CMT:
Convolutional Neural Networks Meet Vision Transformers

이찬호
국방인공지능융합연구소

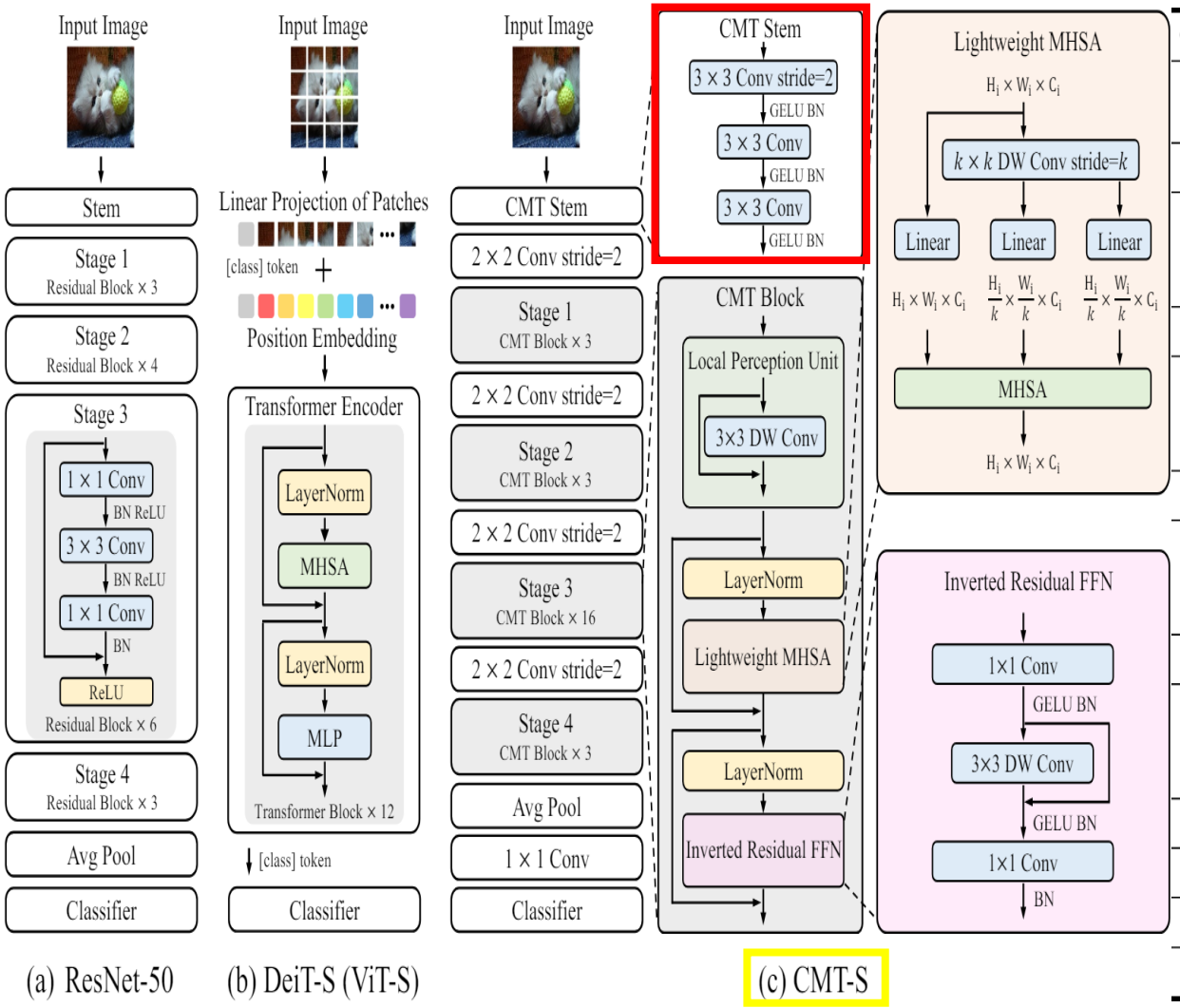
CMT - Introduction

- 이 당시(2021), ViT와 같이 이러한 Transformer 구조를 Vision 영역에 도입하고자 하는 시도가 증가하였으며, Detection, Segmentation, Classification 등 꽤나 유의미한 성능을 보이며, 때로는 CNN을 능가함

- 그럼에도 불구하고, 동일한 크기의 Small CNN에 비해 ViT는 확연히 성능이 떨어지는 모습을 보이는데, 본 논문의 저자들은 그 이유를 3가지로 정의
  1. ViT의 경우 단순히 이미지를 패치 형태로 나누고 1D Sequence 형태로 입력을 진행(이 과정에서 이미지의 2D Locality 손실)하지만, Transformer 이점 덕분에 Patch들 간 Long Range Dependency를 효과적으로 찾아내지만, 이미지의 2D Locality와 같은 근본적인 차이점을 고려하지 않음
  2. ViT의 경우 항상 고정된 Patch size를 사용하여 Multi Scale Feature Map을 추출할 수 없으며, 특히 Low Resolution Feature Map에 취약한 모습을 보임
  3. ViT는 이미지를 패치로 잘라 입력하는 방식을 취하게 되고, 이미지 크기에 대해 Quadratic한 연산량이 필요로 함

- 본 논문에서는 이러한 CNN과 ViT를 적절히 결합한 Architecture인 CMT를 제안

CMT - Architecture



Output Size	Layer Name	CMT-Ti	CMT-XS	CMT-S	CMT-B
112 × 112	Stem	3 × 3, 16, stride 2 [3 × 3, 16] × 2	3 × 3, 16, stride 2 [3 × 3, 16] × 2	3 × 3, 32, stride 2 [3 × 3, 32] × 2	3 × 3, 38, stride 2 [3 × 3, 38] × 2
56 × 56	Patch Embedding	2 × 2, 46, stride 2	2 × 2, 52, stride 2	2 × 2, 64, stride 2	2 × 2, 76, stride 2
Stage 1	LPU LMHSA IRFFN	$\begin{bmatrix} 3\times3, 46 \\ H_1{=}1, k_1{=}8 \\ R_1{=}3.6 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 52 \\ H_1{=}1, k_1{=}8 \\ R_1{=}3.8 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times3, 64 \\ H_1{=}1, k_1{=}8 \\ R_1{=}4 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times3, 76 \\ H_1{=}1, k_1{=}8 \\ R_1{=}4 \end{bmatrix} \times 4$
28 × 28	Patch Embedding	2 × 2, 92, stride 2	2 × 2, 104, stride 2	2 × 2, 128, stride 2	2 × 2, 152, stride 2
Stage 2	LPU LMHSA IRFFN	$\begin{bmatrix} 3\times3, 92 \\ H_2{=}2, k_2{=}4 \\ R_2{=}3.6 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 104 \\ H_2{=}2, k_2{=}4 \\ R_2{=}3.8 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times3, 128 \\ H_2{=}2, k_2{=}4 \\ R_2{=}4 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times3, 152 \\ H_2{=}2, k_2{=}4 \\ R_2{=}4 \end{bmatrix} \times 4$
14 × 14	Patch Embedding	2 × 2, 184, stride 2	2 × 2, 208, stride 2	2 × 2, 256, stride 2	2 × 2, 304, stride 2
Stage 3	LPU LMHSA IRFFN	$\begin{bmatrix} 3\times3, 184 \\ H_3{=}4, k_3{=}2 \\ R_3{=}3.6 \end{bmatrix} \times 10$	$\begin{bmatrix} 3\times3, 208 \\ H_3{=}4, k_3{=}2 \\ R_3{=}3.8 \end{bmatrix} \times 12$	$\begin{bmatrix} 3\times3, 256 \\ H_3{=}4, k_3{=}2 \\ R_3{=}4 \end{bmatrix} \times 16$	$\begin{bmatrix} 3\times3, 304 \\ H_3{=}4, k_3{=}2 \\ R_3{=}4 \end{bmatrix} \times 20$
7 × 7	Patch Embedding	2 × 2, 368, stride 2	2 × 2, 416, stride 2	2 × 2, 512, stride 2	2 × 2, 608, stride 2
Stage 4	LPU LMHSA IRFFN	$\begin{bmatrix} 3\times3, 368 \\ H_4{=}8, k_4{=}1 \\ R_4{=}3.6 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 416 \\ H_4{=}8, k_4{=}1 \\ R_4{=}3.8 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times3, 512 \\ H_4{=}8, k_4{=}1 \\ R_4{=}4 \end{bmatrix} \times 3$	$\begin{bmatrix} 3\times3, 608 \\ H_4{=}8, k_4{=}1 \\ R_4{=}4 \end{bmatrix} \times 4$
1 × 1	Projection	1 × 1, 1280			
1 × 1	Classifier	Fully Connected Layer, 1000			
# Params		9.49 M	15.24 M	25.14 M	45.72 M
# FLOPs		0.64 B	1.54 B	4.04 B	9.33 B

FIG 1. CMT Architecture

CMT - Local Perception Unit

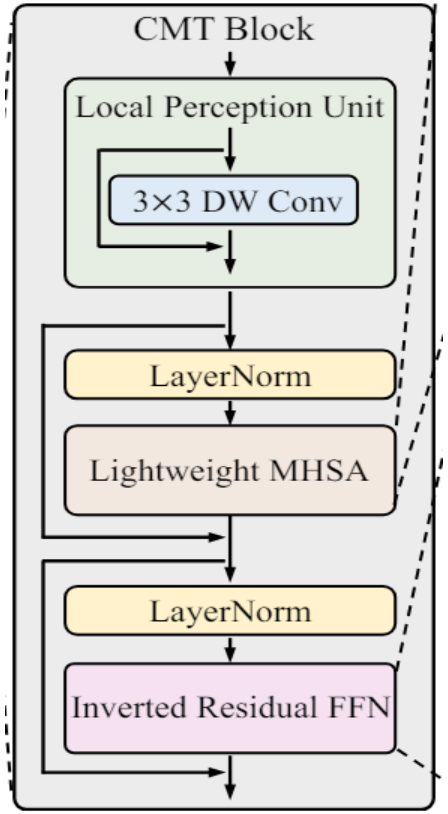


FIG 2. Local Perception Unit

- LPU는 이미지를 작은 타일로 나눠 각 타일에서 중요한 패턴을 찾은 다음 원래 이미지 정보와 결합하는 방식
- Depth Wise Convolution은 각 타일에 별도의 필터를 적용하여 특정 패턴을 찾는 과정
- Residual 연결은 필터가 찾은 정보와 원본 정보를 결합하여 중요한 정보를 보존하는 방식

CMT - Lightweight MHSA

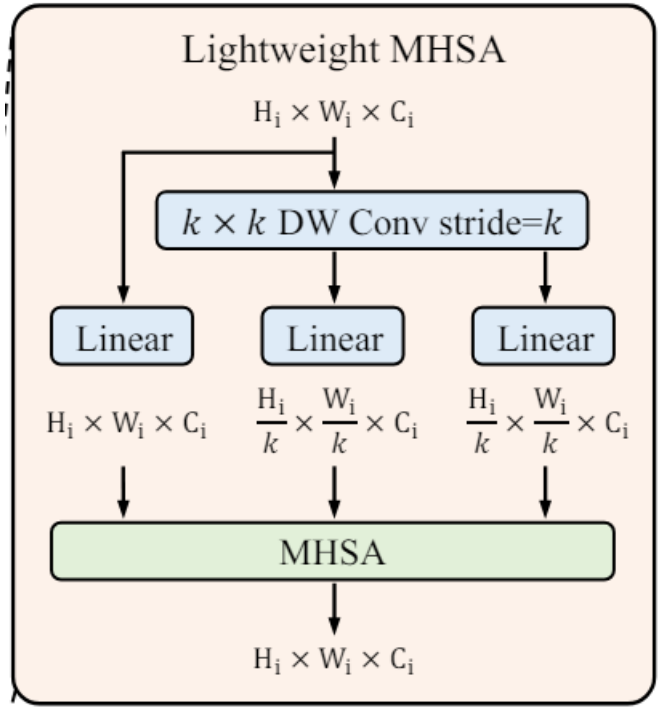


FIG 3. Lightweight MHSA

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}.$$

FIG 4. 기존 Attention

$$\text{LightweightAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}'^T}{\sqrt{d_k}} + \mathbf{B})\mathbf{V}'.$$

FIG 5. LightweightAttention과 Relative Position Bias

- 기존 Self-Attention은 Linear하게 Transform하여 Query, Key, Value를 각각 쪼개서 처리하여 매우 많은 연산량 요구
- 본 논문의 저자들은 K X K 크기의 Depth Wise Conv를 Stride K와 함께 진행하여 Query, Key 크기를 감소시켜 연산량 줄임
- 또한, Swin Transformer에서 사용되었던 Relative Position Bias 개념을 채택하였으며, 저자들은 임의로 초기화 한 후 학습 가능한 방식으로 학습 진행

CMT - Inverted Residual FFN

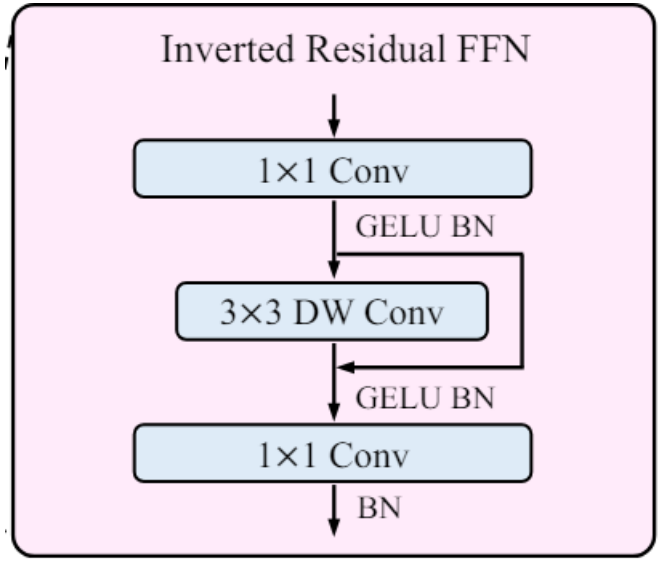


FIG 6. Inverted Residual Feed-Forward Network

$$\mathrm{FFN}(\mathbf{X}) = \mathrm{GELU}(\mathbf{X}W_1 + b_1)W_2 + b_2.$$

FIG 7. 기존 Feed-Forward Network

$$\mathrm{IRFFN}(\mathbf{X}) = \mathrm{Conv}(\mathcal{F}(\mathrm{Conv}(\mathbf{X}))),$$
$$\mathcal{F}(\mathbf{X}) = \mathrm{DWConv}(\mathbf{X}) + \mathbf{X}.$$

FIG 8. Inverted Residual Feed-Forward Network

- 기존 ViT 경우 FFN으로 2개의 Linear Layer를 사용하였으며, 또한, 첫번째 Layer는 Expansion인 4만큼 차원을 증가 시켜주고, 두번째 Layer를 거치며 다시 원래의 차원으로 돌아오는 방식
- 저자들은 기존 Bottleneck 구조인 Wide-Narrow-Wide 방식이 아닌 MobileNetV2에서 사용됐던 Norrow-Wide-Narrow 방식 사용하는데 이는 이미 Bottleneck에 충분히 필요한 데이터가 모두 있다고 가정하고 Input과 Output의 차원을 줄여서 메모리 사용량 개선
- 깊은 Layer 구조에서 효과적인 학습을 위해 ResNet 방식의 Shortcut을 더해, 정보를 보충하는 방식 적용

Result - Classification(ImageNet)

Table 2. **ImageNet Results of CMT**. CNNs and transformers with similar accuracy are grouped together for comparison. The proposed CMTs consistently outperform other methods with less computational cost.

Model	Top-1 Acc.	Top-5 Acc.	Throughput	# Params	Resolution	# FLOPs	Ratio
CPVT-Ti-GAP [6]	74.9%	-	-	6M	224^2	1.3B	2.6×
DenseNet-169 [22]	76.2%	93.2%	-	14M	224^2	3.5B	7×
EfficientNet-B1 [53]	79.1%	94.4%	-	7.8M	240^2	0.7B	1.2×
CMT-Ti	**79.1%**	**94.5%**	1323.5	9.5M	160^2	**0.6B**	**1**×
ResNet-50 [16]	76.2%	92.9%	-	25.6M	224^2	4.1B	2.7×
CoaT-Lite Mini [65]	78.9%	-	-	11M	224^2	2.0B	1.3×
DeiT-S [57]	79.8%	-	940.4	22M	224^2	4.6B	3.1×
EfficientNet-B3 [53]	81.6%	95.7%	732.1	12M	300^2	1.8B	1.2×
CMT-XS	**81.8%**	**95.8%**	857.4	15.2M	192^2	**1.5B**	**1**×
ResNeXt-101-64x4d [64]	80.9%	95.6%	-	84M	224^2	32B	8×
T2T-ViT-19 [68]	81.2%	-	-	39.0M	224^2	8.0B	2×
PVT-M [60]	81.2%	-	528.1	44.2M	224^2	6.7B	1.7×
Swin-T [36]	81.3%	-	755.2	29M	224^2	4.5B	1.1×
CPVT-S-GAP [6]	81.5%	-	-	23M	224^2	4.6B	1.2×
RegNetY-8GF [44]	81.7%	-	591.6	39.2M	224^2	8.0B	2×
CeiT-S [67]	82.0%	95.9%	-	24.2M	224^2	4.5B	1.1×
EfficientNet-B4 [53]	82.9%	96.4%	349.4	19M	380^2	4.2B	1×
Twins-SVT-B [5]	83.1%	-	-	56.0M	224^2	8.3B	2.1×
CMT-S	**83.5%**	**96.6%**	562.5	25.1M	224^2	**4.0B**	**1**×
ViT-B/16$_{\uparrow 384}$ [10]	77.9%	-	85.9	85.8M	384^2	77.9B	8.4×
TNT-B [14]	82.8%	96.3%	-	65.6M	224^2	14.1B	1.5×
DeiT-B$_{\uparrow 384}$ [57]	83.1%	-	85.9	85.8M	384^2	55.6B	6.0×
CvT-21$_{\uparrow 384}$ [63]	83.3%	-	-	31.5M	384^2	24.9B	2.7×
Swin-B [36]	83.3%	-	278.1	88M	224^2	15.4B	1.5×
Twins-SVT-L [5]	83.3%	-	288.0	99.2M	224^2	14.8B	1.7×
CeiT-S$_{\uparrow 384}$ [67]	83.3%	96.5%	-	24.2M	384^2	12.9B	1.4×
BoTNet-S1-128 [48]	83.5%	96.5%	-	75.1M	256^2	19.3B	2.1×
EfficientNetV2-S [54]	83.9%	-	-	22M	224^2	8.8B	1×
EfficientNet-B6 [53]	84.0%	96.8%	96.9	43M	528^2	19.2B	2.0×
CMT-B	**84.5%**	**96.9%**	285.4	45.7M	256^2	**9.3B**	**1**×
EfficientNet-B7 [53]	84.3%	97.0%	55.1	66M	600^2	37B	1.9×
CMT-L	**84.8%**	**97.1%**	150.4	74.7M	288^2	**19.5B**	**1**×

FIG 9. ImageNet Result of CMT

Result - Transfer Learning

Table 6: **Transfer Learning Results of CMT**. All results are fine-tuned with the ImageNet pretrained checkpoint. † means the transfer results are from [25].

Model	# Params	# FLOPs	CIFAR10	CIFAR100	Cars	Flowers	Pets
ResNet-152† [15]	58.1M	11.3B	97.9%	87.6%	92.0%	97.4%	94.5%
Inception-v4† [47]	41.1M	16.1B	97.9%	87.5%	93.3%	98.5%	93.7%
EfficientNet-B7$_{\uparrow 600}$ [50]	64.0M	37.2B	98.9%	**91.7%**	**94.7%**	98.8%	**95.4%**
ViT-B/16$_{\uparrow 384}$ [10]	85.8M	17.6B	98.1%	87.1%	-	89.5%	93.8%
DeiT-B [51]	85.8M	17.6B	99.1%	90.8%	92.1%	98.4%	-
CeiT-S$_{\uparrow 384}$ [61]	24.2M	12.9B	99.1%	90.8%	94.1%	98.6%	94.9%
TNT-S$_{\uparrow 384}$ [13]	**23.8M**	17.3B	98.7%	90.1%	-	**98.8%**	94.7%
CMT-S (ours)	25.1M	**4.04B**	**99.2%**	**91.7%**	94.4%	98.7%	95.2%

FIG 10. Transfer Learning Result of CMT

Result - Object detection

Table 7: **Object detection results on COCO val2017.** All models use RetinaNet as basic framework and are trained in "1x" schedule. FLOPs are calculated on 1280×800 input. † means the results are from [5].

Backbone	# Params	# FLOPs	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ConT-M [60]	217B	27.0M	37.9	58.1	40.2	23.0	40.6	50.4
ResNet-101 [15]	315B	56.7M	38.5	57.6	41.0	21.7	42.8	50.4
RelationNet++ [4]	266B	39.0M	39.4	58.2	42.5	-	-	-
ResNeXt-101-32x4d [58]	319B	56.4M	39.9	59.6	42.7	22.3	44.2	52.5
PVT-S [54]	226B	34.2M	40.4	61.3	43.0	25.0	42.9	55.7
Swin-T† [32]	245B	38.5M	41.5	62.1	44.2	25.1	44.9	55.5
Twins-SVT-S [5]	209B	34.3M	42.3	63.4	45.2	26.0	45.5	56.5
Twins-PCPVT-S [5]	226B	34.4M	43.0	64.1	46.0	27.5	46.3	57.3
CMT-S (ours)	231B	44.3M	**44.3**	**65.5**	**47.5**	**27.1**	**48.3**	**59.1**

FIG 11. Object detection results on COCO val2017.

Result - Instance segmentation

Table 8: **Instance segmentation results on COCO val2017**. All models use Mask R-CNN as basic framework and are trained in "1x" schedule. FLOPs are calculated on 1280×800 input. † means the results are from [5].

Backbone	# Params	# FLOPs	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
ResNet-101 [15]	336B	63.2M	40.0	60.5	44.0	36.1	57.5	38.6
PVT-S [54]	245B	44.1M	40.4	62.9	43.8	37.8	60.1	40.3
ResNeXt-101-32x4d [58]	340B	62.8M	41.9	62.5	45.9	37.5	59.4	40.2
Swin-T† [32]	264B	47.8M	42.2	64.6	46.2	39.1	61.6	42.0
Twins-SVT-S [5]	228B	44.0M	42.7	65.6	46.7	39.6	62.5	42.6
Twins-PCPVT-S [5]	245B	44.3M	42.9	65.8	47.1	40.0	62.7	42.9
CMT-S (ours)	249B	44.5M	**44.6**	**66.8**	**48.9**	**40.7**	**63.9**	**43.4**

FIG 12. Instance segmentation results on COCO val2017.

Conclusion

- 본 논문에서는 Vision Transformer들의 장점인 Long Range Dependency 추론 능력과 CNN의 장점인 Locality를 고려한 추론 능력의 이점을 모두 얻기 위해 두가지 모델을 함께 섞은 Hybrid Architecture인 CMT를 제안

- Inverted Residual FFN, Lightweight MHSA 등을 적용하여 연산량을 효과적으로 감소시켰으며, 이로 인해 적은 FLOPs를 가지면서 CNN을 포함한 기존 모델들에 비해서 높은 정확도를 가짐

THANKS FOR YOUR ATTENTION

이찬호
국방인공지능응용학과