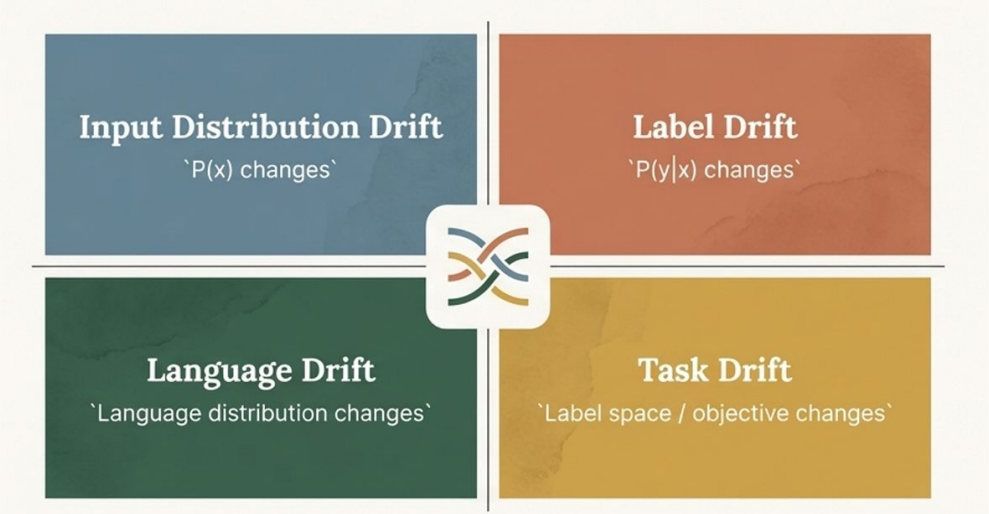
Continual Learning In Natural Language Processing

Shrey Satapara
Researcher - II, AI Lab
Fujitsu Research of India

Language Doesn't Stand Still

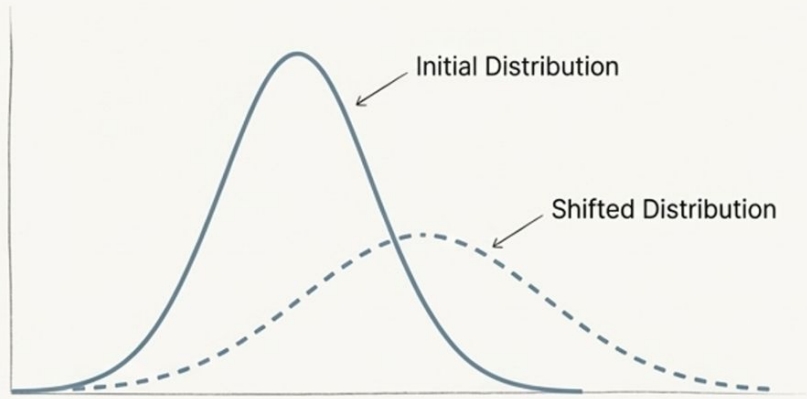
Why even the best NLP models have a hidden expiration date.


Input Distribution Drift
`P(x) changes`
Label Drift
`P(y|x) changes`
Language Drift
`Language distribution changes`
Task Drift
`Label space / objective changes`

Dimension 1: Input Distribution Drift

When the world outside the model changes.

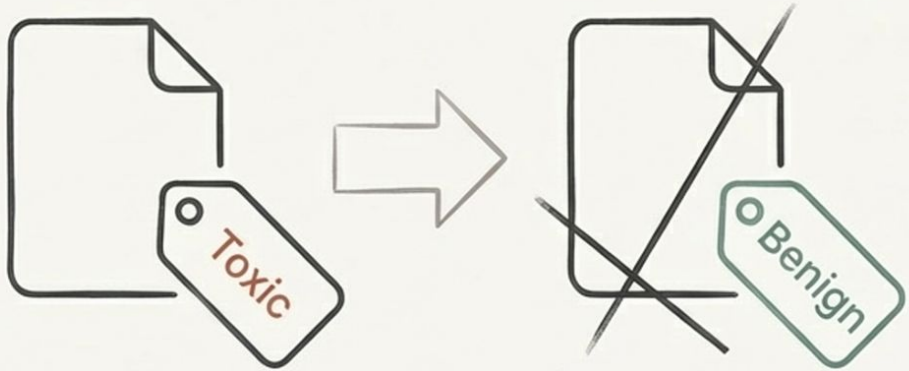
The fundamental nature, topics, and patterns of the input data ($P(x)$) change, even if the meaning and the task remain the same. The model begins to see data it wasn't trained to expect.



Dimension 2: Label Drift

When the meaning of “correct” evolves.

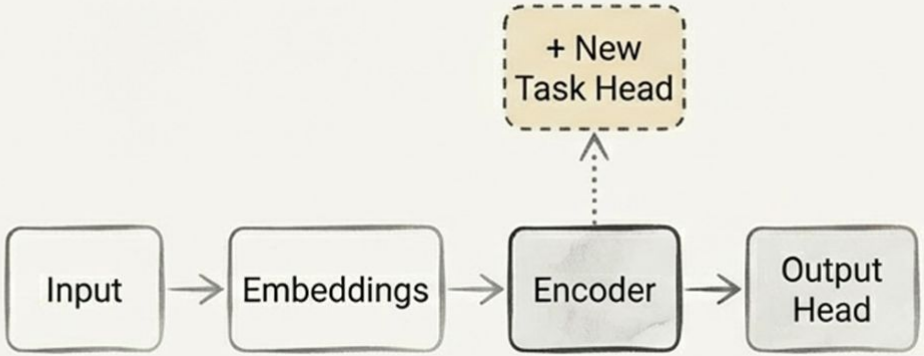
The underlying semantics of the labels change. For the *exact same* input x, the “correct” label y changes over time ($P(y|x)$ changes). This is often driven by evolving human understanding or policy.

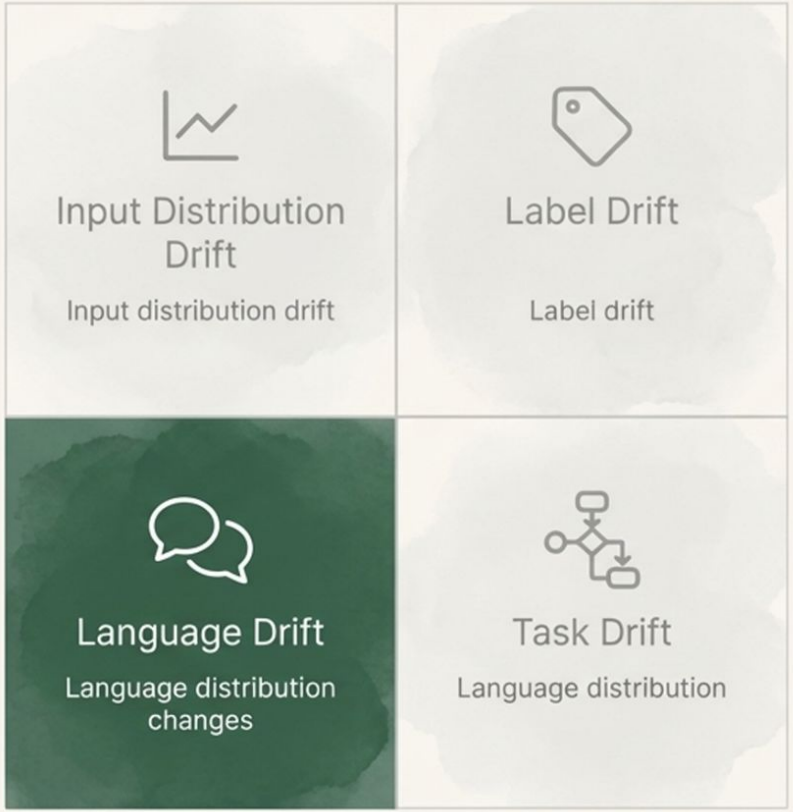


Dimension 3: Task Drift

When the problem the model solves is redefined.

The label space or the fundamental objective of the model changes. This can involve adding or removing classes, or even adding new prediction heads for multi-task learning.





Dimension 4: Language Drift

When the language itself expands or mixes.

The distribution of languages or dialects being processed by the system changes significantly. This is distinct from Input Drift because it alters the fundamental token space and linguistic structures.

Adapting to a Changing World: Input & Label Drift

INPUT DRIFT

The Problem

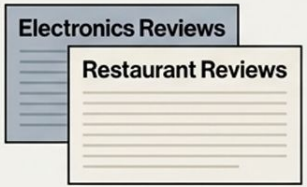
The distribution of text changes. New slang, jargon, topics, and entities appear (e.g., IT tickets vs. HR tickets).

The Playbook: Domain-Incremental CL

What Changes: Input distribution P(x) – style, topic, entities.

What Stays Fixed: The core task and label semantics.

Our Assistant's Journey



The model, initially trained on **electronics reviews**, must now adapt to handle the different language and topics of **restaurant reviews**.

LABEL DRIFT

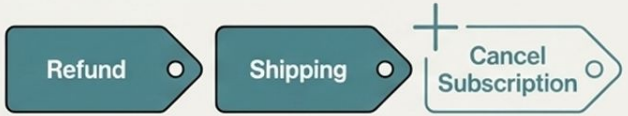
The Problem

The set of valid labels or classes expands over time. The task is the same, but the ontology grows.

The Playbook: Label-Space Growth (Class-Incremental)

What Changes: The label set itself (new intents, new entity types).
What Stays Fixed: The overall task definition ('classify intent,' 'tag entities').

Our Assistant's Journey



A new product launch requires the assistant to recognize new user intents like **'cancel subscription'** and **'report fraud'** without forgetting old ones.

Expanding Capabilities: Task & Language Drift

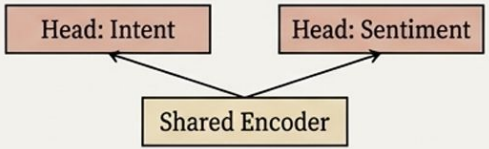
TASK DRIFT

The Problem: The system needs to perform entirely new jobs. Adding new capabilities risks 'catastrophic forgetting' of old ones.

The Playbook: Task-Incremental CL

What Changes: The objective and output space (a new task head).
What Stays Fixed: The identity of each task is known; old tasks must be retained.

Our Assistant's Journey



After mastering intent classification, we now add a new head for **sentiment analysis and escalation prediction** to the same system.

LANGUAGE DRIFT

The Problem: The system must be deployed in new regions, requiring it to handle new languages, scripts, and cultural contexts.

The Playbook: Language-Incremental CL

What Changes: Language distribution, tokenization, morphology.
What Stays Fixed: The task definition and label schema are shared across languages.

Our Assistant's Journey

The successful English-language assistant is now rolled out globally, adding support for **Hindi** and then **Gujarati**, including handling code-mixing ('Hinglish').

It Can Be Even More Interesting

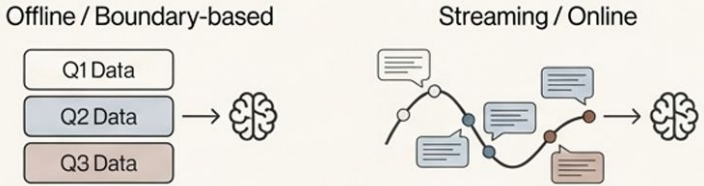
The Realistic Scenario: Task + Language Incremental

In reality, these drifts don't happen in isolation. You often add new tasks *and* new languages over time, compounding the risk of interference and multi-dimensional forgetting.

The challenge: Interference is now multi-dimensional. Tasks interfere with tasks, languages with languages, and task-specific patterns can differ by language.

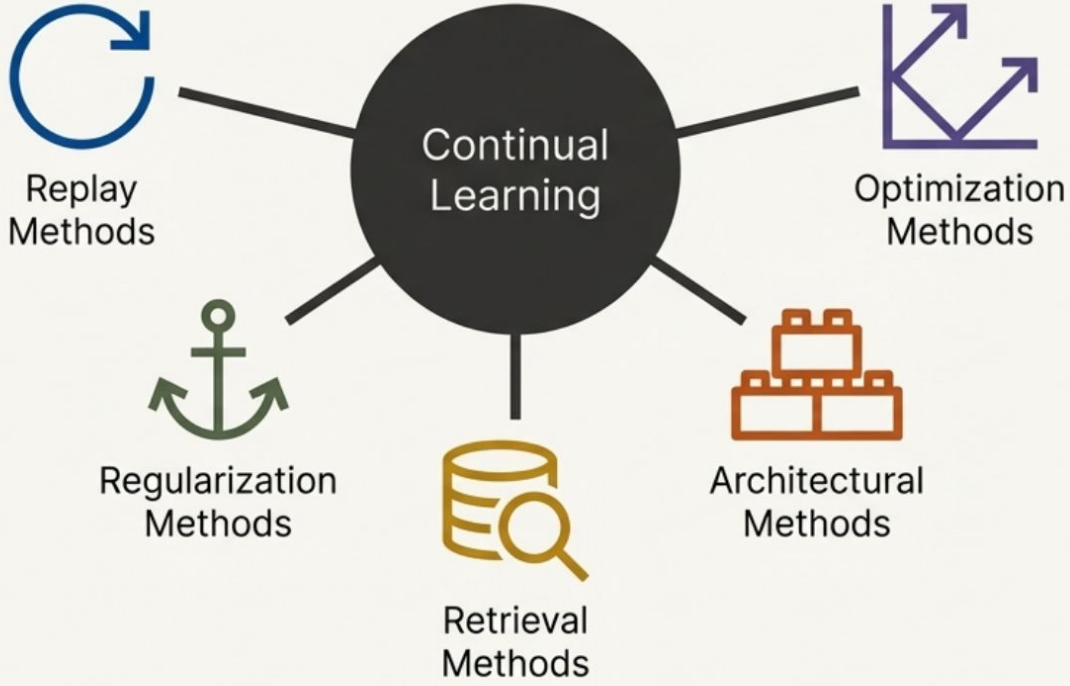
The Critical Constraint: Streaming vs. Offline Data

This isn't a drift type, but a fundamental constraint on *how* you learn.



Why It Matters in NLP

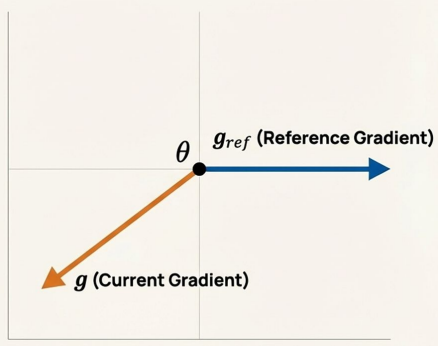
Text streams are highly non-stationary. A news event can cause a sudden distribution shift. The order of data is critical, and evaluation must adapt (e.g., prequential evaluation).



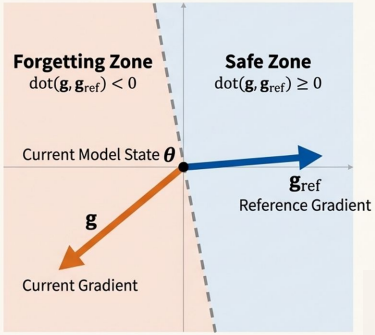
We can categorize the vast landscape of CL methods into five fundamental strategic approaches to preserving knowledge.

A-GEM: Average Gradient Episodic Memory

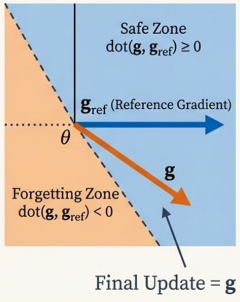
A New Philosophy: From Rehearsal to Constraint



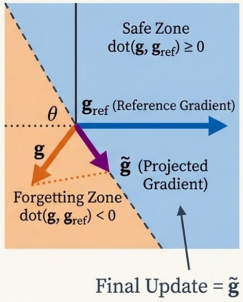
Step 1. Defining the direction



Step 2. The "Safety Check"



Safe Update



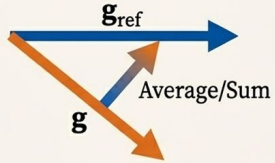
Corrected Update

Objective: $\text{minimize } \|\tilde{\mathbf{g}} - \mathbf{g}\|^2$

Constraint: $\langle \tilde{\mathbf{g}}, \mathbf{g}_{\text{ref}} \rangle \geq 0$

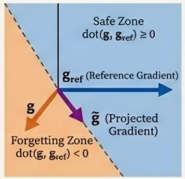
Final Update Rule: $\theta \leftarrow \theta - \eta\, \tilde{\mathbf{g}}$

Experience Replay (ER)



- **Mechanism: Mixing.** Combines mini-batches of new data and stored data.
- **Effect:** The final update is an *average* of the new gradient (**g**) and the past gradient ($\mathbf{g}_{\text{ref}}$).
- **Computational Cost:** Requires a full backward pass on the memory samples to compute their gradients.

A-GEM



- **Mechanism: Constraint.** Uses memory only to check for conflicts.
- **Effect:** Modifies the new gradient **g** *only if* it conflicts with $\mathbf{g}_{\text{ref}}$. Otherwise, it's unchanged.
- **Computational Cost:** Lighter. Only requires a forward pass and gradient computation on memory samples (no weight updates).

Efficient Lifelong Learning with A-GEM, Chaudhry et al 2019 ICLR

IDBR: Information Disentanglement based Regularization

Reply past examples with retaining general language knowledge.

- Rehearsing past data is a common solution but random sample selection is inefficient and incomplete.
- Focusing exclusively on downstream task can cause the general language features from pretraining to degrade.

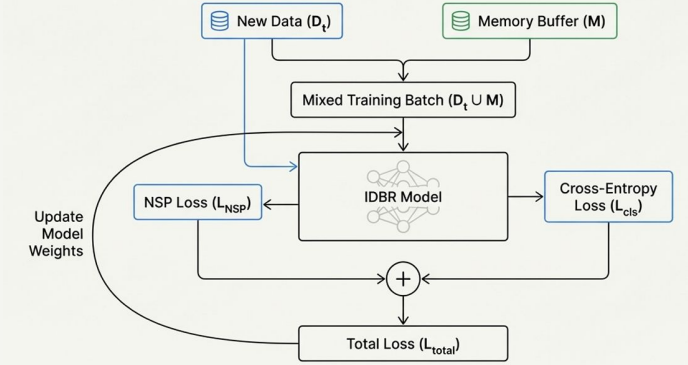
Two-pronged strategy: Smarter memory reply with general language knowledge regularization.

1: Find prototypes that best represents its distribution.

- K-means clustering to select a compact and diverse set of 'prototype' samples.
- The buffer is updated with new centroids and budget selection from the previous memory.

2: General language knowledge regularization

- The aux. Next sentence prediction (NSP) task to preserve the model's core language understanding by adding NSP objective to training process.
- The NSP loss is calculated using only the new task data.



$$L_{total} = L_{cls}(D_t \cup M) + \lambda L_{NSP}(D_t)$$

1. Ablation: Centroid vs. Random Buffer?

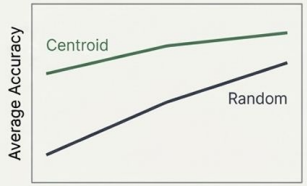
Result: IDBR with centroid selection consistently outperforms standard experience replay with a random buffer, especially at smaller memory sizes. This confirms the value of Pillar 1.

2. Component Analysis: What is the impact of removing the NSP objective?

Result: Removing the L_NSP term (setting λ=0) causes a significant drop in performance, demonstrating the crucial stabilizing value of Pillar 2.

3. Sensitivity Analysis: How does performance scale with memory size?

Result: The performance gap between centroid and random selection is largest with very small memory budgets, highlighting IDBR's efficiency.



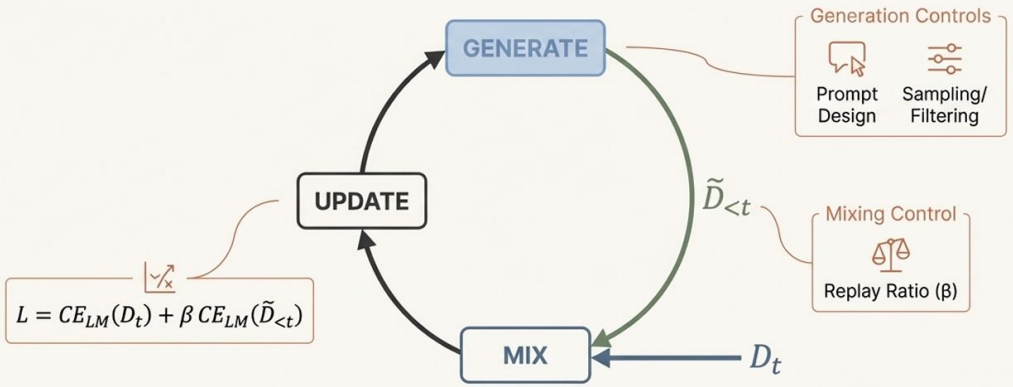
Continual Learning for Text Classification with Information Disentanglement Based Regularization, Huang et al 2021, NAACL

LAMOL: Everything is generation

Treat both problem-solving and memory recall as generation.

- In replay based method there are two drawbacks: Storage and Privacy Concerns.

What if model could generate its own practice data instead of storing it?



Process Description:

Before learning a new task t, LAMOL is prompted to generate pseudo-samples for all previously seen tasks $k < t$.

The Role of Prompts:

- Prompts are simple instructions or IDs that tell the model *which* task to generate data for.
- This makes the process **task-prompted** and targeted, unlike buffer-based methods.

LAMOL: LANGUAGE MODELING FOR LIFELONG LANGUAGE LEARNING, Sun et al 2020, ICLR

PCLL: Prompt-Conditioned Latent Replay

Decoupled Architecture for Controllable Replay

- Early methods of generative replay often relied on sampling directly from a language model that served both as task-solver and data generator, which offered limited control over the distribution and characteristics of the replayed data.

Core Principle:

- A dedicated CVAE acts as a latent generator to create synthetic data.
- The main solver model trains on mix of real data of current task and synthetic data from the CVAE.

The Objective Function

$$L_{total} = L_{task}(D_t \cup \tilde{D}_{(<t)}) + \alpha L_{CVAE}$$

The primary **Task Loss**. This measures the solver's performance on the current task (D_t) and the replayed pseudo-data ($\tilde{D}_{(<t)}$). The goal is to minimise this loss.

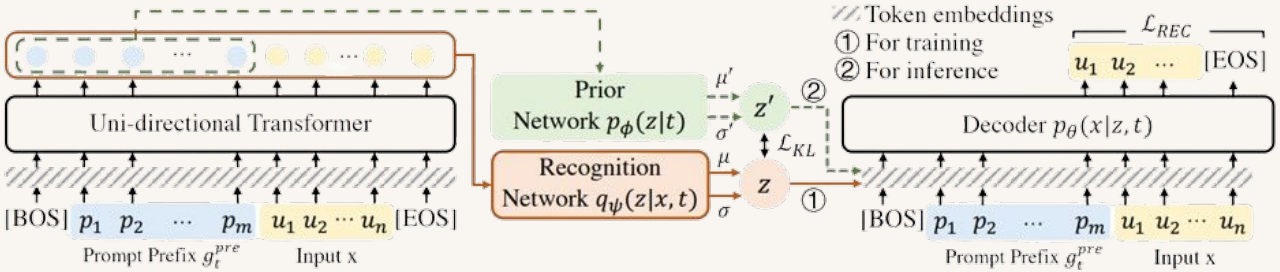
The **Generator Training Loss**. This term ensures the CVAE generator remains effective. It is weighted by a hyperparameter, α.

A Closer Look at the CVAE Loss (ELBO):

$$L_{CVAE} = \mathbb{E}[-\log p(x|z,c)] + KL(q(z|x,c) \,||\, p(z|c))$$

Reconstruction Term: Ensures that the generated data is a faithful reconstruction of past data.

KL Divergence: Acts as a regulariser, ensuring the latent space (z) remains well-structured and smooth.



Prompt Conditioned VAE: Enhancing Generative Replay for Lifelong Learning in Task-Oriented Dialogue, Zhao et al 2022, EMNLP

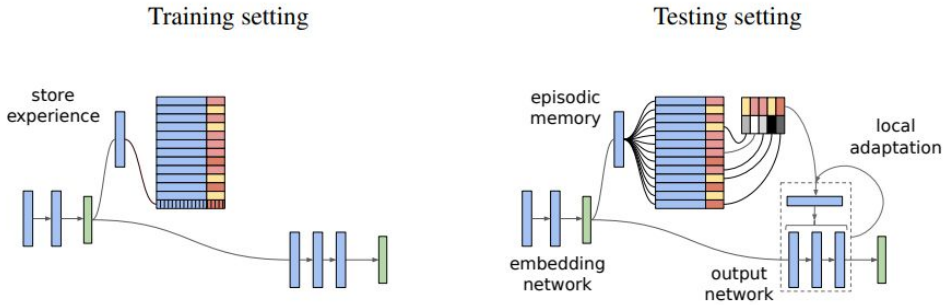
MBPA: Memory Based Parameter Adaption

- augment a base network with key–value episodic memory, where keys are embeddings and values are labels.
- retrieve **KNN** by Euclidean distance
- **Train-time:** standard parametric training; **store** embedded examples in memory (no local adaptation during training in the basic formulation)

$$\theta_x = \arg\max_{\theta_x} \log p(\theta_x \mid \theta) + \sum_{k=1}^{K} w_k^{(x)} \log p\left(v_k^{(x)} \mid h_k^{(x)}, \theta_x, x\right)$$

$$\log p(\theta_x \mid \theta) \propto -\frac{\|\theta_x - \theta\|_2^2}{2\alpha_M}$$

- **Practical adaptation:** do a **few gradient steps** to get ΔM(x,θ) and predict with adapted params; **discard** adaptation after the prediction (base θ unchanged).
- **Why it helps CL:** if a past task/domain is “forgotten”, a few steps on **relevant neighbors** can recover performance without task IDs



Algorithm 1 Model-based Parameter Adaptation

procedure MBPA-TRAIN
Sample mini-batch of training examples $B = \{(x_b, y_b)\}_b$ from training data.
Calculate the embedded mini-batch $B' = \{(f_\gamma(x_b), y_b) : x_b, y_b \in B\}$.
Update θ, γ by maximising the likelihood (1) of θ and γ with respect to mini-batch B
Add the embedded mini-batch examples B' to memory M: $M \leftarrow M \cup B'$.

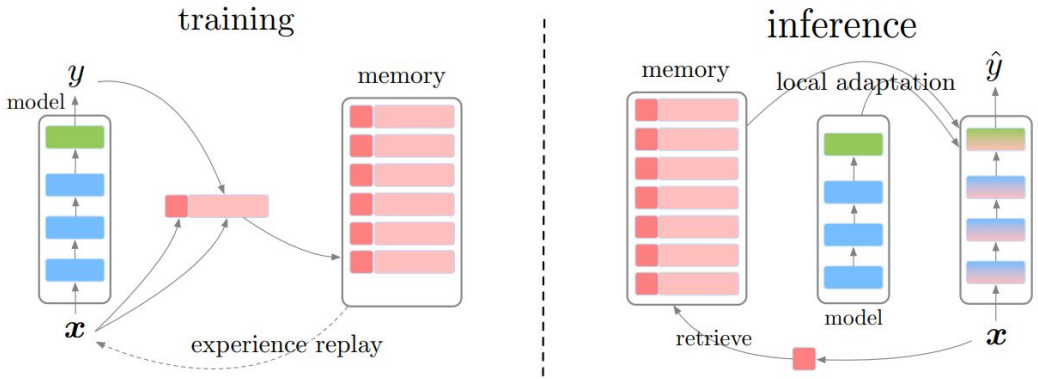
procedure MBPA-TEST(test input: x, output prediction: $\hat{y}$)
Calculate embedding $q = f_\gamma(x)$, and $\Delta_{\text{total}} \leftarrow 0$.
Retrieve K-nearest neighbours to q and producing context, $C = \{(h_k^{(x)}, v_k^{(x)}, w_k^{(x)})\}_{k=1}^K$.
for each step of MbPA **do**
Calculate $\Delta_M(x, \theta + \Delta_{\text{total}})$ according to (4)
$\Delta_{\text{total}} \leftarrow \Delta_{\text{total}} + \Delta_M(x)$.
Output prediction $\hat{y} = g_{\theta + \Delta_{\text{total}}}(h)$

MEMORY-BASED PARAMETER ADAPTATION, Sprechmann et al 2018, ICLR

MBPA++

- Two complementary mechanisms:
  - Sparse experience replay from memory during training, and
  - Local adaptation at inference using KNN neighbors.
- Key practical stabilizer: use a fixed key network (to avoid representation drift); they report trainable key network hurts, so fixed keys are crucial

In one line: MbPA++ is better because it combines **global stability (sparse replay)** with **local plasticity (test-time adaptation)**, and makes retrieval more reliable via **fixed keys**.



Episodic Memory in Lifelong Language Learning, d'Autume et al 2019, NIPS

RMR DSE

Regularized Memory Recall and Domain Shift Estimation

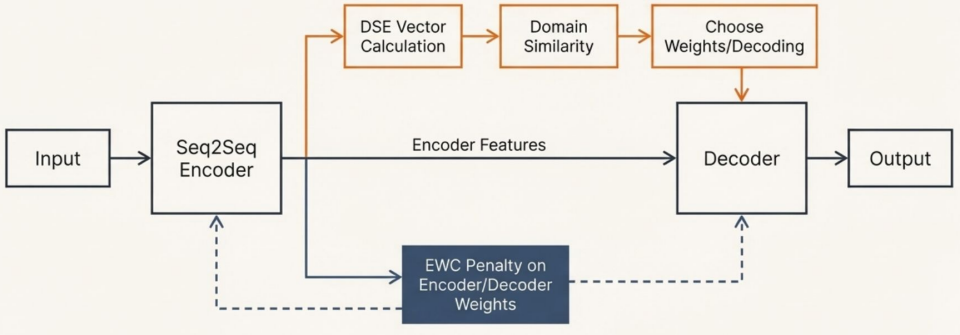
- RMR: learn new domain t while keeping parameters close to the previous model θ* (no old data required).
- DSE: after training on new domain t encoder embeddings for old-domain inputs drift; DSE estimates this drift and subtracts it to reduce forgetting.

Domain Shift Process:

- Train a new model
- Compute embeddings for current-domain data with both models.
- Compute per-sample drift vectors
- Cluster the old-model embeddings
- For points near each center, compute a mean-shift weight using a Gaussian kernel
- Compute one domain-shift vector per cluster

Inference on an old domain: subtract the shift

- If the domain is multiple steps old, Apply sequential correction by subtracting the chain of stored shifts across steps



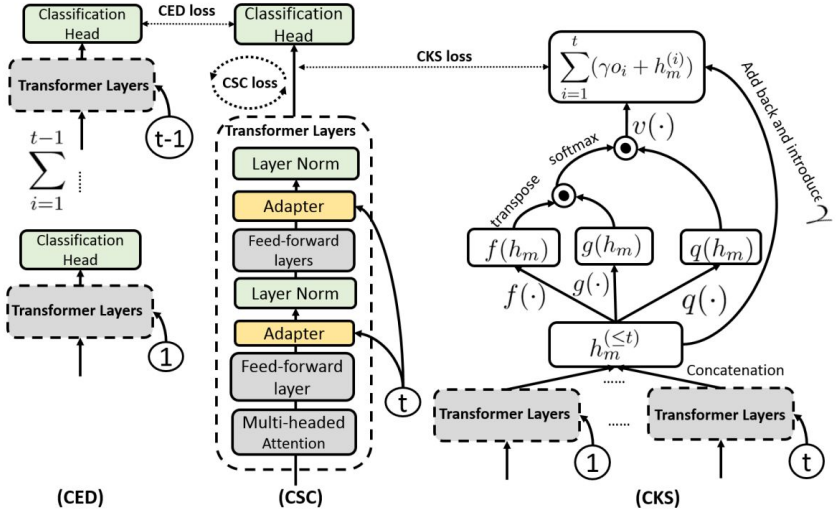
$$\mathcal{L}_t = \lambda(\tau)\, L_t(\theta) \;+\; (1-\lambda(\tau))\,\gamma F \sum_{ij} \Pi_{ij}\,(\theta_{ij} - \theta^*_{ij})^2$$

Overcoming Catastrophic Forgetting During Domain Adaptation of Seq2seq Language Generation, Ke et al 2022, NAACL

CLASSIC:

Core Principle:

- Combines task masking/isolation with contrastive representation learning.
- Construct an augmented view incorporating past-task information; pull current reps toward it.
- Contrastive: positives=(current rep, past-informed rep); negatives=other samples.
- Diff vs EWC: constraints in representation space (alignment), not parameter space.
- Eval: forgetting + representation stability; ablate contrastive weight and view construction.



- **CSC:** $\text{SupCon}(h_{cur}, h_{cur}; y)$
- **CKS:** $h_{cks} = \text{AttnMix}(\{f(x; \text{mask}_i)\}_{i \le t})$
  $\text{SupCon}(h_{cks}, h_{cur}; y)$
- **CED (for each old $i < t$):**
  teacher $z_i = \text{head}(f(x; \text{mask}_i))$ on **same** x
  pos: $(z_i[n], z_{cur}[n])$; **neg:** $(z_i[n], z_{cur}[j \neq n])$

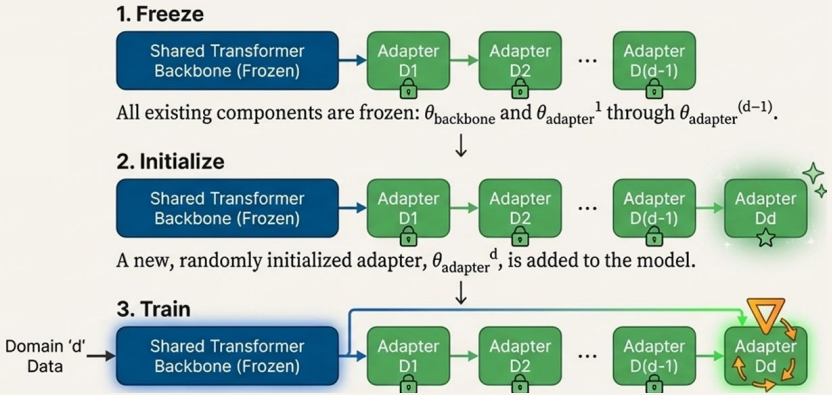
for task $t = 1..T$

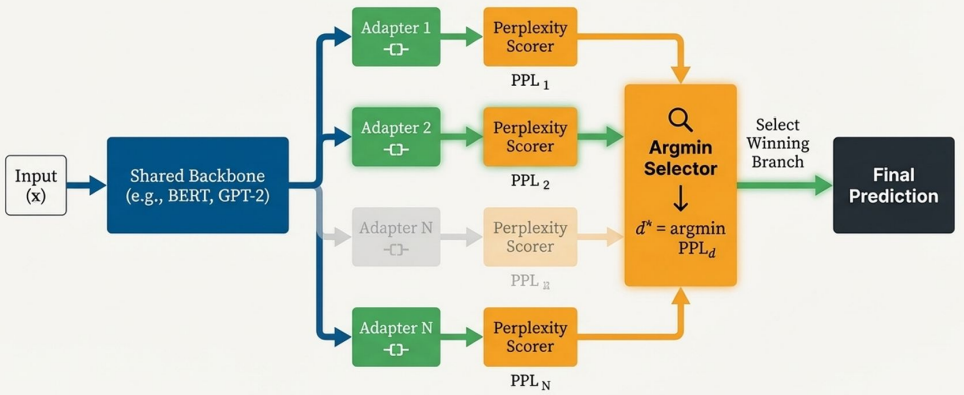
- for batch $B = \{(x_n, y_n)\}_{n=1..N}$
  - $h_{cur} = f(x; \text{mask}_t)$, $z_{cur} = \text{head}(h_{cur})$
  - $L = CE + \lambda_1 CSC + \lambda_2 CED + \lambda_3 CKS$
  - backprop → **gradient mask (old neurons)** → step
- save mask_t

CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks, Ke et al 2021

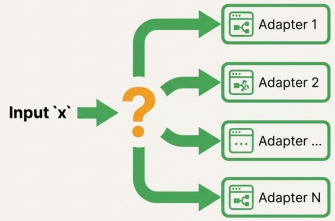
AdapterCL

Instead of retraining entire model, freeze the large pretrained model, and attach a task specific adapter





Key Formula Callout:
Objective: Minimize Cross-Entropy (CE) on domain 'd' data.
Update Rule: Only $\theta_{adapter}^{d}$ is updated.



Key Formula Spotlight:

$$d^*(x) = \underset{d}{\text{argmin}}\, \text{PPL}(x; \theta_{\text{backbone}}, \theta^{d}_{\text{adapter}})$$

Select domain 'd' that minimizes perplexity for input 'x'.

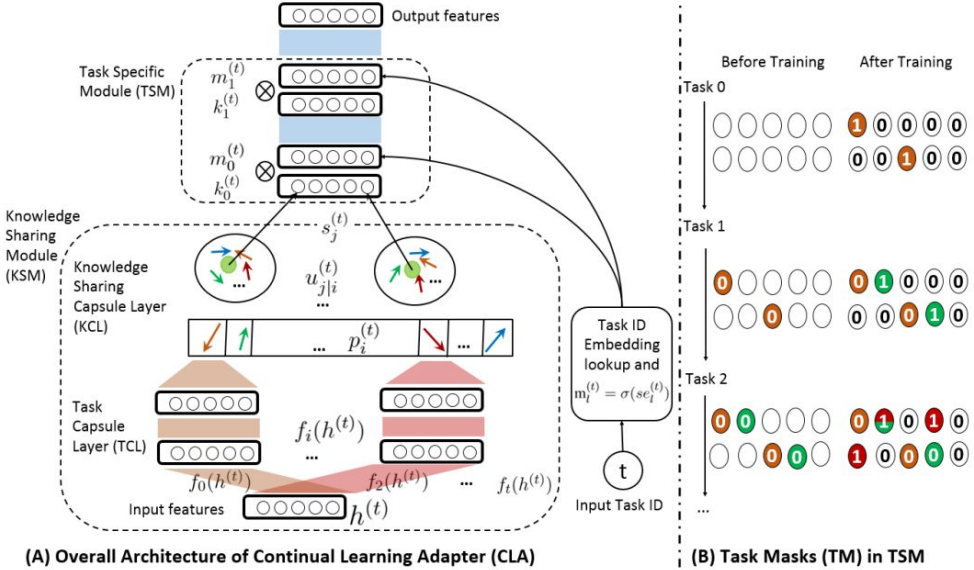
				INTENT	DST	NLG	
Method	**+Param.**	**Mem.**	**Hours↓**	*Accuracy↑*	*JGA↑*	*EER↓*	*BLEU↑*
VANILLA	-	$\emptyset$	0.21 ± 0.02	4.1 ± 1.4	4.91 ± 4.5	48.7 ± 3.9	6.38 ± 0.6
L2	$\lvert\theta\rvert$	$\emptyset$	0.56 ± 0.06	3.8 ± 1.4	3.81 ± 3.4	55.7 ± 7.1	5.4 ± 0.9
EWC	$2\lvert\theta\rvert$	$\emptyset$	0.91 ± 0.10	3.9 ± 1.3	5.22 ± 4.5	58.2 ± 3.7	5.06 ± 0.5
AGEM	-	$t\lvert\mathcal{M}\rvert$	0.38 ± 0.04	34.0 ± 6.4	6.37 ± 4.0	62.1 ± 6.9	4.54 ± 0.6
LAMOL	-	$\emptyset$	2.32 ± 1.24	7.5 ± 6.4	4.55 ± 3.5	66.1 ± 6.9	3.0 ± 0.9
REPLAY	-	$t\lvert\mathcal{M}\rvert$	0.62 ± 0.23	81.1 ± 1.4	30.33 ± 1.2	$\mathbf{17.8} \pm 0.9$	$\mathbf{17.4} \pm 0.7$
ADAPT	$t\lvert\mu\rvert$	$\emptyset$	$\mathbf{0.20} \pm 0.02$	$\mathbf{90.5} \pm 0.6$	$\mathbf{35.1} \pm 0.5$	31.78 ± 1.3	16.76 ± 0.4
MULTI	-	-	4.14 ± 2.23	95.5 ± 0.1	48.9 ± 0.2	12.56 ± 0.2	23.61 ± 0.1

Continual Learning in Task-Oriented Dialogue Systems, Madotto et al 2021, EMNLP

B-CL: BERT with Continual Learning Adapter

- Hard isolation of parameters effectively prevents forgetting but can inhibit positive transfer.

- Key Idea: Replace standard adapters with a Continual Learning Adapter (CLA) having two complementary components:
  - KSM (Knowledge Sharing Module): capsule + dynamic routing decides which previous tasks are most similar → selectively shares knowledge (better forward/backward transfer).
  - TSM (Task-Specific Module): task masks learn per-task neuron usage; when learning new tasks, block gradients for neurons used by older tasks → protects past knowledge.
- KSM promotes selective sharing across related tasks; TSM enforces parameter isolation to prevent overwriting.



Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks, Ke et al 2021, NAACL

CTR: Capsules and Transfer Routing for continual learning

Use Dynamic binary gates to decide whether to share or isolate parameters based on task similarity.

- While soft sharing in B-CL enables knowledge transfer, but since there is a soft gating is via capsule layer, tasks can still interfere into each other when routing isn't perfectly selective leading to forgetting
- Replaces dynamic routing with a dedicated Transfer Routing mechanism:

1) Knowledge Sharing Module (KSM): transfer via capsules + "transfer routing"

KSM explicitly tries to **pull useful representations from prior tasks** when tasks are similar.

Task capsule layer (TK-layer): add one capsule per task; each capsule is a small MLP:

$$p_i^{(t)} = f_i(h^{(t)}) \quad (1)$$

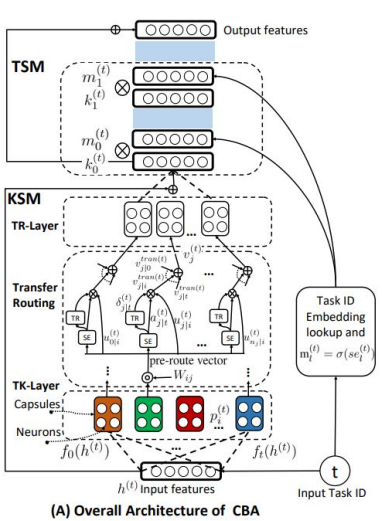
Transfer routing: for each task-capsule output, build "pre-route vectors":

$$u_{j|i}^{(t)} = W_{ij} p_i^{(t)} \quad (2)$$

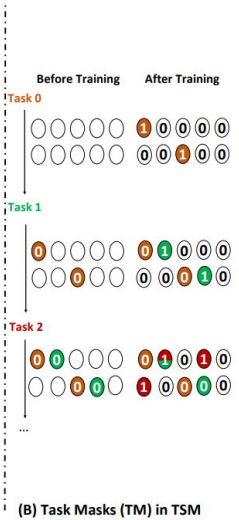
Then compute **task similarity** using convolution + pooling:

$$q_{j|t}^{(t)} = \text{MaxPool}(\text{ReLU}(u_{j|t}^{(t)} * W_q + b_q)) \quad (3)$$

$$a_{j|i}^{(t)} = \text{MaxPool}(\text{ReLU}(u_{j|i}^{(t)} * W_a + f_a(q_{j|t}^{(t)}) + b_a))$$



(A) Overall Architecture of CBA



(B) Task Masks (TM) in TSM

Then a **Task Router (TR)** makes a **binary connect/disconnect decision** (differentiable via Gumbel-Softmax):

$$\delta_{j|i}^{(t)} = \text{Gumbel\_softmax}(a_{j|i}^{(t)} * W_\delta + b_\delta) \quad (5)$$

Finally, only "connected" prior tasks contribute to the transferred representation:

$$v_{j|i}^{tran(t)} = a_{j|i}^{(t)} \otimes u_{j|i}^{(t)}, \quad v_j^{(t)} = \sum_{i:\delta_{ij}^{(t)}=1} v_{j|i}^{tran(t)} \quad (6)$$

Achieving : Forgetting : Prevention and Knowledge Transfer in Continual Learning , Ke et al 2021, NIPS

ACM: Adaptive Compositional Models

Core Idea: ACM augments a base LM with task-specific adapters, but critically, it also use gatting/mixing mechanism to reuse prior adapters when learning a new, similar task.

Candidates per layer l: existing modules $t \in \{1, \ldots, K\}$ + new candidate $t = K + 1$. Each module provides two adapters at layer l: $(\mathrm{MLP}_{MH}^{t,l}, \mathrm{MLP}_{FF}^{t,l})$.

Soft selection via hidden-state mixing (decision-stage only): $\lambda_{t,l} = \mathrm{softmax}(c_{t,l}) = \frac{e^{c_{t,l}}}{\sum_{j-1}^{K+1} e^{c_{j,l}}}$.

Mixed adapter outputs: $h_{mh}^{l} = \sum_{t=1}^{K+1} \lambda_{t,l} \, \mathrm{MLP}_{MH}^{t,l}(o_{mh}^{l})$, $h_{ff}^{l} = \sum_{t=1}^{K+1} \lambda_{t,l} \, \mathrm{MLP}_{FF}^{t,l}(o_{ff}^{l})$. Then standard Add&Norm.

What is trained: only **new candidate adapters** $(t = K + 1)$ + logits $c_{t,l}$. Backbone + old modules $t \leq K$ are **frozen**.

Make selection "sharp" using entropy regularizer: $L_{\text{entropy}} = \gamma \sum_{l} \sum_{t=1}^{K+1} -\lambda_{t,l} \log \lambda_{t,l}$ (minimizing → lower entropy → peaky weights).

Bias toward reuse at init: set old logits $c_{t,l} = +c$ (for $t \leq K$), new $c_{K+1,l} = -c$, so old modules start preferred unless new helps.

Final hard decision per layer: $t^{*}(l) = \arg\max_{t \in \{1,\ldots,K+1\}} \lambda_{t,l}$. If $t^{*}(l) \leq K$: reuse module $t^{*}(l)$ at layer l. If $t^{*}(l) = K + 1$: keep new module at layer l.

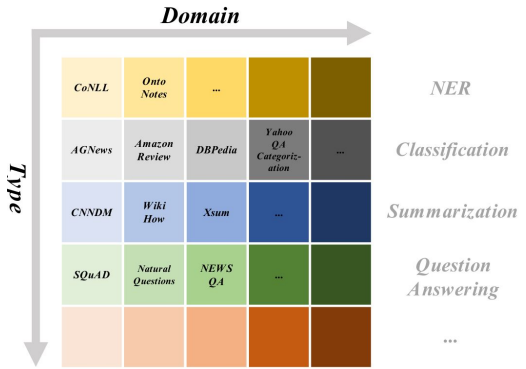
ACM Decision Stage (Adapter Selection) Layer-wise Reuse vs Add

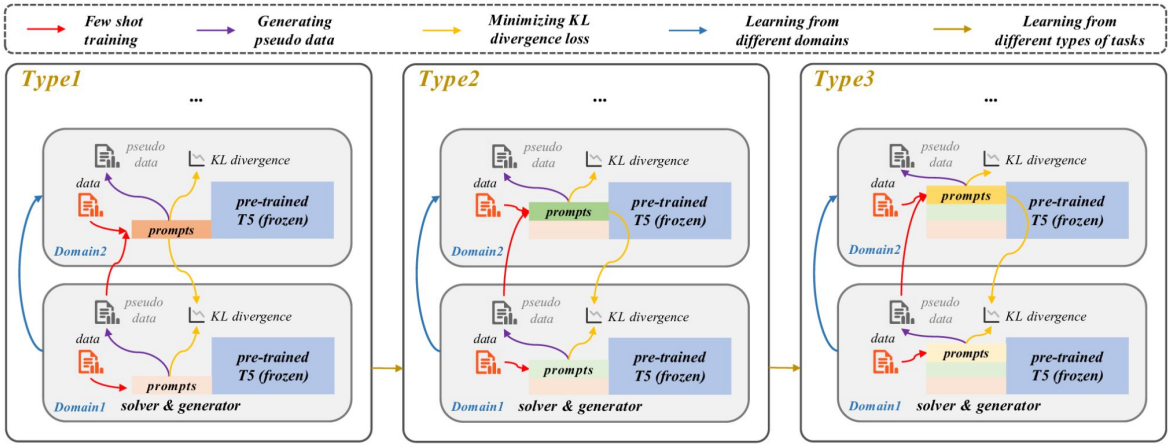
Once the per-layer choices are made, the architecture for that task is fixed, and they train it using **pseudo experience replay** (from LAMOL-style generation)

Continual Sequence Generation with Adaptive Compositional Modules, Zhang et al 2022, ACL

LFPT5

generates pseudo (labeled) samples of previously learned domains, and later gets trained on those samples to alleviate forgetting of previous knowledge as it learns the new domain





$$\mathcal{L}_{\phi}^{\text{lm}} = -\sum_{i=1}^{n} \log p([X_i, Y_i]|[G, P], \phi, \theta)$$

$$\mathcal{L}_{\phi}^{\text{KL}} = \sum_{i=1}^{m}\sum_{j=1}^{t} D_{\text{KL}}(p_j(\mathcal{V}|[P, \tilde{X}_i], \phi', \theta)||p_j(\mathcal{V}|[P, \tilde{X}_i], \phi, \theta))$$

$$\mathcal{L}_{\phi} = \mathcal{L}_{\phi}^{\text{task}} + \lambda_{\text{lm}}\mathcal{L}_{\phi}^{\text{lm}} + \lambda_{\text{kl}}\mathcal{L}_{\phi}^{\text{KL}}$$

- LFPT5 with Forward Knowledge transfer by initializing prompt of new task using previous task

LFPT5: A UNIFIED FRAMEWORK FOR LIFELONG FEW-SHOT LANGUAGE LEARNING BASED ON PROMPT TUNING OF T5, Qin et al 2022, ICLR

ProgressivePrompts

For each new task, append and train a new specialized prompt while keeping the previous prompts frozen.

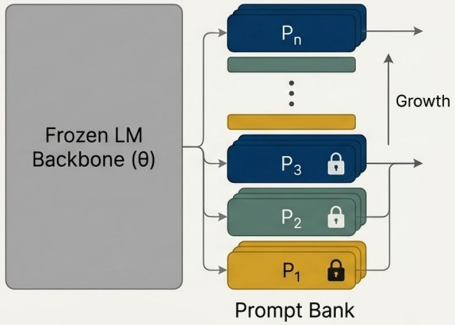
- Previous methods like L2P reuses prompts from a shared pool of fixed size. Due to fix capacity it can lead to performance degradation for large no. of tasks.

FROZEN CORE: Leverage a powerful, pre-trained LM as a stable, generalist knowledge base that is never modified.

EXPANSION OVER MODIFICATION: For each new skill, add a new, task-specific prompt rather than altering existing parameters.

GUARANTEED STABILITY: This "expand-only" architecture completely prevents catastrophic forgetting by design.

The Fundamental Trade-off: It trades ever-growing parameter counts for perfect memory of past tasks.



- Directly optimizing prompt parameters can lead to training instability, while reparameterizing prompt embeddings matrix through a multi-layer perceptron (MLP) can improve performance.

Method	DR	Order 4	5	6	7	avg
Finetune†		14.8	27.8	26.7	4.5	18.4
Replay†	✓	67.2	64.7	64.7	44.6	57.8
A-GEM†	✓	70.6	65.9	67.5	63.6	66.9
MBPA++†	✓	70.8	70.9	70.2	70.7	70.6
IDBR‡	✓	75.9	76.2	76.4	76.7	76.3
ProgPrompt*		**78.0**	**77.7**	**77.9**	**77.9**	**77.9**
Per-task Finetune		73.9	73.9	73.9	73.9	73.9

(b) Results with BERT-base.

Method	DR	Order 1	2	3	avg
Finetune◇		18.9	24.9	41.7	28.5
Replay	✓	35.4	37.1	41.5	38.0
EWC◇		39.0	38.0	44.8	40.6
LFPT5*◇	✓	47.6	52.6	57.9	52.7
ProgPrompt*		**75.2**	**75.0**	**75.1**	**75.1**
Per-task Finetune		70.0	70.0	70.0	70.0

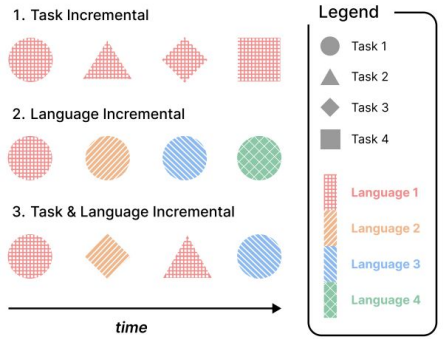
(a) Results with T5-large.

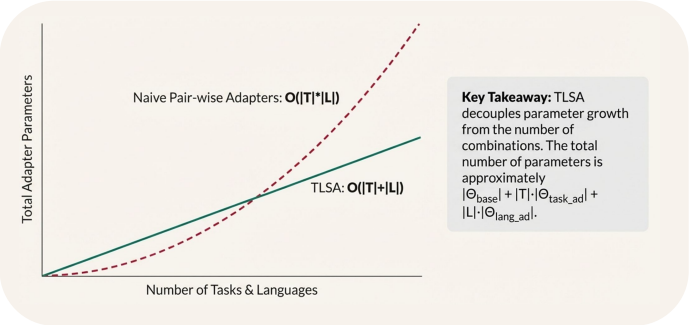
PROGRESSIVE PROMPTS: CONTINUAL LEARNING FOR LANGUAGE MODELS, Razdaibiedina et al 2023, ICLR

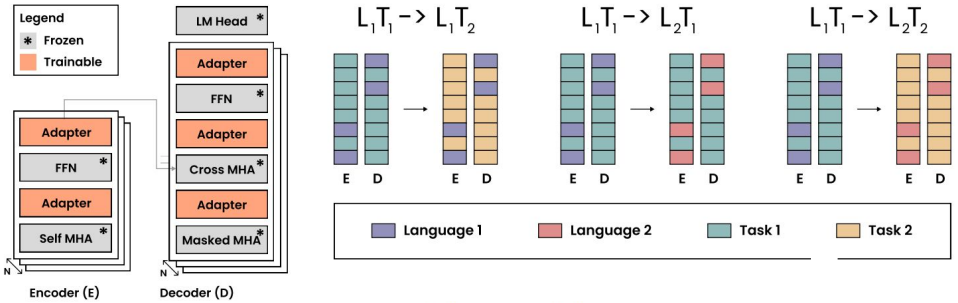
TLSA: Task And Language Specific Adapters

The core idea of the approach is the segregation of task-specific and language-specific information through separate adapters.

- The same task-specific adapters are shared across all languages for a particular task, enabling cross lingual transfer between tasks.
- The same language-specific adapters are shared across all tasks for each language, enabling cross-task knowledge transfer.
- To prevent forgetting related to previous task or previous language EWC regularization is applied.







$$\underset{\theta_t,\theta_l}{\operatorname{argmin}}(\mathcal{L}(Y^{t,l}, F(X^{t,l}, \theta_b, \theta_t, \theta_l)$$

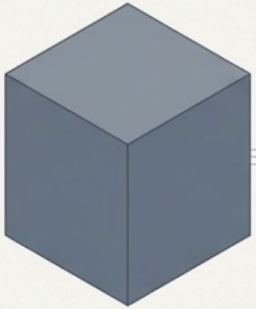
$$+\alpha[\mathbb{1}_{t\in S_{tasks}} R(\theta_t, \theta_t^*) + \mathbb{1}_{l\in S_{langs}} R(\theta_l, \theta_l^*)])$$

$$PercentLoss_{t,l} = 100 \times \left(\frac{UB_{t,l} - R^{k'}_{t,l}}{UB_{t,l}}\right)$$

TL-CL: Task And Language Incremental Continual Learning, Satapara et al 2024, EMNLP

What Changes When We Move from SLM to LLMs

The SLM Paradigm: A Contained Problem



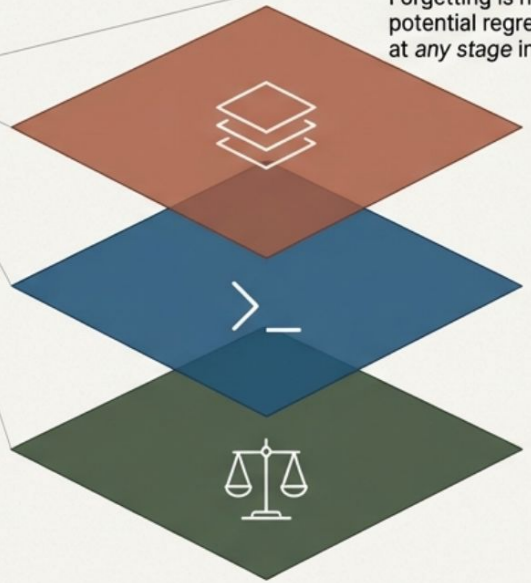
Forgetting = Previous-Task Regression

- In Small Language Models, Catastrophic Forgetting (CL) is a known issue that occurs *post-pretraining*.
- It manifests during supervised fine-tuning over a sequence of distinct tasks or domains.
- Performance is measured by a drop in accuracy (Acc), F1-score, or Exact Match (EM) on earlier tasks after learning new ones.

In LLMs, forgetting evolves from *task regression* to systemic *behavior regression*.

The LLM Reality: A Lifecycle Problem

Forgetting is no longer confined to specific tasks. It is a potential regression of desired behaviors that can occur at *any stage* in the model's development.



Pretraining & Continued Pretraining

Drift in **general knowledge**, **language coverage**, and **style priors**.

Instruction Tuning (SFT)

Drift in **prompt following**, **formatting**, and **generalization**.

Alignment (RLHF/DPO)

Drift in **refusal/safety behavior**, **helpfulness trade-offs**, and **tool-use preferences**.

THANK YOU

Any Questions?

Shrey Satapara
Fujitsu Research of India
https://shreysatapara.github.io