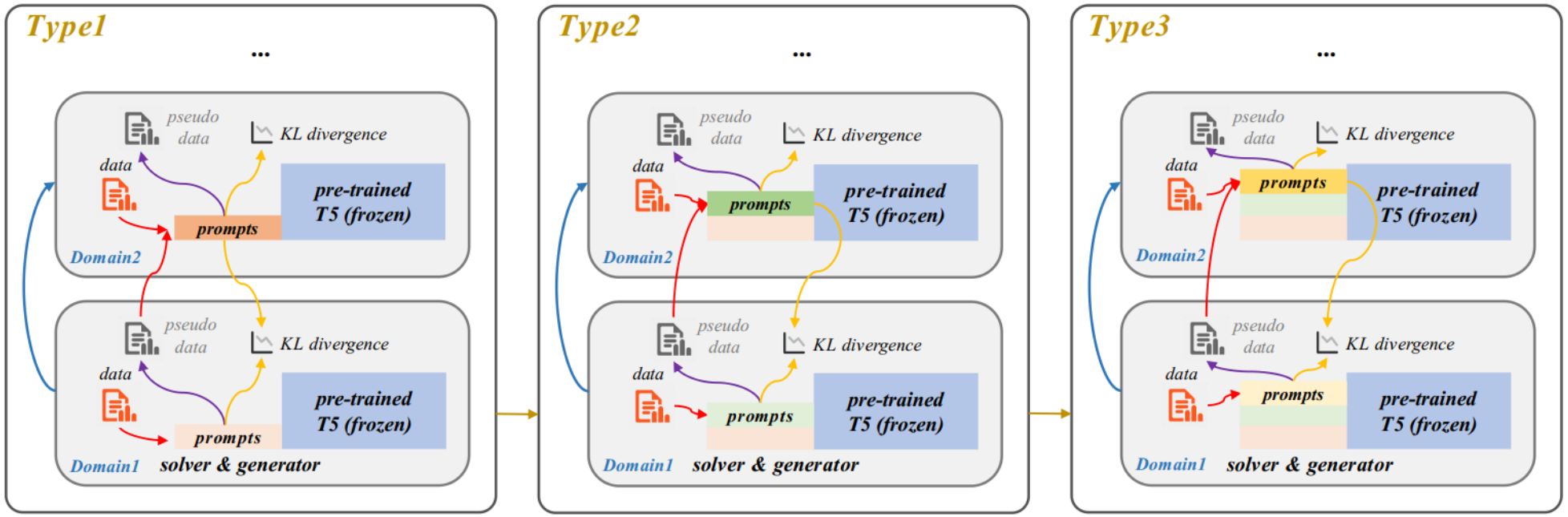
$$t^* = \mathrm{argmin}\, \alpha_0, \cdots, \alpha_N$$

Madotto et al., Continual learning in task-oriented dialogue systems, EMNLP 2020

Approaches for Task Separation

LFPT5

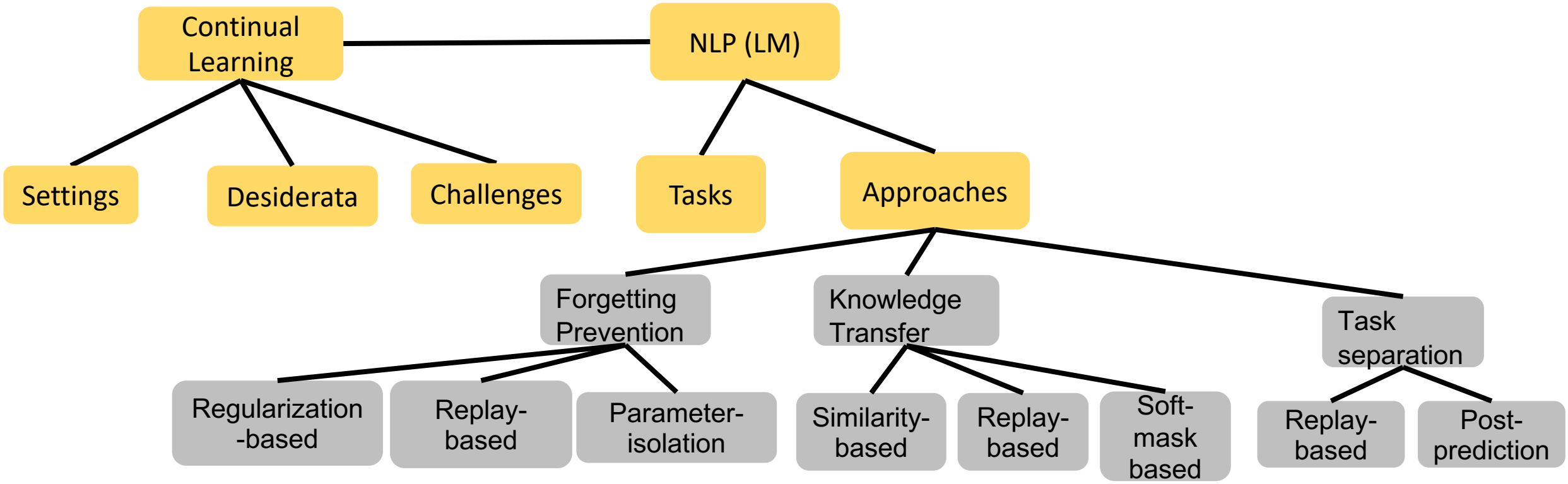
DIL | End-task learning | Replay-based



- Pre-trained LM (T5) serves as both the problem solver and generator
- When a new task comes, T5 first generates the old task data and then trains the pseudo data and new task altogether. Since some previous data is available, the decision boundary is **easier** to establish

Qin et al. A unified framework for lifelong few-shot language learning based on prompt tunning of T5. ICLR, 2022.

Roadmap



*LM: Language Model

Plan

- A quick review of we have talked
- Another setting: Domain-incremental Learning
- Continual learning of NLP Tasks
- Section summary and future work

- Continual Learning
  - Settings
    - TIL
    - CIL
    - **DIL**
  - Desiderata
    - Prevent Forgetting
    - Forward Knowledge Transfer
    - Backward Knowledge Transfer
  - Challenges
    - Stability-plasticity
    - Transfer-inference
    - Task separation
- NLP (LM)
  - Tasks
    - Pre-training
      - General Pre-training
      - **Domain-adaptive Pre-training**
    - End-task learning
      - Regularization-based
      - Replay-based
      - **Parameter-isolation**
  - Approaches
    - Forgetting Prevention
      - Similarity-based
      - Replay-based
      - **Soft-mask based**
    - Knowledge Transfer
    - Task separation
      - **Replay-based**
      - **Post-prediction**

- Knowledge transfer is the major issue in TIL and DIL
  - A task sequence can consist of a combination of similar and dissimilar tasks[1]

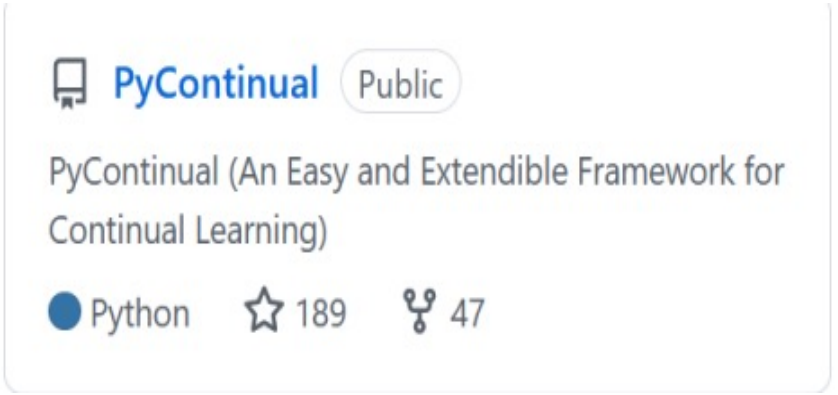
- Forgetting and task separation/discrimination are major issues for CIL and DIL w/o ID
  - Task separation has been proved to be related to OOD detection, but the accuracy is still far from the upper bound[2]

[1]: Ke et al., Continual learning of a mixed sequence of similar and dissimilar tasks, NeurIPS 2020
[2]: Kim et al., A Theoretical Study on Solving Continual Learning, NeurIPS 2022

- Continual learning of language models (LM) (e.g., domain-adaptive pre-training) is still in its infancy
  - How to better preserve the general knowledge? DAS is an initial attempt, but it is still limited.
- Scalability
  - How the network capacity issue affects the performance and how to alleviate the issue effectively are also unclear.
- Temporal continual learning
  - How to keep the LM up-to-date? Everything changes with time.

Thank you

- We have benchmarked many SoTA baselines
  - For continual end-task learning
    - https://github.com/ZixuanKe/PyContinual
  - For continual domain-adaptive pre-training
    - https://github.com/UIC-Liu-Lab/ContinualLM
- More details
  - **Survey:** Continual Learning of Natural Language Processing Tasks: A Survey *(Preprint)*
  - https://vincent950129.github.io/



Open-World Continual Learning

Bing Liu

University of Illinois Chicago

https://www.cs.uic.edu/~liub/

Closed-world and open-world

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

- Traditional machine learning:
  - Training data: D^{train} with class labels $Y^{train} = \{l_1, l_2, ..., l_t\}$.
  - Test data: D^{test}, $Y^{test} \subseteq \{l_1, l_2, ..., l_t\}$
- Closed-world: $Y^{test} \subseteq Y^{train}$
  - Classes appeared in testing must have been seen in training, **nothing new.**
- Open-world: $Y^{test} - Y^{train} \neq \phi$
  - There are unseen classes in the test data, **out-of-distribution** (OOD).
- Open-world continual learning (autonomous learning)
  - (1) detect novel/new objects (OOD detection) and
  - (2) incrementally learn the new objects (continual learning).

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

Open-world CL - a greeting bot in a hotel

(Chen and Liu, 2018; Liu et al., 2021)

Novelty detection = out-of-distribution (**OOD**) detection

- 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)**

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.

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.

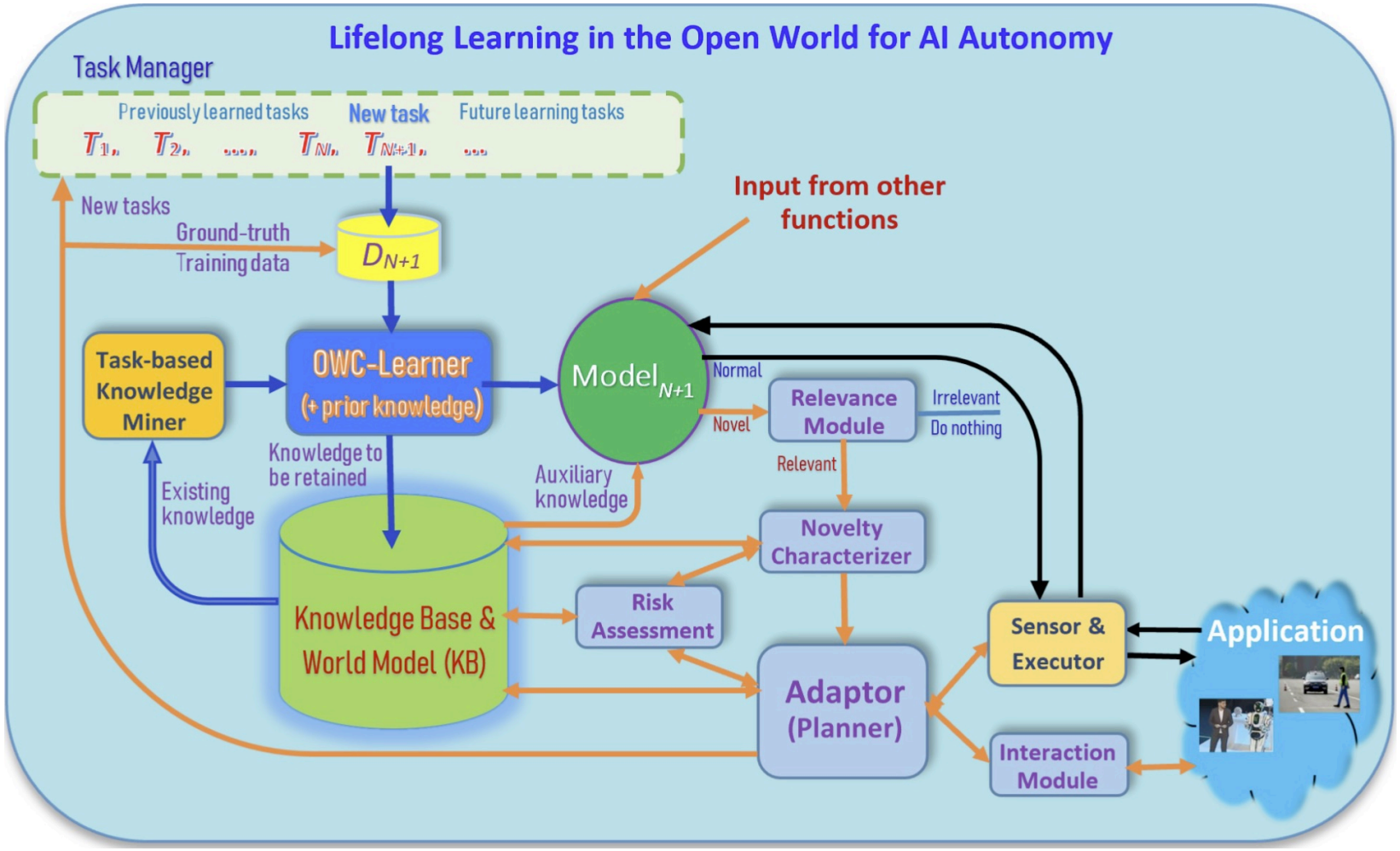
Novelty characterization and adaptation

- 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
- Risk assessment: each decision carries risks

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

SOLA: **S**elf-initiated **O**pen-world continual **L**earning & **A**daptation

(Chen & Liu, 2018, Liu, 2020 Liu et al., 2022)



Blue lines: Existing continual learning

Orange lines:

Learning after model deployment

(Open-world continual learning)

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

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

SOLA example: 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'}.

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
  - 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 used 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!

Summary

Bing Liu

University of Illinois Chicago

https://www.cs.uic.edu/~liub/

Summary

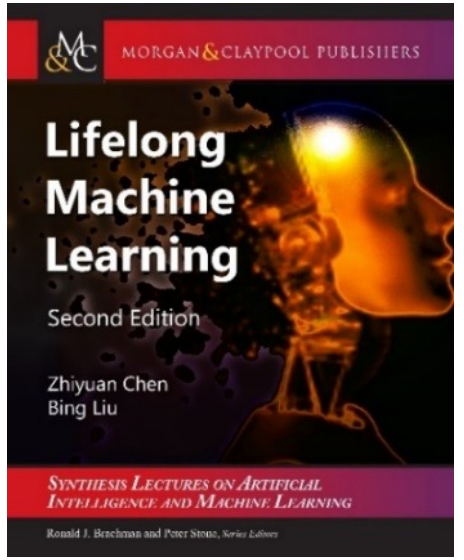
- Classical ML: isolated, closed-world, single-task learning
- This tutorial covered the following
  - Lifelong/continual learning definitions
  - Early research of lifelong learning
  - Task incremental learning (TIL)
  - Class incremental learning (CIL)
  - Domain incremental learning (DIL)
  - Continual learning in NLP
  - Continual learning in the open world

Summary: challenges

- Catastrophic forgetting
  - Still a major challenge, especially for class incremental learning (CIL)
- Knowledge transfer
  - Correctness and applicability of knowledge
  - What to transfer and how to transfer forward and backward
- Open world continual learning is a bigger challenge
  - Novelty, characterization, adaptation, and continual 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 and learning by itself

Thank You!

Q&A


MORGAN & CLAYPOOL PUBLISHERS
Lifelong Machine Learning
Second Edition
Zhiyuan Chen
Bing Liu
Synthesis Lectures on Artificial Intelligence and Machine Learning