CaiT: Going deeper with Image Transformers

School of Industrial and Management Engineering, Korea University

Jong Kook, Heo

KOREA UNIVERSITY

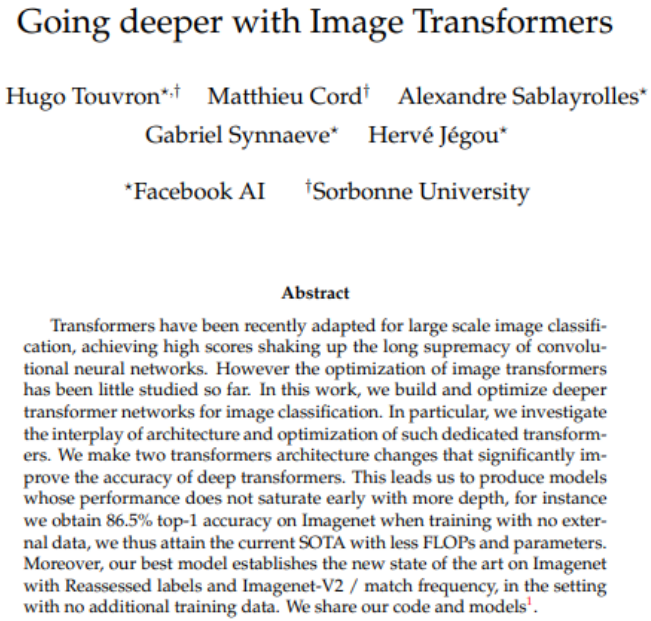
DMQA

Contents

Research Purpose

❖ CaiT: Going deeper with Image Transformers

- Facebook AI Resarch 에서 연구, 2021년 09월 14일 기준 약 47회 인용
- 기존 ViT/DeiT 에 바뀐 점은 아래 두 가지
  - ✓ Transformer block 의 Residual Connection 의 가중치를 조절하는 방법 제안(LayerScale)
  - ✓ CLS Token 에 대해서만 Attention 을 계산하는 별도의 Class Attention Block
- DINO, XCiT 저자들이 작성 (DINO->CaiT->XCiT 순서로 읽으면 좋음)



Going deeper with Image Transformers

Hugo Touvron*,† Matthieu Cord† Alexandre Sablayrolles*
Gabriel Synnaeve* Hervé Jégou*

*Facebook AI †Sorbonne University

Abstract

Transformers have been recently adapted for large scale image classification, achieving high scores shaking up the long supremacy of convolutional neural networks. However the optimization of image transformers has been little studied so far. In this work, we build and optimize deeper transformer networks for image classification. In particular, we investigate the interplay of architecture and optimization of such dedicated transformers. We make two transformers architecture changes that significantly improve the accuracy of deep transformers. This leads us to produce models whose performance does not saturate early with more depth, for instance we obtain 86.5% top-1 accuracy on Imagenet when training with no external data, we thus attain the current SOTA with less FLOPs and parameters. Moreover, our best model establishes the new state of the art on Imagenet with Reassessed labels and Imagenet-V2 / match frequency, in the setting with no additional training data. We share our code and models[1].

DMQA

Overview

CaiT

❖ Preliminaries

- Residual Connection 과 Transformer Block 의 변형에 대해 알려진 결과들
  - ✓ Layernorm 은 Residual Connection 하기 전에 해주는 게 더 좋음**(so-called "pre-norm")**
  - ✓ 여러가지 Skip-Connection 중 Identity Maping 이 가장 최적화하기 좋다.
  - ✓ Transformer 기반 모델은 learning rate warmup 이나 Layernorm 이 없으면 초기 학습이 매우 불안정

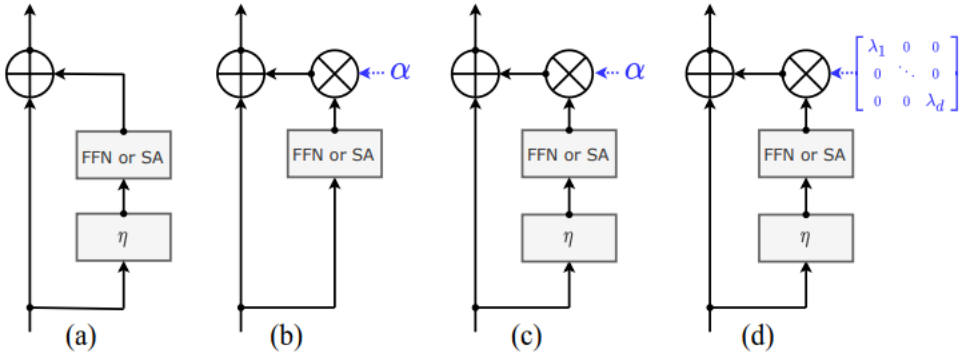
Overview

CaiT

❖ Main Contribution1 : LayerScale

- Transformer Block 에 각기 다른 Normalization Strategy 를 적용했을 때 …
  - ✓ (b) : Residual 값의 가중치를 조절하는 learnable parameter 도입, **그러나 Learning rate warmup 과 LN 의 부재로 인해 ViT 구조에서 수렴이 잘 되지않았음**
  - ✓ (c) : (b) + LN + warmup strategy
  - ✓ **(d) LayerScale : "채널마다" residual connection 의 가중치를 다르게 주자(main proposal)**
  - ✓ (d) 로 하였을 경우, Large depth 의 모델에서도 수렴을 촉진시키고 정확도를 증가시켰다고함
- 기존 ViT 는 imagenet 에서만 학습시킬 경우 깊어질수록 성능이 떨어지고, DeiT 도 12개 블록만 고려

η : *Layernorm*
SA : *Self Attention*
FFN : *Feed forward nets*
α, λ: *learnable parameters*



(a) ViT, DeiT

$$x'_l = x_l + \mathrm{SA}(\eta(x_l))$$
$$x_{l+1} = x'_l + \mathrm{FFN}(\eta(x'_l))$$

(b) Fixup, ReZero, SkipInit

$$x'_l = x_l + \alpha_l \,\mathrm{SA}(x_l)$$
$$x_{l+1} = x'_l + \alpha'_l \,\mathrm{FFN}(x'_l)$$

(d) CaiT(LayerScale)

$$x'_l = x_l + \mathrm{diag}(\lambda_{l,1}, \ldots, \lambda_{l,d}) \times \mathrm{SA}(\eta(x_l))$$
$$x_{l+1} = x'_l + \mathrm{diag}(\lambda'_{l,1}, \ldots, \lambda'_{l,d}) \times \mathrm{FFN}(\eta(x'_l))$$

DMQA

Overview

CaiT

❖ Main Contribution1 : LayerScale

- Parameter Initialization(Appendix A)
  - ✓ λ 의 초깃값은 0에 가까운 작은 값(ϵ)으로 설정하여 residual branch 의 초기 기여도가 작음
  - ✓ 하나의 값으로 가중치를 조정하는 모델(i.e. ReZero/SkipInit, Fixup) 보다 최적화에서 더 많은 다양성을 제공한다고함.
  - ✓ Zero/ Random Initialization 보다 0에 가까운 작은 값(ϵ) 으로 초기화하는게 조금 더 나은 성능을 보임

$$x'_l = x_l + \mathrm{diag}(\lambda_{l,1}, \ldots, \lambda_{l,d}) \times \mathrm{SA}(\eta(x_l))$$
$$x_{l+1} = x'_l + \mathrm{diag}(\lambda'_{l,1}, \ldots, \lambda'_{l,d}) \times \mathrm{FFN}(\eta(x'_l)),$$

Table A.1: Performance when increasing the depth. We compare different strategies and report the top-1 accuracy (%) on ImageNet-1k for the DeiT training (Baseline) with and without adapting the stochastic depth rate d_r (uniform drop-rate), and a modified version of Rezero with LayerNorm and warmup. We compare different initialisation of the diagonal matrix for LayerScale. We also report results with 0 initialization, Uniform initialisation and small constant initialisation. Except for the baseline $d_r = 0.1$, we have adapted the stochastic depth rate d_r.

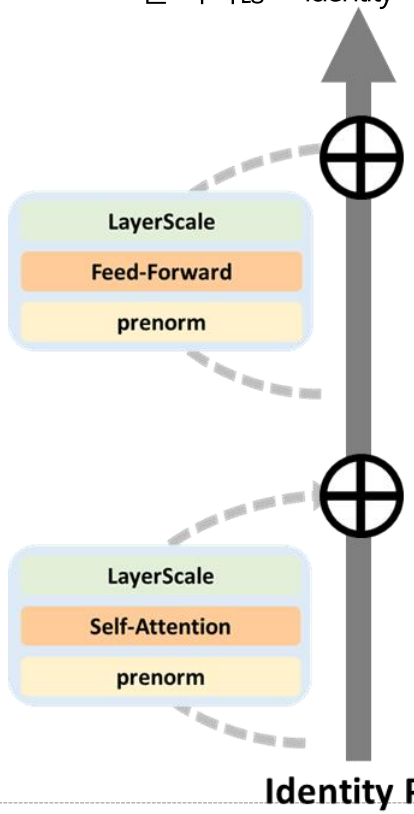
depth	baseline $d_r = 0.1$	baseline $[d_r]$	ReZero $\alpha = 0$	LayerScale [ε] $\lambda_i = 0$	$\lambda_i = \mathcal{U}[0, 2\varepsilon]$	$\lambda_i = \varepsilon$
12	79.9	79.9 [*0.05*]	78.3	79.7	80.2 [*0.1*]	80.5 [*0.1*]
18	80.1	80.7 [*0.10*]	80.1	81.5	80.8 [*0.1*]	81.7 [*0.1*]
24	78.9†	81.0 [*0.20*]	80.8	82.1	82.1 [10^{-5}]	82.4 [10^{-5}]
36	78.9†	81.9 [*0.25*]	81.6	82.7	82.6 [10^{-6}]	82.9 [10^{-6}]

Overview

CaiT

❖ Main Contribution1 : LayerScale

- Implementation Details
  - ✓ Residual Connection(SA/FFN) 에 Stochastic Depth 를 주어 보다 더 깊은 망을 학습시킬 수 있었다고 함
  - ✓ Stochastic Depth : **일정 확률로 특정 layer 자체의 연산을 끊어버림(per-layer dropout)**, 구현 상에서는 FFN/SA 의 연산을 하지않고 Identity 로 넘김



논문 수식에서는 λ라 표기된 diagonal matrix

```
self.gamma_1 = nn.Parameter(init_values * torch.ones((dim)),requires_grad=True)
self.gamma_2 = nn.Parameter(init_values * torch.ones((dim)),requires_grad=True)
```

$X \in R^{b*n*d}$ $\gamma \in R^{d}$

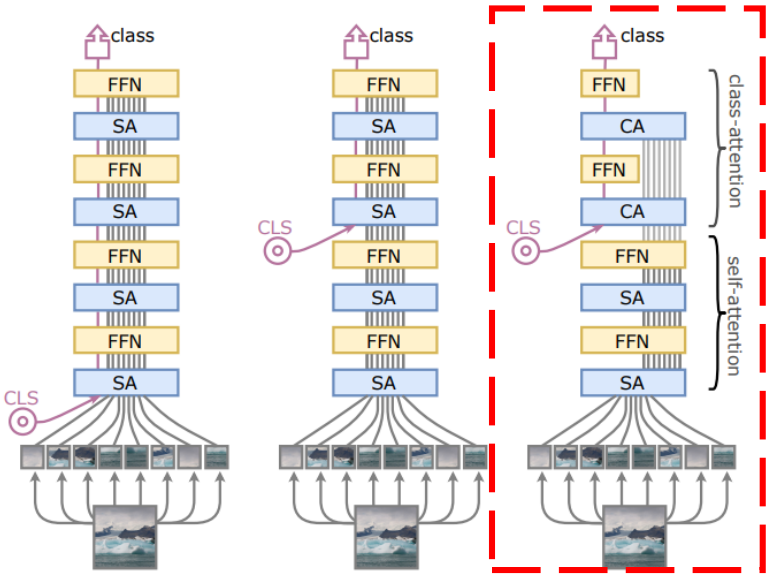
```
x = x + self.drop_path(self.gamma_1 * self.attn(self.norm1(x)))
x = x + self.drop_path(self.gamma_2 * self.mlp(self.norm2(x)))
```

Overview

CaiT

❖ Main Contribution2 : Class Attention

- Later CLS Token
  - ✓ 기존 ViT 에서는 CLS Token을 맨 밑에부터 넣어줌(left)
  - ✓ 이 경우 CLS Token 이 두 개의 목적(1. guiding self-attention between patches, 2. summarizing information useful to the linear classifier) 을 동시에 수행해야하기 때문에 해롭다고 주장
  - ✓ CLS Token 을 중간에 삽입한 결과(middle) 성능이 향상됨을 증명
  - ✓ **CaiT 는 2개의 목적을 따로 수행시키기 위해 Self-Attention(SA) 과 Class Attention(CA) 를 따로 두었음(right)**



DMQA

Overview

CaiT

❖ Main Contribution2 : Class Attention

- Class Attention
  - ✓ Key, Value : 기존 ViT Self-Attention 과 동일하게 CLS Token 과 모든 patch 에 대해 구함
  - ✓ **Query : CLS Token 에 대해서만 구함. 따라서 residual connection 및 업데이트도 CLS Token 에 대해서만 진행!!**

$$Q = W_q \boldsymbol{x_{cls}} \in R^{b*h*1*d}$$
$$K = W_k \boldsymbol{z} \in R^{b*h*p*d}$$
$$V = W_k \boldsymbol{z} \in R^{b*h*p*d}$$

$$where\ z = [x_{cls}; x_{patches}] = [x_{cls}; x_0, \dots, x_n]$$

Total p patches

$$Attention = Softmax(QK^T/\sqrt{d/h}) \in R^{b*h*\mathbf{1}*p}$$

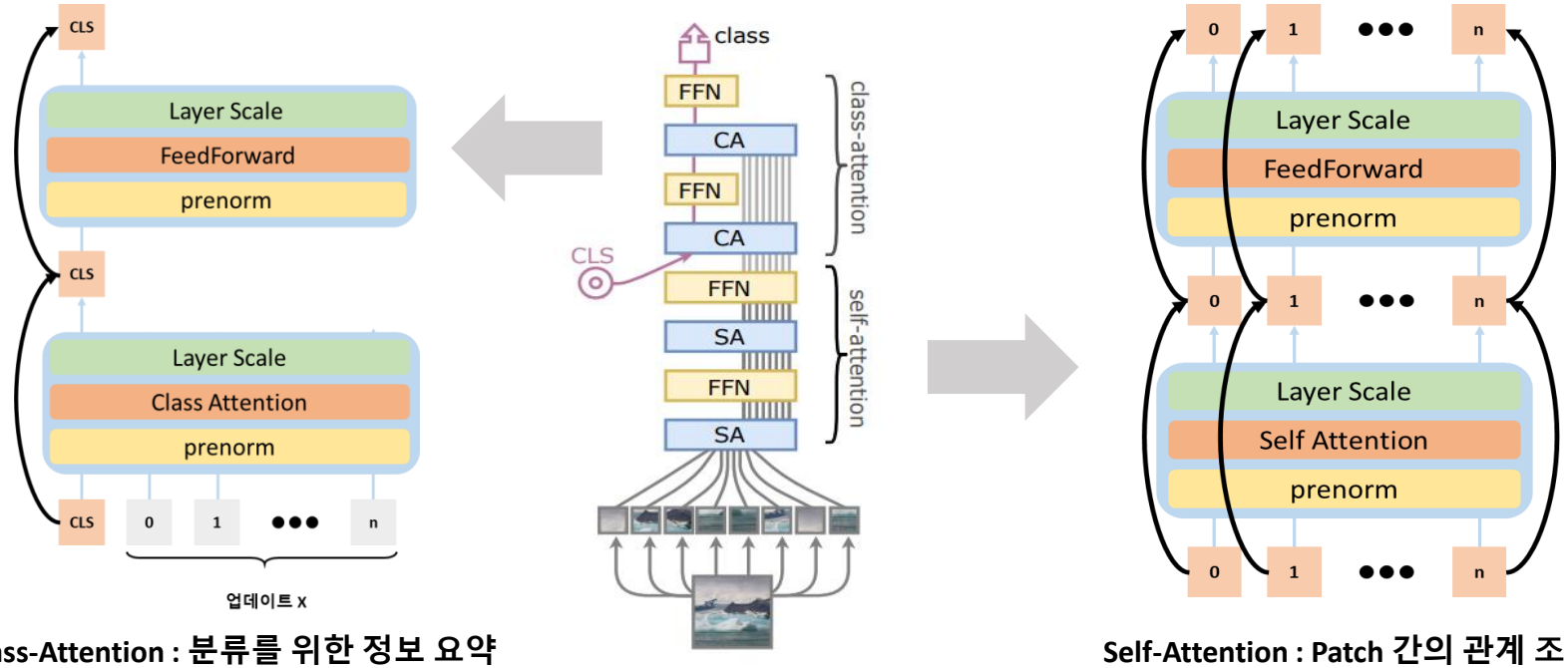
$$Output = W_o\ Attn\ *V$$

Overview

CaiT

❖ Main Contribution2 : Class Attention

- Self-Attention vs Class Attention
  - ✓ SA 모듈을 n개쌓은 후 마지막 CA모듈을 1 or 2개 쌓음(i.e. 9+3, 10+2, 12 +2..)
  - ✓ 실제 구현체의 SA 블록에서는 기존 Self-Attention 대신 변형된 "Talking-Head Attention" 을 사용
    - 논문에 딱 한 줄 언급 되어있고, 기존 SA 대비 구조상/ 성능상 큰 차이는 없음(인용횟수도 2회밖에 안됨)->이하 생략
  - ✓ **파란 상자는 drop path, 검은 실선은 residual connection 을 의미**



Class-Attention : 분류를 위한 정보 요약

Self-Attention : Patch 간의 관계 조명

DMQA

Overview

CaiT

❖ Main Contribution2 : Class Attention

- What about not including CLS in "keys" of Class Attention?(Appendix B)
  - ✓ Class Attention 의 key(z) 에서 cls token 유무의 차이에 따른 ImageNet Top1 Acc 차이는 83.44% vs 83.31% 로 매우 작음 (0.13%). 따라서 안넣어도 크게 문제되지는 않는다.

$$z = [x_{cls}; x_0, \dots, x_n] \ \textbf{vs} \ z = [x_0, \dots, x_n]??$$ It doesn't matter!

- What about not Removing "LayerScale" in Class Attention??(Appendix B)
  - ✓ Class Attention 모듈에서 제안방법론1 인 LayerScale 을 제거하면 ImageNet Top1 Acc 가 83.36% 로 0.08% 밖에 차이 안 남

- How about Computational Complexity??
  - ✓ Class Attention 은 CLS Token 에 대한 Q 만 계산하기 때문에 패치 개수 증가에 따른 계산복잡도 증가가 Quadratic 하지 않고 Linear 함

Experiments

CaiT

❖ 깊이에 따른 Droppath Ratio 는 어떻게 해야할까?

- Adjusting the drop-rate of stochastic depth(Table 1)
  - ✓ 기존 DeiT 에서는 Droppath 의 raio 를 0.05로 할 경우, 18 depth 까지는 성능이 상승하지만 그 이상의 깊이에서는 학습이 불안정해짐.
  - ✓ 깊은 구조일수록 Droppath ratio를 높게하면 36 depth 까지도 안정적으로 수렴하고 성능이 향상됨
  - ✓ 저자 왈, 48 depth 에서는 80.7%로 감소한다고함

DeiT

depth	baseline		scalar α weighting				LayerScale
	$d_r = 0.05$	adjust $[d_r]$	Rezero	T-Fixup	Fixup	$\alpha = \varepsilon$	
12	79.9	79.9 [*0.05*]	78.3	79.4	80.7	80.4	80.5
18	80.1	80.7 [*0.10*]	80.1	81.7	82.0	81.6	81.7
24	78.9†	81.0 [*0.20*]	80.8	81.5	82.3	81.1	82.4
36	78.9†	81.9 [*0.25*]	81.6	82.1	82.4	81.6	82.9

Experiments

CaiT

❖ LayerScale 과 기존 방법론의 비교 실험

- Comparison of Normalization strategies(Table 1)
  - ✓ Rezero, T-Fixup etc 는 Residual Connection Weighting, Layernorm 유무, Warmup strategy 유무에 따른 기존의 방법론들.
  - ✓ LayerScale : Layernorm + Warm up + channel-wise residual weighting 전략(제안방법론1)

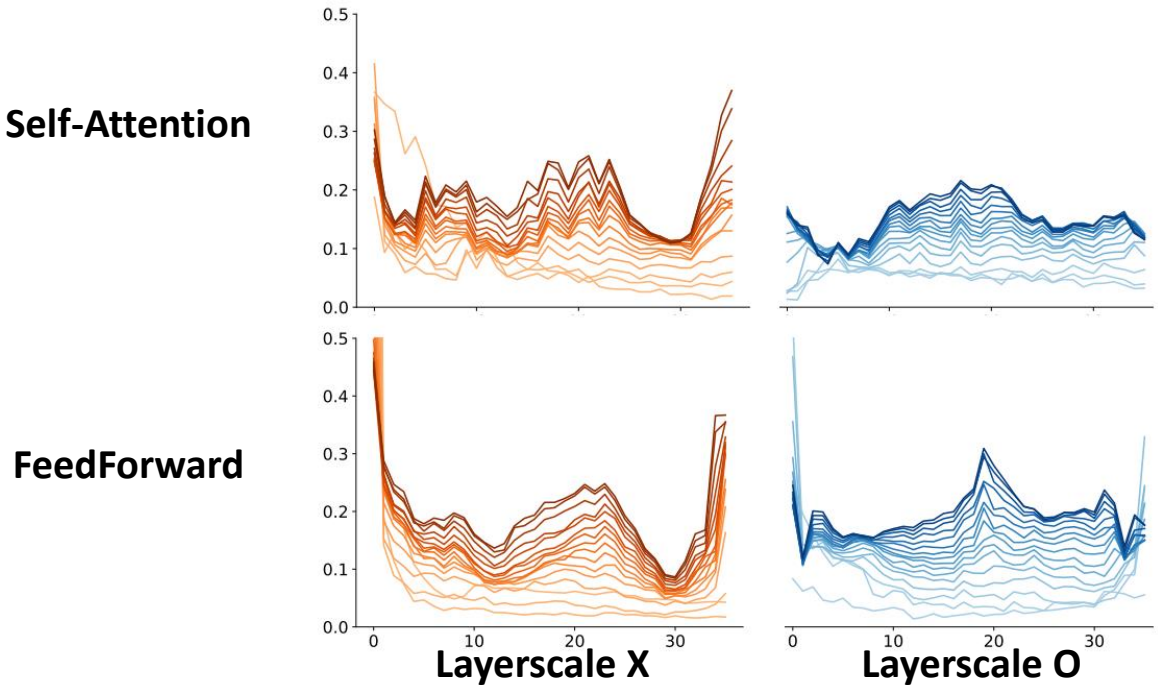
depth	baseline		scalar α weighting				LayerScale
	$d_r = 0.05$	adjust $[d_r]$	Rezero	T-Fixup	Fixup	$\alpha = \varepsilon$	
12	79.9	79.9 [*0.05*]	78.3	79.4	80.7	80.4	80.5
18	80.1	80.7 [*0.10*]	80.1	81.7	82.0	81.6	81.7
24	78.9†	81.0 [*0.20*]	80.8	81.5	82.3	81.1	82.4
36	78.9†	81.9 [*0.25*]	81.6	82.1	82.4	81.6	82.9

Experiments

CaiT

❖ Analysis of Layerscale

- Statistics of branch weighting
  - ✓ 36 depth 구조에서 main branch(Identity mapping) 와 residual connection(attn./mlp block) 의 활성도 크기를 비율로 비교
  - ✓ 가장 어두운게 가장 마지막 epoch
  - ✓ Layerscale 이 들어가지 않았을 때의 기여도 변동이 큼. Layerscale 을 썼을 때의 residual 의 비율은 평균적으로 약 20%로, 이러한 균일성이 최적화에 도움되었을 것이라 주장



DMQA

Experiments

CaiT

❖ Class Attention Layers

- Late Insertion and FLOPS(Table2)
  - ✓ DeiT -S(12 layers) 기준 CLS Token 은 늦게 넣을 수록 성능이 올라감. 최대치는 2개 layer 만 남겨놨을때**(빨간색)**
  - ✓ DeiT-S with avg pooling 보다 CaiT(10+2)가 FLOPs가 적고 성능은 동일 혹은 조금 더 좋음**(파란색)**

depth: SA+CA	insertion layer	top-1 acc.	#params	FLOPs
Baselines: DeiT-S and average pooling				
12: 12 + 0	0	79.9	22M	4.6B
12: 12 + 0	n/a	80.3	22M	4.6B
Late insertion of class embedding				
12: 12 + 0	2	80.0	22M	4.6B
12: 12 + 0	4	80.0	22M	4.6B
12: 12 + 0	8	80.0	22M	4.6B
12: 12 + 0	10	80.5	22M	4.6B
12: 12 + 0	11	80.3	22M	4.6B
DeiT-S with class-attention stage (SA+FFN)				
12: 9 + 3	9	79.6	22M	3.6B
12: 10 + 2	10	80.3	22M	4.0B
12: 11 + 1	11	80.6	22M	4.3B
13: 12 + 1	12	80.8	24M	4.7B
14: 12 + 2	12	80.8	26M	4.7B
15: 12 + 3	12	80.6	27M	4.8B

Experiments

CaiT

❖ Model size & Finetuning Resolution

- 모든 모델들은 224 size 의 이미지로 훈련시킴
- Finetuning 시 384 size 로 하는게 FLOPs 는 더 커지지만 성능이 더 올라감
- Hard Distillation 했을 때의 성능도 보여줌

CAIT model	depth (SA+CA)	d	#params ($\times 10^6$)	FLOPs ($\times 10^9$) @224	FLOPs ($\times 10^9$) @384	Top-1 acc. (%): Imagenet1k-val @224	↑384	@224↑	↑384↑
XXS-24	24+2	192	12.0	2.5	9.5	77.6	80.4	78.4	80.9
XXS-36	36+2	192	17.3	3.8	14.2	79.1	81.8	79.7	82.2
XS-24	24+2	288	26.6	5.4	19.3	81.8	83.8	82.0	84.1
XS-36	36+2	288	38.6	8.1	28.8	82.6	84.3	82.9	84.8
S-24	24+2	384	46.9	9.4	32.2	82.7	84.3	83.5	85.1
S-36	36+2	384	68.2	13.9	48.0	83.3	85.0	84.0	85.4
S-48	48+2	384	89.5	18.6	63.8	83.5	85.1	83.9	85.3
M-24	24+2	768	185.9	36.0	116.1	83.4	84.5	84.7	85.8
M-36	36+2	768	270.9	53.7	173.3	83.8	84.9	85.1	86.1

Experiments

CaiT

❖ Ablations

Table 8: Ablation: we present the ablation path from DeiT-S to our CaiT models. We highlight the complementarity of our approaches and optimized hyper-parameters. Note, Fine-tuning at higher resolution supersedes the inference at higher resolution. See Table 1 for adapting stochastic depth before adding LayerScale. †: training failed.

Improvement	top-1 acc.	#params	FLOPs
DeiT-S [*d=384,300 epochs*]	79.9	22M	4.6B
+ More heads [*8*]	80.0	22M	4.6B
+ Talking-heads	80.5	22M	4.6B
+ Depth [*36 blocks*]	69.9†	64M	13.8B
+ Layer-scale [*init* $\varepsilon = 10^{-6}$]	80.5	64M	13.8B
+ Stch depth. adaptation [d_r*=0.2*]	83.0	64M	13.8B
+ CaiT architecture [*specialized class-attention layers*]	83.2	68M	13.9B
+ Longer training [*400 epochs*]	83.4	68M	13.9B
+ Inference at higher resolution [*256*]	83.8	68M	18.6B
+ Fine-tuning at higher resolution [*384*]	84.8	68M	48.0B
+ Hard distillation [*teacher: RegNetY-16GF*]	85.2	68M	48.0B
+ Adjust crop ratio [*0.875* $\rightarrow$ *1.0*]	85.4	68M	48.0B

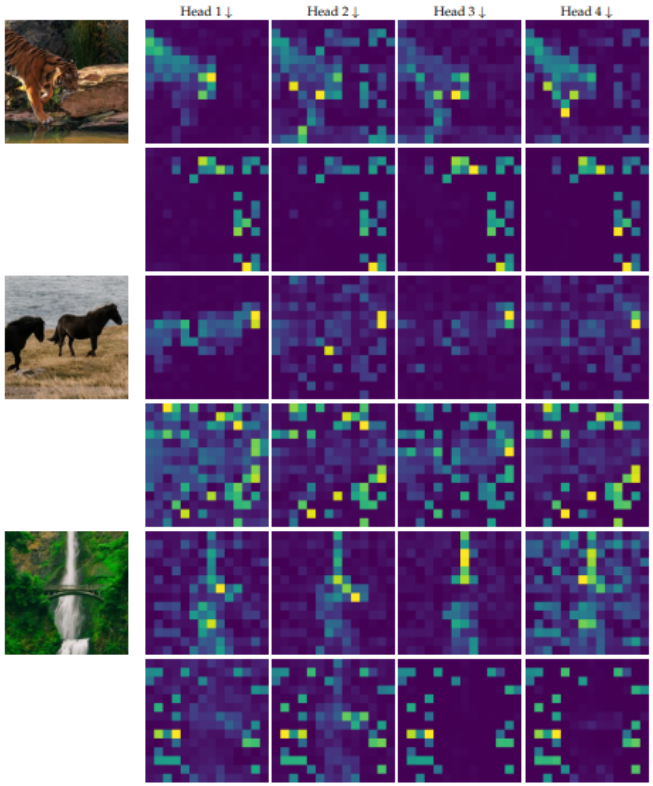
256 으로 resized 된 이미지에서 224 center crop 하는 것이 0.875 ratio인데,
이 비율을 1로 하는 것이 0.2% 성능향상을 가져왔다고 함

Experiments

CaiT

❖ Visualization

- Attention map
  - ✓ 첫번째 Class Attention Layer 에서는 관심있는 객체를 위주로 포착
  - ✓ 두번째 Class Attention Layer 에서는 이미지를 global 하게 보거나 좀 더 context 에 집중한다고함(?)



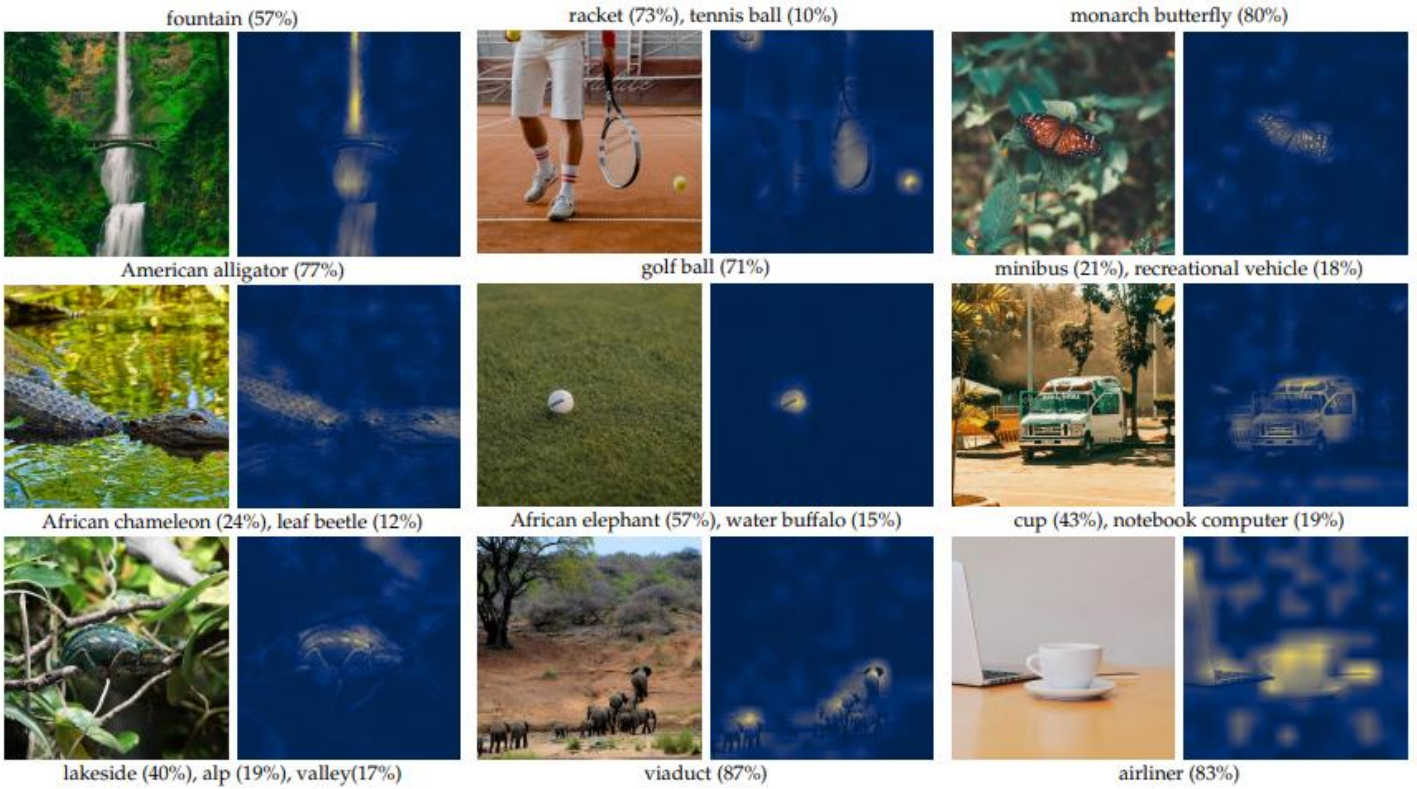
1st row: 1st Class Attention Map per head
2nd row : 2nd Class Attention Map per head

Experiments

CaiT

❖ Visualization

- Saliency Map
  - ✓ 첫번째 Class Attention Layer Map 에서 나온 값을 토대로 원본 이미지에 saliency mapping



Conclusion

CaiT

- Attention Block 내에서 Residual Connection weighting, Normalization, Warmup 에 따라 다양한 실험을 진행

- CLS Token 의 정보 요약과 패치 간의 Self-Attention 역할을 나누기 위해 Class Attention 을 따로 도입한 것이 인상적

- 개인적으로 "2번째 Class Attention Map 은 주변 배경이나 문맥 정보를 살펴본다"고 주장하는데 잘 모르겠음(feat 18p. 말에 대한 2nd CLS Attention Map)