Continual Pre-training of Language Models

Zixuan Ke[1], Yijia Shao[2], Haowei Lin[2], Tatsuya Konishi[3], Gyuhak Kim[1] & Bing Liu[1]

[1] University of Illinois at Chicago

[2] Peking University

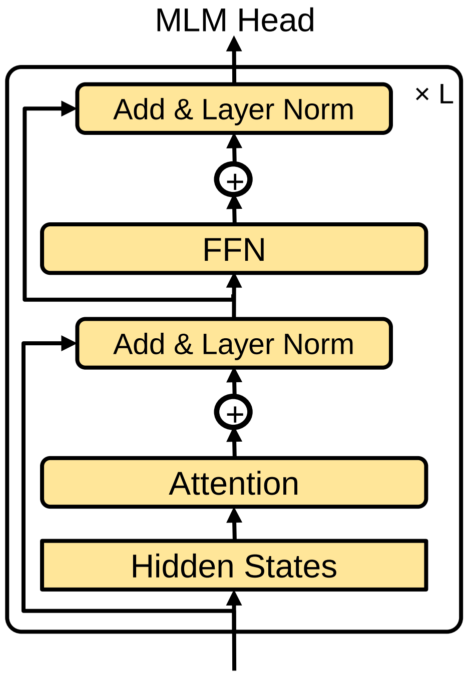
[3] KDDI Research

Code and data: https://github.com/UIC-Liu-Lab/ContinualLM

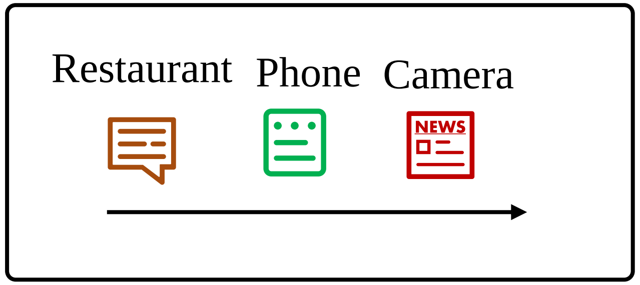
Continual Pre-training of Language Models

- **Existing** language models (LMs) once trained are fixed.
- **However**, in the real world, data shifts constantly and new domains, events or topics keep emerging
- This requires LMs **to be updated** to serve the user better
- **Our focus:**
  - Continually learning/pre-training an LM using a sequence of domain corpora, which we call ***continual domain-adaptive pre-training***
    - **Domain:** an emerging or specialized event or topic

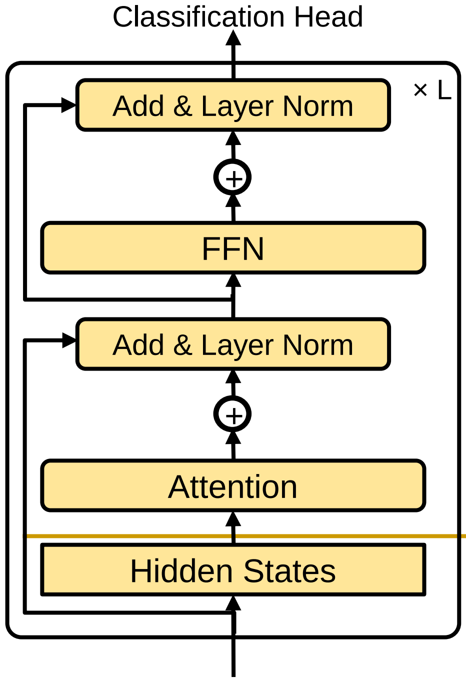
Ke et al., Continual learning of language models, ICLR 2023



(A) Continual Domain-adaptive Pre-training



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



(B) Individual Fine-tuning

End-tasks

ASC-Restaurant

ASC-Phone

ASC-Camera

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

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

ASC: Aspect Sentiment Classification

Continual Domain-adaptive Pre-training

6 domains

Unlabelde Domain Datasets			End-Task Classification Datasets				
Source	Dataset/Domain	Size	Dataset/Domain	Task	#Training	#Testing	#Classes
Reviews	Yelp Restaurant	758MB	Restaurant	Aspect Sentiment Classification (ASC)	3,452	1,120	3
	Amazon Phone	724MB	Phone	Aspect Sentiment Classification (ASC)	239	553	2
	Amazon Camera	319MB	Camera	Aspect Sentiment Classification (ASC)	230	626	2
Academic Papers	ACL Papers	867MB	ACL	Citation Intent Classification	1,520	421	6
	AI Papers	507MB	AI	Relation Classification	2,260	2,388	7
	PubMed Papers	989MB	PubMed	Chemical-protein Interaction Prediction	2,667	7,398	13

Continual domain-adaptive pre-training

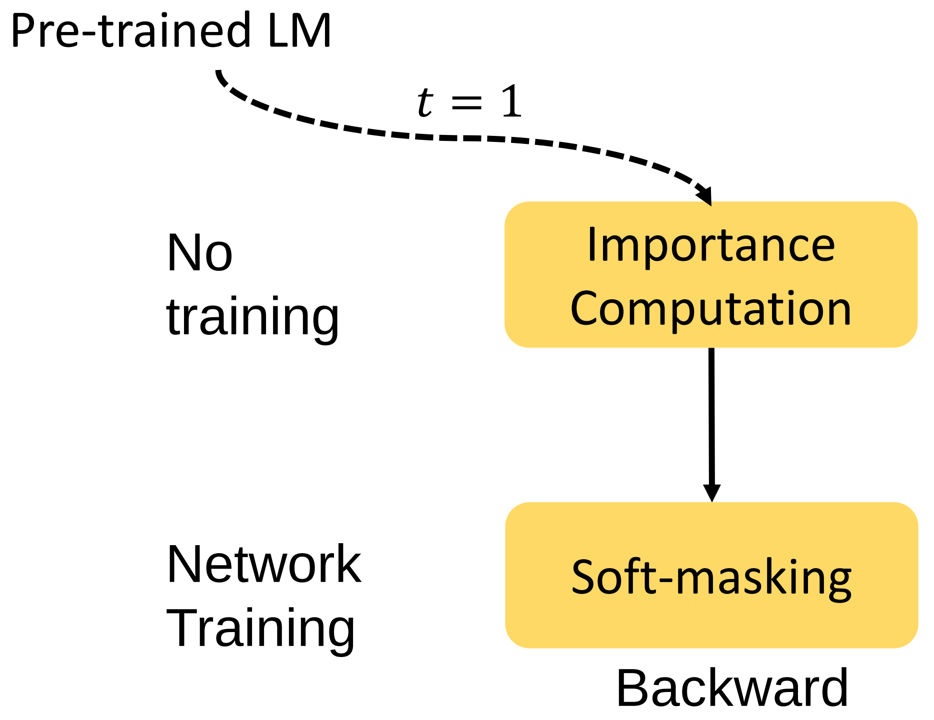
Individual Fine-tuning, after continual domain-adaptive pre-training

Continual Domain-adaptive Pre-training

- Setting
  - Continually learning or pre-training a language model (LM) using a sequence of domain corpora
  - **No access** to the data or corpora used in **the** original **pre-training** or **the previously learned domains**
  - End-task doesn’t know its domain belonging
- Goals
  - Catastrophic forgetting (CF) prevention
  - Knowledge Transfer (KT), including backward and forward KT
- Approach
  - **DAS** (continual **D**omain-**A**daptive pre-training of LMs with **S**oft-masking)

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

Continual Domain-adaptive Pre-training



Key ideas:

1) Detect importance of units for general and domain knowledge

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

3) These can prevent forgetting and allow knowledge transfer

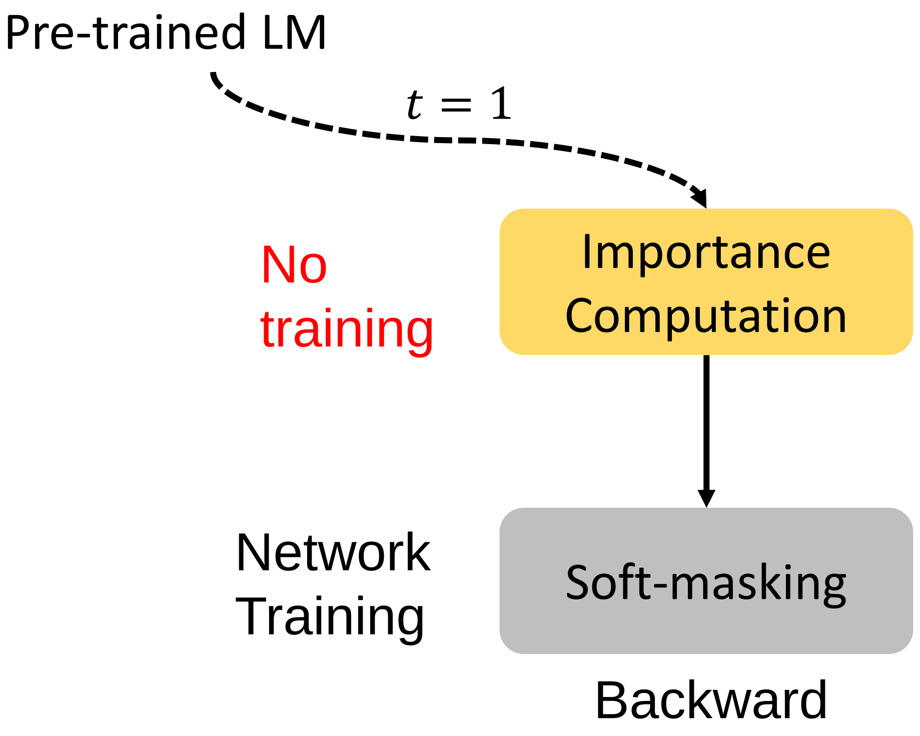
Key challenges:

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

2) How to soft-mask

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

Continual Domain-adaptive pre-training



Goal: Compute the importance of units for **general** (and domain) knowledge

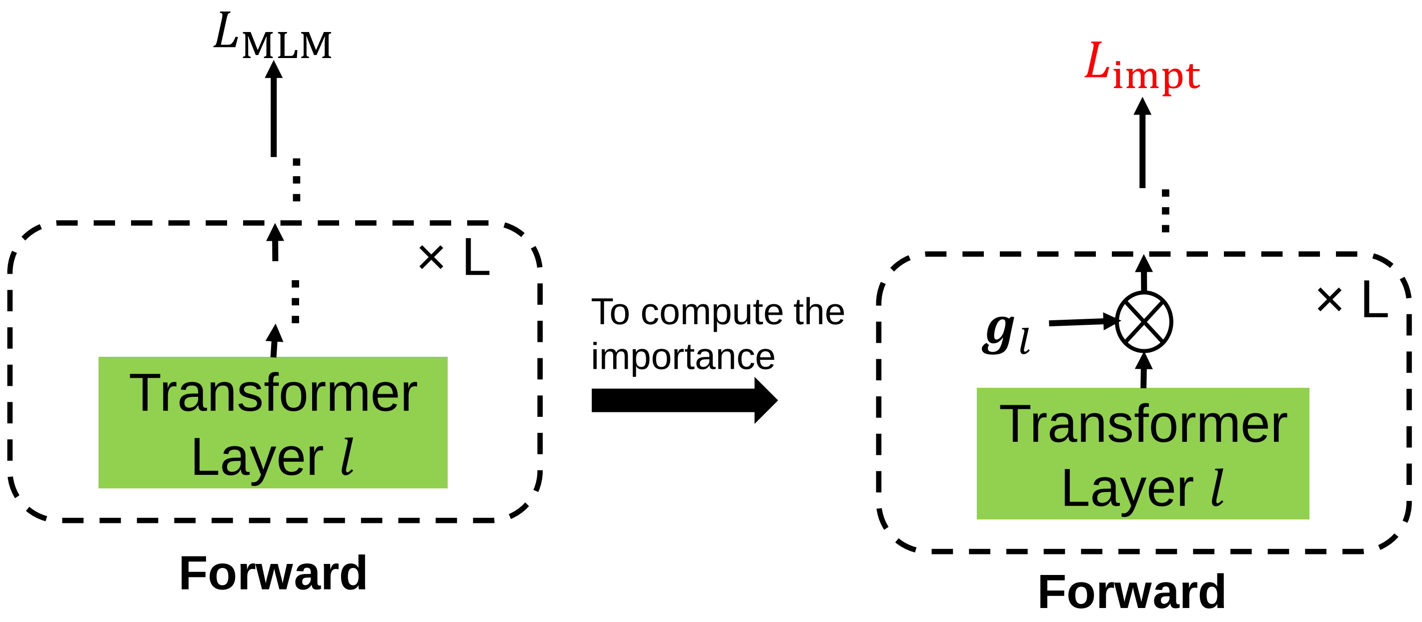
Why?

1) Not all units are important

2) Given the important units, we can protect them afterward

No training involved. We only need the importance

Importance Computation via Virtual Parameters



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

For **domain knowledge,**

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

$$\boldsymbol{\nabla}_{\boldsymbol{g}_l}^{m} = \frac{\partial L_{\text{impt}}(\boldsymbol{x}_m^{(t)}, \boldsymbol{y}_m^{(t)})}{\partial \boldsymbol{g}_l}$$

$$\boldsymbol{I}_l^{(t)} = \frac{1}{M} \sum_M |\boldsymbol{\nabla}_{\boldsymbol{g}_l}^{m}|$$

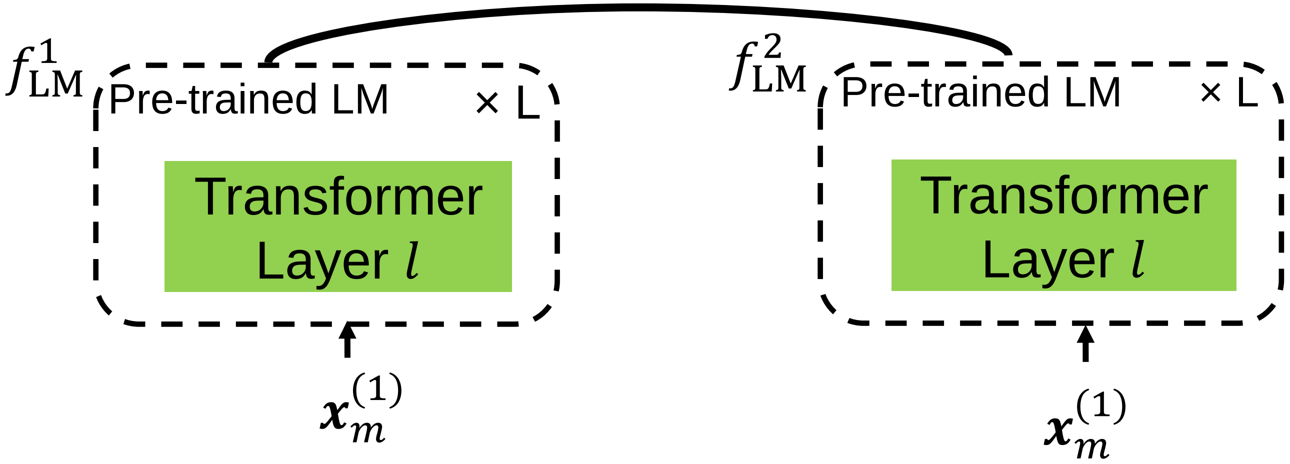
Use the **absolute gradient** to indicate importance[1]

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

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

Importance Computation via Virtual Parameters

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



KL: How different are the two representations?

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

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

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

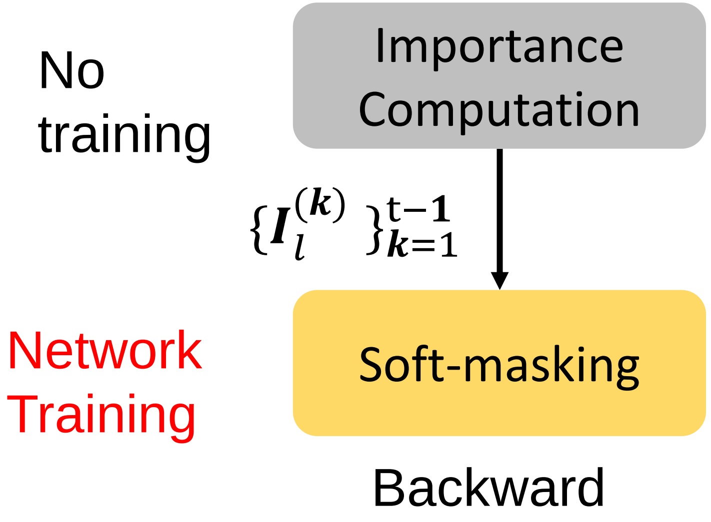
For **general knowledge**, we leverage the **random dropout** in standard Transformer

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

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

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

Continual Domain-adaptive Pre-training



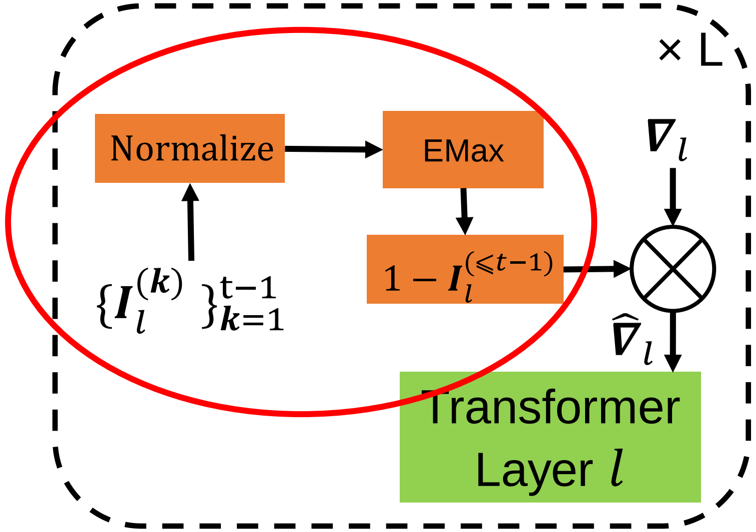
Goal: Soft-mask the **gradient** based on the importance

Why?

1) We need to protect the important units when training new domain

2) We want to allow knowledge transfer

Soft-masking



Backward

First, we normalize the importance so that they are comparable

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

Second, we accumulate the importance

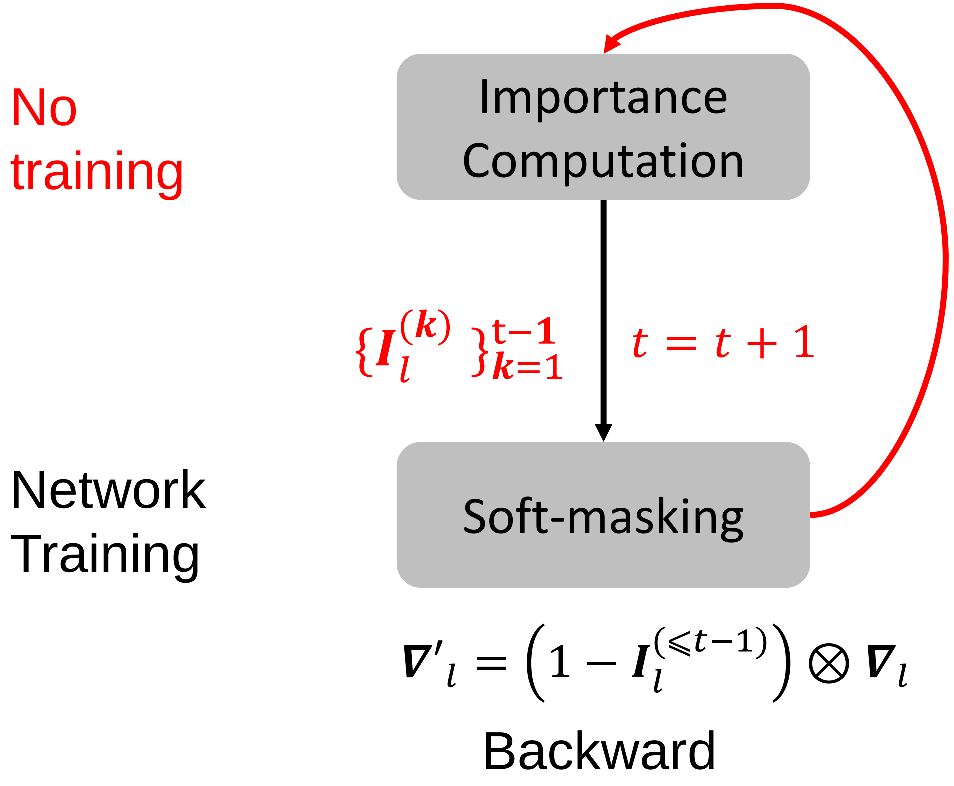
$$I_l^{(\leqslant t-1)} = \mathrm{EMax}(\{I_l^{(t-1)}, I_l^{(t-2)}\})$$

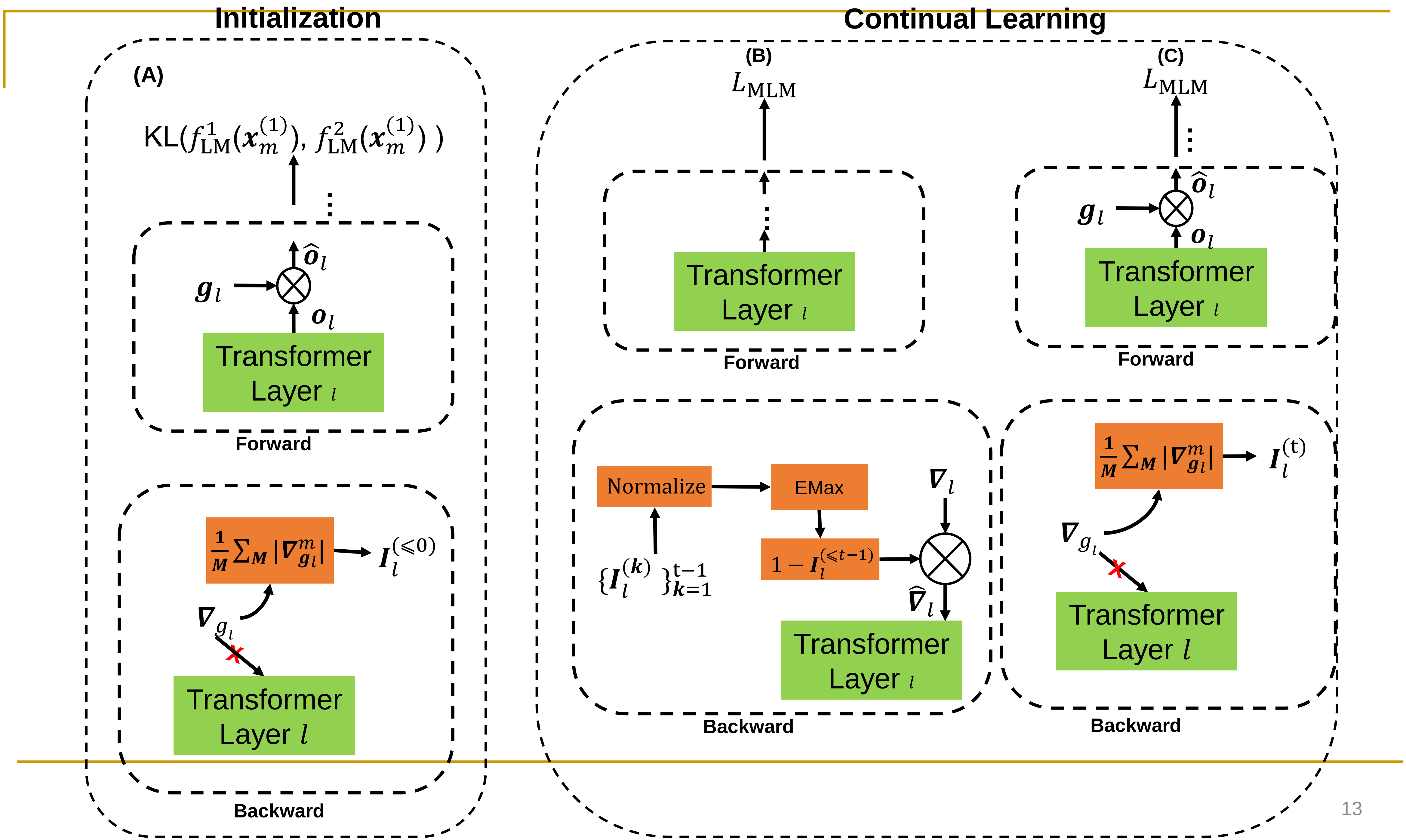
Third, we soft-mask the gradient (only in backward pass)

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

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

Continual Domain-adaptive Pre-training





Initialization
Continual Learning
(A)
$\mathrm{KL}(f_{\mathrm{LM}}^{1}(\boldsymbol{x}_m^{(1)}), f_{\mathrm{LM}}^{2}(\boldsymbol{x}_m^{(1)}))$
$\boldsymbol{g}_l$
$\hat{\boldsymbol{o}}_l$
$\boldsymbol{o}_l$
Transformer Layer l
Forward
$\frac{1}{M}\sum_M |\nabla_{g_l}^m|$
$\boldsymbol{I}_l^{(\leqslant 0)}$
∇_{g_l}
Transformer Layer l
Backward
(B)
L_{MLM}
Transformer Layer l
Forward
Normalize
EMax
$\{\boldsymbol{I}_l^{(k)}\}_{k=1}^{t-1}$
$1-\boldsymbol{I}_l^{(\leqslant t-1)}$
∇_l
$\hat{\nabla}_l$
Transformer Layer l
Backward
(C)
L_{MLM}
$\boldsymbol{g}_l$
$\hat{\boldsymbol{o}}_l$
$\boldsymbol{o}_l$
Transformer Layer l
Forward
$\frac{1}{M}\sum_M |\nabla_{g_l}^m|$
$\boldsymbol{I}_l^{(t)}$
∇_{g_l}
Transformer Layer l
Backward

Results

Overall end-task performance (final performance)

No pre-training

Pre-traing

NCL pre-training

SoTA pre-training

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

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

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

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

- We study the problem of continual pre-training of language model
  - We **incrementally** accumulate knowledge to the LM by
    - Computing **importance** of units for general and domain knowledge, with **different** L_{impt}
    - **Soft-masking** the backward propagation based on importance (help CF and KT)

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

Thank you

- We will have in-person poster @ ICLR23
  - **Tue May 02 11:30 a.m. — 1:30 p.m. (Kigali Time) @ MH1-2-3-4 #90**

- We have benchmarked many SoTA baselines
  - For continual end-task learning
    - https://github.com/ZixuanKe/PyContinual
  - For continual domain-adaptive pre-training
    - https://github.com/UIC-Liu-Lab/ContinualLM

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