Few-shot Font Style Transfer with Localized Representations

NAVER AI Lab.

Song Park

NAVER AI LAB

Contents

- **Introduction**
- **Background**
- **Problem definition: Few-shot font style transfer**
- **Few-shot Font Generation with Localized Style Representations and Factorization** (AAAI'21)
- **Conclusion**

“ ***Font*** is paints for **text** *designs* ”

Font matters.

Building a brand with “Font”

Hyundai Card

배달의민족

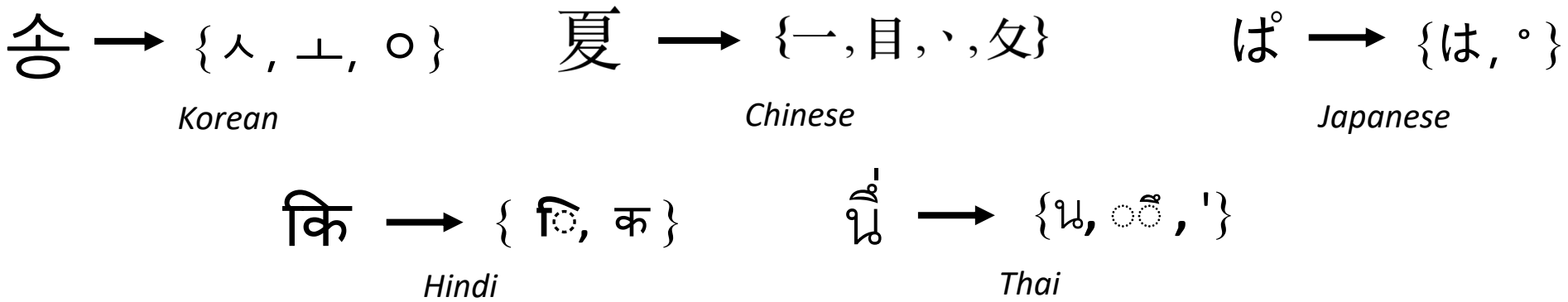
Font design is too expensive,

particularly for
“Glyph-rich” languages.

“Compositionality”

- **A character can be decomposed into a number of components.**
  - 19,514 Chinese characters can be represented by the combination of 371 components.

- **24 of top 30 popular language systems have compositionality.**
  - Korean, Chinese, Japanese, Hindi, Thai…



Can we utilize the compositionality to design the font?

파란 봄 하늘 위에는

뭉게뭉게 구름꽃이 피어나고,

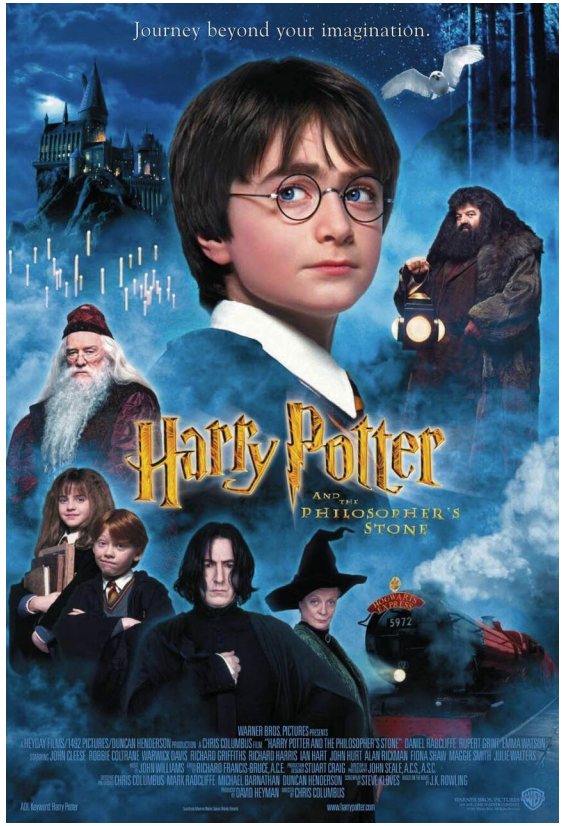
우리 아기 얼굴에는

방긋방긋 웃음꽃이

만발했다.

《AG 안상수체 2012》

Containing multiple languages



Introduction to
“**Automatic Font Generation**”

Goal: Capturing **“font styles”** from *very few* glyph images

- **Font style: global, also very local**

  - A “font style” covers from **local characteristics** to **global structures**.
- **Character content: very complex but sensitive to local changes**
  - All the detailed structures should be maintained.
  - Even very small damages can hurt the meaning of the character.

How can we preserve all these “styles” and “contents”?

Existing automatic font generation works

Fine-tuning with reference images

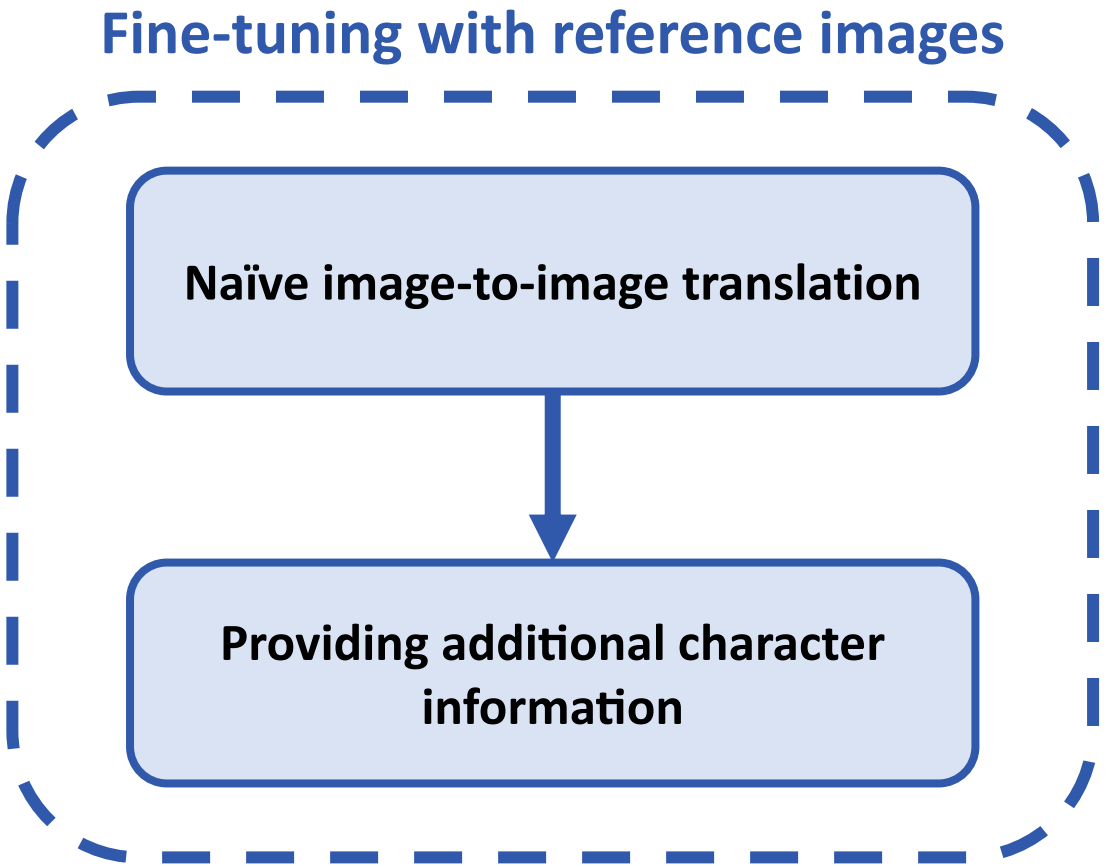
Generation in inference time

Existing automatic font generation works

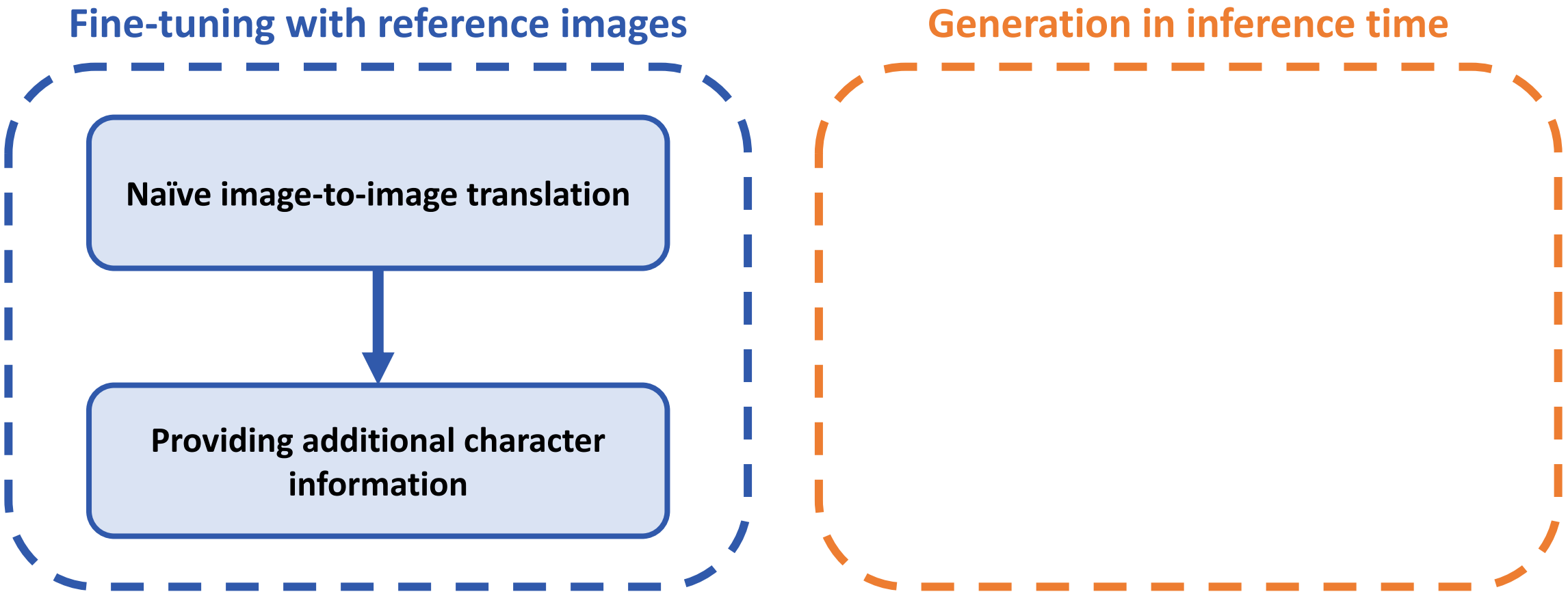
Fine-tuning with reference images

Naïve image-to-image translation

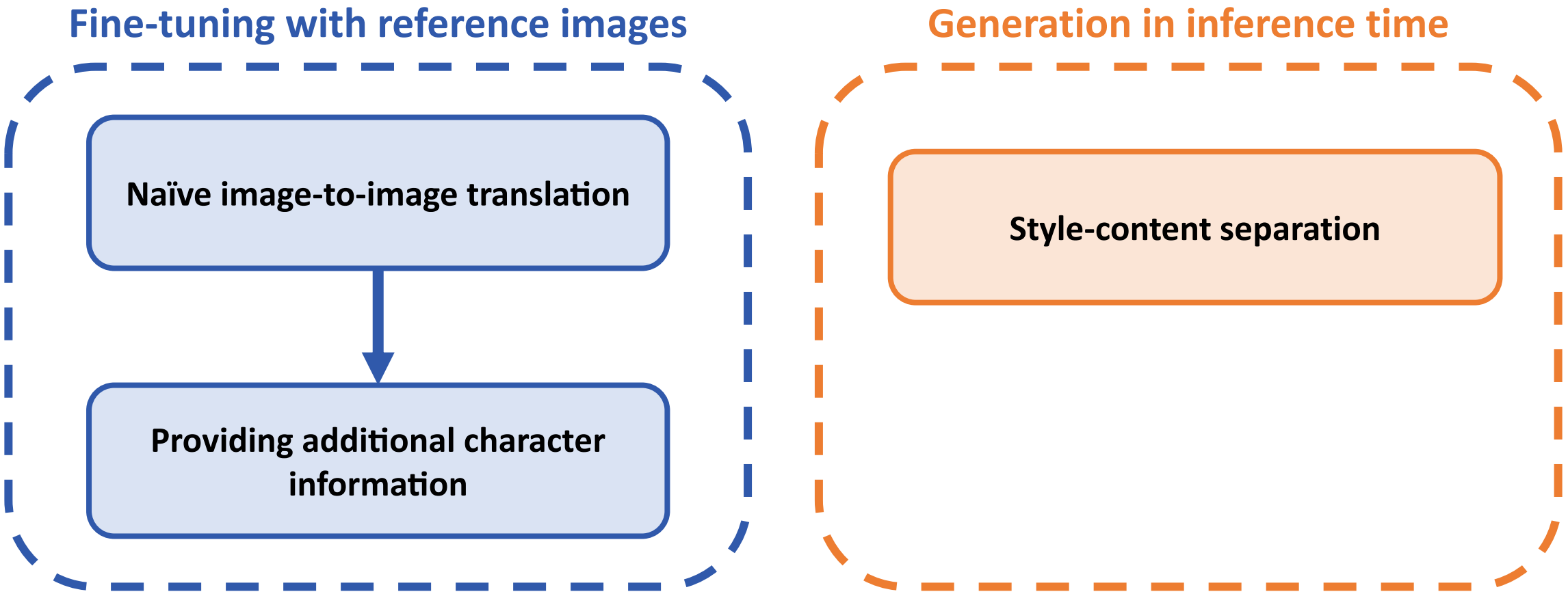
Existing automatic font generation works



Existing automatic font generation works



Existing automatic font generation works



Existing automatic font generation works

Fine-tuning with reference images

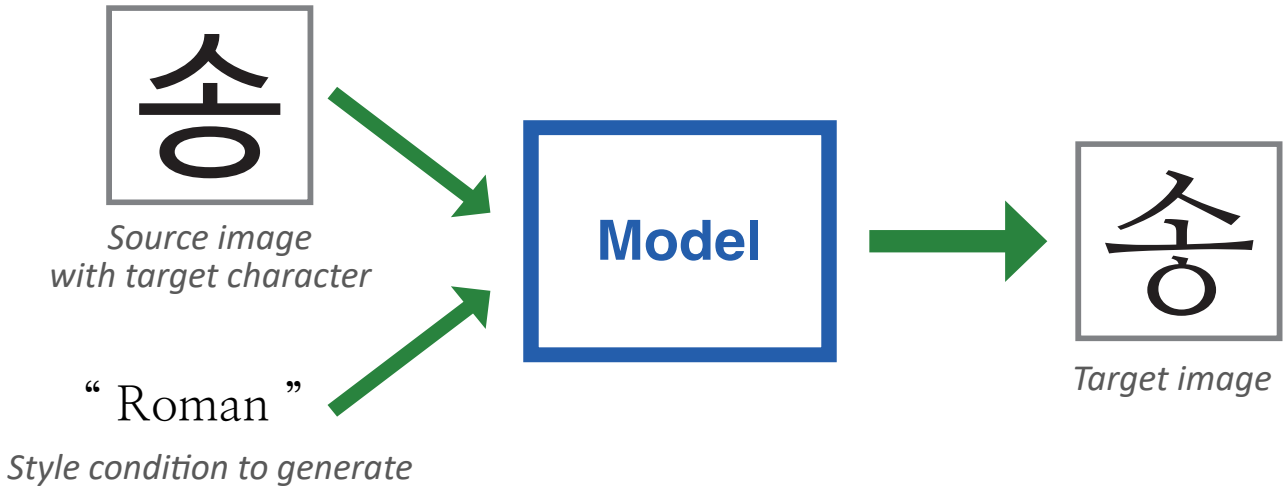
- Naïve image-to-image translation
- Providing additional character information

Generation in inference time

- Style-content separation
- Component-wise style representation

Fine-tuning based: image-to-image translation

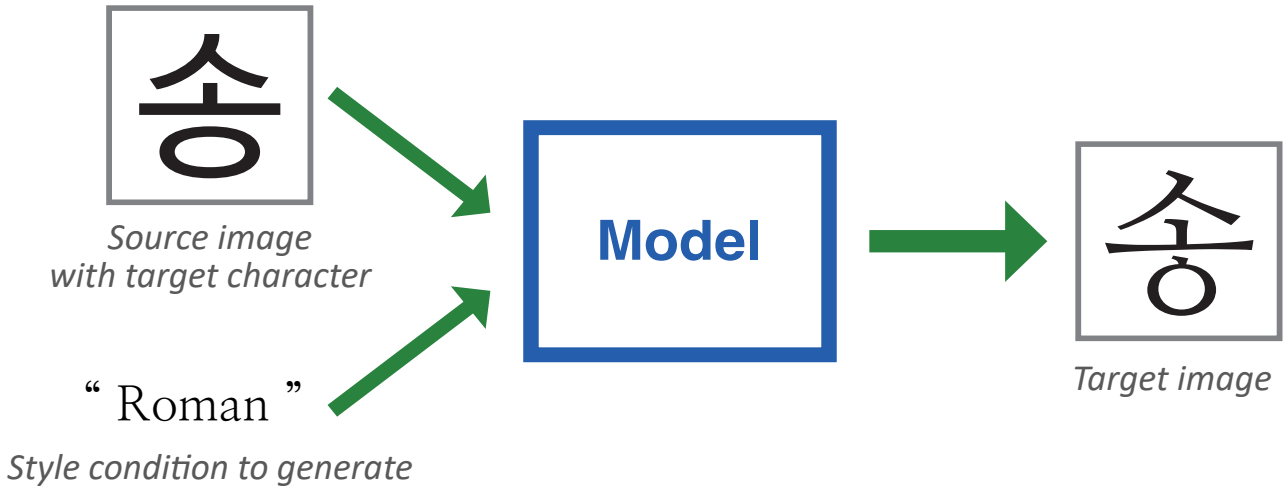
- **“Paired” image-to-image translation between “source” font to “target” font**



Issue:

Fine-tuning based: image-to-image translation

- **“Paired” image-to-image translation between “source” font to “target” font**



Issue:

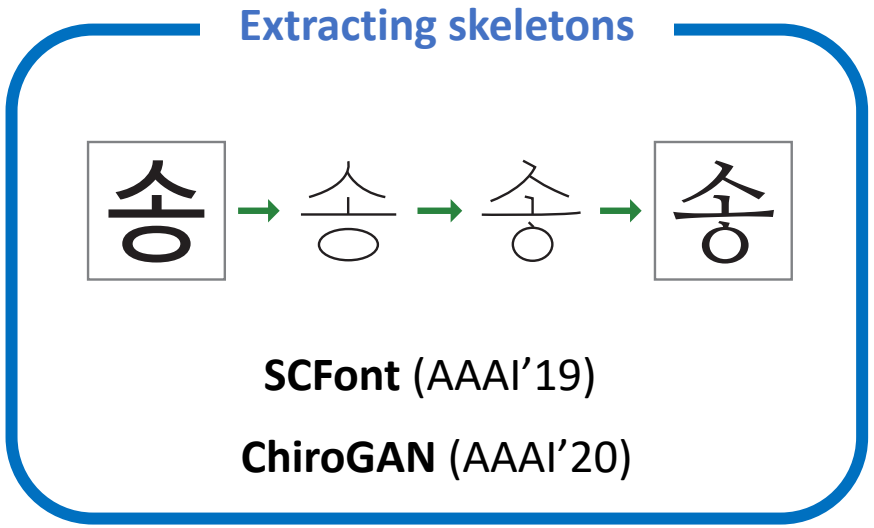
Highly complex structures! (송 ≠ 승)

Fine-tuning based: Additional character information

- **Handling “characters” with additional information**

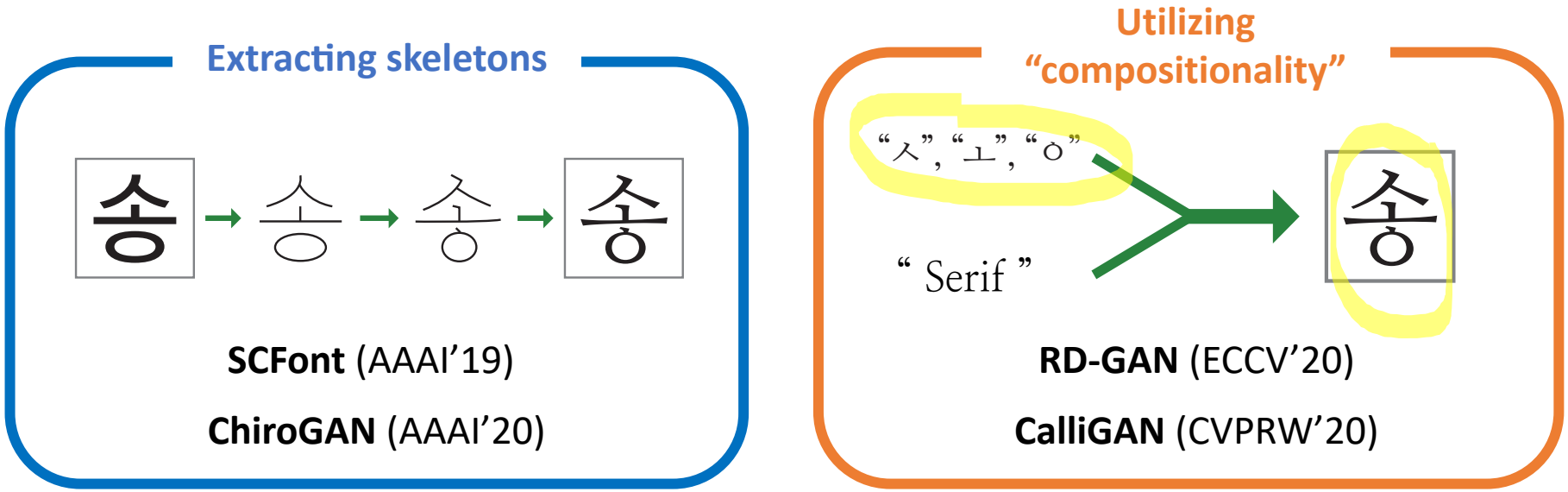
Fine-tuning based: Additional character information

- **Handling “characters” with additional information**



Fine-tuning based: Additional character information

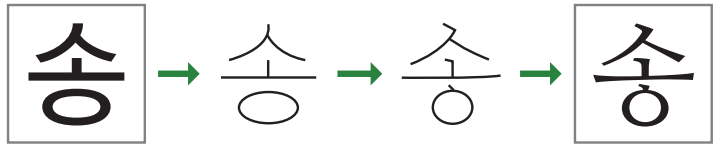
- **Handling "characters" with additional information**



Fine-tuning based: Additional character information

- **Handling "characters" with additional information**

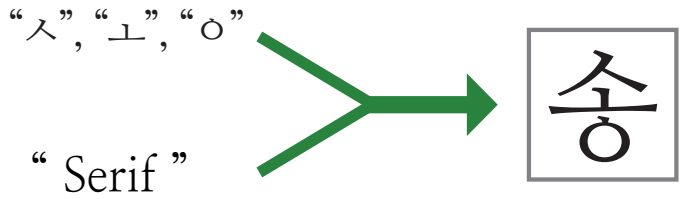
Extracting skeletons



SCFont (AAAI'19)

ChiroGAN (AAAI'20)

Utilizing "compositionality"

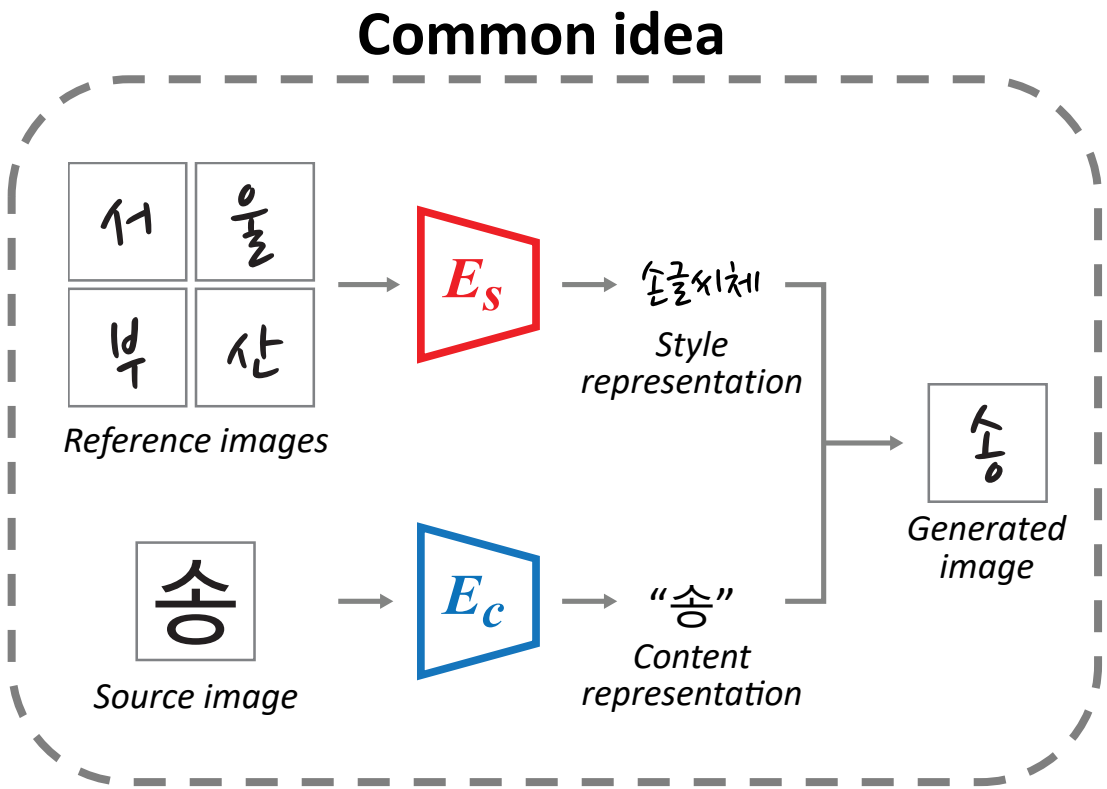


RD-GAN (ECCV'20)

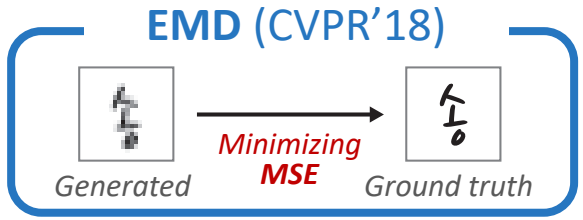
CalliGAN (CVPRW'20)

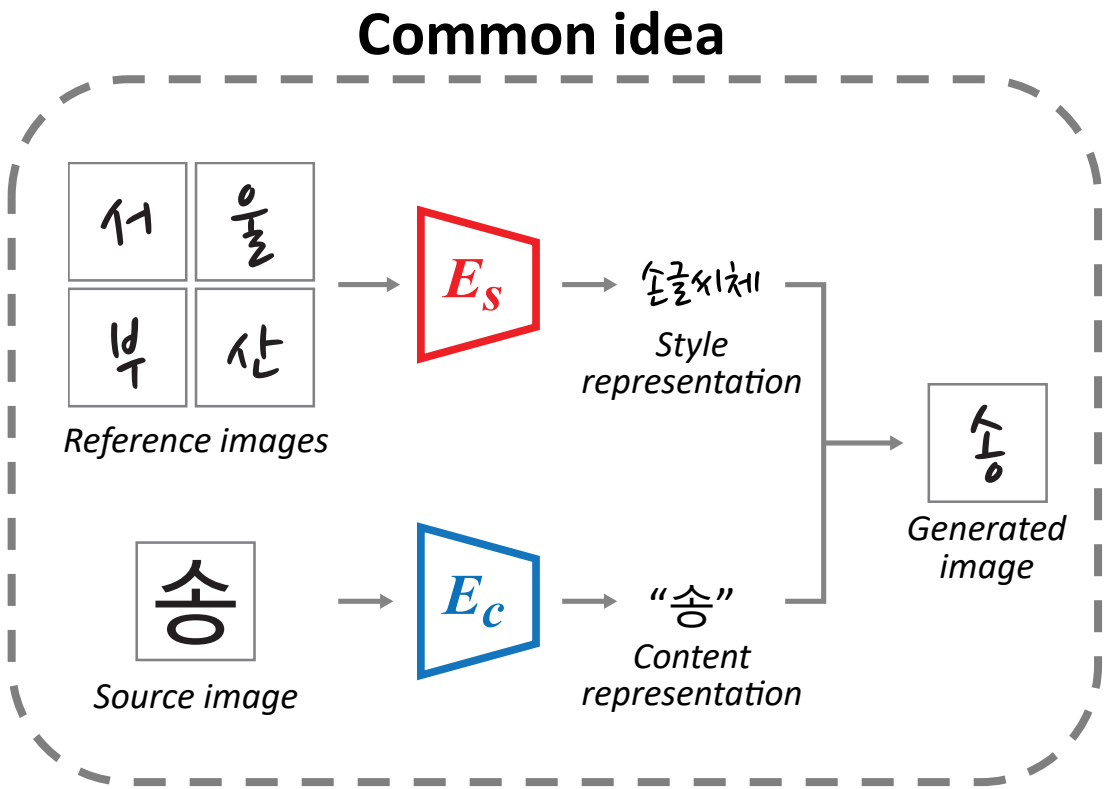
- **Too many source-target pairs** are required for fine-tuning. (> a few hundreds)
- Unsuitable for application on low-resource devices, e.g., mobile devices

Without additional training: style-content separation from glyphs



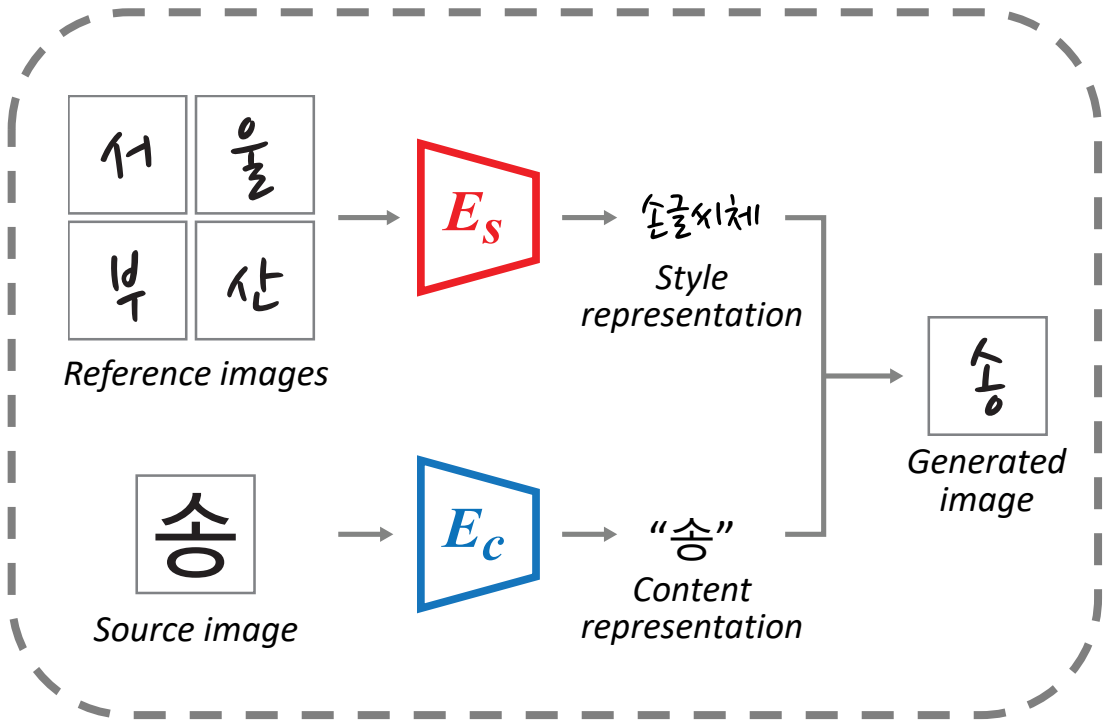
Without additional training: style-content separation from glyphs





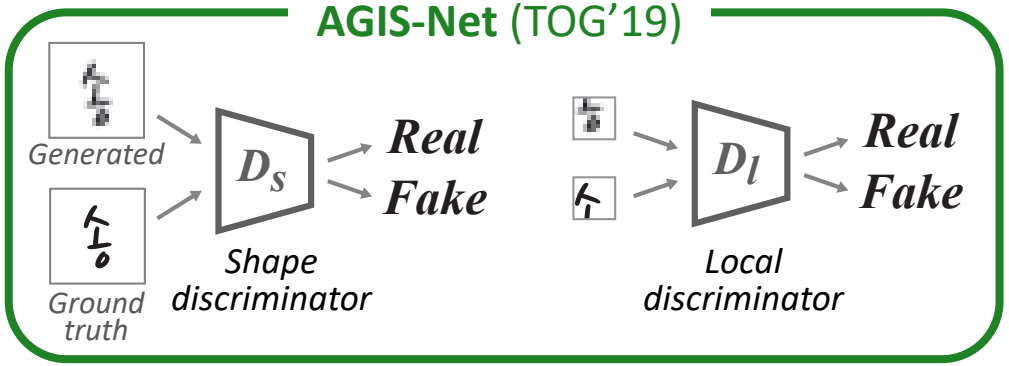
Without additional training: style-content separation from glyphs

Common idea



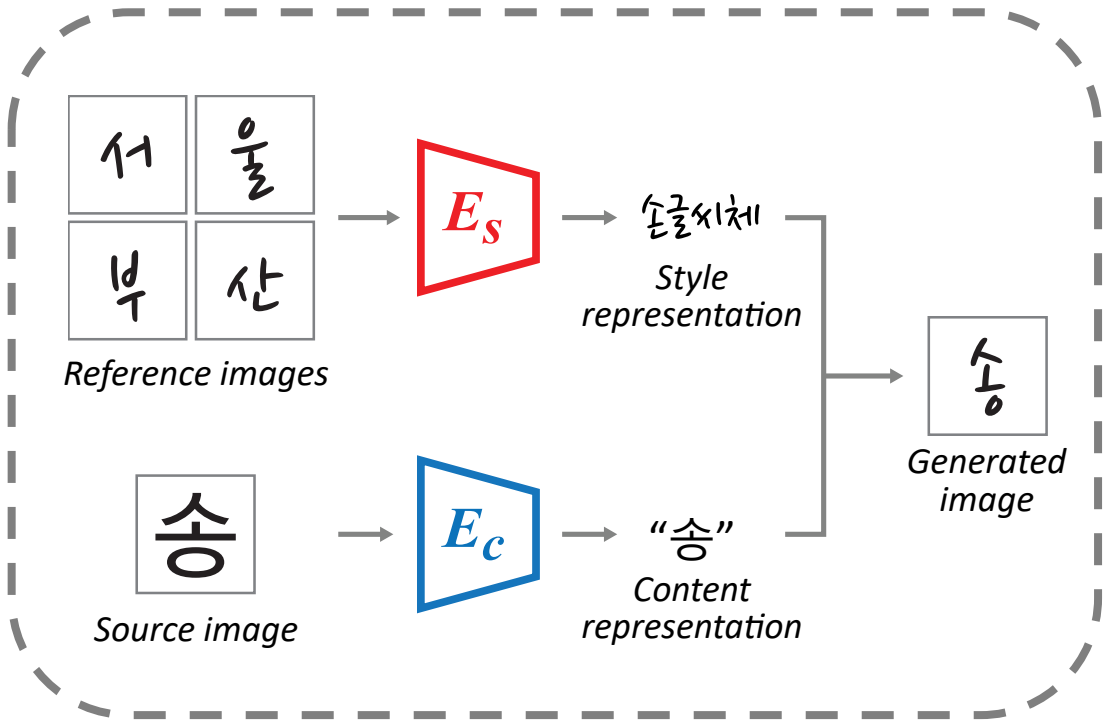
EMD (CVPR'18)

AGIS-Net (TOG'19)



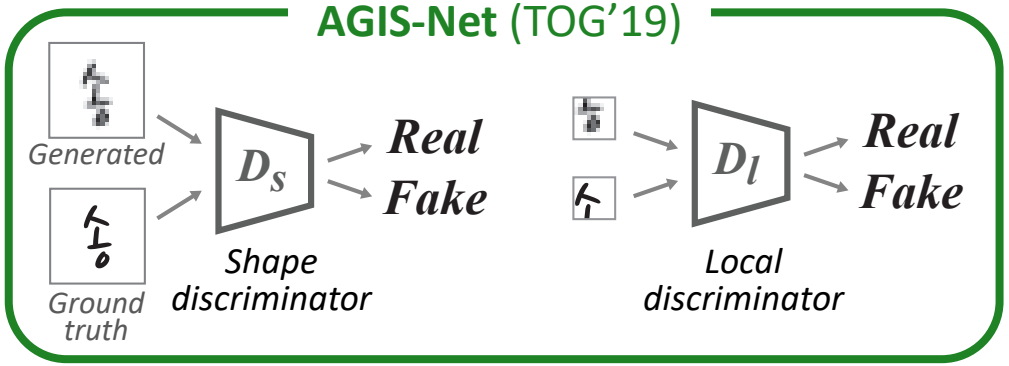
Without additional training: style-content separation from glyphs

Common idea

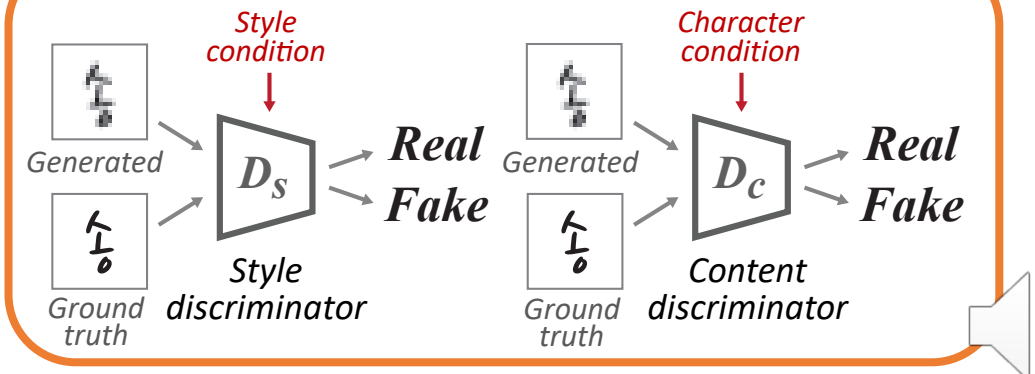


EMD (CVPR'18)

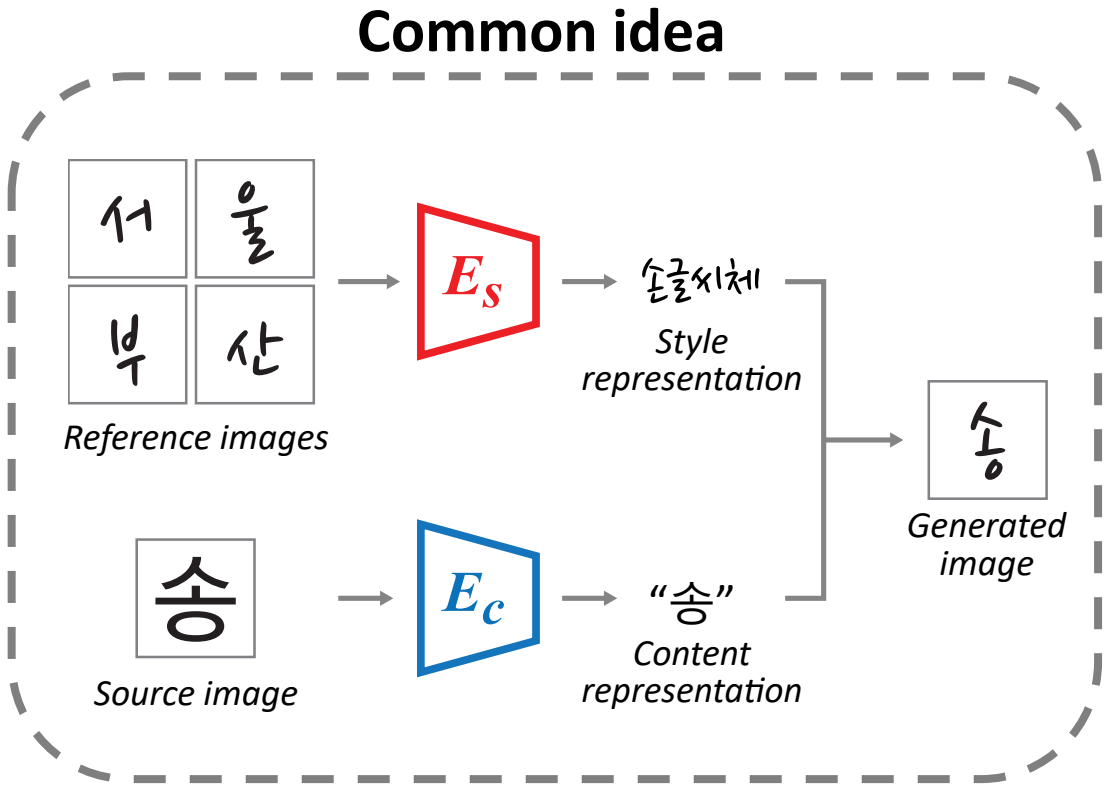
AGIS-Net (TOG'19)



FUNIT (*modified for font*, ICCV'19)



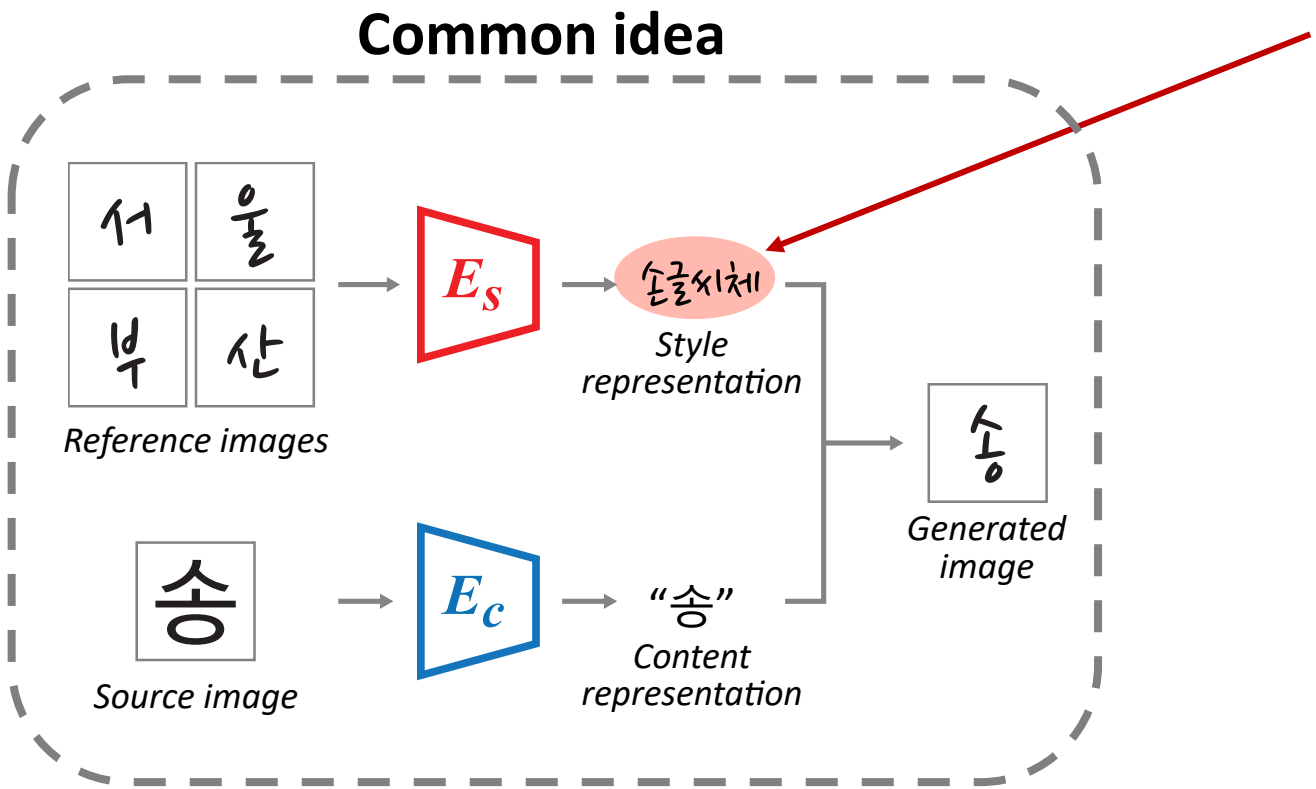
“Universal style representation” is insufficient!



Source	博	城	城	荷
EMD	博	城	城	荷
AGIS-Net	博	城	城	荷
FUNIT	博	城	城	荷
GT	博	城	城	荷

Failure to capture detailed styles!

“Universal style representation” is insufficient!

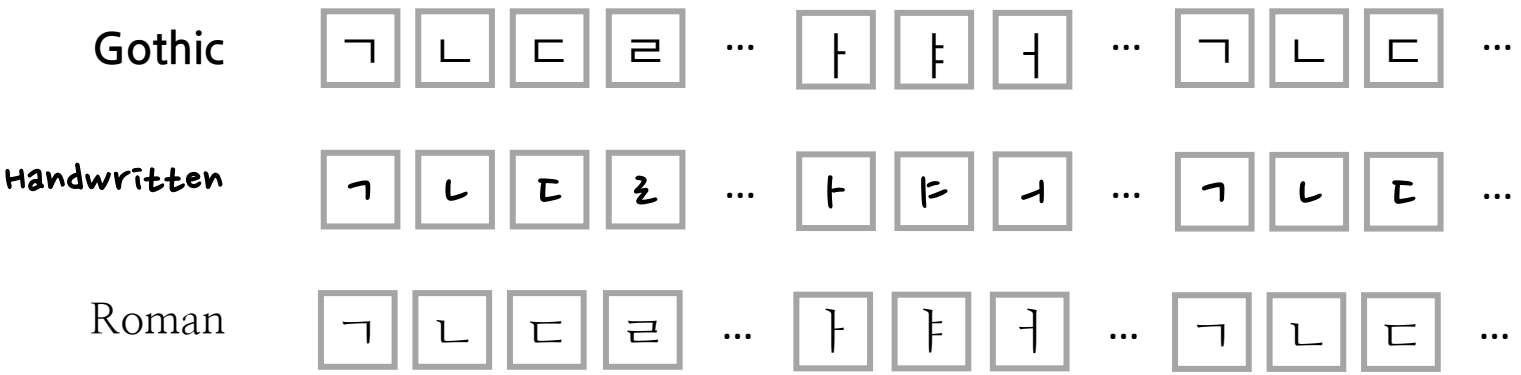


A single representation is insufficient to represent the complex “font style”.

Source	博	城	城	荷
EMD	博	城	城	荷
AGIS-Net	博	城	城	荷
FUNIT	博	城	城	荷
GT	博	城	城	荷

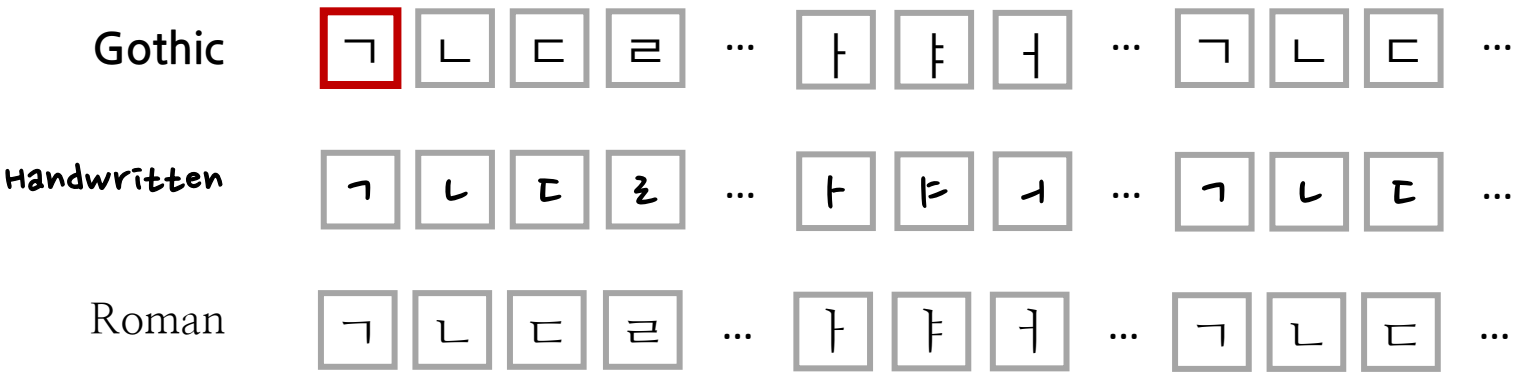
Failure to capture detailed styles!

Without additional training: component-wise style representation



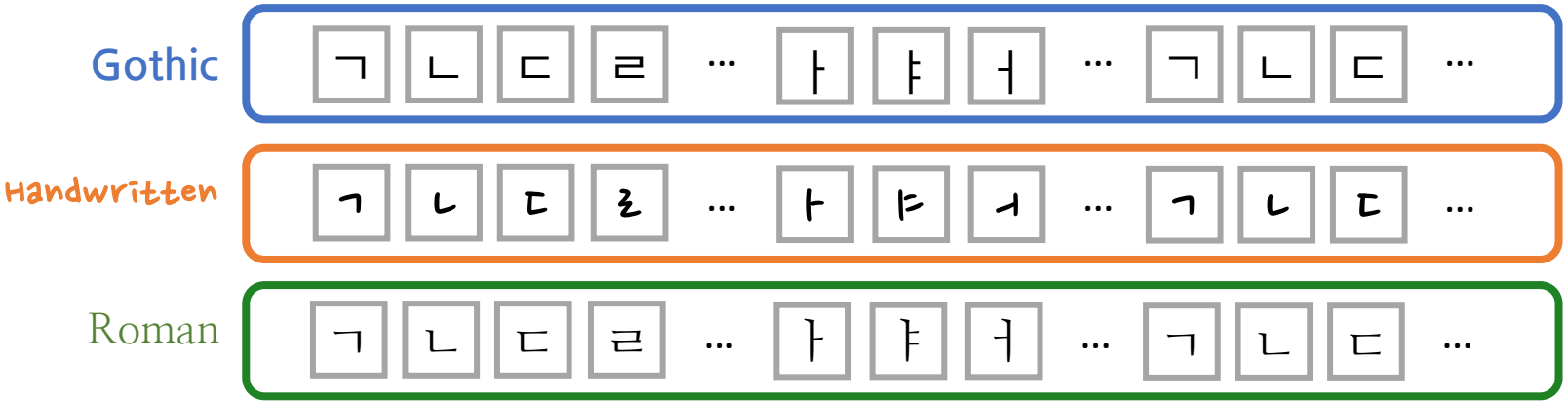
- Defining the “style” as component-wisely.
- A “style” corresponds to a set of component-wise style representations.
- A Korean glyph image corresponds to a set of 3 component-wise style representations.

Without additional training: component-wise style representation



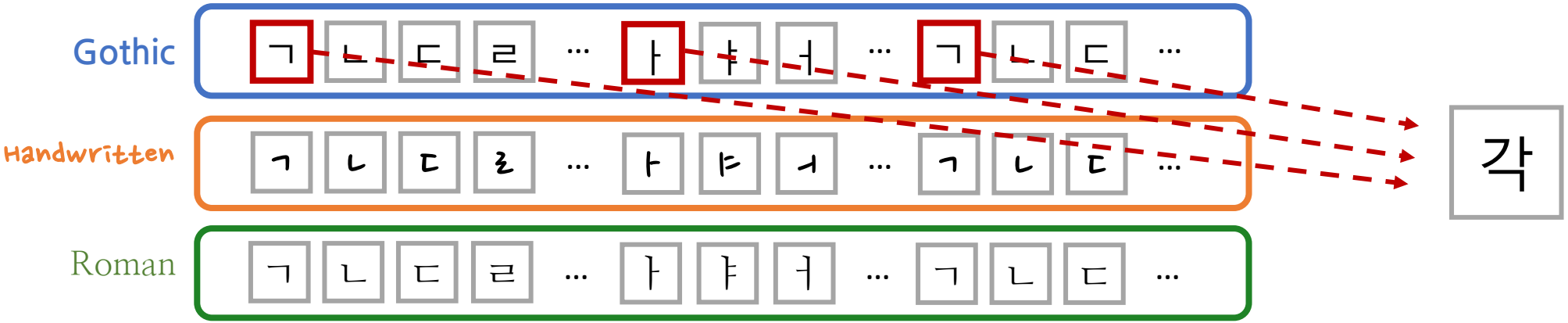
- Defining the “style” as component-wisely.
- A “style” corresponds to a set of component-wise style representations.
- A Korean glyph image corresponds to a set of 3 component-wise style representations.

Without additional training: component-wise style representation



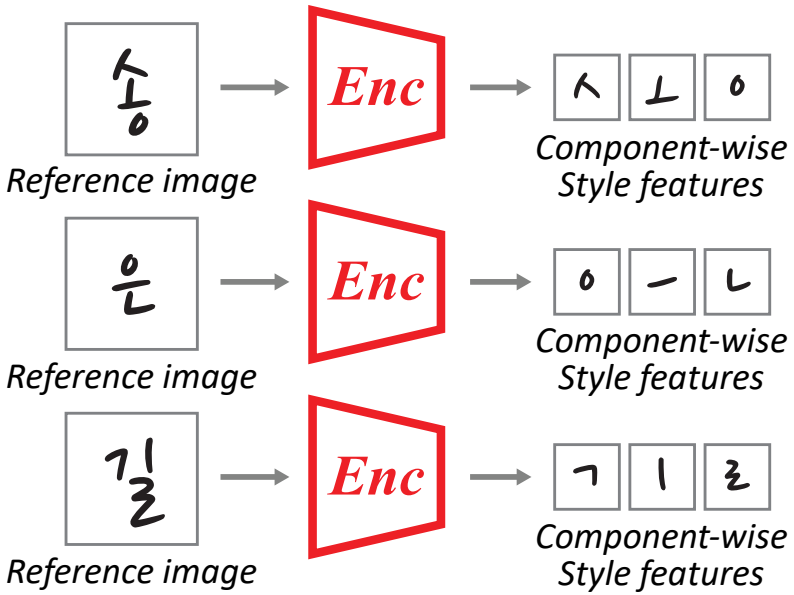
- Defining the “style” as component-wisely.
- A “style” corresponds to a set of component-wise style representations.
- A Korean glyph image corresponds to a set of 3 component-wise style representations.

Without additional training: component-wise style representation



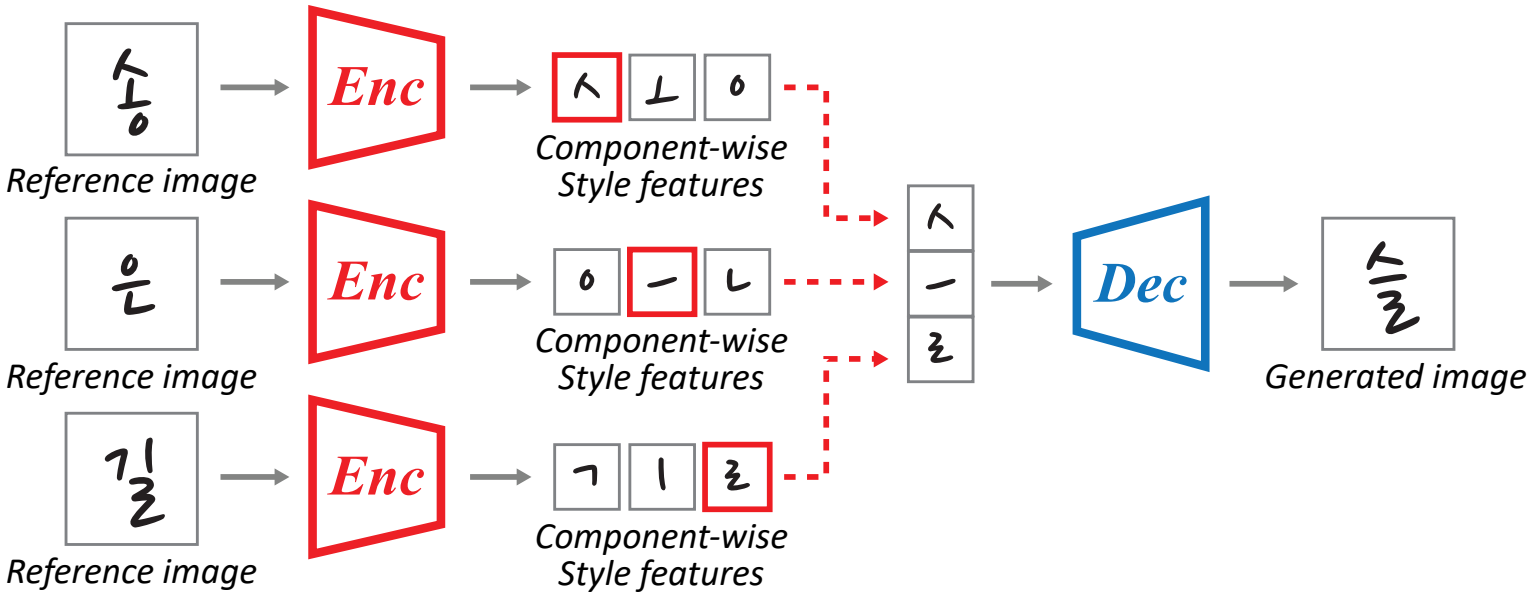
- Defining the “style” as component-wisely.
- A “style” corresponds to a set of component-wise style representations.
- A Korean glyph image corresponds to a set of 3 component-wise style representations.

Without additional training: component-wise style representation



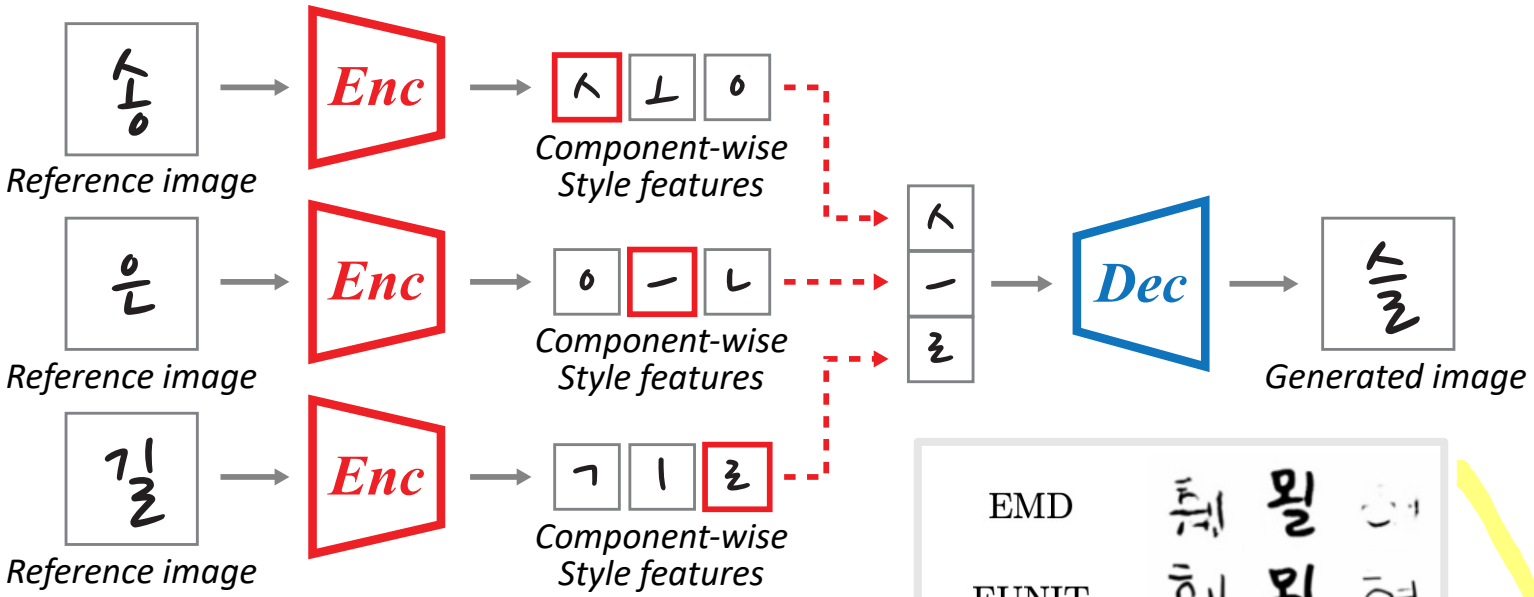
- Disassembling a Korean glyph image to **3** component-wise style representations.

Without additional training: component-wise style representation



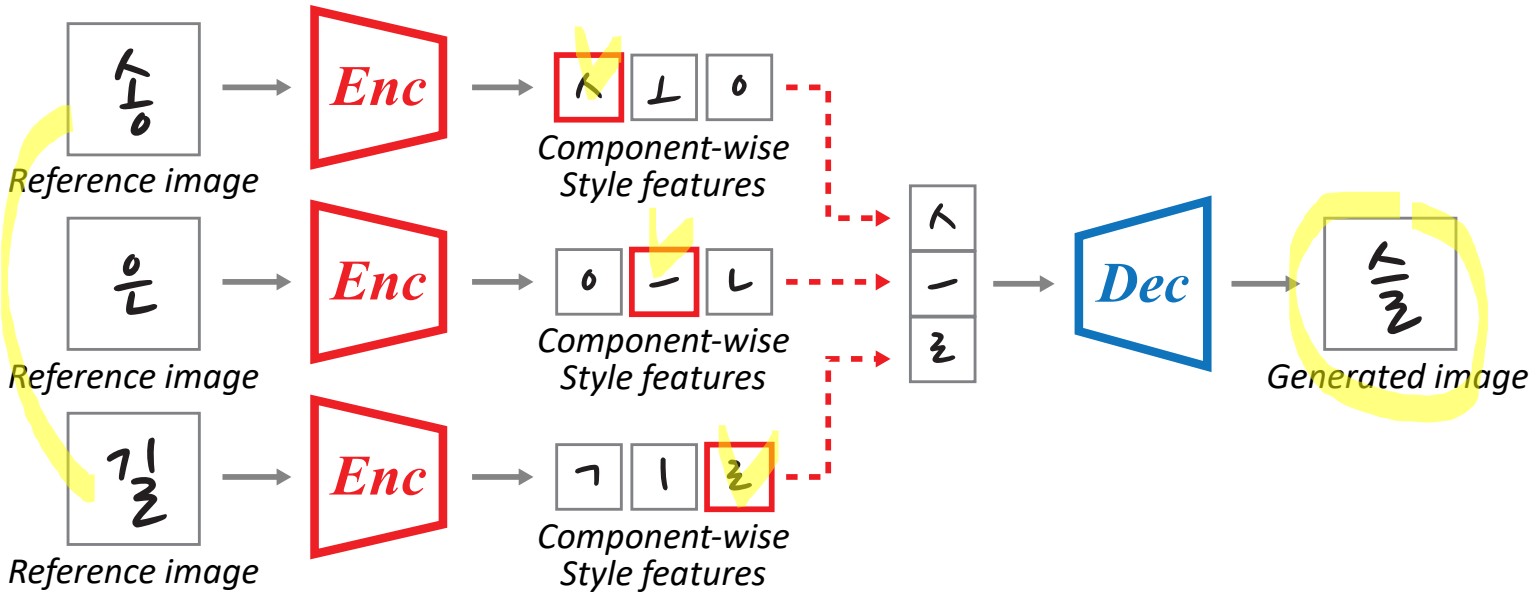
- Disassembling a Korean glyph image to **3** component-wise style representations.
- Re-assembling the components extracted from reference glyphs to generate the target glyph.

Without additional training: component-wise style representation



- Disassembling a Korean glyph image to **3** component-wise style r
- Re-assembling the components extracted from reference glyphs t .

Without additional training: component-wise style representation



- Disassembling a Korean glyph image to **3** component-wise style representations.
- Re-assembling the components extracted from reference glyphs to generate the target glyph.
- **Limitations**
  - Restricted to ***"complete compositional scripts"*** like Korean, Thai.
  - Requiring **all the existing components** to generate full font library.

Milestone

- **Limitations of existing works**:
  - **Universal style representation** methods: **Failure to capture the detailed styles**
  - **Component-wise style representation** method: **Requiring the entire components as the reference**
- **Idea:** Defining **“multiple localized”** style representations using the **“compositionality”**
- Enabling to define the component-wise style representations in Chinese script
- Handling the missing component-wise style representations

Milestone

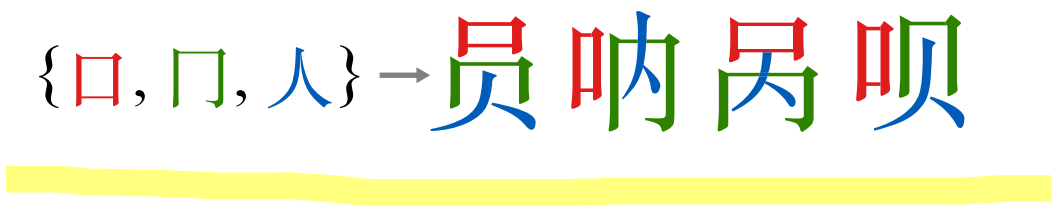
- **Limitations of existing works**:
  - **Universal style representation** methods: **Failure to capture the detailed styles**
  - **Component-wise style representation** method: **Requiring the entire components as the reference**
- **Idea:** Defining **"multiple localized"** style representations using the **"compositionality"**
- Enabling to define the component-wise style representations in Chinese script
- Handling the missing component-wise style representations

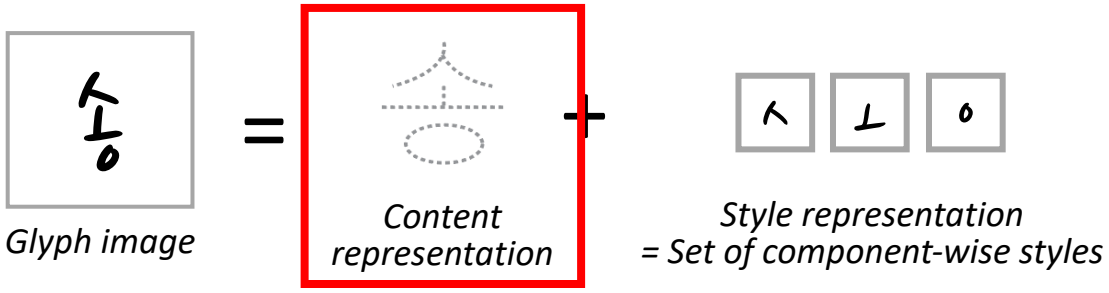
Milestone

- **Limitations of existing works**:
  - **Universal style representation** methods: **Failure to capture the detailed styles**
  - **Component-wise style representation** method: **Requiring the entire components as the reference**
- **Idea:** Defining **“multiple localized”** style representations using the **“compositionality”**
- Enabling to define the component-wise style representations in Chinese script
- Handling the missing component-wise style representations

Key idea

- **Extending DM-Font to the Chinese scripts**

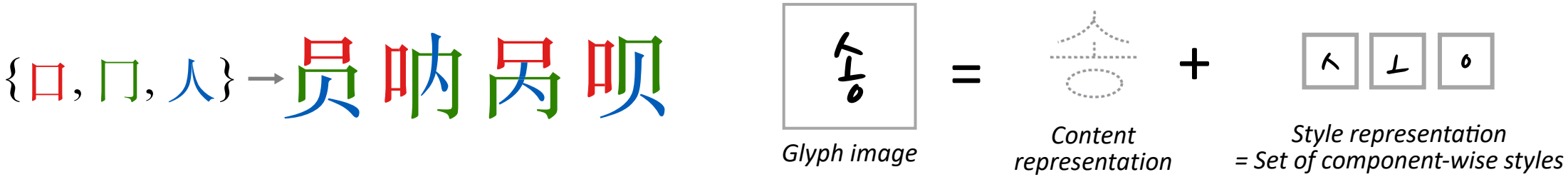




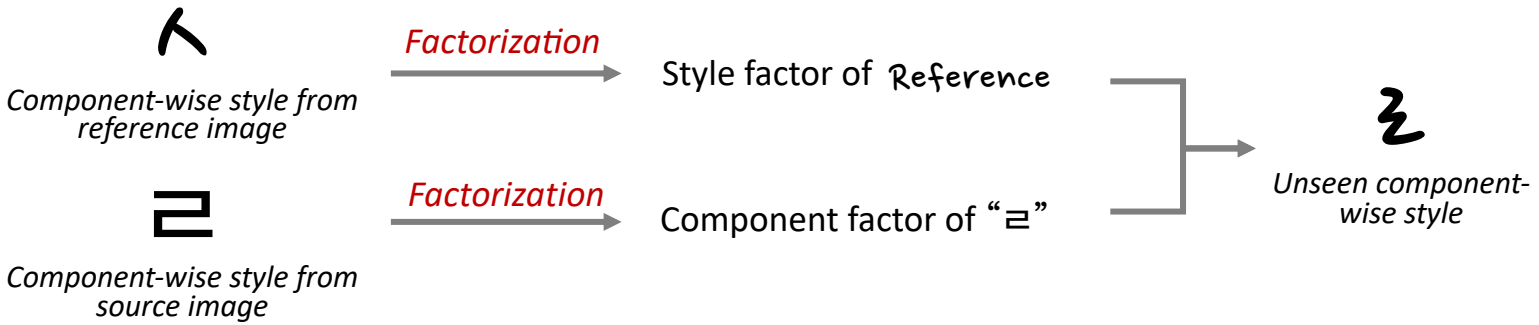
- **Generating unseen component-wise styles by factorization.**

Key idea

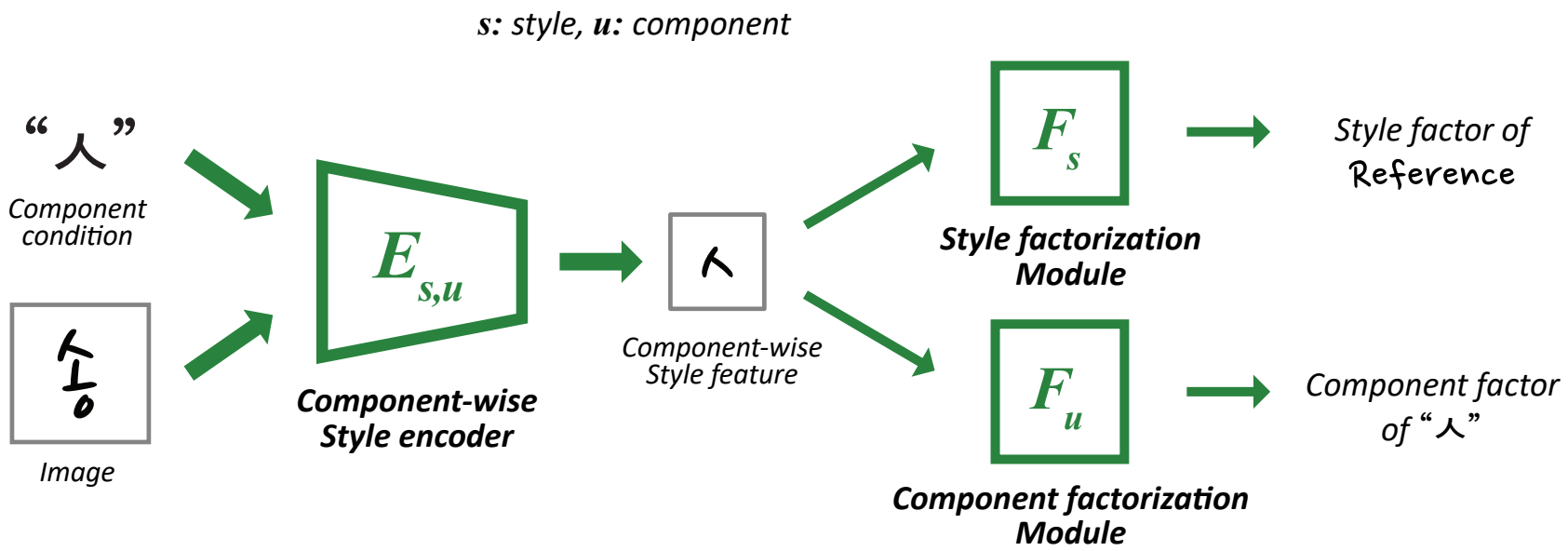
- **Extending DM-Font to the Chinese scripts**



- **Generating unseen component-wise styles by factorization.**

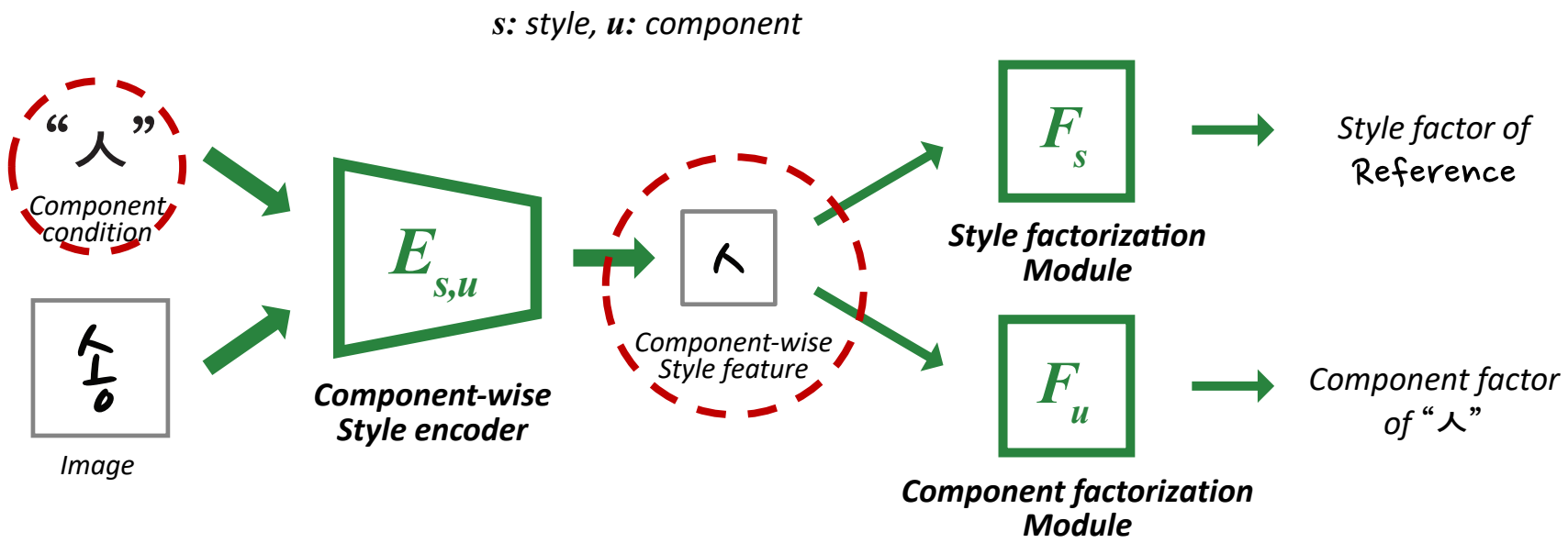


Modules to derive component-wise features



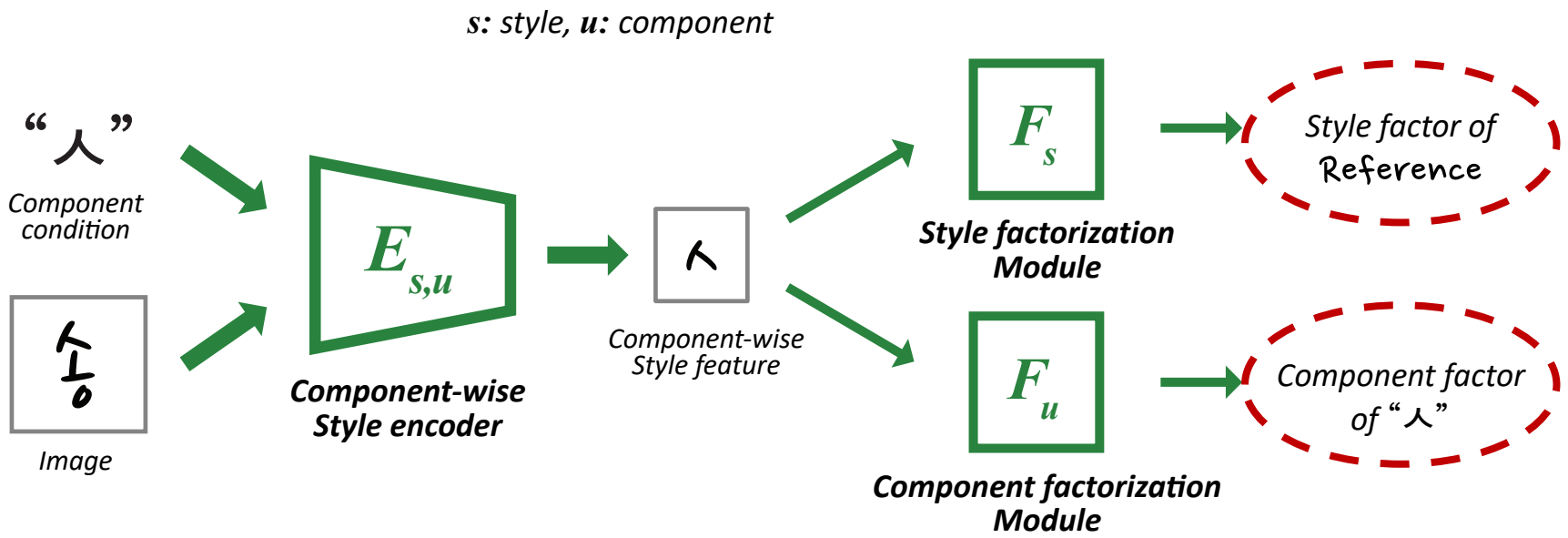
- **Component-wise Style Encoder $E_{s,u}$**
  - Encodes an image to a component-wise style feature which corresponds to given component condition.
- **Factorization Modules F_s , F_u**
  - Factorizes a component-wise style feature into a style factor and a component factor.

Modules to derive component-wise features



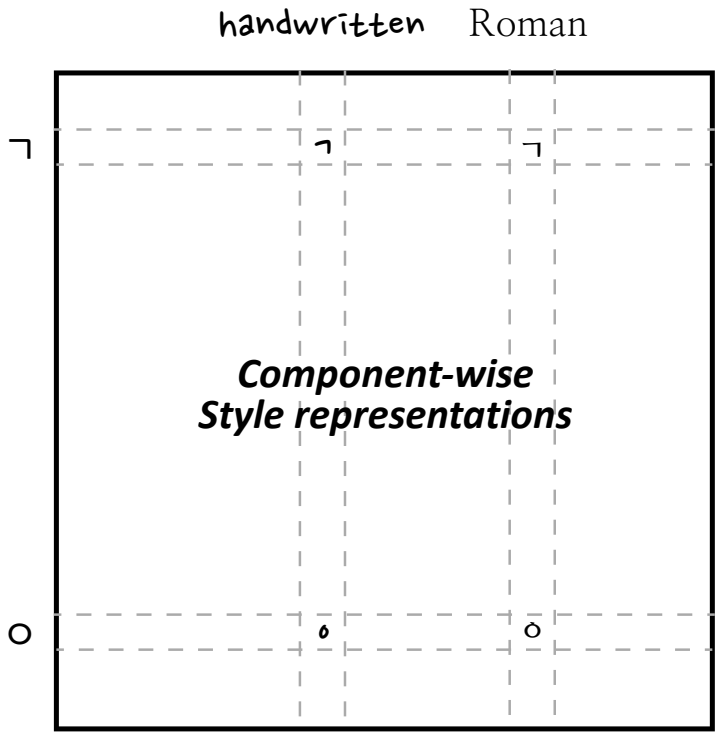
- **Component-wise Style Encoder $E_{s,u}$**
  - Encodes an image to a component-wise style feature which corresponds to given component condition.
- **Factorization Modules F_s , F_u**
  - Factorizes a component-wise style feature into a style factor and a component factor.

Modules to derive component-wise features



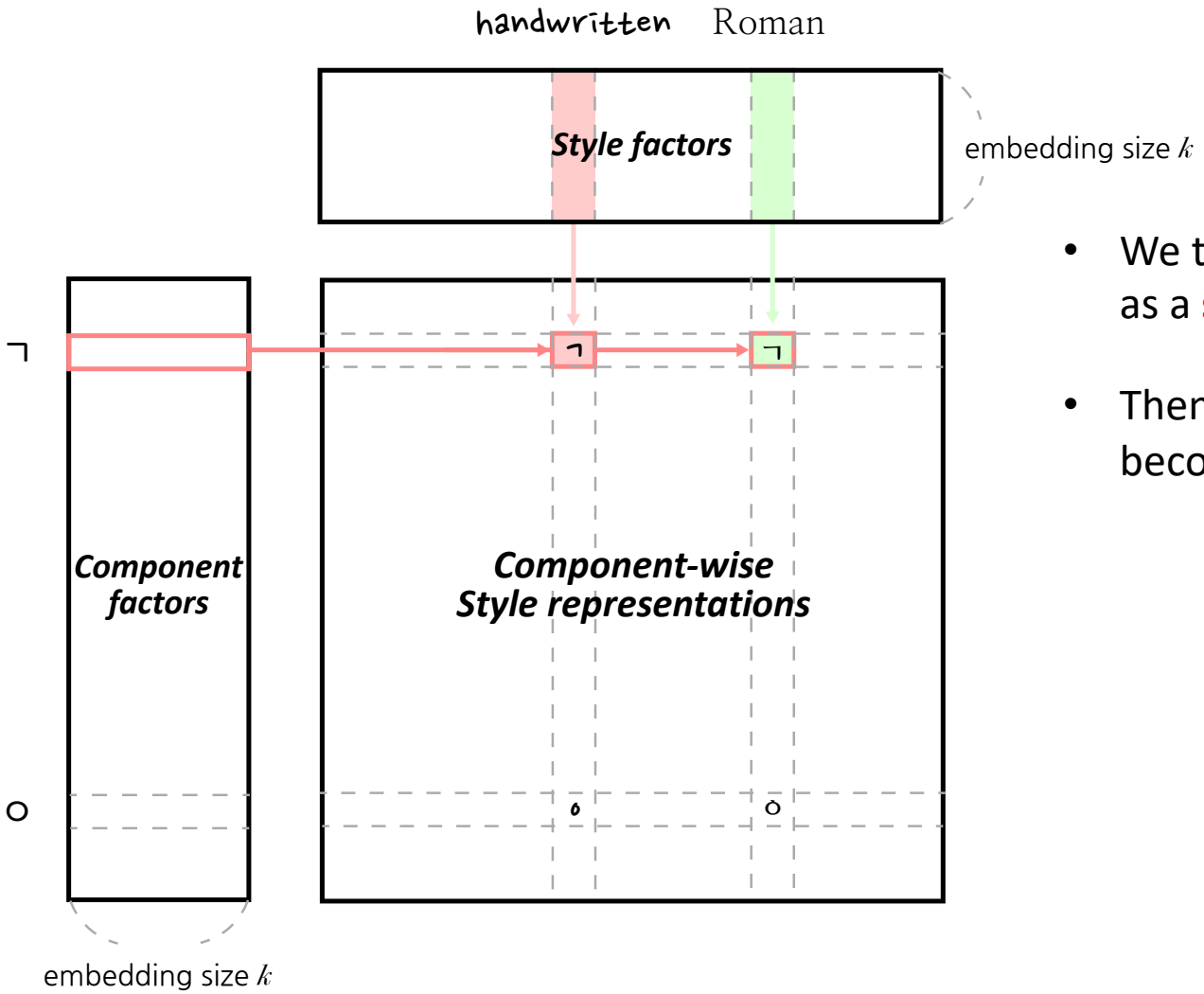
- **Component-wise Style Encoder $E_{s,u}$**
  - Encodes an image to a component-wise style feature which corresponds to given component condition.
- **Factorization Modules F_s , F_u**
  - Factorizes a component-wise style feature into a style factor and a component factor.

Deriving the missing components by factorization



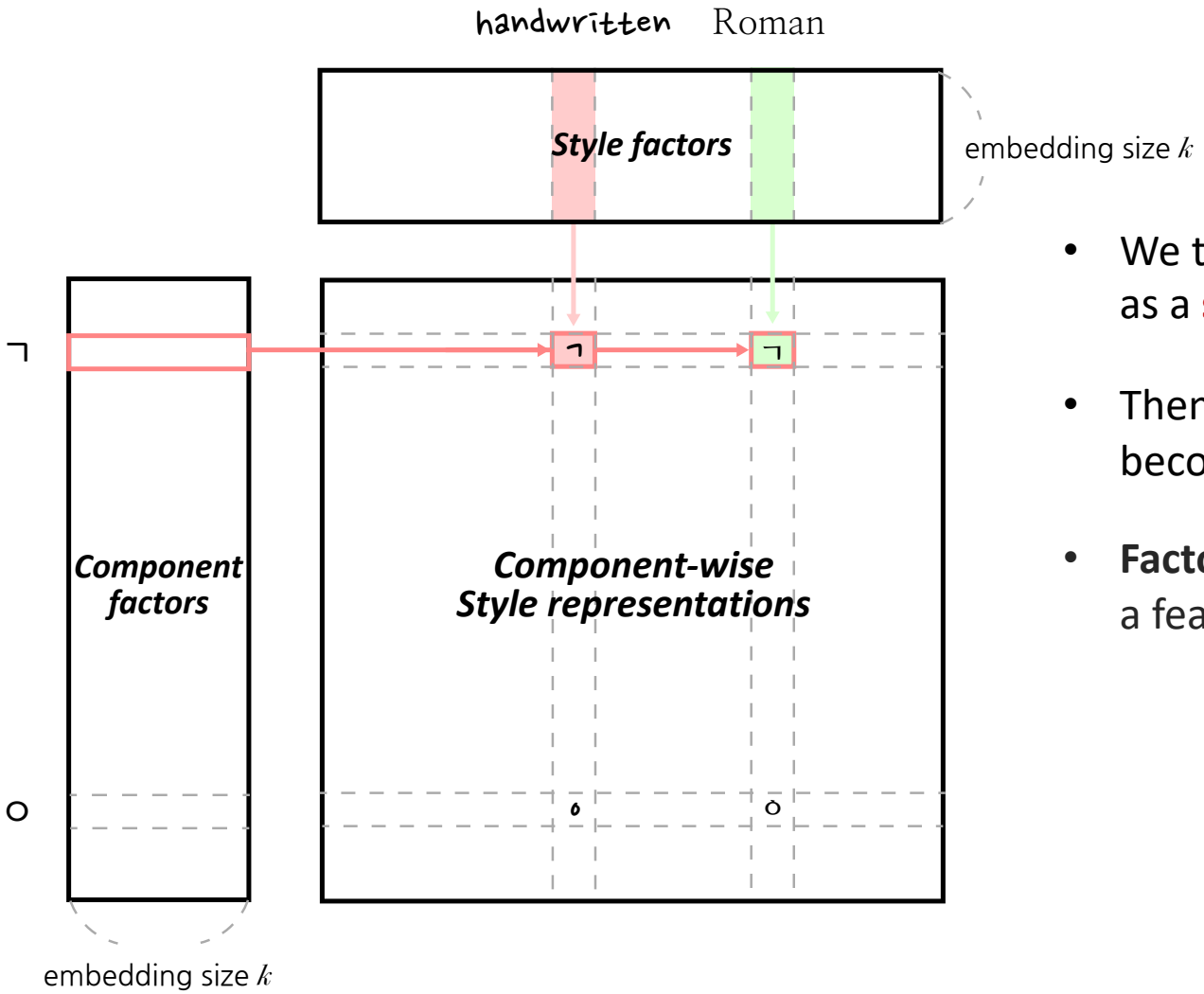
- We treat the given component-wise style representation set as a **sparse matrix!**
- Then, deriving missing component-wise style representations becomes a **matrix completion problem.**

Deriving the missing components by factorization



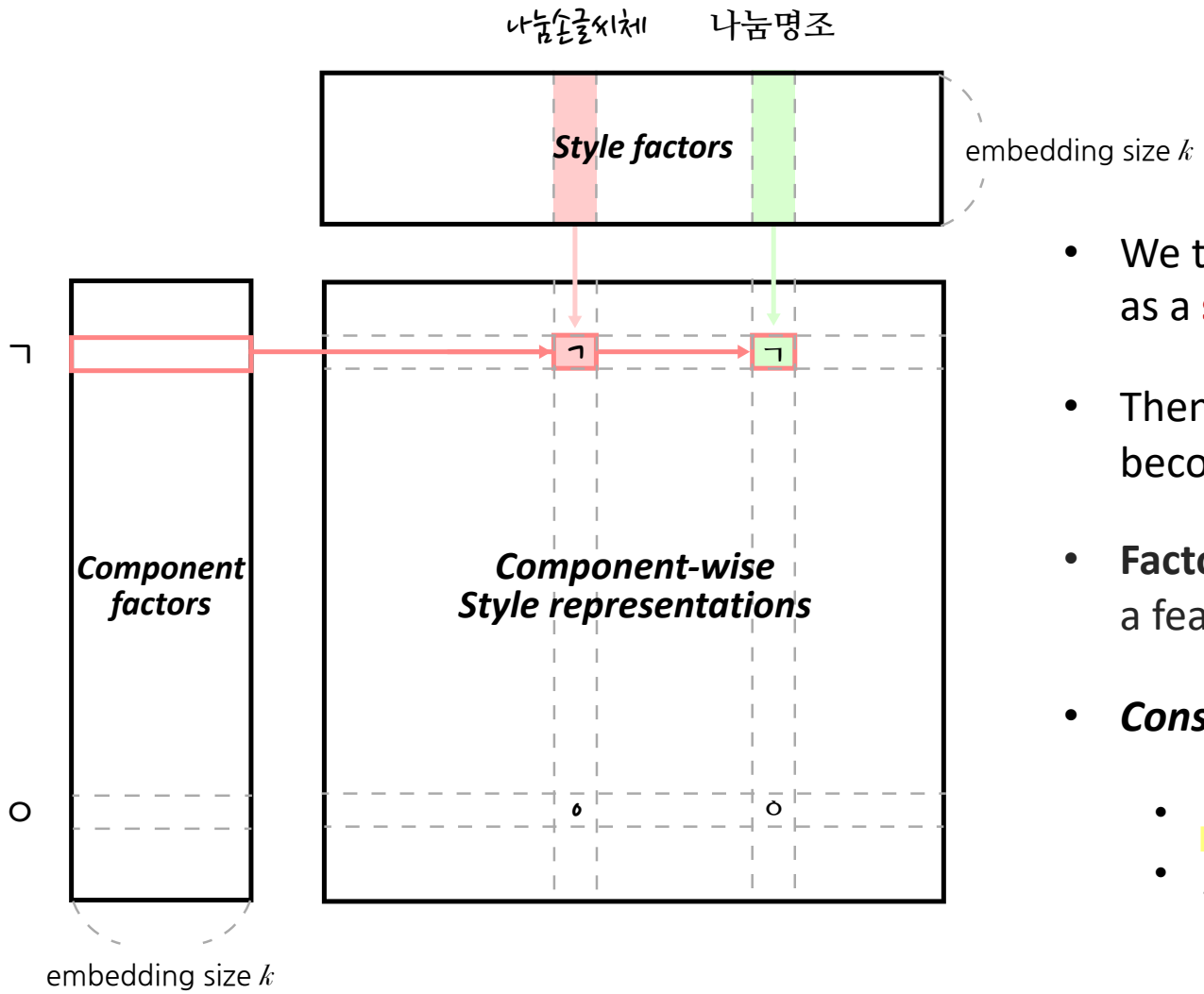
- We treat the given component-wise style representation set as a **sparse matrix!**
- Then, deriving missing component-wise style representations becomes a **matrix completion problem.**

Deriving the missing components by factorization



- We treat the given component-wise style representation set as a **sparse matrix!**
- Then, deriving missing component-wise style representations becomes a **matrix completion problem.**
- **Factorization modules** F_s, F_u extends a feature with shape D to a factor with shape (k, D).

Deriving the missing components by factorization

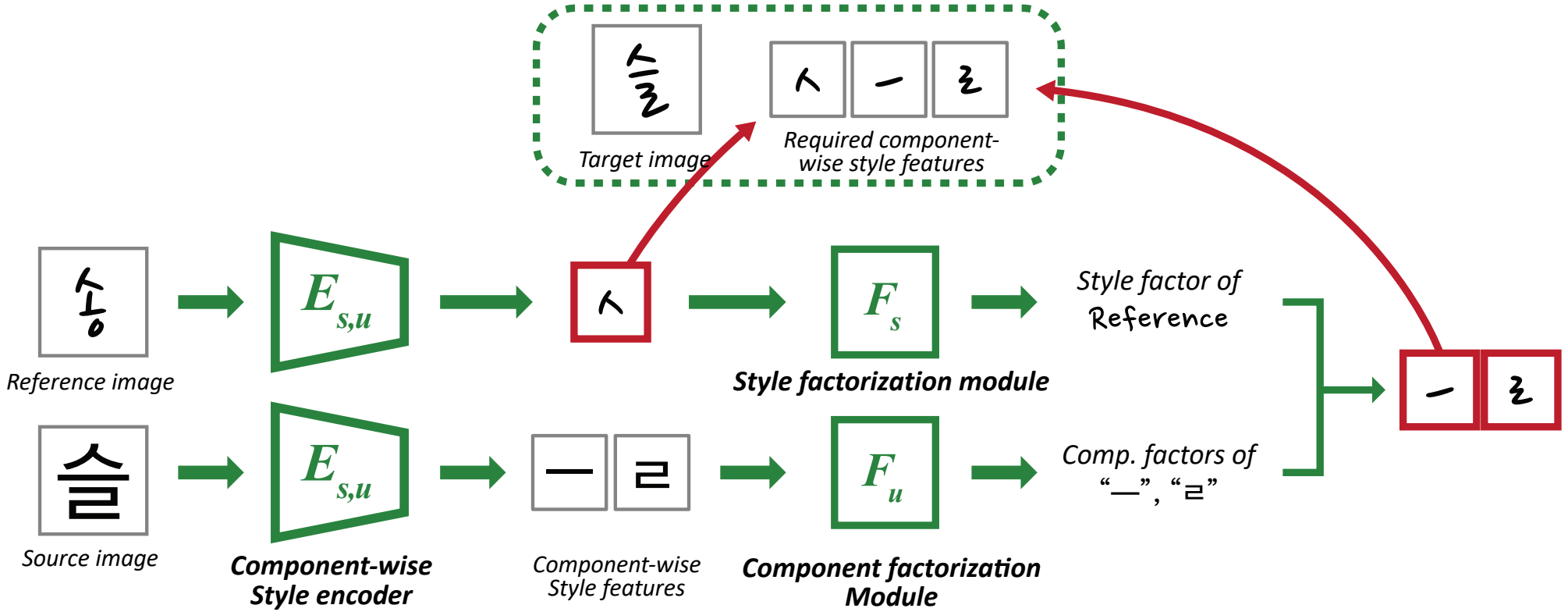


- We treat the given component-wise style representation set as a **sparse matrix!**
- Then, deriving missing component-wise style representations becomes a **matrix completion problem.**
- **Factorization modules** F_s, F_u extends a feature with shape D to a factor with shape (k, D).
- ***Constraint***
  - $F_s(f_{s,u}) = F_s(f_{s,u'}),\ F_u(f_{s,u}) = F_u(f_{s',u})$
  - $f_{s,u} = z_s \cdot z_u$

$f_{s,u}$: Component-wise style feat with style s, compone[illegible] u

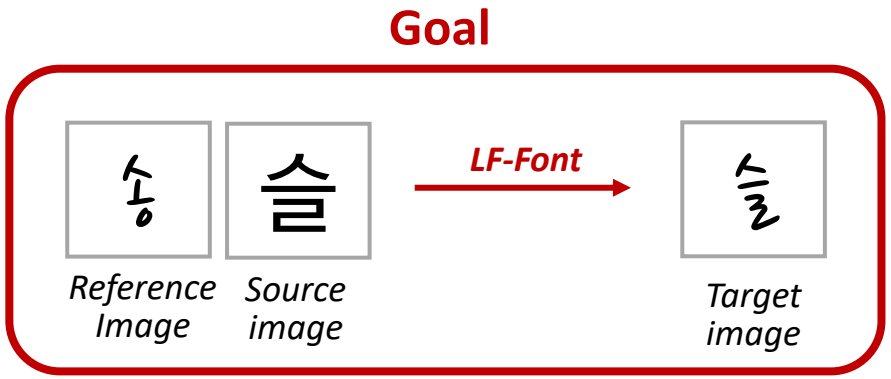
z_s, z_u : Factor of style s, compo[illegible] u

Deriving target component-wise features

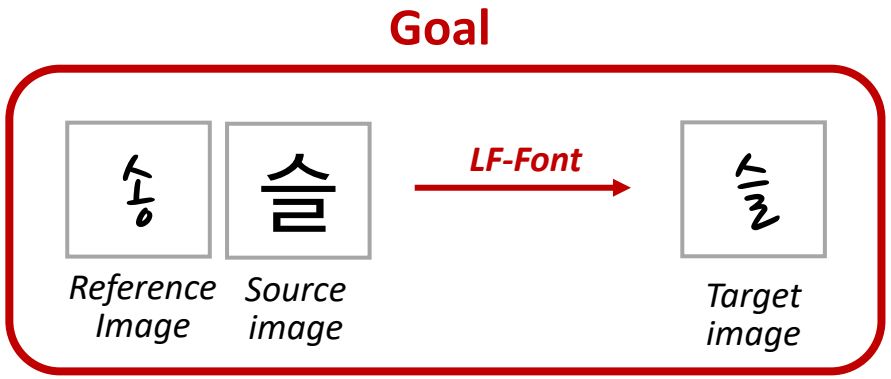


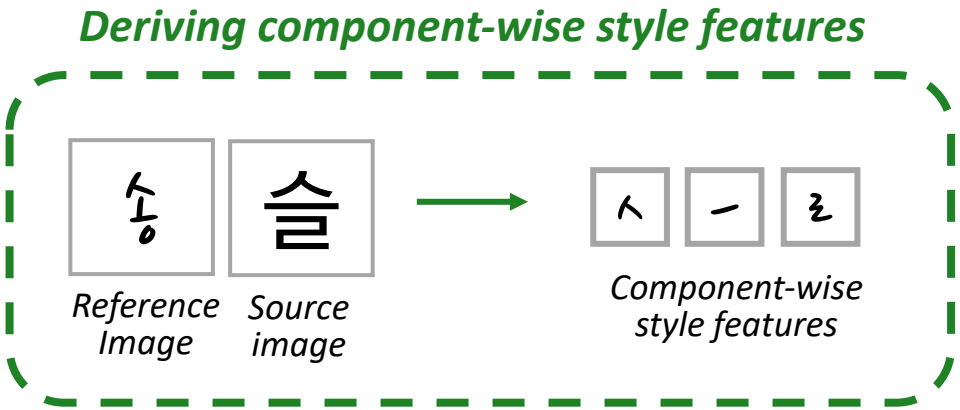
- The "**reconstructed**" component-wise style features should work similarly with the "**original**" component-wise style features.
- The style (or component) factors should contain information of corresponding style (or component).

Generation

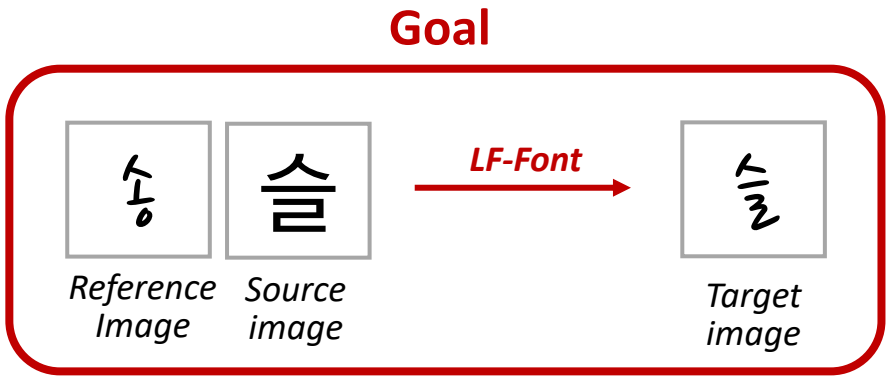


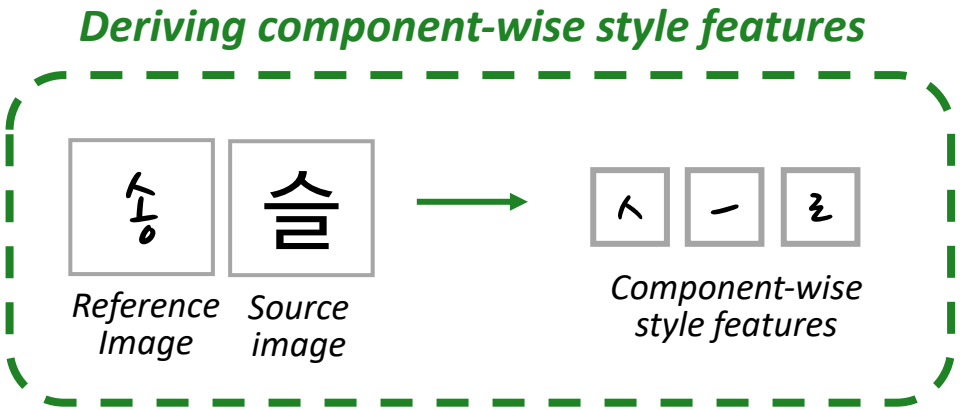
Generation

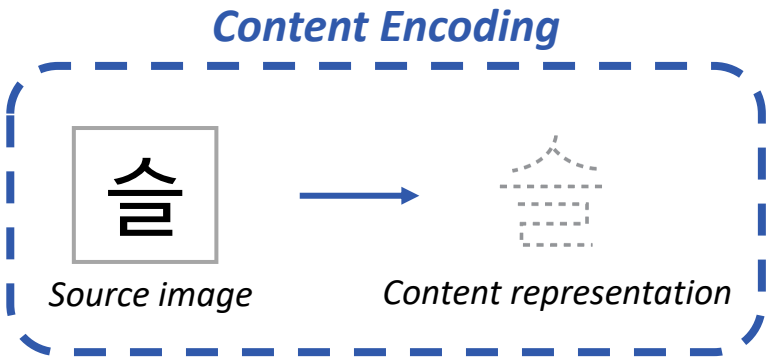

Goal
LF-Font
Reference Image
Source image
Target image


Deriving component-wise style features
Reference Image
Source image
Component-wise style features

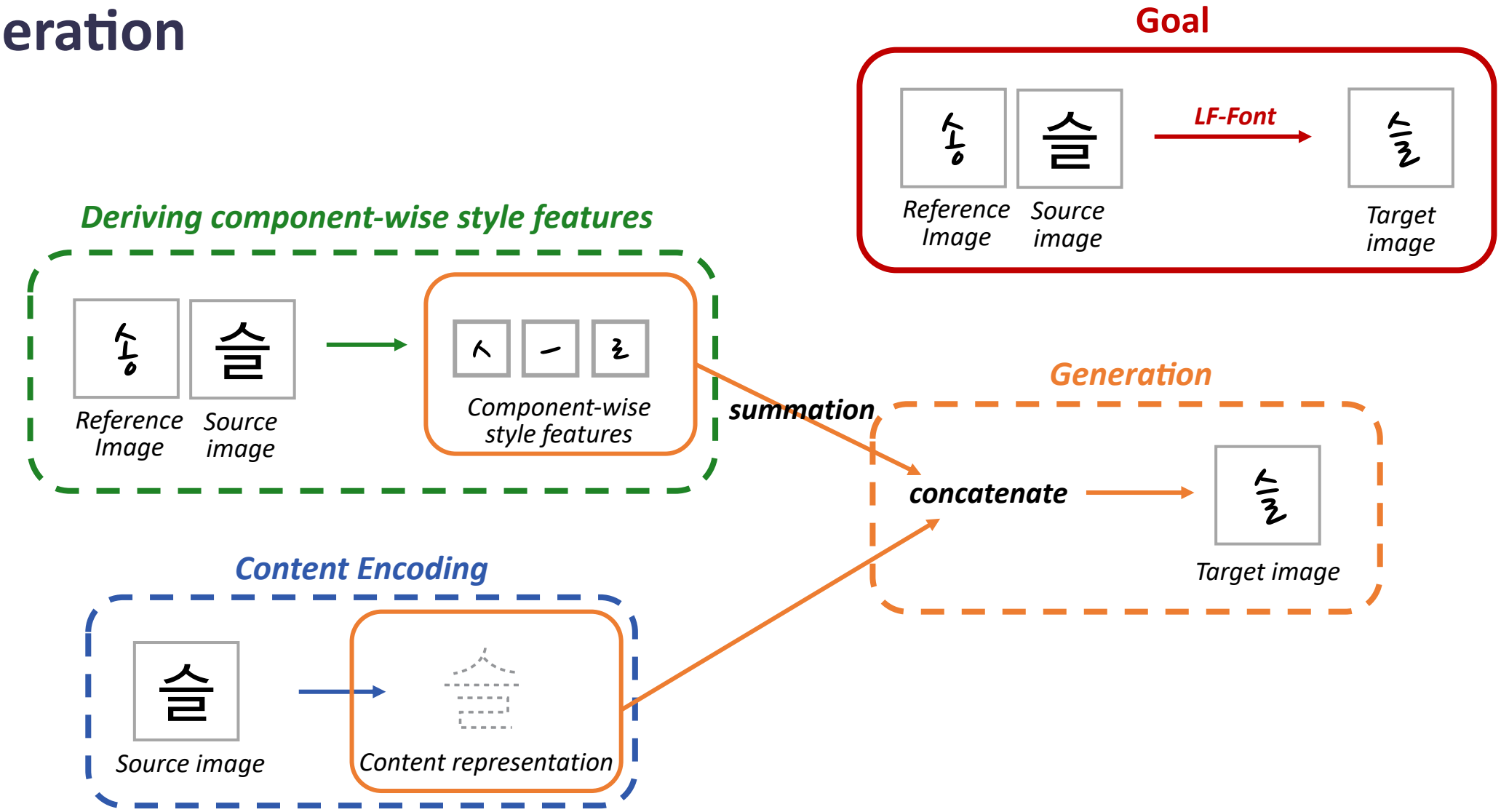
Generation


Goal
LF-Font
Reference Image
Source image
Target image

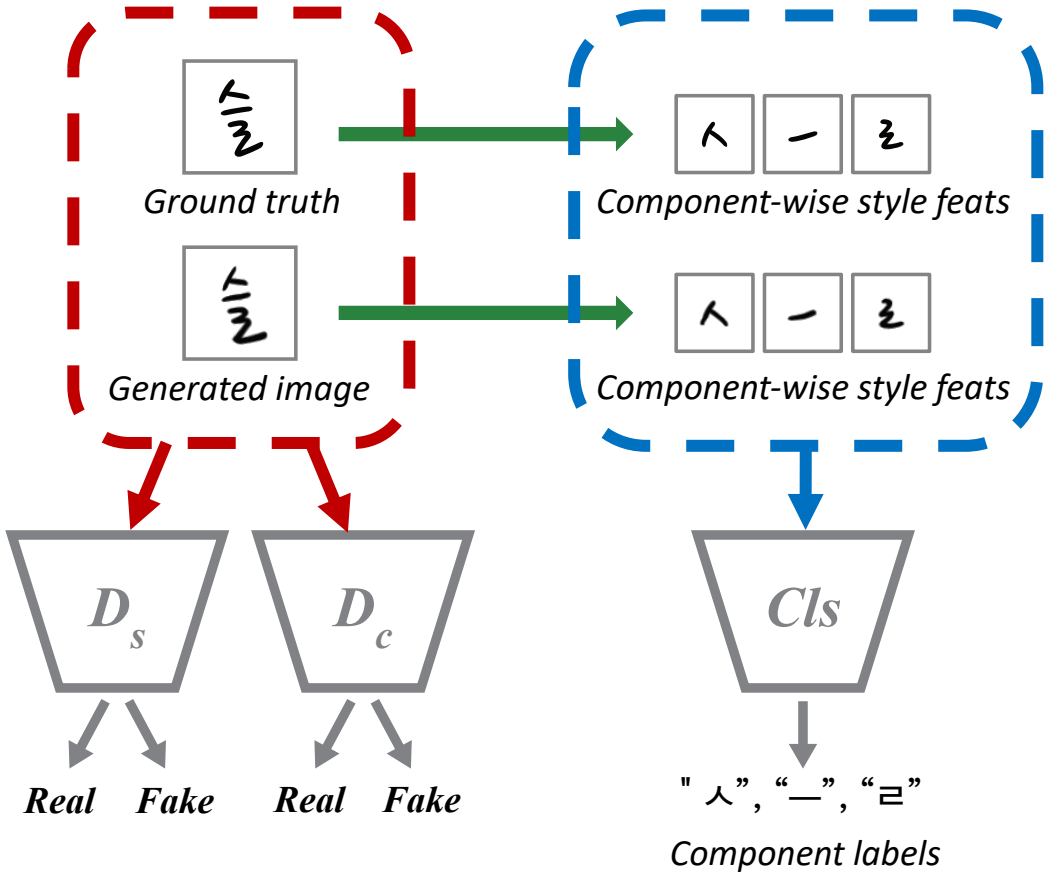

Deriving component-wise style features
Reference Image
Source image
Component-wise style features


Content Encoding
Source image
Content representation

Generation


Goal
LF-Font
Reference Image
Source image
Target image
Deriving component-wise style features
Reference Image
Source image
Component-wise style features
summation
Generation
concatenate
Target image
Content Encoding
Source image
Content representation

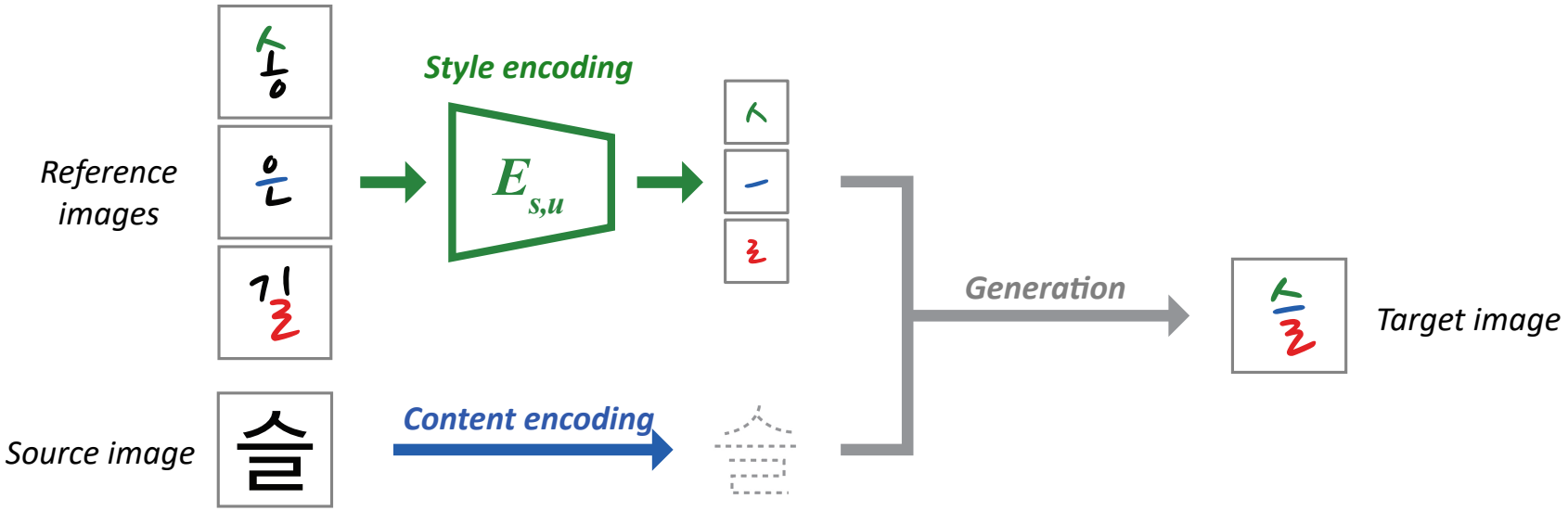
Training



- **Discriminators**
  - Conditional discriminator for style and character
  - ***Adversarial loss***
  - ***Feature matching loss, L1 loss***
- **Component classifier**
  - Classifies the component-wise style features into components.
  - ***Component classification loss***
- **Factorization modules**
  - ***Consistency loss***

Training factorization modules

- **2-Phase learning**
  - Only the “original” component-wise style features are used for generation in **phase 1.**



  - Both the “original” and “reconstructed” component-wise style features are used for generation in **phase 2.**

Comparison methods

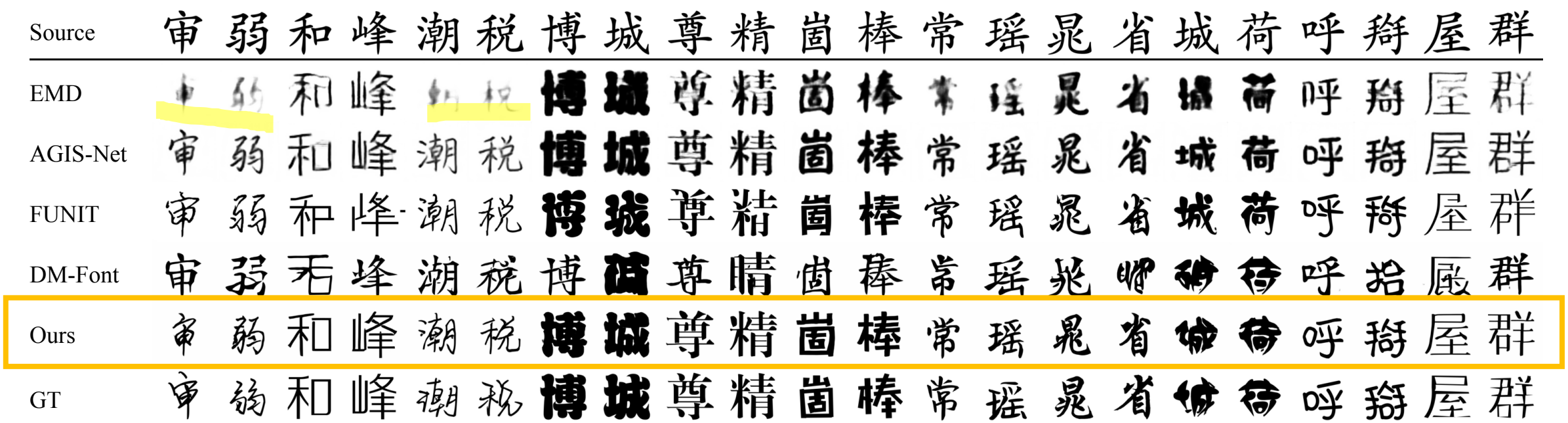
	Localized style?	Contents encoder?	Restricted to generate
EMD	✘	✔	
AGIS-Net	✘	✔	
FUNIT	✘	✔	
DM-Font	✔	✘	unseen components (refs.)
Ours	✔	✔	

- EMD, AGIS-Net, FUNIT extract a **universal style** and employ a content encoder.
- DM-Font utilizes a **localized style**, but cannot handle unseen components.
- LF-Font (Ours) utilizes a **localized style** and also can handle **unseen components.**

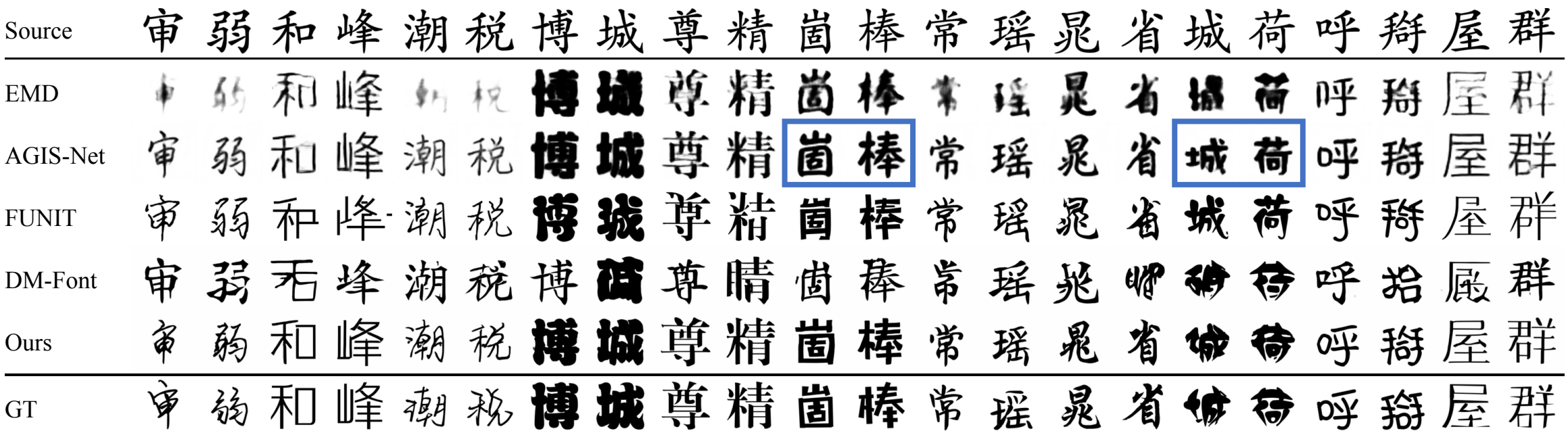
Evaluation protocol

- **Training:** 467 Chinese fonts (each font contains 6,654 characters on average)
- **Evaluation:** unseen 15 Chinese fonts with 2,615 seen characters and 280 unseen characters.
- **8 reference images** are given for the generation.

Generated samples



Generated samples

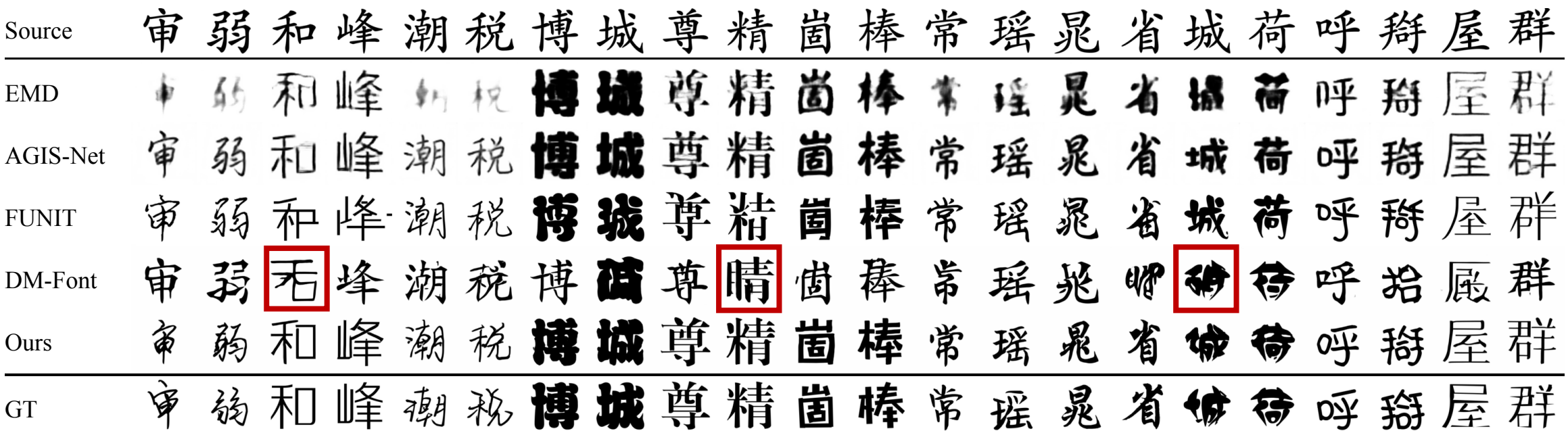


- AGIS-Net fails to preserve local details as serif-ness, varying thickness.

Generated samples

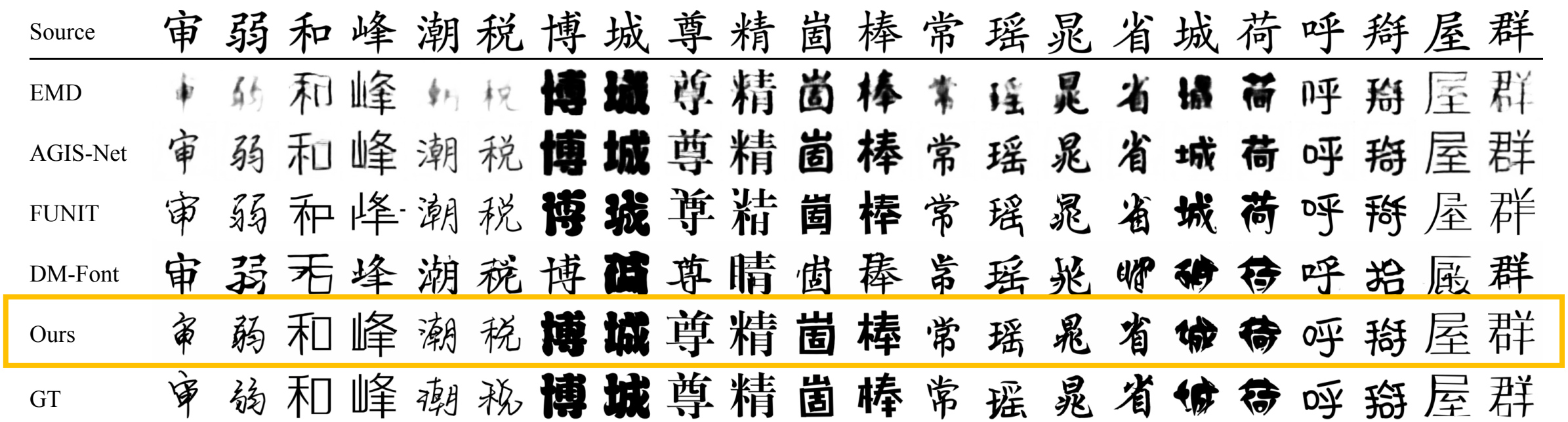
- FUNIT fails to reflect the global style of reference style and generates broken characters.

Generated samples



- DM-Font fails to generate correct characters.

Generated samples



- Ours preserves all the local and global styles and the characters.

Evaluation metrics

- **LPIPS**
  - LPIPS shows the **perceptual similarity** between the generated images and the ground truth images.
- **Acc** (Accuracy)
  - Two **classifiers are trained**; each distinguishes the style or the content of the test dataset.
  - Accuracies of the generated glyphs by style-aware and content-aware models are reported.
- **FID** (Frechet inception distance)
  - FIDs between the **ground truth images** and the generated images are reported.
  - The values are computed by the classifiers which are used to compute the accuracies.

- We repeated the experiments 50 times with different reference characters and reported the average values.

Performance comparison

		LPIPS ↓	Acc (S) ↑	Acc (C) ↑	Acc (Hmean) ↑	FID (S) ↓	FID (C) ↓	FID (Hmean) ↓
Seen chars	EMD (CVPR'18)	0.248	11.9	63.7	20.1	148.1	25.7	43.8
	AGIS-Net (TOG'19)	0.182	34.0	**99.8**	50.7	79.8	4.0	7.7
	FUNIT (ICCV'19)	0.217	39.0	97.1	55.7	58.5	3.6	6.8
	DM-Font (ECCV'20)	0.275	10.2	72.4	17.9	151.8	8.0	15.2
	LF-Font (proposed)	**0.169**	**75.6**	96.6	**84.8**	**40.4**	**2.6**	**4.9**
Unseen chars	EMD (CVPR'18)	0.250	11.6	64.0	19.7	151.7	41.4	65.0
	AGIS-Net (TOG'19)	0.189	33.3	**99.7**	49.9	85.4	10.0	18.0
	FUNIT (ICCV'19)	0.216	38.0	96.8	54.5	63.2	12.3	20.6
	DM-Font (ECCV'20)	0.284	11.1	53.0	18.4	153.4	26.5	45.2
	LF-Font (proposed)	**0.169**	**72.8**	97.1	**83.2**	**44.5**	**8.7**	**14.6**

- Our method outperforms other methods with significant gaps in the metrics that both consider the style and the content.

Performance comparison

		LPIPS ↓	Acc (S) ↑	Acc (C) ↑	Acc (Hmean) ↑	FID (S) ↓	FID (C) ↓	FID (Hmean) ↓
Seen chars	EMD (CVPR'18)	0.248	11.9	63.7	20.1	148.1	25.7	43.8
	AGIS-Net (TOG'19)	0.182	34.0	**99.8**	50.7	79.8	4.0	7.7
	FUNIT (ICCV'19)	0.217	<u>39.0</u>	97.1	55.7	<u>58.5</u>	3.6	6.8
	DM-Font (ECCV'20)	0.275	10.2	72.4	17.9	151.8	8.0	15.2
	LF-Font (proposed)	**0.169**	**75.6**	96.6	**84.8**	**40.4**	**2.6**	**4.9**
Unseen chars	EMD (CVPR'18)	0.250	11.6	64.0	19.7	151.7	41.4	65.0
	AGIS-Net (TOG'19)	0.189	33.3	**99.7**	49.9	85.4	10.0	18.0
	FUNIT (ICCV'19)	0.216	<u>38.0</u>	96.8	54.5	<u>63.2</u>	12.3	20.6
	DM-Font (ECCV'20)	0.284	11.1	53.0	18.4	153.4	26.5	45.2
	LF-Font (proposed)	**0.169**	**72.8**	97.1	**83.2**	**44.5**	**8.7**	**14.6**

- Our method particularly shows robust performance in **style-aware benchmarks.**

Performance comparison

		LPIPS ↓	Acc (S) ↑	Acc (C) ↑	Acc (Hmean) ↑	FID (S) ↓	FID (C) ↓	FID (Hmean) ↓
Seen chars	EMD (CVPR'18)	0.248	11.9	63.7	20.1	148.1	25.7	43.8
	AGIS-Net (TOG'19)	0.182	34.0	**99.8**	50.7	79.8	4.0	7.7
	FUNIT (ICCV'19)	0.217	39.0	97.1	55.7	58.5	3.6	6.8
	DM-Font (ECCV'20)	0.275	10.2	72.4	17.9	151.8	8.0	15.2
	LF-Font (proposed)	**0.169**	**75.6**	96.6	**84.8**	**40.4**	**2.6**	**4.9**
Unseen chars	EMD (CVPR'18)	0.250	11.6	64.0	19.7	151.7	41.4	65.0
	AGIS-Net (TOG'19)	0.189	33.3	**99.7**	49.9	85.4	10.0	18.0
	FUNIT (ICCV'19)	0.216	38.0	96.8	54.5	63.2	12.3	20.6
	DM-Font (ECCV'20)	0.284	11.1	53.0	18.4	153.4	26.5	45.2
	LF-Font (proposed)	**0.169**	**72.8**	97.1	**83.2**	**44.5**	**8.7**	**14.6**

- AGIS-Net shows the best performance in the content-aware accuracy.
  - Because AGIS-Net more concentrates to the characters than the styles.

Extending to other languages

- Our model also easily can be extended to other compositional scripts.

Conclusion

- Enabling few-shot font generation **with only a few references**
- By utilizing **component-wise style representations** and introducing **factorization modules**

Thank you for listening!