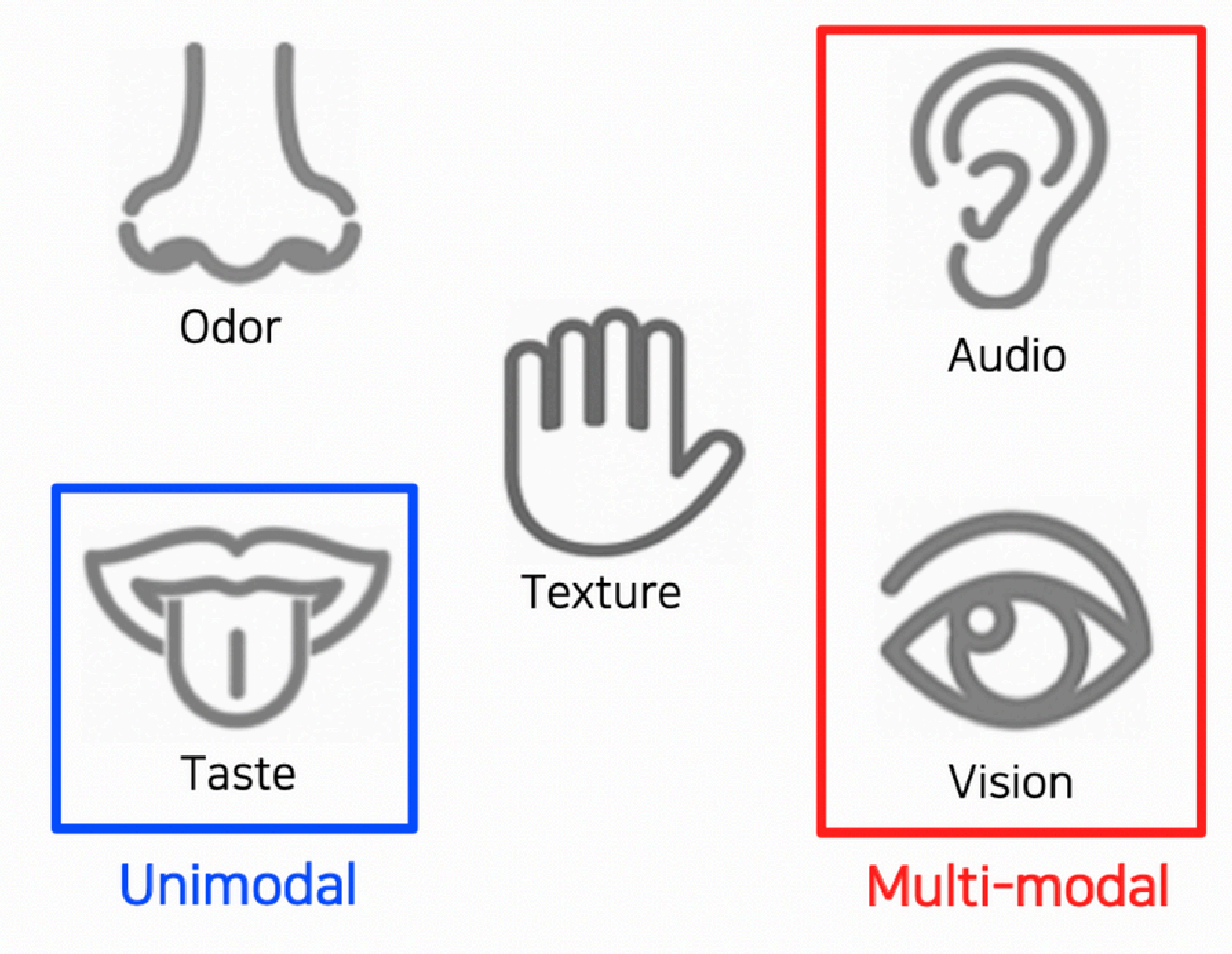
A Study on the Porting of BLIP in Embedded Systems (Raspberry PI 5) Environment

(Written by 임채우)

Background

- **What is Multi-Modal?**
- **Vision Language Pre-training (VLP) Model**
- **VLP Datasets**

Background: Multi-Modal



- When people obtain information from their surroundings, they obtain it in multiple modality.
- Using multiple types of data together is called multi-modal.
- **Unimodal**: It means a model with one modality.
- **Multi-modal**: It means a model with multiple modality.

Background: Vision Language Pre-training Model

Vision-Language Pre-training Model is an artificial intelligence model that has been trained to process and understand text and images simultaneously.

Encoder-Decoder base model

- VL-T5
- SimVLM

Encoder base model

- CLIP
- ALBEF

Background: VLP Datasets

- **Human-annotated Dataset**: COCO, Visual Genome

->This dataset has less noise because it is manually labeled by humans. However, the number of data sets is small.

- **Web Dataset**: Conceptual Captions, SBU Captions, RedCaps

->This dataset obtained by passing web text crawled from the web through a filtering pipeline. So there is a lot of data, but there is also a lot of noise.

- **Nosiy Web Dataset**: LAION

->This dataset obtained by entering a specific keyword and collecting all images that match it when crawling the web. Therefore, there is a lot of data, but there is a lot of noise than Web dataset

Introduction

Limitations of existing VLP models

- **Model Perspectives**

1. Previously, most VLP models used the Encoder base model and Encoder-Decoder base model.

2. **Encoder base model**(CLIP, ALBEF) oes not perform well in generation-based tasks (Image Captioning)

3. **Encoder-Decoder base model**(VL-T5, SimVLM) does not perform well in Understanding-based Task (Image-Text Retrieval)

Introduction

Limitations of existing VLP models

- **Data Perspectives**

1. Most of the latest techniques are characterized by pre-training using images crawled from the web and Alt-text pairs.

2. Even with simple filters there is still a lot of noise.

3. Therefore, despite the performance improvement achieved by expanding the data set, it was confirmed that Web Text with a lot of noise is not optimal for Vision-Lanuage learning.

Introduction

Solution from two perspectives

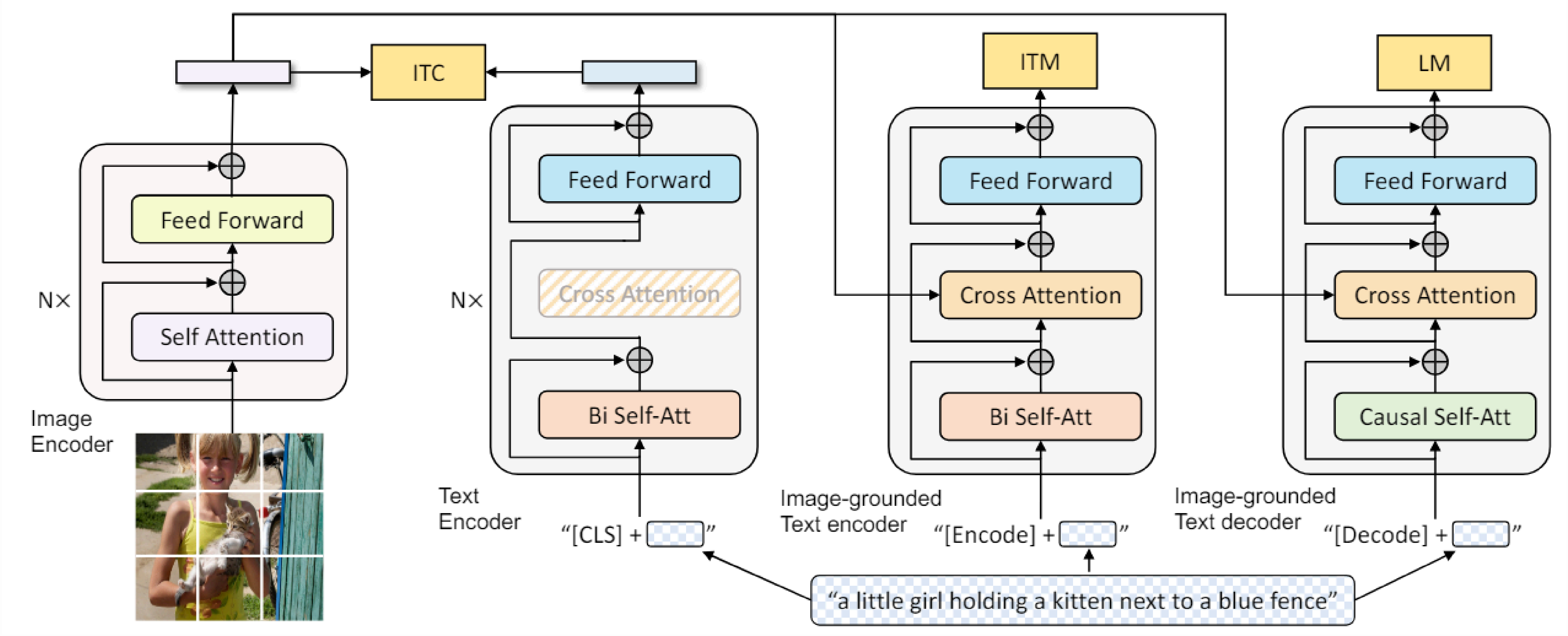
- **Model Perspectives: Multimodal Mixture of Encoder-Decoder(MED)**

1. It is a new model structure for effective multi-task pre-training and transfer learning.
2. MED operates as a Unimodal Encoder and Image-grounded Text Encoder/Decoder.
3. And pre-training is conducted jointly with three objectives.

- **Data Perspectives: Captioning and Filtering(CapFilt)**

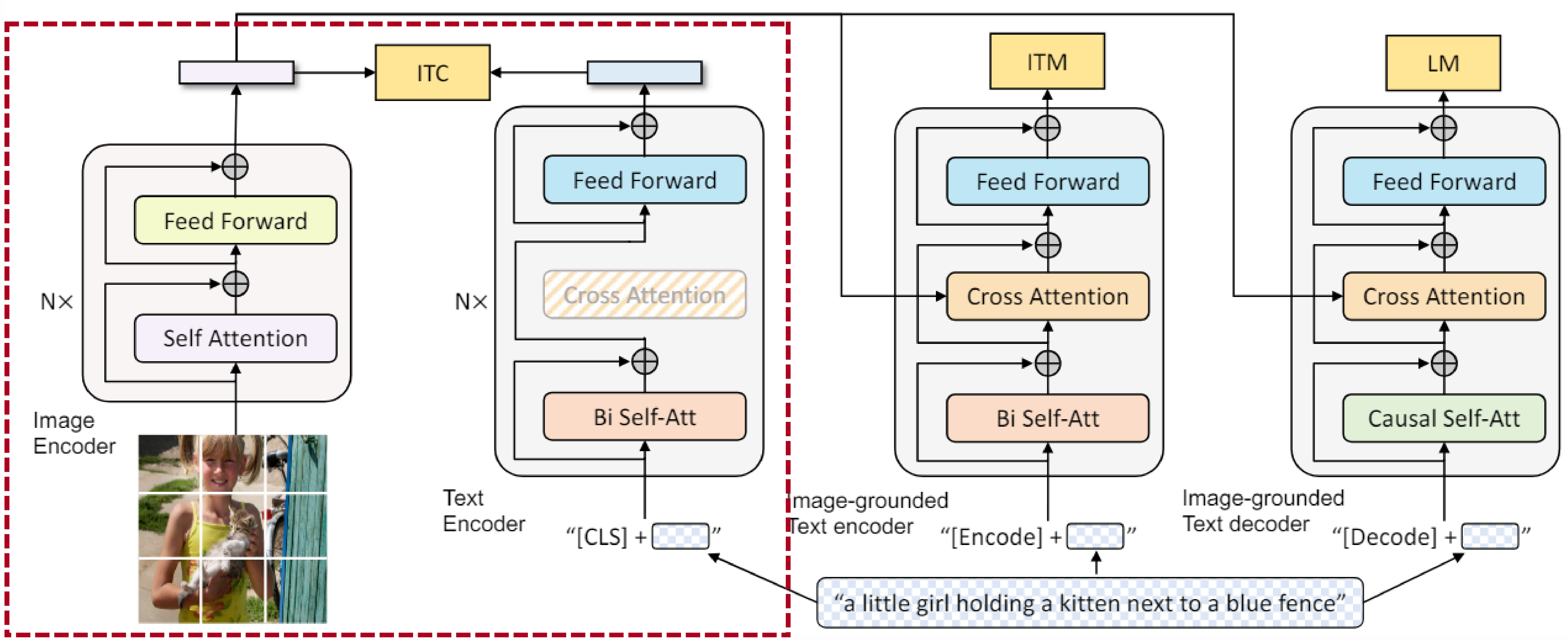
1. This is a new data boostrapping method for learning with an Image-Text Pair with noise.
2. Captioner that creates a synthetic text given a web image
3. Filter to remove captions with noise from each of the original web text and synthetic text

Model Architecture: MED



- The BLIP model consists of a Unimodal encoder(Image, Text), Image-grounded Text Encoder, and Image-grounded Text Decoder
- in this paper, we propose MED to pre-train a Unified Model with both understanding-based Tasks and generation-based Tasks functions.

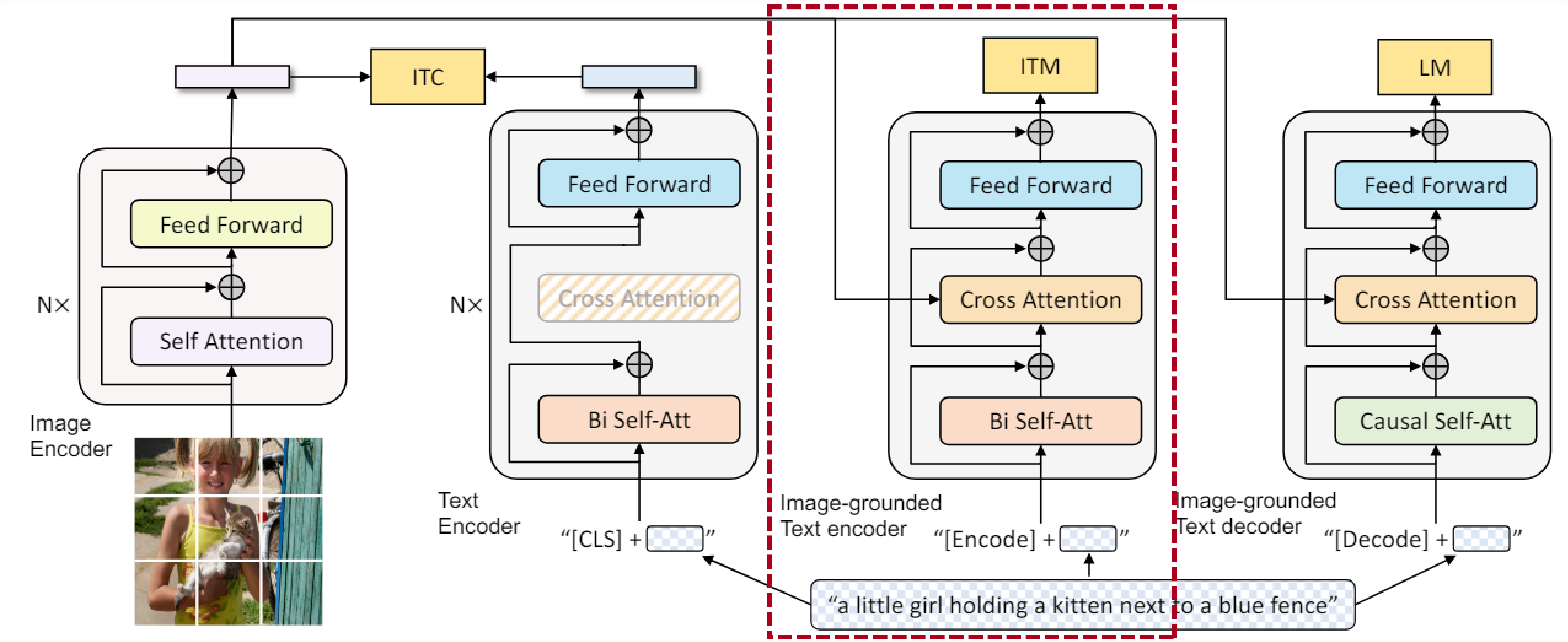
Model Architecture: Unimodal encoder



- Encode Image and Text separately using Unimodal Encoder.
- ViT is used in the Image Encoder to divide the input image into patches and embed them, and the [CLS] token is added to express the Global Image Feature.
- The text encoder used BERT, and the [CLS] token was added to the beginning of text input to summarize the sentence.

(Written by 임채우)

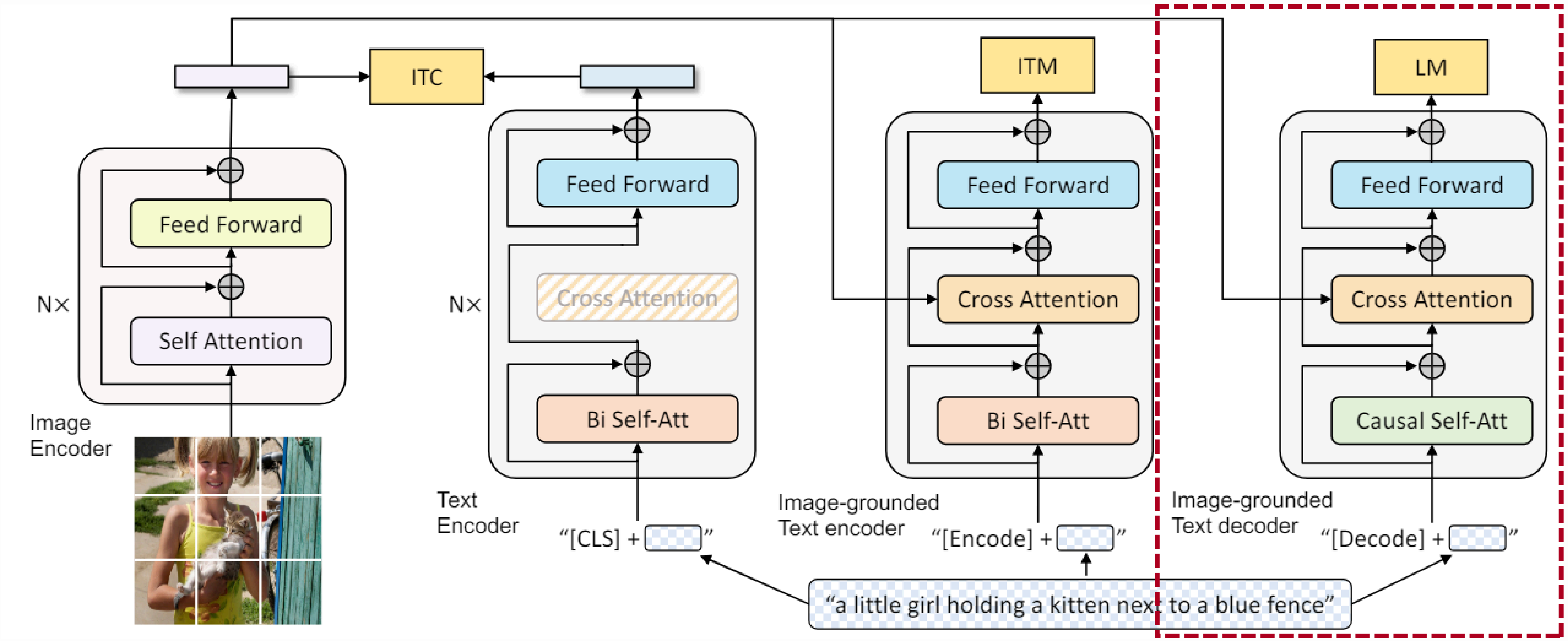
Model Architecture: Image-grounded Text Encoder



- For each Transformer Block, a cross-attention layer was inserted between the self-attention layer and the feed-forward network.
- As input, an [Encode] token is added to the text, and this [Encode] Embedding is used as a multimodal representation of the Image-Text Pair.

(Written by 임채우)

Model Architecture: Image-grounded Text Decoder



- The Bidirectional Self-attention Layer of the Image-grounded Text Encoder was replaced with a Casual Self-attention Layer.
- The [Decoder] token is used to signal the start of the sequence, and the [EOS] token is used to signal the end of the sequence.

Pre-training Objectives

- **The BLIP model jointly optimizes three Objectives.**

1. Understanding-based Objective

->Image-Text Contrastive Loss(ITC) and Image-Text Matching Loss(ITM)

2. Generation-based Objective

->Language Modeling Loss(LM)

$$L = L_{itc} + L_{itm} + L_{lm}$$

(Written by 임채우)

Pre-training Objectives: ITC Loss

- ITC activates Unimodal Encoder.

$$L = \boxed{L_{itc}} + L_{itm} + L_{lm}$$

- By inducing Positive Image-Text Pair to have similar Representation, Visual Transformer and Text Transformer Aims to Align Feature Space

- This is effective in improving vision and language understanding.

- Momentum Encoder was used when calculating ITC Loss.

Pre-training Objectives: ITC Loss

$$p_m^{\mathrm{i2t}}(I) = \frac{\exp(s(I, T_m)/\tau)}{\sum_{m=1}^{M} \exp(s(I, T_m)/\tau)}, \quad p_m^{\mathrm{t2i}}(T) = \frac{\exp(s(T, I_m)/\tau)}{\sum_{m=1}^{M} \exp(s(T, I_m)/\tau)}$$

->For each Image and Text, we calculate the probability distribution using the Softmax-Normalized function based on the similarity.

$$\mathcal{L}_{\mathrm{itc}} = \frac{1}{2}\mathbb{E}_{(I,T)\sim D}\left[\mathrm{H}(\boldsymbol{y}^{\mathrm{i2t}}(I), \boldsymbol{p}^{\mathrm{i2t}}(I)) + \mathrm{H}(\boldsymbol{y}^{\mathrm{t2i}}(T), \boldsymbol{p}^{\mathrm{t2i}}(T))\right]$$

->The loss is calculated using cross entropy using the probability distribution obtained above and y, the ground-truth representing the correct answer.

$$L = \boxed{L_{itc}} + L_{itm} + L_{lm}$$

(Written by 임채우)

Pre-training Objectives: ITC Loss

```
image_embeds_m = self.visual_encoder_m(image)
image_feat_m = F.normalize(self.vision_proj_m(image_embeds_m[:,0,:]),dim=-1)
image_feat_all = torch.cat([image_feat_m.t(),self.image_queue.clone().detach()],dim=1)

text_output_m = self.text_encoder_m(text.input_ids, attention_mask = text.attention_mask,
                                    return_dict = True, mode = 'text')
text_feat_m = F.normalize(self.text_proj_m(text_output_m.last_hidden_state[:,0,:]),dim=-1)
text_feat_all = torch.cat([text_feat_m.t(),self.text_queue.clone().detach()],dim=1)

sim_i2t_m = image_feat_m @ text_feat_all / self.temp
sim_t2i_m = text_feat_m @ image_feat_all / self.temp
```

- Converts each image text input into an embedding and normalizes it.
- Combines the features of the current image batch and the features of the previous image batch.
- Then, measure the similarity between i2t and t2i. Here, temp is a hyper parameter that scales.

Pre-training Objectives: ITC Loss

```
    sim_targets = torch.zeros(sim_i2t_m.size()).to(image.device)
    sim_targets.fill_diagonal_(1)

    sim_i2t_targets = alpha * F.softmax(sim_i2t_m, dim=1) + (1 - alpha) * sim_targets
    sim_t2i_targets = alpha * F.softmax(sim_t2i_m, dim=1) + (1 - alpha) * sim_targets

sim_i2t = image_feat @ text_feat_all / self.temp
sim_t2i = text_feat @ image_feat_all / self.temp

loss_i2t = -torch.sum(F.log_softmax(sim_i2t, dim=1)*sim_i2t_targets,dim=1).mean()
loss_t2i = -torch.sum(F.log_softmax(sim_t2i, dim=1)*sim_t2i_targets,dim=1).mean()

loss_ita = (loss_i2t+loss_t2i)/2
```

- Create sim_targets to generate ground-truth.
- Next, similarity is measured using the main encoder.
- Then, the loss is calculated using sim_i2t_targets, which is the correct answer, and sim_i2t obtained from the main encoder. Same with t2i.

Pre-training Objectives: ITM Loss

- ITM activates the Image-grounded Text encoder.
- Learn Image-Text Multimodal Representation that captures fine-grained alignment between Vision-Language.
- As a binary classification task, when the model uses ITM Head (Linear Layer) and the Image-Text pair considers multimodal features, Predict whether it is Positive (Matched) or Negative (Unmatched)
- We use the Hard Negative strategy to find negative examples that provide more information.

$$L = L_{itc} + \boxed{L_{itm}} + L_{lm}$$

(Written by 임채우)

Pre-training Objectives: ITM Loss

$$\mathcal{L}_{\mathrm{itm}} = \mathbb{E}_{(I,T)\sim D}\mathrm{H}(\boldsymbol{y}^{\mathrm{itm}}, \boldsymbol{p}^{\mathrm{itm}}(I,T))$$

->ITM is obtained as cross entropy between y, the ground-truth indicating the correct answer, and P indicating whether or not it matches.

$$L = L_{itc} + \boxed{L_{itm}} + L_{lm}$$

(Written by 임채우)

Pre-training Objectives: ITM Loss

```
vl_embeddings = torch.cat([output_pos.last_hidden_state[:,0,:], output_neg.last_hidden_state[:,0,:]],dim=0)
vl_output = self.itm_head(vl_embeddings)
                                          self.itm_head = nn.Linear(text_width, 2)
itm_labels = torch.cat([torch.ones(bs,dtype=torch.long),torch.zeros(2*bs,dtype=torch.long)],
                       dim=0).to(image.device)
loss_itm = F.cross_entropy(vl_output, itm_labels)
```

- v1_embeddings is an embedding to determine whether there is a match between the image and the text. This embedding combines the embeddings for the positive and negative pairs together.

- Enter v1_embeddings as input into ITM_head to check whether the image and text match.

- Then, the loss is calculated using itm_lables and Cross Entropy, which represent the correct answer.

(Written by 임채우)

Pre-training Objectives: LM Loss

- LM activates the Image-grounded Text Decoder.

$$L = L_{itc} + L_{itm} + \boxed{L_{lm}}$$

- It aims to generate a text description for a given image.
- Unlike MLM, which is widely used in VLP, LM has excellent generalization ability and is good for converting visual information into consistent captions.
- LM is optimal with a cross-entropy loss that trains the model to maximize the likelihood of the text in an auto regressive manner.

Pre-training Objectives: LM Loss

```
decoder_targets = decoder_input_ids.masked_fill(decoder_input_ids == self.tokenizer.pad_token_id, -100)

decoder_output = self.text_decoder(decoder_input_ids,
                                   attention_mask = text.attention_mask,
                                   encoder_hidden_states = image_embeds,
                                   encoder_attention_mask = image_atts,
                                   labels = decoder_targets,
                                   return_dict = True,
                                  )

loss_lm = decoder_output.loss
```

- Create decoder_targets. At this time, set pad to -100 so as not to predict the pad part.

- The loss is calculated by calculating the difference between the result predicted by the model and the actual next word.

Parameter Sharing

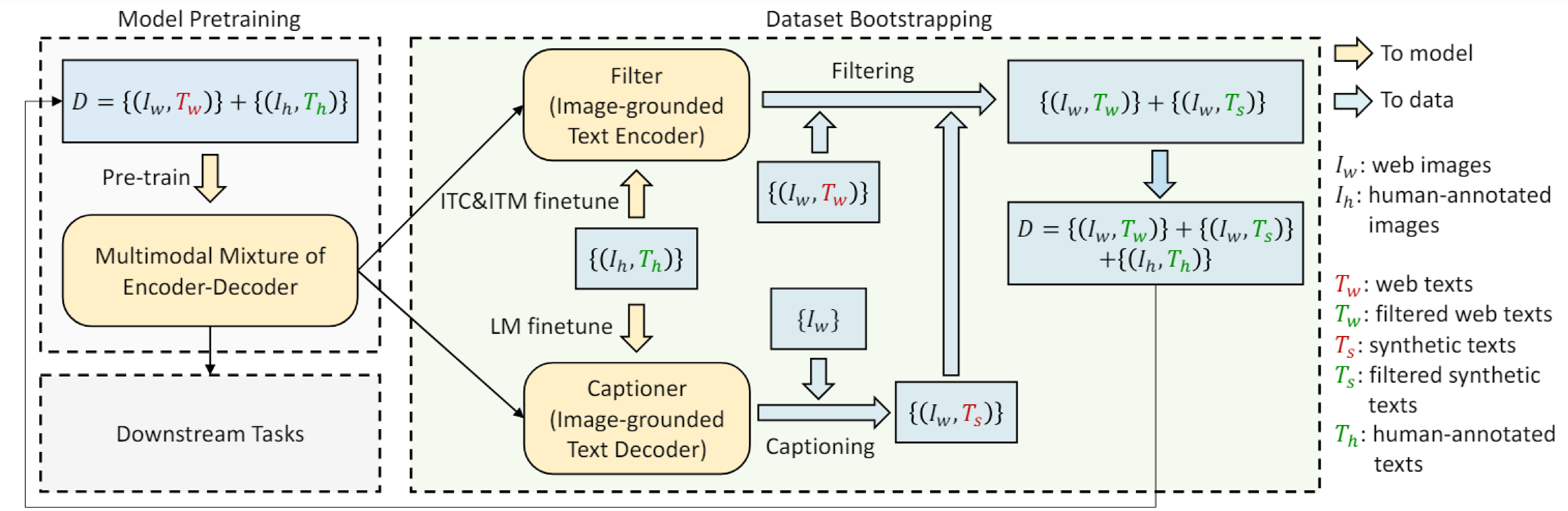
- In order to perform efficient pre-training, the text encoder and decoder share all parameters except self-attention.

- The reason why self-attention parameters are not shared is because the information contained in the image and text is different.

- By sharing parameters, we were able to increase the efficiency of pre-training and reduce the size of the model.

(Written by 임채우)

CapFilt

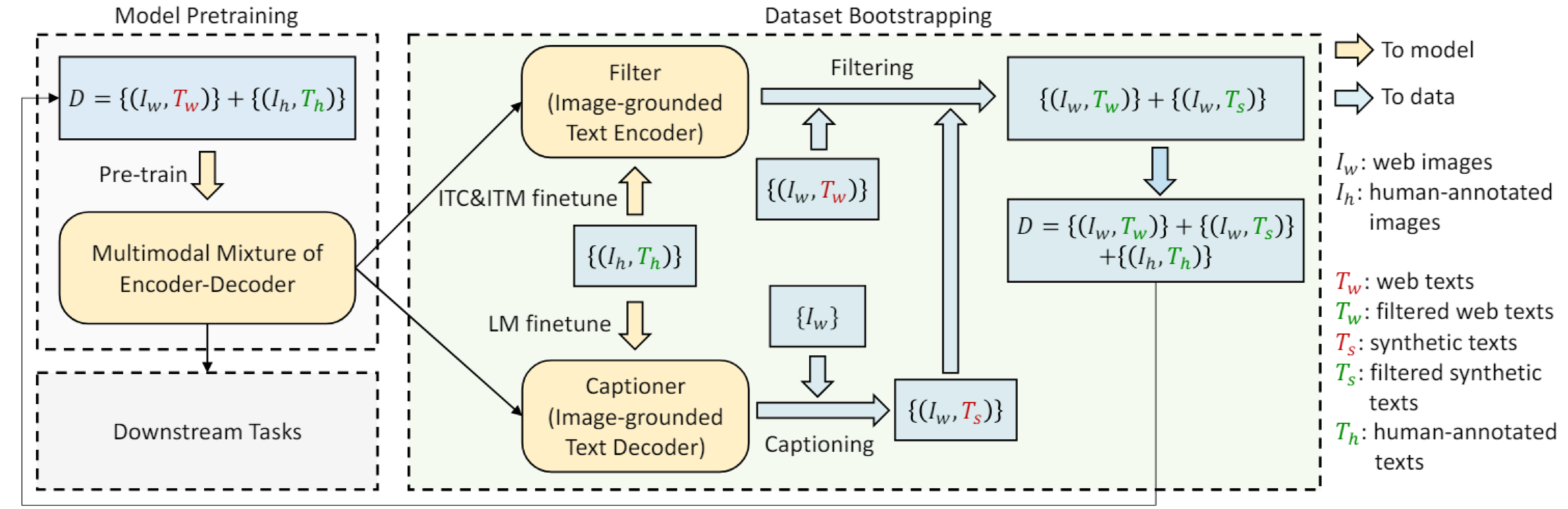


- This is a new data boostrapping method for learning with an Image-Text Pair with noise.

1. Captioner that creates a synthetic text given a web image
2. Filter to remove captions with noise from each of the original web text and synthetic text

(Written by 임채우)

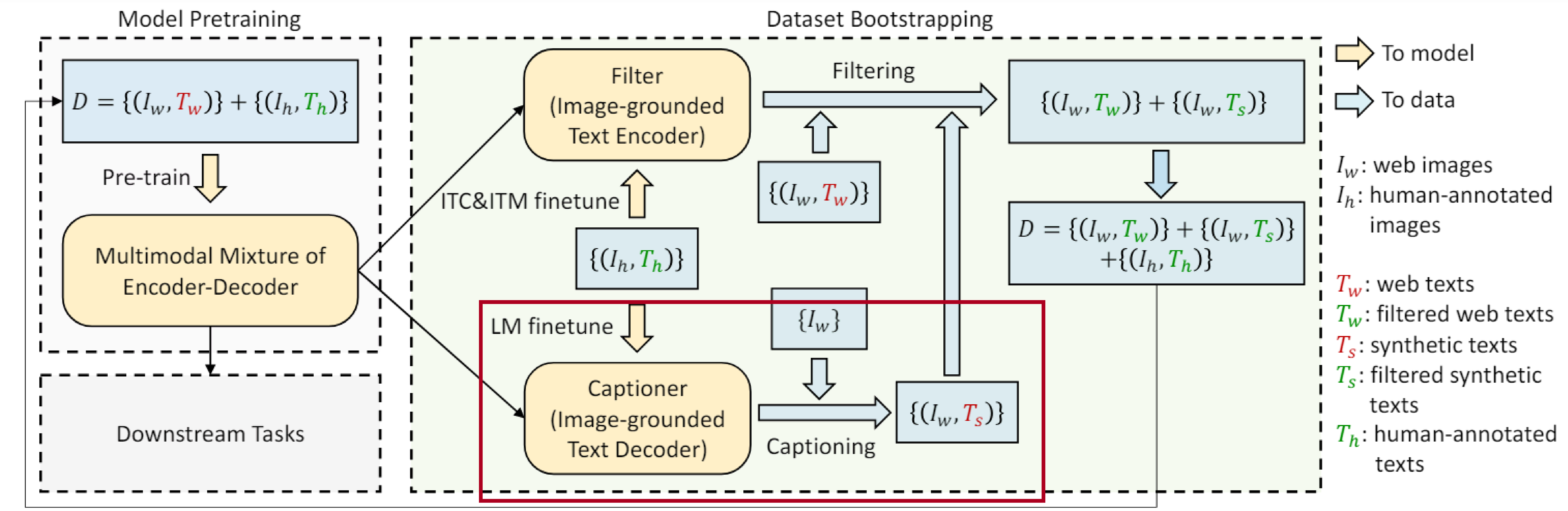
CapFilt Framework



- I means Image, and T means Text
- W means Web
- H means Human-annotated
- S means Synthetic

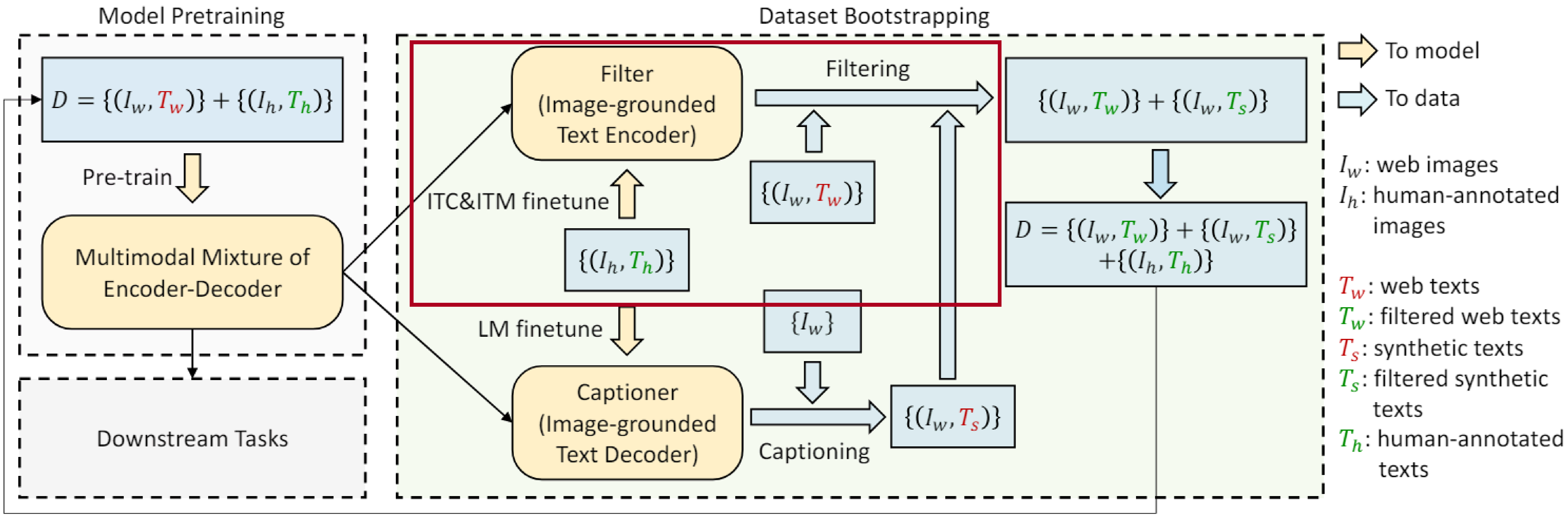
(Written by 임채우)

CapFilt: Captioning



- Captioner means Image-grounded Text Decoder.
- To decode the text of a given image, it is fine-tuned according to the LM Objective.
- Given a web image {Iw}, Captioner creates Synthetic Caption {Ts} with one caption per image.

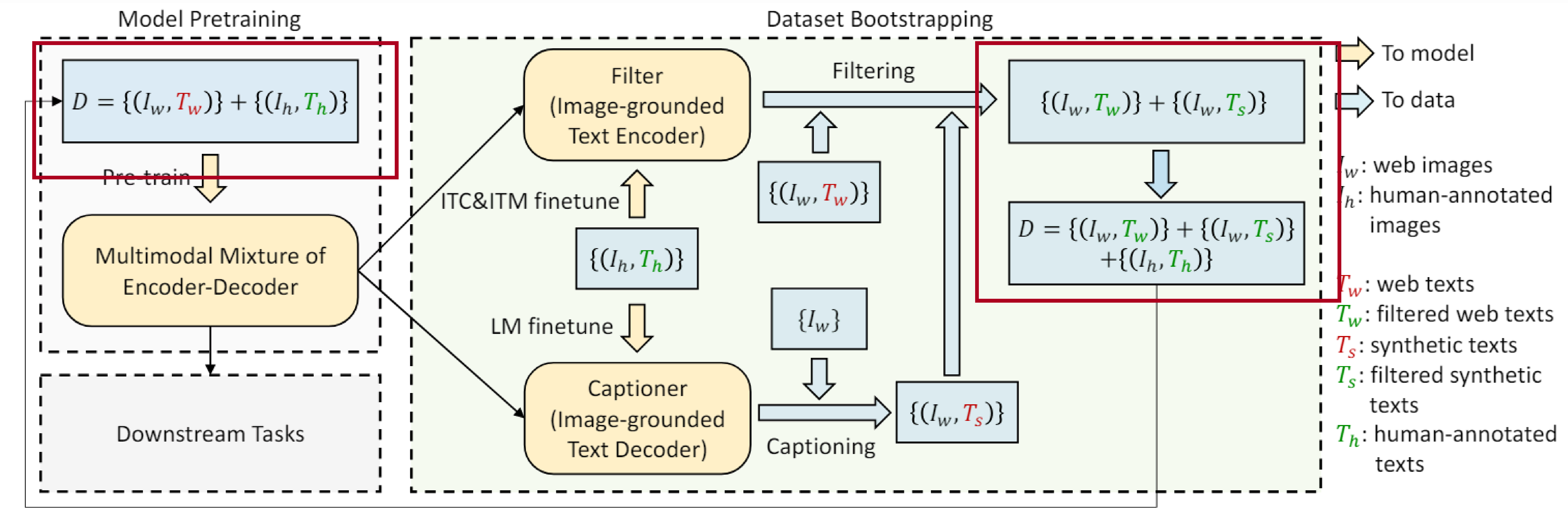
CapFilt: Filtering



- Filter Means Image-grounded Text Encoder.
- In order to learn whether the text matches the image, detailed fine-tuning is performed according to ITC, ITM, and Objective.
- Filter removes text with noise from both the original web text {Tw} and the synthetic text {Ts}.

(Written by 임채우)

CapFilt: Combination



- Filtered Image-Text Pairs are combined with human-annotated pairs to form a new data set and used to pre-train a new model.

Experiments

- **Effect of CapFilt**

Pre-train dataset	Bootstrap C	F	Vision backbone	Retrieval-FT (COCO) TR@1	IR@1	Retrieval-ZS (Flickr) TR@1	IR@1	Caption-FT (COCO) B@4	CIDEr	Caption-ZS (NoCaps) CIDEr	SPICE
COCO+VG +CC+SBU (14M imgs)	✗	✗	ViT-B/16	78.4	60.7	93.9	82.1	38.0	127.8	102.2	13.9
	✗	✓$_B$		79.1	61.5	94.1	82.8	38.1	128.2	102.7	14.0
	✓$_B$	✗		79.7	62.0	94.4	83.6	38.4	128.9	103.4	14.2
	✓$_B$	✓$_B$		80.6	63.1	94.8	84.9	38.6	129.7	105.1	14.4
COCO+VG +CC+SBU +LAION (129M imgs)	✗	✗	ViT-B/16	79.6	62.0	94.3	83.6	38.8	130.1	105.4	14.2
	✓$_B$	✓$_B$		81.9	64.3	96.0	85.0	39.4	131.4	106.3	14.3
	✓$_L$	✓$_L$		81.2	64.1	96.0	85.5	39.7	133.3	109.6	14.7
	✗	✗	ViT-L/16	80.6	64.1	95.1	85.5	40.3	135.5	112.5	14.7
	✓$_L$	✓$_L$		82.4	65.1	96.7	86.7	40.4	136.7	113.2	14.8

- The efficiency of CapFilt was demonstrated for downstream tasks such as Image-Text Retrieval and Image Captioning.

Experiments

- **Diversity is Key for Synthetic Captions**

Generation method	Noise ratio	Retrieval-FT (COCO) TR@1	Retrieval-FT (COCO) IR@1	Retrieval-ZS (Flickr) TR@1	Retrieval-ZS (Flickr) IR@1	Caption-FT (COCO) B@4	Caption-FT (COCO) CIDEr	Caption-ZS (NoCaps) CIDEr	Caption-ZS (NoCaps) SPICE
None	N.A.	78.4	60.7	93.9	82.1	38.0	127.8	102.2	13.9
Beam	19%	79.6	61.9	94.1	83.1	38.4	128.9	103.5	14.2
Nucleus	25%	80.6	63.1	94.8	84.9	38.6	129.7	105.1	14.4

- CapFilt used Nucleous Sampling to create synthetic captions.

->Nucleous Sampling is a technique that generates text by considering only words that are in the top p% of the probability distribution of words

- In this table, it is compared with Beam Search, a deterministic decoding method that aims to generate captions with the highest probability.

Experiments

- **Image-Text Retrieval**

Method	Pre-train # Images	COCO (5K test set) TR R@1	R@5	R@10	IR R@1	R@5	R@10	Flickr30K (1K test set) TR R@1	R@5	R@10	IR R@1	R@5	R@10
UNITER (Chen et al., 2020)	4M	65.7	88.6	93.8	52.9	79.9	88.0	87.3	98.0	99.2	75.6	94.1	96.8
VILLA (Gan et al., 2020)	4M	-	-	-	-	-	-	87.9	97.5	98.8	76.3	94.2	96.8
OSCAR (Li et al., 2020)	4M	70.0	91.1	95.5	54.0	80.8	88.5	-	-	-	-	-	-
UNIMO (Li et al., 2021b)	5.7M	-	-	-	-	-	-	89.4	98.9	99.8	78.0	94.2	97.1
ALIGN (Jia et al., 2021)	1.8B	77.0	93.5	96.9	59.9	83.3	89.8	95.3	99.8	100.0	84.9	97.4	98.6
ALBEF (Li et al., 2021a)	14M	77.6	94.3	97.2	60.7	84.3	90.5	95.9	99.8	100.0	85.6	97.5	98.9
BLIP	14M	80.6	95.2	97.6	63.1	85.3	91.1	96.6	99.8	**100.0**	87.2	97.5	98.8
BLIP	129M	**81.9**	95.4	97.8	**64.3**	85.7	91.5	**97.3**	**99.9**	**100.0**	87.3	97.6	**98.9**
$\text{BLIP}_{\text{CapFilt-L}}$	129M	81.2	**95.7**	**97.9**	64.1	**85.8**	**91.6**	97.2	**99.9**	**100.0**	**87.5**	**97.7**	**98.9**
$\text{BLIP}_{\text{ViT-L}}$	129M	82.4	95.4	97.9	65.1	86.3	91.8	97.4	99.8	99.9	87.6	97.7	99.0

- BLIP evaluation for both Image-to-Text Retrieval (TR) and Text-to-image Retireval (IR) on COCO and Flicker Datasets

(Written by 임채우)

Experiments

- **Image-Text Retrieval**

Method	Pre-train # Images	Flickr30K (1K test set) TR			IR		
		R@1	R@5	R@10	R@1	R@5	R@10
CLIP	400M	88.0	98.7	99.4	68.7	90.6	95.2
ALIGN	1.8B	88.6	98.7	99.7	75.7	93.8	96.8
ALBEF	14M	94.1	99.5	99.7	82.8	96.3	98.1
BLIP	14M	94.8	99.7	**100.0**	84.9	96.7	98.3
BLIP	129M	**96.0**	**99.9**	**100.0**	85.0	**96.8**	98.6
$\text{BLIP}_{\text{CapFilt-L}}$	129M	**96.0**	**99.9**	**100.0**	**85.5**	**96.8**	**98.7**
$\text{BLIP}_{\text{ViT-L}}$	129M	96.7	100.0	100.0	86.7	97.3	98.7

- By transferring the fine-tuned model from COCO to Flick30k and performing Zero-Shot Retrieval, SOTA was achieved with a high margin.

Experiments

- **Image Captioning**

Method	Pre-train #Images	NoCaps validation in-domain C	S	near-domain C	S	out-domain C	S	overall C	S	COCO Caption Karpathy test B@4	C
Enc-Dec (Changpinyo et al., 2021)	15M	92.6	12.5	88.3	12.1	94.5	11.9	90.2	12.1	-	110.9
VinVL† (Zhang et al., 2021)	5.7M	103.1	14.2	96.1	13.8	88.3	12.1	95.5	13.5	38.2	129.3
$\text{LEMON}_{\text{base}}$† (Hu et al., 2021)	12M	104.5	14.6	100.7	14.0	96.7	12.4	100.4	13.8	-	-
$\text{LEMON}_{\text{base}}$† (Hu et al., 2021)	200M	107.7	14.7	106.2	14.3	107.9	13.1	106.8	14.1	**40.3**	**133.3**
BLIP	14M	111.3	15.1	104.5	14.4	102.4	13.7	105.1	14.4	38.6	129.7
BLIP	129M	109.1	14.8	105.8	14.4	105.7	13.7	106.3	14.3	39.4	131.4
$\text{BLIP}_{\text{CapFilt-L}}$	129M	**111.8**	**14.9**	**108.6**	**14.8**	**111.5**	**14.2**	**109.6**	**14.7**	39.7	**133.3**
$\text{LEMON}_{\text{large}}$† (Hu et al., 2021)	200M	116.9	15.8	113.3	15.1	111.3	14.0	113.4	15.0	40.6	135.7
$\text{SimVLM}_{\text{huge}}$ (Wang et al., 2021)	1.8B	113.7	-	110.9	-	115.2	-	112.2	-	40.6	143.3
$\text{BLIP}_{\text{ViT-L}}$	129M	114.9	15.2	112.1	14.9	115.3	14.4	113.2	14.8	40.4	136.7

- Evaluating the performance of Image Captioning with NoCaps and COCO datasets

(Written by 임채우)

Experiments

- **Visual Question Answering(VQA)**

Method	Pre-train #Images	VQA test-dev	VQA test-std	NLVR² dev	NLVR² test-P
LXMERT	180K	72.42	72.54	74.90	74.50
UNITER	4M	72.70	72.91	77.18	77.85
VL-T5/BART	180K	-	71.3	-	73.6
OSCAR	4M	73.16	73.44	78.07	78.36
SOHO	219K	73.25	73.47	76.37	77.32
VILLA	4M	73.59	73.67	78.39	79.30
UNIMO	5.6M	75.06	75.27	-	-
ALBEF	14M	75.84	76.04	82.55	83.14
$\text{SimVLM}_{\text{base}}$†	1.8B	77.87	78.14	81.72	81.77
BLIP	14M	77.54	77.62	**82.67**	82.30
BLIP	129M	78.24	78.17	82.48	**83.08**
$\text{BLIP}_{\text{CapFilt-L}}$	129M	**78.25**	**78.32**	82.15	82.24

- BLIP using 129M images achieves better performance than SimVLM with 13x more data and an additional Conv Stage.

Raspberry PI 5 Environment

- Inference is performed using the BLIP model in the Raspberry Pi 5 environment
- By checking the results, it is immediately clear whether the BLIP model is suitable for On-Device AI.

- The implementation environment of Raspberry Pi 5 to be used in the experiment is shown in Table 1.

항목		내용
H/W	CPU	BCM2712 (2.4GHz)
	GPU	VideoCore VII (800MHz)
	MEMORY	SDRAM 4267
	SD card	micro card slot, SDR104 Supports high-speed mode
S/W	O/S	Debian GNU/Linux 12
	Library	Pytorch=2.2.2, Transformers=4.41.1

(Table 1 Implementation environment of Raspberry Pi 5)

(Written by 임채우)

Experiments

- The applied downstream task is Image-Captioing. Compare the inference time for one image with and without Conditional Caption.
- The picture used for image-captioning is shown in Figure 1.
- The shell operation screen is shown in Figure 2.

Figure 1. Pictures used in Image-Captioning

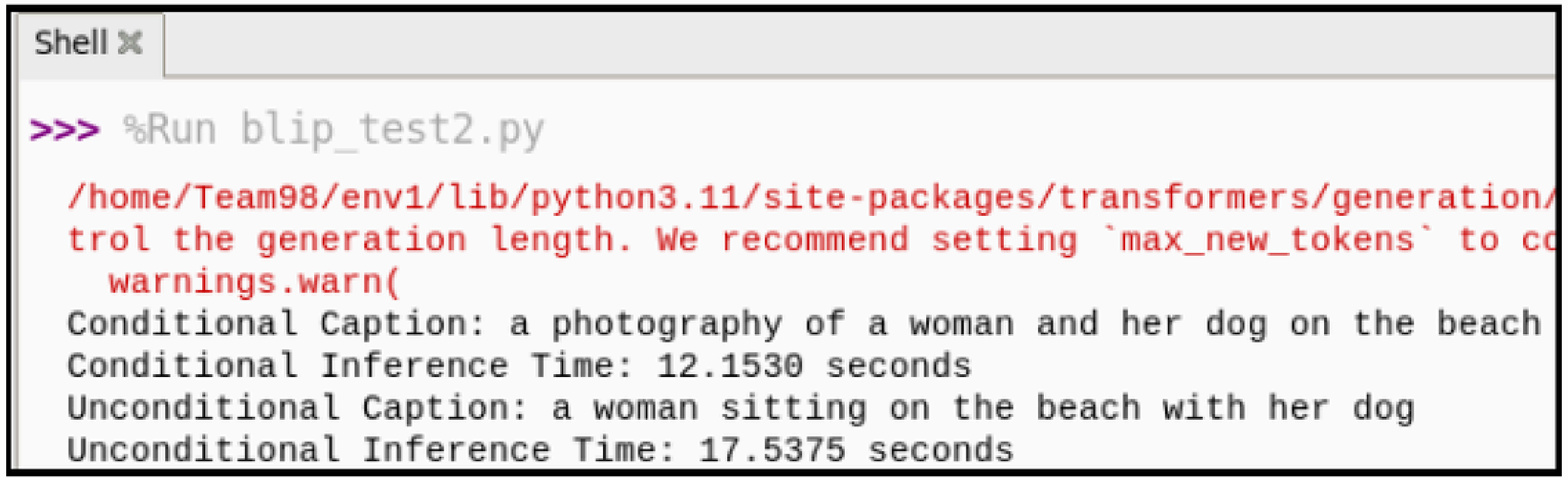


Figure 2. Shell operation screen

https://storage.googleapis.com/sfr-vision-language-research/BLIP/demo.jpg

(Written by 임채우)

Experiment result

- Table 2 shows the results of comparing inference times.

- As a result of measuring the inference time, it was confirmed that the inference time was faster when the Conditional Caption “a photography of” was included

.

- However, it was confirmed that the BLIP model took longer inference time than other CV models or NLP models.

- Therefore, if the BLIP model can be made lightweight, it is expected that it can be used as On-Device AI.

Score / Method	Inf. time (second)
Used Conditional Caption	12.15
Not Used Conditional Caption	17.53

Table 2 Performance measurement results

(Written by 임채우)