가천대학교 Gachon University

iRASC INTELLIGENT ROBOTICS AND AUTONOMOUS SYSTEM CONTROL LAB

iRASC Lab Seminar 2024

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models

Junnan Li, Dongxu Li, Silvio Savarese, Steven Hoi (ICML 2023)

Salesforce Research (# of citation : 2567)

2024.07.21. Sun

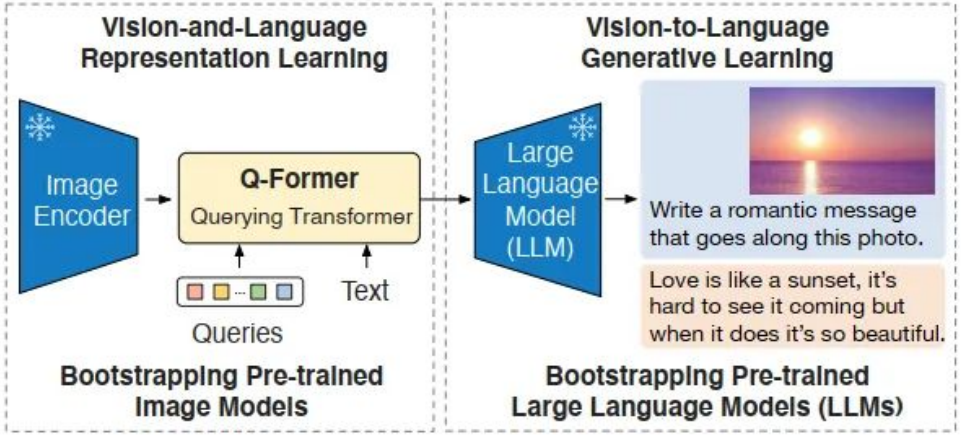
가천대학교 AI/소프트웨어학부 인공지능전공

iRASC 연구실 학부연구생 조태완

Contents

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models

Highlights

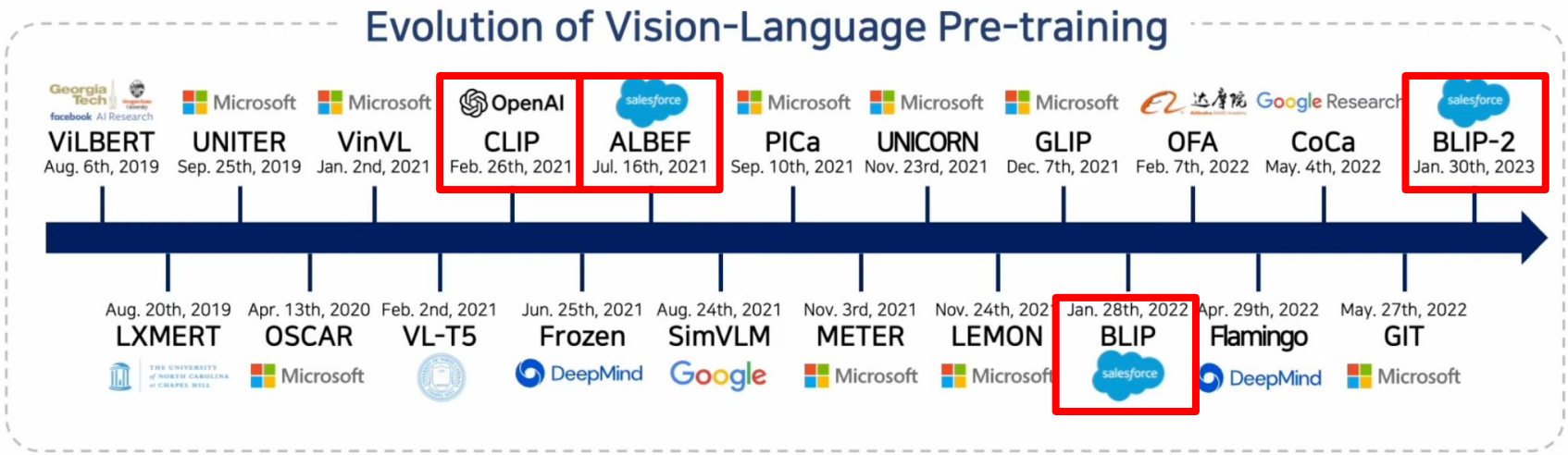


- Frozen model간의 modality gap을 메우기 위한 Q-Former라는 새로운 방법을 제시하고, 여러 VL task에서 성능을 향상시킴
- Pre-Training 과정에서의 Trainable Parameter 개수를 줄여 학습 효율성을 높임
- Multimodal 챗봇의 가능성을 제시

Background

✓ VLP의 진화 과정

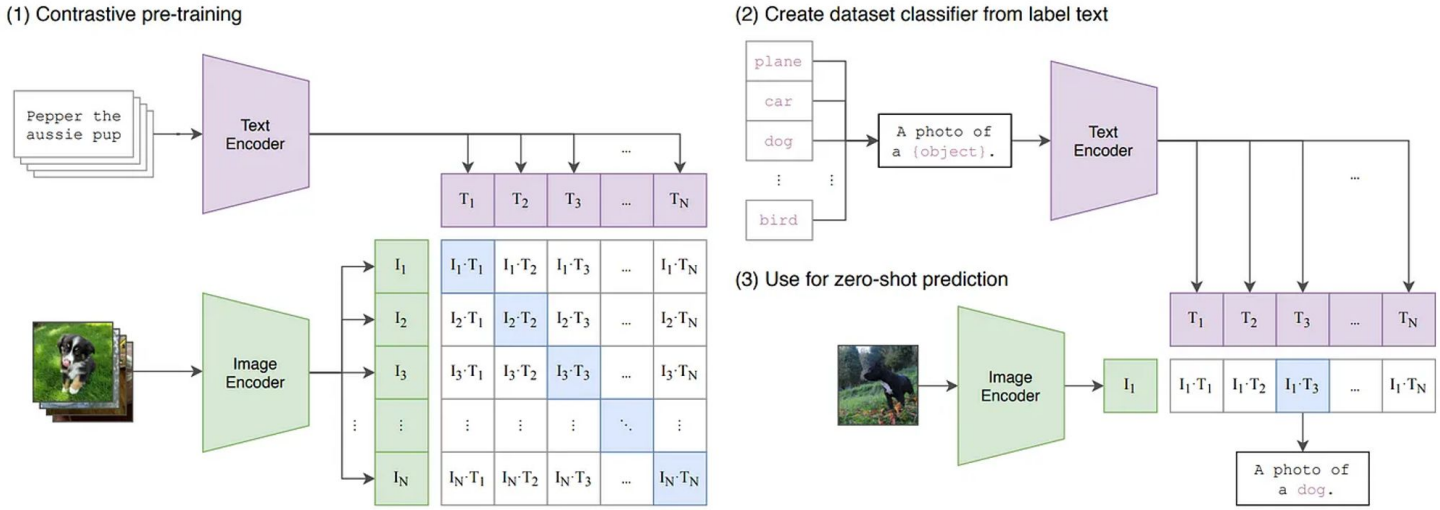
- Transformer가 발전함에 따라 대량의 데이터로 사전 학습하는 모델이 부각
- Pre-training 모델을 학습하고 Downstream Task 성능을 개선하는 방법은 Multimodal Learning의 초점이 되고 있음



Background

✓ CLIP (Contrastive Language-Image Pre-training)

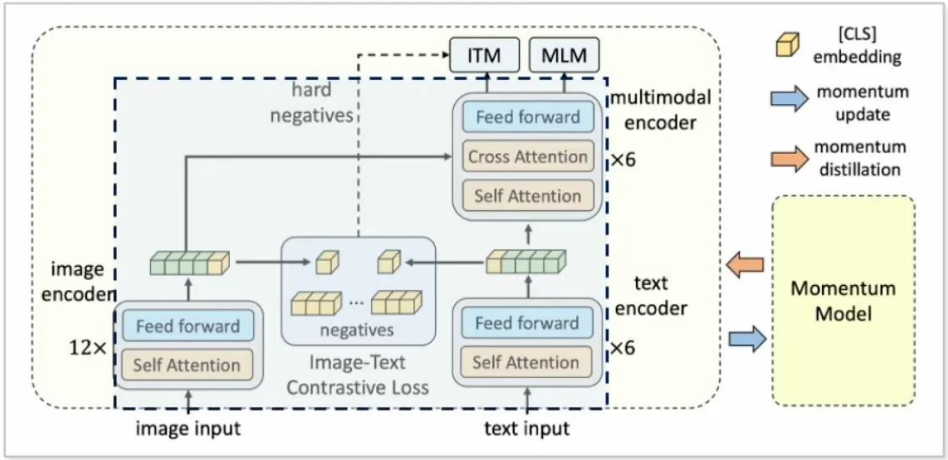
- Encoder base모델로 이미지-텍스트를 Contrastive Learning하는 것이 핵심
- 웹에서 직접 4억개의 데이터셋을 만들어 학습시키고 Zero-shot 성능이 뛰어나다는 것이 특징



Background

✓ ALBEF (ALign the image & text representations BEfore Fusing)

- Encoder base모델로 이미지-텍스트의 representation을 align시켜 학습하는 것이 핵심
- 또한 Momentum Model에 의해 생성된 Pseudo-target으로 학습하는 momentum distillation을 사용

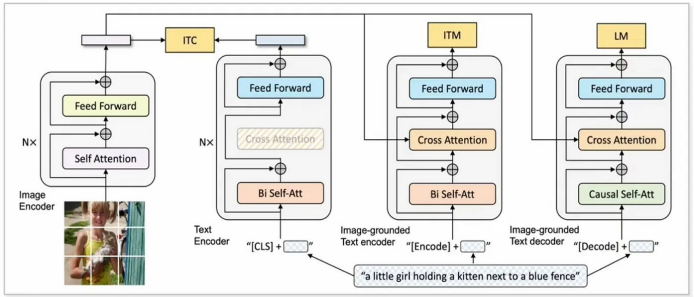


< ALBEF Architecture >

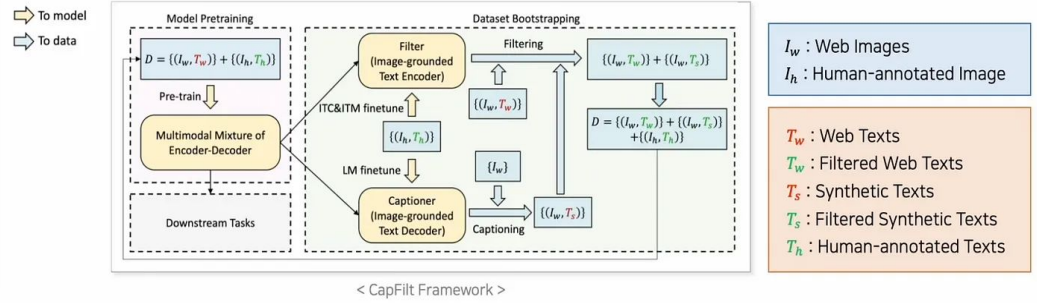
Background

✓ BLIP (Bootstrapping Language-Image Pre-training)

- ALBEF에서 제시한 모델과 유사한 구조
- Text generation, Image-text retrieval task를 모두 잘하는 새로운 모델 구조 제시(MED)
- CapFilt를 통해 web data의 noisy caption 문제 해결



< BLIP Architecture >

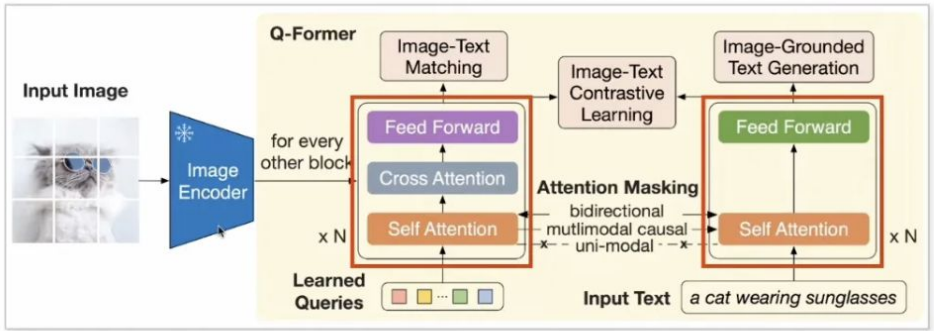


< CapFilt Framework >

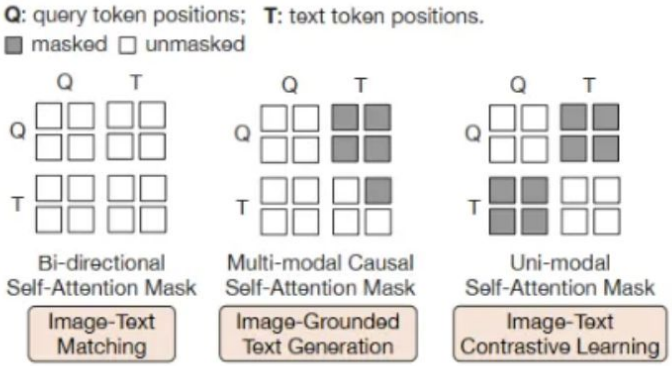
Method

✓ Q-Former

- Image Encoder에서 고정된 수의 Feature를 추출해서 Q-Former에 전달
- Query가 텍스트에서 가장 많은 정보를 제공하는 Visual Representation을 추출하는 방법을 학습할 수 있도록 Q-Former를 학습
- **최종적으로 Prompt와 Q-Former의 출력을 Frozen LLM에게 넘겨주어 텍스트를 생성**



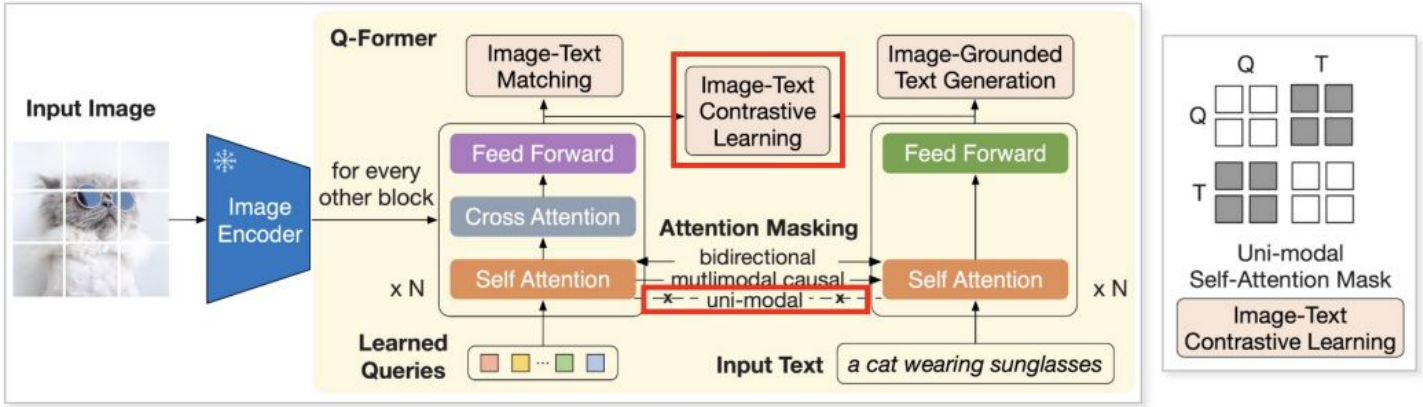
< BLIP-2 1st Stage Architecture >



Method

✓ ITC (Image-Text Contrastive Loss)

- **Image representation과 text representation을 align하는 것이 목표**
- 이 때, 이미지와 텍스트가 서로 정보를 교환할 수 없도록 Uni-modal Self-Attention Mask를 사용

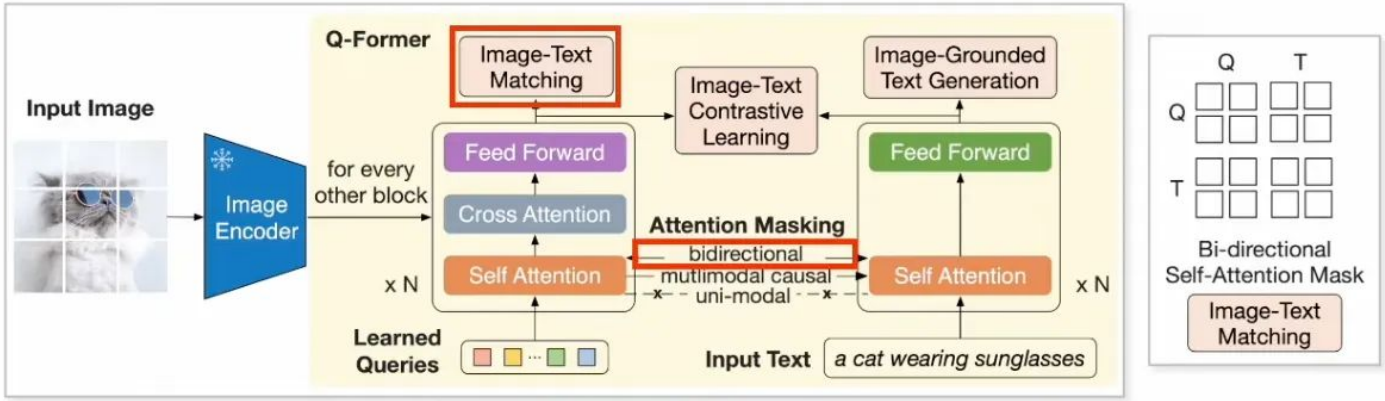


< BLIP-2 1st Stage Architecture >

Method

✓ ITM (Image-Text Matching)

- Image-Text Pair가 Positive인지 Negative인지 예측하도록 모델에 요청하는 이진 분류 작업
- 이 때, 모든 Query와 Text가 Bi-directional Self-Attention Mask를 사용합니다.
- Hard Negative Mining 전략을 채택하여 informative한 Negative Pair를 생성하여 학습

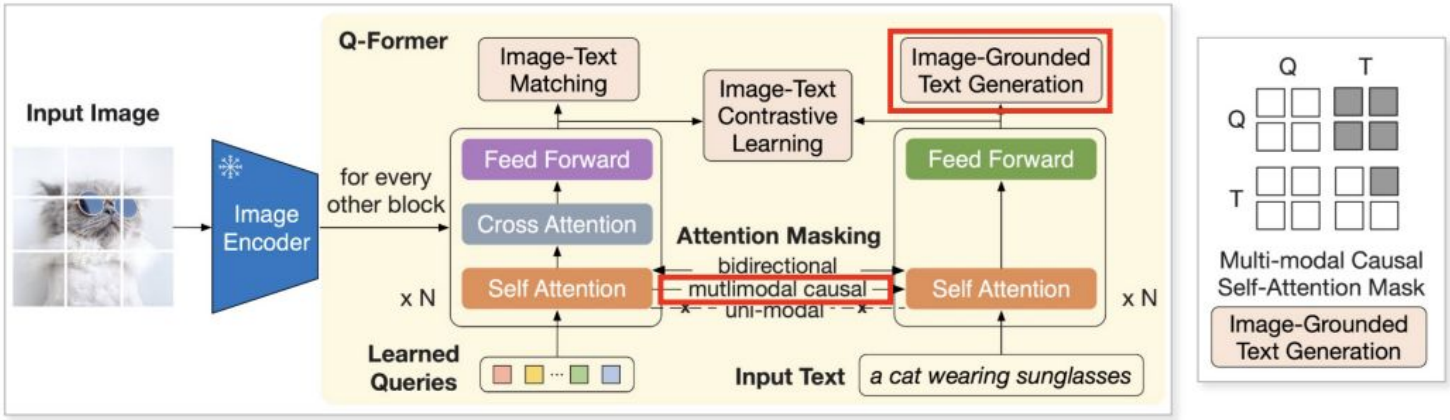


< BLIP-2 1st Stage Architecture >

Method

✓ ITG (Image-grounded Text Generation)

- 입력 이미지가 조건으로 주어지면 텍스트를 생성하도록 Q-Former를 학습
- Image Encoder에서 뽑아낸 이미지 정보는 Self-Attention layer를 통해 text tokens로 전달
- **이 때, Query는 Text와 관련된 이미지 정보들을 뽑도록 학습**

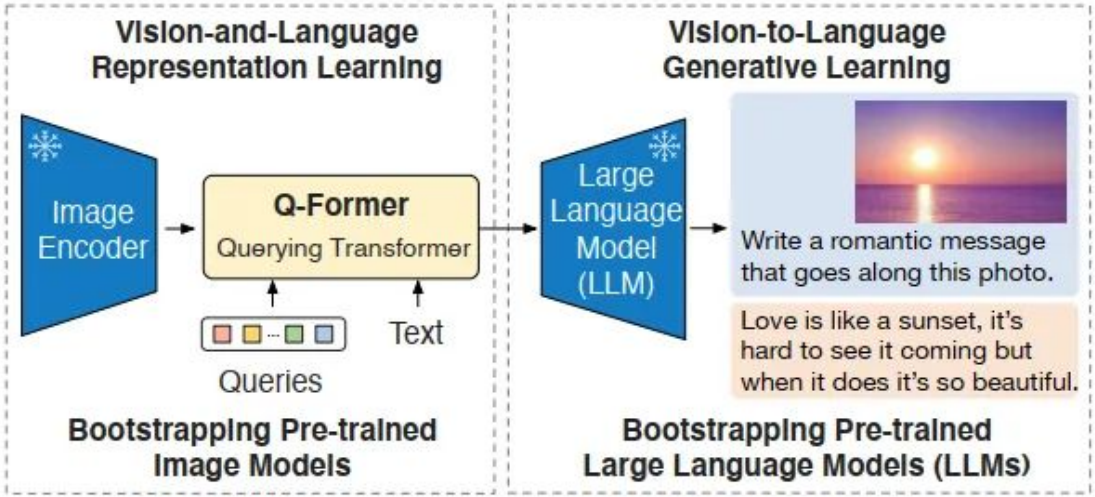


< BLIP-2 1st Stage Architecture >

Code Review

✓ How to train Q-Former?

- Pre-trained된 ViT와 LLM(ex Llama 3)을 가지고 중간에 정보를 전달해주는 Q-Former
- **32개(예시 코드 기준)의 Learnable Query를 통해 추출한 이미지 특징을 LLM에게 전달**



Code Review

✓ Q-Former 초기화

- 사전 훈련된 BERT 모델을 로드하고, 변경된 설정을 적용하여 새로운 Q-former 모델을 생성
- 쿼리 토큰을 무작위로 초기화되며, 초기화 방법은 정규 분포를 따름

```
@classmethod
def init_Qformer(cls, num_query_token, vision_width, cross_attention_freq=2):
    encoder_config = BertConfig.from_pretrained("bert-base-uncased")
    encoder_config.encoder_width = vision_width
    # insert cross-attention layer every other block
    encoder_config.add_cross_attention = True
    encoder_config.cross_attention_freq = cross_attention_freq
    encoder_config.query_length = num_query_token
    Qformer = BertLMHeadModel.from_pretrained(
        "bert-base-uncased", config=encoder_config
    )
    query_tokens = nn.Parameter(
        torch.zeros(1, num_query_token, encoder_config.hidden_size)
    )
    query_tokens.data.normal_(mean=0.0, std=encoder_config.initializer_range)
    return Qformer, query_tokens
```

Code Review

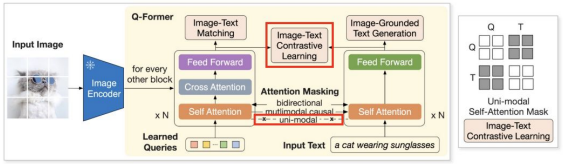
✓ 쿼리 특징과 텍스트 특징 추출

- 사전 학습된 ViT 모델을 로드하고 학습을 진행
- 이미지와 텍스트가 서로 정보를 교환할 수 없도록 Attention mask를 설정

```python
def forward(self, samples):
    image = samples["image"]
    text = samples["text_input"]

    image_embeds = self.ln_vision(self.visual_encoder(image))
    image_atts = torch.ones(image_embeds.size()[:-1], dtype=torch.long).to(
        image.device
    )

    query_tokens = self.query_tokens.expand(image_embeds.shape[0], -1, -1)

    query_output = self.Qformer.bert(
        query_embeds=query_tokens,
        encoder_hidden_states=image_embeds,
        encoder_attention_mask=image_atts,
        use_cache=True,
        return_dict=True,
    )

    image_feats = F.normalize(
        self.vision_proj(query_output.last_hidden_state), dim=-1
    )
```

```python
    text_tokens = self.tokenizer(
        text,
        padding="max_length",
        truncation=True,
        max_length=self.max_txt_len,
        return_tensors="pt",
    ).to(image.device)
    text_output = self.Qformer.bert(
        text_tokens.input_ids,
        attention_mask=text_tokens.attention_mask,
        return_dict=True,
    )
    text_feat = F.normalize(
        self.text_proj(text_output.last_hidden_state[:, 0, :]), dim=-1
    )
    ...
```

Code Review



< BLIP-2 1st Stage Architecture >

✓ Image-text Contrastive

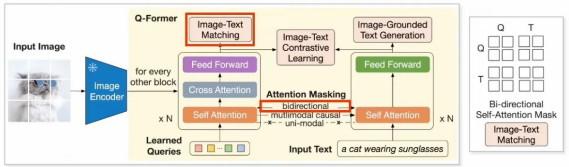
- 이미지(query)-텍스트 유사도와 텍스트-이미지(query) 유사도 계산
- 두 유사도를 기반으로 cross-entropy loss를 계산

```python
def forward(self, samples):
    ...

    ###============== Image-text Contrastive ===================###
    image_feats_all = concat_all_gather(
        image_feats
    )  # [batch_size*num_gpu, num_query_tokens, embed_dim]
    text_feat_all = concat_all_gather(text_feat)  # [batch_size*num_gpu, embed_dim]

    sim_q2t = torch.matmul(
        image_feats.unsqueeze(1), text_feat_all.unsqueeze(-1)
    ).squeeze()
    # [batch_size, batch_size*num_gpu, num_query_tokens]

    # image-text similarity: aggregate across all query tokens
    sim_i2t, _ = sim_q2t.max(-1)
    sim_i2t = sim_i2t / self.temp

    # text-query similarity: [batch_size, batch_size*num_gpu, num_query_tokens]
    sim_t2q = torch.matmul(
        text_feat.unsqueeze(1).unsqueeze(1), image_feats_all.permute(0, 2, 1)
    ).squeeze()

    # text-image similarity: aggregate across all query tokens
    sim_t2i, _ = sim_t2q.max(-1)
    sim_t2i = sim_t2i / self.temp  # [batch_size, batch_size*num_gpu]
```

```python
rank = dist.get_rank()
bs = image.size(0)
targets = torch.linspace(rank * bs, rank * bs + bs - 1, bs, dtype=int).to(
    image.device
)

if "image_id" in samples.keys(): #coco retrieval finetuning
    image_ids = samples["image_id"].view(-1,1)
    image_ids_all = concat_all_gather(image_ids)
    pos_idx = torch.eq(image_ids, image_ids_all.t()).float()
    sim_targets = pos_idx / pos_idx.sum(1,keepdim=True)
    sim_targets = 0.9 * sim_targets \
                + 0.1 * torch.ones_like(sim_targets) / sim_targets.size(1)

    loss_t2i = -torch.sum(F.log_softmax(sim_t2i, dim=1)*sim_targets,dim=1).mean()
    loss_i2t = -torch.sum(F.log_softmax(sim_i2t, dim=1)*sim_targets,dim=1).mean()
    loss_itc = (loss_t2i+loss_i2t)/2
else:
    loss_itc = (
        F.cross_entropy(sim_i2t, targets, label_smoothing=0.1)
        + F.cross_entropy(sim_t2i, targets, label_smoothing=0.1)
    ) / 2
```

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models

Code Review



< BLIP-2 1st Stage Architecture >

✓ Image-text Matching

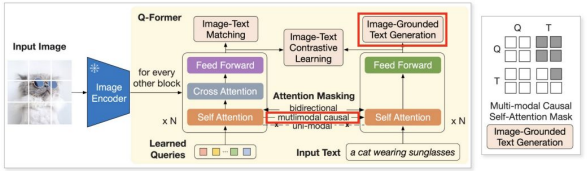
- query-text의 매칭 여부를 판단하는 분류 작업
- 정답 샘플을 미리 음수로 만들고, Hard Negative Mining 학습

```python
def forward(self, samples):
    ...

    ###============== Image-text Matching ===================###
    text_input_ids_world = concat_all_gather(text_tokens.input_ids)
    text_attention_mask_world = concat_all_gather(text_tokens.attention_mask)
    image_embeds_world = all_gather_with_grad(image_embeds)
    with torch.no_grad():
        if "image_id" in samples.keys():
            mask = torch.eq(image_ids, image_ids_all.t())
            sim_t2i.masked_fill_(mask, -10000)
            sim_i2t.masked_fill_(mask, -10000)
        else:
            sim_t2i[:, rank * bs : rank * bs + bs].fill_diagonal_(-10000)
            sim_i2t[:, rank * bs : rank * bs + bs].fill_diagonal_(-10000)

        weights_t2i = F.softmax(sim_t2i, dim=1)
        weights_i2t = F.softmax(sim_i2t, dim=1)

    # select a negative image for each text
    image_embeds_neg = []
    for b in range(bs):
        neg_idx = torch.multinomial(weights_t2i[b], 1).item()
        image_embeds_neg.append(image_embeds_world[neg_idx])
    image_embeds_neg = torch.stack(image_embeds_neg, dim=0)
```

```python
# select a negative image for each text
image_embeds_neg = []
for b in range(bs):
    neg_idx = torch.multinomial(weights_t2i[b], 1).item()
    image_embeds_neg.append(image_embeds_world[neg_idx])
image_embeds_neg = torch.stack(image_embeds_neg, dim=0)

# select a negative text for each image
text_ids_neg = []
text_atts_neg = []
for b in range(bs):
    neg_idx = torch.multinomial(weights_i2t[b], 1).item()
    text_ids_neg.append(text_input_ids_world[neg_idx])
    text_atts_neg.append(text_attention_mask_world[neg_idx])

text_ids_neg = torch.stack(text_ids_neg, dim=0)
text_atts_neg = torch.stack(text_atts_neg, dim=0)

text_ids_all = torch.cat(
    [text_tokens.input_ids, text_tokens.input_ids, text_ids_neg], dim=0
)  # pos, pos, neg
text_atts_all = torch.cat(
    [text_tokens.attention_mask, text_tokens.attention_mask, text_atts_neg],
    dim=0,
)
```

```python
query_tokens_itm = self.query_tokens.expand(text_ids_all.shape[0], -1, -1)
query_atts_itm = torch.ones(query_tokens_itm.size()[:-1], dtype=torch.long).to(
    image.device
)
attention_mask_all = torch.cat([query_atts_itm, text_atts_all], dim=1)

image_embeds_all = torch.cat(
    [image_embeds, image_em beds_neg, image_embeds], dim=0
)  # pos, neg, pos
image_atts_all = torch.ones(image_embeds_all.size()[:-1], dtype=torch.long).to(
    image.device
)

output_itm = self.Qformer.bert(
    text_ids_all,
    query_embeds=query_tokens_itm,
    attention_mask=attention_mask_all,
    encoder_hidden_states=image_embeds_all,
    encoder_attention_mask=image_atts_all,
    return_dict=True,
)

vl_embeddings = output_itm.last_hidden_state[:, : query_tokens_itm.size(1), :]
vl_output = self.itm_head(vl_embeddings)
logits = vl_output.mean(dim=1)

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(logits, itm_labels)
```

Code Review



< BLIP-2 1st Stage Architecture >

✓ Image-grounded Text Generation (LM)

- 디코더 입력과 어텐션 마스크를 통해 언어 모델을 실행
- loss는 모델이 예측한 텍스트와 실제 텍스트 간의 차이

```python
def forward(self, samples):
    ...

    ##================ Image Captioning ======================##
    decoder_input_ids = text_tokens.input_ids.clone()
    decoder_input_ids[:, 0] = self.tokenizer.bos_token_id
    labels = decoder_input_ids.masked_fill(
        decoder_input_ids == self.tokenizer.pad_token_id, -100
    )

    query_atts = torch.ones(query_tokens.size()[:-1], dtype=torch.long).to(
        image.device
    )
    attention_mask = torch.cat([query_atts, text_tokens.attention_mask], dim=1)
    lm_output = self.Qformer(
        decoder_input_ids,
        attention_mask=attention_mask,
        past_key_values=query_output.past_key_values,
        return_dict=True,
        labels=labels,
    )

    loss_lm = lm_output.loss

    return BlipOutput(
        loss=loss_itc + loss_itm + loss_lm,
        loss_itc=loss_itc,
        loss_itm=loss_itm,
        loss_lm=loss_lm,
    )
```

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

Code Review

✓ BLIP-2 with Vicuna

- Llama 기반 모델, ViT와 Q-former를 결합하여 Vision기반 Text 생성
- **Vision Encoder와 LLM의 파라미터를 학습 가능하지 않도록 고정**
- Q-Former의 출력을 LLM의 입력으로 넣기 위한 함수 정의(Linear Projection)

```
for name, param in self.visual_encoder.named_parameters():
    param.requires_grad = False

for name, param in self.llm_model.named_parameters():
    param.requires_grad = False
```

```
self.llm_proj = nn.Linear(
    self.Qformer.config.hidden_size, self.llm_model.config.hidden_size
)
```

Code Review

✓ BLIP-2 with Vicuna

- Q-former에서 나온 결과값을 linear projection 후 llm에 (Q-former 출력 + 입력 텍스트 + 출력 텍스트) 형식의 텐서로 입력 -> **생성 텍스트, 출력 텍스트간 loss를 통해 Q-former를 학습**

```python
def forward(self, samples):

    image = samples["image"]
    with self.maybe_autocast():
        image_embeds = self.ln_vision(self.visual_encoder(image))
    image_atts = torch.ones(image_embeds.size()[:-1], \
                dtype=torch.long).to(image.device)

    bs = image.size(0)

    query_tokens = self.query_tokens.expand(image_embeds.shape[0], -1, -1)
    if self.qformer_text_input:
        text_Qformer = self.tokenizer(
            samples["text_input"],
            padding='longest',
            truncation=True,
            max_length=self.max_txt_len,
            return_tensors="pt",
        ).to(image.device)
        query_atts = torch.ones(query_tokens.size()[:-1], \
                    dtype=torch.long).to(image.device)
        Qformer_atts = torch.cat([query_atts, text_Qformer.attention_mask],dim=1)

        query_output = self.Qformer.bert(
            text_Qformer.input_ids,
            attention_mask=Qformer_atts,
            query_embeds=query_tokens,
            encoder_hidden_states=image_embeds,
            encoder_attention_mask=image_atts,
            return_dict=True,
        )
    else:
        query_output = self.Qformer.bert(
            query_embeds=query_tokens,
            encoder_hidden_states=image_embeds,
            encoder_attention_mask=image_atts,
            return_dict=True,
        )
```

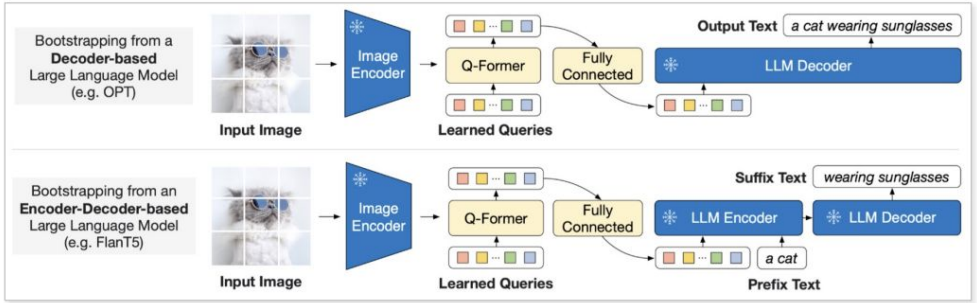
```python
inputs_llm = self.llm_proj(query_output.last_hidden_state[:,:query_tokens.size(1),:])
atts_llm = torch.ones(inputs_llm.size()[:-1], dtype=torch.long).to(image.device)

self.llm_tokenizer.padding_side = "right"
self.llm_tokenizer.truncation_side = 'left'
text_input_tokens = self.llm_tokenizer(
    samples['text_input'],
    return_tensors="pt",
    padding="longest",
    truncation=True,
    max_length=self.max_txt_len,
).to(image.device)

self.llm_tokenizer.truncation_side = 'right'
text_output_tokens = self.llm_tokenizer(
    [t + self.llm_tokenizer.eos_token for t in samples['text_output']],
    return_tensors="pt",
    padding="longest",
    truncation=True,
    max_length=self.max_output_txt_len,
).to(image.device)

llm_tokens, input_part_targets_len = self.concat_text_input_output(
    text_input_tokens.input_ids,
    text_input_tokens.attention_mask,
    text_output_tokens.input_ids,
    text_output_tokens.attention_mask,
)
```

```python
# do not apply loss to the padding
targets = llm_tokens['input_ids'].masked_fill(
    llm_tokens['input_ids'] == self.llm_tokenizer.pad_token_id, -100
)

# do not apply loss to the text input (i.e., instruction)
for i, l in enumerate(input_part_targets_len):
    targets[i][:l] = -100

# do not apply loss to the query tokens
empty_targets = (
    torch.ones(atts_llm.size(), dtype=torch.long).to(image.device).fill_(-100)
)
targets = torch.cat([empty_targets, targets], dim=1)

inputs_embeds = self.llm_model.get_input_embeddings()(llm_tokens['input_ids'])
inputs_embeds = torch.cat([inputs_llm, inputs_embeds], dim=1)
attention_mask = torch.cat([atts_llm, llm_tokens['attention_mask']], dim=1)

with self.maybe_autocast():
    outputs = self.llm_model(
        inputs_embeds=inputs_embeds,
        attention_mask=attention_mask,
        return_dict=True,
        labels=targets,
    )

loss = outputs.loss

return {"loss": loss}
```

Experiments

✓ Benchmark & Setup

- BLIP과 동일한 사전학습 데이터 셋 사용
  - COCO, Visual Genome, CC3M, CC12M, SBU, LAION 400M (Total 129M)
- 웹 이미지에 대한 Synthetic Caption을 생성하기 위해 CapFilt 방법 채택
- Frozen Image Encoder
  - CLIP - ViT-L/14,
  - EVA-CLIP - ViT-g/14
- Frozen Language Model
  - OPT (Decoder-based)
  - FlanT5 (Encoder-Decoder-based)

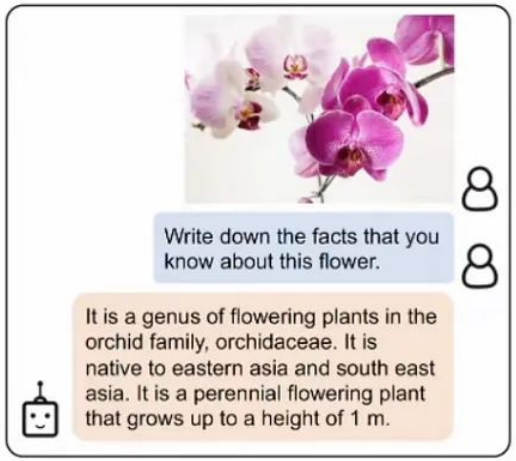


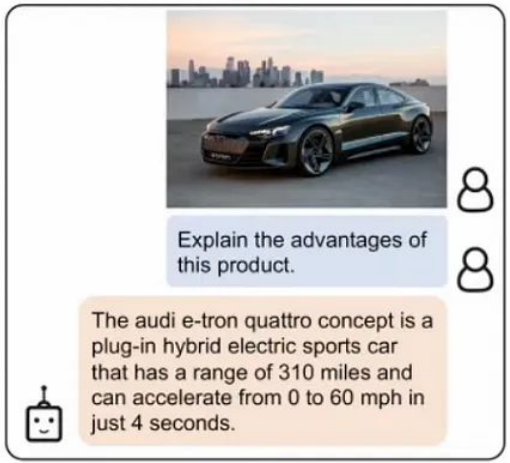
< BLIP-2 2nd Stage Architecture >

Experiments

✓ Zero-Shot Image-to-Text Generation

- Image-to-Text 생성을 위해, Image에 Text를 붙여서 LLM의 Input으로 넣어주어 실험을 진행





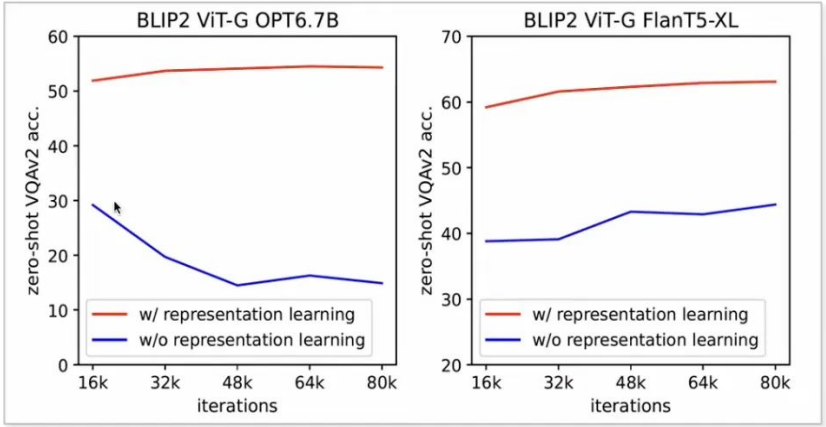
Experiments

✓ Zero-Shot Image-to-Text Generation

- 전반적으로 **더 강력한 Image Encoder 또는 LLM을 사용하는 것이 더 높은 성능**을 달성
- Q-former의 Representation Learning을 생략할 경우 성능이 저하됨, 그 중요성을 강조

Models	#Trainable Params	#Total Params	VQAv2 val	VQAv2 test-dev	OK-VQA test	GQA test-dev
VL-T5$_{no\text{-}vqa}$	224M	269M	13.5	-	5.8	6.3
FewVLM (Jin et al., 2022)	740M	785M	47.7	-	16.5	29.3
Frozen (Tsimpoukelli et al., 2021)	40M	7.1B	29.6	-	5.9	-
VLKD (Dai et al., 2022)	406M	832M	42.6	44.5	13.3	-
Flamingo3B (Alayrac et al., 2022)	1.4B	3.2B	-	49.2	41.2	-
Flamingo9B (Alayrac et al., 2022)	1.8B	9.3B	-	51.8	44.7	-
Flamingo80B (Alayrac et al., 2022)	10.2B	80B	-	56.3	**50.6**	-
BLIP-2 ViT-L OPT$_{2.7B}$	104M	3.1B	50.1	49.7	30.2	33.9
BLIP-2 ViT-g OPT$_{2.7B}$	107M	3.8B	53.5	52.3	31.7	34.6
BLIP-2 ViT-g OPT$_{6.7B}$	108M	7.8B	54.3	52.6	36.4	36.4
BLIP-2 ViT-L FlanT5$_{XL}$	103M	3.4B	62.6	62.3	39.4	44.4
BLIP-2 ViT-g FlanT5$_{XL}$	107M	4.1B	63.1	63.0	40.7	44.2
BLIP-2 ViT-g FlanT5$_{XXL}$	108M	12.1B	**65.2**	**65.0**	45.9	**44.7**

Table 2. Comparison with state-of-the-art methods on zero-shot visual question answering.



Experiments

✓ Image Captioning

- Prompt를 “a photo of”로 사용하여 이미지가 어떤 이미지인지 분류할 수 있도록 실험
- No Caps Zero-shot 성능이 상당히 개선됨과 동시에 SOTA를 달성 -> **강력한 일반화 성능**

Models	#Trainable Params	NoCaps Zero-shot (validation set)								COCO Fine-tuned Karpathy test	
		in-domain		near-domain		out-domain		overall			
		C	S	C	S	C	S	C	S	B@4	C
OSCAR (Li et al., 2020)	345M	-	-	-	-	-	-	80.9	11.3	37.4	127.8
VinVL (Zhang et al., 2021)	345M	103.1	14.2	96.1	13.8	88.3	12.1	95.5	13.5	38.2	129.3
BLIP (Li et al., 2022)	446M	114.9	15.2	112.1	14.9	115.3	14.4	113.2	14.8	40.4	136.7
OFA (Wang et al., 2022a)	930M	-	-	-	-	-	-	-	-	**43.9**	145.3
Flamingo (Alayrac et al., 2022)	10.6B	-	-	-	-	-	-	-	-	-	138.1
SimVLM (Wang et al., 2021b)	~1.4B	113.7	-	110.9	-	115.2	-	112.2	-	40.6	143.3
BLIP-2 ViT-g $OPT_{2.7B}$	1.1B	123.0	15.8	117.8	15.4	123.4	**15.1**	119.7	15.4	43.7	**145.8**
BLIP-2 ViT-g $OPT_{6.7B}$	1.1B	**123.7**	15.8	119.2	15.3	124.4	14.8	121.0	15.3	43.5	145.2
BLIP-2 ViT-g $FlanT5_{XL}$	1.1B	**123.7**	**16.3**	**120.2**	**15.9**	**124.8**	**15.1**	**121.6**	**15.8**	42.4	144.5

Table 3. Comparison with state-of-the-art image captioning methods on NoCaps and COCO Caption. All methods optimize the cross-entropy loss during finetuning. C: CIDEr, S: SPICE, B@4: BLEU@4.

Experiments

✓ Visual Question Answering

- Open-ended Generation Models은 Classification 모델과 달리 특정한 정답(라벨)이 정해져 있지 않은 다양한 응답이나 결과물을 생성할 수 있는 모델 (ex ChatGPT)
- VQA작업에서 Open-ended Generation Models 중 SOTA를 달성

Models	#Trainable Params	VQAv2 test-dev	VQAv2 test-std
Open-ended generation models			
ALBEF (Li et al., 2021)	314M	75.84	76.04
BLIP (Li et al., 2022)	385M	78.25	78.32
OFA (Wang et al., 2022a)	930M	82.00	82.00
Flamingo80B (Alayrac et al., 2022)	10.6B	82.00	82.10
BLIP-2 ViT-g FlanT5$_{XL}$	1.2B	81.55	81.66
BLIP-2 ViT-g OPT$_{2.7B}$	1.2B	81.59	81.74
BLIP-2 ViT-g OPT$_{6.7B}$	1.2B	**82.19**	**82.30**
Closed-ended classification models			
VinVL	345M	76.52	76.60
SimVLM (Wang et al., 2021b)	~1.4B	80.03	80.34
CoCa (Yu et al., 2022)	2.1B	82.30	82.30
BEIT-3 (Wang et al., 2022b)	1.9B	**84.19**	**84.03**

Table 4. Comparison with state-of-the-art models fine-tuned for visual question answering.

Experiments

✓ Image-Text Retrieval

- k를 128로 사용한 결과 Zero-Shot Image-Text Retrieval에서 SOTA를 달성

Model	#Trainable Params	Flickr30K Zero-shot (1K test set)						COCO Fine-tuned (5K test set)					
		Image → Text			Text → Image			Image → Text			Text → Image		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
Dual-encoder models													
CLIP (Radford et al., 2021)	428M	88.0	98.7	99.4	68.7	90.6	95.2	-	-	-	-	-	-
ALIGN (Jia et al., 2021)	820M	88.6	98.7	99.7	75.7	93.8	96.8	77.0	93.5	96.9	59.9	83.3	89.8
FILIP (Yao et al., 2022)	417M	89.8	99.2	99.8	75.0	93.4	96.3	78.9	94.4	97.4	61.2	84.3	90.6
Florence (Yuan et al., 2021)	893M	90.9	99.1	-	76.7	93.6	-	81.8	95.2	-	63.2	85.7	-
BEIT-3(Wang et al., 2022b)	1.9B	94.9	99.9	**100.0**	81.5	95.6	97.8	84.8	96.5	98.3	67.2	**87.7**	**92.8**
Fusion-encoder models													
UNITER (Chen et al., 2020)	303M	83.6	95.7	97.7	68.7	89.2	93.9	65.7	88.6	93.8	52.9	79.9	88.0
OSCAR (Li et al., 2020)	345M	-	-	-	-	-	-	70.0	91.1	95.5	54.0	80.8	88.5
VinVL (Zhang et al., 2021)	345M	-	-	-	-	-	-	75.4	92.9	96.2	58.8	83.5	90.3
Dual encoder + Fusion encoder reranking													
ALBEF (Li et al., 2021)	233M	94.1	99.5	99.7	82.8	96.3	98.1	77.6	94.3	97.2	60.7	84.3	90.5
BLIP (Li et al., 2022)	446M	96.7	**100.0**	**100.0**	86.7	97.3	98.7	82.4	95.4	97.9	65.1	86.3	91.8
BLIP-2 ViT-L	474M	96.9	**100.0**	**100.0**	88.6	97.6	**98.9**	83.5	96.0	98.0	66.3	86.5	91.8
BLIP-2 ViT-g	1.2B	**97.6**	**100.0**	**100.0**	**89.7**	**98.1**	**98.9**	**85.4**	**97.0**	**98.5**	**68.3**	**87.7**	92.6

Table 5. Comparison with state-of-the-art image-text retrieval methods, finetuned on COCO and zero-shot transferred to Flickr30K.

Conclusion

✓ 이 연구는 **Frozen model**간의 **modality gap**을 메우기 위한 **Q-Former**라는 새로운 방법을 제시

- LLM이 단일 Image-Text 쌍을 학습했기 때문에 성능의 한계 -> 다중 데이터셋을 개발할 예정
- LLM 성능 의존성, LLM의 부정확한 지식으로부터의 논증은 정확하지 않은 결과 도출
- Pre-Training 과정에서의 Trainable Parameter 개수를 줄여 학습 효율성을 높임
- Multimodal 챗봇의 가능성을 제시함

✓ **VLM 발전의 큰 혁신을 가져온 연구**

- 다양한 종류의 모달리티를 고성능 LLM에 통합하여 쉽게 Multimodal LLM을 제작할 수 있는 가능성을 제시함
- 이미지 인코더와 LLM의 성능이 좋아짐에 따라 더 높은 VL Task의 Benchmark 점수 달성 가능
- 실제로 이후 VLM 모델들이 BLIP-2 기반으로 진행되고 있는 것을 확인

Reference

- https://arxiv.org/abs/2301.12597
- https://seandoprep.tistory.com/5
- https://seandoprep.tistory.com/6
- https://www.youtube.com/watch?v=rTEObsTGjNw
- https://github.com/salesforce/LAVIS
- https://medium.com/@taewan2002/clip-connecting-text-and-images-1c76cc1bae65
- https://medium.com/@taewan2002/blip-bootstrapping-language-image-pre-training-for-unified-vision-language-understanding-and-c78b6810aa8d
- https://medium.com/@taewan2002/blip-2-bootstrapping-language-image-pre-training-with-frozen-image-encoders-and-large-language-afb620f7af76

Thank You

Intelligent Robotics and Autonomous System Control Lab.

Undergraduate Student

조 태 완

Gachon University

School of Computing,
Dept. of AI-Software

taewan2002@gachon.ac.kr

Assistant Professor

최 재 용

Gachon University

School of Computing,
Dept. of AI-Software

andrewjchoi@gachon.ac.kr