2026 동계 세미나

Outliers: Meaning & Control

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
정윤성

Outline

- Background

- Outlier Suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling [EMNLP 2023]

- Vision Transformers Need Registers [ICLR 2024]

Background

- Quantization
  - FP32 data를 INT8와 같은 낮은 bit로 mapping하는 기술
  - 수식
    - $A_{int} = Round\left(\frac{X}{S}\right) + Z$
    - $A_{dequant} = (X_{int} - Z)$
      - S: $\frac{Max-Min}{2^b-1}$
      - Z: 0위치 맞추는 offset
      - b: bit-width

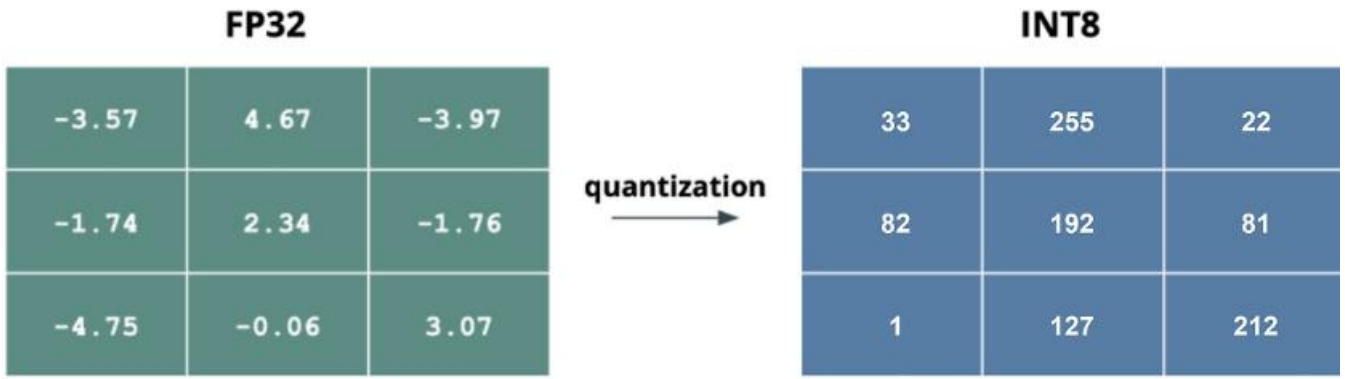


Figure 1. Quantization example

Outlier Suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling

[EMNLP 2023]

Introduction

- Problem setting
  - LLM에서 Transformer가 커질수록 특정 채널에서 값이 비정상적으로 큰 Outlier가 발생
    - Activation 값 중에 다른 값들보다 수백 배 큰 outliers가 존재
    - Outliers가 특정 채널에 집중되어 있을 뿐 아니라 그 분포가 0을 중심으로 하지 않고 한쪽으로 치우쳐 있음
  - 기존의 Min-Max Quantization을 쓰면 이 Outlier 때문에 Dynamic Range가 너무 넓어져서, 대부분의 정보들이 0으로 변환됨.
    - 대부분의 정보가 작은 값이기 때문
  - 또한 기존 Quant 방식은 outlier값을 scaling을 통해 줄이기만 했기 때문에 치우친 분포 문제를 해결하지 못해 효율이 낮았음

Introduction

- Problem setting
  - Channel shifting과 scaling을 통해 outlier의 영향을 최소화 하는 outlier suppression 방식을 제안

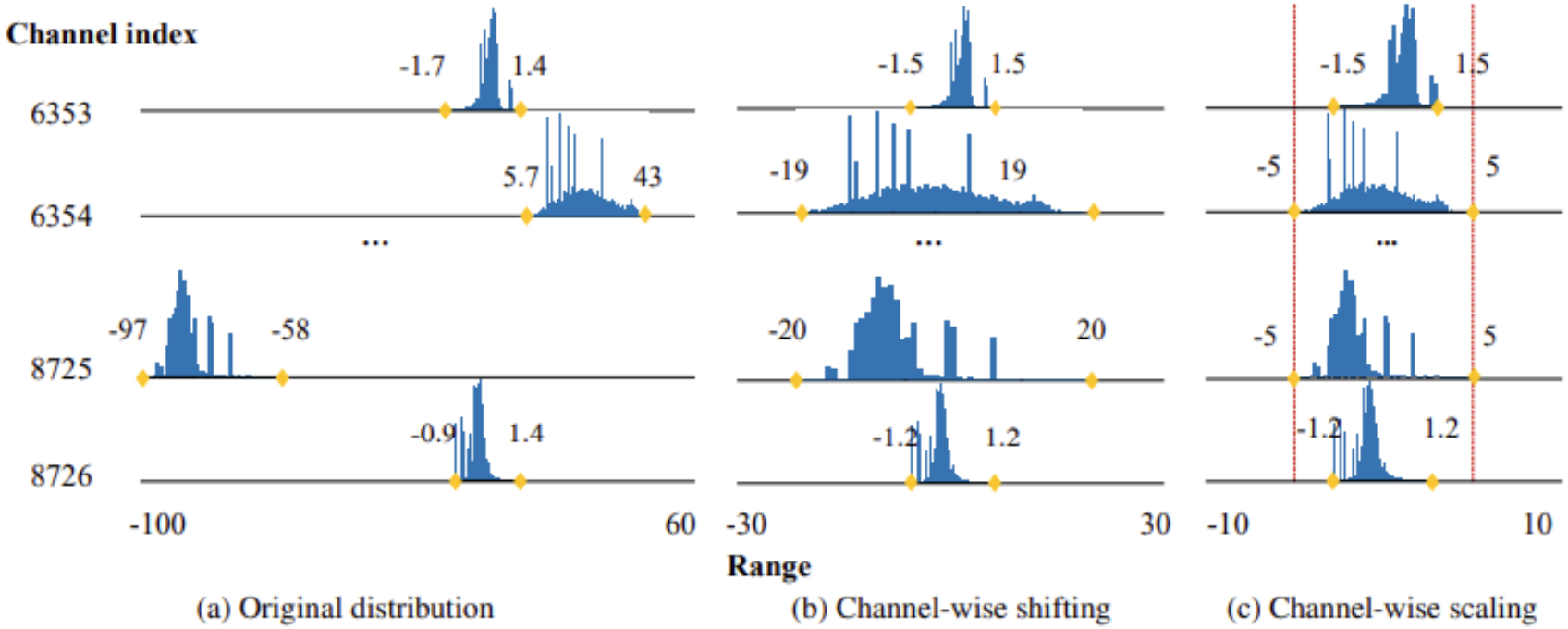


Figure 2. Outlier distribution

Method

- Channel-wise Shifting
  - 각 채널 j의 최솟값과 최댓값을 구한 후, center를 0으로 맞추기 위해 모든 값에서 z_j를 뺌
  - 수식
    - $\widetilde{X}' = X - z$
    - $z_j = \frac{\max(X_{:,j}) + \min(X_{:,j})}{2}$
    - Ex) range[-100,-60] → [-20,20]
      - 이를 통해 대칭 Quant를 적용해도 문제가 없음

Method

- Channel-wise Scaling
  - Shifting 후에도 여전히 값이 큰 채널은 스케일링 벡터 s로 나누어 범위를 더 좁힘
  - Activation 값을 조정한 만큼 weight와 bias를 수정하여 전체 출력 결과인 Y값은 변하지 않게 조정
    - Offline에서 변환이 이루어지기 때문에 실제 모델을 돌릴 때 추가적인 연산이 발생하지 않음
  - 수식
    - $\tilde{X} = (X - z) \oslash s$
    - $\widetilde{W} = W \odot s$
    - $\tilde{b} = zW^T + b$

① $Y = XW^T + b$

② $\tilde{X} = (X - z) \oslash s \Leftrightarrow X = (\tilde{X} \odot s) + z$

③ $Y = \{(\tilde{X} \odot s) + z\} W^T + b$

④ $Y = \tilde{X} \odot s \cdot W^T + zW^T + b$

$W' = W^T \cdot s^T \qquad b' = zW^T + b$

$Y = \tilde{X}W'^T + b'$

Method

- 최적화 전략
  - Activation의 outlier를 줄이기 위해서 s로 나누면 수학적 등가성을 유지하기 위해 W에는 s를 곱해줘야 함
    - S가 클 경우 Activation의 outlier는 줄어 Quant하기 좋아졌지만 Weight값이 커져서 Weight Quant할 때 오차가 커짐
  - 단순히 Activation 값만 줄이는 것이 아니라 Activation Quant, Weight Quant를 결합했을 때 최종 출력의 오차가 가장 적은 지점을 찾음
    - 단일 layer의 경우
      - Quant를 적용한 Q(X) * Q(W)값이 원래 값 X*W 와 가장 비슷한 s를 사용
    - Attention 구조의 경우
      - 최종적으로 계산된 attention map과 최종 출력이 Quant 전과 가장 비슷한 s를 사용

Method

- 최적화 전략
  - 각 채널에 맞게 s를 최적화 하기보다는 threshold t 값을 조금씩 바꿔가면서 가장 오차가 적은 t를 사용
    - 채널 범위가 t보다 크면 t만큼 나눠주고 t보다 작으면 그대로 유지
    - $s_j = \max\left(1.0, \frac{\max|X_{:,j} - z_j|}{t}\right)$
  - LayerNorm의 위치가 어디든 상관없이 등가 변환이 가능하기 때문

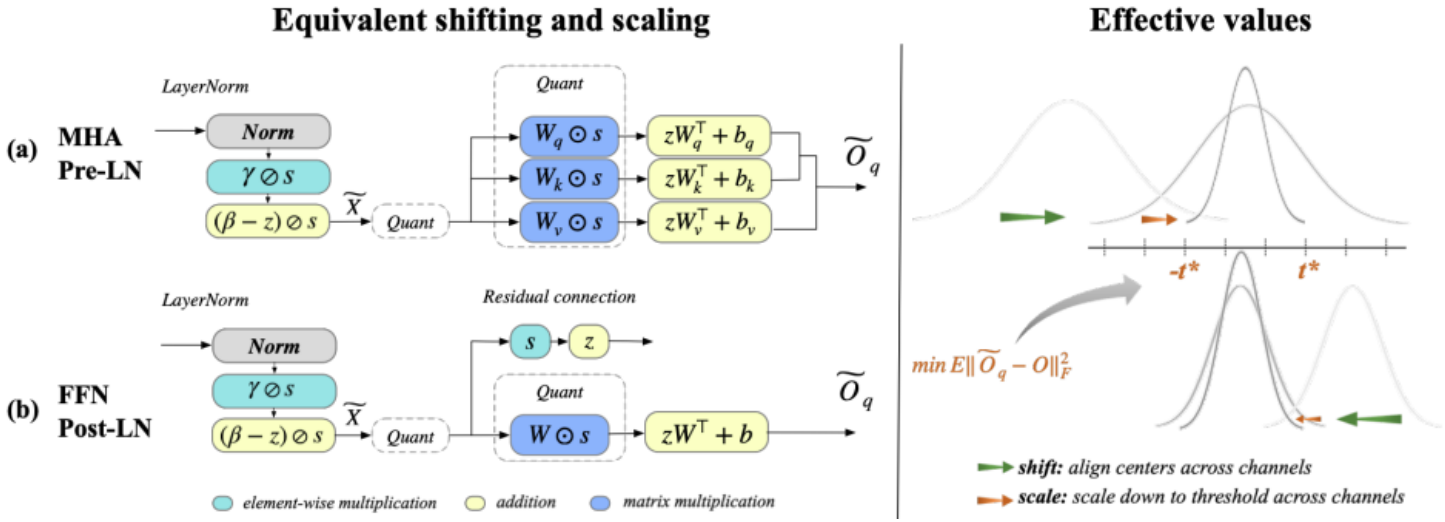


Figure 3. OS+(Outlier Suppression +) Pipeline

Experiments

- Setting
  - 대상 모델
    - Small: BERT-base, BERT-large
    - Large: OPT, BLOOM
  - Quant bit
    - INT8, 6, 4
  - Calibration Data)
    - Train dataset 중 무작위로 추출한 128개 sample만 사용
    - GLUE 과제는 해당 도메인 데이터를, LLM은 PILE 데이터셋을 활용
  - 비교 방법론
    - MinMax, Percentile, OMSE
    - PEG, Outlier Suppression, SmoothQuant, ZeroQuant

Experiments

- 정량 평가

Method	CoLA	MNLI	QNLI	SST-2	STS-B	Avg.
FP32	59.6	84.9	91.8	93.4	89.5	83.8
INT8*						
MinMax	52.3	81.3	89.0	91.1	86.2	79.5
OMSE	54.8	82.1	89.7	91.3	87.7	81.6
PEG	59.4	81.3	91.1	92.7	87.9	82.5
OS	60.3	83.9	90.2	**92.9**	88.2	83.0
OS+	**60.9**	**84.4**	**91.1**	92.7	**88.3**	**83.5**
INT6						
OMSE	35.4	73.7	84.7	86.3	85.8	73.5
Percentile	37.3	72.1	79.4	87.3	86.8	72.9
OS	54.4	81.8	89.8	91.9	88.7	81.2
OS+	**56.0**	**84.5**	**90.9**	**92.4**	**89.5**	**82.8**
INT4						
OMSE	4.7	38.5	52.2	50.3	0.2	41.1
Percentile	7.0	53.0	61.5	77.1	66.1	57.0
OS	28.5	57.9	72.5	80.4	67.8	62.7
OS+	**50.0**	**80.2**	**85.4**	**91.4**	**86.5**	**78.2**

Table 1.
일반적인 Quant 방법과의 비교

Model	Method	PIQA (↑)			Winogrande (↑)			HellaSwag (↑)			LAMBADA (↑)		
		FP16	INT8*	INT6	FP16	INT8*	INT6	FP16	INT8*	INT6	FP16	INT8*	INT6
OPT-13B	ZeroQuant	75.8	54.1	53.0	65.1	52.1	51.1	52.5	26.5	25.8	68.6	42.9	0.0
	SmoothQuant		76.0	73.5		64.9	60.3		52.2	49.2		68.3	65.2
	OS+		**76.4**	**75.8**		**65.0**	**64.0**		**52.3**	**51.7**		**68.3**	**65.7**
OPT-30B	ZeroQuant	77.6	54.2	52.0	68.5	51.8	51.8	54.3	26.4	25.7	71.5	9.7	0.0
	SmoothQuant		77.2	66.7		**68.2**	55.0		54.2	37.4		**71.0**	13.4
	OS+		**77.4**	**77.4**		68.0	**68.9**		**54.2**	**53.7**		70.8	**69.6**
OPT-66B	ZeroQuant	78.7	53.2	51.9	68.9	50.7	48.0	56.4	26.1	25.7	73.9	0.6	0.0
	SmoothQuant		78.3	52.0		68.3	52.1		55.9	26.5		72.9	0.0
	OS+		**78.7**	**77.5**		**69.0**	**69.4**		**56.2**	**55.8**		**73.0**	**72.7**
OPT-175B	ZeroQuant	79.7	52.3	53.1	72.5	50.2	49.1	59.3	25.4	25.6	74.7	0.0	0.0
	SmoothQuant		**79.7**	52.6		71.2	49.1		58.9	26.0		74.6	0.5
	OS+		79.6	**80.0**		**72.5**	**71.7**		**59.2**	**58.5**		**74.7**	**74.2**
BLOOM-176B	ZeroQuant	78.8	76.0	61.2	70.3	69.4	52.0	55.9	54.8	30.5	67.7	67.8	7.5
	SmoothQuant		77.7	76.7		68.6	67.6		54.1	52.1		**69.2**	60.2
	OS+		**78.4**	**78.1**		**69.8**	**70.3**		**55.2**	**54.8**		68.0	**69.2**
BLOOMZ-176B	ZeroQuant	80.6	79.1	54.0	72.5	70.9	49.6	57.1	56.3	28.2	67.8	67.6	1.4
	SmoothQuant		79.7	**80.0**		70.8	69.9		56.3	55.0		68.7	65.2
	OS+		**79.9**	79.9		**71.3**	**70.6**		**56.7**	**56.4**		**68.8**	**69.2**

Table 2.
다른 Quant 논문과의 비교

Conclusion

- Outlier가 channel 마다 다르게 발생한다는 사실을 발견
  - 이를 shifting과 scaling으로 조절하고 수학적 등가성을 이용해 해결

- 하지만 outlier가 발생하는 근본적인 원인이나 속성에 대해서는 알지 못한다는 한계가 존재

VISION TRANSFORMERS NEED REGISTERS

[ICLR 2024]

Introduction

- ViT 구조는 이미지를 패치 단위로 나누어 분석
  - 학습된 ViT모델의 feature map에서 특정 영역에서 값이 튀는 spike 현상이 발생함
- 이미지의 배경 영역에서 비정상적으로 큰 L2 norm을 가진 토큰이 나타남
  - Outlier 토큰들은 전체의 2%에 불과
    - 하지만 softmax연산 특성상 큰 값에 가중치가 쏠림
    - 따라서 attention map이 물체가 아닌 배경의 특정 지점에 쏠리는 현상 발생

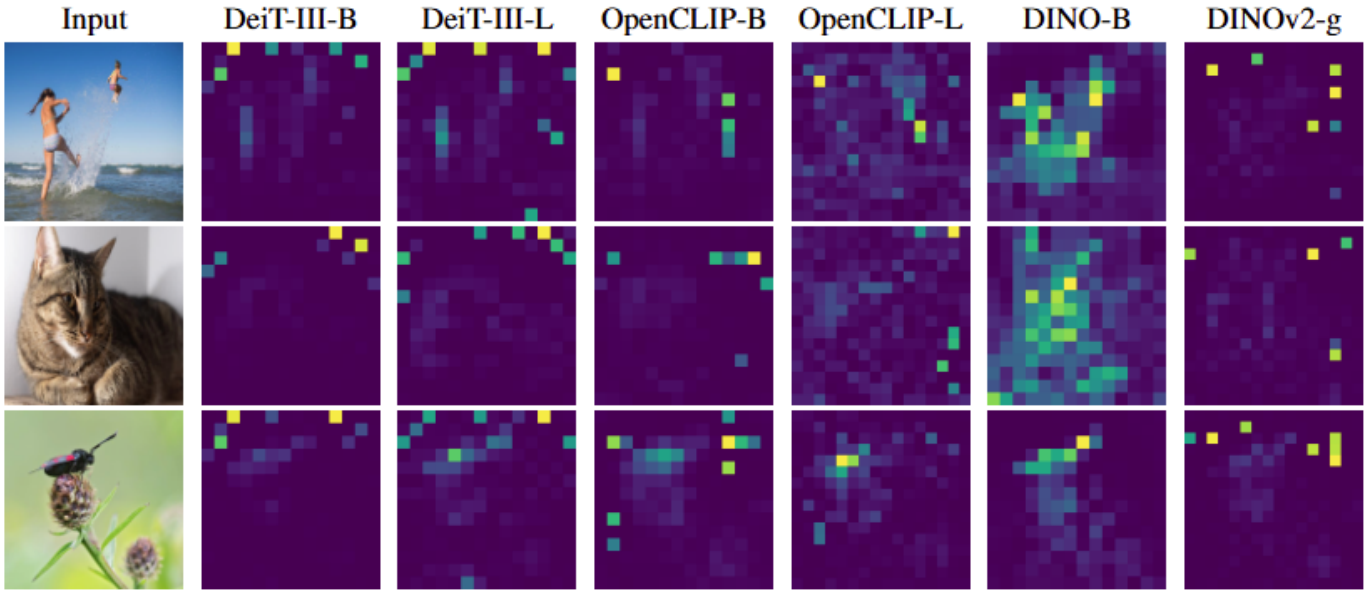
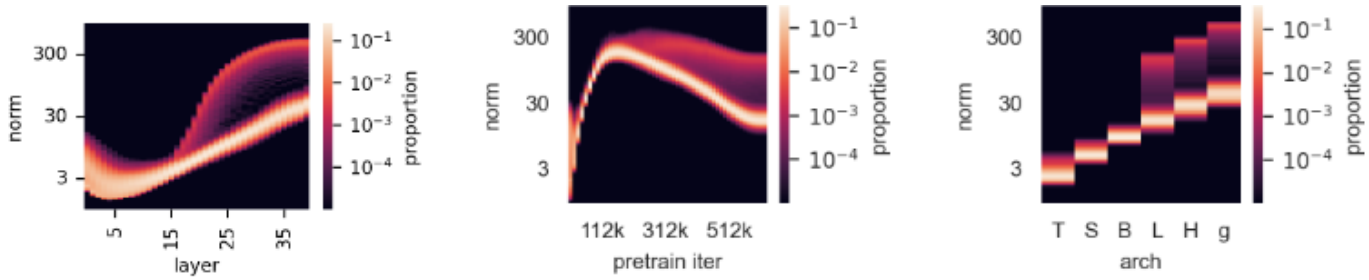


Figure 4. Attention map without registers

Introduction

- 모든 모델에서 발생하는 것이 아니라 모델의 크기가 커질수록, 학습 시간이 충분히 길 때 주로 발생
  - 같은 모델이라도 large에서는 발생하고 small에서는 발생하지 않을 수 있음
  - DINOv2가 DINO보다 성능이 좋지만 내부적으로 outlier가 발생



(a) Norms along layers. (b) Norms along iterations. (c) Norms across model size.

Figure 5. Outlier 발생 시기 및 조건

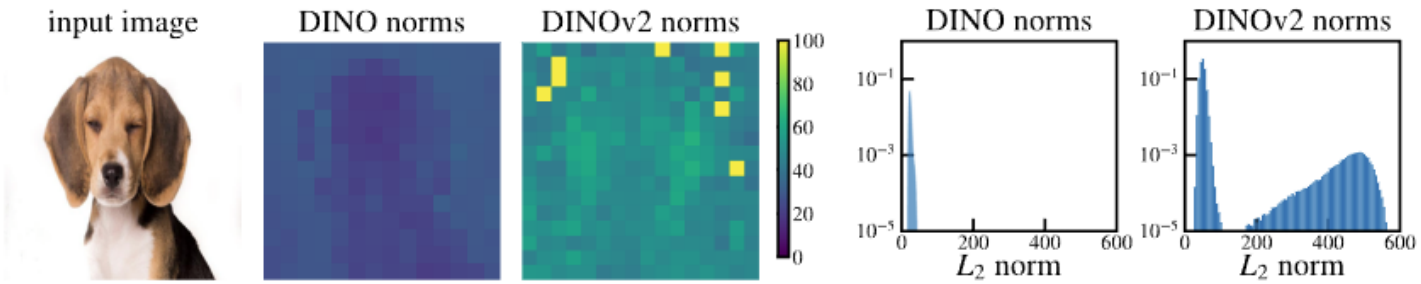


Figure 6. DINO, DINOv2 norm distribution

Introduction

- 가정
  - ViT 구조는 이미지를 처리하면서 global 정보를 특정 공간에 저장
    - 보통 [CLS] 토큰을 사용하지만 [CLS] 토큰 하나만으로는 용량이 부족
  - 따라서 [CLS] 토큰에 저장하고 남은 global 정보는 이미지 내에서 가장 정보가 적은 배경 패치에 저장
    - 즉 배경 패치를 덜 중요하다고 판단해 임시 register로 사용
    - 주변 패치와 유사도가 높아 local 정보로의 가치가 낮은 패치들을 사용
  - [CLS] 토큰, 일반 토큰, outlier 토큰을 사용해서 classification 실험을 진행
    - Outlier 토큰의 정확도가 일반 토큰보다 월등히 높고 [CLS]토큰의 정확도와 비슷

	IN1k	P205	Airc.	CF10	CF100	CUB	Cal101	Cars	DTD	Flow.	Food	Pets	SUN	VOC
[CLS]	**86.0**	**66.4**	**87.3**	**99.4**	**94.5**	**91.3**	96.9	**91.5**	**85.2**	**99.7**	**94.7**	**96.9**	**78.6**	89.1
normal	65.8	53.1	17.1	97.1	81.3	18.6	73.2	10.8	63.1	59.5	74.2	47.8	37.7	70.8
outlier	69.0	55.1	79.1	99.3	93.7	84.9	**97.6**	85.2	84.9	99.6	93.5	94.1	78.5	**89.7**

Table 3. 각 토큰을 사용한 classification

Introduction

- 가정
  - 하지만 Outlier 토큰으로 원래 이미지 패치를 복원하려고 하면 성능이 매우 나빠짐
    - Outlier 토큰은 주변 4개 패치와 매우 비슷한 영역에 발생
    - 일반 토큰에 비해 자기 위치를 예측하는 정확도가 떨어짐
  - 결론적으로 이 토큰은 원래 local 정보는 버리고 global 정보만 담고 있음

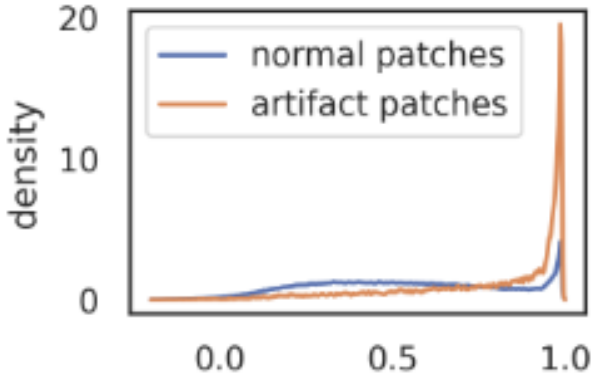


Figure 7. 주변 픽셀과의 cosine 유사도

	position prediction		reconstruction
	top-1 acc	avg. distance ↓	L2 error ↓
normal	**41.7**	**0.79**	**18.38**
outlier	22.8	5.09	25.23

Table 4. 각 토큰의 위치 예측

Method

- 모델이 global 정보를 저장할 공간이 필요하기 때문에 이미지와 관련 없는 빈 토큰(register)를 추가
  - 이미지를 패치 단위로 쪼갤 때 n개의 register를 추가로 넣음
  - 수식

$$X = \left[CLS, \underbrace{r_1, r_2, \ldots, r_n}_{\text{Registers}}, x_1, x_2, \ldots, x_L\right]$$

    - r_i: 학습 가능한 파라미터로 초기화된 register 토큰
    - Attention 방식을 통해 global 정보를 수집하는 역할

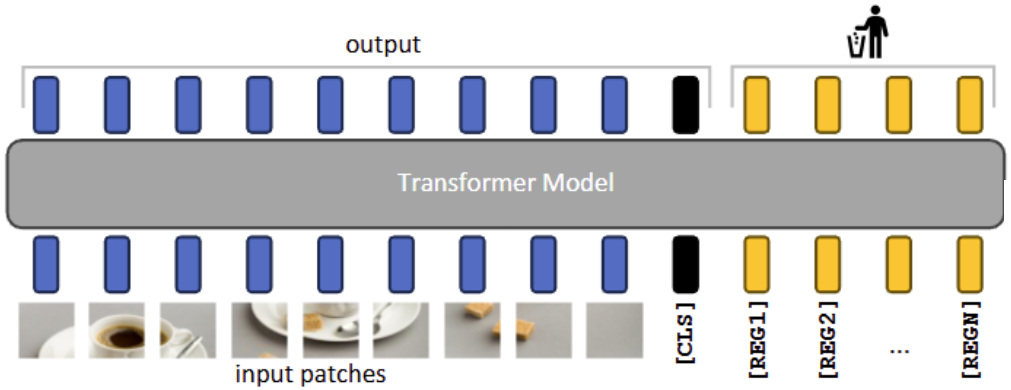


Figure 8. Image patch with registers

#registers	top-1 accuracy [CLS]	top-1 accuracy normal patch	top-1 accuracy outlier patch	top-1 accuracy register
0	84.6	15.5	73.3	N/A
1	85.2	14.5	N/A	71.1

Table 5. register 사용 시 global 정보의 이동

Method

- 배경 패치의 일부가 local 정보를 버리고 global 정보를 수집하지 않아도 됨
  - 배경 패치의 Norm이 정상 범위의 값으로 변경
  - 특정 배경 픽셀에 발생하는 attention spike 완화
- Register 토큰은 학습 중에만 사용되며 최종 출력 시에는 사용하지 않음
  - 최종 출력 시 기존과 동일하게 [CLS]와 이미지 패치 토큰만 사용
  - 기존 ViT 구조를 거의 변경하지 않음
    - 4개의 register를 추가했을 때 연산 증가량이 2% 미만
  - 추가적인 fine-tuning 없이 성능 향상

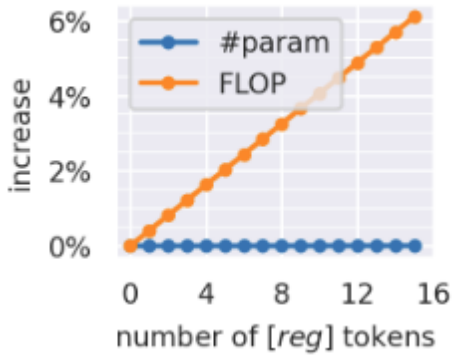


Figure 9. Register 추가 시 overhead

	ImageNet Top-1	ADE20k mIoU	NYUd rmse ↓
DeiT-III	84.7	38.9	0.511
DeiT-III+reg	84.7	39.1	0.512
OpenCLIP	78.2	26.6	0.702
OpenCLIP+reg	78.1	26.7	0.661
DINOv2	84.3	46.6	0.378
DINOv2+reg	84.8	47.9	0.366

(a) Linear evaluation with frozen features.

	ImageNet Top-1
OpenCLIP	59.9
OpenCLIP+reg	60.1

(b) Zero-shot classification.

Figure 10. Register 추가 시 성능 향상

Experiments

- Register를 1개만 추가해도 배경의 high-norm outlier가 즉각적으로 사라짐
  - Register를 많이 추가한다고 해서 성능이 선형적으로 좋아지는 것은 아님
- Registers를 사용했을 때 각 register의 역할을 따로 지정해 주지 않아도 물체를 따로 관찰하는 현상 발견
  - Register를 사용했을 때 객체 탐지나 segmentation 성능이 향상된 이유

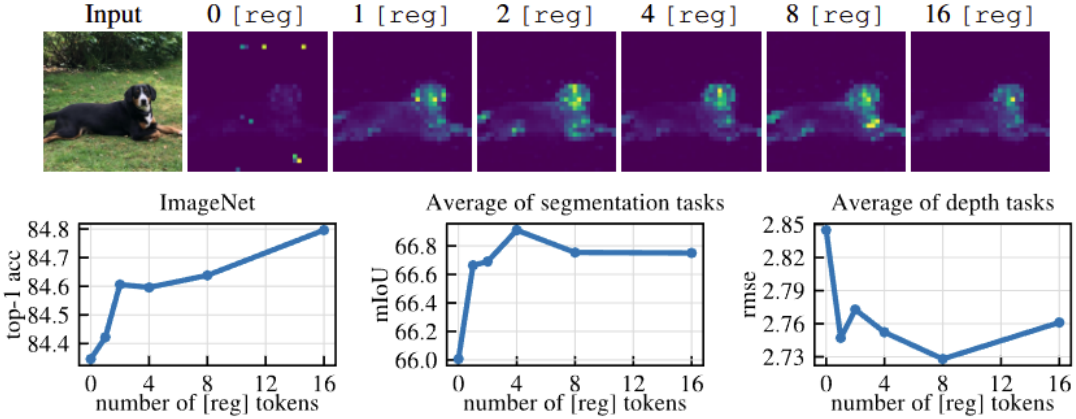


Figure 11. Register 개수에 따른 성능 변화

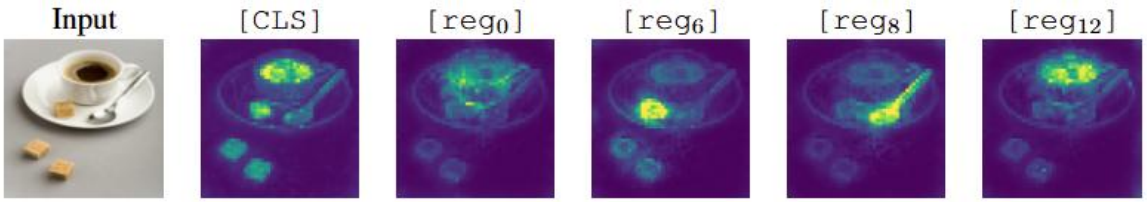


Figure 12. Register마다 다른 feature map

Experiments

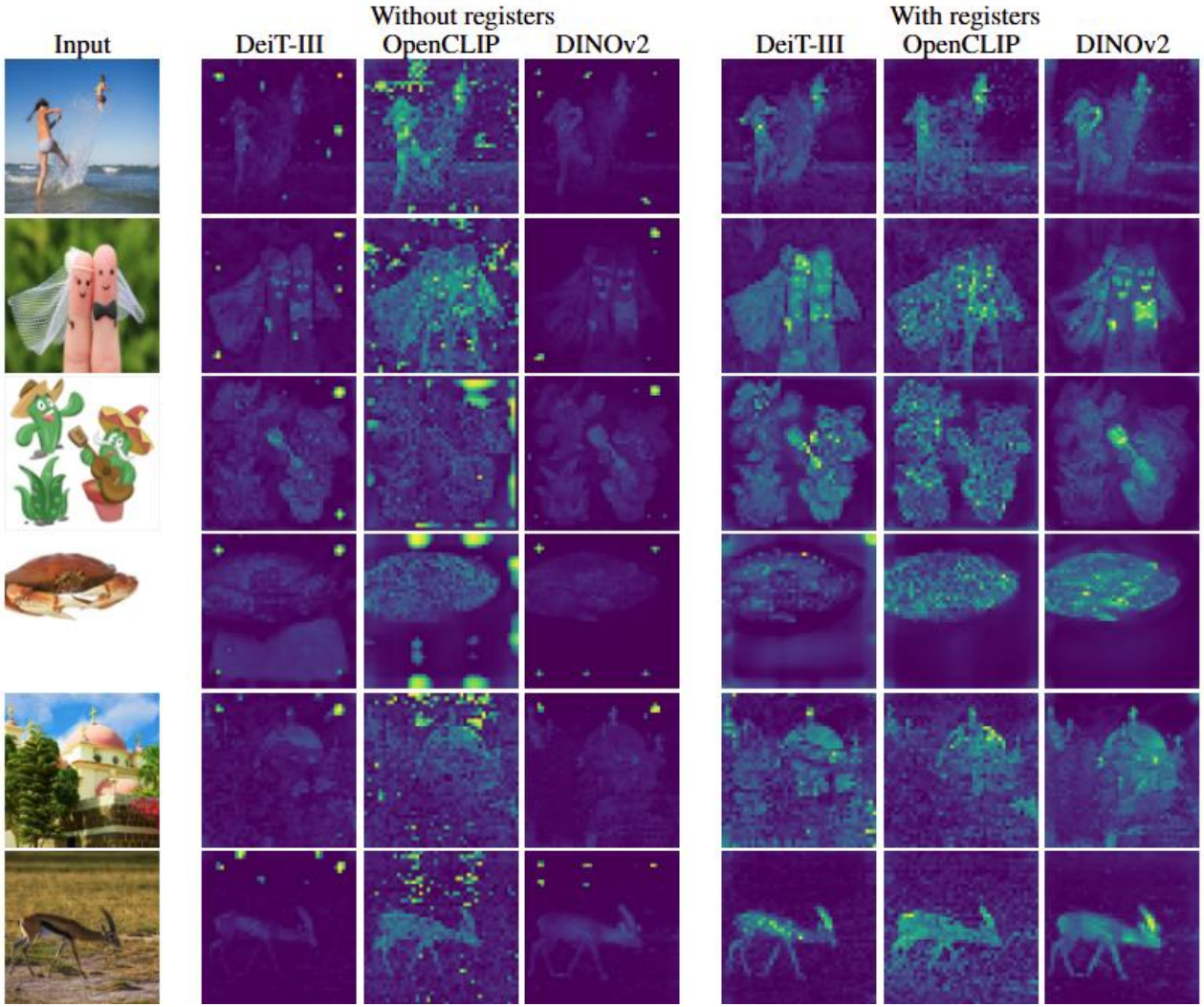


Figure 13. Register 유무에 따른 attention map 변화

Conclusion

- ViT 모델에서 나타나는 high norm spike는 모델 오류가 아닌 정보량이 적은 패치를 재활용 하는 것
  - 이를 단순히 register 토큰을 추가하는 것만으로 해결
- 하지만 최적의 register 개수를 정하는 수학적 공식이 부족
  - 이 논문의 경우 단순 실험으로 4~8개의 register가 적절하다고 제안

감사합니다