Vision AI

2022 arXiv Trends

2022-04

hutom

no.	Paper Title	Research group
1	A ConvNet for the 2020s	FAIR, UC Berkeley
2	The CLEAR Benchmark: Continual LEArning on Real-World Imagery	Carnegie Mellon Univ. Argo AI
3	Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery	University of Texas

no.	Paper Title	Research group
4	Emerging Properties in Self-Supervised Vision Transformers	Facebook AI Research
5	Exploring Plain Vision Transformer Backbones for Object Detection	Facebook AI Research
6	Proper Reuse of Image Classification Features Improves Object Detection	Google Research, Brain Team

no.	Paper Title	Research group
7	MiniViT: Compressing Vision Transformers with Weight Multiplexing (CVPR 2022)	Microsoft Research, Microsoft Cloud+AI
8	Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation (CVPR 2021)	Google
9	Unsupervised Semantic Segmentation by Distilling Feature Correspondences (ICLR 2022)	MIT, Microsoft, Cornell univ, Google

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

Ziheng Wang · Ann Majewicz Fey

https://arxiv.org/abs/1806.05796

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

- minimally invasive robot-assisted surgery 도입으로 문제
  - trainee skills의 평가는 outcome-based analysis / structured checklists / rating scales에 의해 수행
  - 많은 양의 전문가 모니터링. manual rating이 필요
  - surgical motion profiles by nature are nonlinear, non-stationary stochastic processes with large variability, both throughout a procedure, as well within repetitions of the same type of surgical task

- Current approaches with a focus on surgical motions can be divided into two main categories:
  - **descriptive statistic analysis**
  - **predictive modeling-based methods**

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

a novel analytical framework with deep surgical skill model is proposed to directly process multivariate time series via an automatic learning.
We hypothesize the learning-based approach could help to explore the intrinsic motion characteristics for decoding skills and promote an optimal performance in online skill assessment systems.

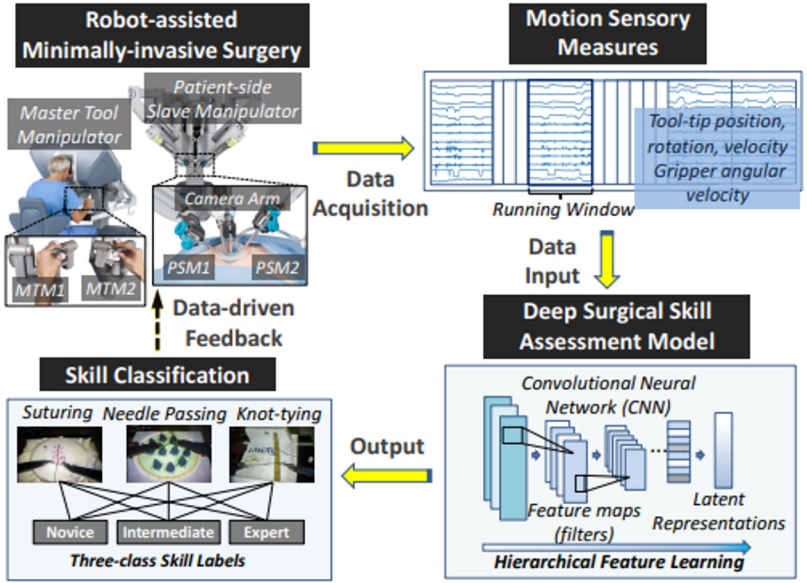


Fig. 1: **An end-to-end framework for online skill assessment in robot-assisted minimally-invasive surgery.** The framework utilizes window sequences of multivariate motion data as an input, recorded from robot end-effectors, and outputs a discriminative assessment of surgical skills via a deep learning architecture.

Without performing manual feature extraction and selection,
latent feature learning is automatically employed on multivariate motion data and directly outputs classifications.

- input:multivariate time series (MTS) of motion kinematics measured from surgical robot end-effectors, X,
- output:the predicted labels representing corresponding expertise levels of trainees, which can be one-hot encoded as $y \in \{1 :$ "*Novice*",2 : "*Intermediate*",3 : "*Expert*"$\}$.

네트워크를 훈련하기 위한 objective cost function는
multinomial cross-entropy cost J로 정의

$$J(\theta) = -\sum_{i=1}^{m}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log p(y^{(i)} = k | x^{(i)}; \theta) \qquad (1)$$

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

introduce a deep architecture using Convolutional Neural Network (CNN) to assess surgical skills from an end-to-end classification.

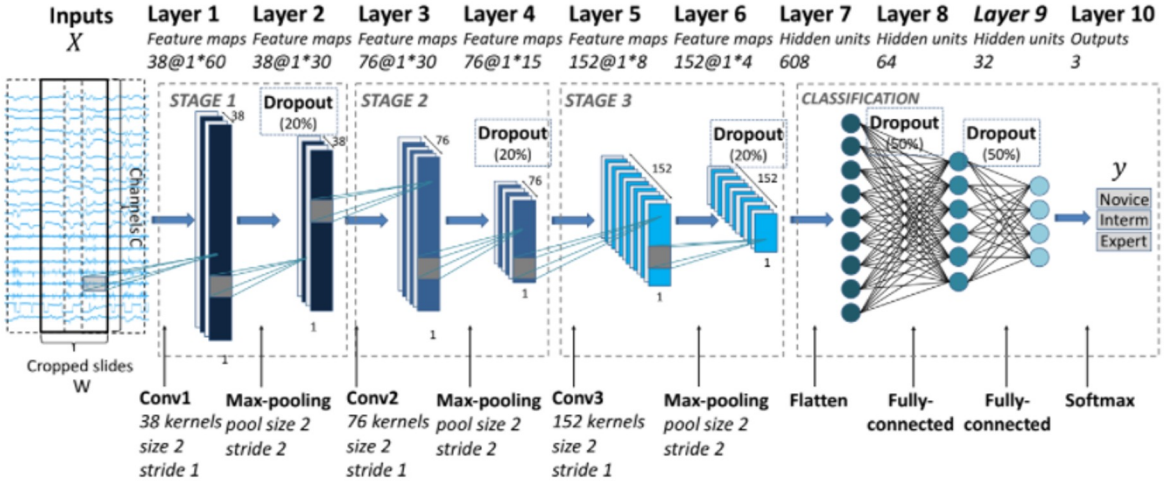


Fig. 2: **Illustrations of the proposed deep architecture using a 10-layer convolutional neural network.** The window width W used in this example is 60. Starting from the inputs, this model consists of three conv-pool stages with a convolution and max-pooling each, one flatten layer, two fully-connected layers, and one softmax layer for outputs. Note that the max-pooling dropout (with probability of 20%) and fully-connected dropout (with probability of 50%) is applied during training.

- convolutional layer,
- pooling layer,
- flatten layer,
- fully-connected layer
- softmax layer.

in the max-pooling and fully-connected layers
the hyper-parameters used for CNN implementation

These hyper-parameters are chosen and fine-tuned by employing the validation set, which is split from training data.

Deep Learning with Convolutional Neural Network for Objective Sk

This paper proposes **an analytical deep learning framework for skill assessment in surgical train**
A deep convolutional neural network is implemented to map multivariate time series data of the motic

Fig. 3: **Snapshots of operation tasks during robot-assisted minimally invasive surgical training**. The operations are implemented using the *da Vinci* robot and are reported in JIGSAWS [40]: (A) *Suturing*, (B) *Needle-passing*, (C) *Knot-tying*.

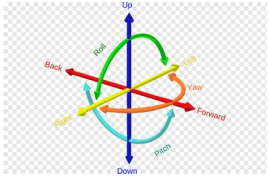
[Dataset]
JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS),
the only public- available minimally invasive surgical database,
which is collected from the *da Vinci* tele-robotic surgical system [40, 49].

The *da* ***Vinci*** **robot** is comprised of

- two master tool manipulators (MTMs) on left and right sides,
- two patient-sides slave manipulators (PSMs),
- an endoscopic camera arm.

Robot motion data are captured (sampling frequency 30 Hz)
as multivariate time series
with 19 measurements
for each end-effector:

- tool tip Cartesian positions (*x*, *y*, *z*),
- rotations (denoted by a 3 × 3 matrix *R*),
- linear velocities (*vx*, *vy*, *vz*),
- angular velocities (ωx′, ωy′, ωz′),
- the gripper angle θ.



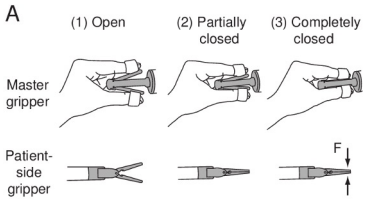


Table 1: **Variables of sensory signals from end-effectors of *da Vinci* robot.** These variables are captured as multivariate time series data in each surgical operation trial.

End-effector Category		Description	Variables	Channels
Master Tool Manipulator (MTM)	*MTM1* *MTM2*	Positions (3), rotation matrix (9), velocities (6) of tool tip, gripper angular velocity (1)	$x, y, z, R \in \mathbb{R}^{3\times3}$, $v_x, v_y, v_z, \omega'_x, \omega'_y, \omega'_z$, α	19×2
Patient-side Manipulator (PSM)	*PSM1* *PSM2*	Positions (3), rotation matrix (9), velocities (6) of tool tip, gripper angular velocity (1)	$x, y, z, R \in \mathbb{R}^{3\times3}$, $v_x, v_y, v_z, \omega'_x, \omega'_y, \omega'_z$, α	19×2

The two ways in which skill labels are reported in JIGSAWS dataset:

- self-proclaimed skill labels
  based on practice hours with *expert* reporting greater than 100 hours, *intermediate* between 10-100 hours, and *novice* reporting less than 10 hours of total surgical robotic operation time,
- a modified global rating scale (GRS) ranging from 6 and 30, manually graded by an experienced surgeon.

novice	intermediate	expert
0-10	10-100	100-

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

[Data Preparation and Inputs]

- *Z-normalization*
  원시 데이터의 각 채널 x는 개별적으로 정규화 $z = \frac{x-\mu}{\sigma}$,
  μ: vector X의 평균
  α: vector X의 표준 편차
  This normalization process can be performed online by feeding the network with the batch of sensory data.
  이 normalization process는 네트워크에 the batch of sensory data를 공급함으로써 온라인으로 실행 가능

- *Data Augmentation*
  JIGSAWS에 대규모 데이터 샘플이 부족 (라벨이 부착된 샘플의 수는 각 수술 작업에 대해 총 40개(8개 피험자, 시험 반복 횟수 5개))
  -> 사용 가능한 데이터 집합의 크기가 제한되면 오버피팅으로 어려움을 겪을 수도
  -> 극복하기 위해 딥 러닝 모델의 오버피팅을 방지하고 generalization를 개선하기 위해 *Data Augmentation* 을 도입
  -> 주로 스케일링, 자르기 및 회전과 같은 여러 가지 방법을 사용하는 이미지 인식에서 확인
  -> time series data에 유사한 *Data Augmentation*을 적용하여 소규모 데이터 세트를 확장하고 decoding 정확도를 높임

  -> 시행착오를 바탕으로 실험한 결과,
  -> JIGSAWS에서는 원기록 길이가 시험마다 다르고, 전체적으로 채취한 크롭의 개수가 다르기 때문에 선택된 설정에 따라
  -> window width W = 60, step size L = 30 선택
  -> 데이터 증강으로 봉합, 니들 패싱 및 매듭-타이에 대해 각각 6290, 6780 및 3542개의 crop을 산출

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

- Training and Testing
  모델 분류를 검증하기 위해 2가지 individual validation schemes 사용
  목적: 딥 러닝의 경우 시스템 개발에 적합한 최적의 validation strategy을 찾기
  방법: 각 cross- validation 설정을 기반으로 봉합(SU), 매듭 묶기(KT), Needle-passing(NP) 등 각 수술 작업에 대한 수술 기술 모델을 교육하고 테스트
  - Leave-One-Supertrial-out(LOSO)
    Leave-one-supertrial-out (LOSO) cross-validation:
    여러 파티션에서 테스트하기 위해 **단일 서브셋을 반복적으로 생략**
    skill assessment를 위한 방법의 건전성을 평가하는 데 가치

실제 상황 (ground truth)	예측 결과 (predict result)	
	Positive	Negative
Positive	TP(true Positive) 옳은 검출	FN(false negative) 검출되어야 할 것이 검출되지 않음
Negative	FP(false positive) 틀린 검출	TN(true negative) 검출되지 말아야할 것이 검출되지 않음

  - Hold-out
    LOSO 교차 검증과는 달리,
    train/test split 분할을 한 번 실시하여 구현되며, 일반적으로 대규모 데이터 세트가 제시될 때 딥 러닝 모델에서 채택

- Modeling Performance Measure
  - average *accuracy* ratio between the sum of correct predictions and the total number of predictions;
  - *precision* ratio of correct positive predictions (*Tp*) and the total positive results predicted by the classifier (*Tp* +*Fp*);
  - *recall* ratio of positive predictions (*Tp*) and the total positive results in the ground-truth (*Tp* +*Fn*);
  - *f1-score* – a weighted harmonic average between *precision* and *recall*.

$$precision = \frac{T_p}{T_p + F_p}$$

$$recall = \frac{T_p}{T_p + F_n}$$

$$f1\text{-}score = \frac{2 * (recall * precision)}{recall + precision}$$

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

Result

Table 2: **Comparison of existing algorithms employed for skill assessment using motion data from JIGSAWS dataset.** We benchmark the results in terms of accuracy based on *LOSO* cross-validation. Models conducting classification on the trial level are categorized as *per-trial basis*.

Author	Algorithm	Labeling Approach	Metric Extraction	Accuracy			Characteristics
				SU	NP	KT	
Lingling *2012* [32]	S-HMM	*Self-proclaim*	gesture segments	97.4	96.2	94.4	• generative modeling • segment-based • per-trial basis
Forestier *2017* [29]	VSM	*Self-proclaim*	bag of words features	89.7	96.3	61.1	• descriptive modeling • feature-based • per-trial basis
Zia *2018* [31]	NN	*Self-proclaim*	entropy features	100	99.9	100	• descriptive modeling • feature-based • per-trial basis
Fard *2017* [24]	*k*-NN LR SVM	*GRS-based*	movement features	89.7 89.9 75.4	*N/A* *N/A* *N/A*	82.1 82.3 75.4	• descriptive modeling • feature-based • two-class skill only • per-trial basis
Current study	CNN	*Self-proclaim* *GRS-based*	*N/A*	93.4 92.5	89.8 95.4	84.9 91.3	• deep learning modeling • no manual feature • per-window basis • online analysis

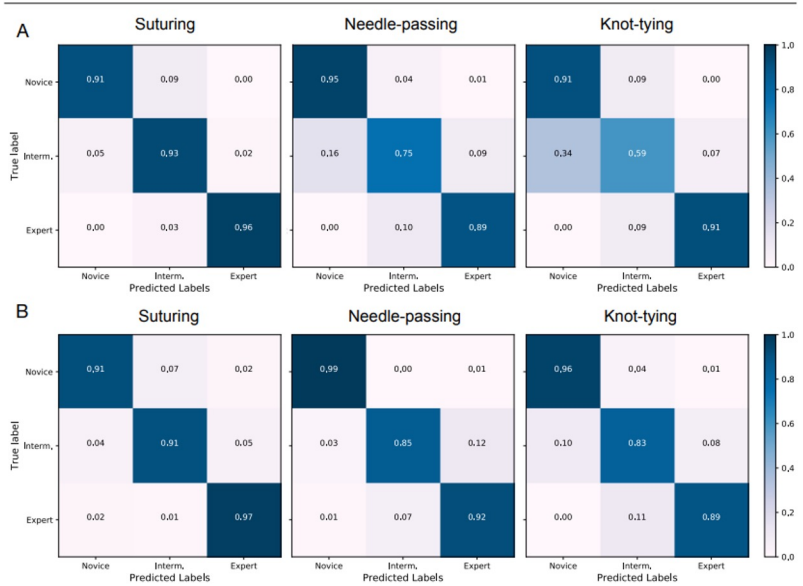


Fig. 4: **Confusion matrices of classification results in three surgical training tasks. (A) self-proclaimed skill classification, (B) GRS-based skill classification.** Element value (i, j) and color represent the probability of predicted skill label j, given the self-proclaimed skill label i, where i and $j \in \{1 : \text{“}Novice\text{”}, 2 : \text{“}Intermediate\text{”}, 3 : \text{“}Expert\text{”}\}$. The diagonal corresponds to correct predictions.

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

다양한 크기의 슬라이딩 윈도우를 사용하여 self-proclaimed skill classification에 대한 실험을 반복

Result

Table 3: **Summary table showing self-proclaimed skill classification performance based on different validation schemes and sliding windows.** Window size is set as $W1 = 30$, $W2 = 60$ and $W3 = 90$. Running time quantifies the computing effort involved in classification. Bold numbers denote best results regarding f1-score, accuracy, and running time

Task	Validation Scheme	Window Size	F1-score			Accuracy	Running Time (ms)
			Novice	Interm.	Expert		
Suturing	LOSO	W1	0.94	0.83	0.95	0.930	**146.45**
		W2	0.94	0.83	0.97	0.934	185.40
		W3	**0.95**	**0.86**	**0.96**	**0.941**	247.01
	Hold-out	W1	0.98	0.92	0.94	0.961	**98.10**
		W2	**0.99**	0.94	0.96	0.972	146.40
		W3	**0.99**	**0.98**	**0.97**	**0.983**	194.79
Needle-passing	LOSO	W1	0.95	0.73	0.88	0.889	**153.36**
		W2	0.95	0.75	**0.90**	0.898	194.98
		W3	**0.96**	**0.76**	0.89	**0.903**	248.03
	Hold-out	W1	0.97	0.80	0.91	0.919	**113.49**
		W2	**0.98**	0.81	0.91	0.925	169.72
		W3	**0.98**	**0.86**	**0.94**	**0.945**	207.12
Knot-tying	LOSO	W1	0.90	0.57	0.90	0.847	**101.83**
		W2	0.90	0.62	**0.92**	0.849	138.25
		W3	**0.92**	**0.64**	0.91	**0.868**	147.38
	Hold-out	W1	0.87	0.42	0.91	0.803	**74.5**
		W2	**0.88**	**0.48**	**0.92**	**0.817**	113.55
		W3	**0.88**	0.47	0.91	0.816	139.39

효율적인 평가를 위해 필요한 최적의 슬라이딩 윈도우에 초점
각 창의 시간 스텝 시간은 입력 신호에서 decoding skill하는 데 필요한 최소 시간과 거의 일치해야

LOSO cross validation: 시스템 성능을 신뢰할 수 있는 추정치를 얻을 수 있음

Hold-out: 상대적으로 큰 편차
-> *Hold-out* validation에서 무작위로 선택된 예제의 차이로 설명 가능
-> 다양한 작업 및 창 크기에 걸쳐 LOSO 체계의 결과와 일관성

-> 각 trial (per-trial basis)에 대한 외과 동작을 완전히 관찰하지 않고도, 윈도우 단위로 매우 시간 효율적인 skill classification를 통해 기존 접근 방식보다 이점을 제공할 수 있음
-> 슬라이딩 윈도우 사이즈가 커짐에 따라 평균 정밀도가 높아짐
-> 90개의 타임 스텝(W 3 = 90)이 포함된 3초 슬라이딩 윈도우는
2초 윈도우(W 2 = 60)에 비해 더 나은 결과(평균 정확도)
-> 봉합에서 0.75%, 니들 패싱에서 0.56%, 매듭-타이팅에서 2.38% 향상

데이터 세트가 클 경우 모델 평가에
LOSO 교차 검증이 덜 효율적일 수 있음
-> LOSO 모델링의 컴퓨팅 부하가 크게 증가
-> 복잡한 딥 아키텍처에는 적합하지 않을 수도
-> 그러나 *Hold-out*은 한 번만 실행하면 되고 모델링 비용이 적게

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

Fig. 4: **Confusion matrices of classification results in three surgical training tasks. (A) self-proclaimed skill classification, (B) GRS-based skill classification.** Element value (i, j) and color represent the probability of predicted skill label j, given the self-proclaimed skill label i, where i and $j \in \{1 : \text{“}Novice\text{”}, 2 : \text{“}Intermediate\text{”}, 3 : \text{“}Expert\text{”}\}$. The diagonal corresponds to correct predictions.

그림 4(A)와 (B):
self-proclaimed skill classification와 GRS-based skill classification에서 봉합과 니들 패싱이 매듭 묶기보다 더 좋은 결과
-> 매듭이 평가하기에 더 어려운 과제임

self-proclaimed skill classification의 경우 대부분의 오분류 오류는 매듭 묶기 작업 중에 발생
self- proclaimed *Intermediate* are misclassified as actual *Novice*.
이 작업에서는 self- proclaimed *Intermediate*이 실제 Novice로 잘못 분류

그림 4(A): distribution across *Intermediate*는 0.34가 Novice로 잘못 분류될 확률로 pronounced
-> robot operations에 소요된 시간을 기반으로 하는 self-proclaimed skill labels이 ground- truth knowledge of expertise을 정확하게 반영하지 못할 수 있기 때문일 수도

Deep Learning with Convolutional Neural Network for Objective Skill Evaluation in Robot-assisted Surgery

This paper proposes **an analytical deep learning framework for skill assessment in surgical training.**
A deep convolutional neural network is implemented to map multivariate time series data of the motion kinematics to individual skill levels.

Table 2: **Comparison of existing algorithms employed for skill assessment using motion data from JIGSAWS dataset.** We benchmark the results in terms of accuracy based on *LOSO* cross-validation. Models conducting classification on the trial level are categorized as *per-trial basis*.

Author	Algorithm	Labeling Approach	Metric Extraction	Accuracy			Characteristics
				SU	NP	KT	
Lingling *2012* [32]	S-HMM	*Self-proclaim*	gesture segments	97.4	96.2	94.4	• generative modeling • segment-based • per-trial basis
Forestier *2017* [29]	VSM	*Self-proclaim*	bag of words features	89.7	96.3	61.1	• descriptive modeling • feature-based • per-trial basis
Zia *2018* [31]	NN	*Self-proclaim*	entropy features	100	99.9	100	• descriptive modeling • feature-based • per-trial basis
Fard *2017* [24]	*k*-NN	*GRS-based*	movement features	89.7	*N/A*	82.1	• descriptive modeling • feature-based • two-class skill only • per-trial basis
	LR			89.9	*N/A*	82.3	
	SVM			75.4	*N/A*	75.4	
Current study	CNN	*Self-proclaim*	*N/A*	93.4	89.8	84.9	• deep learning modeling • no manual feature • per-window basis • online analysis
		GRS-based		92.5	95.4	91.3	

high classification accuracy can be achieved
by a few existing methods
using generative modeling and descriptive modeling.

generative model, sparse HMM (S-HMM):
high predictive accuracy ranging from 94.4% to 97.4%.
-> This result might benefit
from a precise description of motion structures and pre-defined gestures in each task.
-> However, such an approach requires prerequisite segmentation of motion sequences, as well as different complex class-specific models for each skill level [32].

descriptive models sometimes may be superior
to provide highly accurate results, such as the use of novel entropy features.
-> However, the deficiency is that
significant domain-specific knowledge and development is required
to define the most informative features manually,
which directly associate with the final assessment accuracy.
->This deficiency could also explain why there exists a larger variance in accuracy between other studies (61.1%-100%), which are sensitive to the choice of pre-defined features, as shown in Table 2.

A ConvNet for the 2020s

Zhuang Liu[1,2*] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1†]
[1]Facebook AI Research (FAIR) [2]UC Berkeley
Code: https://github.com/facebookresearch/ConvNeXt

https://arxiv.org/pdf/2201.03545v1.pdf

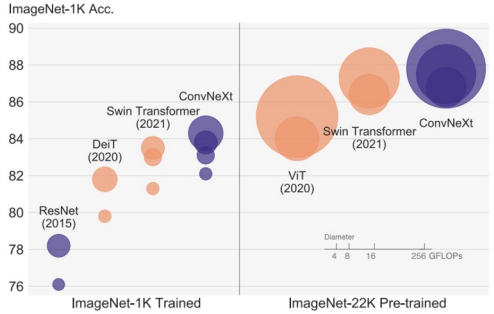
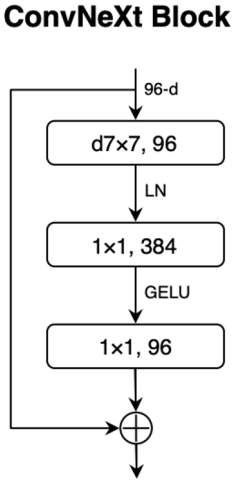


Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take 224^2/384^2 images respectively. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.



<Abstract>

- the effectiveness of hybrid approaches is still largely credited to the intrinsic superiority of Transformers, rather than the inherent inductive biases of convolutions (ex. Swin Transformers)
- **따라서 본 논문에서는 pure ConvNet의 한계를 시험해보겠다! (너의 능력을 보여줘! 느낌)**
  ViT 이전의 ConvNet과 ViT 이후의 ConvNet 의 gap 을 잇는 bridge ⇒ pure ConvNet 이 어디까지 할 수 있는지?
- How do design decisions in Transformers impact ConvNets' performance?
  Transformer 에서 가진 디자인을 ConvNet 에 적용시킨다면, 성능을 더 끌어올릴 수 있지 않을까?
- **ConvNeXt: Constructed entirely from standard ConvNet modules, 새로운 CNN 아키텍처**
  - achieving 87.8% ImageNet top-1 accuracy
  - outperforming Swin Transformers on COCO detection and ADE20K segmentation
  - maintaining the simplicity and efficiency of standard ConvNets

Modernizing a ConvNet

- ResNet-50 / Swin-T / FLOPs around 4.5×10^9 (주로 비교)
- ResNet-200 / Swin-B / FLOPs around 15.0 × 10^9

Network Modernization

1. Training Techniques: Applying similar training techniques used to train ViT and obtain much improved results compared to the original ResNet-50
   a. 최신의 ViT 테크닉을 resnet-50 에 적용
2. macro design
3. ResNeXt
   a. resnet -> resnext 의 테크닉 적용
4. inverted bottleneck
5. large kernel size
6. various layer-wise micro designs.

1. Training Techniques

- Training epochs: 300 epochs (기존 ResNets: 90 epochs. 더 길게 학습.)
- Optimizer: AdamW
- Data augmentation
  - Mixup
  - Cutmix
  - RandAugment
  - Random Erasing
- Regularization
  - Stochastic Depth
  - Label Smoothing
- This training recipe increased the performance of the ResNet-50 model from **76.1% to 78.8%(+2.7%)**

2. Macro Design : Changing stage compute ratio

- Swin-T's computation ratio of each stage: 1:1:3:1
- larger than Swin-T: 1:1:9:1
- ConvNeXt :
  - 기존 resnet 50 의 각 Stage 안의 block 수가 (3,4,6,4) 이었는데,
  - following the design 하여 the number of blocks in each stage: (3, 3, 9, 3) 으로 바꿈.
  - **The accuracy: 78.8% ⇒ 79.4% (+0.6%)**
  - ***we will use this stage compute ratio.***

C: number of channels
B: number of blocks

- ConvNeXt-T: $C = (96, 192, 384, 768), B = (3, 3, 9, 3)$
- ConvNeXt-S: $C = (96, 192, 384, 768), B = (3, 3, 27, 3)$
- ConvNeXt-B: $C = (128, 256, 512, 1024), B = (3, 3, 27, 3)$
- ConvNeXt-L: $C = (192, 384, 768, 1536), B = (3, 3, 27, 3)$
- ConvNeXt-XL: $C = (256, 512, 1024, 2048), B = (3, 3, 27, 3)$

2. Macro Design : Changing stem to "Patchify"

* Stem : 네트워크 처음 들어가는 부분. ⇒ patchify 형태로 바꾸겠다.

- ResNet: The stem cell in standard ResNet: 7×7 convolution layer with stride 2, followed by a max pool (results in a 4×4 downsampling of the input images)
- VIT: "patchify" strategy is used as the stem cell, which corresponds to a large kernel size (e.g. kernel size = 14 or 16) and non-overlapping convolution.
- **Swin Transformer: uses a similar "patchify" layer, but with a smaller patch size of 4 (architecture's multi-stage 때문에)**
- **ConvNeXt: ResNet-style stem cell with a patchify layer implemented using a 4x4, stride 4 convolution layer**
  - ConvNeXt 에서 patchify 를 4*4, stride 4 convolution layer 사용.
- **The accuracy: 79.4% ⇒ 79.5%**
- ***We will use the "patchify stem" (4×4 non-overlapping convolution) in the network.***

3. ResNeXt-ify

- ResNeXt's guiding principle: "use more groups, expand width" : **ResNeXt 에서 group convolution 사용.**
  - 그룹 (cardinality) 을 많이 나누고, 대신에 width (channel) 를 늘려라
- **ConvNeXt: uses depthwise convolution (극단적으로 한다. 그룹 수=256),** a special case of grouped convolution where the number of groups equals the number of channels, ⇒ 연산량이 감소. (group convolution 을 극단적으로 사용) ⇒ depth wise convolution 과 동일.
  increases the network width to the same number of channels as Swin-T's (from 64 to 96)
- **The network performance to 80.5% with increased FLOPs (5.3G)**
  - 그룹을 늘리면서 채널을 줄였더니, FLOPs 가 약간 증가함.
- ***We will now employ the ResNeXt design.***

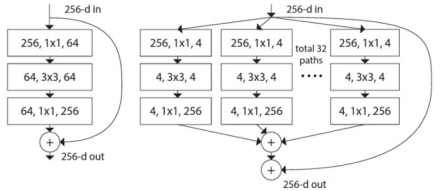


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

- ResNeXt의 핵심은 ResNet 구조의 Architecture에 Inception에서 사용하던 Cardinality 개념을 도입하여 Convolution 연산을 쪼개서 진행하고, 서로 다른 Weight를 구한뒤 합쳐주는 Split-Transform-Merge를 추가한 것이다.
- 이 때, 쪼개진 CNN이 몇개의 path를 가지는지를 결정하는 하이퍼파라미터가 바로 Cardinality이며, 각각의 path에서 가지는 채널을 depth라고 정의한다.
- 따라서 위 그림은, Conv2 Stage 기준 32개의 path와 4의 사이즈를 가지므로, Cardinality = 32, detph = 4의 ResNeXt-50 (32x4d)로 정의할 수 있다.

4. Inverted Bottleneck

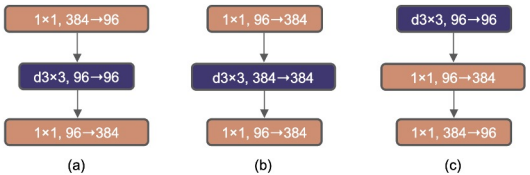
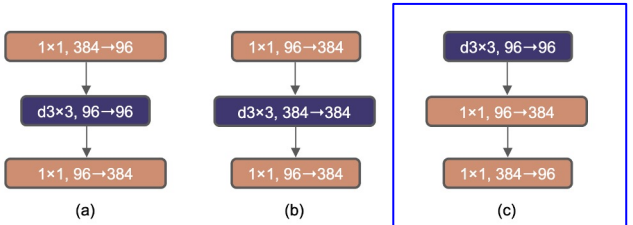


Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

- **results: reduces the whole network FLOPs to 4.6G, slightly improved performance, 80.5% ⇒ 80.6%**
- ***We will now use inverted bottlenecks***



실제 사용되는 inverted block

Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

5. Large Kernel Sizes : Moving up depthwise conv layer

- To explore large kernels, 한 가지 전제조건은 **depthwise conv layer의 위치를 위로 이동하는 것** (Transformers에서 확인가능)
- The MSA block 이 MLP layers 보다 앞선다.
  - MSA (multihead self-attention) 은 depthwise conv layer (3*3) 와 동일한 역할.
  - MSA 가 먼저 하고 MLP layer 를 사용하는 것처럼 depthwise conv layer 를 사용하는 것이 자연스럽다.
- **results: reduces the FLOPs to 4.1G, resulting in a temporary performance degradation to 79.9%.**

5. Large Kernel Sizes : Increasing the kernel size

- experimented with several kernel sizes, including 3, 5, 7, 9, and 11.
- The network's performance increases from 79.9% (3x3) to **80.6% (7x7),** while the network's FLOPs stay roughly the same
- **results: performance increases from 79.9% (3*3) to 80.6% (7*7) while the network's FLOPs stay roughly the same.**
- ***We will use 7×7 depthwise conv in each block***

6. Micro Design : Replacing ReLU with GELU

- Activation functions을 Gaussian Error Linear Unit (GELU) 사용 (ReLU 대신) ⇒ can be thought of as a smoother variant of ReLU
  - Google's BERT, and OpenAI's GPT-2, and, most recently, ViTs
- **ReLU can be substituted with GELU in ConvNet, although the accuracy stays unchanged (80.6%).**

6. Micro Design : Fewer activation functions

- Transformer와 ResNet의 차이: Transformers have fewer activation functions.
- 1x1 conv을 포함한 각 convolutional layer에 activation function을 추가하는 것이 일반적이나,
- **ConvNeXt는 Transformer block의 스타일을 복제하여 두 개의 1x1 layer 사이에 있는 layer을 제외한 나머지 블록에서 모든 GELU layer을 제거.**
- **Result: Increasing 0.7% (81.3%), practically matching the performance of Swin-T.**

6. Micro Design : Fewer normalization layers

- Transformer blocks: usually have fewer normalization layers
- ConvNeXt는 두 개의 BN(BatchNorm) layer을 제거하여 **1x1 layer 앞에 하나의 BN layer만 남김**
- **The accuracy: 81.3% ⇒ 81.4%, surpassing Swin-T's result.**

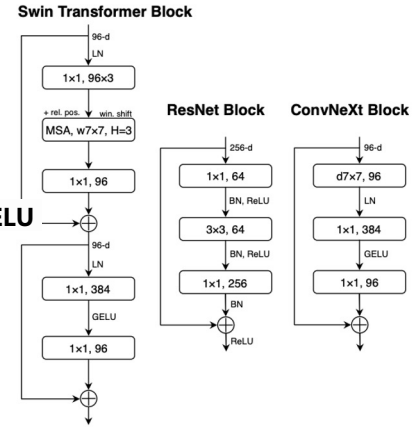


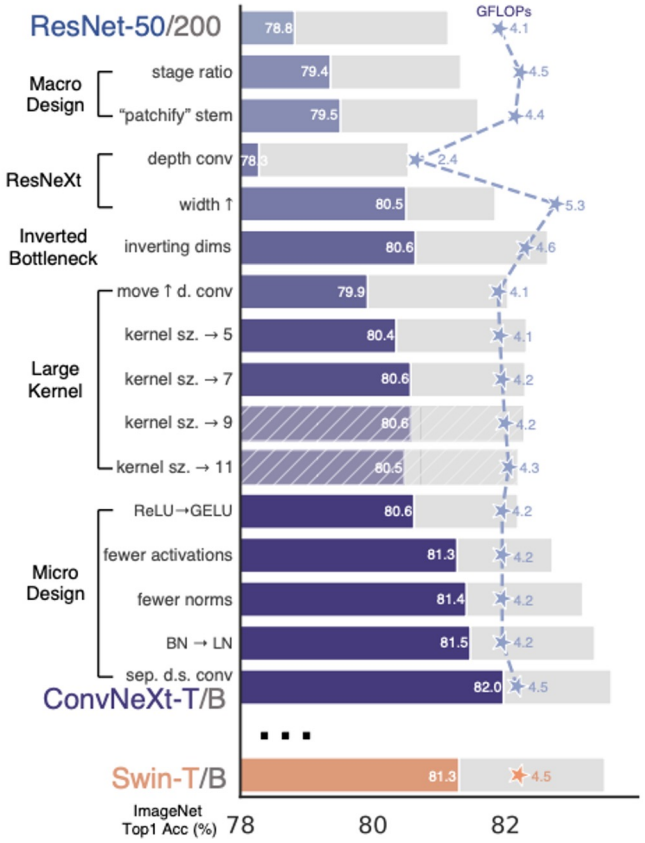
Figure 4. **Block designs** for a ResNet, a Swin Transformer, and a ConvNeXt. Swin Transformer's block is more sophisticated due to the presence of multiple specialized modules and two residual connections. For simplicity, we note the linear layers in Transformer MLP blocks also as "1×1 convs" since they are equivalent.

6. Micro Design : Substituting BN with LN

- **BN보다 Layer Normalization(LN)** has been used in Transformers (좋은 성능 나옴)
- 오리지널 ResNet에서 LN을 BN으로 직접 대체하면 성능이 좋지는 않으나, **ConvNeXt 모델은 LN을 사용한 training에 문제 없음**
- **The accuracy: 81.4% ⇒ 81.5%**

6. Micro Design :Separate downsampling layers

- In ResNet: is achieved by the residual block at the start of each stage, using 3x3 conv with stride 2 (1x1 conv with stride 2 at the shortcut connection)
- In Swin T: a separate downsampling layer is added between stages
  - **stage 사이에 따로 downsampling layer 존재.**
- ConvNeXt: uses **2x2 conv layers with stride 2 for spatial downsampling** and **adding normalization layers** wherever spatial resolution is changed
- **The accuracy: 81.5% ⇒ 82% (Swin T 상당히 능가)**



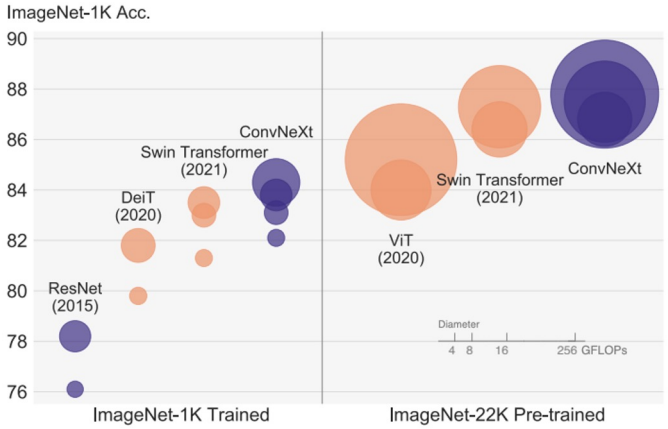


Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take 224^2/384^2 images respectively. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

Table 9. **Detailed architecture specifications** for ResNet-50, ConvNeXt-T and Swin-T.

Results – ImageNet-1K

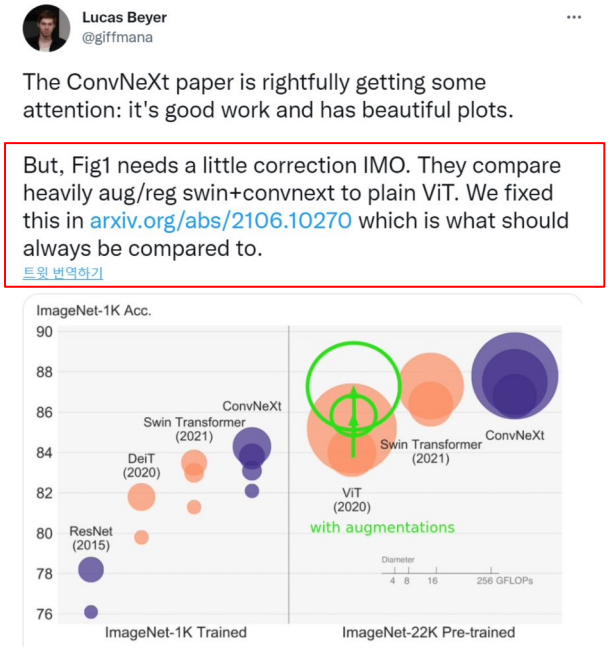
model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-4G [51]	224^2	21M	4.0G	1156.7	80.0
• RegNetY-8G [51]	224^2	39M	8.0G	591.6	81.7
• RegNetY-16G [51]	224^2	84M	16.0G	334.7	82.9
• EffNet-B3 [67]	300^2	12M	1.8G	732.1	81.6
• EffNet-B4 [67]	380^2	19M	4.2G	349.4	82.9
• EffNet-B5 [67]	456^2	30M	9.9G	169.1	83.6
• EffNet-B6 [67]	528^2	43M	19.0G	96.9	84.0
• EffNet-B7 [67]	600^2	66M	37.0G	55.1	84.3
○ DeiT-S [68]	224^2	22M	4.6G	978.5	79.8
○ DeiT-B [68]	224^2	87M	17.6G	302.1	81.8
○ Swin-T	224^2	28M	4.5G	757.9	81.3
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.1**
○ Swin-S	224^2	50M	8.7G	436.7	83.0
• ConvNeXt-S	224^2	50M	8.7G	447.1	**83.1**
○ Swin-B	224^2	88M	15.4G	286.6	83.5
• ConvNeXt-B	224^2	89M	15.4G	292.1	**83.8**
○ Swin-B	384^2	88M	47.1G	85.1	84.5
• ConvNeXt-B	384^2	89M	45.0G	95.7	**85.1**
• ConvNeXt-L	224^2	198M	34.4G	146.8	**84.3**
• ConvNeXt-L	384^2	198M	101.0G	50.4	**85.5**

Results – ImageNet-22K

ImageNet-22K pre-trained models					
• R-101x3 [36]	384^2	388M	204.6G	-	84.4
• R-152x4 [36]	480^2	937M	840.5G	-	85.4
○ ViT-B/16 [18]	384^2	87M	55.5G	93.1	84.0
○ ViT-L/16 [18]	384^2	305M	191.1G	28.5	85.2
○ Swin-B	224^2	88M	15.4G	286.6	85.2
• ConvNeXt-B	224^2	89M	15.4G	292.1	**85.8**
○ Swin-B	384^2	88M	47.0G	85.1	86.4
• ConvNeXt-B	384^2	89M	45.1G	95.7	**86.8**
○ Swin-L	224^2	197M	34.5G	145.0	86.3
• ConvNeXt-L	224^2	198M	34.4G	146.8	**86.6**
○ Swin-L	384^2	197M	103.9G	46.0	87.3
• ConvNeXt-L	384^2	198M	101.0G	50.4	**87.5**
• ConvNeXt-XL	224^2	350M	60.9G	89.3	**87.0**
• ConvNeXt-XL	384^2	350M	179.0G	30.2	**87.8**

Table 1. **Classification accuracy on ImageNet-1K.** Similar to Transformers, ConvNeXt also shows promising scaling behavior with higher-capacity models and a larger (pre-training) dataset. Inference throughput is measured on a V100 GPU, following [42]. On an A100 GPU, ConvNeXt can have a much higher throughput than Swin Transformer. See Appendix E.

Google Brain

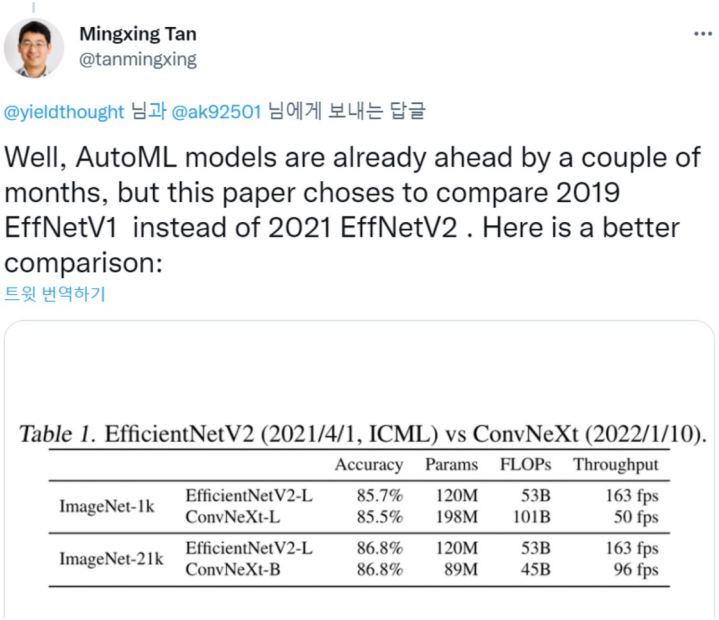


Lucas Beyer
@giffmana

The ConvNeXt paper is rightfully getting some attention: it's good work and has beautiful plots.

But, Fig1 needs a little correction IMO. They compare heavily aug/reg swin+convnext to plain ViT. We fixed this in arxiv.org/abs/2106.10270 which is what should always be compared to.

트윗 번역하기



Mingxing Tan
@tanmingxing

@yieldthought 님과 @ak92501 님에게 보내는 답글

Well, AutoML models are already ahead by a couple of months, but this paper choses to compare 2019 EffNetV1 instead of 2021 EffNetV2 . Here is a better comparison:

트윗 번역하기

Table 1. EfficientNetV2 (2021/4/1, ICML) vs ConvNeXt (2022/1/10).

		Accuracy	Params	FLOPs	Throughput
ImageNet-1k	EfficientNetV2-L	85.7%	120M	53B	163 fps
	ConvNeXt-L	85.5%	198M	101B	50 fps
ImageNet-21k	EfficientNetV2-L	86.8%	120M	53B	163 fps
	ConvNeXt-B	86.8%	89M	45B	96 fps

Isotropic ConvNeXt vs. ViT

model	#param.	FLOPs	throughput (image / s)	training mem. (GB)	IN-1K acc.
○ ViT-S	22M	4.6G	978.5	4.9	79.8
● ConvNeXt-S (*iso.*)	22M	4.3G	1038.7	4.2	79.7
○ ViT-B	87M	17.6G	302.1	9.1	81.8
● ConvNeXt-B (*iso.*)	87M	16.9G	320.1	7.7	82.0
○ ViT-L	304M	61.6G	93.1	22.5	82.6
● ConvNeXt-L (*iso.*)	306M	59.7G	94.4	20.4	82.6

Table 2. **Comparing isotropic ConvNeXt and ViT.** Training memory is measured on V100 GPUs with 32 per-GPU batch size.

Results – **Object Detection** and **Segmentation** on COCO, **Semantic Segmentation** on ADE20K

backbone	FLOPs	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask-RCNN 3× schedule								
○ Swin-T	267G	23.1	46.0	68.1	50.3	41.6	65.1	44.9
● ConvNeXt-T	262G	25.6	**46.2**	67.9	50.8	**41.7**	65.0	44.9
Cascade Mask-RCNN 3× schedule								
● ResNet-50	739G	11.4	46.3	64.3	50.5	40.1	61.7	43.4
● X101-32	819G	9.2	48.1	66.5	52.4	41.6	63.9	45.2
● X101-64	972G	7.1	48.3	66.4	52.3	41.7	64.0	45.1
○ Swin-T	745G	12.2	50.4	69.2	54.7	43.7	66.6	47.3
● ConvNeXt-T	741G	13.5	**50.4**	69.1	54.8	**43.7**	66.5	47.3
○ Swin-S	838G	11.4	51.9	70.7	56.3	45.0	68.2	48.8
● ConvNeXt-S	827G	12.0	**51.9**	70.8	56.5	**45.0**	68.4	49.1
○ Swin-B	982G	10.7	51.9	70.5	56.4	45.0	68.1	48.9
● ConvNeXt-B	964G	11.4	**52.7**	71.3	57.2	**45.6**	68.9	49.5
○ Swin-B‡	982G	10.7	53.0	71.8	57.5	45.8	69.4	49.7
● ConvNeXt-B‡	964G	11.5	**54.0**	73.1	58.8	**46.9**	70.6	51.3
○ Swin-L‡	1382G	9.2	53.9	72.4	58.8	46.7	70.1	50.8
● ConvNeXt-L‡	1354G	10.0	**54.8**	73.8	59.8	**47.6**	71.3	51.7
● ConvNeXt-XL‡	1898G	8.6	**55.2**	74.2	59.9	**47.7**	71.6	52.2

Table 3. **COCO object detection and segmentation results** using Mask-RCNN and Cascade Mask-RCNN. ‡ indicates that the model is pre-trained on ImageNet-22K. ImageNet-1K pre-trained Swin results are from their Github repository [3]. AP numbers of the ResNet-50 and X101 models are from [42]. We measure FPS on an A100 GPU. FLOPs are calculated with image size (1280, 800).

backbone	input crop.	mIoU	#param.	FLOPs
ImageNet-1K pre-trained				
○ Swin-T	512^2	45.8	60M	945G
● ConvNeXt-T	512^2	**46.7**	60M	939G
○ Swin-S	512^2	49.5	81M	1038G
● ConvNeXt-S	512^2	**49.6**	82M	1027G
○ Swin-B	512^2	49.7	121M	1188G
● ConvNeXt-B	512^2	**49.9**	122M	1170G
ImageNet-22K pre-trained				
○ Swin-B‡	640^2	51.7	121M	1841G
● ConvNeXt-B‡	640^2	**53.1**	122M	1828G
○ Swin-L‡	640^2	53.5	234M	2468G
● ConvNeXt-L‡	640^2	**53.7**	235M	2458G
● ConvNeXt-XL‡	640^2	**54.0**	391M	3335G

Table 4. **ADE20K validation results** using UperNet [80]. ‡ indicates IN-22K pre-training. Swins' results are from its GitHub repository [2]. Following Swin, we report mIoU results with multi-scale testing. FLOPs are based on input sizes of (2048, 512) and (2560, 640) for IN-1K and IN-22K pre-trained models, respectively.

The CLEAR Benchmark: Continual LEArning on Real-World Imagery

Zhiqiu Lin[1] **Jia Shi**[1] **Deepak Pathak**[1*] **Deva Ramanan**[1,2*]
[1]Carnegie Mellon University [2]Argo AI

- 기존 CL Bencchmarks, e.g, Permuted-MNIST and Split-CIFAR 의 경우 인공적인 temporal variation으로 데이터셋을 생성
  ⇒ real world 를 잘 반영하지 못함.
- CLEAR

1. 최초의 img classification dataset for CL Learning
   - YFCC100M 기반사용, vision-lingustic data curation (CLIP)
   - even YFCC-100M에서 적절하지 않은 것도 걸러서 error removing
   2. continual semi-supervised를 위한 일정기간의 labeled class, unlabele class 로 구성.
      a. unsupervised 사용하는 것 ⇒ fully supervised data 만 사용하는 SOTA CL algorithm 성능강화
2. IID data 에서 CL evalutation 할때 performance inflate 확인 및 분석
   - 좀더 정확하게 (future 관점으로) “streaming protocol”제안
     - 오늘의 test셋을 내일의 trainset에 맞게 변경할 수 있어 Data curation 단순화
     - 각 기간으로 지정된 모든 데이터가 train-test 모두에 사용되기 때문에 정확한 성능측정으로 model generalization

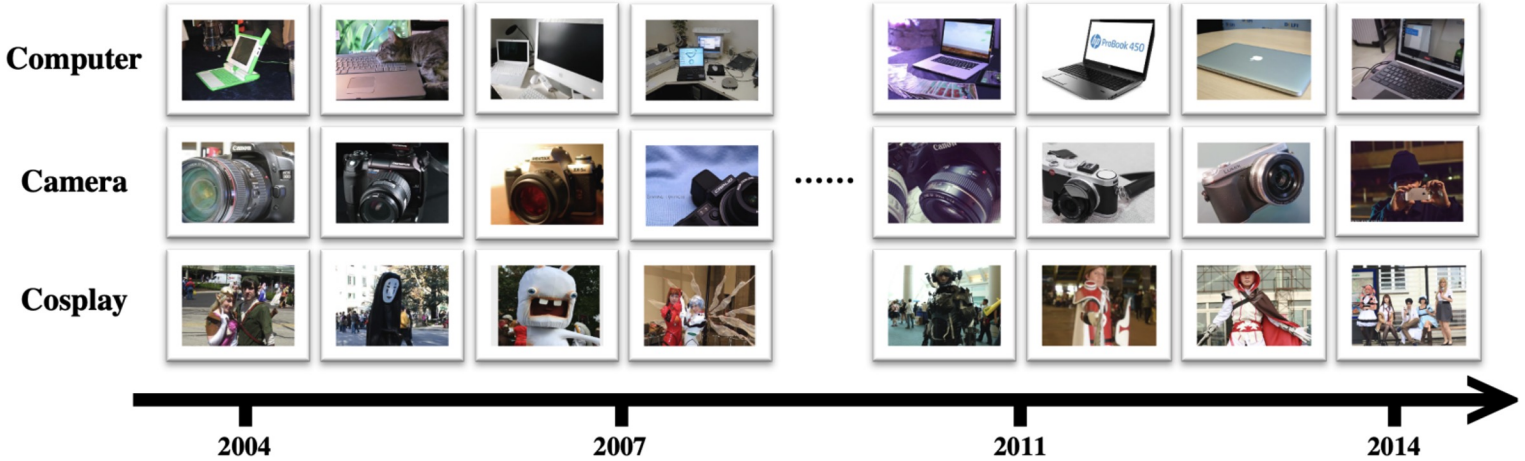


Figure 1: **Temporal evolution of visual concepts in Internet images.** We show the evolution of three concepts (`computer`, `camera`, and `cosplay`) from the Flickr YFCC100M with timestamps spanning 2004 to 2014. The industry advanced rapidly over this decade from Canon EOS 30D (2006) to Canon EOS 6D (2013), and from Apple Powerbook G4 (2004) to Macbook Pro (2011) with substantial design changes. The definition of visual concepts also expanded, e.g., the common usage of the term `camera` evolved from standalone ones to ones in smartphones (iphone 5 in last image of second row). We see evolution in other visual concepts such as `cosplay` as well: Cosplayers often dress as topical characters of the day. From 2004-2007, popular characters reflect anime such as `Rayman Rabbit (2006)`, `Rebuild of Evangelion (2007)`, etc. while 2011-2014 includes characters such as `Assassin's Creed II (2010)`, `Steins;Gate (2011)` and so on.

CL dataset scenario?

- **Incremental Scenario on static datasets**

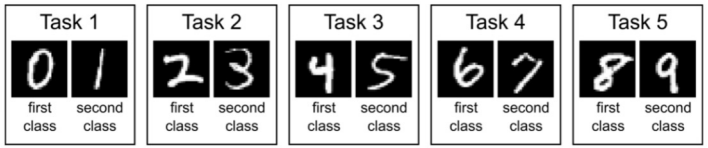


Figure 1: Schematic of split MNIST task protocol.

Task-IL	With task given, is it the 1st or 2nd class? (e.g., 0 or 1)
Domain-IL	With task unknown, is it a 1st or 2nd class? (e.g., in $[0, 2, 4, 6, 8]$ or in $[1, 3, 5, 7, 9]$)
Class-IL	With task unknown, which digit is it? (i.e., choice from 0 to 9)

Table 2: Split MNIST according to each scenario.

CL dataset scenario?

- **Pixel permutation**

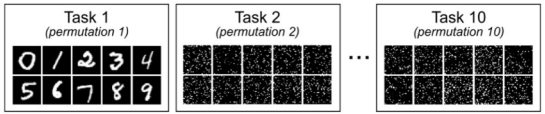


Figure 2: Schematic of permuted MNIST task protocol.

Task-IL	**Given permutation *X*, which digit?**
Domain-IL	With permutation unknown, which digit?
Class-IL	Which digit *and* which permutation?

Table 3: Permuted MNIST according to each scenario.

The second task protocol is 'permuted MNIST' [16], in which each task involves classifying all ten MNIST-digits but with a different permutation applied to the pixels for every new task (Figure 2). Although permuted MNIST is most naturally performed according to the Domain-IL scenario, it can be performed according to the other scenarios too (Table 3).

CL dataset scenario?

- **New Instance Learning**

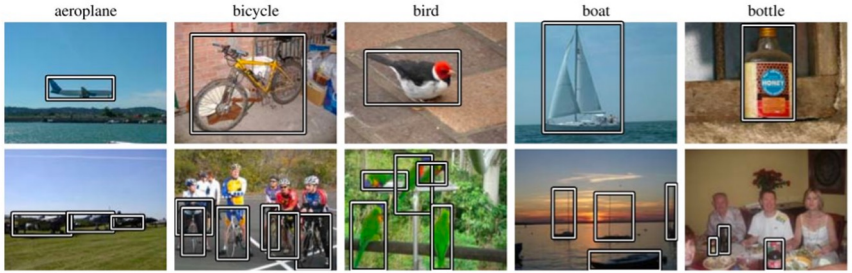
Dataset

CORe50, specifically designed for (**C**)ontinual (**O**)bject (**Re**)cognition, is a collection of 50 domestic objects belonging to 10 categories: plug adapters, mobile phones, scissors, light bulbs, cans, glasses, balls, markers, cups and remote controls. Classification can be performed at object level (50 classes) or at category level (10 classes). The first task (the default one) is much more challenging because objects of the same category are very difficult to be distinguished under certain poses.

Objects are hand hold by the operator and the camera *point-of-view* is that of the operator eyes. The operator is required to extend his arm and smoothly move/rotate the object in front of the camera. A subjective point-of-view with objects at grab-distance is well-suited for a number of robotic applications. The grabbing hand (left or right) changes throughout the sessions and relevant object occlusions are often produced by the hand itself.

CL dataset scenario?

- **Two-task transfer?**
  - **ImageNet followed by Pascal VOC**



ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

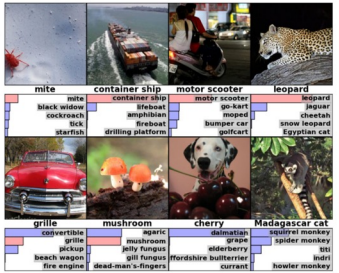


Figure 3: **Example images from CLEAR.** For each bucket (per column), we show a random sample from 5 of the classes (`computer`, `bus`, `camera`, `hockey`, `cosplay`) in CLEAR.

The CLEAR Benchmark: Continual LEArning on Real-World Imagery

- **Continual Learning Setting**
  1. **Task based sequential learning**
     a. **dominant CL paradigm (task based sequantial learning)**
     b. **11-way classification tasks by splitting the temporal stream into 11 buckets.**
     c. **Real world scenario에 따라 boundary 의 정보를 주지 않는 시나리오 (bucket 정보?)**
  2. **Locally-iid assumption**
     a. **task based 학습설정에서 각 task data from an iid distribution**
     b. **동일한 iid 분포에서 train-test data를 sampling하는 주류 eval protocol은 이 iid 가정이 유지될때만 작동**
     c. **즉 eval 할 경우, does not implicitily assume that each task has its own iid distribution.**
  3. **Incremental Task/domain/class learning**
     a. **task based sequential learning을 3가지 incremental learning scecario로 categorize**
     b. **task: 각 task에 대해 classification head 를 나누어서 test sample 이 known 일때, 어떤 head를 사용할건지?**
     c. **domain/class : test sample 이 unknown에 대해서도 들어옴, fixing the label space for all tasks (11-way classification), input distribution is changing over time.**
  4. **Online v.s Offline Continual learning**
     a. **기존 task based sequential learning 은 model학습을 위해 각 sample 을 한번만 보기.**
     b. **online: can be reviseted, but stored memory size 존재**
     c. **onffline: can be revisited without contraint.**

- **Visio-Linguistic Dataset Curation**

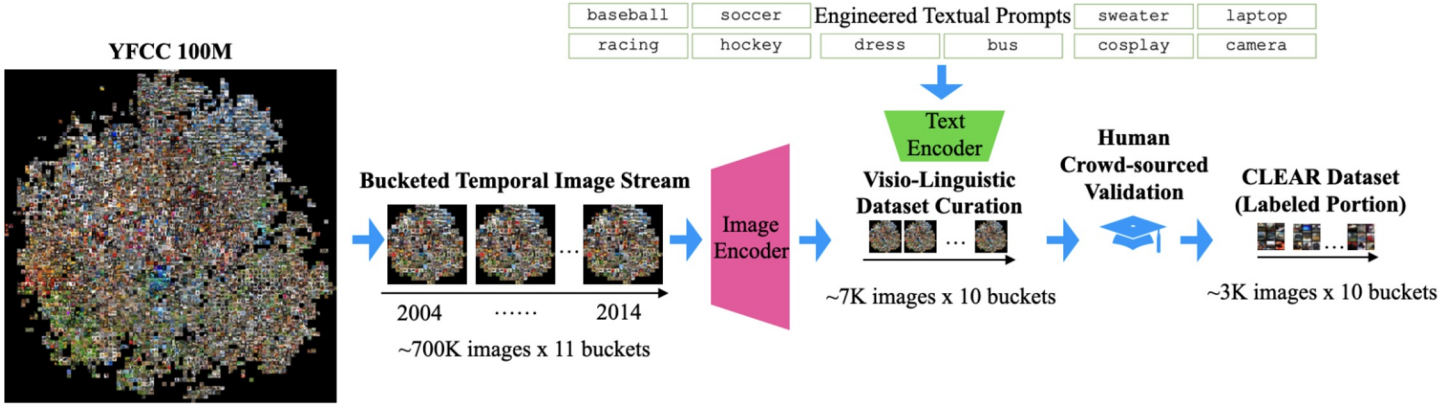


Figure 2: **Visio-Linguistic Dataset Curation.** We download a random subset of 7.8M images and their associated metadata from YFCC100M. We use upload timestamps to reconstruct a temporal stream, splitting it into 11 time-indexed buckets of roughly 700K images each. Given a list of text queries (found by text-prompt engineering in order to effectively retrieve the visual concepts of interest), we use CLIP [46] to extract their respective L2-normalized query features, ranking each image by the cosine similarity of its image feature to each query. We assign the top-ranked (0.7K out of 700K) images of each bucket to the query, removing ambiguous images that rank high across multiple queries. We also include a *background* class with images that rank low across the queries (details in Sec 1. of supplement). We then use human crowd-sourcing (MTurk) to remove misclassified and inappropriate images from the CLIP-retrieved images. As a result, CLEAR contains 3.3K high-quality labeled images for 10 buckets (excluding bucket 0^{th} for unsupervised pre-training only). Our dataset curation pipeline reduces the annotation cost by 99%.

- **Evaluation protocols for continual learning**

Standard iid evaluation protocols of CL algorithms mostly adopt the following 3 metrics, which can be readily calculated from accuracy matrix $\mathcal{R}$:

1. **In-domain Accuracy** (termed Average Accuracy in [12, 38]) measures the test accuracy on the current task immediately after training on it (averaged over all timestamps). This can calculated as the average of the diagonal entries of $\mathcal{R}$.
2. **Backward Transfer** measures the performance of previous tasks (i.e., learning without forgetting) by averaging lower triangular entries of $\mathcal{R}$.
3. **Forward Transfer** measures performance of future tasks (i.e., generalizing to future) by averaging upper triangular entries of $\mathcal{R}$.

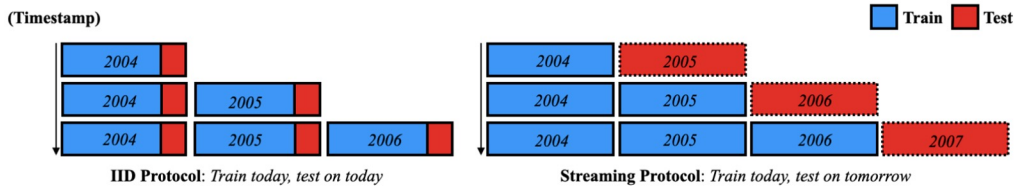
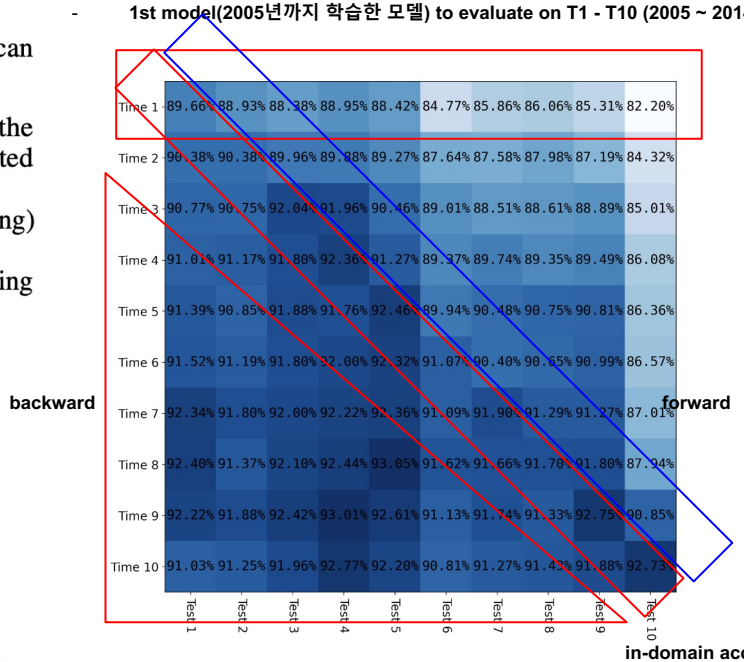


Figure 4: **IID vs Streaming Protocols for CL.** Traditional CL protocols (**left**) split incoming data buckets into a train/test split, typically 70/30%. However, this may overestimate performance since the train and test data are drawn from the same iid distribution, ignoring the train-test domain gap. We advocate a streaming protocol (**right**) where one must always evaluate on near-future data. This allows us to repurpose today's testset as tomorrow's trainset, increasing the total amount of recent data available for both training and testing. Note that the streaming protocol naturally allows for asynchronous training and testing; by the end of year 2006, one can train a model on data up to 2006, but needs additional data from 2007 to test it.

- 1st model(2005년까지 학습한 모델) to evaluate on T1 - T10 (2005 ~ 2014)



- average of superdiagonal of R

$$\textbf{Next-domain Accuracy} = \frac{\sum_{i=1}^{N-1} \mathcal{R}_{i,i+1}}{N-1}$$

The CLEAR Benchmark: Continual LEArning on Real-World Imagery

- **Evaluation protocols for continual learning**

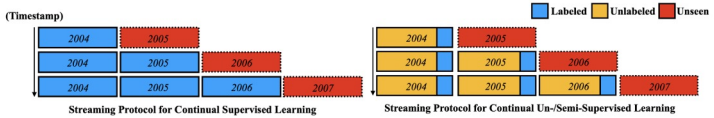


Figure 5: **Streaming Protocols for Continual Supervised vs. Un-/Semi-supervised Learning.** We compare streaming protocols for continual supervised (**left**) and un-/semi-supervised learning (**right**). In real world, most incoming data will not be labeled due to the annotation cost; it is more natural to assume a small labeled subset along with large-scale unlabeled samples per time period. In this work, we achieve great performance ... ing unlabeled samples in the first time period (bucket 0^{th}) for a self-supervised pre-training ... encourage future works to embrace unlabeled samples in later buckets for continual ...ing.

Network	Unsup Repr.	Sampling Strategy	Method	IID Protocol					Streaming Protocol	
				In-domain Acc	Next-domain Acc	Acc	BwT	FwT	Next-domain Acc	FwT
ResNet18	-	N/A	EWC [28]	76.6% ± .2%	74.3% ± .6%	76.7% ± .3%	76.5% ± .4%	71.1% ± .6%	77.1% ± .6%	74.4% ± .6%
ResNet18	-	N/A	SI [58]	76.0% ± .2%	73.6% ± .2%	76.0% ± .5%	76.0% ± .6%	71.0% ± .4%	76.9% ± .2%	74.3% ± .2%
ResNet18	-	N/A	LwF [31]	77.8% ± .3%	75.7% ± .3%	79.2% ± .3%	79.6% ± .3%	72.5% ± .3%	78.8% ± .2%	76.1% ± .3%
ResNet18	-	N/A	CWR [36]	69.5% ± .2%	67.8% ± .3%	68.9% ± .3%	68.8% ± .3%	66.6% ± .3%	71.1% ± .4%	69.9% ± .3%
ResNet18	-	GDumb [44]	GDumb [44]	66.0% ± .4%	64.3% ± .5%	68.4% ± .4%	68.9% ± .4%	61.4% ± .5%	67.4% ± .1%	64.9% ± .1%
ResNet18	-	ER [49]	ER [49]	77.3% ± .1%	75.6% ± .3%	79.0% ± .1%	79.3% ± .1%	72.4% ± .2%	78.1% ± .2%	75.8% ± .2%
ResNet18	-	Reservoir	AGEM [6]	76.2% ± .3%	73.6% ± .2%	75.9% ± .2%	75.9% ± .3%	70.7% ± .2%	77.4% ± .2%	74.5% ± .2%
ResNet18	-	Reservoir	Finetuning	69.5% ± .3%	67.7% ± .2%	70.0% ± .2%	70.0% ± .1%	66.5% ± .2%	71.6% ± .2%	70.6% ± .2%
ResNet18	-	Biased Reservoir	Finetuning	75.5% ± .2%	72.7% ± .3%	75.7% ± .2%	75.8% ± .2%	70.2% ± .2%	77.2% ± .3%	74.4% ± .2%
Linear	YFCC-B0	N/A	EWC [28]	91.6% ± .1%	90.6% ± .0%	91.7% ± .0%	91.7% ± .1%	88.6% ± .0%	91.1% ± .0%	89.3% ± .0%
Linear	YFCC-B0	N/A	SI [58]	91.7% ± .0%	90.5% ± .1%	91.7% ± .0%	91.7% ± .0%	88.5% ± .1%	91.1% ± .0%	89.2% ± .0%
Linear	YFCC-B0	N/A	LwF [31]	91.6% ± .0%	90.7% ± .0%	92.2% ± .0%	92.3% ± .0%	88.7% ± .1%	91.2% ± .0%	89.3% ± .0%
Linear	YFCC-B0	N/A	CWR [36]	90.5% ± .0%	89.8% ± .0%	91.3% ± .0%	91.4% ± .1%	88.1% ± .1%	90.4% ± .0%	88.9% ± .0%
Linear	YFCC-B0	GDumb [44]	GDumb [44]	85.5% ± .3%	85.0% ± .3%	85.5% ± .2%	85.5% ± .3%	84.3% ± .3%	85.8% ± .1%	85.2% ± .1%
Linear	YFCC-B0	ER [49]	ER [49]	91.7% ± .0%	90.9% ± .0%	92.2% ± .0%	92.3% ± .0%	88.9% ± .2%	91.4% ± .1%	89.6% ± .1%
Linear	YFCC-B0	Reservoir	AGEM [6]	91.9% ± .0%	90.7% ± .1%	92.1% ± .1%	92.1% ± .1%	88.6% ± .1%	91.4% ± .0%	89.4% ± .0%
Linear	YFCC-B0	Reservoir	Finetuning	89.3% ± .0%	88.7% ± .0%	90.3% ± .0%	90.6% ± .0%	87.2% ± .0%	89.4% ± .0%	88.0% ± .1%
Linear	YFCC-B0	Biased Reservoir	Finetuning	91.6% ± .0%	90.5% ± .0%	91.7% ± .0%	91.7% ± .0%	88.5% ± .0%	91.1% ± .0%	89.2% ± .0%

Table 1: **Results of baseline CL algorithms on CLEAR.** We evaluated a variety of SOTA algorithms under both **IID** and **Streaming Protocols**. Under the classic **IID Protocol**, **In-domain Acc** (avg of diagonal entries of $\mathcal{R}$) is consistently larger than **Next-domain Acc** (avg of superdiagonal entries of $\mathcal{R}$), indicating that classic (70%-30%) iid train-test construction overestimates performance of real-world CL systems, which must be deployed on future data. Crucially, this drop can be addressed by our **Streaming Protocol**, which trains on all data of the previous bucket (by repurposing yesterday's testset as today's trainset). Moreover, while most prior CL algorithms make use of supervised learning, we find that unsupervised pre-trained representations (YFCC-B0) can boost performances of all baseline algorithms; especially, linear models are far more effective than ResNet18 trained from scratch, even when using naive **Finetuning** strategy with buffer populated with simple reservoir sampling. Note that for replay-based methods, we keep the same buffer size of one bucket of images (2310 for IID and 3300 for streaming protocol).

$$\textbf{Next-domain Accuracy} = \frac{\sum_{i=1}^{N-1} \mathcal{R}_{i,i+1}}{N-1}$$

Emerging Properties in Self-Supervised Vision Transformers

https://arxiv.org/pdf/2104.14294.pdf
2021.5.24. arXiv
meta AI, DINO
https://github.com/facebookresearch/dino

Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron[1,2] Hugo Touvron[1,3] Ishan Misra[1] Hervé Jegou[1]
Julien Mairal[2] Piotr Bojanowski[1] Armand Joulin[1]

[1] Facebook AI Research [2] Inria* [3] Sorbonne University

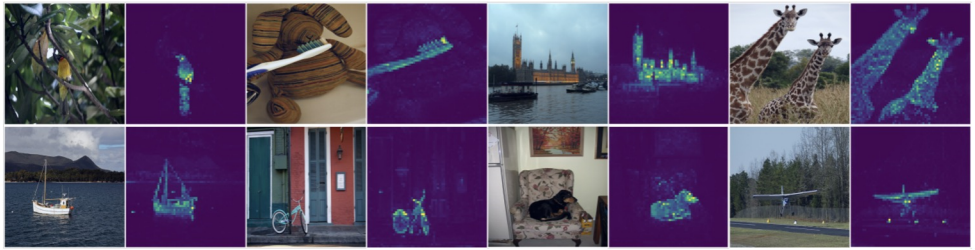


Figure 1: **Self-attention from a Vision Transformer with** 8×8 **patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

The original video is shown on the left. In the middle is a segmentation example generated by a supervised model, and on the right is one generated by DINO. (All examples are licensed from iStock.)

In the **self-supervised learning**, 기존 convolutional network 성능 < ViT 성능 ?!
In this paper,

- self supervised VIT feature 가 convnet 못지않게 semantic segmentation을 위한 정보를 잘 담고 있음을 확인 ⇒ last layer의 cls token self-attention을 통해 직접 시각화 확인
- 해당 feature가 fine-tuning 없이도 k-NN과 함께 사용할 경우 classifier 성능 ↑ (78.3% Top-1 acc on ImageNet)
- Contrastive learning과 같은 기존 self-supervised approach 사용 x
  => Self-Distillation 기법 적용 self-supervised 성공 (*Knowledge distill, teacher와 student 모델이 같음)
  => Self-Distillation with NO labels (DINO) 제안

Emerging Properties in Self-Supervised Vision Transformers

Self-Distillation with NO labels (DINO)

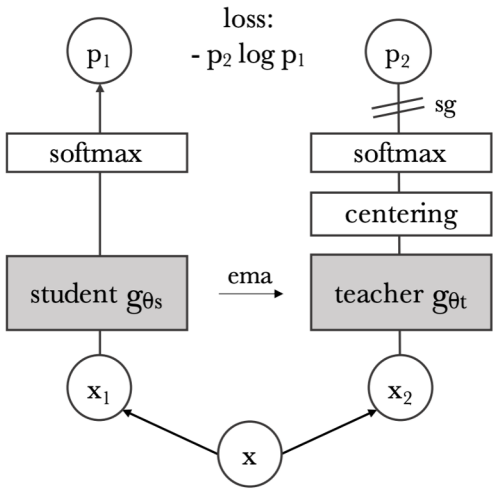
미리 학습된 fixed teacher network를 사용하지 않고

<u>student, teacher 모두 같은 network를 사용해 동시에 학습을 진행하면서</u>

student network로 부터 parameter 의 EMA 를 사용하여 teacher network를 update.

= 매 배치에서 student param update 후에 ema를 통해 teacher model update.

⇒ student가 teacher와 비슷한 class probability distribution을 갖도록 학습.



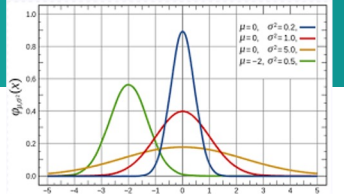
Approach details

- Input Image X에 대해 random augmentation을 적용한 두가지 view인 x1, x2 를 input 으로 사용.
- **Multi-crop augmentation**
  - teacher 는 global crop 만 input으로 사용 (*Global crop: 원본 이미지의 50% 이상 차지하는 crop)
  - student 는 global, local crop 모두를 input으로 사용.

  ⇒ Global crop 에서 얻은 feature 와 local crop 에서 얻는 feature가 동일한 output을 내는 방향으로 model update.
- **Centering & Sharpening 기법**

  teacher 에 적용하여 softmax output이 uniform distribution으로 수렴하는 등의 **model collapse**를 방지.

Emerging Properties in Self-Supervised Vision Transformers



타우의 크기에 따라 정규 분포의 모양이 달라

- 타우가 크면 높이가 낮고, 옆으로 퍼
- 타우가 작으면 높이가 높고, 좁은 모

Self-Distillation with NO labels (DINO)

- Normalized Softmax Output sharpening

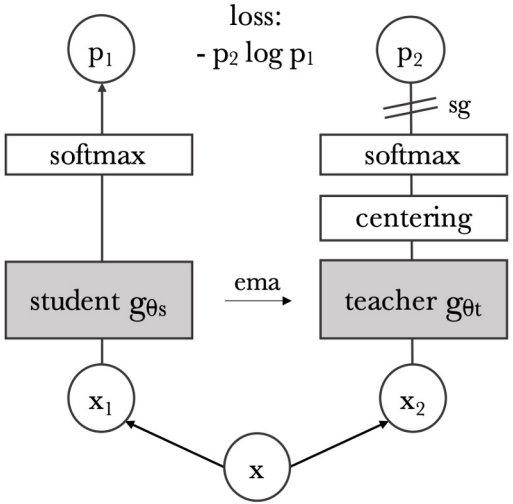
$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K}\exp(g_{\theta_s}(x)^{(k)}/\tau_s)}, \qquad (1)$$

- Loss function (Cross-entropy loss)

$$\min_{\theta_s} H(P_t(x), P_s(x)), \qquad (2)$$

$$\text{where } H(a,b) = -a\log b.$$

- Pseudo Algorithm



Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Emerging Properties in Self-Supervised Vision Transformers

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [30]	ViT-S	21	1007	71.4	66.6
MoCov2* [15]	ViT-S	21	1007	72.7	64.4
SwAV* [10]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
SCLR [12]	RN50w4	375	117	76.8	69.3
SwAV [10]	RN50w2	93	384	77.3	67.3
BYOL [30]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [10]	RN50w5	586	76	78.5	67.1
BYOL [30]	RN50w4	375	117	78.6	–
BYOL [30]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [13]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

Linear and kNN classification on ImageNet:

기존 BYOL, MoCo에 비해 좋은성능 + 큰 모델에서 더 좋은 성능

Video Instance Segmentation on DAVIS 2017 dataset

Video instance segmentation:

첫 frame 에서 target object 에 대한 segmentation mask가 주어지면,

그 뒤에 이어지는 프레임들에서 해당 objects에 대한 segmentation 수행

(J_m: mean region similarity / F_m: contour-based accuracy)

Table 5: **DAVIS 2017 Video object segmentation.** We evaluate the quality of frozen features on video instance tracking. We report mean region similarity $\mathcal{J}_m$ and mean contour-based accuracy $\mathcal{F}_m$. We compare with existing self-supervised methods and a supervised ViT-S/8 trained on ImageNet. Image resolution is 480p.

Method	Data	Arch.	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
Supervised					
ImageNet	INet	ViT-S/8	66.0	63.9	68.1
STM [48]	I/D/Y	RN50	81.8	79.2	84.3
Self-supervised					
CT [71]	VLOG	RN50	48.7	46.4	50.0
MAST [40]	YT-VOS	RN18	65.5	63.3	67.6
STC [37]	Kinetics	RN18	67.6	64.8	70.2
DINO	INet	ViT-S/16	61.8	60.2	63.4
DINO	INet	ViT-B/16	62.3	60.7	63.9
DINO	INet	ViT-S/8	**69.9**	**66.6**	**73.1**
DINO	INet	ViT-B/8	**71.4**	**67.9**	**74.9**

Video Object Segmentation using Space-time Memory Network (ICCV 2019)

Emerging Properties in Self-Supervised Vision Transformers

Attention maps from multiple heads for CLS Token

- DINO로 훈련시킨 VIT-S/8 모델의 마지막 layer 의 self-attention.
- 서로 다른 attention head들이 다양한 object나 region에 focus 하고 있는 것을 볼수 있음.

Segmentation from supervised versus DINO.

- 60% threshold한 self-attention maps를 기준으로 segmentation task 성능 비교.
- PASCAL VOC12 validation set

Supervised

DINO

	Random	Supervised	DINO
ViT-S/16	22.0	27.3	45.9
ViT-S/8	21.8	23.7	44.7

Emerging Properties in Self-Supervised Vision Transformers

Ablation study for self-supervised VIT

- **highlight the difference from the default DINO setting.**
- MAE 가 빠질경우 완전히 Model Collapse.
- Multi-crop 또한 성능에 있어서 중요담당.
- CE ⇒ MSE 로 변경하였을 경우 성능이 많이 떨어짐.

	Method	Mom.	SK	MC	Loss	Pred.	k-NN	Lin.
1	DINO	✓	✗	✓	CE	✗	72.8	76.1
2		✗	✗	✓	CE	✗	0.1	0.1
3		✓	✓	✓	CE	✗	72.2	76.0
4		✓	✗	✗	CE	✗	67.9	72.5
5		✓	✗	✓	MSE	✗	52.6	62.4
6		✓	✗	✓	CE	✓	71.8	75.6
7	BYOL	✓	✗	✗	MSE	✓	66.6	71.4
8	MoCov2	✓	✗	✗	INCE	✗	62.0	71.6
9	SwAV	✗	✓	✓	CE	✗	64.7	71.8

SK: Sinkhorn-Knopp, MC: Multi-Crop, Pred.: Predictor
CE: Cross-Entropy, MSE: Mean Square Error, INCE: InfoNCE

Transfer learning by finetuning pretrained models on different datasets.

- DINO, self-supervised learning 후 다양한 dataset으로 finetuning.
- supervised learning top 1 acc < DINO top 1 acc

Table 6: **Transfer learning by finetuning pretrained models on different datasets.** We report top-1 accuracy. Self-supervised pretraining with DINO transfers better than supervised pretraining.

	Cifar_{10}	Cifar_{100}	INat_{18}	INat_{19}	Flwrs	Cars	INet
ViT-S/16							
Sup. [69]	**99.0**	89.5	70.7	76.6	98.2	92.1	79.9
DINO	**99.0**	**90.5**	**72.0**	**78.2**	**98.5**	**93.0**	**81.5**
ViT-B/16							
Sup. [69]	99.0	90.8	**73.2**	77.7	98.4	92.1	81.8
DINO	**99.1**	**91.7**	72.6	**78.6**	**98.8**	**93.0**	**82.8**

Conclusion

1. Self-supervised vision transformer의 학습된 feature 자체로 k-NN classifier로 사용되어 classification에서 상당한 성능향상.
1. self-supervised VIT의 last layer가 지닌 attention map이 object 에 대한 segmentation mask 정보를 집접적으로 담고있음.
1. self-supervised pre-training 과 fine-tuning 을 통해서 VIT model이 일반적인 supervised learning 보다 더 뛰어난 classification 성능을 보이게 됨

Exploring Plain Vision Transformer Backbones for Object Detection

Yanghao Li Hanzi Mao Ross Girshick† Kaiming He†
†equal contribution

Facebook AI Research

https://arxiv.org/pdf/2203.16527.pdf
2022 Tech Report (Ross Girshick, Kaiming He)

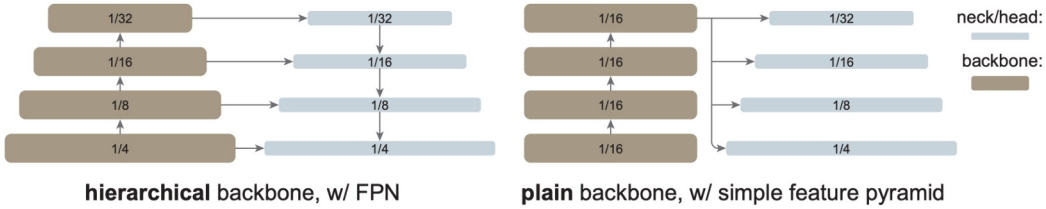


Figure 1: A typical hierarchical-backbone detector (left) *vs*. our plain-backbone detector (right). Traditional hierarchical backbones can be naturally adapted for multi-scale detection, *e.g.*, using FPN. Instead, we explore building a simple pyramid from only the last, large-stride (16) feature map of a plain backbone.

In this paper,

- ***plain, non-hierarchical* Vision Transformer (ViT) as a backbone network for object detection 에 대한 설명 (plain-backbone detectors)**
- object detection 을 위한 original ViT 아키텍처가 **pre-training 을 위한 hierarchical backbone 의 redesign 없이, fine-tuning 되도록** 한다.
- 제안하는 plain-backbone detector 높은 성능을 보임
  - **FPN 디자인 없이, single-scale feature map 으로부터 simple feature pyramid 생성**
  - **shifting 없이 window attention 을 few cross-window propagation blocks 으로 사용하는데 충분함.**

Goal : Remove the hierarchical constraint on the backbone and to enable explorations of plain-backbone object detection.

- 이를 위해, fine-tuning time 동안에만 object detection task 를 위한 plain-backbone 을 최소한 수정하고자 하였다.
- 새로운 components 를 개발하고자 한게 아니라, we focus on what new insights can be drawn in our exploration.

-> Masked Autoencoders로써 plain ViT backbones pre-trained 한 ViTDet 은 좋은 성능을 보임.

- FPN 디자인 없이, single-scale feature map 으로부터 **simple feature pyramid** 생성
- shifting 없이 window attention 을 few cross-window propagation blocks 으로 사용하는데 충분함.

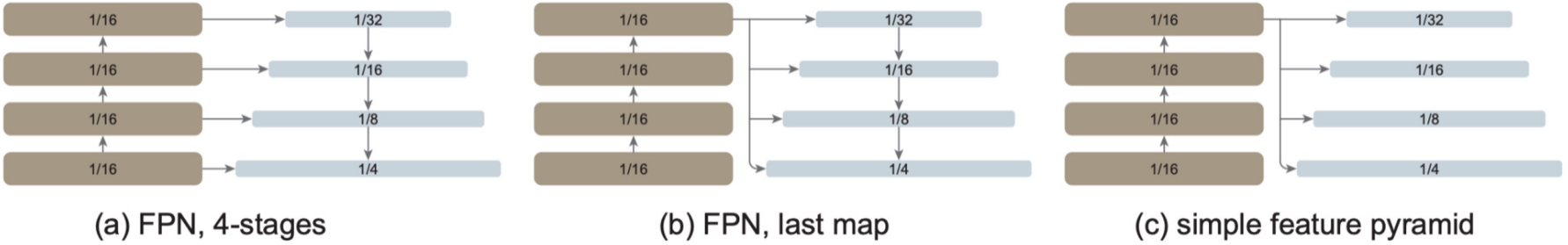


Figure 2: Building a feature pyramid on a plain backbone. **(a)** FPN-like: to mimic a hierarchical backbone, the plain backbone is artificially divided into multiple stages. **(b)** FPN-like, but using only the last feature map without stage division. **(c)** Our simple feature pyramid without FPN. In all three cases, strided convolutions/deconvolutions are used whenever the scale changes.

1. Simple feature pyramid

- FPN:
  - object detection 수행을 위해서, 일반적으로 사용되는 solution
  - 초기 stage에서 higher-resolution feature를, 후기 stage에서 strong features를 combine 하여 사용함.
- 본 시나리오:
  backbone의 last feature map (which should have the strongest features) 만을 사용함.
  해당 feature map 에 convolution, deconvolution 을 *병렬적으로* 적용하여 multi-scale feature maps 을 생성함.

- FPN 디자인 없이, single-scale feature map 으로부터 simple feature pyramid 생성
- shifting 없이 window attention 을 **few cross-window propagation blocks** 으로 사용하는데 충분함.

2. **Backbone adaptation**

- 본 시나리오:
  pre-trained backbone performs global self-attention, which is then *adapted* to higher-resolution inputs during fine-tuning 에 초점을 맞춤.
  - 이는, backbone pre-training 할 때, attention computation 을 직접 수정하는 최근 method 와 차이가 있음.
  - **제안하는 시나리오는 pre-training 아키텍처를 redesigning 할 필요 없이, detection 을 위한 original viT backbone 을 그대로 사용 가능하다.**

- **Using window attention with a few cross-window blocks.**
  - Fine-tuning 할 동안에, regular 형태의 non-overlapping window 로 high-resolution feature map 을 나누고, 각각의 window 에서 Self-attention 을 수행함.
  - Swin ViT 와 다르게, layer 별로 window 를 'shift' 하지 않음.
    information propagation 수행 방법: 4개의 block을 4개의 subset의 pre-trained backbone 으로 나누고, subset 의 마지막 블록에서 2가지 strategies 사용.
    - **1) Global propagation**
    - **2) Convolutional propagation**

[Implementation]

- use:
  the vanilla ViT-B, ViT-L, ViT-H as the pretraining backbones.
- set:
  the patch size as 16
  the feature map scale is 1/16
  i.e., stride = 16.6
- detector heads follow Mask R-CNN [24] or Cascade Mask R-CNN
- input image is 1024×1024
- augmented with large-scale jittering during training
- fine-tune for up to 100 epochs in COCO (Due to this heavy regularization)
- AdamW optimizer
- search for optimal hyper-parameters using a baseline version

Experiments

- **A simple feature pyramid is sufficient**

pyramid design	ViT-B AP^{box}	ViT-B AP^{mask}	ViT-L AP^{box}	ViT-L AP^{mask}
no feature pyramid	47.8	42.5	51.2	45.4
(a) FPN, 4-stage	50.3 (+2.5)	44.9 (+2.4)	54.4 (+3.2)	48.4 (+3.0)
(b) FPN, last-map	50.9 (+3.1)	45.3 (+2.8)	**54.6** (+3.4)	48.5 (+3.1)
(c) simple feature pyramid	**51.2** (+3.4)	**45.5** (+3.0)	**54.6** (+3.4)	**48.6** (+3.2)

Table 1: **Ablation on feature pyramid design** with plain ViT backbones, using Mask R-CNN evaluated on COCO. The backbone is ViT-B (left) and ViT-L (right). The entries (a-c) correspond to Figure 2 (a-c), compared to a baseline without any pyramid. Both FPN and our simple pyramid are substantially better than the baseline, while our simple pyramid is sufficient.

In sum, Table 2 shows that various forms of propagation are helpful, while *we can keep using window attention in most or all blocks*. Importantly, all these architecture adaptations are performed only during fine-tuning time; they do not require a redesign of the pre-training architecture.

- **Window attention is sufficient when aided by a few propagation blocks**

(a) Compare our global and convolutional propagation strategies vs. the no propagation baseline.

(a) Compare different types of residual blocks for convolutional propagation.
  (i) basic (two 3*3), bottlenect (1*1 -> 3*3 -> 1*1), naive (one 3*3)

(b) Study where **cross-window propagation** should be located in the backbone.

(c) Compare the number of global propagation blocks to use.

prop. strategy	AP^{box}	AP^{mask}
none	52.9	47.2
4 global blocks	54.6 (+1.7)	48.6 (+1.4)
4 conv blocks	**54.8** (+1.9)	**48.8** (+1.6)
shifted win.	54.0 (+1.1)	47.9 (+0.7)

(a) Window attention with various cross-window propagation strategies.

prop. conv	AP^{box}	AP^{mask}
none	52.9	47.2
naïve	54.3 (+1.4)	48.3 (+1.1)
basic	**54.8** (+1.9)	**48.8** (+1.6)
bottleneck	54.6 (+1.7)	48.6 (+1.4)

(b) Convolutional propagation with different residual block types (4 blocks).

prop. locations	AP^{box}	AP^{mask}
none	52.9	47.2
first 4 blocks	52.9 (+0.0)	47.1 (–0.1)
last 4 blocks	54.3 (+1.4)	48.3 (+1.1)
evenly 4 blocks	**54.6** (+1.7)	**48.6** (+1.4)

(c) Locations of cross-window global propagation blocks.

prop. blks	AP^{box}	AP^{mask}
none	52.9	47.2
2	54.4 (+1.5)	48.5 (+1.3)
4	54.6 (+1.7)	48.6 (+1.4)
24†	**55.1** (+2.2)	**48.9** (+1.7)

(d) Number of global propagation blocks. †: Memory optimization required.

Table 2: **Ablation on backbone adaptation strategies** using a plain ViT backbone and Mask R-CNN evaluated on COCO. All blocks perform window attention, unless modified by the propagation strategy. In sum, compared to the baseline that uses only window attention (52.9 AP^{box}) most configurations work effectively as long as information can be well propagated across windows. Here the backbone is ViT-L; the observations on ViT-B are similar (see the appendix).

Experiments

- **Masked Autoencoder provide strong pre-trained backbones**
  - Compare backbone pre-training strategies.
  - MAE pre-training can help to relieve this problem.

pre-train	ViT-B AP^{box}	ViT-B AP^{mask}	ViT-L AP^{box}	ViT-L AP^{mask}
none (random init.)	48.1	42.6	50.0	44.2
IN-1K, supervised	47.6 (–0.5)	42.4 (–0.2)	49.6 (–0.4)	43.8 (–0.4)
IN-21K, supervised	47.8 (–0.3)	42.6 (+0.0)	50.6 (+0.6)	44.8 (+0.6)
IN-1K, MAE	**51.2** (+3.1)	**45.5** (+2.9)	**54.6** (+4.6)	**48.6** (+4.4)

Table 4: **Ablation on pre-training strategies** with plain ViT backbones using Mask R-CNN evaluated on COCO.

- **Tradeoffs 실험**

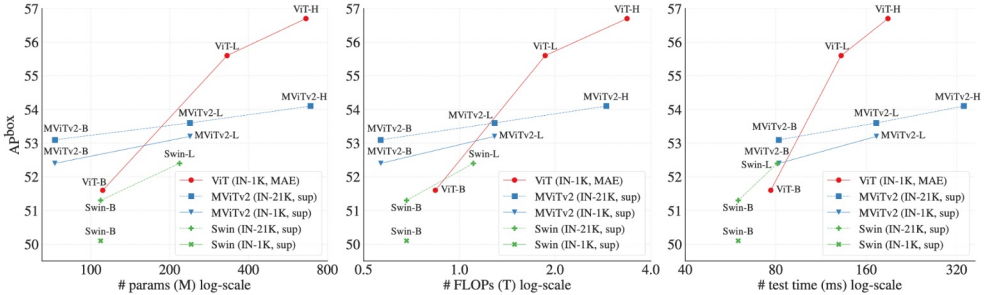


Figure 3: Tradeoffs of accuracy *vs*. model sizes (left), FLOPs (middle), and wall-clock testing time (right). All entries are implemented and run by us to align low-level details. Swin [40] and MViTv2 [32] are pre-trained on IN-1K/21K with supervision. The ViT models are pre-trained using MAE [23] on IN-1K. Here the detector head is Mask R-CNN; a similar trend is observed for Cascade Mask R-CNN. Detailed numbers are in the appendix (Table 9).

Proper Reuse of Image Classification Features Improves Object Detection

https://arxiv.org/pdf/2204.00484.pdf
2020.04.04. arXiv

Cristina Vasconcelos, Vighnesh Birodkar, Vincent Dumoulin
Google Research, Brain Team
{crisnv,vighneshb,vdumoulin}@google.com

In object detection specifically, the **feature backbone** is typically **initialized with ImageNet classifier weights + fine-tuned on the object detection task.**
Recent works:

- this is **not strictly necessary under longer training** regimes
- **recipes** for training the backbone from scratch.
- **transfer learning for object detection에 대한 모순적인 관찰**
  - object detectors benefit from the amount of classification data used in pre-training.
  - the performance gap between transferring from a pre-trained backbone initialization and training the backbone from scratch with smaller, in-domain datasets vanishes with longer training.

In this paper:
revisit **transfer learning in its simplest form**, where the backbone's classifier initialization is frozen during detection training.

- fine-tuning으로 인한 confounding factors 없이 usefulness of the pre-trained representation을 더 잘 이해할 수 있다. (simple, resource saving, easy to replicate)
- weights of a fine-tuned backbone가 pre-trained initialization에서 더 멀어지기 때문에,
  더 긴 training은 usefulness of the pre-trained representation을 조사하는 데 confounding factor이다.
- upstream classification task에서 학습한 representation이
  더 작은 도메인 내 데이터 세트를 사용하여 object detection의 fine-tuning or training 을 통해 얻은 representation보다
  객체 감지에서 더 낫다는 것을 보여주기 때문에 중요하다

Proper Reuse of Image Classification Features Improves Object Detection

freezing backbone을 통해 pre trained representation을 할 때, 대규모 데이터셋에 대한 pre-training 을 통해 consistent performance 관찰

- subsequent detector components가 충분한 용량을 가진 경우,
  models trained with a frozen backbone은 performance of their fine-tuned or “from scratch” counterparts을 능가
- 유사하거나 우수한 성능을 가진 off-the-shelf object detection model 을 훈련하는 동시에 메모리와 FLOP에 대한 계산 리소스의 필요성을 크게 줄일 수 있음
- 제안된 upstream task knowledge preservation의 성능 이점은 클래스 및 사용 가능한 annotation 수별로 결과를 계층화할 때 훨씬 더 명확
- extreme formulation of model reuse가 annotation 수가 적은 클래스에 분명한 긍정적인 영향(

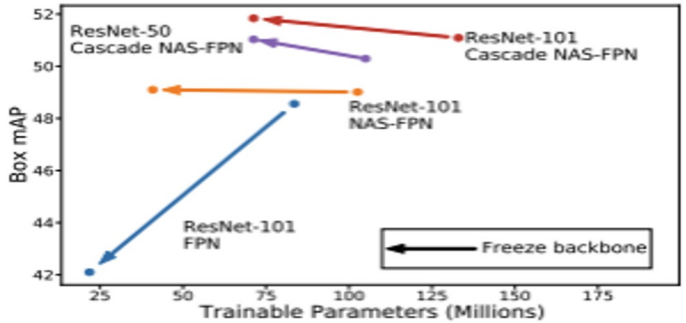
[전략]
classification 중에 preserve the knowledge learnt 하기 위해 classification network의 weights(=backbone)를 freeze

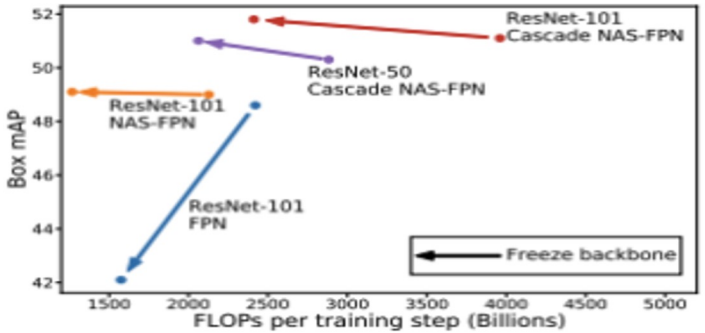
[components]
detection-specific components가 충분한 용량을 가질 때
classification task에서 초기화하고 가중치를 동결하는 것이 fine-tuning이나 처음부터 training하는 것보다 더 잘 수행,
보다 다양한 classification dataset에 대해 pre-train할 때 성능 이득이 증가

[데이터 세트]
MSCOCO (91개의 클래스(실제로 80개 사용)를 포함하는 118k개의 이미지)
LVIS 1.0 (1203개의 클래스를 포함하는 100k개의 이미지)
ImageNet or JFT-300M to pretrain the backbone on classification tasks, except when training from scratch

[Data augmentation]
Large Scale Jittering (LSJ) for all of our experiments + Copy-and-paste augmentation for our best results with EfficientNet

[Training parameters]
frozen backbones hyper-parameters == non-frozen hyper-parameters
batch size 64
lr 0.08
epoch resnet:600 / efficientnetb7s:390





Proper Reuse of Image Classification Features Improves Object Detection

[Experimental]

- **ResNet reuse for object detection**
- EfficientNet-B7's reuse for object detection
- How does preserving pre-trained representations help?
- Beyond backbone freezing

Model	Pretraining	mAP	AP @ 50
ResNet-101 + FPN	From scratch	48.4	70.1
	ImageNet	48.6	70.5
	+ Freeze backbone	(−6.5) 42.1	(−5.2) 65.3
	JFT-300M	48.7	70.5
	+ Freeze backbone	(−5.6) 43.1	(−3.3) 67.2
ResNet-101 + NAS-FPN	From scratch	47.2	68.2
	ImageNet	49.0	70.0
	+ Freeze backbone	(+0.1) 49.1	(+0.3) 70.3
	JFT-300M	49.1	70.2
	+ Freeze backbone	(+1.0) 50.1	(+1.5) 71.8
ResNet-50 + NAS-FPN + Cascade	From scratch	50.0	68.1
	ImageNet	50.3	68.0
	+ Freeze backbone	(+0.7) 51.0	(+1.1) 69.1
	JFT-300M	50.4	68.1
	+ Freeze backbone	(+1.7) 52.1	(+2.3) 70.4
ResNet-101 + NAS-FPN + Cascade	From scratch	50.4	68.4
	ImageNet	51.1	69.1
	+ Freeze backbone	(+0.8) 51.8	(0.8) 69.9
	JFT-300M	51.1	69.0
	+ Freeze backbone	(+1.7) 52.8	(+2.1) 71.1

ImageNet + Freeze backbone과 JFT300M + Freeze backbone 간의 비교
-> 서로 다른 백본 및 테스트된 detector component에 걸쳐 성능이 일관되게 향상되었음

Paper	Pre-training	Schedule	Freeze?	mAP
Pre-training on larger classification datasets helps.				
Sun et al. [47]	ImageNet	Short	Yes	47.8
	JFT	Short	Yes	49.0
Pre-training does not help with longer training schedules.				
He et al. [14]	ImageNet	Long	No	49.0
	JFT	Long	No	49.1
Backbone freezing & high-capacity detector components help.				
Ours	JFT	Long	Yes	**50.1**

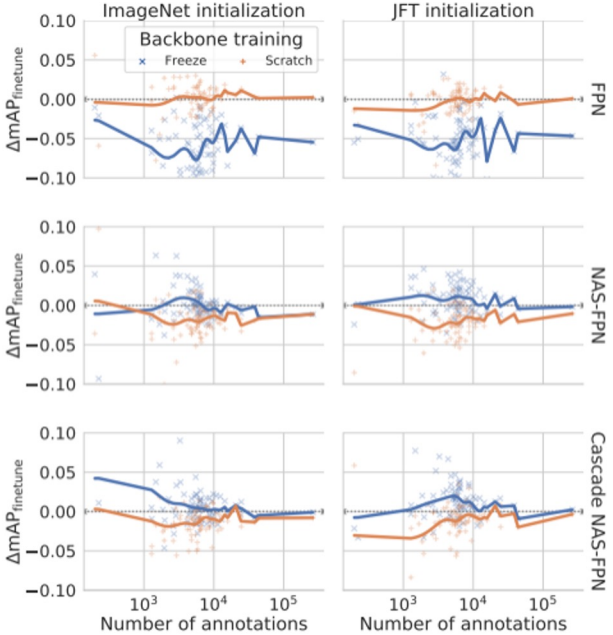
Detection model	#Params (Million)		Δ mAP
	Original	Trained	
FPN	83.5	(26.1%) 21.9	−6.5
NAS-FPN	102.6	(39.9%) 40.9	+0.1
Cascade + NAS-FPN	132.9	(53.6%) 71.3	+0.7

일반적인 설정(예: FPN 사용)에서는 용량이 충분하지 않음
-> benefits from freezing the backbone을 볼 수 없다.
= 파라미터가 충분하다면(NAS-FPN) 대규모 데이터 세트에 대한 pre-train을 통한 학습이 나음
-> comparable performances achieved by fine-tuning the backbone and training it from scratch under a long schedule could simply be a consequence of the fact that fine-tuning for longer moves it further away from the good representation found by pretraining on the classification task.

Proper Reuse of Image Classification Features Improves Object Detection

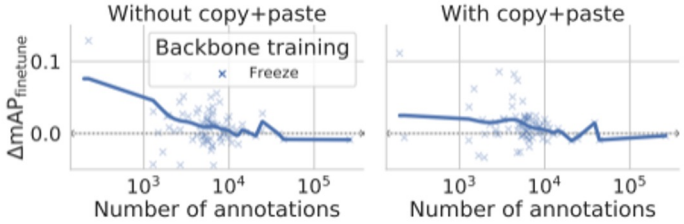
[Experimental]

- ResNet reuse for object detection
- EfficientNet-B7's reuse for object detection
- **How does preserving pre-trained representations help?**
- Beyond backbone freezing



freeze 및 스크래치 train 전략에서 백본을 fine-tunning하여 얻은 것과 비교한 클래스별 박스 mAP

충분한 용량이 주어지면
pre-trained backbone을 detection data에 대해 fine-tune하거나 처음부터 훈련하는 것보다 freeze하는 것이 낫다
-> 왜?
-> 다양한 양의 train annotation을 가진 클래스의 성능에 어떤 영향을 미치는지 시각화함으



Class-wise mAP as a function of the number of training annotations *relative to fine-tuning the backbone*.

LVIS	Box				Mask			
	mAP	mAP_r	mAP_c	mAP_f	mAP	mAP_r	mAP_c	mAP_f
First stage results: regular training								
Copy and Paste [12]	38.5	19.3	37.3	48.2	35.0	19.5	34.9	42.1
+ Freeze backbone	(+1.2) 39.7	(+2.9) 22.2	(+1.5) 38.8	(+0.1) 48.3	(+1.5) 36.5	(+2.9) 22.4	(+1.9) 36.8	(+0.4) 42.5
Second stage: tunes detection-classifier final layer using class-balanced loss								
Copy and Paste [12]	41.6	31.5	39.8	48.0	38.1	32.1	37.1	41.9
+ Freeze backbone	(+1.5) 43.1	(+1.7) 33.2	(+2.1) 41.9	(+0.7) 48.7	(+1.1) 39.9	(+1.5) 33.6	(+2.5) 39.6	(+1.0) 42.9

Table 5. Performance using EfficientNet-B7 + NAS-FPN. Freezing the backbone has the strongest positive performance impact on rare (mAP_r) and common (mAP_c) classes, while still improving frequent (mAP_f) classes. Original first phase results are provided by the authors of [12]. Results without Copy-Paste augmentations can be found in Appendix D.

Proper Reuse of Image Classification Features Improves Object Detection

[Experimental]

- ResNet reuse for object detection
- EfficientNet-B7's reuse for object detection
- How does preserving pre-trained representations help?
- **Beyond backbone freezing**

backbone freezing as a knowledge preservation strategy that yields performance benefits for object detection.
-> better ways of using a pre-trained model for downstream object detection applications,
-> it does **not** mean that backbone freezing is itself an **optimal strategy.**

-> a lightweight alternative strategy in the form of residual adapters, which have been successfully applied to adapt to downstream tasks

Model	Pretraining	mAP	AP @ 50
ResNet-101 + NAS-FPN + Cascade	ImageNet	51.1	69.1
	+ Freeze backbone	(+0.8) 51.8	(+0.8) 69.9
	+ Res. adapters	(+1.9) 53.0	(+2.4) 71.5
	JFT-300M	51.1	69.0
	+ Freeze backbone	(+1.7) 52.8	(+2.1) 71.1
	+ Res. adapters	(+2.5) 53.6	(+3.0) 72.0

Table 7. Adding residual adapters to the frozen backbone and training them along with the subsequent detector components yields another performance increase.

equipping the frozen backbone with residual adapter layers
-> full backbone fine-tuning에 비해 성능개선
-> 추측
-> object detection task에 유용한 aspects of the image classifier representation 보존하면서
-> object detection training signal을 incorporate할 수 있다

MiniViT: Compressing Vision Transformers with Weight Multiplexing

Jinnian Zhang[1,*], Houwen Peng[1,*,†], Kan Wu[1,*], Mengchen Liu[2], Bin Xiao[2], Jianlong Fu[1], Lu Yuan[2]
[1] Microsoft Research, [2] Microsoft Cloud+AI
{v-jinnizhang, houwen.peng, v-kanwu, mengcliu, bin.xiao, jianf, luyuan}@microsoft.com

1. Introduction

"Only Mini Can Do It."

— BMW Mini Cooper

https://arxiv.org/pdf/2204.07154.pdf
Accepted by CVPR 2022

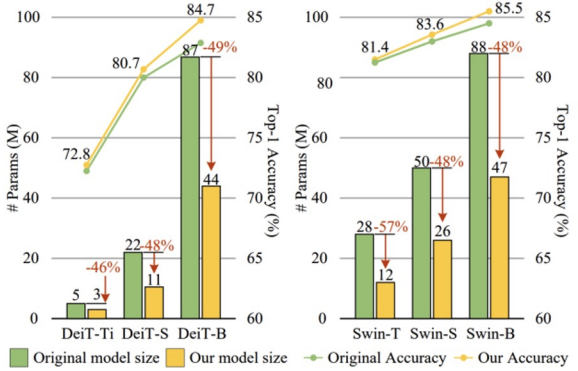


Figure 1. Comparisons between MiniViTs and popular vision transformers, such as DeiT [52] and Swin Transformers [36]

MiniVit : a new compression framework, which achieves parameter reduction in vision transformers while retaining the same performance.

- **성능은 유지하면서, 파라미터 수를 감소시킨 compression (압축) 프레임워크**
- 기존의 ViT 모델은 많은 양의 파라미터 때문에 좋은 모델 성능에도 불구하고, 제한된 메모리에서 사용하는데 제약이 있다. ⇒ MiniVit 제안
- Central Idea
  - Transformer block 의 weight 를 multiplex. 즉, **가중치 변환**에 다양성을 주기 위해서, layer 간 **shared weights** 를 만든다.
  - 또한 large-scale ViT 모델로부터 weight-multiplexed compact 모델로 **weight distillation** 적용.
- Result
  - Pre-trained Swin-B transformer 보다 48% 의 사이즈를 줄이면서, Top-1 accuracy 는 1% 향상시킴.
  - DeiT-B 의 9.7배 (86M -> 9M) 파라미터를 줄미녀서, 성능에 큰 차이가 없도록 함.
- Code, model 제공
  - https://github.com/microsoft/Cream

Weight Multiplexing Strategy

- 두 개의 key component : **1) Weight transformation, 2) weight distillation**
  - 학습 안정성 (training stability) 과 model performance (모델 성능) during weight sharing 을 위하여 두 개의 key component 사용.
- 최종적으로, **weight multiplexing 을 통한 모델 압축 파이프라인 (Compression Pipeline)** 을 소개

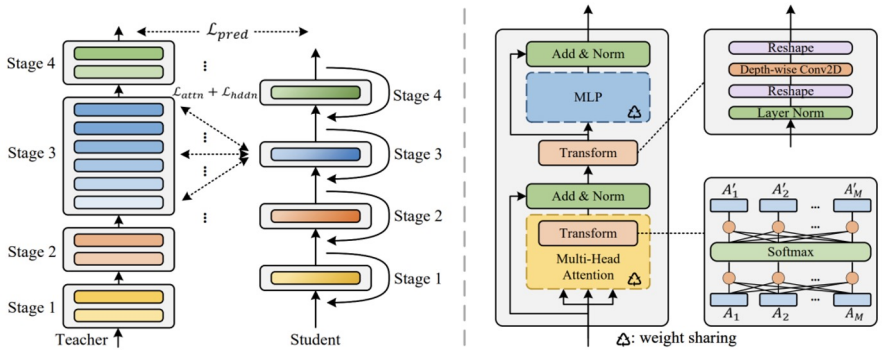


Figure 3. **Left:** The overall framework of MiniViT. Note that the number of stages are configurable, instead of being fixed in Swin transformers [36]. The transformer layers in each stage of the original models to be compressed should have identical structures and dimension. **Right:** The detailed transformer block in a MiniViT. We share weights of MSA and MLP in each stage, and add two transformation blocks to increase the parameter diversity. The transformation blocks and normalization layers are not shared.

Weight transformation

: attention matrices 와 feed-forward network 에 부여됨.

: 파라미터 다양성 (parameter diversity) 를 위함.

- **Transformation for MSA**
  - Two linear transformations before and after the softmax self-attention module.
  - Linear transformation kernels before and after the softmax.
- **Transformation of MLP**
  - MLP 단에서 lightweight transformation 수행.
  - Locality 를 위하여 depth-wise convolution 도입.

Weight Multiplexing Strategy

- 두 개의 key component : **1) Weight transformation, 2) weight distillation**
  - 학습 안정성 (training stability) 과 model performance (모델 성능) during weight sharing 을 위하여 두 개의 key component 사용.
- 최종적으로, **weight multiplexing 을 통한 모델 압축 파이프라인 (Compression Pipeline)** 을 소개

Weight Distillation

- **모델 경량화(압축)를 위해 transfer knowledge from the large models to the small**
- **3가지 distillation 타입**
  - **prediction-logit distillation, self-attention distillation, hidden-state distillation**

$$\mathcal{L}_{train} = \mathcal{L}_{pred} + \beta\mathcal{L}_{attn} + \gamma\mathcal{L}_{hddn}, \qquad (12)$$

where β and γ are hyperparameters with default values of 1 and 0.1 respectively, unless otherwise specified.

Prediction-Logit Distillation. Hinton et al. [25] firstly demonstrated that deep learning models can achieve better performance by imitating the output behavior of well-performing teacher models during training. We leverage this idea to introduce a prediction loss, as follows:

$$\mathcal{L}_{pred} = CE\left(\text{softmax}\left(\frac{\mathbf{z}_s}{T}\right), \text{softmax}\left(\frac{\mathbf{z}_t}{T}\right)\right), \qquad (9)$$

where $\mathbf{z}_s$ and $\mathbf{z}_t$ are the logits predicted by the student and teacher models, respectively, and T is a temperature value which controls the smoothness of logits. In our experiments, we set $T = 1$. CE denotes the cross-entropy loss.

Self-Attention Distillation. Recent literature has shown that utilizing attention maps in transformer layers to guide the training of student models is beneficial [28, 32, 50]. To solve the dimension inconsistency between the student and teacher models due to differing numbers of heads, and inspired by [55], we apply cross-entropy losses on relations among queries, keys, and values in MSA.

$$\mathcal{L}_{attn} = \frac{1}{9N}\sum_{n=1}^{N}\sum_{\substack{i,j\in\\{1,2,3\}}} CE(\mathbf{R}^s_{ij,n}, \mathbf{R}^t_{ij,n}), \qquad (10)$$

Hidden-State Distillation. Similarly, we can generate relation matrices for hidden states, *i.e.*, the features output by MLP. Denoting the hidden states of a transformer layer by $\mathbf{H} \in \mathbb{R}^{N\times d}$, the hidden-state distillation loss based on relation matrices is defined as

$$\mathcal{L}_{hddn} = \frac{1}{N}\sum_{n=1}^{N} CE(\mathbf{R}^s_{H,n}, \mathbf{R}^t_{H,n}), \qquad (11)$$

where $\mathbf{R}_{H,n}$ indicates the n^{th} row of $\mathbf{R}_H$, which is computed by $\mathbf{R}_H = \text{softmax}(\mathbf{H}\mathbf{H}^T/\sqrt{d})$.

Weight Multiplexing Strategy

- 두 개의 key component : **1) Weight transformation, 2) weight distillation**
    - 학습 안정성 (training stability) 과 model performance (모델 성능) during weight sharing 을 위하여 두 개의 key component 사용.
- 최종적으로, **weight multiplexing 을 통한 모델 압축 파이프라인 (Compression Pipeline)** 을 소개

Compression Pipeline

- **2단계로 구성**
    - **Phase 1: Generating compact architectures with weight transformation**
        - Large pre-trained ViT 모델 각 layer에 제안한 weight transformation 적용.
    - **Phase 2: Training the compressed models with weight distillation**
        - Large pre-trained model 에서 small model 로 지식 전이 (transfer knowledge) 를 위하여 제안한 weight distillation 적용.
        - Teacher 과 Student 모델 모두 transformer 아키텍처야 함.
        - 만약 그렇지 않다면, prediction-logit distillation 만 적용함.

Experiments (3가지)

1) **제안한 weight transformation과 weight distillation 기법의 효과 확인**
2) **SoTA 모델과 비교했을 때 파라미터 효율성 (parameter efficiency)**
3) **Compression 모델의 downstream task 성능 (transferability)**

제안한 weight transformation과 weight distillation 기법의 효과 확

Model	#	WS	MUX WD	MUX WT	#Params	Top-1 Acc(%)	Top-5 Acc(%)
Swin-T	1				28M	81.2	95.5
	2	✓			12M	79.0	94.4
	3	✓	✓		12M	79.8	95.1
	4	✓		✓	12M	79.2	94.3
	5	✓	✓	✓	12M	81.4	95.8
DeiT-S	6				22M	79.9	95.0
	7	✓			11M	78.3	94.4
	8	✓	✓		11M	80.1	95.1
	9	✓		✓	11M	79.3	94.8
	10	✓	✓	✓	11M	80.7	95.4

Table 1. Component-wise analysis on ImageNet-1K [16]. WS: Weight Sharing, WD: Weight Distillation, WT: Weight Transformation, MUX: Weight Multiplexing including both WD and WT.

Sharing Strategy	Swin-T #Params	Swin-T Top-1	Swin-T Top-5	DeiT-B #Params	DeiT-B Top-1	DeiT-B Top-5
original	28M	81.2	95.5	86M	81.8	95.6
every-2	16M(1.8×)	82.2	96.2	44M(2.0×)	83.2	96.5
all layers	12M(2.3×)	81.4	95.8	9M(9.7×)	79.8	94.9

Table 2. Ablation study on the number of sharing blocks on Imagenet-1K [16]. Fisrt row: the original model. Second row: sharing two consecutive layers. Third-row: sharing all layers of DeiT [52] and in each stage of Swin [36].

Model	GT	$\mathcal{L}_{pred}$	$\mathcal{L}_{attn}$	$\mathcal{L}_{hddn}$	Top-1	Top-5
Swin-B (22k) (Teacher)	✓				85.2	97.5
Mini-Swin-T w/o distillation	✓				79.2	94.3
Mini-Swin-T		✓			81.2	95.6
	✓	✓			80.9	95.5
		✓	✓		81.4	95.7
		✓		✓	81.5	95.8
		✓	✓	✓	81.4	95.8

Table 3. Ablation study on different distillation losses on ImageNet-1K [16]. We use Swin-B [36] pre-trained on ImageNet-22K [16] as the teacher model. The weights for all losses are 1 except using both GT and $\mathcal{L}_{pred}$, where both weights are 0.5.

Experiments

SoTA 모델과 비교했을 때 파라미터 효율성 (parameter efficiency)

Model	#Param. (M)	MACs (B)	IN-1k Acc (%)	IN-Real Acc (%)	IN-V2 Acc (%)	Input
Convnets						
ResNet-50 [23, 57]	25	4.1	69.8	77.3	57.1	224^2
RegNetY-16GF [44]	84	15.9	82.9	88.1	72.4	224^2
EfficientNet-B1 [51]	8	0.7	79.1	84.9	66.9	240^2
EfficientNet-B5 [51]	30	9.9	83.6	88.3	73.6	456^2
Transformers						
ViT-B/16 [18]	86	55.6	77.9	83.6	-	384^2
ViT-L/16 [18]	307	191.5	76.5	82.2	-	384^2
VTP (20%) [64]	67	13.8	81.3	-	-	224^2
VTP (40%) [64]	48	10.0	80.7	-	-	224^2
AutoFormer-T [11]	6	1.3	74.7	-	-	224^2
AutoFormer-S [11]	23	5.1	81.7	-	-	224^2
AutoFormer-B [11]	54	11.0	82.4	-	-	224^2
S^2ViTE-T [12]	4	1.0	70.1	-	-	224^2
S^2ViTE-S [12]	15	3.1	79.2	-	-	224^2
S^2ViTE-B [12]	57	11.8	82.2	-	-	224^2
DeiT-Ti [52]	5	1.3	72.2	80.1	60.4	224^2
DeiT-S [52]	22	4.6	79.9	85.7	68.5	224^2
DeiT-B [52]	86	17.6	81.8	86.7	71.5	224^2
DeiT-B↑384 [52]	87	55.6	82.9	87.7	72.4	384^2
DeiT-B⚗ ↑384 [52]	88	55.7	84.5	89.0	74.8	384^2
Mini-DeiT-Ti **(ours)**	3(1.7×)	1.3	72.8(+0.6)	83.5(+3.4)	61.3(+0.9)	224^2
Mini-DeiT-S **(ours)**	11(2.0×)	4.7	80.7(+0.8)	88.4(+2.7)	69.5(+1.0)	224^2
Mini-DeiT-B **(ours)**	44(2.0×)	17.7	83.2(+1.4)	89.6(+2.9)	73.0(+1.5)	224^2
Mini-DeiT-B↑384 **(ours)**	44(2.0×)	56.9	84.7(+1.8)	89.9(+2.2)	75.2(+2.8)	384^2
Swin-B (22k) [36]	88	15.4	85.2	89.2	75.3	224^2
Swin-B↑384 (22k) [36]	88	47.1	86.4	90.0	76.6	384^2
Swin-T [36]	28	4.5	81.2	86.6	69.6	224^2
Swin-S [36]	50	8.7	83.2	87.6	71.9	224^2
Swin-B [36]	88	15.4	83.5	87.8	72.5	224^2
Swin-B↑384 [36]	88	47.1	84.5	88.6	73.2	384^2
Mini-Swin-T **(ours)**	12(2.3×)	4.6	81.4(+0.2)	87.1(+0.5)	70.5(+0.9)	224^2
Mini-Swin-S **(ours)**	26(1.9×)	8.9	83.6(+0.4)	88.7(+1.1)	73.8(+1.9)	224^2
Mini-Swin-B **(ours)**	46(1.9×)	15.7	84.3(+0.8)	89.0(+1.2)	74.4(+1.9)	224^2
Mini-Swin-B↑384 **(ours)**	47(1.9×)	49.4	85.5(+1.0)	89.5(+0.9)	76.1(+2.9)	384^2

Table 4. MiniViT Top-1 accuracy on ImageNet-1K [16], Real [7] and V2 [47] with comparisons to state-of-the-art models. Our MiniViTs consistently outperform existing transformer-based visual models and CNNs with fewer parameters. ↑ denotes fine-tuning with 384^2 resolution.

Compression 모델의 downstream task 성능 (transferability)

Image Classification

Model	#Params	ImageNet1k	CIFAR-10	CIFAR-100	Flowers	Cars	Pets
Grafit ResNet-50 [53]	25M	79.6	-	-	98.2	92.5	-
EfficientNet-B5 [51]	30M	83.6	98.7	91.1	98.5	-	-
EfficientNet-B7 [51]	66M	84.3	98.9	91.7	98.8	94.7	-
ViT-B/32 [18]	86M	73.4	97.8	86.3	85.4	-	92.0
ViT-B/16 [18]	86M	77.9	98.1	87.1	89.5	-	93.8
NViT-T [3]	6.4M	73.9	98.2	85.7	-	-	-
NVP-T [3]	6.9M	76.2	98.3	85.9	-	-	-
DeiT-B [52]	86M	81.8	99.1	90.8	98.4	92.1	-
DeiT-B↑384 [52]	87M	83.1	99.1	90.8	98.5	93.3	-
DeiT-B⚗ [52]	87M	83.4	99.1	91.3	98.8	92.9	-
DeiT-B⚗ ↑384 [52]	88M	84.4	99.2	91.4	98.9	93.9	-
Mini-DeiT-B↑384	44M	84.7	99.3	91.5	98.3	93.8	95.5

Table 5. MiniViT results on downstream classification datasets.

Object Detection

#	Backbone	#Params	WT	WD	Det KD	ImageNet Top1-Acc	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	Swin-T [36]	28M				81.2	48.1	67.1	52.1	31.1	51.2	63.5
2	Mini-Swin-T	12M				79.0	46.5	65.5	50.6	29.9	50.0	61.6
3	Mini-Swin-T	12M	✓			79.2	47.5	66.1	51.8	30.8	50.6	62.2
4	Mini-Swin-T	12M	✓	✓		81.4	47.8	66.5	51.9	30.4	51.3	63.2
5	Mini-Swin-T	12M	✓	✓	✓	81.4	48.6	67.2	52.6	31.0	52.4	64.2

Table 6. Comparison on COCO [35] object detection using Cascade Mask R-CNN [8, 22]. We replace the original backbone with our compressed models, and report the number of parameters of the backbone. We train detectors for 12 epochs.

Conclusion

- A new compression framework (i.e. MiniVit, for vision transformers)
- Method
    - Combines weight sharing, transformation, and distillation to reduce the number of parameters while achieving even better performance compared to the original models.

- Limitations
    - Parameter efficiency 향상에도 불구하고, computational cost 는 weight transformation block 때문에 기존 weight sharing strategy 보다 살짝 증가함.
    - Compression ratio 를 증가시키면, MiniViT 성능이 약간 떨어짐.

Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation

Golnaz Ghiasi*[1] Yin Cui*[1] Aravind Srinivas*†[1,2]
Rui Qian†[1,3] Tsung-Yi Lin[1] Ekin D. Cubuk[1] Quoc V. Le[1] Barret Zoph[1]

[1]Google Research, Brain Team [2] UC Berkeley [3] Cornell University

Figure 2. We use a simple copy and paste method to create new images for training instance segmentation models. We apply random scale jittering on two random training images and then randomly select a subset of instances from one image to paste onto the other image.

[Paper link](#)
CVPR 2021 Oral

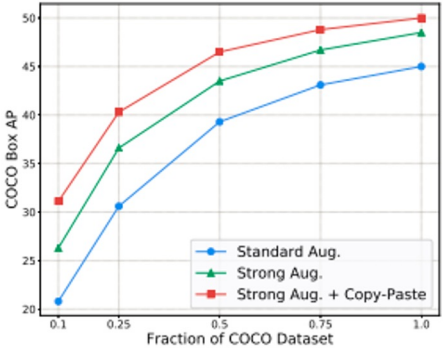


Figure 1. Data-efficiency on the COCO benchmark: Combining the Copy-Paste augmentation along with Strong Aug. (large scale jittering) allows us to train models that are up to 2× more data-efficient than Standard Aug. (standard scale jittering). The augmentations are highly effective and provide gains of +10 AP in the low data regime (10% of data) while still being effective in the high data regime with a gain of +5 AP. Results are for Mask R-CNN EfficientNet-B7 FPN trained on an image size of 640×640.

- **For Instance segmentation, Data Augmentation 기법제안**
- 인스턴스 개체의 **Simple Copy-paste Augmentation**에 대한 연구진행
  - copy-paste 대한 이전 연구는 객체를 붙여넣기 위해 주변의 visual context 를 모델링 하는데 의존. (relate
  - **randomly paste objects onto an image 만으로 확실한 이점제공.**
  - 또한, copy-paste로 pseudo labeling을 통해 추가데이터를 활용하는 방법 제안. (self-training)
- **COCO instance segmentation (49.1 mask AP and 57.3 box AP) ⇒ +0.6 mask AP and +1.5 box AP over the SOTA**
- significant improvements on the LVIS benchmark.
- https://github.com/tensorflow/tpu/tree/master/models/official/detection/projects/copy_paste

- **Copy Paste Augmentation**
  - 1장의 이미지의 객체를 다른 이미지에 붙여 넣는 것
    1. 객체 복사를 위한 source 이미지, 붙여넣을 target 이미지 선택
    2. source 이미지에서 복사를 위한 객체 인스턴스 선택
    3. 복사한 인스턴스들을 target 이미지 내 어디에 붙일지 선택

  - 기존 copy-paste 대한 이전 연구는 객체를 붙일 위치를 결정하기 위해 주변의 visual context 를 모델링 하는데 의존.

    ⇒ 본 논문에서는 더 간단하게 Randomly Copy-paste로 변경.

Figure 2. We use a simple copy and paste method to create new images for training instance segmentation models. We apply random scale jittering on two random training images and then randomly select a subset of instances from one image to paste onto the other image.

Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation

- **Contributions**
  - **Copy-paste show that data efficiency is Improved more than twice as much as “scale jittering data augmentation”**
  - **The augmentations are highly effective and provide gains of +10 AP in the low data regime (10% of COCO dataset)**
  - **easy code portability**

To compose transforms with copy-paste augmentation:

```
import albumentations as A
from albumentations.pytorch.transforms import ToTensorV2
from copy_paste import CopyPaste

transform = A.Compose([
      A.RandomScale(scale_limit=(-0.9, 1), p=1), #LargeScaleJitter from scale of 0.1 to 2
      A.PadIfNeeded(256, 256, border_mode=0), #constant 0 border
      A.RandomCrop(256, 256),
      A.HorizontalFlip(p=0.5),
      CopyPaste(blend=True, sigma=1, pct_objects_paste=0.5, p=1)
    ], bbox_params=A.BboxParams(format="coco")
)
```

[copy-paste github]

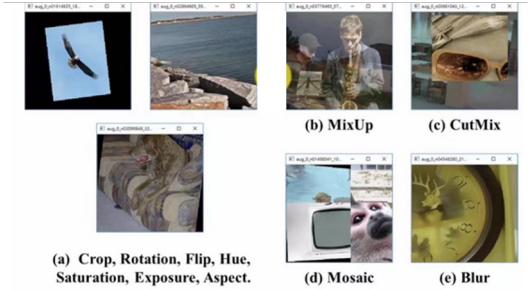
Related work (Data Augmentation)

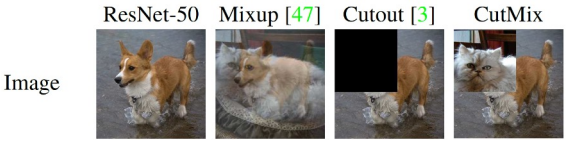
1. **Random Crop, Color Jittering** … ⇒ Image Classification, Self / semi

   (supervised learning 에서 좋은 결과 ⇒범용적으로 사용)

1. **Mixing Image Augmentation**

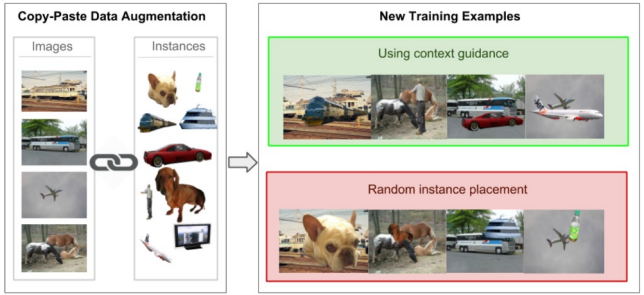
   (Mix 기반, 다른이미지에 포함된 정보와 GT에 대한 변경을 혼합)





3. **기존 Copy-Paste**

   - mixup, cutmix = bonding box 기반으로 다른이미지에 paste.
   - copy-paste = 객체에 대한 부분만 paste.
   - **© context guidance를 주어 적절히 객체 copy paste**



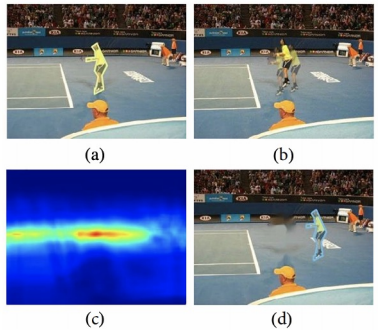


Figure 1. An example of random InstaBoost and appearance consistency heatmap guided InstaBoost. (a) An original image with ground truth mask label from COCO dataset. (b) The result of random InstaBoost. Multiple pastes are visualized showing the randomness. (c) Appearance consistency heatmap of this image. (d) The result of appearance consistency heatmap guided InstaBoost.

- **본 논문에서는 Randomly Copy-Paste, Blending Pasted Objects 사용**
  - adaptive image transformation, visual context 가 필요없음.
  - 붙여진 객체의 edge 를 부드럽게 만들기 위한 Gaussian filter 적용

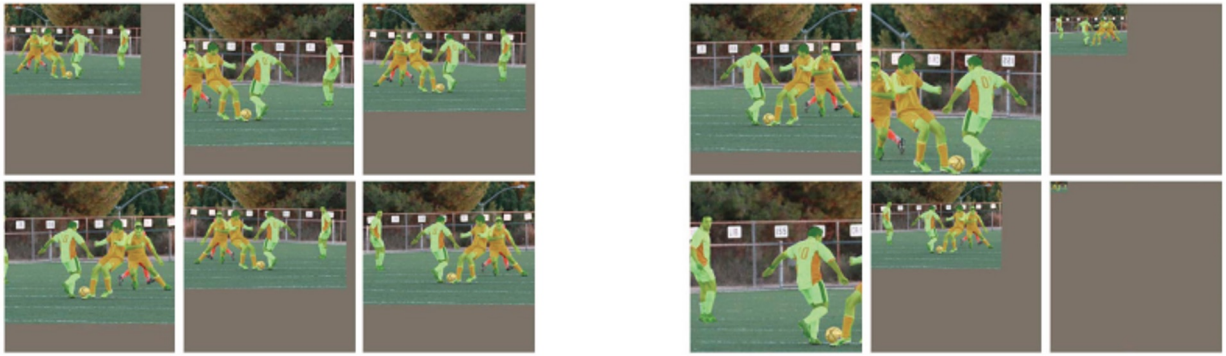
Method

1. **first, randomly pick two images, then applying random scale jittering (SSJ, LSJ) , random horizontal flipping.**
2. **randomly choice to sub-section of objects from one of the image.**
3. **paste sub-section of objects to another image.**
4. **remove fully covered objects, partially covered objects will be updated from bounding box and mask.**

Figure 2. We use a simple copy and paste method to create new images for training instance segmentation models. We apply random scale jittering on two random training images and then randomly select a subset of instances from one image to paste onto the other image.

Large Scale Jittering

- Copy Paste 로 이미지 결합 시 2가지 방법 사용
  - Standard Scale jittering (SSJ)
    - resize range of 0.8 to 1.25 of the original image size
  - **Large Scale Jittering (LSJ) ⇒ 더 좋은 성능.**
    - resize range of 0.1 to 2.0 of the original image size



(a) **Standard Scale Jittering (SSJ)** (b) **Large Scale Jittering (LSJ)**

Figure 3. Notation and visualization of the two scale jittering augmentation methods used throughout the paper. Standard Scale Jittering (SSJ) resizes and crops an image with a resize range of 0.8 to 1.25 of the original image size. The resize range in Large Scale Jittering (LSJ) is from 0.1 to 2.0 of the original image size. If images are made smaller than their original size, then the images are padded with gray pixel values. Both scale jittering methods also use horizontal flips.

Experiments

- Mask R-CNN (EfficientNet, Resent) / Cascade R-CNN (EfficientNet-B7)
    - **Cascade R-CNN best performance model (EfficientNet-B7)**
- 256 batch size, 512 batch size (gpu 용량 생각해서 디자인)
- **Self-training copy-paste experiments (table 실험)**
    - init pretrained ImageNet backbone / init random (2개 모델 비교)
    - LSJ - 576 epoch, SSJ - 96 epoch (그 이후로 떨어지거나 더 이상 변화폭 없을 것으로 추측)
- Dataset
    - 118k coco dataset
    - self-training: coco dataset 120k (unlabeled), Objects365 dataset 610k
    - transfer learning : coco dataset (pre train), PASCAL VOC dataset (fine tune)
    - semantic segmentation: PASCAL VOC 2012 dataset
    - object detection: PASCAL VOC 2007, 2012 dataset
    - benchmark: LVIS v1.0

4.2. Copy-Paste is robust to training configurations

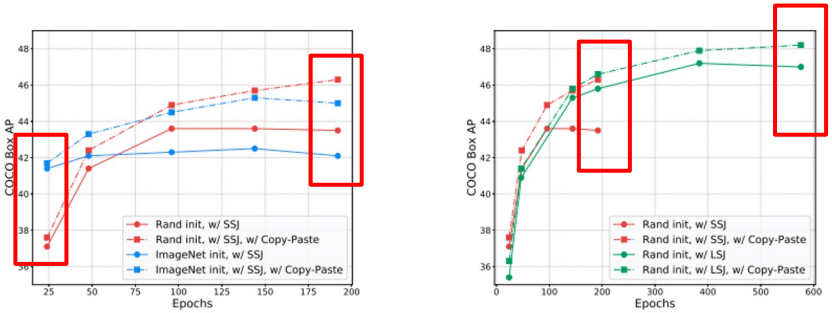


Figure 4. Copy-Paste provides gains that are robust to training configurations. We train Mask R-CNN (ResNet-50 FPN) on 1024×1024 image size for varying numbers of epochs. **Left Figure:** Copy-Paste with and without initializing the backbone by ImageNet pre-training. **Right Figure:** Copy-Paste with standard and large scale jittering. Across all of the configurations training with Copy-Paste is helpful.

4.3. Copy-Paste helps data-efficiency

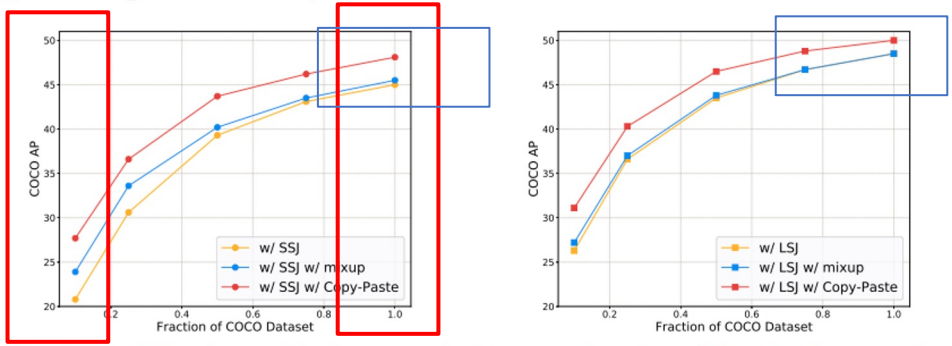


Figure 5. Copy-Paste is additive to large scale jittering augmentation. Improvement from mixup and Copy-Paste data augmentation on top of standard scale jittering **(Left Figure)** and large scale jittering **(Right Figure)**. All results are from training Mask R-CNN EfficientNetB7-FPN on the image size of 640×640.

Model	FLOPs	Box AP	Mask AP
Res-50 FPN (1024)	431 B	47.2	41.8
w/ Copy-Paste	431 B	(+1.0) 48.2	(+0.6) 42.4
Res-101 FPN (1024)	509 B	48.4	42.8
w/ Copy-Paste	509 B	(+1.4) 49.8	(+0.8) 43.6
Res-101 FPN (1280)	693 B	49.1	43.1
w/ Copy-Paste	693 B	(+1.2) 50.3	(+1.1) 44.2
Eff-B7 FPN (640)	286 B	48.5	42.7
w/ Copy-Paste	286 B	(+1.5) 50.0	(+1.0) 43.7
Eff-B7 FPN (1024)	447 B	50.8	44.7
w/ Copy-Paste	447 B	(+1.1) 51.9	(+0.5) 45.2
Eff-B7 FPN (1280)	595 B	51.1	44.8
w/ Copy-Paste	595 B	(+1.5) 52.6	(+1.1) 45.9

Table 1. Copy-paste works well across a variety of different model architectures, model sizes and image resolutions. See table 13 in the Appendix for benchmark results on different object sizes.

4.4. Copy-Paste and self-training are additive

Setup	Box AP	Mask AP
Eff-B7 FPN (640)	48.5	42.7
w/ self-training	(+1.5) 50.0	(+1.3) 44.0
w/ Copy-Paste	(+1.5) 50.0	(+1.0) 43.7
w/ self-training Copy-Paste	**(+2.9) 51.4**	**(+2.3) 45.0**

Table 2. Copy-Paste and self-training are additive for utilizing extra unlabeled data. We get significant improvement of 2.9 box AP and 2.3 mask AP by combining self-training and Copy-Paste.

Setup	Pasting into	Box AP	Mask AP
self-training	-	50.0	44.0
+Copy-Paste	COCO	(+0.4) 50.4	44.0
+Copy-Paste	Pseudo data	(+0.8) 50.8	(+0.5) 44.5
+Copy-Paste	COCO & Pseudo data	(+1.4) 51.4	(+1.0) 45.0

Table 3. Pasting ground-truth COCO objects into both COCO and pseudo labeled data gives higher gain in comparison to doing either on its own.

4.5. Copy-Paste improves COCO state-of-the-art

Model	FLOPs	# Params	AP_{val}	$AP_{test\text{-}dev}$	Mask AP_{val}	Mask $AP_{test\text{-}dev}$
SpineNet-190 (1536) [11]	2076B	176M	52.2	52.5	46.1	46.3
DetectoRS ResNeXt-101-64x4d [43]	—	—	—	55.7†	—	48.5 †
SpineNet-190 (1280) [11]	1885B	164M	52.6	52.8	—	—
SpineNet-190 (1280) w/ self-training [72]	1885B	164M	54.2	54.3	—	—
EfficientDet-D7x (1536) [57]	410B	77M	54.4	55.1	—	—
YOLOv4-P7 (1536) [61]	—	—	—	55.8†	—	—
Cascade Eff-B7 NAS-FPN (1280)	1440B	185M	54.5	54.8	46.8	46.9
w/ Copy-Paste	1440B	185M	(+1.4) **55.9**	(+1.2) **56.0**	(+0.4) **47.2**	(+0.5) **47.4**
w/ self-training Copy-Paste	1440B	185M	(+2.5) **57.0**	(+2.5) **57.3**	(+2.1) **48.9**	(+2.2) **49.1**

Table 4. Comparison with the state-of-the-art models on COCO object detection and instance segmentation. Parentheses next to the model name denote the input image size. † indicates results with test time augmentation.

Model	AP50	AP
RefineDet512+ [68]	83.8	-
SNIPER [52]	86.9	-
Cascade Eff-B7 NAS-FPN	88.6	75.0
w/ Copy-Paste pre-training	(+0.7) **89.3**	(+1.5) **76.5**

Table 5. PASCAL VOC 2007 detection result on `test` set. We present results of our EfficientNet-B7 NAS-FPN model pre-trained with and without Copy-Paste on COCO.

Model	mIOU
DeepLabv3+ † [4]	84.6
ExFuse † [70]	85.8
Eff-B7 [73]	85.2
Eff-L2 [73]	88.7
Eff-B7 NAS-FPN	83.9
w/ Copy-Paste pre-training	(+2.7) **86.6**

Table 6. PASCAL VOC 2012 semantic segmentation results on `val` set. We present results of our EfficientNet-B7 NAS-FPN model pre-trained with and without Copy-Paste on COCO. † indicates multi-scale/flip ensembling inference.

4.7. Copy-Paste provides strong gains on LVIS

performance improvements on Rare class (1-10 images)

	Mask AP	Mask AP_r	Mask AP_c	Mask AP_f	Box AP
cRT (ResNeXt-101-32×8d) [33]	27.2	19.6	26.0	31.9	—
LVIS Challenge 2020 Winner† [55]	38.8	28.5	39.5	42.7	41.1
ResNet-50 FPN (1024)	30.3	22.2	29.5	34.7	31.5
w/ Copy-Paste	(+2.0) 32.3	(+4.3) 26.5	(+2.3) 31.8	(+0.6) 35.3	(+2.8) 34.3
ResNet-101 FPN (1024)	31.9	24.7	30.5	36.3	33.3
w/ Copy-Paste	(+2.1) 34.0	(+2.7) 27.4	(+3.4) 33.9	(+0.9) 37.2	(+3.1) 36.4
EfficientNet-B7 FPN (1024)	33.7	26.4	33.1	37.6	35.5
w/ Copy-Paste	(+2.3) 36.0	(+3.3) 29.7	(+2.7) 35.8	(+1.3) 38.9	(+3.7) 39.2
EfficientNet-B7 NAS-FPN (1280)	34.7	26.0	33.4	39.8	37.2
w/ Copy-Paste	(+3.4) 38.1	(+6.1) 32.1	(+3.7) 37.1	(+2.1) 41.9	(+4.4) 41.6

Table 7. Comparison with the state-of-the-art models on LVIS v1.0 object detection and instance segmentation. Parentheses next to our models denote the input image size. † We report the 2020 winning entry's result without test-time augmentation.

LVIS dataset

- **Conclusion**
  - copy-paste data augmentation method is very effective and robust
  - easy code portability
  - self-training 에도 잘 적용될 수 있음을 확인
  - **instance segmentation task에서 copy-paste 가 기본적인 augmentation 사용되는 방법으로 자리매김**

- **Supply**

Subset of pasted objects. In our method, we paste a *random* subset of objects from one image onto another image. Table 10 shows that although we get improvements from pasting only *one* random object or *all* the objects of one image into another image, we get the best improvement by pasting a *random* subset of objects. This shows that the added randomness introduced from pasting a subset of objects is helpful.

Blending. In our experiments, we smooth out the edges of pasted objects using alpha blending (see Section 3). Table 10 shows that this is not an important step and we get the same results without any blending in contrast to [13] who find blending is crucial for strong performance.

Setup	Box AP	Mask AP
EfficientNetB7-FPN (640)	48.5	42.7
w/ Copy-Paste (one object)	(-0.9) 49.1	(-0.6) 43.1
w/ Copy-Paste (all objects)	(-0.3) 49.7	(-0.4) 43.3
w/ Copy-Paste (no blending)	**50.0**	**43.7**
w/ Copy-Paste	**50.0**	**43.7**

Table 10. Ablation studies for the Copy-Paste method on COCO. We study the value of applying blending to pasted objects along with how many objects to paste from one image to another.

Scale jittering. In this work, we show that by combining large scale jittering and Copy-Paste we obtain a significant improvement over the baseline with standard scale jittering (Figure 1). In the Copy-Paste method, we apply independent random scale jittering on both the pasted image (image that pasted objects are being copied from) and the main image. In Table 11 we study the importance of large scale jittering on both the main and the pasted images. Table 11 shows that most of the improvement from large scale jittering is coming from applying it on the main image and we only get slight improvement (0.3 box AP and 0.2 Mask AP) from increasing the scale jittering range for the pasted image.

Main Image	Pasted Image	Box AP	Mask AP
SSJ	SSJ	(-1.9) 48.1	(-1.6) 42.1
SSJ	LSJ	(-2.3) 47.7	(-1.9) 41.8
LSJ	SSJ	(-0.3) 49.7	(-0.2) 43.5
LSJ	LSJ	**50.0**	**43.7**

Table 11. Ablation study on scale jittering methods for the main image and the pasted image.

- **Conclusion**
  - copy-paste data augmentation method is very effective and robust
  - easy code portability
  - self-training 에도 잘 적용될 수 있음을 확인
  - instance segmentation task에서 copy-paste 가 기본적인 augmentation 사용되는 방법으로 자리매김

- **Supply**

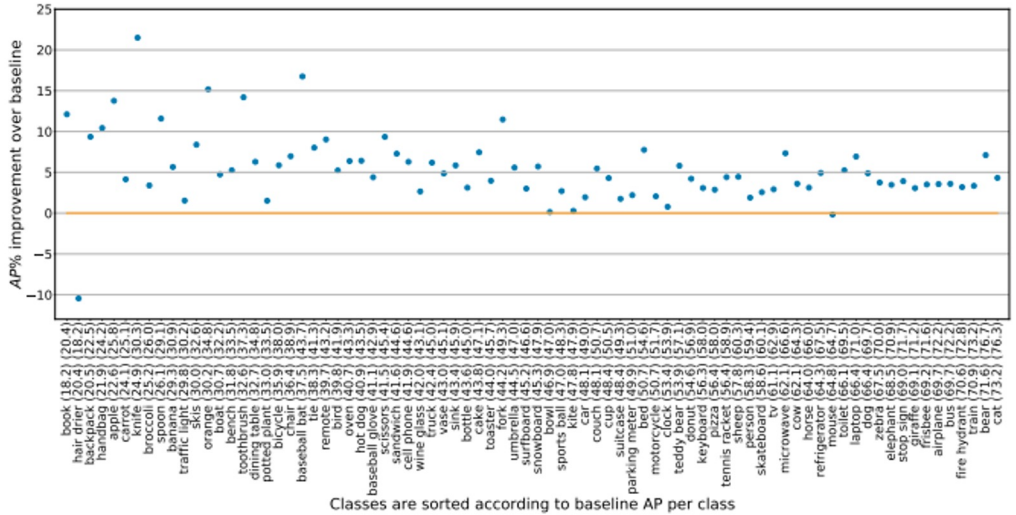


Figure 6. Per category relative AP improvement from Copy-Paste on 80 classes of COCO dataset. Numbers in the parentheses show the AP per category of the baseline model (first number) and the model trained with Copy-Paste (second number). Each number is the average over 5 runs. Classes are sorted based on the baseline AP per class.

Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation

Simple Copy-Paste Is a Strong Data ... - CVF Open Access

G Ghiasi 저술 · 2021 · **172회 인용** — We use a **simple copy** and **paste method** to create new images for training **instance segmentation** models. We apply random scale jittering on two…

Instance Segmentation

Instance Segmentation on LVIS v1.0 val

Leaderboard | Dataset

Filter: untagged — Edit Leaderboard

Rank	Model	mask AP	Paper	Code	Result	Year	Tags
1	Eff-B7 NAS-FPN (1280, Copy-Paste pre-training)	38.1	Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation			2020	
2	Cascade-R101-EQLv2	28.8	Equalization Loss v2: A New Gradient Balance Approach for Long-tailed Object Detection			2020	
3	R101-MaskRCNN-LOCE	28	Exploring Classification Equilibrium in Long-Tailed Object Detection			2021	
4	R101-FPN-MaskRCNN-EQLv2	27.5	Equalization Loss v2: A New Gradient Balance Approach for Long-tailed Object Detection			2020	
5	R50-MaskRCNN-LOCE	26.6	Exploring Classification Equilibrium in Long-Tailed Object Detection			2021	
6	R50-FPN-MaskRCNN-EQLv2	23.7	Equalization Loss v2: A New Gradient Balance Approach for Long-tailed Object Detection			2020	

Images

LVIS

Introduced by Gupta et al. in LVIS: A Dataset for Large Vocabulary Instance Segmentation

LVIS is a dataset for long tail instance segmentation. It has annotations for over 1000 object categories in 164k images.

Source: LVIS

Homepage

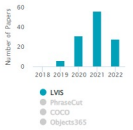
Edit

Benchmarks

Edit

Trend	Task	Dataset Variant	Best Model	Paper	Code
	Few-Shot Object Detection	LVIS v1.0 val	Organizer Provided Baseline		
	Instance Segmentation	LVIS v1.0 val	Eff-B7 NAS-FPN		
	Few-Shot Object Detection	LVIS v1.0 test-dev	Asynchronous SSL		
	Object Detection	LVIS v1.0 val	Eff-B7 NAS-FPN		
	Instance Segmentation	LVIS v1.0 test-dev	R50-FPN-MaskRCNN-TTA		

Usage

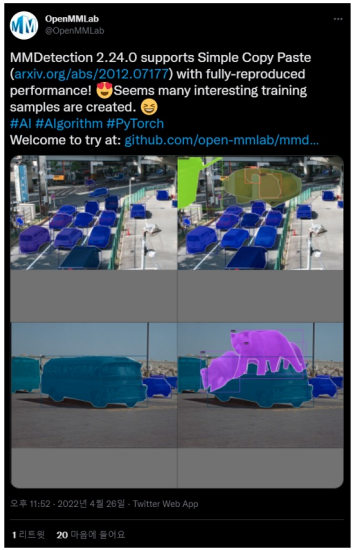


tensorflow/tpu OFFICIAL

Reference models and tools for Cloud TPUs.

4.7K TF — GitHub 7 months ago

twitter
- mmDetection 코드 추가



UNSUPERVISED SEMANTIC SEGMENTATION BY DISTILLING FEATURE CORRESPONDENCES

Published as a conference paper at ICLR 2022

UNSUPERVISED SEMANTIC SEGMENTATION BY DISTILLING FEATURE CORRESPONDENCES

Mark Hamilton
MIT, Microsoft
markth@mit.edu

Zhoutong Zhang
MIT

Bharath Hariharan
Cornell University

Noah Snavely
Cornell University, Google

William T. Freeman
MIT, Google

https://arxiv.org/pdf/2203.08414.pdf

Submitted on 16 Mar 2022
Published as a conference paper at ICLR 2022
a short video detailing the work at https://aka.ms/stego-video

Unsupervised semantic segmentation은 label 없이 하나의 이미지 내에서 semantically meaningful 한 카테고리를 발견하고 localize 하는 것을 목표로 함.
이를 위해서

- **모든 픽셀 단위별**로 **1) semantically meaningful** 하고, **2) cluster 를 구분할 수 있을만큼 compact** 한 **feature 를 생성**해야 함.
- Features for every pixel that are both **semantically meaningful** and **compact enough to form distinct clusters.**

This paper proposes **to separate feature learning from cluster compactification.**
즉, cluster compactification 을 활용하여 separate feature learning 을 제안함. <-> 기존에는 single end-to-end framework 로 consistency 를 잘 반영해줄만한 feature representation 을 학습.

STEGO (Self-supervised Transformer with Energy-based Graph Optimization)

- **Unsupervised features 를 high-quality discrete semantic label 로 distill.**
- **Novel contrastive loss function**
  - that encourages features to form compact clusters while preserving their relationships across the corpora.
  - 즉, unsupervised visual features을 **novel contrastive loss**을 사용하여 semantic clusters 로 distill

- STEGO의 성능:
  - significant improvement on both the **CocoStuff (+14 mIoU)** and **Cityscapes (+9 mIoU)** semantic segmentation challenges.

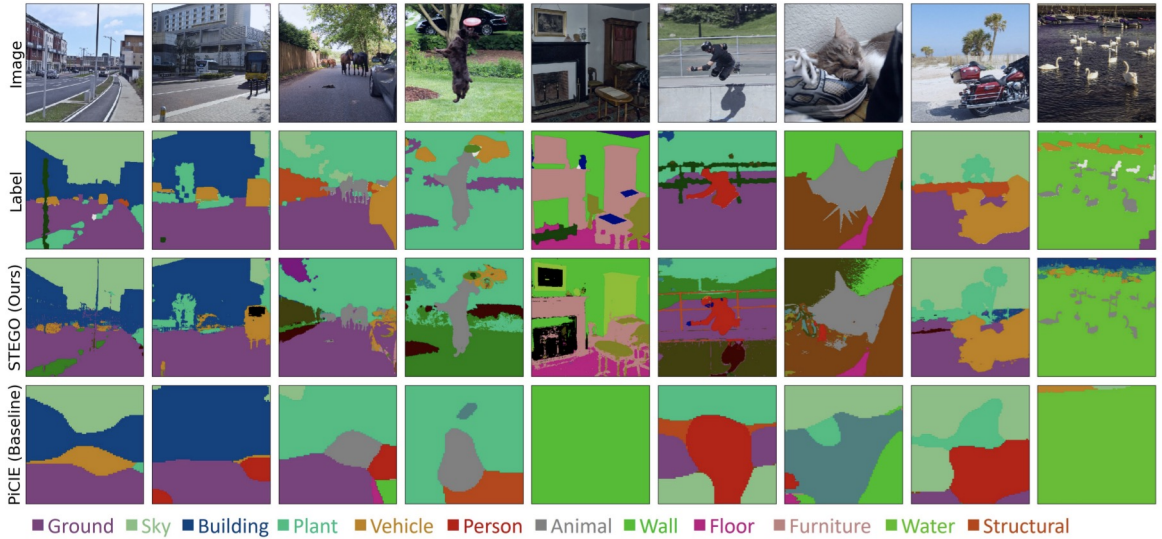


Figure 1: Unsupervised semantic segmentation predictions on the CocoStuff (Caesar et al., 2018) 27 class segmentation challenge. Our method, STEGO, does not use labels to discover and segment consistent objects. Unlike the prior state of the art, PiCIE (Cho et al., 2021), STEGO's predictions are consistent, detailed, and do not omit key objects.

- STEGO의 성능:
  - significant improvement on both the **CocoStuff (+14 mIoU)** and **Cityscapes (+9 mIoU)** semantic segmentation challenges.

Method

- **Feature correspondence**
  - **Feature correspondence 는 unsupervised segmentation 에서 quality learning signal potential 을 지님.**
  - Feature correspondences have the potential to be a quality learning signal for unsupervised segmentation.

- Figure 2: Feature correspondences from DINO (DINO as the feature extractor)
  - Source - within the image (Self Correspondence)
    - Feature correspondence 는 single 이미지에서 key aspect 로 사용할 수 있음.
  - Source - K-nearest neighbors image (KNN Correspondence)
    - Feature correspondence 는 유사한 이미지 (KNN 과 같은) 에서 key aspect 로 사용할 수 있음.

Figure 2: Feature correspondences from DINO. Correspondences between the source image (left) and the target images (middle and right) are plotted over the target images in the respective color of the source point (crosses in the left image). Feature correspondences can highlight key aspects of shared semantics within a single image (middle) and across similar images such as KNNs (right)

Method

- **STEGO architecture**
  - **Segmentation head 를 학습 하기 위해서 3가지 correspondence loss 를 사용함.**
    1) **(Self Corr.) between an image and itself**
    2) **(KNN Corr.) between an image and its K-Nearest Neighbors (KNNs)**
    3) **(Random Image Corr.) between an image and random other images**

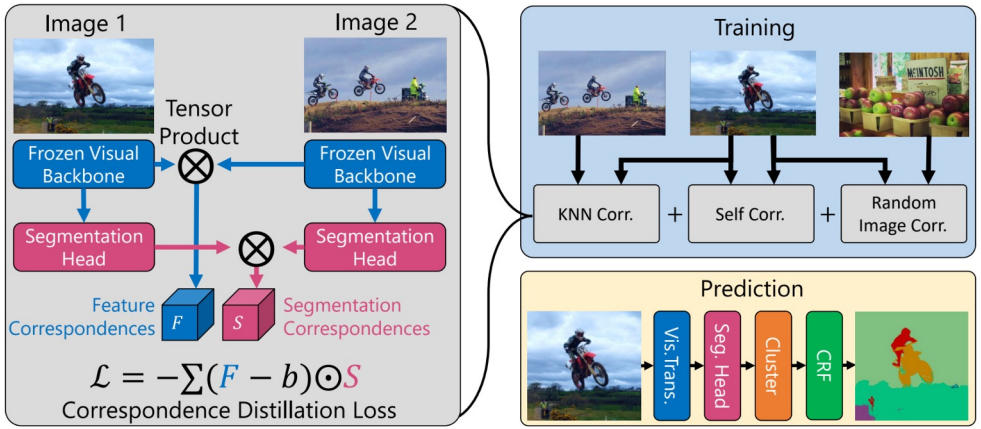


Figure 4: High-level overview of the STEGO architecture at train and prediction steps. Grey boxes represent three different instantiations of the main correspondence distillation loss which is used to train the segmentation head.

Self and KNN correspondence losses

- tend to provide positive, attractive, signal

Random image pairs

- tend to provide negative, repulsive, signal.

Segmentation head

- For predicting distilled features.
- A simple feed forward network with ReLU activations.
- 즉, backbone 을 re-train 하거나 fine-tune 하지 않으며, 이로써 학습을 효율적으로 할 수 있다.

$$\mathcal{L} = \lambda_{self}\mathcal{L}_{corr}(x, x, b_{self}) + \lambda_{knn}\mathcal{L}_{corr}(x, x^{knn}, b_{knn}) + \lambda_{rand}\mathcal{L}_{corr}(x, x^{rand}, b_{rand}) \quad (5)$$

UNSUPERVISED SEMANTIC SEGMENTATION BY DISTILLING FEATURE CORRESPONDENCES

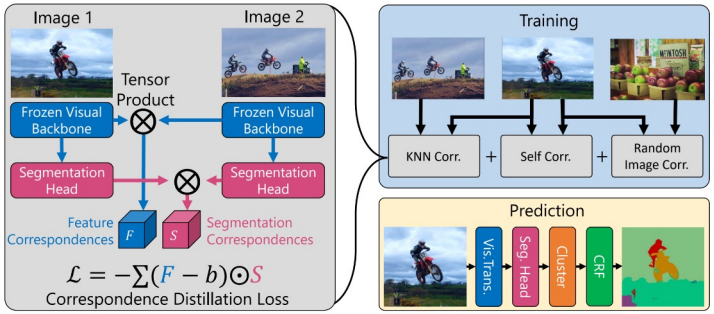


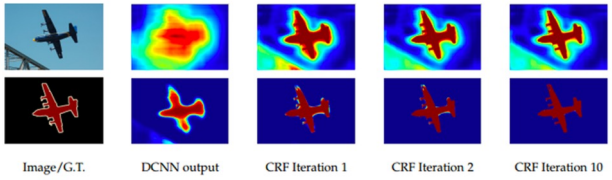
Figure 4: High-level overview of the STEGO architecture at train and prediction steps. Grey boxes represent three different instantiations of the main correspondence distillation loss which is used to train the segmentation head.

$$\mathcal{L} = \lambda_{self}\mathcal{L}_{corr}(x, x, b_{self}) + \lambda_{knn}\mathcal{L}_{corr}(x, x^{knn}, b_{knn}) + \lambda_{rand}\mathcal{L}_{corr}(x, x^{rand}, b_{rand}) \quad (5)$$

1. Backbone 을 사용해서 global image feature 추출.
2. Each training minibatch 는 1) random image x, 2) random nearest neighbors x_{knn}, 3) sample random images x_{rand} 로 구성되어 있음.
3. 감마, b : the balance of the learning signals, the ratio of positive to negative pressure.

+ CocoStuff 와 Cityscapes 데이터셋의 작은 object 를 잘 다루기 위해
  - KNN 학습 시, five-crop training images 를 사용함.
    - 이는 closer details 를 더 잘 볼 뿐만 아니라, KNN 의 퀄리티도 향상시킴.
  - 즉, global image embedding 을 각 crop 마다 계산하고, 이로써 finer details 성능을 향상시킴.

+ CRF refinement step
  - Clustering 이후에, spatial resolution 향상을 위하여 CRF 적용.



3.4 Relation to Potts Models and Energy-based Graph Optimization

$$E(\phi) := \sum_{v_i, v_j \in \mathcal{V}} w(v_i, v_j)\mu(\phi(v_i), \phi(v_j)) \qquad p(\phi|w, \mu) = \frac{\exp(-E(\phi))}{\int_\Phi \exp(-E(\phi'))d\phi'}$$

In general, sampling from this probability distribution is difficult because of the often-intractable normalization factor. However, it is easier to compute the maximum likelihood estimate (MLE), $\arg\max_{\phi \in \Phi} p(\phi|w, \mu)$. In particular, if Φ is a smoothly parameterized space of functions and ϕ and μ are differentiable functions, one can compute the MLE using stochastic gradient descent (SGD) with highly-optimized automatic differentiation frameworks (Paszke et al., 2019; Abadi et al., 2015). In Section A.8 of the supplement we prove that the finding the MLE of Equation 7 is equivalent to minimizing the loss of Equation 4 when $|V|$ is the set of pixels in our image training set, $\phi = \mathcal{S} \circ \mathcal{N}$, w is the cosine distance between features, and μ is cosine distance. Like STEGO, the CRF is also a Potts model, and we use this connection to re-purpose the STEGO loss function to create continuous, minibatch, and unsupervised variants of the CRF. We detail this exploration in Section A.9 of the Supplement.

Experiments

- evaluate: STEGO on the 27 mid-level classes of the CocoStuff class hierarchy and on the 27 classes of Cityscapes.
  the quality of the distilled segmentation features is through transfer learning effectiveness.
- resize: images to 320 pixels along the minor axis followed by a (320 × 320) center crops of each validation image.
- use: mean intersection over union (mIoU) and Accuracy for evaluation metrics.
  Hungarian matching algorithm to align our unlabeled clusters and the ground truth labels for evaluation and visualization purposes.

- Unsupervised segmentation 아키텍처 비교

Table 1: Comparison of unsupervised segmentation architectures on 27 class CocoStuff validation set. STEGO significantly outperforms prior art in both unsupervised clustering and linear-probe style metrics.

	Unsupervised		Linear Probe	
Model	**Accuracy**	**mIoU**	**Accuracy**	**mIoU**
ResNet50 (He et al., 2016)	24.6	8.9	41.3	10.2
MoCoV2 (Chen et al., 2020c)	25.2	10.4	44.4	13.2
DINO (Caron et al., 2021)	30.5	9.6	66.8	29.4
Deep Cluster (Caron et al., 2018)	19.9	-	-	-
SIFT (Lowe, 1999)	20.2	-	-	-
Doersch et al. (2015)	23.1	-	-	-
Isola et al. (2015)	24.3	-	-	-
AC (Ouali et al., 2020)	30.8	-	-	-
InMARS (Mirsadeghi et al., 2021)	31.0	-	-	-
IIC (Ji et al., 2019)	21.8	6.7	44.5	8.4
MDC (Cho et al., 2021)	32.2	9.8	48.6	13.3
PiCIE (Cho et al., 2021)	48.1	13.8	54.2	13.9
PiCIE + H (Cho et al., 2021)	50.0	14.4	54.8	14.8
STEGO (Ours)	**56.9**	**28.2**	**76.1**	**41.0**

Experiments

- 아키텍처 ablation study

Table 2: Architecture ablation study on the CocoStuff Dataset (27 Classes).

Arch.	0-Clamp	5-Crop	SC	CRF	Unsup. Acc.	Unsup. mIoU	Linear Probe Acc.	Linear Probe mIoU
MoCoV2	✓				48.4	20.8	70.7	26.5
ViT-S					34.2	7.3	54.9	15.6
ViT-S	✓				44.3	21.3	70.9	36.8
ViT-S	✓	✓			47.6	23.4	72.2	36.8
ViT-S	✓	✓	✓		47.7	24.0	72.9	38.4
ViT-S	✓	✓	✓	✓	48.3	24.5	74.4	38.3
ViT-B	✓	✓	✓		54.8	26.8	74.3	39.5
ViT-B	✓	✓	✓	✓	**56.9**	**28.2**	**76.1**	**41.0**

Table 3: Results on the Cityscapes Dataset (27 Classes). STEGO improves significantly over all baselines in both accuracy and mIoU.

Model	Unsup. Acc.	Unsup. mIoU
IIC (Ji et al., 2019)	47.9	6.4
MDC (Cho et al., 2021)	40.7	7.1
PiCIE (Cho et al., 2021)	65.5	12.3
STEGO (Ours)	**73.2**	**21.0**

- STEGO 성능

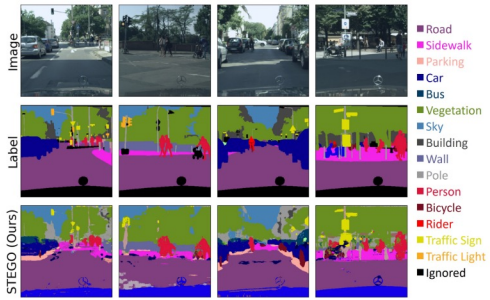


Figure 5: Comparison of ground truth labels (middle row) and cluster probe predictions for STEGO (bottom row) for images from the Cityscapes dataset.

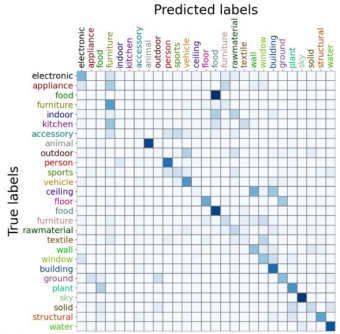


Figure 6: Confusion matrix of STEGO cluster probe predictions on CocoStuff. Classes after the "vehicle" class are "stuff" and classes before are "things". Rows are normalized to sum to 1.

STEGO의 대표적인 오류:

- CocoStuff의 "thing"에서 "food" 범주를 혼동하는 것
- CocoStuff의 "stuff"에서 "food" 범주를 혼동하는 것
- "ceiling"과 "wall"
- "indoor", "accessory", "rawmaterial" and "textile"

Challenges of evaluating unsupervised segmentation methods:
⇒ label ontologies can be arbitrary

Conclusion

- Modern **self-supervised visual backbones (DINO) can be refined to yield state of the art unsupervised semantic segmentation method.**
- Entirely unsupervised, learning signal by introducing **a novel contrastive loss that ‘distills’ the correspondences between features.**
- **STEGO yields a significant improvement over the prior state of the art, on both the CocoStuff (+14 mIoU) and Cityscapes (+9 mIoU) semantic segmentation challenges.**

End of the Document