Share or Not?

Learning to Schedule Language-Specific Capacity for Multilingual Translation

Biao Zhang, Ankur Bapna, Rico Sennrich, Orhan Firat

International Conference on Learning Representations 2021

Share or Not? A Dilemma in Multi-Task Learning

- Sharing inductive bias among tasks encourages universal representations that could benefit inter-task knowledge transfer
- However, different tasks may conflict, interfere each other, necessitating task-specific modeling
- How to design balanced architecture to handle this dilemma broadly challenges research communities (NLP, CV, .etc)
- We explore this research question in multilingual translation setup

Rebuff et al., 2018; Ruder et al., 2019; Wang et al., 2020;

Multilingual Translation: **One Model for All**

Share all parameters across all language pairs

- ✅ allow knowledge transfer towards low-resource task
- ❌ limits expressivity

<2zh> Happy families are all alike.

幸福的家庭都是相似的。

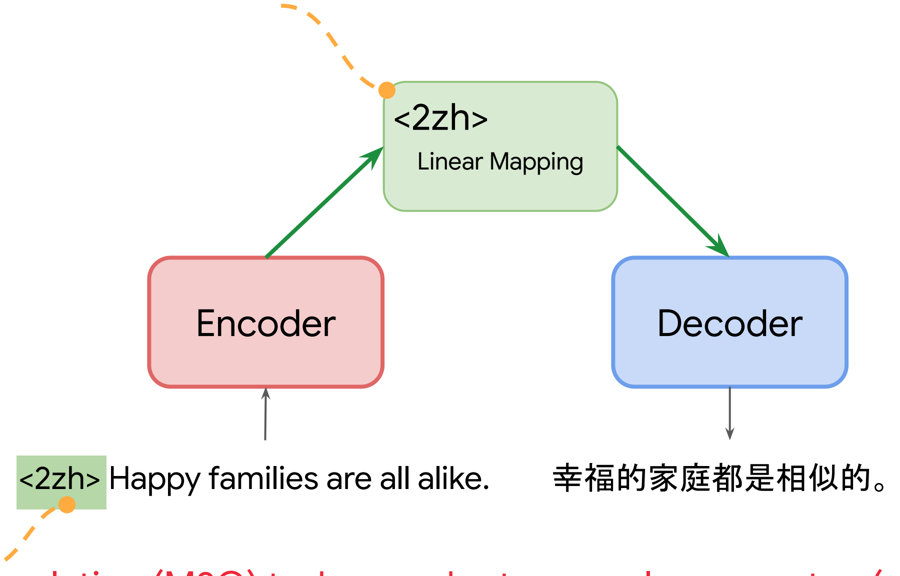
The target language tag tells model where to translate

The encoder/decoder can be any neural model.

Firat et al., 2016, Ha et al., 2016; Johnson et al., 2017

Why share everything? Language-Specific (LS) Modeling

Associate each target language with a unique mapping matrix

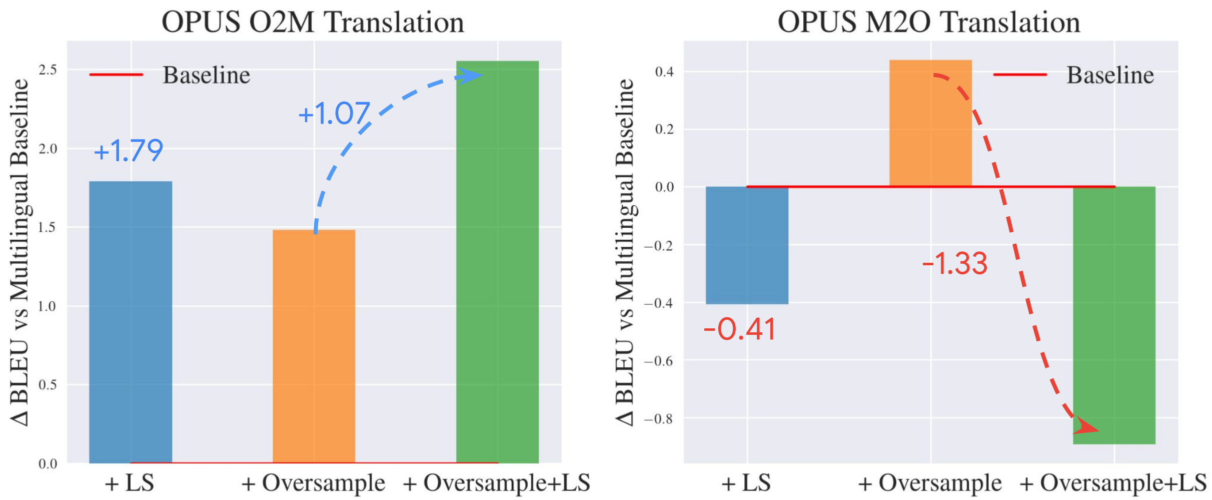


For many-to-one translation (M2O) task, we adopt source language tag (<en2>) instead

Experimental Setup

- Transformer-base model, 8 heads, 512/2048 model size, 6 layers
- Setting: one-to-many task (O2M) and many-to-one task (M2O)
- Dataset:
  - OPUS-100 (Zhang et al. 2020): massively multilingual corpus
  - WMT-14: multilingual dataset from WMT benchmarks
- Data conditions:
  - original training corpus
  - oversampled corpus with a temperature of 5 (T=5)
- Evaluation
  - average BLEU
  - win ratio (Zhang et al, 2020)

LS modeling **improves O2M**, but **hurts M2O**

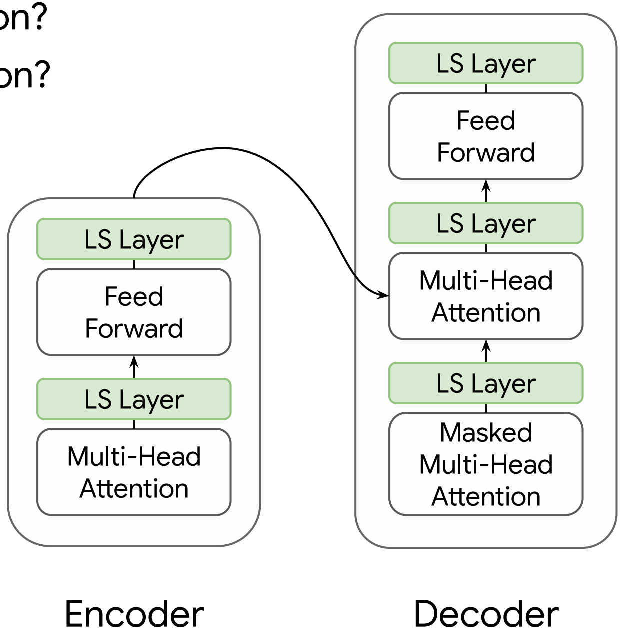


- LS capacity consistently benefits O2M translation, regardless of data distribution
- **LS capacity doesn't always improve translation, see M2O**

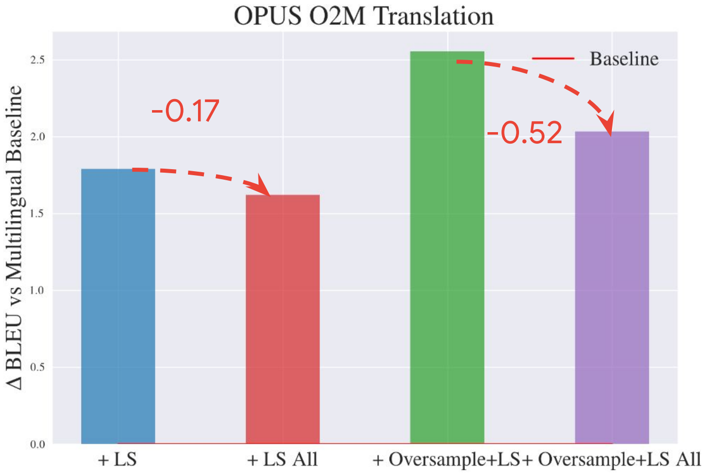
Blackwood et al. (2018) also observed negative performance with source language-specific modeling

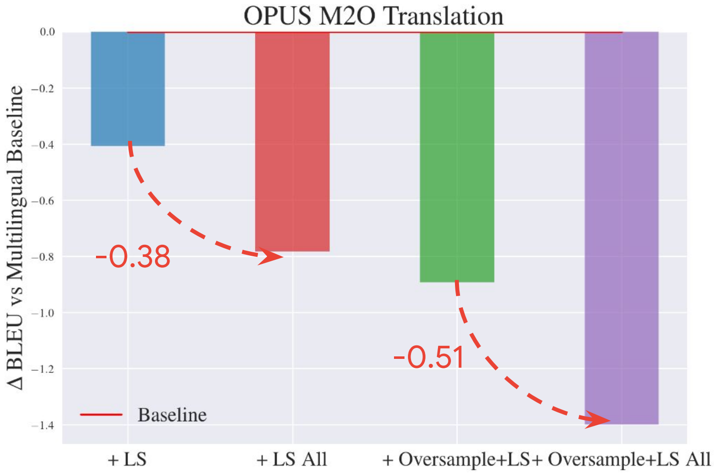
What if we add **LS layer** into each Transformer sublayer?

- Would these LS layers further improve O2M translation?
- Can we get positive improvements for M2O translation?



Aggressive LS modeling **hurts** both O2M and M2O





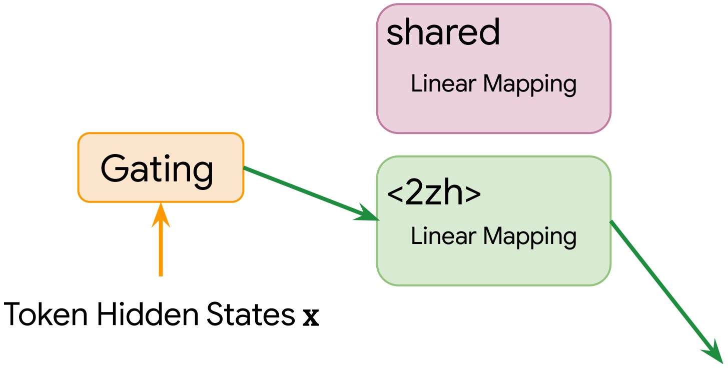
Answer to the questions: **NO!**

Too much LS modeling delivers **worse quality** for both O2M and M2O

Motivation

- LS modeling improves model's expressivity, but discourages knowledge transfer, revealing a trade-off
- Share or Not? When and where does LS modeling matter in multilingual Transformer?
- Answering these questions requires to search over large space that heuristics can hardly cover. Instead, we resort to data-driven approach

Conditional Language-Specific Routing (CLSR)



CLSR uses a **binary gate** to determine where to go: share or not?

$$\mathbf{h}^{\text{share}} = \mathbf{W}^{\text{share}} f(\mathbf{x})$$

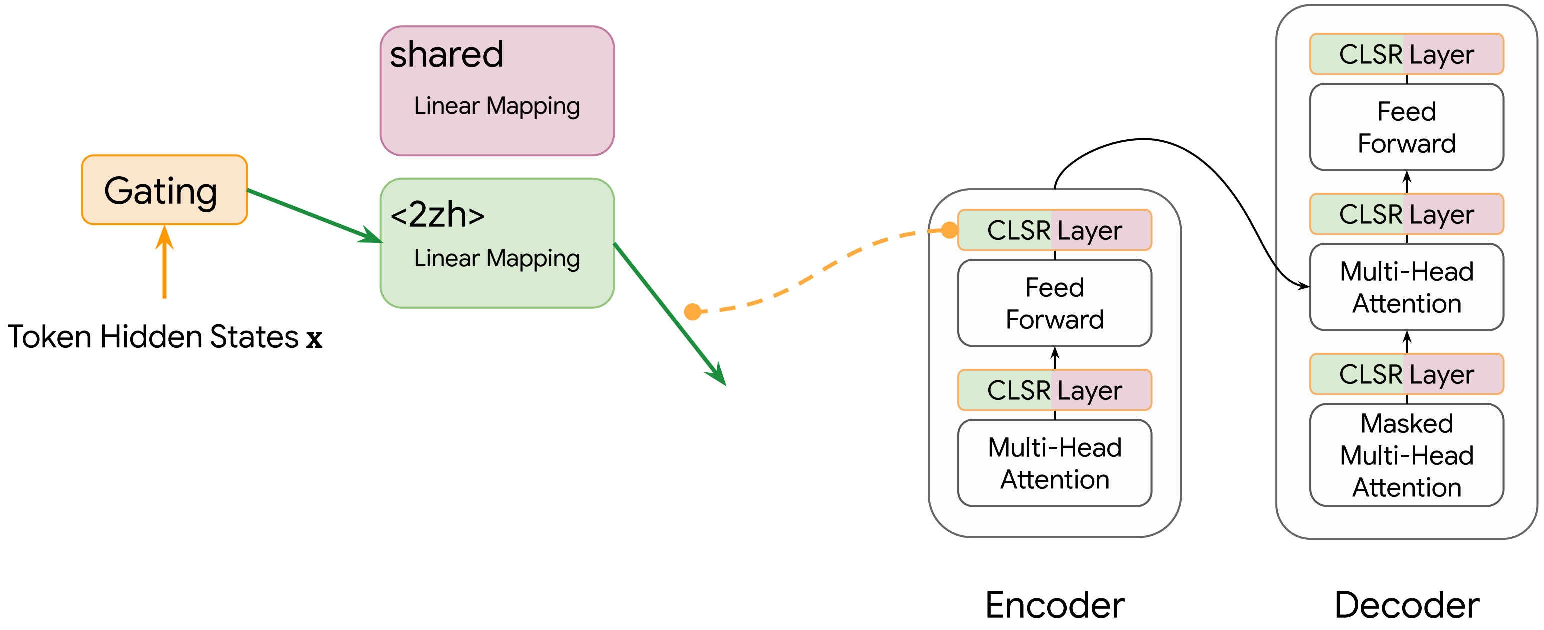
$$\mathbf{h}^{\text{lang}} = \mathbf{W}^{\text{<2zh>}} f(\mathbf{x})$$

$$\mathbf{h} = g \odot \mathbf{h}^{\text{lang}} + (1 - g) \odot \mathbf{h}^{\text{share}}$$

where $\mathbf{x}$ is an input state, $f(\cdot)$ is a Transformer sublayer, and the gate g is **a scalar value.**

Applying CLSR to Multilingual Transformer

We apply CLSR layer to all Transformer sublayers; let the model automatically learn the sharing pattern



Binarizing and Constraining Gates in CLSR

- Training:

  $$g(\mathbf{x}) = \mathrm{sigmoid}(\mathrm{MLP}(\mathbf{x}) + \alpha(t)N(0,1))$$

- Decoding:

  $$g(\mathbf{x}) = 0 \text{ if } \mathrm{MLP}(\mathbf{x}) < 0\text{; else } 1$$

- CLSR objective:

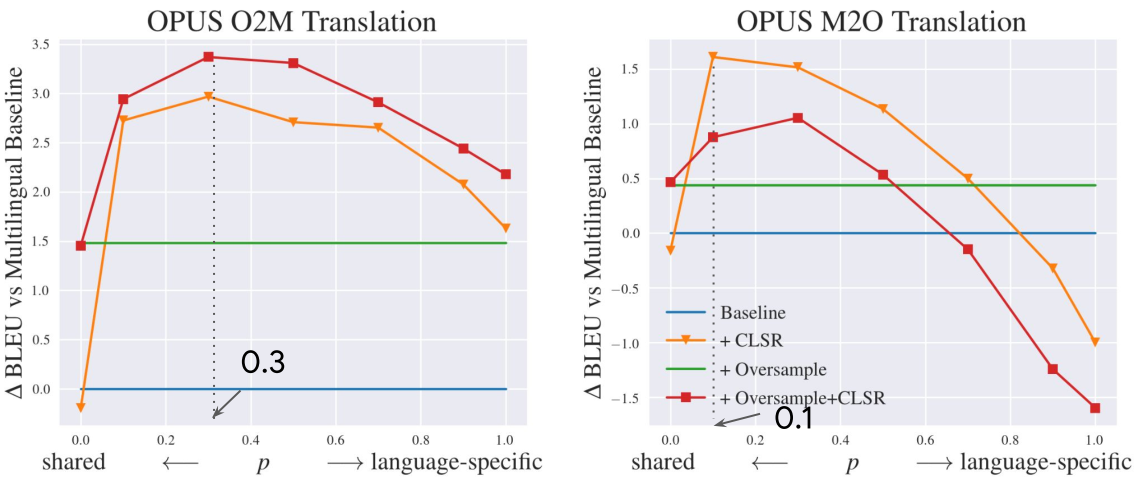
  $$\mathcal{L}(g) = \mathrm{abs}(\mathrm{AvgAllGates} - p)$$

Gaussian noise, variance controlled by $\alpha(t)$ which increases linearly along t.

Key Idea: smaller budget constraint p allows fewer open gates, forcing the model schedule LS computation to critical layers.

Controlling Computation versus Quality for Neural Sequence Models (Bapna et al. 2020)

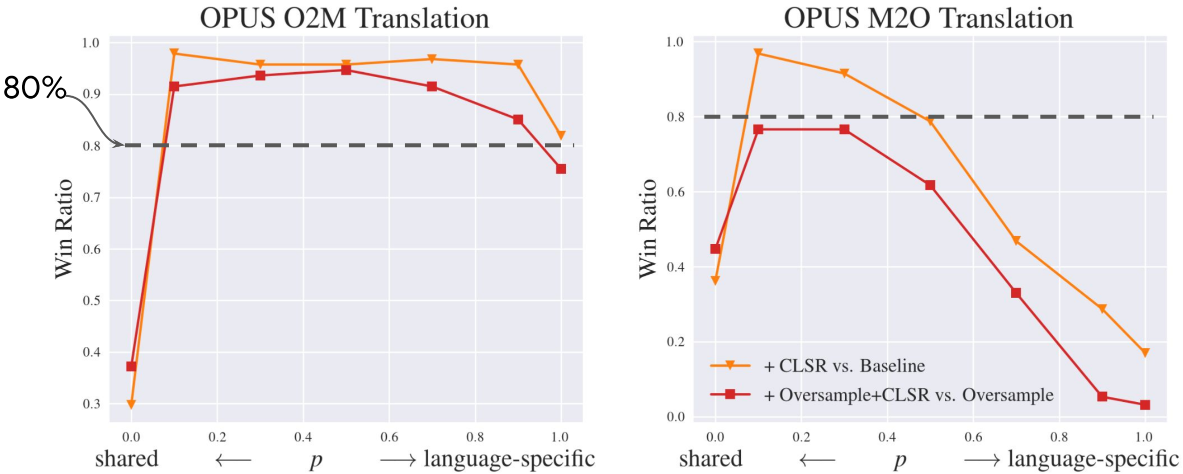
Is there a trade-off between LS and shared capacity?



YES

- CLSR discovers the optimal budget: **10%~30%**
- **O2M** translation requires more LS capacity, and also **benefit more** (+>1.5 BLEU)
- CLSR enables **positive** improvements on **M2O** translation

Evaluate Multilingual Translation with **Win Ratio**



Win Ratio counting the proportion of language pairs where new model outperforms its baseline

- CLSR achieves a WR of >~**80%,** consistent across data conditions and translation tasks.

Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation (Zhang et al. 2020)

LSScore: Layer Preference to LS Capacity

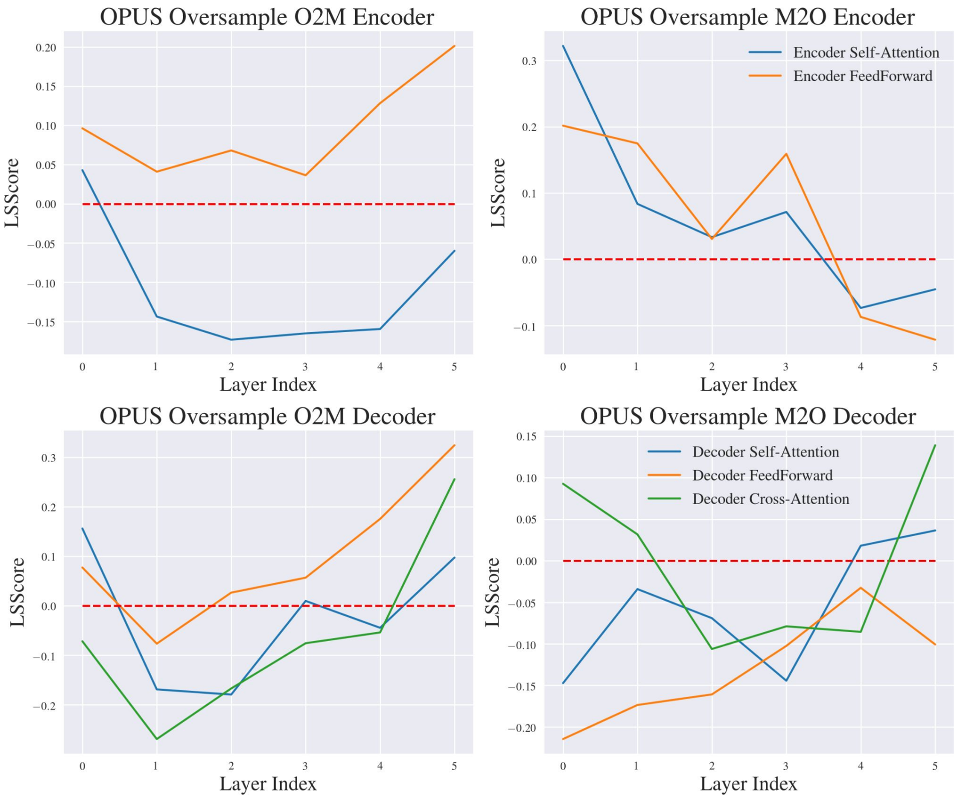
- A metric to evaluate how each sublayer favors LS capacity

$$\mathtt{LSScore}_f = \mathtt{Average}(g_{f,p} - p)$$

"$g_{f,p}$": average gating value over all test set tokens at sub-layer f when trained with budget p

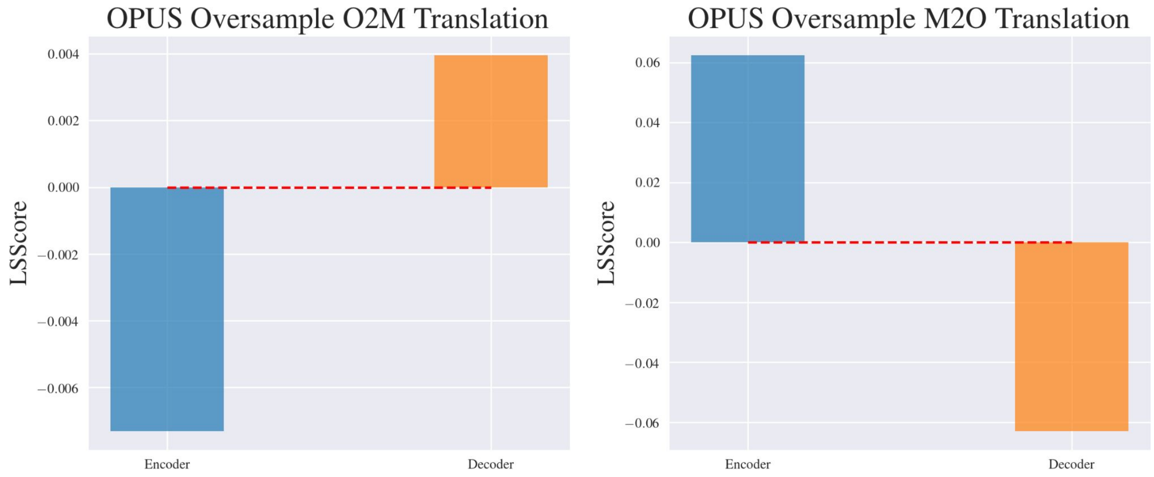
  - The training objective pushes $g_{f,p}$ towards budget p
  - Thus, $g_{f,p} > p$ means this layer uses more LS capacity than average

Which layers tend to be language-specific?



- Generally, **top and/or bottom encoder/decoder** layers favors LS capacity
- **FFN** sublayers consume more LS capacity in **O2M** translation
- It's **unclear** which types of sublayer use more LS capacity in **M2O** translation

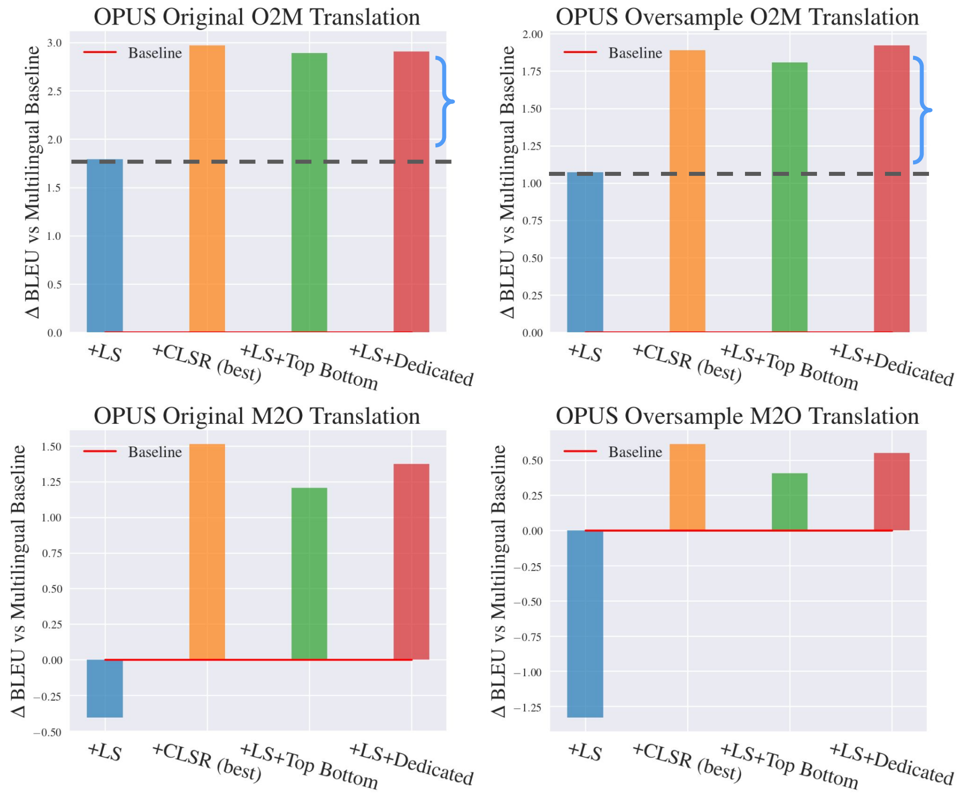
Should more LS capacity be used in encoder or decoder?



In multilingual translation, the “M” (other languages) side tends to use more LS capacity.

- O2M: encoder < **decoder**
- M2O: **encoder** > decoder

Could CLSR help to build better multilingual Transformer?



Yes

- We explore **two** modified Transformers:
  - **top and bottom** encoder/decoder layers
  - **dedicated sublayers** based on LSScore
- Overall, top-bottom and dedicated model outperform models with LS alone
- Dedicated performs better than top-bottom model

Summary

- We propose conditional language-specific routing to explore the trade-off between shared and LS modeling in multilingual Transformer
- We find both the amount and the position of LS layers matter
- The best performance is achieved by distributing 10%-30% LS computation to the top and/or bottom encoder/decoder layers

Paper: https://openreview.net/pdf?id=Wj4ODo0uyCF

Code: https://github.com/googleinterns/cct-m4