AAAI 2025

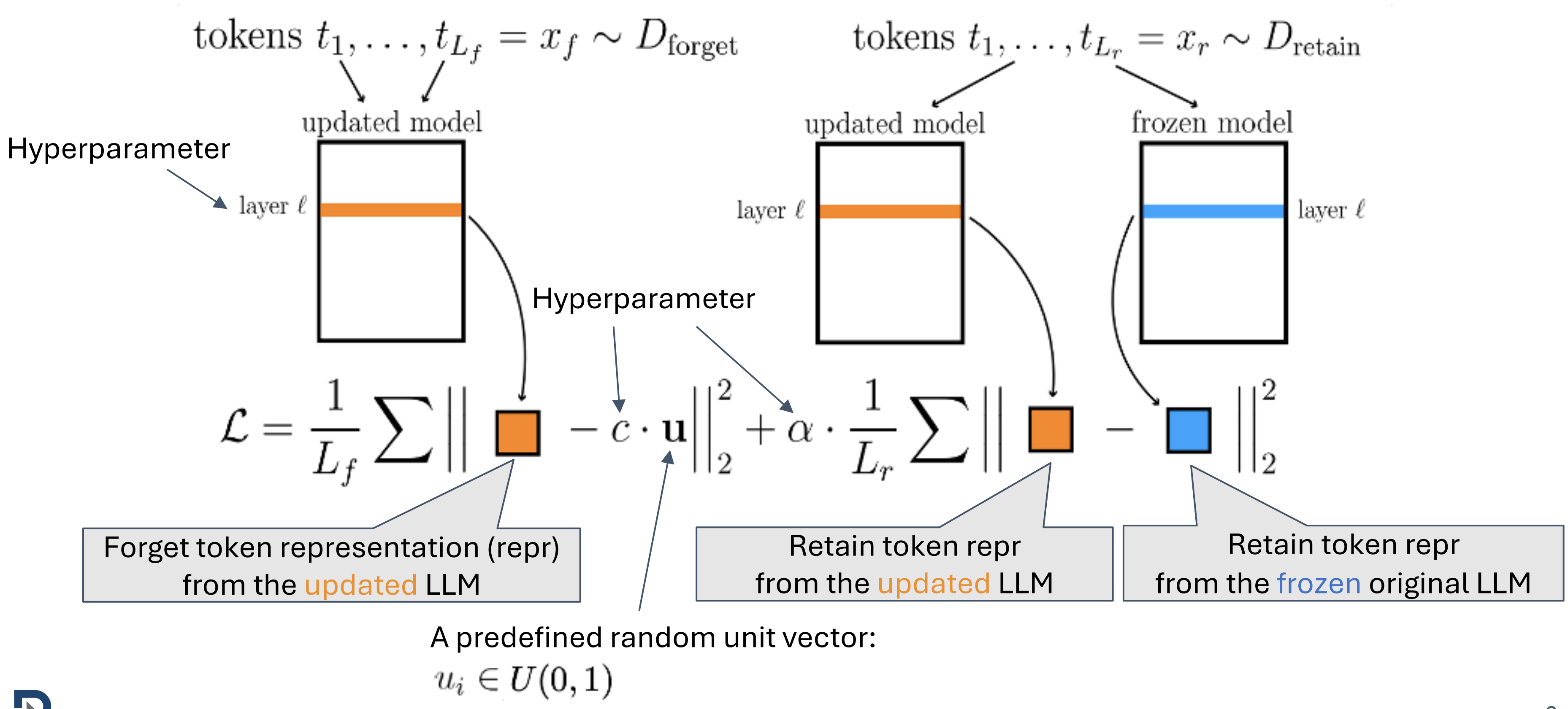
On Effects of Steering Latent Representation for Large Language Model Unlearning

Huu-Tien Dang   Tin Pham   Hoang Thanh-Tung   Naoya Inoue

LLM Unlearning

- Remove or suppress specific knowledge from a pretrained LLMs, while retaining their other knowledge
- Inputs:
  - LLM parameter: θ
  - Forget set (sentences): D_{forget} (e.g., private sensitive information)
  - Retain set (sentences): D_{retain} (e.g., Wikipedia)
- Goal:
  - Update θ so that:
  - Acc.(Questions about D_{forget}) ↓ (e.g., What is Naoya Inoue's home address? → ABC)
  - Acc.(Questions about D_{retain}) → (e.g., Where is the capital of Japan? → Tokyo)

RMU: Representation Misdirection for Unlearning

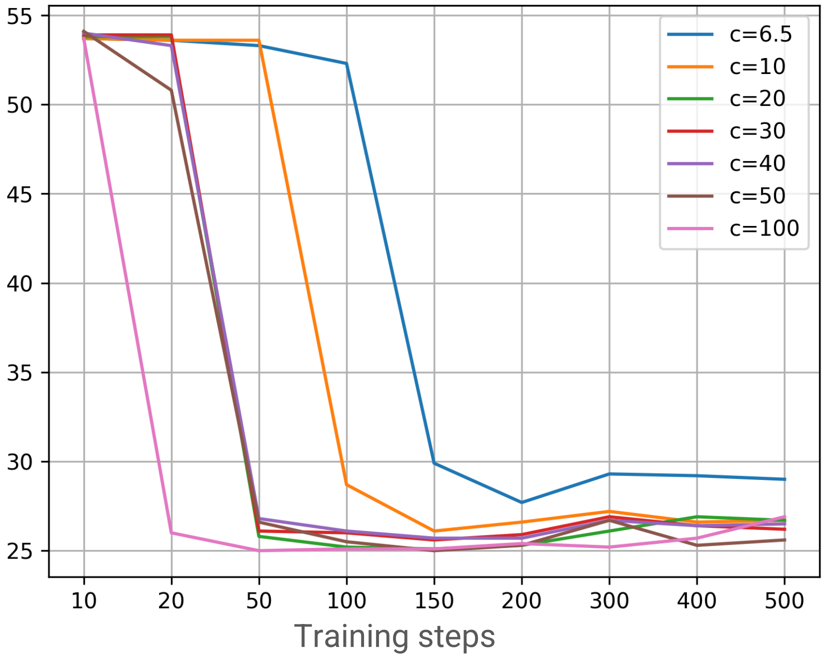


Li, Nathaniel, et al. "The WMDP Benchmark: Measuring and Reducing Malicious Use with Unlearning." *ICML*.

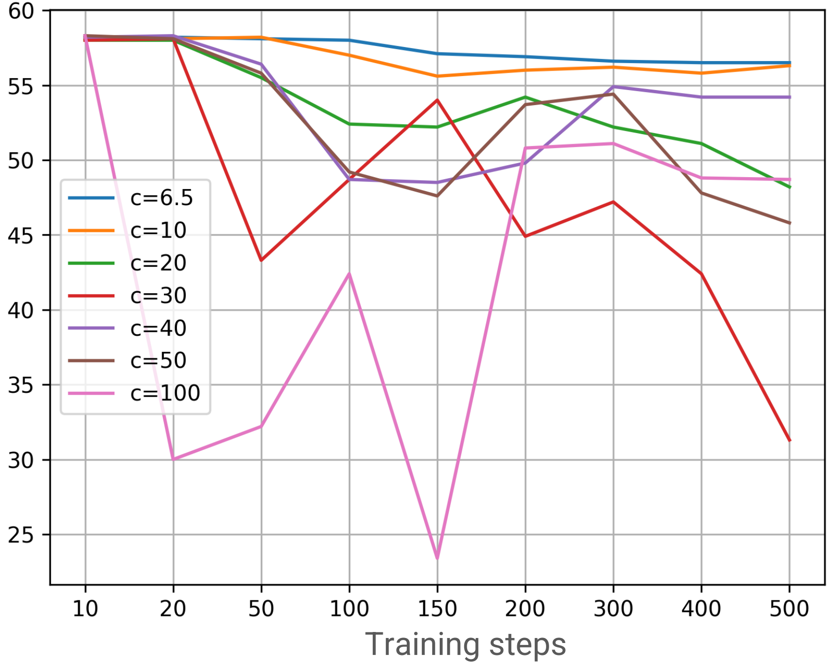
Issues in RMU: hyperparameter tuning is hard and costly

- RMU is empirically shown to be effective for unlearning and robust against knowledge recovery attacks
- **However**: Hyperparameters c, l need careful calibration, but there is no principled way to determine c, l
  - Needs grid search over both l and c ... but it is computationally expensive!

Demo: c needs sweetspot



QA accuracy on
forget set (WMDP)



QA accuracy on
retain set (MMLU)

Our contributions

- 🎓Theoretical and 🧪empirical analysis of RMU:
    1. How does c affect next token token prediction?
    2. What is the role and effect of c?
    3. What is the optimal value of c for effective unlearning across layers?
    4. Why is RMU robust against knowledge recovery attacks? (see the paper)
- Propose **Adaptive RMU**, which dynamically adjusts c during unlearning
    - Higher drop-in-accuracy for forget knowledge, retaining general knowledge
    - Effective unlearning for most unlayers without additional computational overhead
    - Still needs grid search, but not over both l and c!

Preliminaries

- **Definition 1:** Unlearned models & Logits of forget tokens

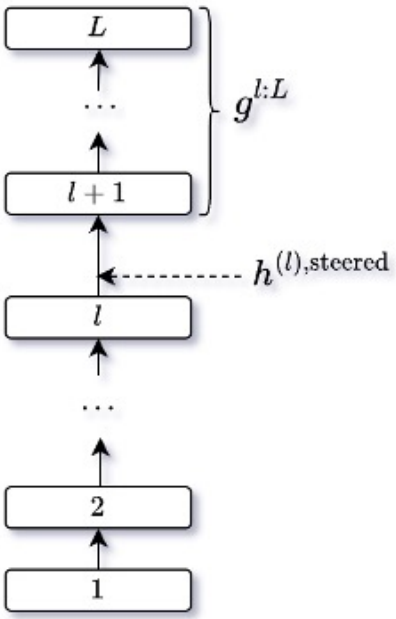
Final logits of generated forget-tokens

Unembedding matrix

$$
\begin{aligned}
f^{unlearn}(x_{F,n+1}|x_{F,1:n}) &= \boldsymbol{W} f^{(l:L),steered}(x_{F,n+1}|x_{F,1:n}) \\
&= \boldsymbol{W}(g^{(l:L)} \circ h^{(l),\text{steered}})(x_{F,n+1}|x_{F,1:n}) \\
&= \boldsymbol{W} g^{(l:L)}(h^{(l),\text{steered}}(x_{F,n+1}|x_{F,1:n})) \qquad (2)
\end{aligned}
$$

Composition of transformer layers

Steered representations at layer l



Transformer layers

Preliminaries

- **Assumption 1**: A well-unlearned model pushes the representations of all forget tokens toward a predefined random vector

$$h^{(l),\text{steered}}(x_{F,i}) = c\boldsymbol{u} + \boldsymbol{\epsilon},$$

Optimization Error $\mathcal{N}(\mathbf{0}, \eta \boldsymbol{I})$

A predefined coefficient

A predefined random unit vector

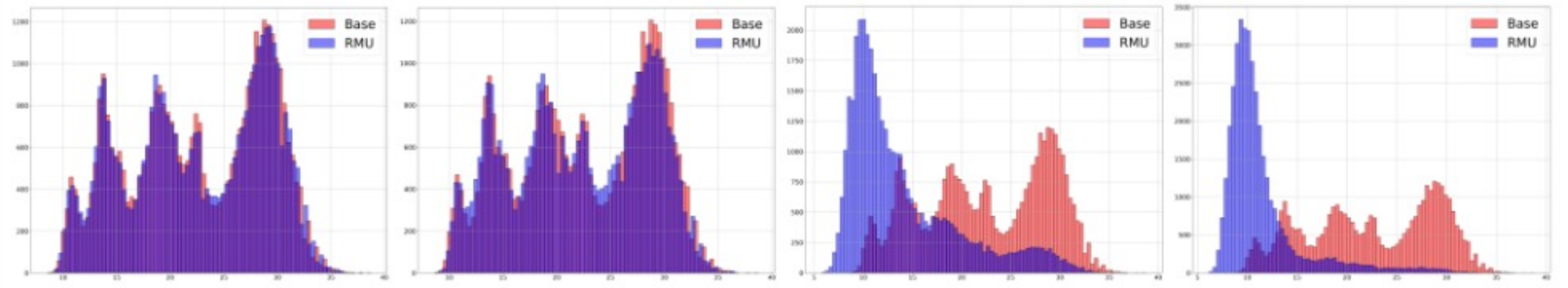
1) Logits are more randomized given larger c

Proposition 1. *If Assumption 1 holds, by Definition 1, the logit value of forget token* $x_{F,n+1}$ *generated by unlearned model* f^{unlearn} *given as* $f^{\text{unlearn}}(x_{F,n+1}|x_{F,1:n})$ *follows the Normal distribution* $\mathcal{N}\left(\boldsymbol{W}g^{(l:L)}(\boldsymbol{z}), \eta\boldsymbol{W}\nabla_{\boldsymbol{z}}g^{(l:L)}(\boldsymbol{z})^{\top}\nabla_{\boldsymbol{z}}g^{(l:L)}(\boldsymbol{z})\boldsymbol{W}^{\top}\right)$, *where* $\boldsymbol{z} = c\boldsymbol{u}$.

Varies depending on the specific characteristics of sub-networks g, but ***a larger c could introduce more randomness to the logit?***

1) Logits are more randomized given larger c

- Ask LLMs about questions related to forget set
- Distribution of answer confidence (by max logit values of ans. tokens)



(a) $c = 1.0$ (b) $c = 2.0$ (c) $c = 5.0$ (d) $c = 10.0$

- With larger c, RMU-unlearned model generates answer tokens with lower confidence → Larger c introduces more randomness to logits

2) Larger c aligns forget token reprs more with random vector

Jacobian matrix—a linearized $g^{(l:k)}$ at a given input

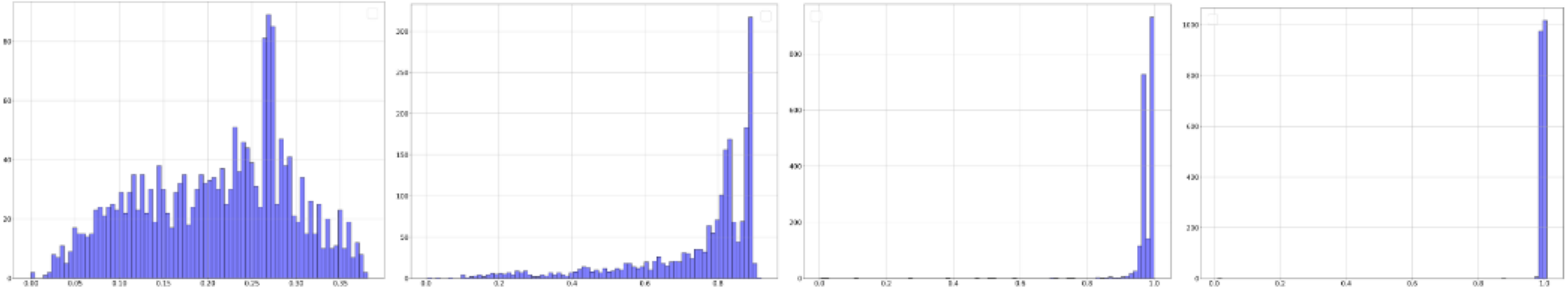
- **Proposition 2:** c and $\cos\left(\boldsymbol{J}^{(l:k)}\boldsymbol{u}, \boldsymbol{J}^{(l:k)}(\hat{h}^{(l)} - \boldsymbol{\epsilon})\right)$ are positively correlated.

A predefined random unit vector

Forget token repr at layer l

2) Larger c aligns forget token reprs more with random vector

- Extract token reprs from forget set
- Compute cosine sim. between them and $\boldsymbol{u}$



(e) $c = 1.0$ (f) $c = 2.0$ (g) $c = 5.0$ (h) $c = 10.0$

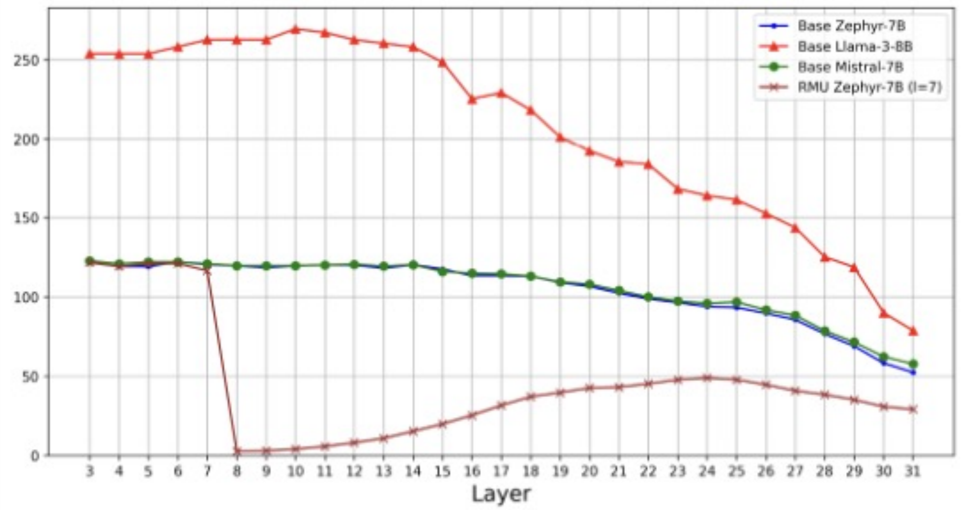
- Clearly, larger c promotes the alignment

3) Different layers/models require different c

- Define noise sensitivity of layers:

Injected Noise

$$\Phi(g^{(l:k)}, \mathcal{D}_{\text{forget}}) = \frac{||g^{(l:k)}(\hat{h}^{(l)}(x_F) + \boldsymbol{\xi}) - g^{(l:k)}(\hat{h}^{(l)}(x_F))||^2}{||g^{(l:k)}(\hat{h}^{(l)}(x_F))||^2}$$

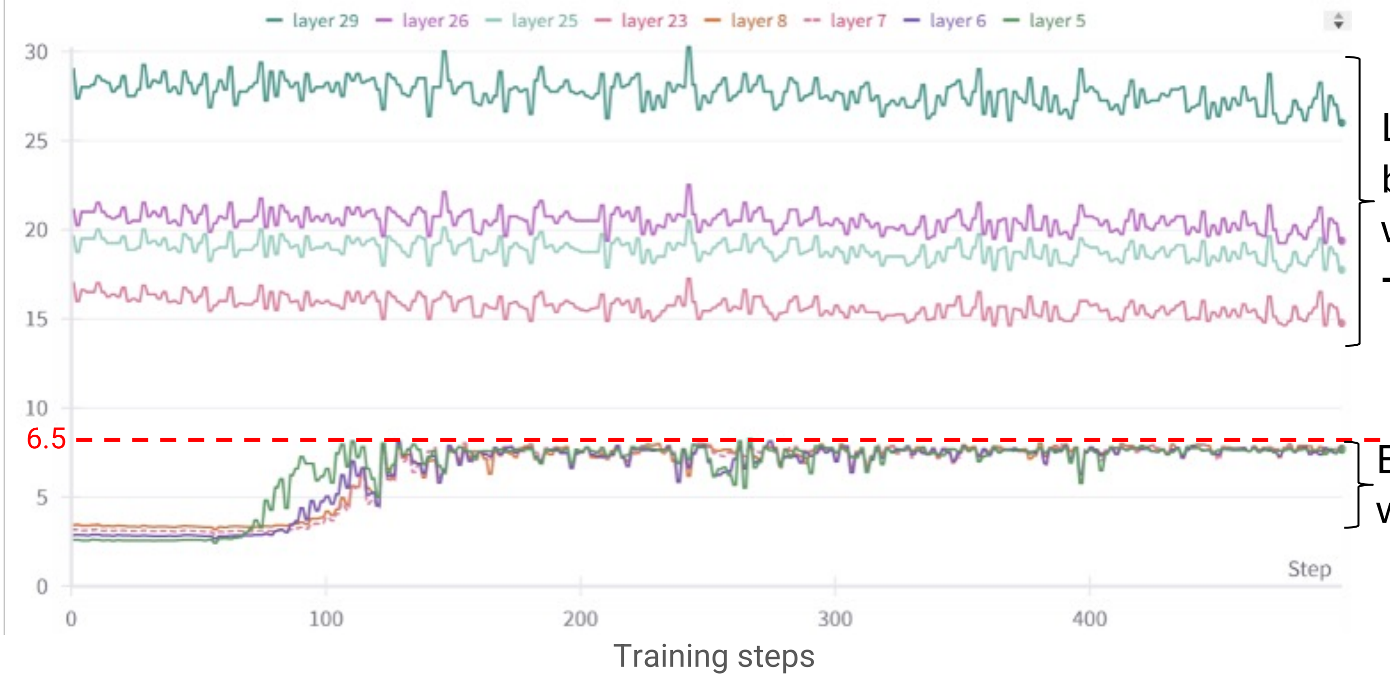


- Later layers are more robust to noise

→ Unlearning with later layer also needs larger c?

3) Different layers/models require different c

- Fix c (=6.5) and unlearn with various layers l
- Observe how L2 norm of each layer's repr changes



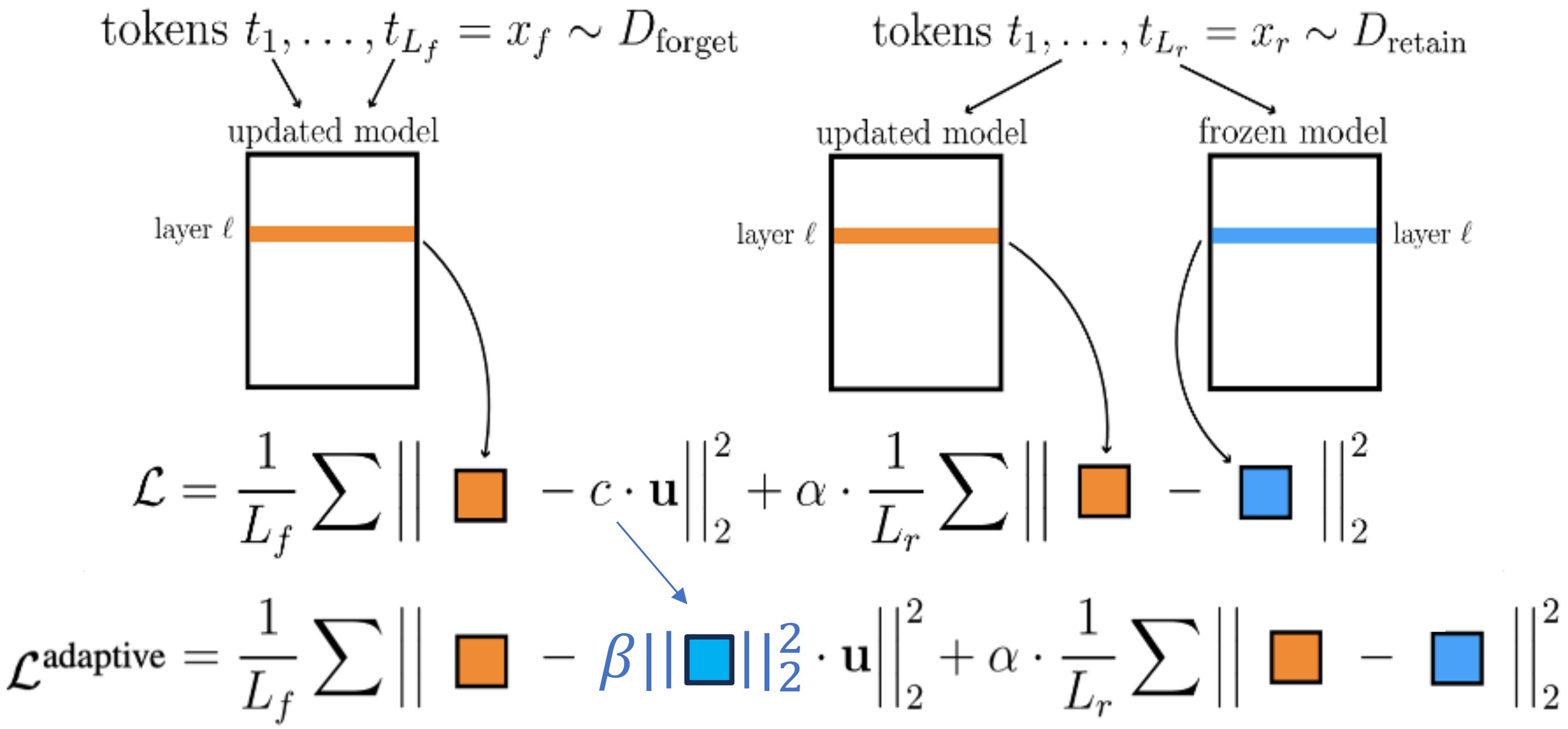
Later layers cannot be adjusted to $c\boldsymbol{u}$ with smaller c
→ **Unlearn fails!**

Earlier layers can be well adjusted to $c\boldsymbol{u}$

The findings lead to AdaptiveRMU

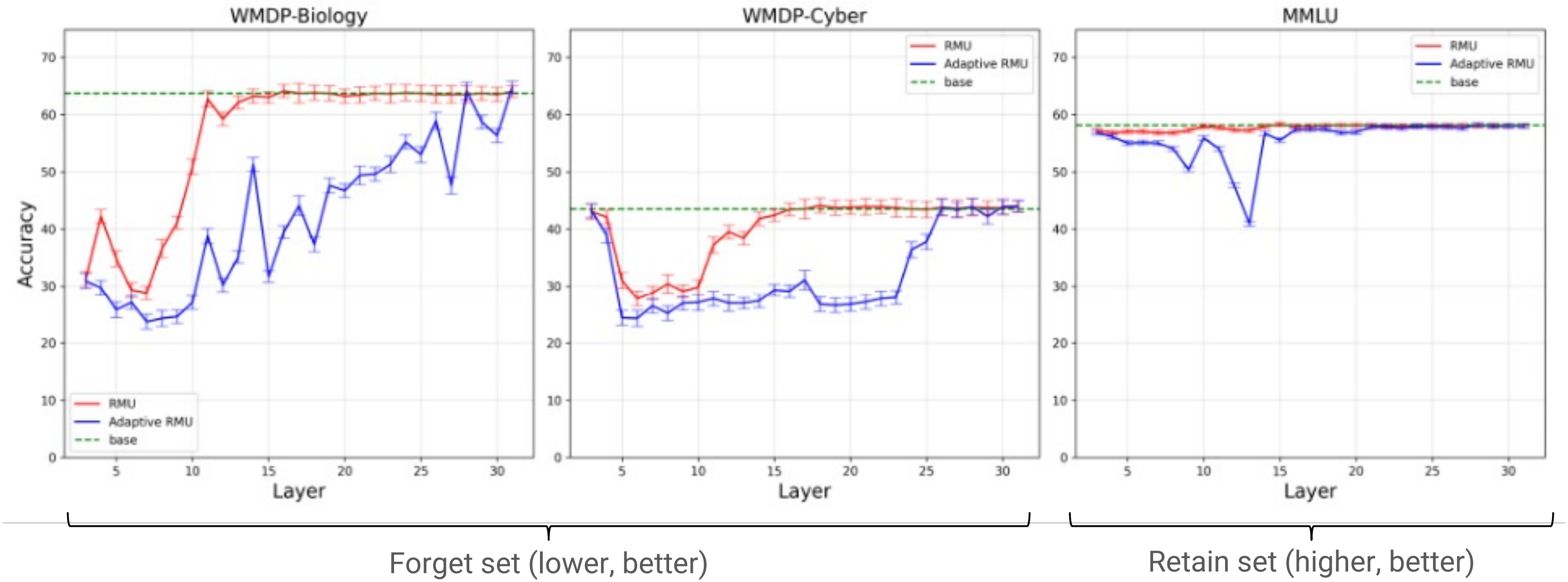
- How does c affect next token prediction?
  - RMU tries to push all forget reprs at the intermediate layer toward a random repr
  - This randomness is propagated through layers, causing the reduction in generated token confidence
- What is the role and effect of c?
  - Higher c leads to more randomness of the output
  - Higher c leads to more alignment between forget reprs and the random vector
- What is the optimal value of c for effective unlearning across layers?
  - Early layers require smaller noise (smaller c) whereas later layers require larger noise (larger c) to produce the same level of output randomness

Proposed: Adaptive RMU (very simple yet effective)



Results: AdaptiveRMU works for most layers!

- Ablation test: Fixed c (=6.5) v.s. Adaptive c



Summary

- Theoretical and empirical analysis of RMU
- Propose to use layer-adaptive c, which eliminates the need of hyperparameter tuning and even improves the unlearning performance

- Code: https://github.com/RebelsNLU-jaist/llm-unlearning
- Contact: Tien (dtienuet@gmail.com) and Naoya (naoya-i@jaist.ac.jp)
- Lab: https://rebelsnlu.super.site/