Lifelong and Continual Learning

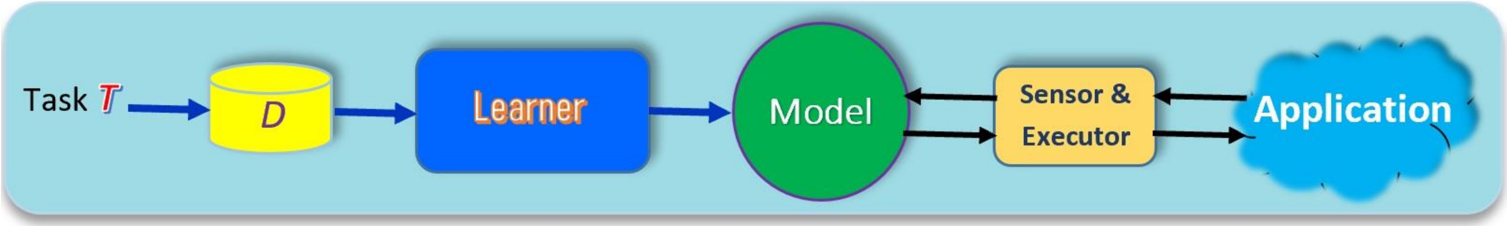
Part I – Slides for June 14, 2022

Bing Liu and **Zixuan Ke**
Department of Computer Science
University of Illinois at Chicago

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

Introduction

- Classical machine learning: Isolated single-task learning



- Existing ML algorithms such as
  - SVM, NB, DT, CRF, Deep NN
  - Have been very successful in practice
- But isolated learning has many limitations
  - Not sufficient for intelligence

Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2016, 2018
Liu and Mazumder. Lifelong and Continual Learning Dialogue Systems: Learning during Conversation. AAAI-2021

Introduction

- Key weaknesses of classical machine learning
  - **No lifelong/continuous learning**:
    - Learning each task separately in isolation
    - No knowledge accumulation or transfer (needs a large amount of training data)
  - **Closed-world assumption**:
    - Nothing new/novel in application
  - **No learning after deployment**:
    - Model fixed after deployment
- We humans learn continually and accumulate knowledge over time. We never/cannot learn in isolation

Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2016, 2018
Liu and Mazumder. Lifelong and Continual Learning Dialogue Systems: Learning during Conversation. AAAI-2021

Topics

- Lifelong or continual learning (30 mins)
- Early research on lifelong learning (90 mins)
- Continual learning using deep neural networks (240 mins)
- Continual learning in the open-world (105 mins)
- Summary and QA (15 mins)

Sub-topics

- Lifelong or continual learning
  - Definition of lifelong or continual learning
  - Three main continual learning settings
  - Related machine learning paradigms

Old 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 in its *knowledge base* (KB) to help learn the (*N*+1)th task.
- After learning (*N*+1)th task, KB is updated with learned results from the (*N*+1)th task.

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

New definition of lifelong/continual learning

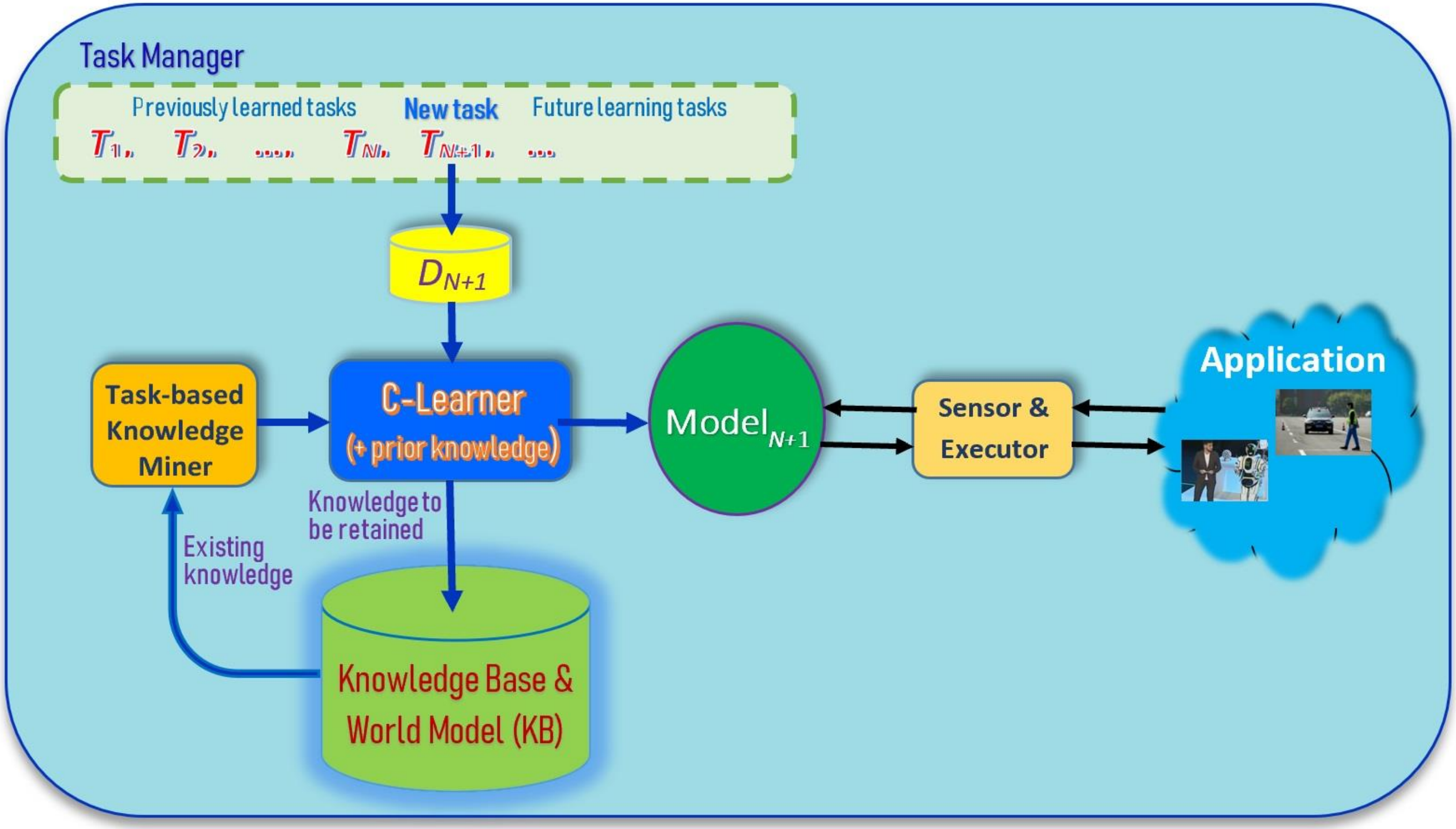
(Chen and Liu, 2014, 2018)

- 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: learn each new task T_{N+1} incrementally
  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}, and
  - its training data D_{N+1} are **given** by the user.

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

Characteristics of continual learning

(Chen and Liu, 2018, Liu, 2020)

- **Continuous and incremental learning process**
  - **Catastrophic forgetting:** In learning a new task, the system will need to change the network parameters, which may cause deterioration in performance of previous tasks.
- **Knowledge accumulation in KB** (long-term memory)
- **Knowledge transfer/adaptation** (across tasks)
  - Forward transfer: old task knowledge helps future task learning
  - Backward transfer: future task learning helps improve previous task models.

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

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

Sub-topics

- Lifelong or continual learning
  - Definition of lifelong or continual learning
  - Three main continual learning settings
  - Related machine learning paradigms

Different continual learning settings

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

Class incremental learning (CIL)

In CIL, the learning process builds a single classifier for all tasks/classes learned so far. In testing, a test instance from any class may be presented for the model to classify. No prior task information (e.g., task-id) of the test instance is provided. Formally, CIL is defined as follows.

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 $y_k^i \in \mathbf{Y}_k$ is its class label. All $\mathbf{Y}_k$'s are disjoint 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.

- Example: Today we learn to recognize *pig* and *chicken* (one task), and tomorrow, we also learn to recognize *sheep* (another task)

Task incremental learning (TIL)

In the TIL setup, each task is a separate classification problem (e.g., one task could be to classify different breeds of dogs and another task could be to classify different types of birds). Here, one model is built for each task in a shared network. In testing, the task-id of each test instance is provided and the system uses only the specific model for the task (dog or bird classification) to classify the test instance. Formally, TIL is defined as follows.

Task incremental learning (TIL). TIL learns a sequence of tasks, $1, 2, ..., T$. Each task k has a training dataset $\mathcal{D}_k = \{((x_k^i, k), y_k^i)_{i=1}^{n_k}\}$, where n_k is the number of data samples in task $k \in \mathbf{T} = \{1, 2, ..., T\}$, and $x_k^i \in \mathbf{X}$ is an input sample and $y_k^i \in \mathbf{Y}_k \subset \mathbf{Y}$ is its class label. The goal of TIL is to construct a predictor $f : \mathbf{X} \times \mathbf{T} \to \mathbf{Y}$ to identify the class label $y \in \mathbf{Y}_k$ for (x, k) (the given test instance x from task k).

Domain incremental learning (DIL)

In the DIL setting, all the tasks have the same set of classes and no task-id is provided for each test instance in testing. In other words, the tasks solve the same problem in different domains and different domains have different input distributions. Formally, DIL is defined as follows:

Domain incremental learning (DIL). DIL 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 $y_k^i \in \mathbf{Y}$ is its class label. $\mathbf{Y}$ is shared by all tasks. The goal of DIL is to construct a predictive function or classifier $f : \mathbf{X} \rightarrow \mathbf{Y}$ that can identify the class label y of each given test instance x.

- Example: One task is to classify car reviews as positive or negative and another task is to classify camera reviews as positive or negative. Car and camera are two domains.

Batch and online continual learning

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

Sub-topics

- Lifelong or continual learning
  - Definition of lifelong or continual learning
  - Three main continual learning settings
  - Related machine learning paradigms

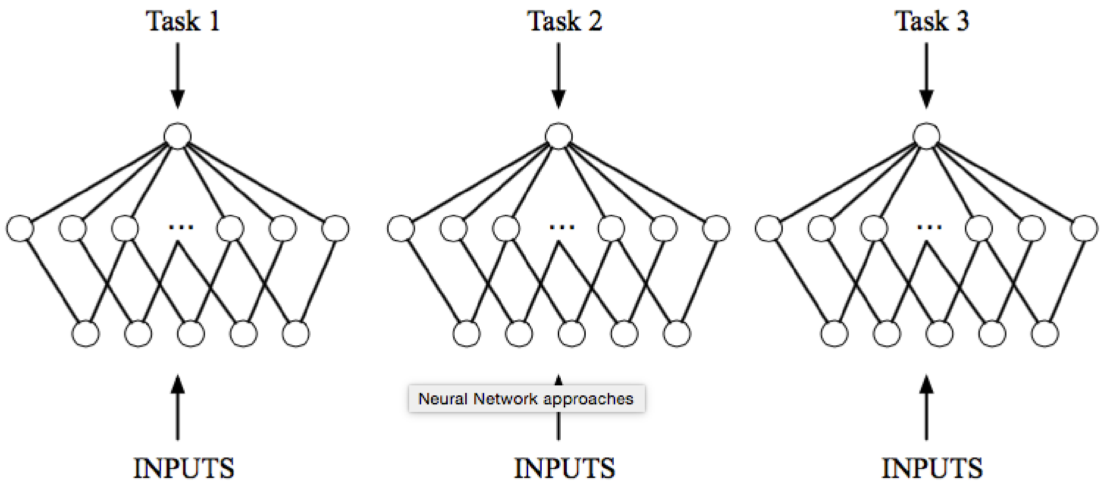
Transfer learning

- Source domain(s): With labeled training data
- Target domain: With little/no labeled training data
- Goal: leverage the information from the source domain(s) to help learn in the target domain
    - Only optimize the target domain/task learning

Multi-task learning (MTL)

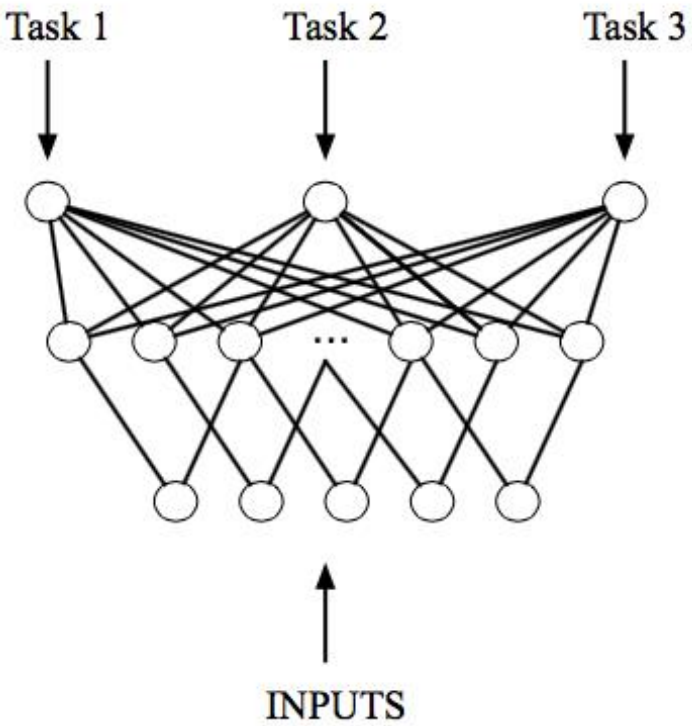
- Problem statement: Co-learn multiple related tasks simultaneously:
  - All tasks have labeled data and are treated equally
  - Goal: optimize learning across all tasks through shared knowledge
- Rationale: exploit the task relatedness structure and their shared knowledge.
- MTL is often considered the upper bound of continual learning
  - But we humans do not seem to be great at multitask learning, especially when many tasks are involved.

Neural Network approaches



Single task learning

Multitask learning



Transfer, Multitask vs. Continual Learning (CL)

- Transfer learning vs. CL
  - Transfer learning is not continuous
  - The source must be very similar to the target (decided by users)
  - No retention or accumulation of knowledge
  - Only one directional: source helps target
- Multitask learning vs. CL
  - Multitask learning retains no knowledge except data
  - Hard to re-learn all tasks whenever a new task appears
- Incremental (online) multitask learning is CL

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

Online Learning

- Training examples come in incrementally (online setting)
  - Computationally infeasible to train over the entire dataset
  - Training data come gradually in a data stream
- Different from CL
  - Online learning still performs the same learning task over time, no data distribution change.
  - CL aims to learn from a sequence of different tasks, retaining and accumulating knowledge

Topics

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

Sub-topics

- Early research on lifelong learning
  - Lifelong supervised learning
  - Semi-supervised never-ending learning
  - Lifelong topic modeling

Early work on 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 $f: I \to \{0, 1\}$. $f(x)=1$ means x is a particular concept.
  - For example, $f_{dog}(x)=1$ means x is a dog.
- For *n*th task, we have its training data X
  - Also the training data X_k of $k = 1, 2, \ldots, n-1$ 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.

Intuition of (Thrun, 1996)

- The paper proposed a few approaches based on two learning algorithms,
  - Memory-based, e.g., kNN or shepard's method
  - Neural networks,
- Intuition: when we learn $f_{dog}(x)$, we can use 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.

Memory based lifelong learning

- First method: use the support sets of $\{f_1, f_2, \ldots, f_k, \ldots, f_{n\text{-}1}\}$ 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.

Second Method

- It learns a distance function using the support sets

  d: $I \times I \rightarrow [0, 1]$

  - d is trained with neural network using back-prop, and used as a general distance function
  - It takes two input vectors x and x' from a pair of examples <x, y>, <x', y'> of the same support set X_k (k = 1, 2, , …, n-1) to form training examples:

$$\langle (x, x'), 1 \rangle \quad \text{if } y{=}y'{=}1$$
$$\langle (x, x'), 0 \rangle \quad \text{if } (y{=}1 \wedge y'{=}0) \text{ or } (y{=}0 \wedge y'{=}1)$$

Making Decision

- Given the new task training set X_n and a test vector x, for each +ve example, (x', y'=1)$\in X_n$,
  - d(x, x') is the probability that x is a member of the target concept.
- Decision is made by using votes from positive examples, <x_1, 1>, <x_2, 1>, … $\in X_n$ combined with Bayes' rule

$$P(f_n(x)=1) = 1 - \left(1 + \prod_{\langle x', y'=1\rangle \in X_n} \frac{d(x,x')}{1-d(x,x')}\right)^{-1}$$

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 **S**. **S** captures the task grouping structure.

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

GO-MTL objective and inefficiency

- Since GO-MTL is a batch multitask learning method, the optimization goes through all tasks and their training instances (Kumar et al., 2012).

$$\sum_{t=1}^{T}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\boldsymbol{x}_i^{(t)};\boldsymbol{L}\boldsymbol{s}^{(t)}),y_i^{(t)}\right)+\mu\|\boldsymbol{S}\|_1+\lambda\|\boldsymbol{L}\|_F^2$$

- Very inefficient and impractical for a large number of tasks.
  - It cannot incrementally add a new task efficiently

Kumar, Daum, and Daume Iii. Learning Task Grouping and Overlap in Multi-task Learning. ICML-2012

Initial Objective Function of ELLA

- Objective Function (**Average** rather than sum)

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

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 detailed derivations)

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

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

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

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

Simplify optimization

- GO-MTL: when computing a single candidate $\boldsymbol{L}$, an optimization problem must be solved to re-compute the value of each $s^{(t)}$.

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

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)	$98.90 \pm 1.5\%$

Batch MTL is GO-MTL

ELLA Speed Result

- ELLA vs. GO-MTL

Dataset	Batch Runtime (seconds)	ELLA All Tasks (speedup)	ELLA New Task (speedup)
Land Mine	231±6.2	1,350±58	39,150±1,682
Facial Expr.	2,200±92	1,828±100	38,400±2,100
Syn. Data	1,300±141	5,026±685	502,600±68,500
London Sch.	715±36	2,721±225	378,219±31,275

ELLA is 1K times faster than GO-MTL on all tasks, 30K times on a new task

Lifelong Sentiment Classification

(Chen, Ma, and Liu 2015)

- "*I bought a cellphone a few days ago. It is such a nice phone. The touch screen is really cool. The voice quality is great too. ....*"
- Goal: classify docs or sentences as + or -.
  - Need to manually label a lot of training data for each domain, which is highly labor-intensive
- Can we not label for every domain or at least not label so many docs/sentences?

Chen, Ma and Liu. Lifelong Learning for Sentiment Classification. ACL-2015.

A Simple Lifelong Learning Method

Assuming we have worked on a *large number of past domains* with all their training data *D*

- Build a classifier using *D*, test on new domain
  - Note - using only one past/source domain as in ***transfer learning*** is not good.
- In many cases – improve accuracy by as much as 19% (= 80%-61%). Why?
- In some others cases – not so good, e.g., it works poorly for toy reviews. Why? “toy”

Lifelong Sentiment Classification (LSC)

(Chen, Ma and Liu, 2015)

- It adopts a Bayesian optimization framework for LL using stochastic gradient decent (SGD)

- Lifelong learning (LL) uses
  - Word counts from the past data as priors.
  - Penalty terms to deal with domain dependent sentiment words and reliability of knowledge.

Naïve Bayesian Text Classification

- Key parameter

$$P(w|c_j) = \frac{\lambda + N_{c_j,w}}{\lambda|V| + \sum_{v=1}^{|V|} N_{c_j,v}}$$

- Only depends on the count of words in each class

Exploiting Knowledge via Penalties

- Objective: given a new domain training data D^t, the objective is

$$\sum_{i=1}^{|D^t|} \left(P\left(c_j|d_i\right) - P\left(c_f|d_i\right)\right)$$

- Penalty terms for two types of knowledge
  - Document-level knowledge in the target domain

$$\frac{1}{2}\alpha \sum_{w \in V_T} \left(\left(X_{+,w} - N_{+,w}^t\right)^2 + \left(X_{-,w} - N_{-,w}^t\right)^2\right)$$

    - Optimization variables $X_{+,w}$ and $X_{-,w}$ as the number of times that a word w appears in the positive and negative class in the new/target domain

Exploiting Knowledge via Penalties

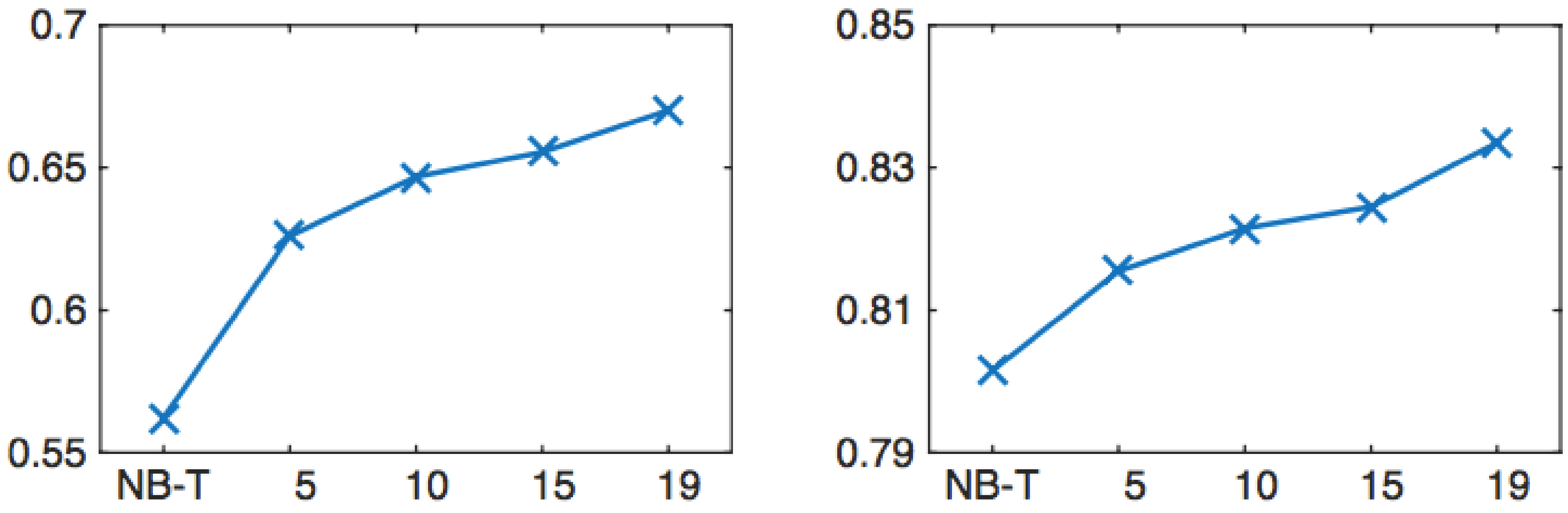
- Penalty terms for two types of knowledge
  - Document-level knowledge in the target domain
  - Domain-level knowledge across all previous domains

$$\frac{1}{2}\alpha \sum_{w \in V_S} \left(X_{+,w} - R_w \times X^0_{+,w}\right)^2 + \frac{1}{2}\alpha \sum_{w \in V_S} \left(X_{-,w} - (1 - R_w) \times X^0_{-,w}\right)^2$$

  - R_W: ratio of #tasks where w is positive / #all tasks
  - $X^0_{+,w} = N^t_{+,w} + N^{KB}_{+,w}$ and $X^0_{-,w} = N^t_{-,w} + N^{KB}_{-,w}$

One Result of LSC model

- Better F1-score (left) and accuracy (right) with more past tasks



Sub-topics

- Early research on lifelong learning
  - Lifelong supervised learning
  - Semi-supervised never-ending learning
  - Lifelong topic modeling

Never Ending Language Learner

(Carlson et al., 2010; Mitchell et al., 2015)

- NELL: Never Ending Language Learner
- Perhaps the only live LML system
  - It has been reading the Web to extract certain types of information (or knowledge)
  - 24/7 since January 2010.
- NELL has accumulated millions of facts with attached confidence weights
  - called beliefs

Mitchell, Cohen, Hruschka, et al. Never-ending learning. AAAI-2015.

Input to NELL

- An ontology defining a set of target categories and relations to be learned,
  - a handful of seed training examples for each, and
  - a set of coupling constraints about categories and relations (Person & Sport are mutually exclusive).
- Webpages crawled from the Web
- Interactions with human trainers to correct some mistakes made by NELL

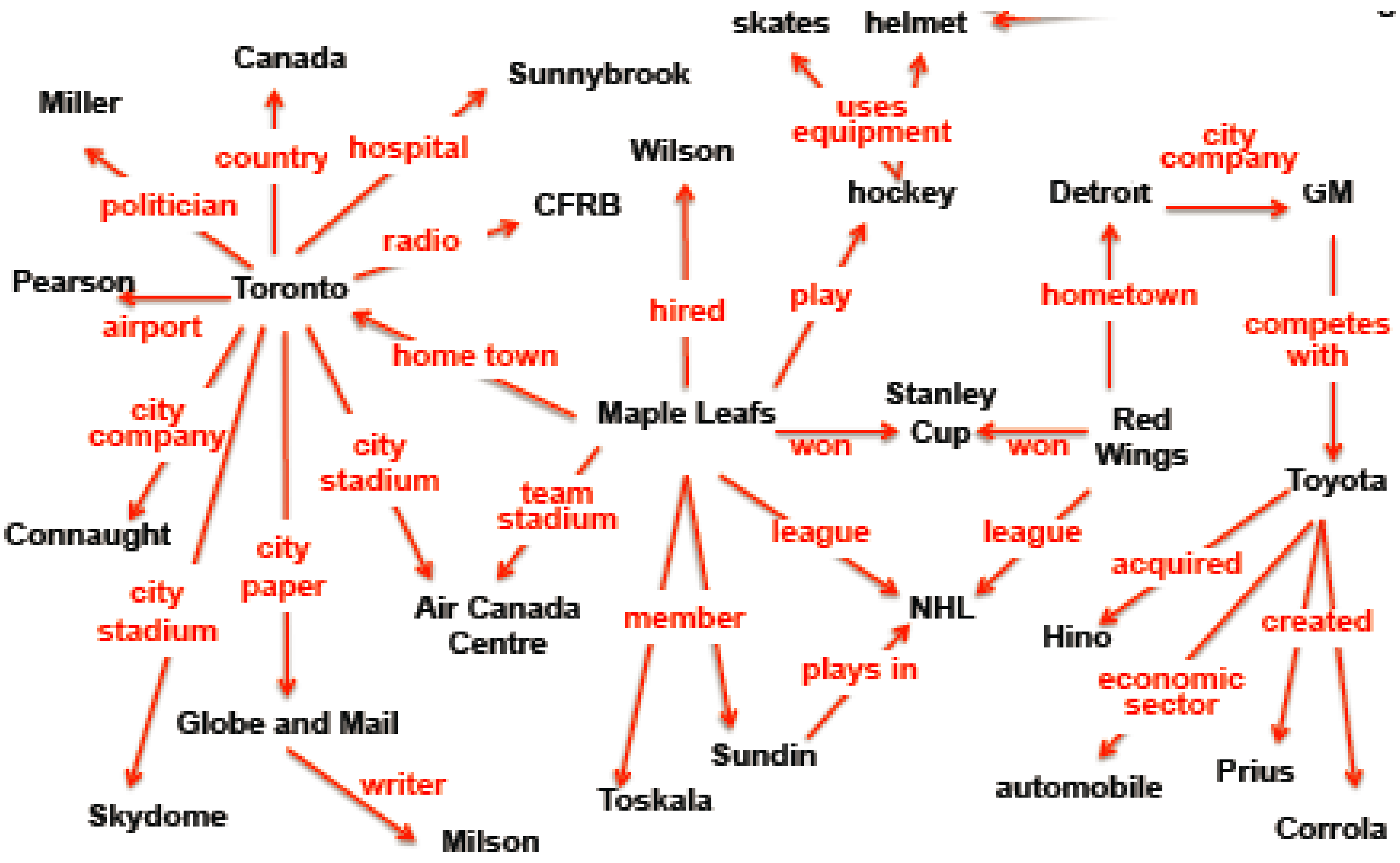
Goal of NELL

- Reading - extract facts from the webpages to populate the initial ontology
  - *category* of a noun or noun phrase, e.g., Los Angeles is a *city*
  - *relations* of a pair of noun phrases
    - hasMajor(Stanford, Computer Science)
- Learn to perform the above extraction tasks better each day.

Knowledge Base

- Instance of category: which noun phrases refer to which specified semantic categories
  - For example, *Los Angeles* is in the category *city*.
- Relationship of a pair of noun phrases, e.g., given a name of an organization and the location, check if
  - *hasOfficesIn(organization, location)*.
- …

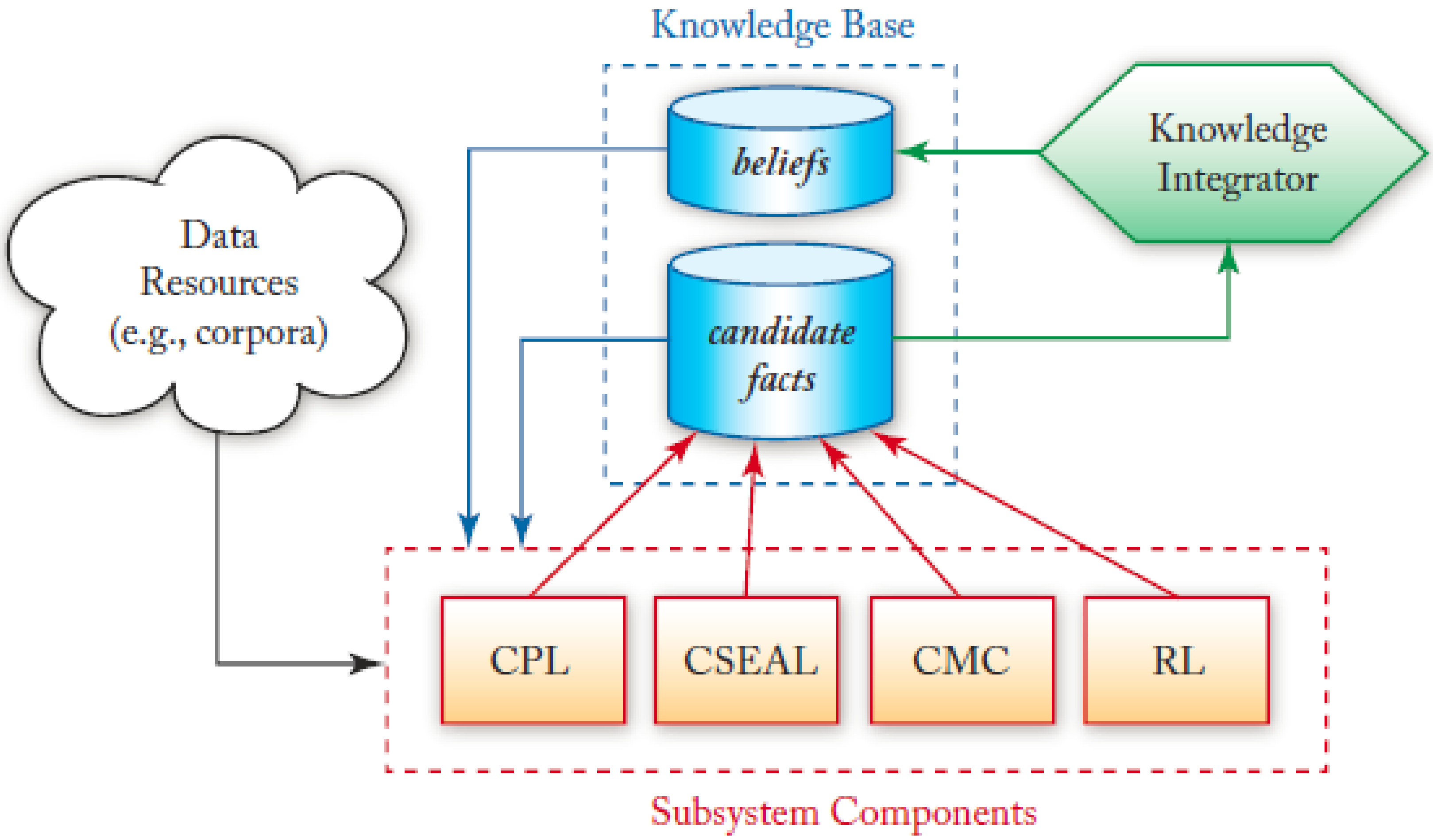
NELL Knowledge Fragment



Semi-supervised Learning

- Training examples
  - human-labeled instances in NELL's ontology
  - labeled examples contributed over time through NELL's crowdsourcing website,
  - a set of NELL self-labeled training examples corresponding to NELL's current knowledge base,
  - a large amount of unlabeled Web text.
- 2nd and 3rd sets of the training examples propel NELL's lifelong learning

NELL Architecture



Coupled Pattern Learner (CPL):

- CPL: extractors extracting both category and relation instances using contextual patterns.
  - Examples
    - Category pattern: “mayor of X” and
    - Relation pattern: “X plays for Y”
- Such patterns can also be learned.
- Mutual exclusion & type-checking constraints
  - filtered out candidate facts to ensure correctness

Coupled SEAL (CSEAL)

- CSEAL: an extraction and learning system that extracts facts from semi-structured webpages using wrapper induction
- Based on set expansion or PU learning
  - Wrapper: character strings specifying the left and right context of an entity.
- Mutual exclusion & type-checking constraints:
  - filtered out likely errors

Coupled Morphological Classifier (CMC)

- CMC: a set of binary classifiers, one for each category,
  - To classify whether the extracted candidate facts/beliefs by other subsystems are indeed of their respective categories.
- Positive training examples:
  - beliefs in the current knowledge base.
- Negative training examples
  - beliefs satisfying mutual exclusion constraints

Rule Learner (RL)

- Its goal is to learn probabilistic Horn clauses
  - to use them to infer new relations from the existing relations in the knowledge base.

- This reasoning capability
  - represents an important advance of NELL
  - It does not exist in most current LML systems.

Coupling Constraints in NELL

- *Multi-view co-training coupling constraint*
  - Agreement: the same category or relation learned from different data sources, or *views*.
- *Subset/superset coupling constraint*
  - When a new category is added to NELL's ontology, its parents (supersets) are also specified.
- *Horn clause coupling constraint*
  - E.g., "X living in Chicago" and "Chicago being a city in U.S." $\rightarrow$ "X lives in U.S."

Sub-topics

- Early research on lifelong learning
  - Lifelong supervised learning
  - Semi-supervised never-ending learning
  - Lifelong topic modeling

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 terms/words, e.g.,
    - {*price*, *cost*, *cheap*, *expensive*, …}
- **Question**: how to find good past knowledge and use it to help new topic modeling tasks?
- **Data**: product reviews in the sentiment analysis context

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 (product features): size, picture
- Why lifelong learning can help SA?
  - Online reviews: Excellent data with extensive sharing of aspect/concepts across domains
    - A large volume for all kinds of products
- Why big (and diverse) data?
  - Learn a broad range of reliable knowledge. More knowledge makes future learning easier.

Key observation in practice

- 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.
  - It is rather "silly" not to exploit such sharing in learning

Problem setting

- Given a large set of document collections, $D = \{D_1, D_2, \ldots, D_N\}$, learn from each D_i to produce the results S_i. Let $S = \mathrm{U}_i\, S_i$.
  - S is called *topic base*
- Goal: Given a test/new collection D^t, learn from D^t with the help of S (and possibly D).
  - D^t in D or D^t not in D
  - The results learned this way should be better than those without the guidance of S (and D)

What knowledge?

- Should be in the same aspect/topic

  => Must-Links

  e.g., {picture, photo}

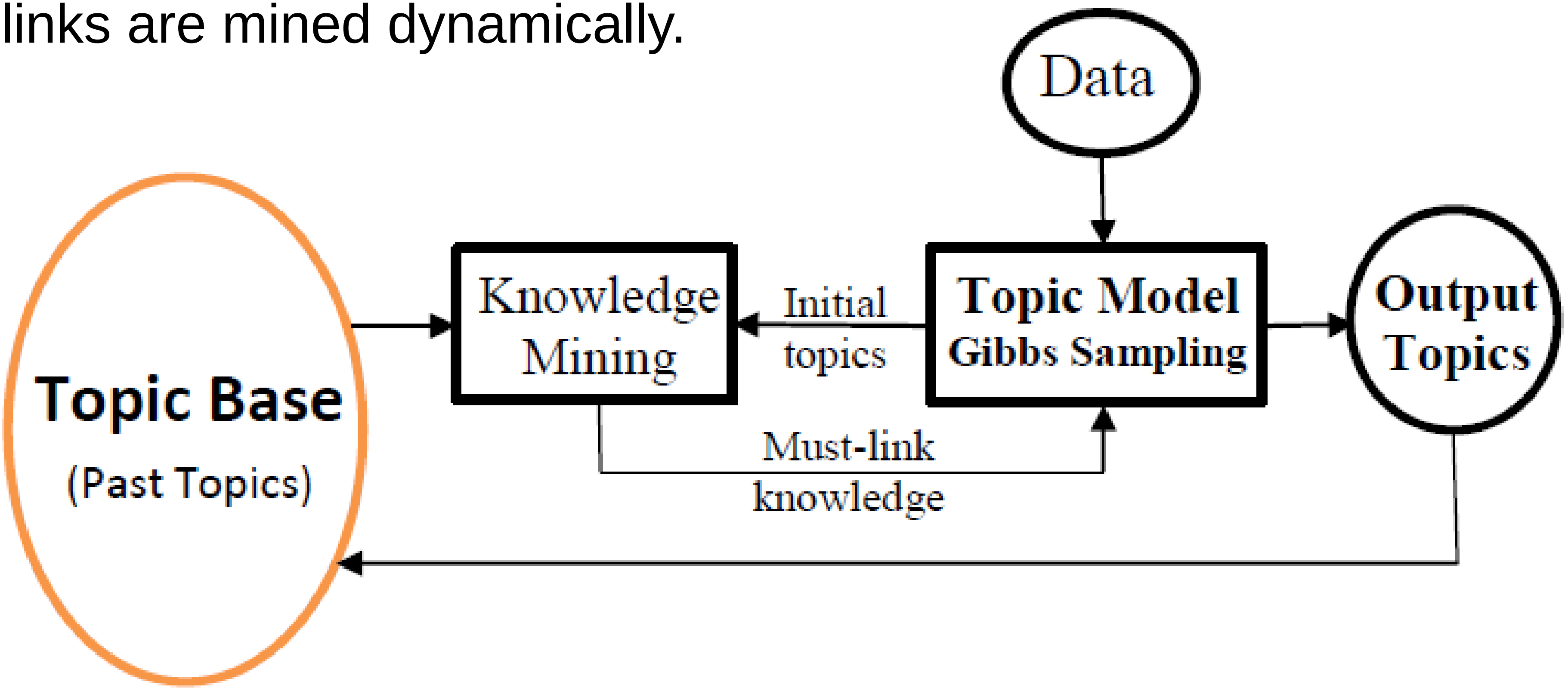
- Should not be in the same aspect/topic

  => 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 a topic model (e.g., LDA) on each 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.

LTM Model

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: $\quad K^t \leftarrow$ KnowledgeMining(A^t, S);
4: $\quad A^t \leftarrow$ GibbsSampling(D^t, K^t, 1); // Run with knowledge K^t.
5: **end for**

Knowledge Mining Function

- Topic matching: find similar topics from topic base for each topic in the new domain

- Pattern mining: find frequent itemsets from the matched topics

An Example

- Given a newly discovered topic:

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

  - We find 3 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.

Knowledge Mining Function

Algorithm 3 KnowledgeMining(A^t, S)

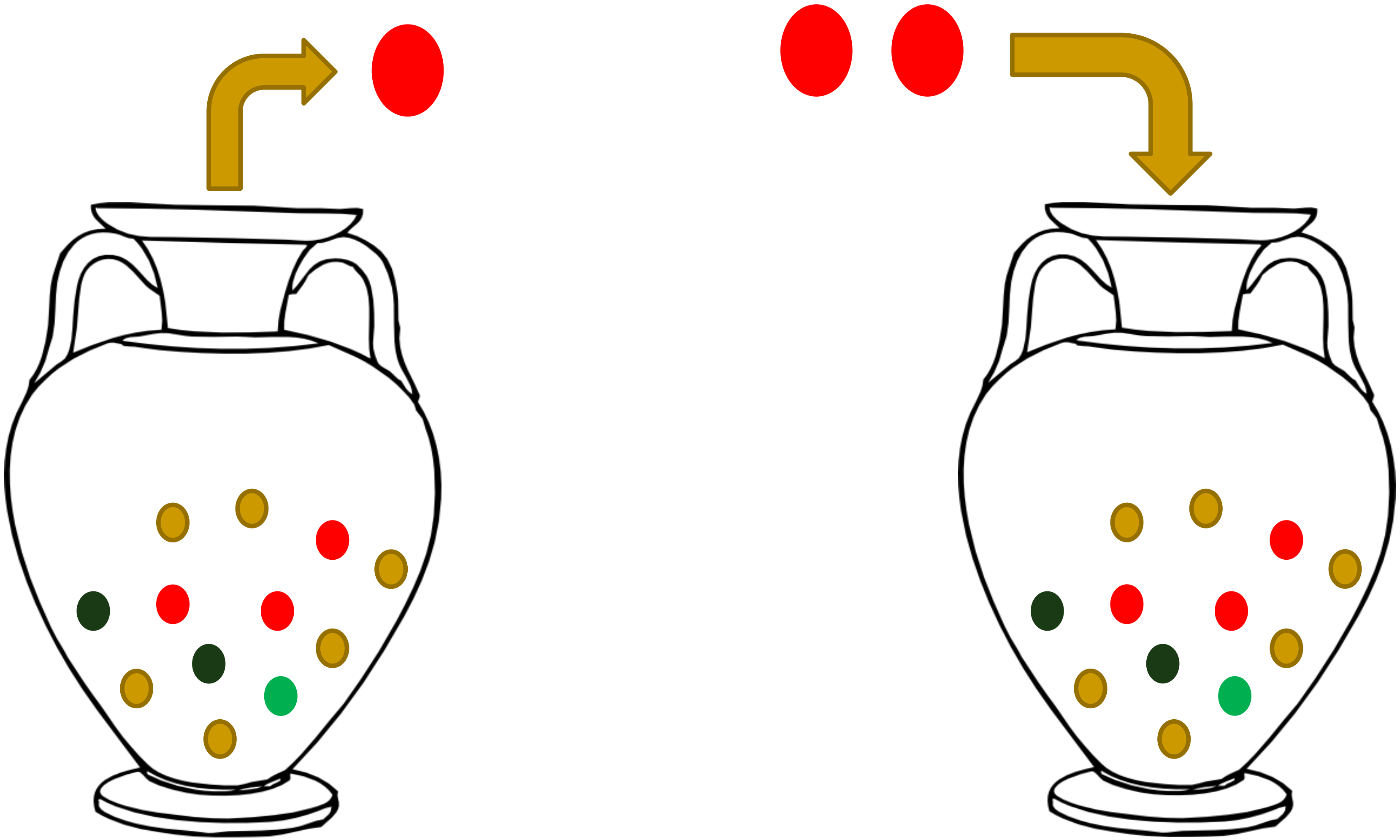
1: **for** each p-topic $s_k \in S$ **do**
2: $\quad j^* = \min_j$ KL-Divergence(a_j, s_k) for $a_j \in A^t$;
3: $\quad$ **if** KL-Divergence(a_{j^*}, s_k) $\leq \pi$ **then**
4: $\quad\quad M^t_{j^*} \leftarrow M^t_{j^*} \cup s_k$;
5: $\quad$ **end if**
6: **end for**
7: $K^t \leftarrow \cup_{j^*}$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

Model Inference: Gibbs Sampling

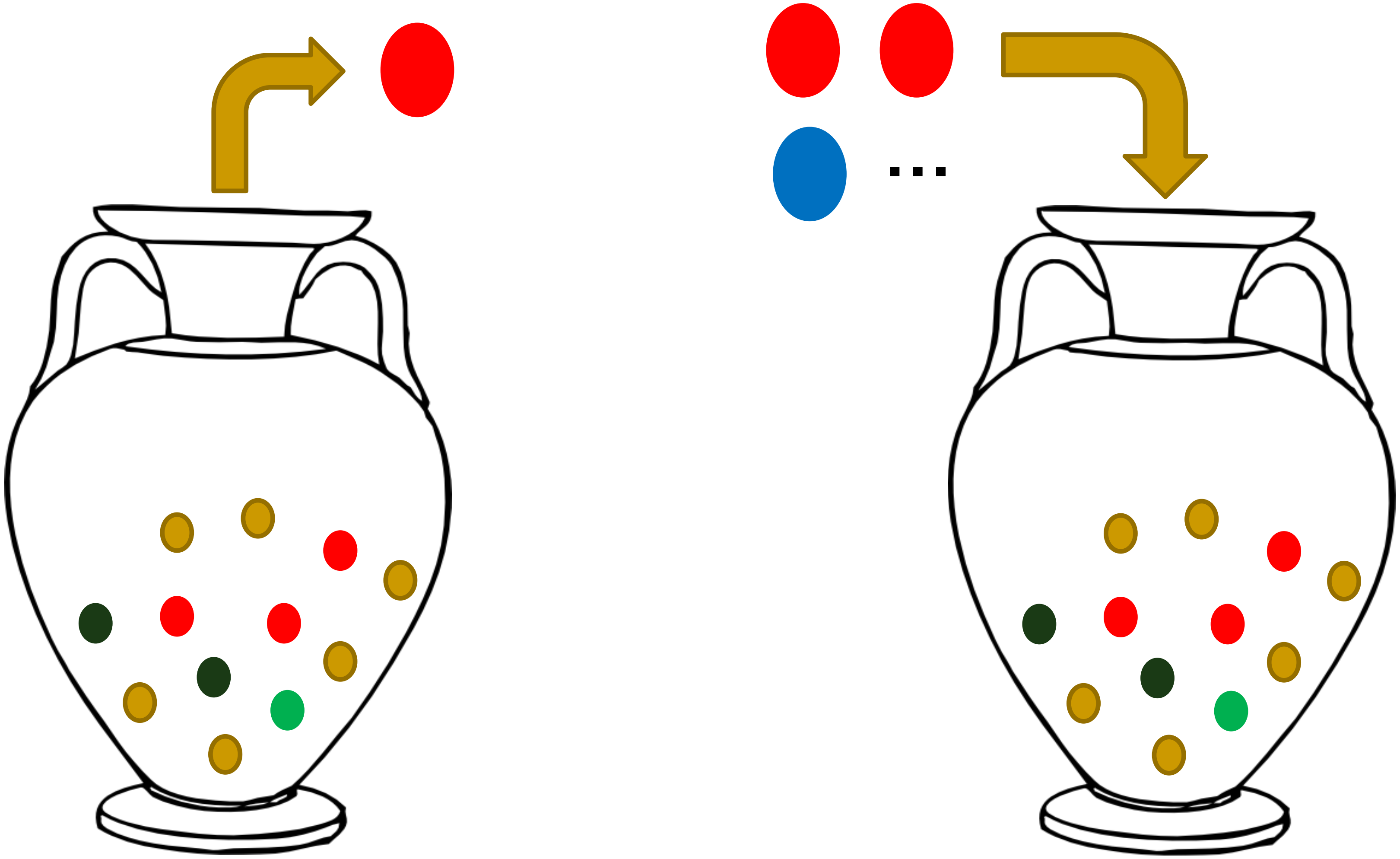
- How to use the *must-links* knowledge?
  - e.g., {*price*, *cost*} & {*price*, *expensive*}

- 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

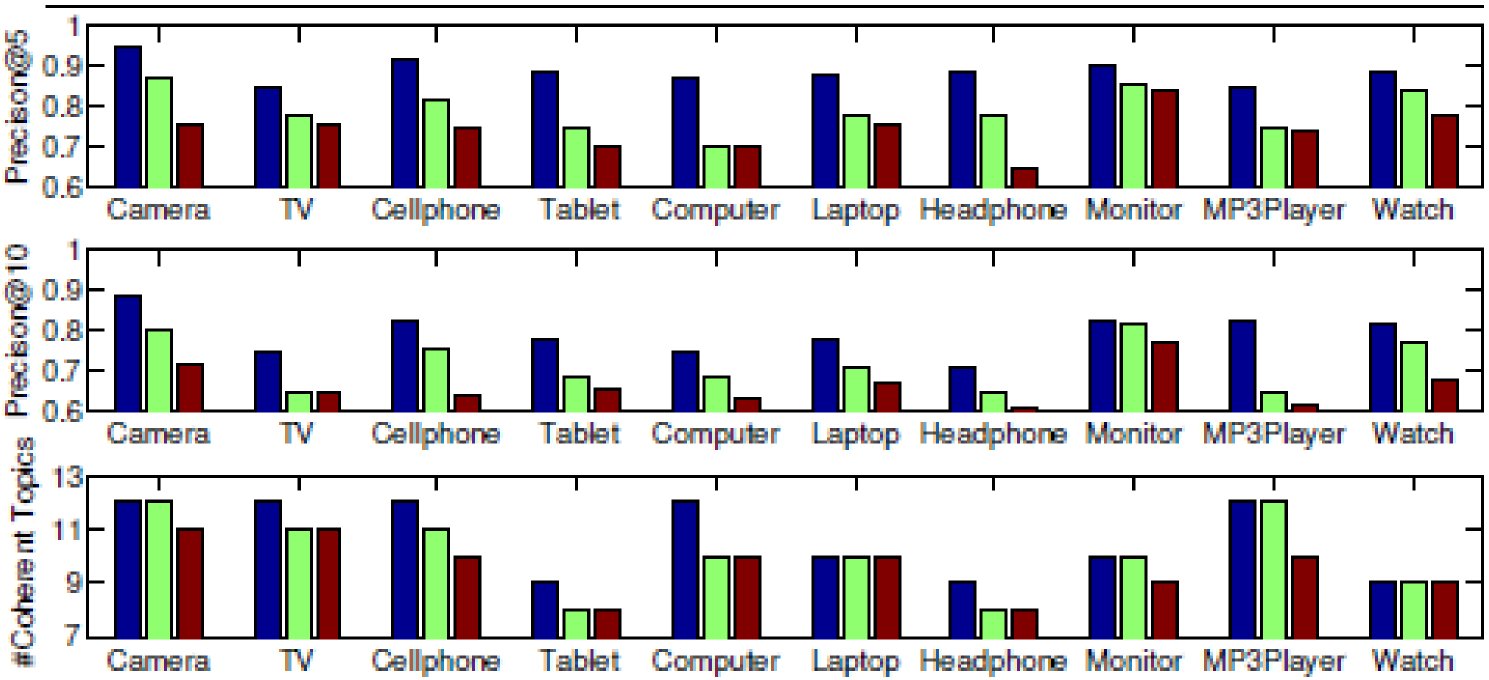


Figure 2. Top & Middle: Topical words *Precision@5* & *Presicion@10* of coherent topics of each model respectively; Bottom: number of coherent (#Coherent) topics discovered by each model. The bars from left to right in each group are for LTM, LDA, and DF-LDA. On average, for *Precision@5* and

Topics

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

Brief Self-introduction

- Who am I?
  - Zixuan Ke
    - **Affiliation:** University of Illinois, Chicago
    - **Advisor:** Bing Liu
    - **Research Interests:** Continual Learning, Natural Language Processing

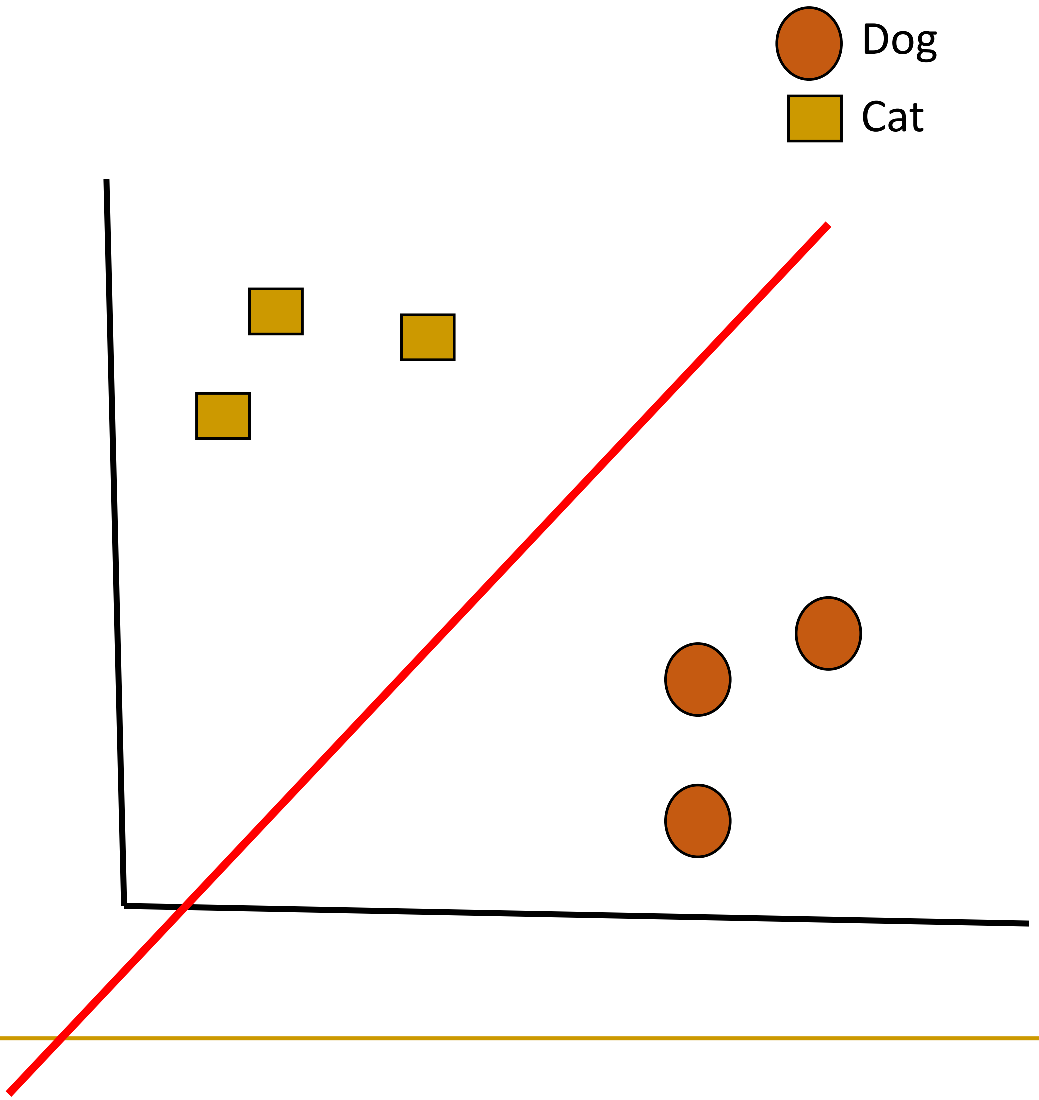
What are the challenges in deep learning era?

- Recall
  - After a task is learned, its training data (at least a large proportion of it) is no longer accessible.
  - Earlier work on lifelong/continual learning mainly builds separate models for knowledge transfer.

- New Goal:
  - We want one **single** neural model to be able to do well in all tasks
  - What will happen?

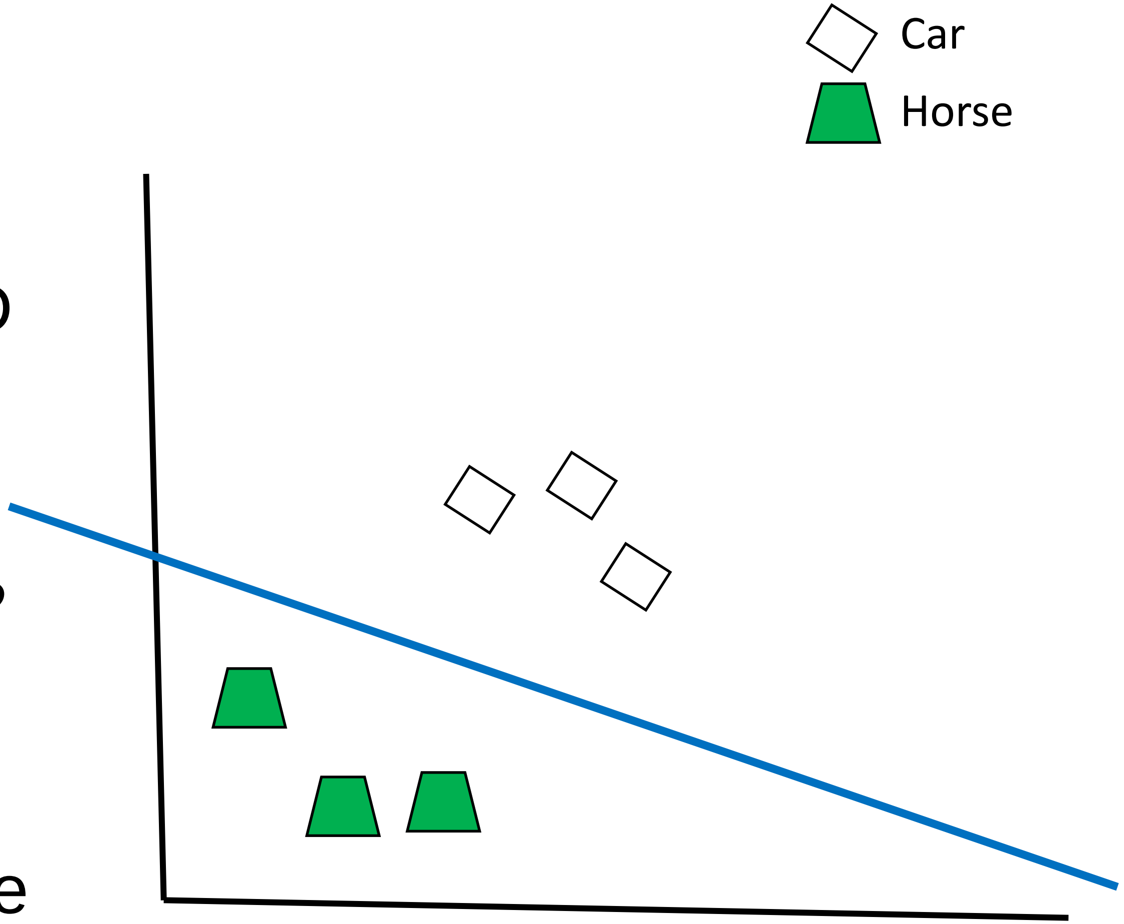
What are the challenges in the deep learning era?

- A simple case:
  - 2 features (x,y)
  - 2 degree of freedom (line in 2D plane)
- One can easily draw a perfect line for task 1



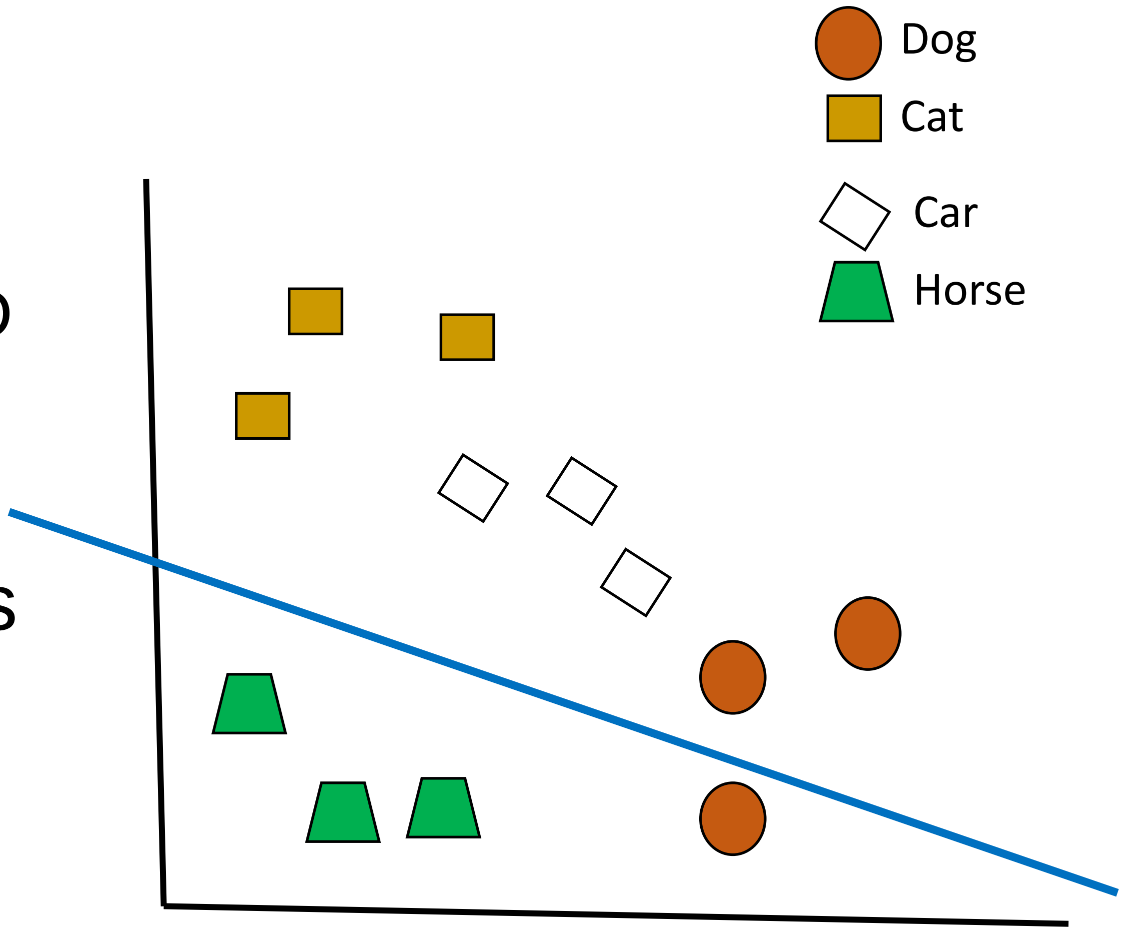
What are the challenges in the deep learning era?

- A simple case:
  - 2 features (x,y)
  - 2 degree of freedom (line in 2D plane)
- A new task 2 comes
  - *You train your model to learn 2*
  - The learned parameters are changed.
  - A different and perfect separate line for task 2 is drawn



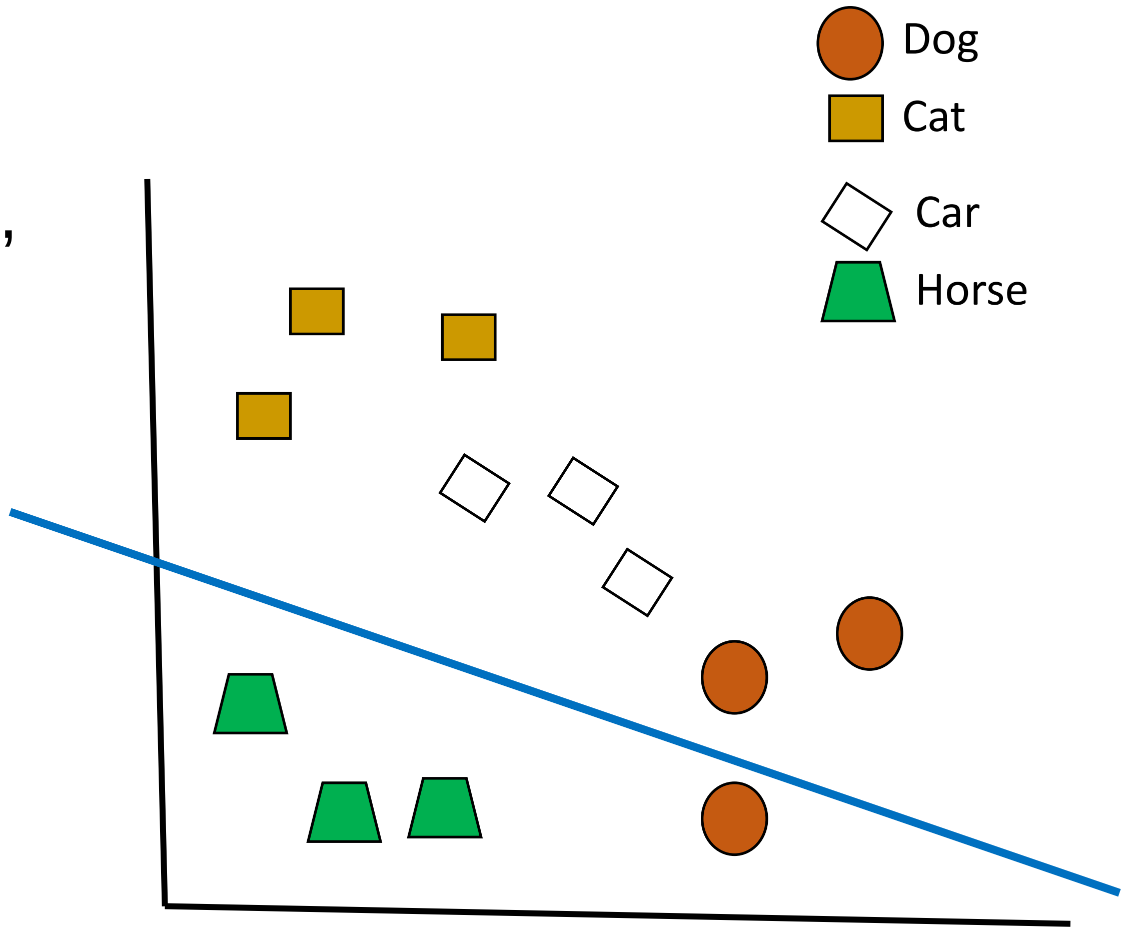
What are the challenges in the deep learning era?

- A simple case:
  - 2 features (x,y)
  - 2 degree of freedom (line in 2D plane)
- Recall we want a single model works well on all tasks
  - If we test the current model on all tasks
  - It cannot do well on the previous tasks anymore!



What are the challenges in the deep learning era?

- Take away
  - After Learning the second task, the model forgets how to deal with the first task!
  - Catastrophic forgetting



Forgetting! But why?

Overall Goal:

$$\sum_{t=1}^{\mathcal{T}} \mathop{\mathbb{E}}_{(\mathcal{X}^{(t)},\mathcal{Y}^{(t)})} [\mathcal{L}(f_t(\mathcal{X}^{(t)};\theta), \mathcal{Y}^{(t)})$$

#tasks seen so far

Forgetting! But why?

Overall Goal:

$$\sum_{t=1}^{\mathcal{T}} \mathop{\mathbb{E}}_{(\mathcal{X}^{(t)},\mathcal{Y}^{(t)})} [\mathcal{L}(f_t(\mathcal{X}^{(t)};\theta),\mathcal{Y}^{(t)})$$

#tasks seen so far

Data drawn from task t's distribution

Forgetting! But why?

Overall Goal:

Network for task t

Network weights

$$\sum_{t=1}^{\mathcal{T}} \mathbb{E}_{(\mathcal{X}^{(t)},\mathcal{Y}^{(t)})}[\mathcal{L}(f_t(\mathcal{X}^{(t)};\theta),\mathcal{Y}^{(t)})]$$

#tasks seen so far

Data drawn from task t's distribution

Forgetting! But why?

Overall Goal: $$\sum_{t=1}^{\mathcal{T}} \mathop{\mathbb{E}}_{(\mathcal{X}^{(t)},\mathcal{Y}^{(t)})} [\mathcal{L}(f_t(\mathcal{X}^{(t)};\theta),\mathcal{Y}^{(t)})$$

For the current task, we can approximate the empirical risk:

$$\frac{1}{N_{\mathcal{T}}} \sum_{i=1}^{N_{\mathcal{T}}} \ell(f(x_i^{(\mathcal{T})};\theta), y_i^{(\mathcal{T})}) \ .$$

#samples in current task

Forgetting! But why?

Overall Goal:

$$\sum_{t=1}^{\mathcal{T}} \mathop{\mathbb{E}}_{(\mathcal{X}^{(t)},\mathcal{Y}^{(t)})} [\mathcal{L}(f_t(\mathcal{X}^{(t)};\theta),\mathcal{Y}^{(t)})$$

For current task:

$$\frac{1}{N_{\mathcal{T}}} \sum_{i=1}^{N_{\mathcal{T}}} \ell(f(x_i^{(\mathcal{T})};\theta), y_i^{(\mathcal{T})}) \ .$$

For old tasks: Old tasks data is NO longer available.

So, we are NOT able to compute the empirical risk for them using the new parameters

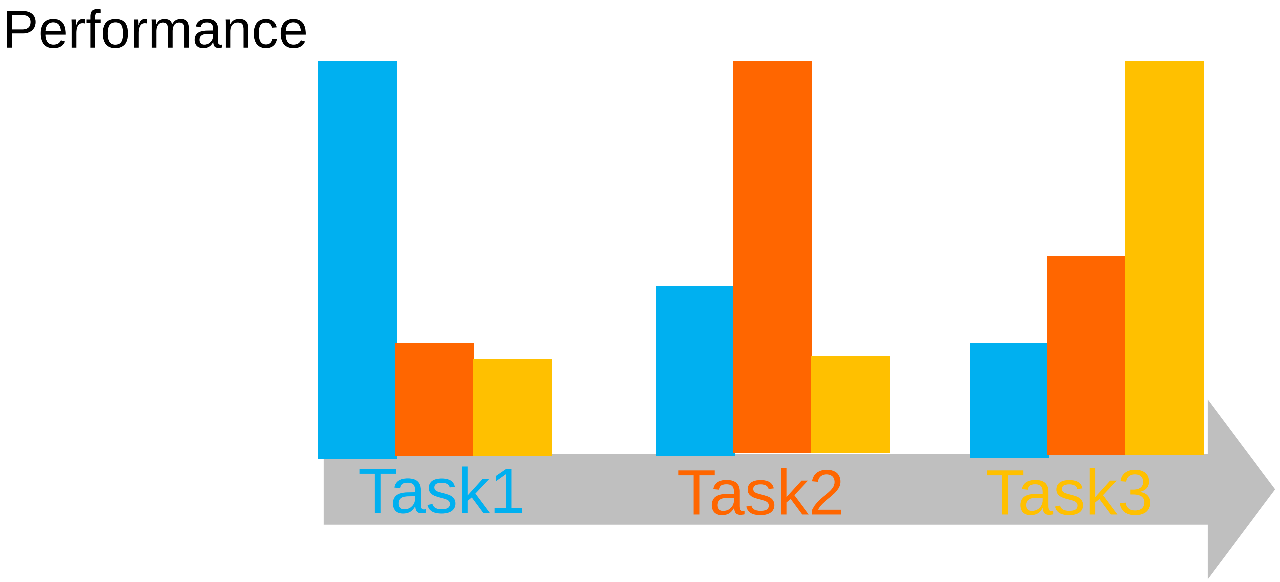
This is why we have **Catastrophic Forgetting (CF)**

Research Question: how to mitigate the CF?

Possible Scenarios in CL

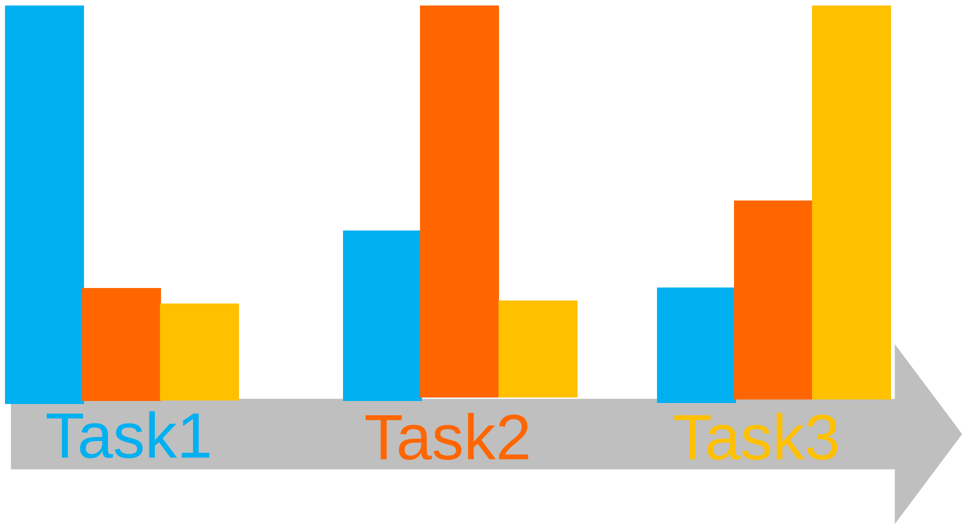
- We now know the catastrophic forgetting problem in continual learning
  - Both intuitively and mathematically
- Quiz:
  - Given possible experiments results in continual learning, can you differentiate them and know its meaning?

Possible Scenarios in CL

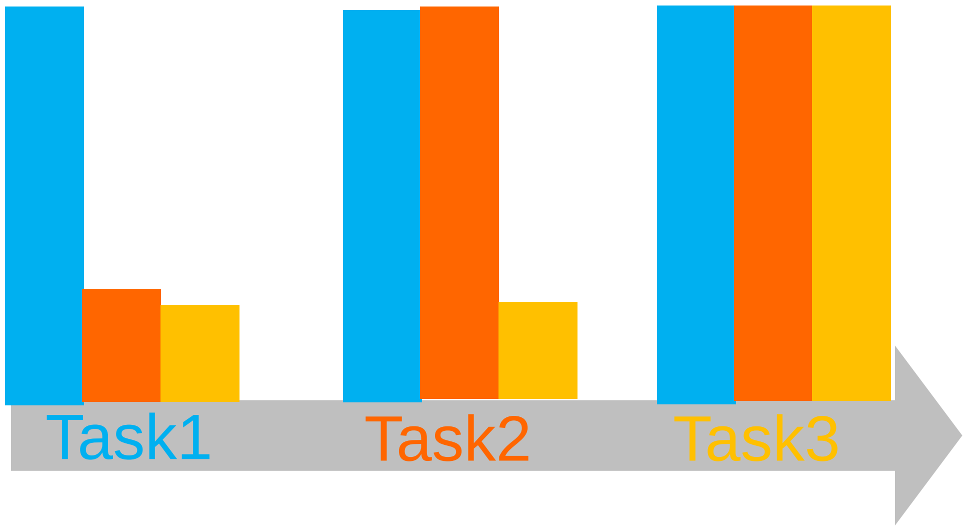


Possible Scenarios in CL

- Learned task performance is getting worse after a new task is trained
- <u>Catastrophic forgetting</u>

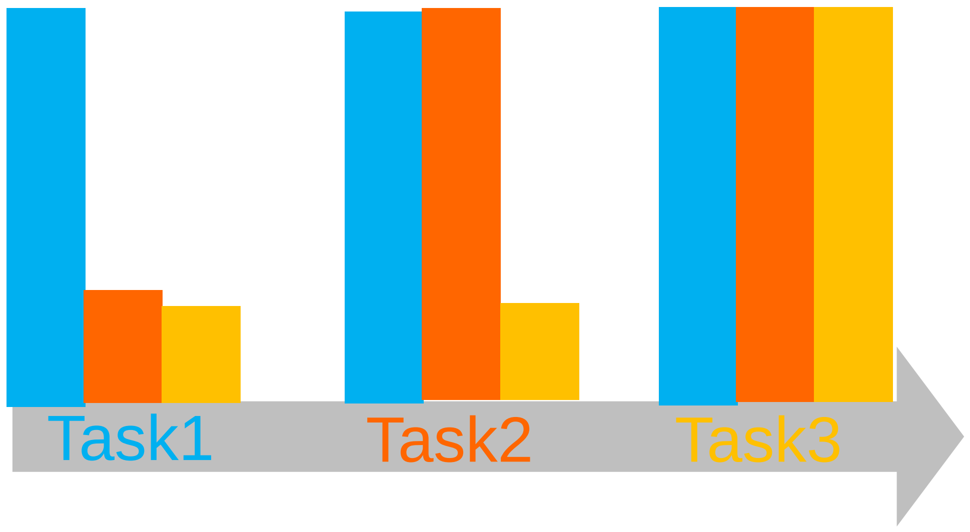


Possible Scenarios in CL

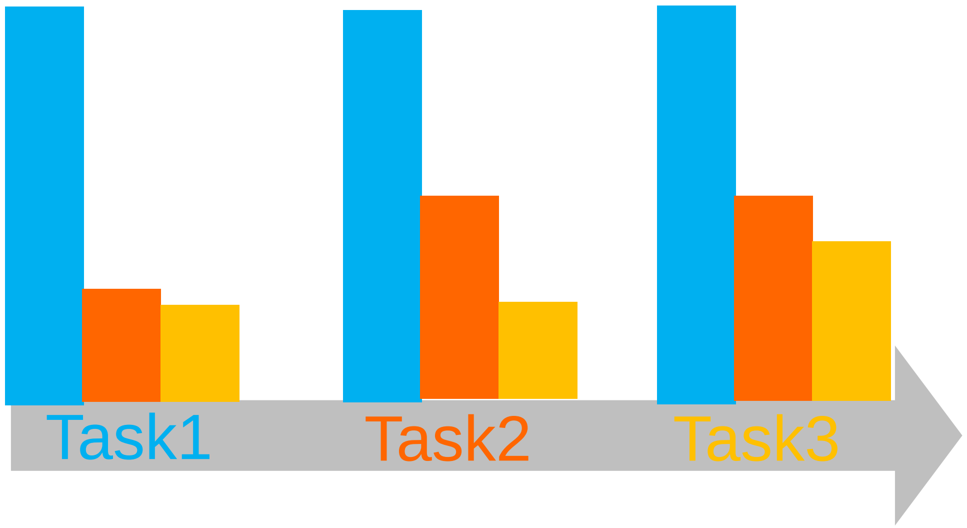


Possible Scenarios in CL

- Learned task performance is retained after a new task is trained
- <u>No forgetting</u>

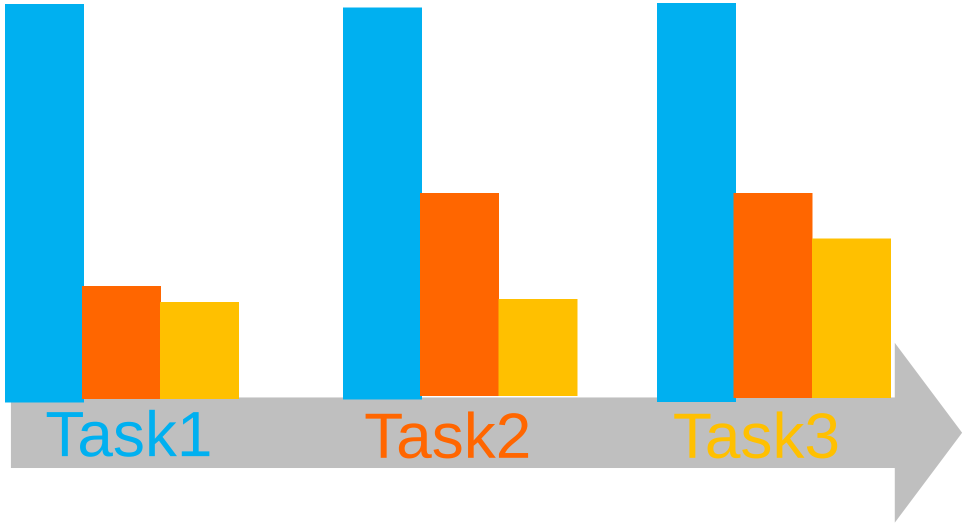


Possible Scenarios in CL

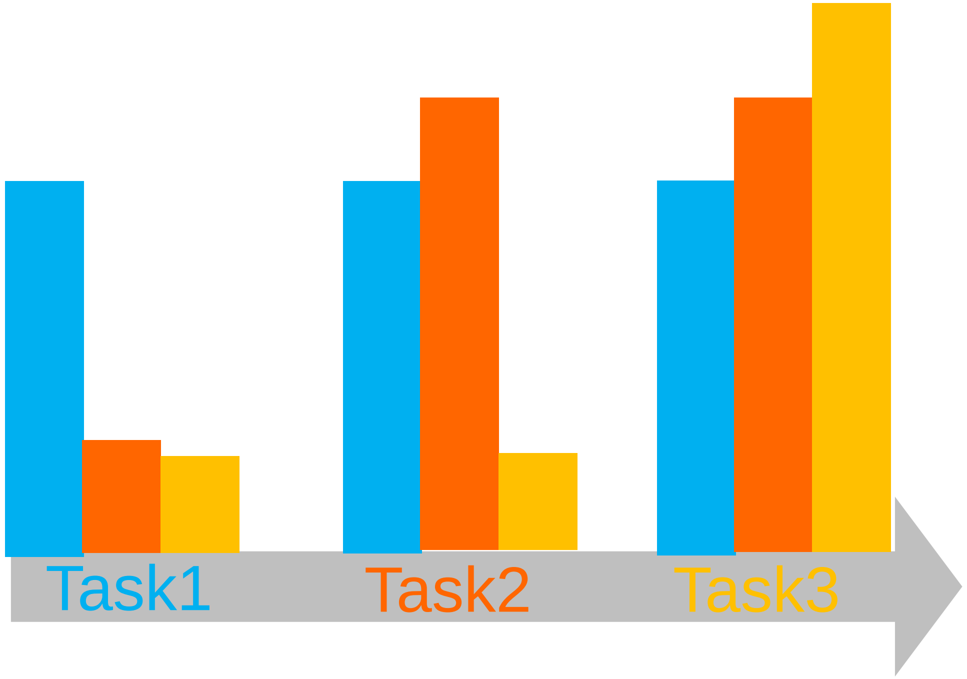


Possible Scenarios in CL

- Learned task performance is retained, but the new tasks are not trained well
- Problematic Learning
  - Why?
    - Over regularization (we will introduce the regularization-based approach later)
    - Lack of network capacity (we will introduce the architecture-based approach later)

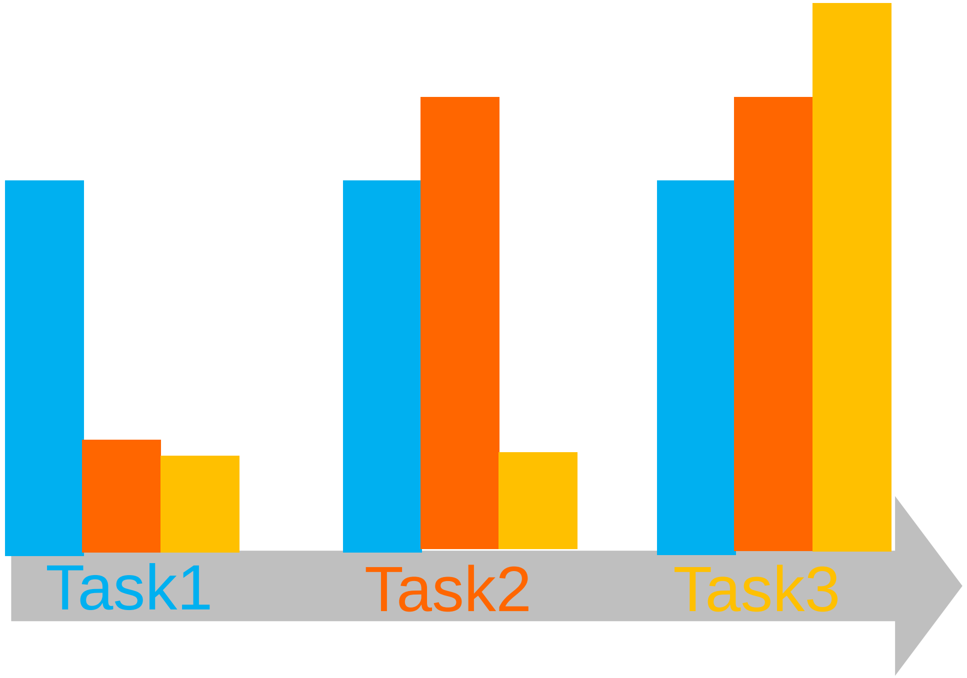


Possible Scenarios in CL

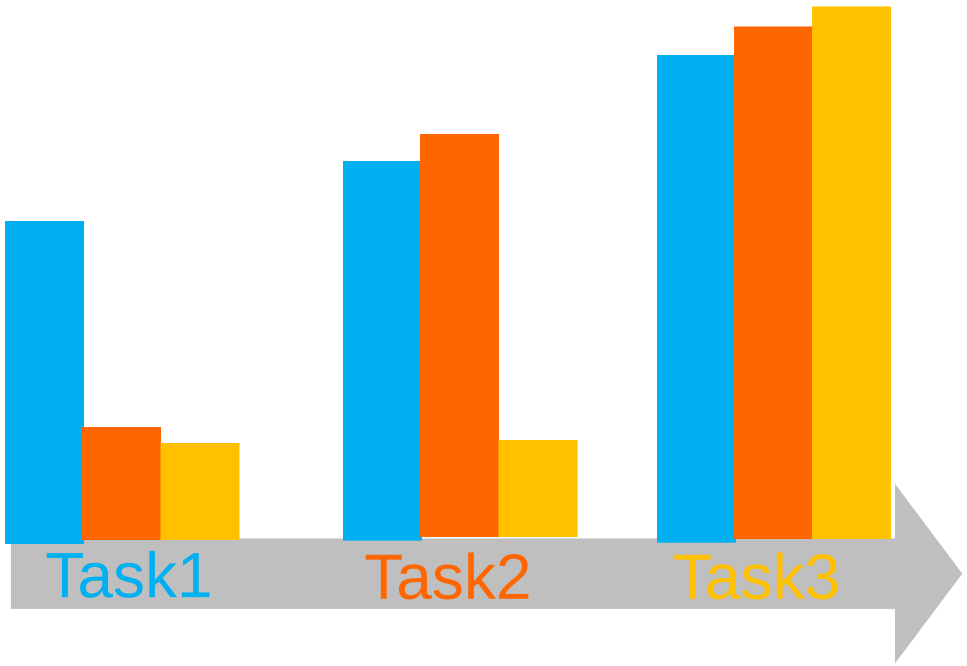


Possible Scenarios in CL

- Learned task performance is retained, and the new task is improved
- <u>Forward transfer</u>
  - The learned knowledge is transferrable to the new task

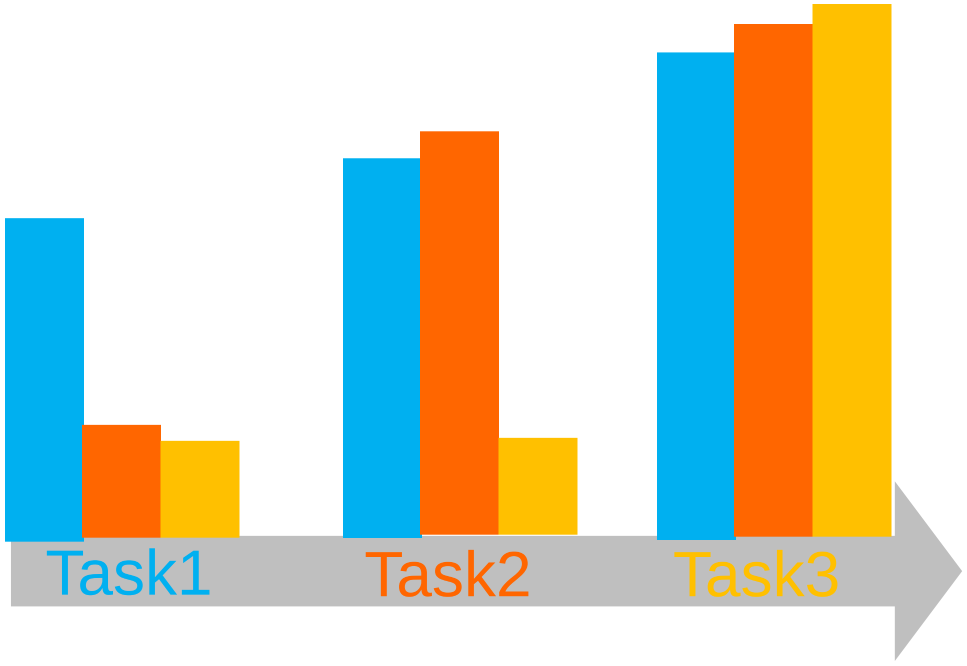


Possible Scenarios in CL



Possible Scenarios in CL

- Both learned task performance and new tasks are improved
- <u>Forward and backward transfer</u>
  - The learned knowledge is transferrable to the new task
  - The new task knowledge is also transferrable to old tasks



Possible Scenarios in CL

- We now know
  - What and why there is forgetting
  - Possible scenarios in CL and their underlying meaning
- But how do we quantize these possible scenarios in a paper?
  - Different metrics have been proposed
  - Before we go into detailed approaches, let us first introduce those evaluation metrics

Evaluation Metrics in CL

- Evaluation
  - Metrics
    - There are no commonly agreed metrics, but all metrics are designed to check which scenario the model output is in
      - In other words, metrics are to check whether there is forgetting or knowledge transfer
    - Next, we are going to see a table shows the “meta” statistics
      - Almost all metrics are derived from it

Evaluation Metrics in CL

- Evaluation Metrics
  - Meta Table

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

Testing task

Tasks trained so far

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

Evaluation Metrics in CL

Testing task

Tasks trained so far

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

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

If goes down -> forgetting
If goes up -> positive transfer

Y. Liu et al,. Mnemonics training: Multi-class incremental learning without forgetting. CVPR, 2020
Lopez-Paz and Ranzato, Gradient Episodic Memory for Continual Learning, NIPS 2017

Evaluation Metrics in CL

Testing task

Tasks trained so far

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

If goes down -> forgetting
If goes up -> positive transfer

Why?

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

Backward/Final results

Forward results

Evaluation Metrics in CL

Testing task

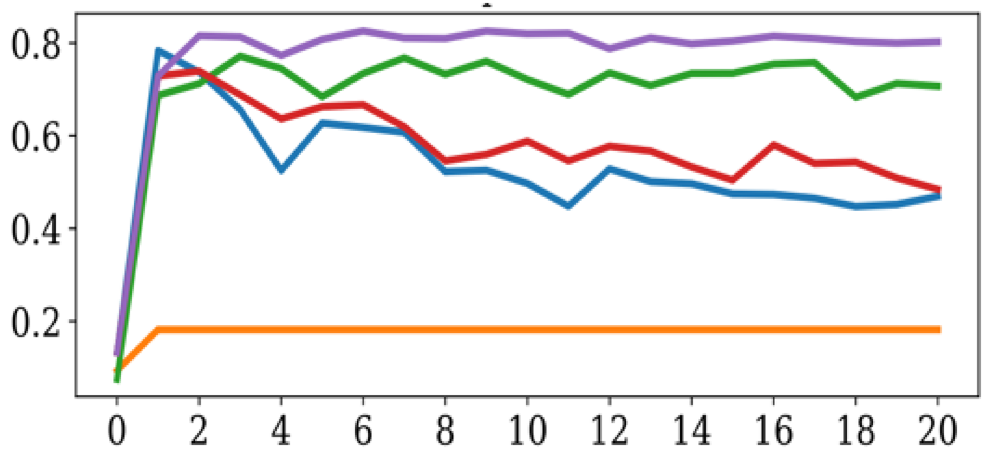
Tasks trained so far

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

If goes down -> forgetting

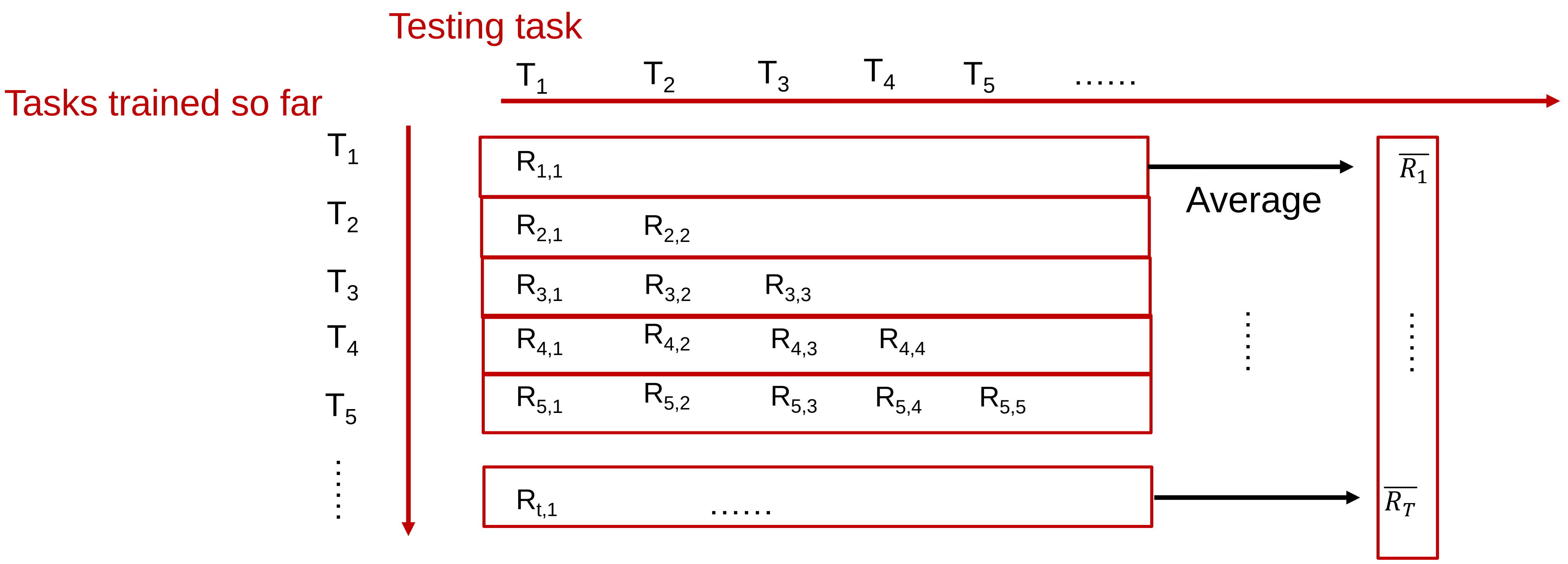
If goes up -> positive transfer

Sometimes people will draw a curve for that



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

Evaluation Metrics in CL



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

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

Evaluation Metrics in CL

Testing task

		T_1	T_2	T_3	T_4	T_5	......
		R_1	R_2	R_3	R_4	R_5	......
Tasks trained so far	T_1	$R_{1,1}$					
	T_2	$R_{2,1}$	$R_{2,2}$				
	T_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
	T_4	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
	T_5	$R_{5,1}$	$R_{5,2}$	$R_{5,3}$	$R_{5,4}$	$R_{5,5}$	
	......	$R_{t,1}$	......				

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

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

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

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

Evaluation Metrics in CL

Testing task

Tasks trained so far

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

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

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

Evaluation Metrics in CL

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

Possible Scenarios in CL

- We now know
  - What and why there is forgetting
  - Possible scenarios in CL and their underlying meaning
  - The quantity metrics and popular baselines
- Next
  - We will study some representative approaches in CL
- Keep in minds the goals
  - Prevent forgetting
  - Encourage forward and backward transfer (the last two scenarios)

Approaches

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

Replayed-based

- Where do the replayed samples come from?
  - Saved raw samples from each previous task
  - Generated pseudo-samples of previous tasks by a generator

- How to replay?
  - Input the replayed samples together with the current task samples
  - Use the replayed samples to constrain the optimization

Replayed-based: Real/Raw samples

- GEM[1]
  - Idea:
    - Store a small amount of data per task in an **explicit memory** (that's why it is called "replay-based" or "memory-based")
    - When making updates for the new task, adapt the previous knowledge to ensure the training will not **forget** the previous tasks
  - How to accomplish it?
    - The first item above is trivial, the key is the second one
    - Intuition: Since we have some previous task data, we can compute the loss for previous tasks. To avoid forgetting, we can simply **constrain the loss of previous tasks: avoid its increase but allowing its decrease**

[1]: Lopez-Paz and Ranzato, Gradient Episodic Memory for Continual Learning, NIPS 2017

Replayed-based: Real/Raw samples

- GEM
  - Constrain the loss:

learning predictor $y_t = f_\theta(x_t, z_t)$ memory: $\mathscr{M}_k$ for task z_k

For $t = 0,\ldots,T$

minimize $\mathscr{L}(\ f_\theta(\ \cdot\ , z_t)\ ,\ (x_t, y_t)\ )$

subject to $\mathscr{L}(\ f_\theta\ ,\ \mathscr{M}_k\ ) \leq \mathscr{L}(\ f_\theta^{t-1}\ ,\ \mathscr{M}_k\ )$ for all $z_k < z_t$

(i.e. s.t. loss on previous tasks doesn't get worse)

Assume local linearity:
$$\langle g_t, g_k \rangle := \left\langle \frac{\partial \mathscr{L}(\ f_\theta\ ,\ (x_t, y_t)\ )}{\partial \theta}, \frac{\mathscr{L}(\ f_\theta\ ,\ \mathscr{M}_k\ )}{\partial \theta} \right\rangle \geq 0 \quad \text{for all } z_k < z_t$$

Can formulate & solve as a QP.

Replayed-based: Real/Raw samples

- GEM
  - Datasets
    - The datasets below are popular benchmarks, and you will keep seeing them if you are working on continual learning
    - MNIST (binary images, 10 classes)
      - Hand-written digits (0-9) classification
      - Total Permutation: randomly permute the pixels to form a sequence of tasks
      - Rotate: randomly rotate (between 0 and 180 degrees) the images to form a sequence of tasks
    - CIFAR-100
      - 100 classes images
      - Typically, split based on classes (e.g., each task contain 5 classes) to form 20 tasks

Replayed-based: Real/Raw samples

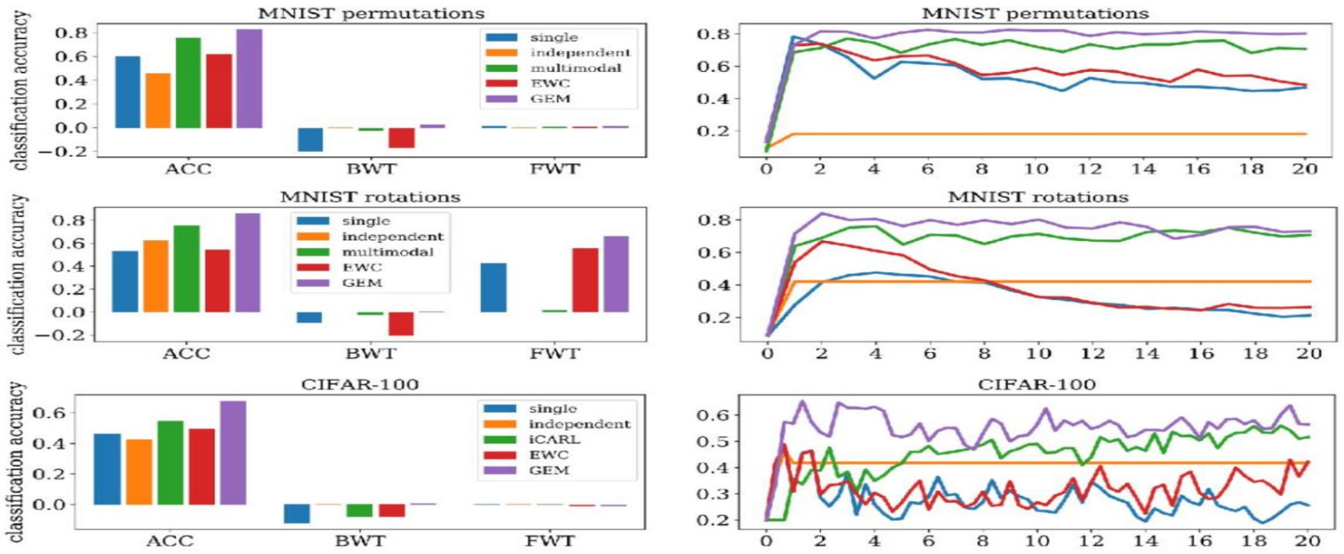
- GEM
  - Now, let’s try to read the results of GEM.

Experiments

Problems:
- MNIST permutations
- MNIST rotations
- CIFAR-100 (5 new classes/task)

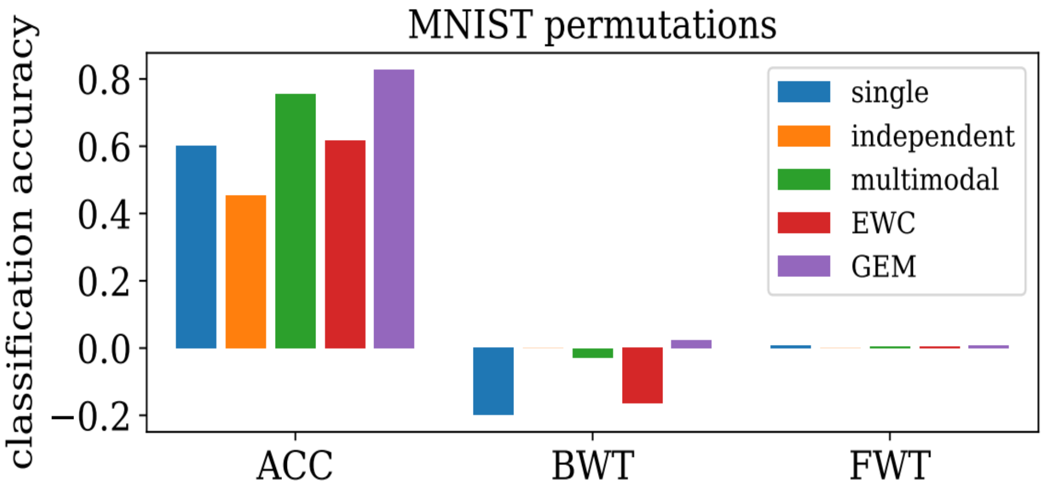
BWT: backward transfer,
FWT: forward transfer

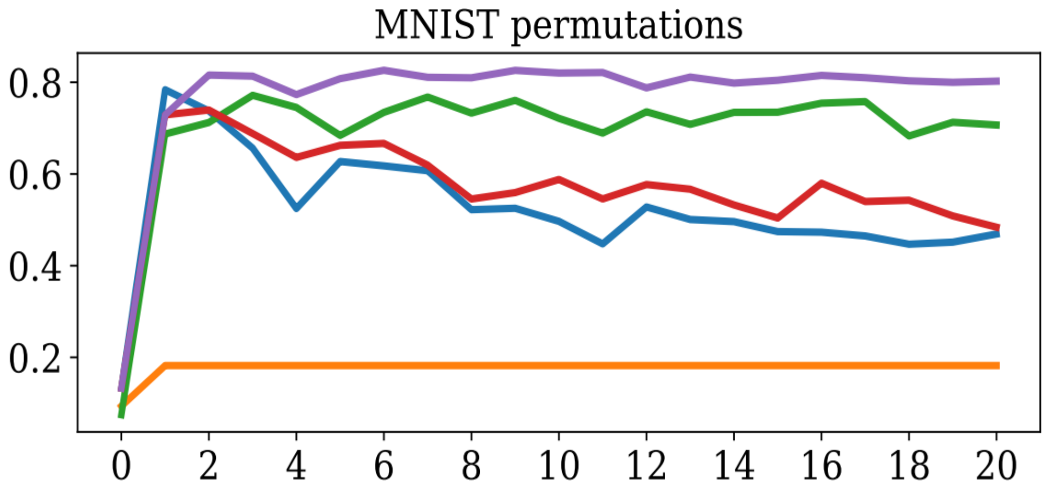
Total memory size:
5012 examples



Replayed-based: Real/Raw samples

- GEM





Replayed-based: Real/Raw samples

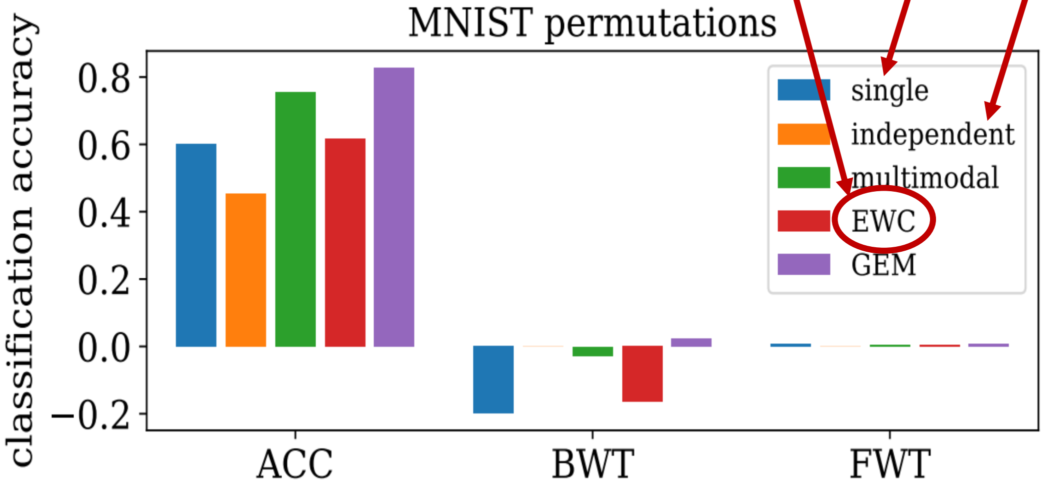
- GEM

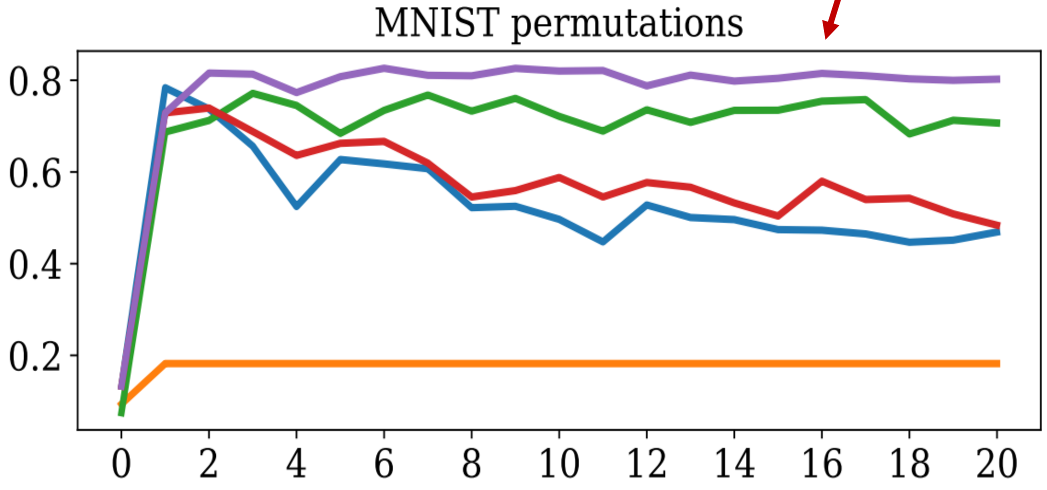
A baseline we will introduce later

NCL

ONE

Evolution of the first task





Replayed-based: Real/Raw samples

- GEM
  - How about pros and cons?
  - Pros:
    - Able to constrain the training using real data
    - No task information is needed in testing
  - Cons:
    - Inefficient because additional memory is needed to save previous task data
    - Easily over-fit the subset of stored samples because the replayed data is the only information the model has for previous tasks
      - Need to find more **representative** samples to do replay (many ways have been proposed)

Recall we have
- task-incremental
- class-incremental
- domain-incremental

Only the task-incremental has task information available in testing

Replayed-based: Real/Raw samples

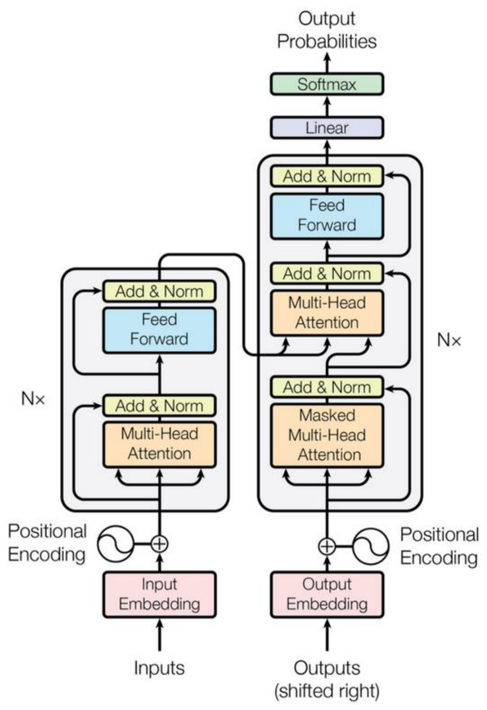
- GEM
  - Wow! We have learned our first CL model!
  - Kurt Lewin: “if you want truly understand something, try to change it”
  - Want to have some hands-on experience and write some codes?
    - Go to
      - https://github.com/ZixuanKe/PyContinual
      - You can run 20+ models and their variants (totally 40+) in a few lines
      - All models we discussed today are included
      - For example, you may change the *memory size, dataset, learning rate, backbone model* and see how these affect the performance

Replayed-based: Generated pseudo samples

- We have seen the case in the CV area, how about in NLP
  - Background
    - Pre-trained Language Model
      - What is a language model? $P(w_t \mid \text{context})\ \forall t \in V.$
        - A network with the distribution of words, given their contexts
        - It is naturally useful for language generation

Replayed-based: Generated pseudo samples

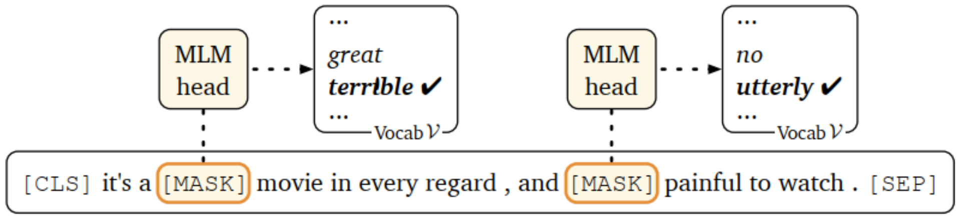
- We have seen the case in the CV area, how about in NLP
  - Background
    - Pre-trained Language Model
      - What is a language model (LM)?
      - Pre-trained language model
        - Recently, almost all NLP models are based on pre-trained LM
        - They are typically Transformer-based,
        - and (unsupervised) pre-trained using very large datasets (e.g. masked language model loss)
        - It has been shown that pre-trained LMs are very strong in transferring knowledge to different down-stream tasks

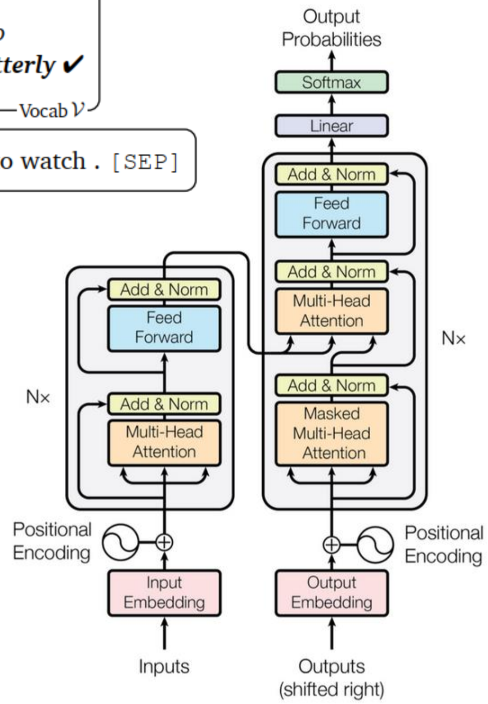


Replayed-based: Generated pseudo samples

- We have seen the case in CV area, how about in NLP
  - Background
    - Pre-trained Language Model
      - What is a language model (LM)?
      - Pre-trained language model
        - Recently, almost all NLP models are based on pre-trained LM
        - They are typically Transformer-based,
        - and (unsupervised) pre-trained using very large datasets (e.g. masked language model loss)
        - It has been shown that pre-trained LMs are very strong in transferring knowledge to different down-stream tasks

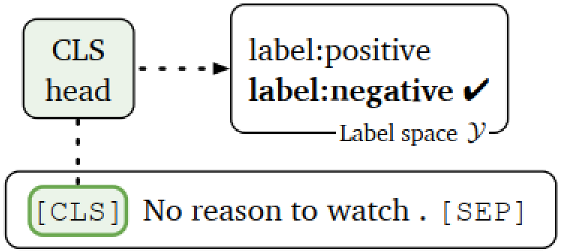
Head = Fully Connected Network





Replayed-based: Generated pseudo samples

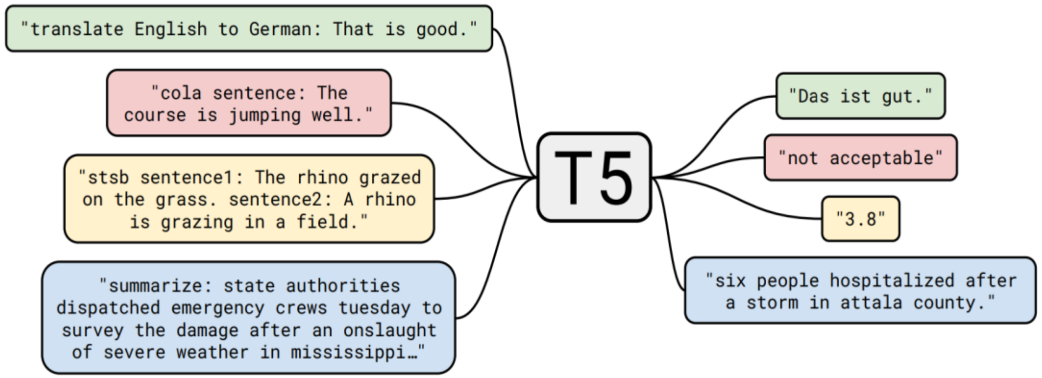
- We have seen the case in CV area, how about in NLP
  - Background
    - Pre-trained Language Model
      - Fine-tuning



        - Add classification head (depending on task, different head can be added), fine-tune both the LM and the head
        - What are the issues?
          - *This is not how the LM is pre-trained (MLM)*
          - *The model can only do one thing (e.g., classification), but the LM is naturally an auto-encoder (generative model)*
          - Training the LM may cause forgetting
            - we will see how to address this in the architecture-based approach

Replayed-based: Generated pseudo samples

- We have seen the case in CV area, how about in NLP
  - Background
    - Pre-trained Language Model
      - How to use
        - Fine-tuning, but there are some issues
        - How to address?
          - Manually use some words to convert the classification to language generation (becoming a popular way to interact with LM)
          - So, a unified template is used for different tasks
          - This is also attractive in CL, Why?



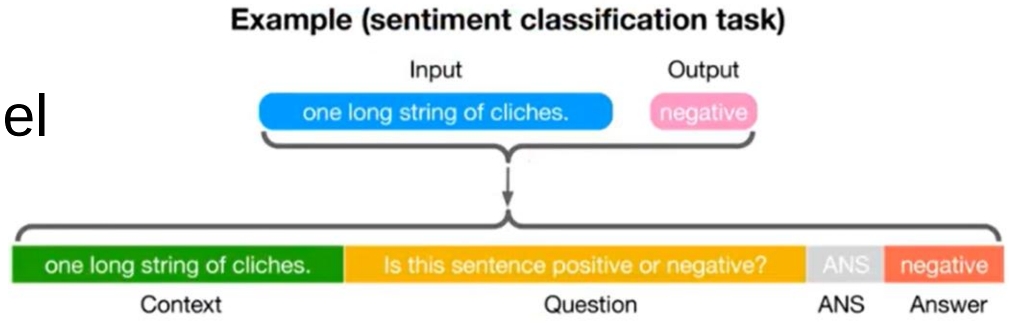
Replayed-based: Generated pseudo samples

- LAMOL[2]
  - Idea:
    - All tasks are formulated as a QA problem
      - In this way, the LM can be **both** a generator and a task solver
    - Before training the new task, the model first generates previous task data, and then input both the new and generated data
    - The model learns how to solve the new task and generate its data at the same time
  - How does it work?

[2]: Sun et al., LAMOL: LAnguage MOdeling for Lifelong Language Learning, ICLR 2020

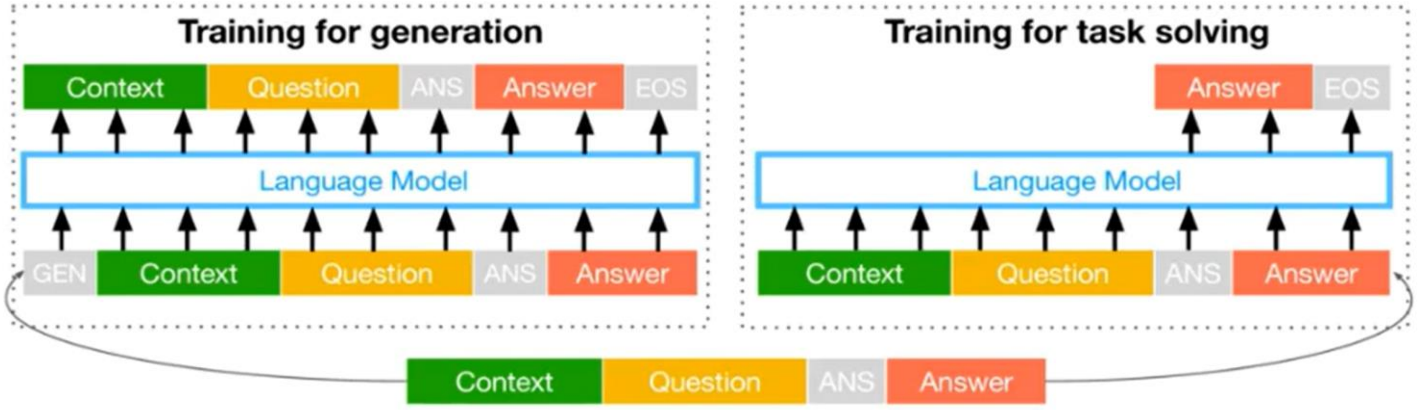
Replayed-based: Pseudo-generative samples

- LAMOL
  - Backbone Language Model
    - A language model itself is a generative model
    - Pre-trained Language Model: GPT-2
  - Data Formatting
    - It is becoming popular to use a unified task format so that one can better leverage the pre-trained LM.
    - This paper turns all tasks into QA (we show a sentiment task as an example).
  - Training
    - Train the LM as task data generator and task solver, simultaneously



Replayed-based: Pseudo-generative samples

- LAMOL
  - Training



- In this way, the resulting LM is not only able to solve the task but also to generate previous task data.
- When a new task arrives, the LM first generates pseudo-samples, and then combine both the generated and new samples for learning.

Replayed-based: Pseudo-generative samples

- LAMOL
  - How is it performing
    - In continual NLP, unlike continual CV, there are no commonly agree benchmarks. You will see different benchmarks are used in different papers
    - Datasets
      - 6 tasks. See the table
    - Metrics
      - Backward results

Task	Dataset	# Train	# Test	Metric
Question answering	SQuAD	87599	10570	nF1
Semantic parsing	WikiSQL	56355	15878	lfEM
Sentiment analysis	SST	6920	1821	EM
Semantic role labeling	QA-SRL	6414	2201	nF1
Goal-oriented dialogue	WOZ	2536	1646	dsEM
Text classification	AGNews Amazon DBPedia Yahoo Yelp	115000	7600	EM

Replayed-based: Pseudo-generative samples

- LAMOL
  - How is it performing

Methods	SST⇒SRL⇒WOZ	SST⇒WOZ⇒SRL	SRL⇒SST⇒WOZ	SRL⇒WOZ⇒SST	WOZ⇒SST⇒SRL	WOZ⇒SRL⇒SST	**Avg.**
Fine-tuned	50.2	24.7	62.9	31.3	32.8	33.9	39.3
EWC[2]	50.6	48.4	64.7	35.5	43.9	39.0	47.0
MAS[3]	36.5	45.3	56.6	31.0	49.7	30.8	41.6
GEM[4]	50.4	29.8	63.3	32.6	44.1	36.3	42.8
LAMOL	80	80.7	79.6	78.7	78.4	80.5	**79.7**
Multitasked			81.5				

Replayed-based: Pseudo-generative samples

- LAMOL
  - How about pros and cons?
  - Pros:
    - We can generate as many as examples we want
    - No task information is needed in testing
  - Cons:
    - How to train a good generator
      - Need high-quality and diverse data
      - Generator itself may be exposed to forgetting
      - The generated data may not be representative of real samples
    - *Like* the real-data replay, easily overfit to the generated samples
      - How to balance the generated samples and the real samples

Approaches

- Replayed-based
  - Use an explicit memory to maintain a subset of samples
  - We have seen 2 representative models and their pros and cons
  - Don't forget to play with them using the code base
    - https://github.com/ZixuanKe/PyContinual

- Next, what if we choose not to replay?

Approaches

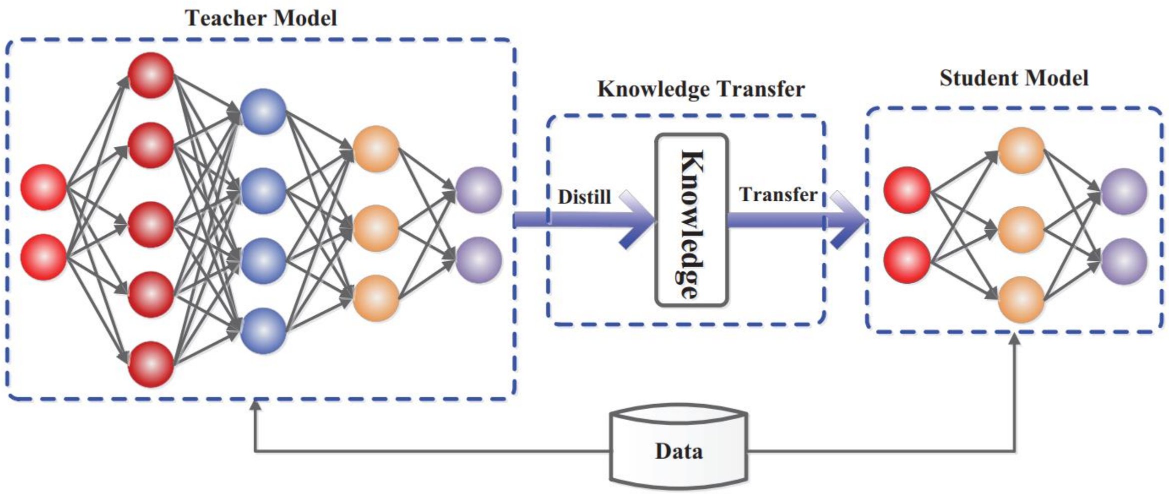
- Replayed-based
  - Use an explicit memory to maintain a subset of training samples, or
  - Learn a data generator
- Regularization-based
  - Add a regularization term to loss function
- Architecture-based
  - Each task use a different sub-network

Regularization-based

- **Regularization = loss + penalty term**
- Where does the penalty term come from?
  - Distillation
  - Prior based on (parameter) importance

Regularization-based: Distillation

- What is distillation?
  - Most popular distillation loss
    - KL-divergence
    - Mean Square Error (MSE)



- Why it is important for CL
  - We can distill knowledge from previous task model to current task model

Regularization-based: Distillation

- DER++[3]
  - Idea:
    - Distillation provides us a way to inject previous knowledge to the current model.
    - However, how to distillation itself is a research area
    - This paper finds a “good” way to distillate in the context of continual learning
  - How does it work?

[3]: Buzzega et al., Dark Experience for General Continual Learning: a Strong, Simple Baseline, NeurIPS 2020

Regularization-based: Distillation

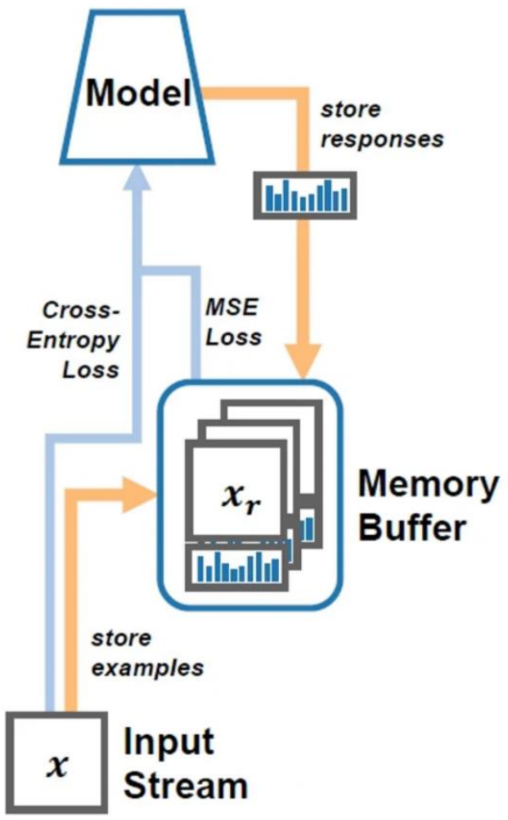
- DER++
  - Distillation:

We propose a GCL-compliant approach (**Dark Experience Replay, DER**) relying on *Dark Knowledge* [3] for retaining past experiences.

It maintains a subset of past network responses in a buffer $\mathcal{B}$ (handled via *reservoir* sampling); then, in addition to the loss of the current task $\mathcal{L}_{t_c}$, DER minimizes the L^2 distances between past and current outputs for buffer datapoints:

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

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



Regularization-based: Distillation

- DER++
  - Distillation:

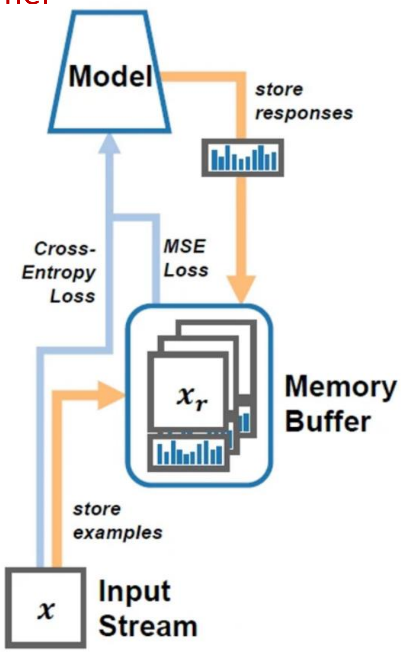
Not the logits!
Response is the representation before classifier

We propose a GCL-compliant approach (**Dark Experience Replay, DER**) relying on *Dark Knowledge* [3] for retaining past experiences.

It maintains a subset of past network responses in a buffer $\mathcal{B}$ (handled via *reservoir* sampling); then, in addition to the loss of the current task $\mathcal{L}_{t_c}$, DER minimizes the L^2 distances between past and current outputs for buffer datapoints:

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

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

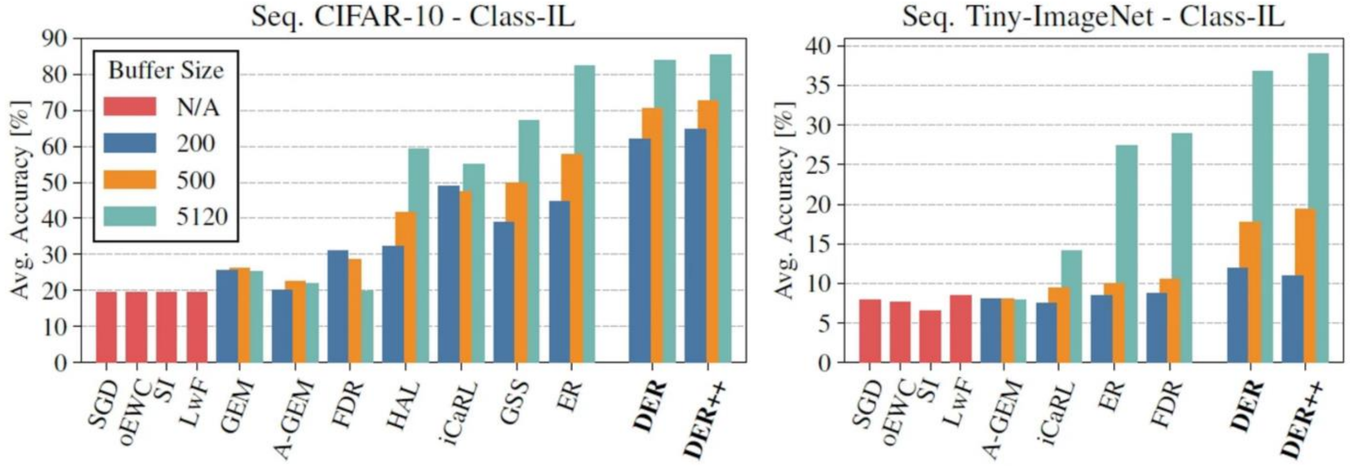


Penalty term comes from distillation loss

Regularization-based: Distillation

- DER++
  - How is it performing:

ER consistently outperforms other CL approaches in literature. DER and DER++ outperform even ER across all experimental settings.

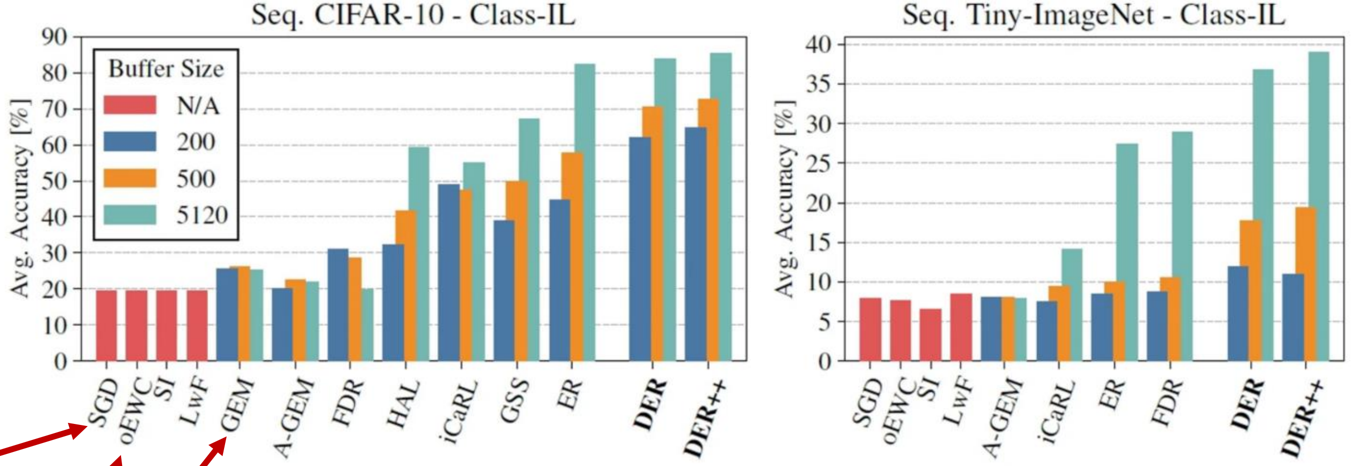


Results on many other benchmarks are available in the paper.

Regularization-based: Distillation

- DER++
  - How is it performing:

ER consistently outperforms other CL approaches in literature. DER and DER++ outperform even ER across all experimental settings.



NCL

Results on many other benchmarks are available in the paper.

We already know that

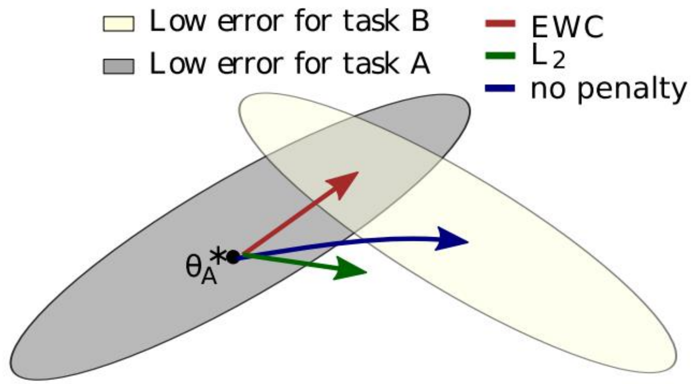
Regularization-based: Distillation

- DER++
  - How is it performing
    - Datasets
      - CIFAR, Tiny ImageNet, MNIST
    - Metrics
      - Backward results
      - Buffer size, training time

Regularization-based: Distillation

- DER++
  - How about pros and cons:
  - Pros
    - Efficient and network independent
    - No task information is needed in testing
  - Cons
    - How to nest distill knowledge is an open question
    - Distillation is not perfect. There will be knowledge loss
    - How to balance the previous knowledge and current knowledge
    - Not much control, only in the penalty term

Regularization-based: Prior



- EWC[4]
  - Idea:
    - Probably the most well-known continual learning system (not the best performing though)
    - Instead of distillation, it computes prior based on the parameter importance to constrain the new task updating
      - So that, those parameters that are important to previous tasks are not changed much
  - How does it work?
    - How to estimate the importance of parameters
    - How to design a penalty term from parameters importance

[4]: Kirkpatrick et al., Overcoming catastrophic forgetting in neural networks, PNAS 2017

Regularization-based: Prior

- EWC
  - Parameter importance
    - The key is the magnitude of gradient.
      - If the gradient is high, the corresponding parameter is more important
      - This intuition is also seen in other areas, for example in the neuron network pruning research, though they have very different goals
    - More details
      - *After* training a task, EWC inputs the data again and compute average magnitude of gradient (a.k.a., fisher matrix)
      - The fisher matrix is then used to constrain the updating in learning the new task

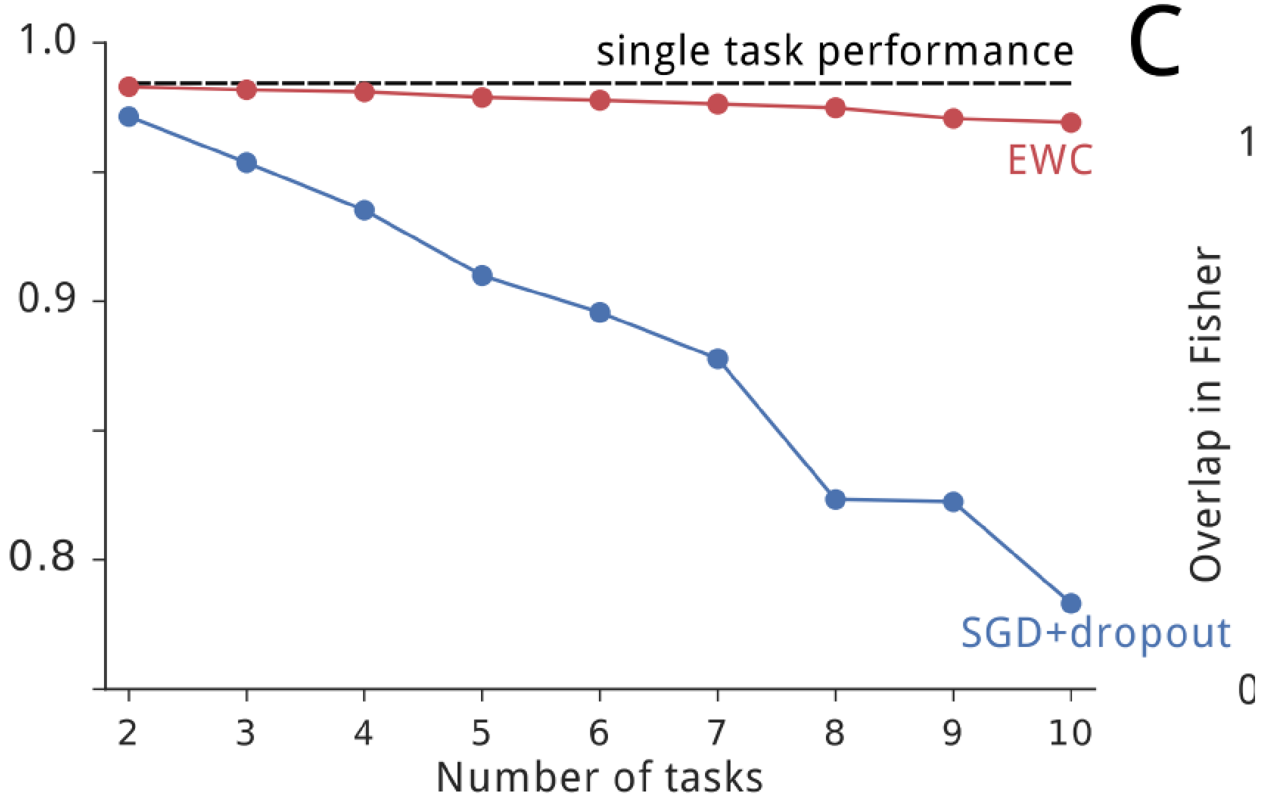
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta^*_{A,i})^2$$

Regularization-based: Prior

- EWC
  - How is it performing?
    - Datasets
      - MNIST, Atari (reinforcement learning)
    - Metrics
      - Progressive results

Regularization-based: Prior

- EWC
  - How is it performing?



Regularization-based: Prior

- EWC
  - How about pros and cons?
  - Pros
    - Efficient and network independent
    - No task information is needed in testing
  - Cons
    - Soft penalty is usually not sufficient to constrain the optimization to overcome forgetting