연세대학교 인공지능학과 석사과정
조우현

포트폴리오

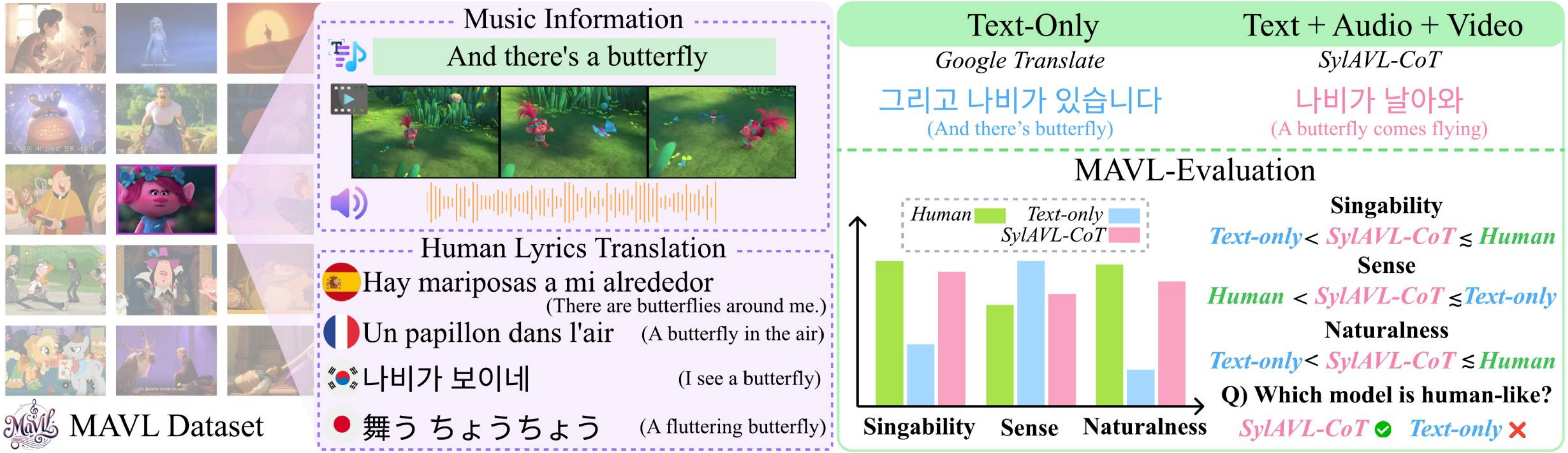
MAVL: A Multilingual Audio-Video Lyrics Dataset for Animated Song Translation

Woohyun Cho[1], Youngmin Kim[1], Sunghyun Lee[1], Youngjae Yu[2*]

[1]Yonsei University, [2]Seoul National University
*: Corresponding Author

Introduction: The Challenges of Lyrics Translation

- **Text Alone is Not Enough.**
- Especially in animated musicals, songs are closely tied to specific scenes, characters' expressions, and actions.
- Existing text-based translation often misses this **Audio-Visual Context**.



That's why I made MAVL dataset

- **Multilingual Audio-Video Lyrics Benchmark**
- MAVL is the first multimodal parallel dataset for animated song translation research.
- **Multilingual Support:** English, Spanish, French, Japanese, Korean (5 languages)
- **Multimodal Composition:** The three elements of Lyrics (Text), Song (Audio), and Video are precisely synchronized.
- **Scale:** Includes extensive data for a total of 228 songs.

Language	# Songs	# Video	# Sections	# Lines
English	228	228	1,923	6,623
Spanish	201	181	1,595	5,739
French	158	143	1,421	4,821
Japanese	138	114	1,264	4,280
Korean	133	117	1,138	3,974

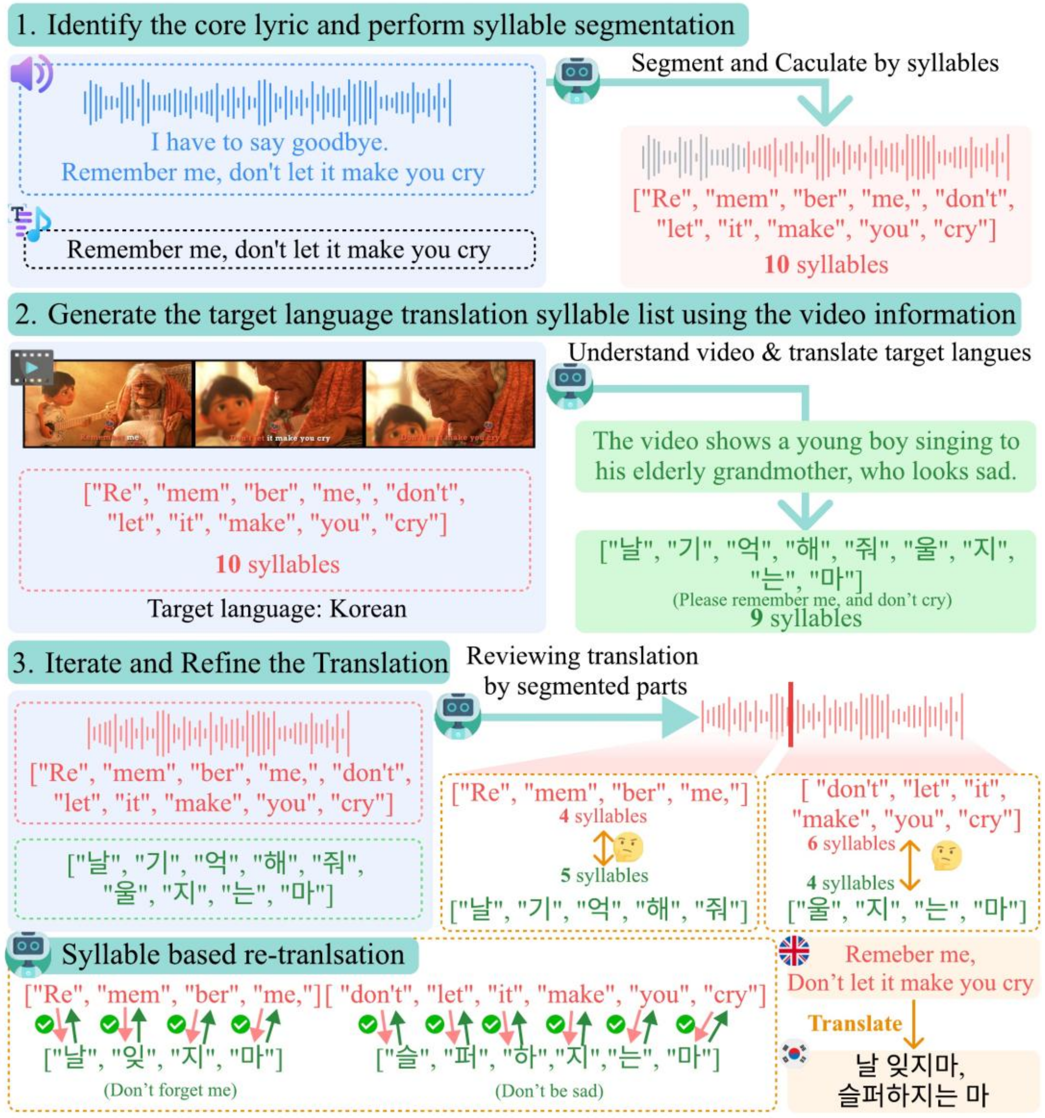
MAVL Collection Pipeline

- I've made the automatic pipeline to find animation songs metadata on Last.fm and to use the metadata for lyrics crawling on genius.com and lyricstranslate.com
- And manually aligned the lyrics across the languages manually, and across audio and videos automatically using Whisper-based tool, Stable-ts.

(a) Multilingual Lyrics Collection (b) Lyrics Human Alignment (c) Audio-Video Lyrics Alignment

SylAVL-CoT Pipeline

- Using MAVL Benchmark, I evaluated my new audio-video lyrics translation framework, SylAVL-CoT.
- This framework utilizes the audio and video data with Gemini using Chain-of-Thoughts prompt, to fully utilize Gemini's potential at lyrics translation.



Revisiting Residual Connection: Orthogonal Updates for Stable and Efficient Deep Networks

Giyoung Oh[1], **Woohyun Cho[1]**, Siyeol Kim[1], Suhwan Choi[2], Youngjae Yu[3*]

[1]Yonsei University, [2]Maum AI [3]Seoul National University
*: Corresponding Author

Motivation

- What if We apply residual connection with only with orthogonal component?
- Does it help improving the performance of the model by stabilizing the training?

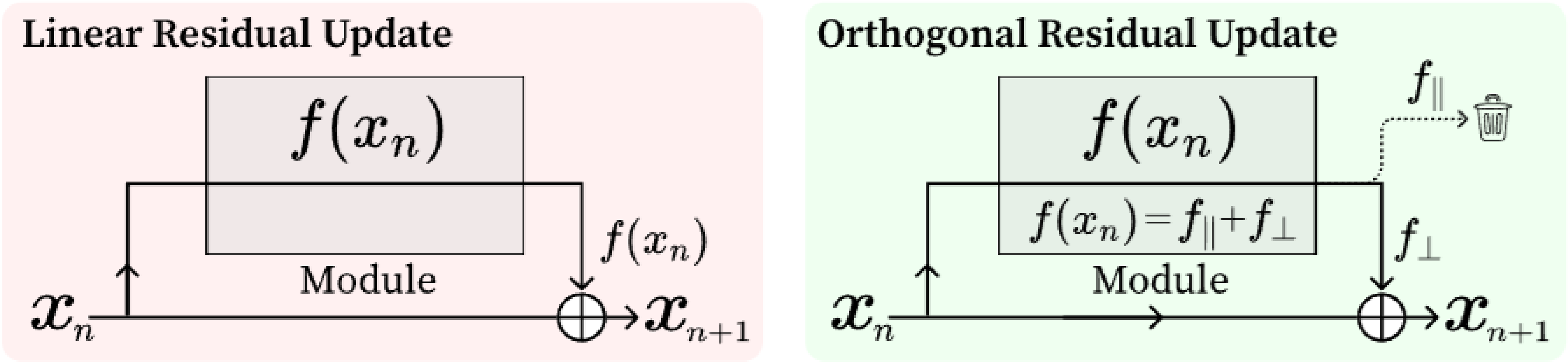


Figure 1: **Intuition behind our orthogonal residual update.** *Left:* The standard residual update adds the full output of module $f(x_n)$ to the input stream x_n. *Right:* Our proposed update first decomposes the module output $f(x_n)$ into a component parallel to x_n ($f_{||}$) and a component orthogonal to x_n ($f_{\perp}$). We then discard $f_{||}$ and add only the orthogonal component $f_{\perp}$ to the stream.

What I did for the project

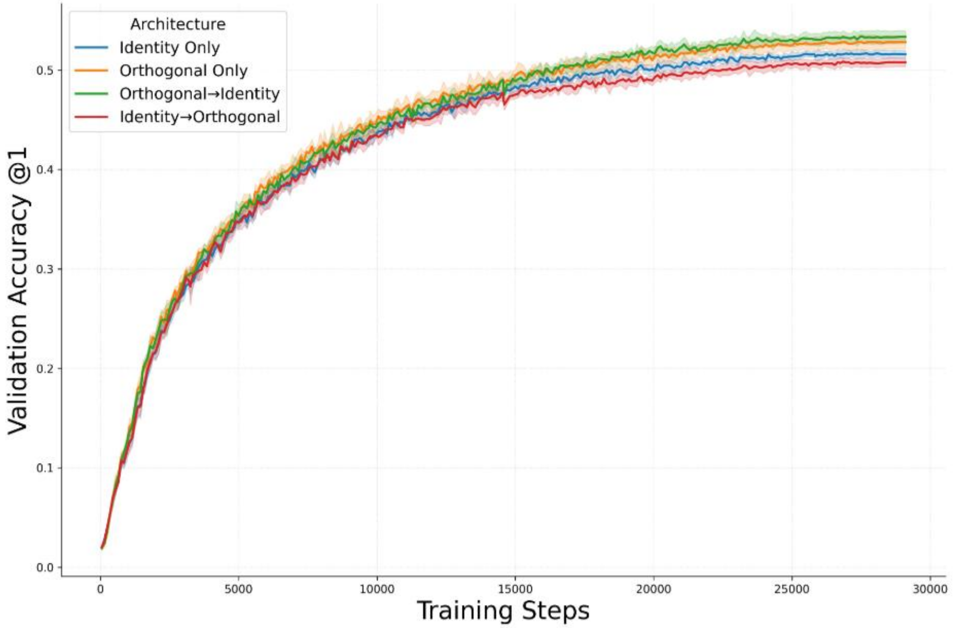
- For the ablation studies, I trained the image classification models, where I changed the model architecture during the training to see on which architecture, the model performs the best.
- The model performed the best when it starts training with orthogonal architecture and ends with linear (original architecture).

Table 3: Mean ± std. of top-1 (Acc@1) and top-5 (Acc@5) accuracy (%) from 5 independent runs for ViT-S, evaluating adaptability to connection type changes. Models are trained for 300 epochs (**Start Arch.**) then another 300 epochs (**End Arch.**) on the same dataset, with connections (Linear '**L**' or Orthogonal '**O**') potentially switched. Compared are **L→L**, **L→O**, **O→L**, and **O→O** on CIFAR-10, CIFAR-100, and Tiny ImageNet. Results are averaged over the final epochs of the **End Arch.** phase.

Dataset	**Start Arch.→ End Arch.**	**Acc@1 (%)**	**Acc@5 (%)**
CIFAR-10	Linear → Linear	92.78±0.06	99.74±0.03
	Linear → Orthogonal	92.88±0.14	99.72±0.03
	Orthogonal → Linear	93.89±0.12	**99.75**±0.04
	Orthogonal → Orthogonal	**94.10**±0.12	**99.73**±0.04
CIFAR-100	Linear → Linear	74.22±0.13	92.26±0.13
	Linear → Orthogonal	74.02±0.24	91.96±0.17
	Orthogonal → Linear	**75.63**±0.17	**92.91**±0.17
	Orthogonal → Orthogonal	75.38±0.35	92.20±0.13
TinyImageNet	Linear → Linear	53.24±0.13	75.25±0.21
	Linear → Orthogonal	52.14±0.18	74.20±0.20
	Orthogonal → Linear	**54.58**±0.10	**76.45**±0.24
	Orthogonal → Orthogonal	53.88±0.29	75.34±0.23

What I did for the project

- For the ablation studies, I trained the image classification models, where I changed the model architecture during the training to see on which architecture, the model performs the best.
- The model performed the best when it starts training with orthogonal architecture and ends with linear (original architecture).



(c) Tiny ImageNet, Acc@1

Thank You