Vision AI

2021 arXiv Trends

2021-10

hutom

no.	Paper Title	research group
1	Multi-Task Self-Training for Learning General Representations	Google Research, Brain team
2	Panoptic SegFormer	NVIDIA
3	An End-to-End Transformer Model for 3D Object Detection	Facebook AI Research
4	torch.manual seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision	

no.	Paper Title	research group
5	UNETR: Transformers for 3D Medical Image Segmentation	NVIDIA
6	Object DGCNN: 3D Object Detection using Dynamic Graphs	MIT
7	Noisy Labels Can Induce Good Representations	
8	Ego4D: Around the World in 3,000 Hours of Egocentric Video	

Multi-Task Self-Training for Learning General Representations

Golnaz Ghiasi,* Barret Zoph,* Ekin D. Cubuk,* Quoc V. Le, Tsung-Yi Lin
Google Research, Brain Team
{golnazg,barretzoph,cubuk,qvl,tsungyi}@google.com

https://arxiv.org/pdf/2108.11353v1.pdf

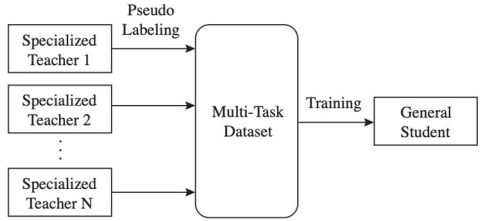


Figure 1. **An overview of Multi-Task Self-Training (MuST).** Specialized Teacher represents a supervised model trained on a single task and dataset (*e.g.*, classification model trained on ImageNet). Specialized Teacher models are trained independently on their own tasks and datasets. They then generate pseudo labels on a shared dataset. Finally, a single General Student model is trained jointly using the pseudo (and supervised) labels on the shared dataset.

- **Multi-Task learning**
  - 학습하려는 여러 종류의 task가 동일한 domain 에 존재하는 경우
    - e.g. 얼굴 데이터로부터 얼굴 인식, 표정 인식, 성별 분류, 포즈 분류 등의 task를 모두 하나의 모델로 처리하고자 하는 경우
  - Computer vision 의 multi-task 에서 잘 작동하는 single general model 학습은 challenging

- **Multi-Task Self-Training (MuST)**
  - Independent Specialized Teacher model 의 knowledge 를 single general student model 학습하는 것에 이용함
  - 3 steps
    1. 각각의 labeled datasets 에 대해서 specialized teacher model 학습 (각각의 task)
    2. specialized teacher model 를 사용하여 unlabeled dataset 을 label 함 (**multi-task pseudo labeled dataset 생성**)
    3. 각각의 (서로 다른 task로 학습된) teacher model 로부터 추론한 pseudo lebel 를 포함하고 있는 dataset (=multi-task pseudo labeled dataset) 으로 student model 을 multi-task learning 수행함

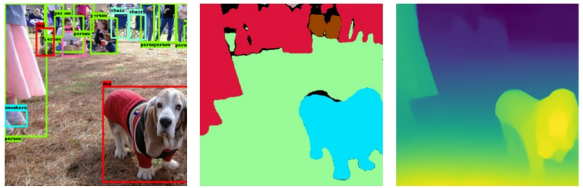


Figure 2. **Examples of pseudo labels on ImageNet. Left:** bounding boxes labeled with an Objects365 teacher model. **Middle:** semantic segmentation labeled with a COCO teacher model. **Right:** depth labeled with a MiDaS teacher model.

- **Specialized Teacher Models**
  - Computer vision task (Pseudo labeling)
    - Detection
      - Objects365, pseudo box labels 생성을 위해, 0.5 threshold score
    - Segmentation
      - COCO, semantic segmentation masks 생성을 위해, 0.5 threshold score
    - Classification
      - ImageNet, soft labels (모든 클래스에 대한 probability distribution 포함)
    - Depth
      - pre-trained checkpoint from the open-source repository, 추가 작업 없이, predicted depth 사용
  - (Fig.2) 각각의 teacher 모델에서 label 한 pseudo label
  - **pseudo labeling 을 통해 specialized teacher model 에서 unlabeled, partially labeled dataset 에 knowledge 전이 (transfer)**

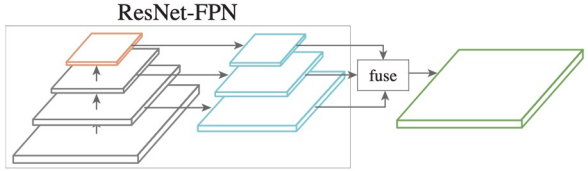


Figure 3. **The ResNet-FPN backbone architecture for multi-task learning. Orange:** the top-level features for classification. **Cyan:** multi-scale features for box detection and instance segmentation. **Green:** the high resolution features for pixel-wise tasks (*e.g.*, segmentation, depth, and surface normal estimation.)

- **Multi-Task Student Model**
  - multi-task 에 대해 대부분의 파라미터 공유하는 모델 아키텍처
  - backbone 모델은 multi-task (4개)에 대해 동일한 feature representation 공유 (Each category of task shared the same feature representations in the backbone model)
  - overview of architecture
    - (Orange) classification : C5 feature map
    - (Cyan) detect : {P3, P4, P5, P6, P7} feature pyramid
    - (Green) pixel-wise prediction (segmentation, depth) : fuse {P3, P4, P5, P6, P7} → into P2 feature map
      - fuse operation : 단순히 모든 feature map 을 level 2 로 rescale 후, sum. (새로운 파라미터 사용하지 않음)

- model distillation, noisy student 와는 다르게, teacher 과 student 아키텍처를 Fig.3 로 동일하게 둠. (same model capacity and data augmentation)

- **Learning from multiple teachers**
  - 모든 이미지가 모든 task 에 대해 supervision.
    - e.g. ImageNet 학습 시, classification (supervised label) + detection, segmentation, depth (pseudo labels)
  - loss func
    - 기존 multi-task learning 에 사용되는 loss func

research area in multi-task learning [26, 6, 7, 63]. The loss of multi-task learning is the weighted sum of the losses from all tasks $L = \sum_i w_i L_i$. The weight w_i decides the loss contribution for the task i. In ImageNet experiments, we adopt

- **Experiments**

Training Datasets			Evaluation Datasets		
Name	Task	Num Images	Name	Task	Num Images
ImageNet [47]	Classification	1.2M	CIFAR-100 [31]	Classification	50k
Objects365 [49]	Detection	600k	Pascal [13]	Detection	16.5k
COCO [37]	Segmentation	118k	Pascal [13]	Segmentation	1.5k
MiDaS [44]	Depth	1.9M	NYU V2 [50]	Depth	47k
JFT [51]	Classification	300M	ADE [66]	Segmentation	20k
			DIODE [54]	Surface Normal	17k

Table 1. **Datasets using for MuST and for downstream fine-tuning evaluation**.

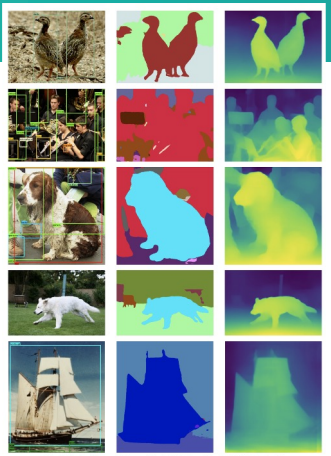


Figure 6. The visualization of inference on ImageNet dataset made by single MuST student model.

Settings		Transfer Learning Performance					
Method	Epochs	CIFAR-100 Classification	Pascal Detection	Pascal Segm.	NYU Depth	ADE Segm.	DIODE Normal
Self-supervised (SimCLR [5])	800	**87.1**	83.3	72.2	83.7	41.0	**52.8**
ImageNet Supervised	90	85.4	79.3	70.6	81.0	39.8	48.9
+ Multi-task Pseudo Labels	90	86.3 (+0.9)	**85.1** (+5.8)	**80.6** (+10.0)	**87.8** (+6.8)	**43.5** (+3.7)	**52.7** (+3.8)

Table 2. **Multi-Task Self-Training (MuST) outperforms supervised and self-supervised representations on ImageNet**. We compare MuST to state-of-the-art self-supervised and supervised learning using the same pre-training dataset (ImageNet). MuST learns more general features and achieves the best performance on 4/6 downstream fine-tuning tasks. The performance differences show the impact of different training objectives.

Settings	Transfer Learning Performance					
Method	CIFAR-100 Classification	Pascal Detection	Pascal Segm.	NYU Depth	ADE Segm.	DIODE Normal
ImageNet Supervised	85.4	79.3	70.6	81.0	39.8	48.9
+ Depth Pseudo Labels	84.4(-1.0)	79.3(+0.0)	71.0(+0.4)	86.0(+5.0)	39.5(-0.3)	51.3(+2.4)
+ Depth / Segm. Pseudo Labels	85.3(-0.1)	81.6(+2.3)	78.6(+8.0)	87.2(+6.2)	41.5(+1.7)	52.4(+3.5)
+ Depth / Segm. / Detection Pseudo Labels	**86.3**(+0.9)	**85.1**(+5.8)	**80.6**(+10.0)	**87.8**(+6.8)	**43.5**(+3.7)	**52.7**(+3.8)

Table 3. **Multi-Task Self-Training (MuST) benefits from increasing the number of different pseudo label tasks.** We add depth, segmentation, and detection pseudo labels in addition to supervised ImageNet classification labels and test the representational quality. The results reveal that adding pseudo labels from more tasks leads to more general pre-trained models. All models are trained for 90 epochs on ImageNet.

Settings		Performance		Transfer Learning Performance					
Task	Train Dataset	Obj365 Detection	COCO Segm.	CIFAR-100 Classification	Pascal Detection	Pascal Segm.	NYU Depth	ADE Segm.	DIODE Normal
		Teacher Model							
Detection	Objects365	26.1	—	84.0	87.6	78.8	90.1	46.0	55.6
Segmentation	COCO	—	53.8	80.8	82.2	80.2	86.6	42.8	51.0
		Student Model							
Detection	ImageNet	20.6	—	83.2	86.0	78.5	88.5	44.7	55.2
Detection	JFT	20.7	—	85.2	87.7	79.5	89.6	45.4	55.0
Segmentation	ImageNet	—	55.5	82.3	80.5	79.2	86.3	41.8	51.2
Segmentation	JFT	—	49.0	83.1	82.8	78.2	86.6	41.9	51.6

Table 4. **Models trained on supervised data or pseudo labeled data have similar transfer learning performance.** Results comparing how representations transfer if they are trained on supervised data or on pseudo labels that are generated by the supervised model. Pseudo labels effectively compress the knowledge in a supervised dataset. The performance of student models increases with the size of the unlabeled dataset. As the unlabeled dataset size increased, the performance of student model increases. This reveals the scalability of MuST. All student models are trained for the same training iterations (90 ImageNet epochs and 0.36 JFT epochs).

Settings	Transfer Learning Performance					
Method	CIFAR-100 Classification	Pascal Detection	Pascal Segm.	NYU Depth	ADE Segm.	DIODE Normal
Supervised Multi-Task	85.3	85.1	82.1	87.6	43.9	53.4
Supervised Multi-Task + Pseudo Labels	**86.3** (+1.1)	**86.2** (+1.1)	**82.3** (+0.2)	**88.2** (+0.6)	**45.4** (+1.5)	**54.7** (+1.3)

Table 5. **Comparing Multi-Task Training versus Multi-Task Self-Training.** We compare MuST against a baseline of doing supervised multi-task training on the union of all teacher datasets. We use three datasets: ImageNet, COCO and Objects365. Supervised model is jointly trained on the supervised labels of these three datasets. MuST trains jointly on all three supervised and pseudo labels generated by the teacher models. The transfer learning performance gets strong improvements by incorporating pseudo labels into every image.

E

Panoptic SegFormer

Zhiqi Li[1], Wenhai Wang[1], Enze Xie[2], Zhiding Yu[3],
Anima Anandkumar[3,4], Jose M. Alvarez[3], Tong Lu[1], Ping Luo[2]

[1]Nanjing University [2]The University of Hong Kong [3]NVIDIA [4]Caltech

https://arxiv.org/pdf/2109.03814.pdf

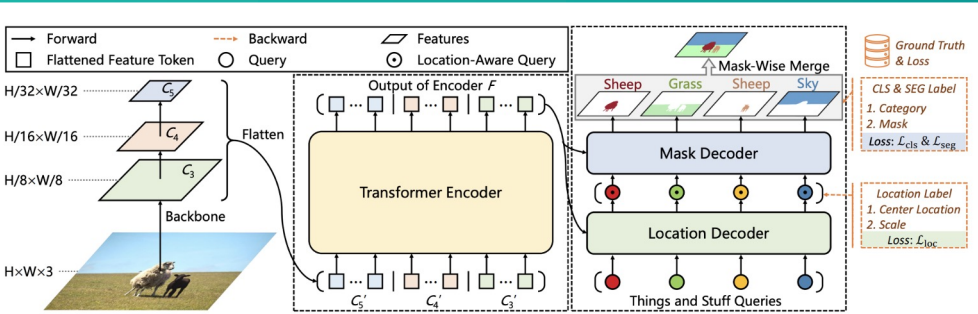


Figure 2: **Overview of Panoptic SegFormer.** Panoptic SegFormer is composed of backbone, encoder, and decoder. The backbone and the encoder output and refines multi-scale features. Inputs of the decoder are N queries and the multi-scale features. The decoder consists of two sub-decoders: location decoder and mask decoder, where location decoder aims to learn reference points of queries, and mask decoder predicts the final category and mask. Details of the decoder will be introduced below. We use a mask-wise merging method instead of the commonly used pixel-wise argmax method to perform inference.

- **Panoptic Segmentation**
  - semantic segmentation + instance segmentation
- **Panoptic SegFormer**
  - **Transformers 를 사용한 end-to-end panoptic segmentation 을 위한 general framework**
  - 기존 panoptic segmentation 아키텍처인 Deformable DETR 의 성능 능가 (실험 성능 비교)
  - workflow
    - backbone
      - input 이미지에 각각 ⅛, 1/16, 1/32 의 일정한 resolution을 주어 C3, C4, C5 feature map 생성
      - FC layer 통해서 3개의 feature map 을 256 channel 로 projection 시킨 후
      - flatten into feature tokens C'3, C'4, C'5 => 이 때, C'3, C'4, C'5 shape (H/8*W/8*256, H/16*W/16*256, H/32*W/32*256)
    - encoder
      - input: concat (C'3 + C'4 + C'5) feature token
      - output: refined features of size (L1+L2+L3)*256
    - decoder
      - 랜덤하게 N개 query 선택 (things, stuff)
      - query 에 location clues (center location , scale) 추가
      - mask-wise strategy 적용 → predict mask 를 panoptic segmentation result 에 결합.

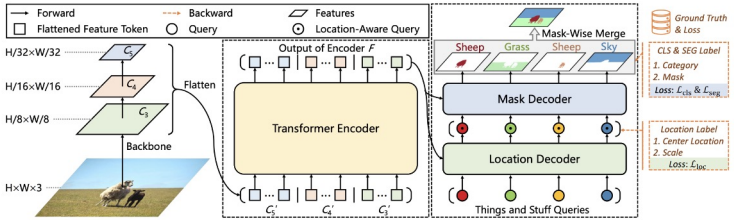


Figure 2: **Overview of Panoptic SegFormer.** Panoptic SegFormer is composed of backbone, encoder, and decoder. The backbone and the encoder output and refines multi-scale features. Inputs of the decoder are N queries and the multi-scale features. The decoder consists of two sub-decoders: location decoder and mask decoder, where location decoder aims to learn reference points of queries, and mask decoder predicts the final category and mask. Details of the decoder will be introduced below. We use a mask-wise merging method instead of the commonly used pixel-wise argmax method to perform inference.

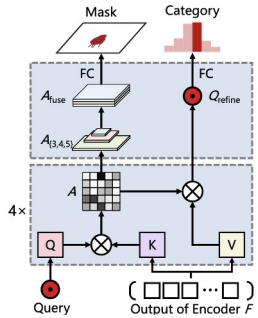


Figure 3: **Architecture of mask decoder.** The attention maps A are the product of query Q and key K^T. We split and reshape multi-scale attention maps to $A_{\{3,4,5\}}$,then we upsample and cat these features to A_{fuse}. The mask is generated through attention maps with one 1×1 conv layer. The category label is predicted from the refined query Q_{refine} with one linear projection layer.

$$\mathcal{L} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{seg}}\mathcal{L}_{\text{seg}} + \lambda_{\text{loc}}\mathcal{L}_{\text{loc}},$$

$$\mathcal{L}_{\text{loc}} = \sum_{i}^{N} \mathbb{1}_{\{y_i \neq \varnothing\}} \left(\mathcal{L}_1(f_c(m_i), \hat{u}_{\sigma(i)}) + \mathcal{L}_1(f_s(m_i), \hat{v}_{\sigma(i)})\right),$$

where λ_{cls}, λ_{seg}, and λ_{loc} are the weights to balance three losses. $\mathcal{L}_{\text{cls}}$ is the classification loss that is implemented by Focal loss [28], and $\mathcal{L}_{\text{seg}}$ is the segmentation loss implemented by Dice loss [38]. $\mathcal{L}_{\text{loc}}$ is the location loss as formulated in Eqn. 4:

- workflow
  - **encoder ⇒ deformable attention layer**
    - input: concat (C'3 + C'4 + C'5) feature token
    - output: refined features of size (L1+L2+L3)*256
  - **decoder**
    - query 에 location clues (center location , scale) 추가 (location-aware query)
    - mask decoder
      - classification
        - 4개의 디코더 레이어를 통과한 후, fetch attention map A -> refine

$$A \in \mathbb{R}^{N\times h\times(L_1+L_2+L_3)} \longrightarrow Q_{\text{refine}} \in \mathbb{R}^{N\times 256} \text{ from the last decoder layer,}$$

$Q_{\text{refine}} \in \mathbb{R}^{N\times 256}$ from the last decoder layer, where N is the query number, h is the head number of the multi-head attention layer, and $L_1 + L_2 + L_3$ is the length of feature tokens F.

      - object mask
        - attention map A 를 C3, C4, C5 와 동일한 spatial resolution 을 가진 A3, A4, A5 로 split -> H/8*W/8 resolution 으로 attention map upsample

$$(A_3, A_4, A_5) = \text{Split}(A), \quad A_i \in \mathbb{R}^{\frac{H}{2^{i+2}}\times\frac{W}{2^{i+2}}\times h}, \qquad (1) \longrightarrow A_{\text{fuse}} = \text{Concat}(A_1, \text{Up}_{\times 2}(A_2), \text{Up}_{\times 4}(A_3)). \qquad (2)$$

        - α
        - fused attention map A{fuse} 를 기반으로 binary mask predict (through 1*1 convolution)

Method	Backbone	Epochs	PQ	PQ^{Th}	PQ^{St}	#Param	FLOPs
Panoptic FPN [2]	R50-FPN [24, 39]	36	41.5	48.5	31.1	-	-
SOLOv2 [12]	R50-FPN	36	42.1	49.6	30.7	-	-
DETR [15]	R50	$\sim 150 + 25$	43.4	48.2	36.3	42.8M	137G
Panoptic FCN [13]	R50-FPN	36	43.6	49.3	35.0	37.0M	244G
K-Net [14]	R50-FPN	36	45.1	50.3	37.3	-	-
MaskFormer [17]	R50	300	46.5	51.0	39.8	45.0M	181G
DETR [15]	R101	$\sim 150 + 25$	45.1	50.5	37.0	61.8M	157G
Max-Deeplab-S [16]	Max-S	54	48.4	53.0	41.5	61.9M	162G
MaskFormer [17]	R101	300	47.6	52.5	40.3	64.0M	248G
Max-Deeplab-L [16]	Max-L	54	51.1	57.0	42.2	451.0M	1846G
MaskFormer [17]	Swin-L† [20]	300	52.7	58.5	44.0	212.0M	792G
Panoptic SegFormer	R50	12	46.4	52.6	37.0	47.0M	246G
Panoptic SegFormer	R50	50	50.0	56.1	40.8	47.0M	246G
Panoptic SegFormer	R101	50	50.4	56.3	41.6	65.9M	322G
Panoptic SegFormer	PVTv2-B0 [40]	50	49.6	55.5	40.6	22.2M	156G
Panoptic SegFormer	PVTv2-B2 [40]	50	52.6	58.7	43.3	41.6M	219G
Panoptic SegFormer	PVTv2-B5 [40]	50	54.1	60.4	44.6	100.9M	391G

Table 1: **Experiments on COCO val set.** Panotic SegFormer achieves 50.0% PQ on COCO `val` with ResNet-50 as backbone, surpasses previous methods such as DETR [15] and Panoptic FCN [17] over 6.6% PQ and 6.4% PQ respectively. Under training for 12 epochs, Panoptic SegFormer can achieve 46.4% PQ, which is comparable with 46.5% PQ of MaskFormer [17] that training for 300 epochs. † notes that backbones are pre-trained on ImageNet-22K.

Method	Backbone	Epochs	PQ	PQ^{Th}	PQ^{St}	#Param	FLOPs
Panoptic FPN [2]	R101-FPN	36	43.5	50.8	32.5	-	-
DETR [15]	R101	$\sim 150 + 25$	46.0	-	-	61.8M	157G
Panoptic FCN [13]	R101-FPN	36	45.5	51.4	36.4	56.0M	310G
K-Net [14]	R101-FPN	36	47.0	52.8	38.2	-	-
Max-Deeplab-S [16]	Max-S [16]	54	49.0	54.0	41.6	61.9M	162G
K-net [14]	Swin-L†	36	52.1	58.2	42.8	-	-
Max-Deeplab-L [16]	Max-L [16]	54	51.3	57.2	42.4	451.0M	1846G
Innovation [22]	ensemble	-	53.5	61.8	41.1	-	-
Panoptic SegFormer	R50	50	50.0	56.2	40.8	47.0M	246G
Panoptic SegFormer	R101	50	50.9	57.1	41.4	65.9M	322G
Panoptic SegFormer	PVTv2-B5 [40]	50	54.4	61.1	44.3	100.9M	391G

Table 2: **Experiments on COCO test-dev set.** With PVTv2-B5 [40] as backbone, Panoptic SegFormer achieves 54.4% PQ on COCO `test-dev`, surpassed previous SOTA methods Max-Deeplabe-L [16] and competition-level methods Innovation [22] over 3.1% PQ and 0.9% PQ respectively with fewer parameters and computation cost.

Method	Backbone	Epochs	AP^{seg}	AP^{seg}_S	AP^{seg}_M	AP^{seg}_L
Mask R-CNN [29]	R50-FPN	36	37.5	21.1	39.6	48.3
SOLOv2 [12]	R50-FPN	36	38.8	16.5	41.7	56.2
SOLQ (300 queries) [31]	R50	50	39.7	21.5	42.5	53.1
HTC [41]	R50-FPN	36	40.1	23.3	42.1	52.0
QueryInst (300 queries) [32]	R50-FPN	36	40.6	**23.4**	42.5	52.8
Panoptic SegFormer (300 queries)	R50	50	**41.7**	21.9	**45.3**	**56.3**

Table 3: **Instance segmentation experiments on COCO `test-dev` set.** When training with things only, Panoptic SegFormer can perform instance segmentation. With ResNet-50 as backbone, Panoptic SegFormer achieves 41.7 mask AP on COCO `test-dev`, which is 1.6 AP higher than HTC [41].

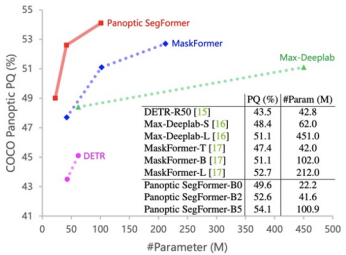


Figure 1: **Comparison to the prior arts in panoptic segmentation methods on the COCO val2017 split.** Under comparable number of parameters, Panoptic SegFormer models outperform the other counterparts among different models. Panoptic SegFormer (PVTv2-B5) achieves a new state-of-the-art 54.1%PQ, outperforming the previous best method MaskFormer by 1.4% with significantly fewer parameters.

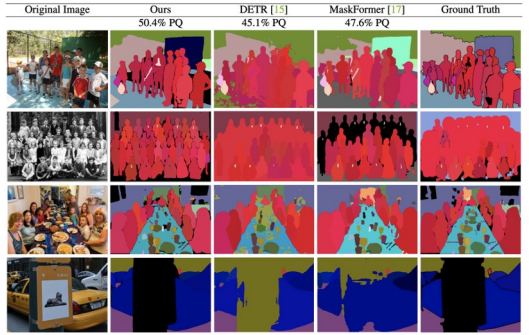


Figure 4: **Comparing visualization results of Panoptic SegFormer with other methods on the COCO `val` set.** For a fair comparison, all results are generated with ResNet-101 [24] as the backbone. The second and fourth row results show that our method still performs well in highly crowded or occluded scenes. Benefits from our mask-wise inference strategy, our results have few artifacts, which often appear in the results of DETR [15] (*e.g.*, dining table of the third row).

- Experiments & Results
  - COCO 2017 datasets
  - ResNet-50 backbone 으로, COCO test-dev split 에서 PQ 50.0% (SoTA)
  - PVTv2-B5 backbone 으로, single scale input COCO val 에서 54.1% PQ, test-dev split 에서 54.4% PQ

An End-to-End Transformer Model for 3D Object Detection

Ishan Misra Rohit Girdhar Armand Joulin
Facebook AI Research
https://facebookresearch.github.io/3detr

https://arxiv.org/pdf/2109.08141v1.pdf

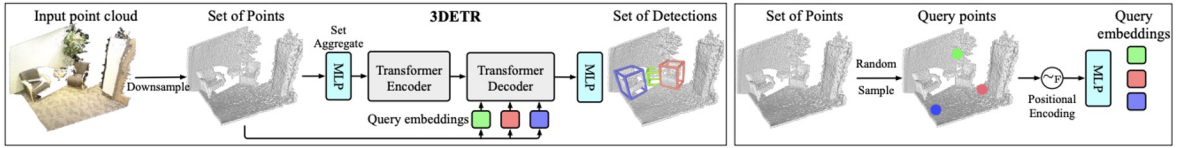


Figure 2: Approach. *(Left)* 3DETR is an end-to-end trainable Transformer that takes a set of 3D points (point cloud) as input and outputs a set of 3D bounding boxes. The Transformer encoder produces a set of per-point features using multiple layers of self-attention. The point features and a set of 'query' embeddings are input to the Transformer decoder that produces a set of boxes. We match the predicted boxes to the ground truth and optimize a set loss. Our model does not use color information (used for visualization only). *(Right)* We randomly sample a set of 'query' points that are embedded and then converted into bounding box predictions by the decoder.

- **3DETR:** 3d object DEtection TRansformer
  - 기존 DETR + 3D point clouds
    - DETR : End-to-end Object detection with Transformers
  - Panoptic Segmentation (semantic + instance segmentation)

- 3DETR: Encoder-decoder Transformer
  - **(input) 3D point cloud - (output) positions of objects in the form of 3D bbox**
  - input point cloud (N dimension)
    - unordered set of N points (각각의 point 는 XYZ 3-dim 할당)
    - points의 수가 아주 많기 때문에 **⇒ set-aggregation downsampling operation 을 통해 point downsampling 및 N' dimensional feature 로 projection.**
  - Encoder
    - (input) N' dimension feature (N' features of d=256 dimensions using an MLP) ⇒ (output) N' features of d=256 dimension
    - transformer encoder 에서 3D data에 대한 추가적인 modifications X
  - Decoder
    - (input) encoder output (N' point features) and a set of B query embeddings
    - 특별한 순서 없이 (no particular ordering) set of boxed prediction
    - parallel decoder composed of transformer blocks

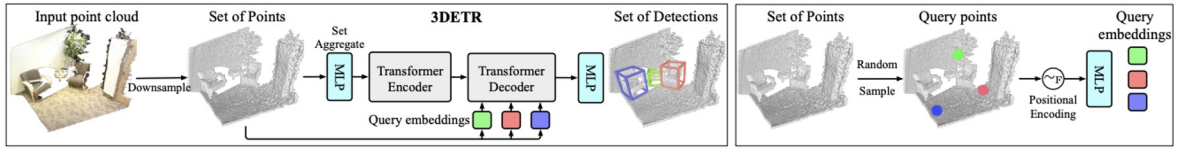


Figure 2: Approach. *(Left)* 3DETR is an end-to-end trainable Transformer that takes a set of 3D points (point cloud) as input and outputs a set of 3D bounding boxes. The Transformer encoder produces a set of per-point features using multiple layers of self-attention. The point features and a set of 'query' embeddings are input to the Transformer decoder that produces a set of boxes. We match the predicted boxes to the ground truth and optimize a set loss. Our model does not use color information (used for visualization only). *(Right)* We randomly sample a set of 'query' points that are embedded and then converted into bounding box predictions by the decoder.

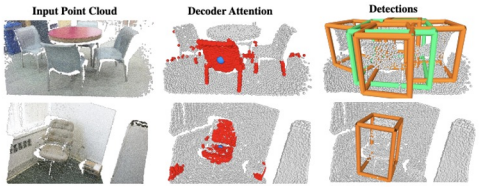


Figure 1: 3DETR. We train an end-to-end Transformer model for 3D object detection on point clouds. Our model has a Transformer encoder for feature encoding and a Transformer decoder for predicting boxes. For an unseen input, we compute the self-attention from the reference point (blue dot) to all points in the scene and display the points with the highest attention values in red. The decoder attention groups points within an instance which presumably makes it easier to predict bounding boxes.

- **Bounding box parametrization and prediction**
  - encoder-decoder 아키텍처를 통해, B features 생성 → MLP 으로 bbox predict
  - 3D bbox attributes
    - location, size, orientation, the class of the object contained in it.

- **Set Matching and Loss Function**
  - predicted 3D bounding box {b_hat} 과 gt bounding box {b} 매핑
  - Bipartite Matching

Bipartite Matching. We define a matching cost for a pair of boxes, predicted box $\hat{\mathbf{b}}$ and ground truth box $\mathbf{b}$, using a geometric and a semantic term.

$$C_{\text{match}}(\hat{\mathbf{b}}, \mathbf{b}) = \underbrace{-\lambda_1 \text{GIoU}(\hat{\mathbf{b}}, \mathbf{b}) + \lambda_2 \|\hat{\mathbf{c}} - \mathbf{c}\|_1}_{\text{geometric}} \underbrace{- \lambda_3 \hat{\mathbf{s}}[s_{\text{gt}}] + \lambda_4 (1 - \hat{\mathbf{s}}[s_{\text{bg}}])}_{\text{semantic}} \quad (1)$$

$$\mathcal{L}_{\text{3DETR}} = \lambda_c \|\hat{\mathbf{c}} - \mathbf{c}\|_1 + \lambda_d \|\hat{\mathbf{d}} - \mathbf{d}\|_1 + \lambda_{ar} \|\hat{\mathbf{a}}_r - \mathbf{a}_r\|_{\text{huber}} - \lambda_{ac} \mathbf{a}_c^\intercal \log \hat{\mathbf{a}}_c - \lambda_s \mathbf{s}_c^\intercal \log \hat{\mathbf{s}}_c \quad (2)$$

- **Experiments**

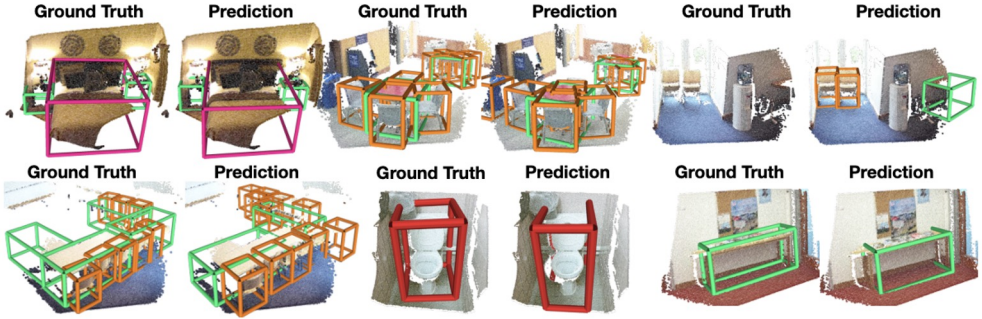


Figure 3: Qualitative Results using 3DETR. Detection results for scenes from the val set of the SUN RGB-D dataset. 3DETR does not use color information (used only for visualization) and predicts boxes from point clouds. 3DETR can detect objects even with single-view depth scans and predicts amodal boxes *e.g.*, the full extent of the bed (top left) including objects missing in the ground truth (top right).

Method	ScanNetV2		SUN RGB-D	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
BoxNet† [42]	49.0	21.1	52.4	25.1
3DETR	62.7	37.5	58.0	30.3
VoteNet† [42]	60.4	37.5	58.3	33.4
3DETR-m	65.0	47.0	59.1	32.7
H3DNet [89]	67.2	48.1	60.1	39.0

Table 1: Evaluating 3DETR on 3D detection. We compare 3DETR with BoxNet and VoteNet methods and denote by † our improved implementation of these baselines. 3DETR achieves comparable or better performance to these improved baselines despite having fewer hand-coded 3D or detection specific decisions. We report state-of-the-art performance from [89] that improves VoteNet by using 3D primitives. Detailed state-of-the-art comparison in Appendix B.

Method	Encoder	Decoder	Loss	ScanNetV2		SUN RGB-D	
				AP_{25}	AP_{50}	AP_{25}	AP_{50}
3DETR	Tx.	Tx.	Set	62.7	37.5	58.0	30.3
	PN++	Tx.	Set	61.4	34.7	56.8	26.9

PN++: PointNet++ [45], Tx.: Transformer, Set loss § 3.4

Table 2: 3DETR with different encoders. We vary the encoder used in 3DETR and observe that the performance is unchanged or slightly worse when moving to a PointNet++ encoder. This suggests that the decoder design and the loss function in 3DETR are compatible with prior 3D specific encoders.

#	Method	Encoder	Decoder	Loss	ScanNetV2		SUN RGB-D	
					AP_{25}	AP_{50}	AP_{25}	AP_{50}
Comparing different decoders								
1	3DETR	Tx.	Tx.	Set	62.7	37.5	58.0	30.3
2		Tx.	Box	Box	31.0	10.2	36.4	14.4
3		Tx.	Vote	Vote	46.1	23.4	47.5	24.9
Comparing different losses								
4		Tx.	Tx.	Box	49.6	20.5	49.5	21.1
5		Tx.	Tx.	Vote	54.0	31.9	53.4	28.3

Tx.: Transformer, Vote/Box loss [42], Set loss § 3.4

Table 3: 3DETR with different decoders and losses. We vary the decoder and losses used with our transformer encoder. As the Box and Vote decoders are only compatible with their losses, we vary the loss function while using them. The Vote loss is compatible with our Transformer encoder-decoder, however a simpler set loss performs the best.

`torch.manual_seed(3407)` is all you need: On the influence of random seeds in deep learning architectures for computer vision

David Picard DAVID.PICARD@ENPC.FR
LIGM, École des Ponts, 77455 Marnes la vallée, France

https://arxiv.org/pdf/2109.08203.pdf

실험 No.	Count of Seeds	Dataset	Model	Training
1	500	CIFAR 10	Resnet 9	Scratch
2	10000	CIFAR 10	Resnet 9	Scratch
3	50	ImageNet	Supervised Resnet 50 SSL Resnet 50 SSL ViT	Pretrain

- **random seed selection 이 accuracy 에 미치는 영향 연구**
  - top hype 상위권
  - computer vision 에서 인기있는 모델에 많은 양의 seed 를 scratch 학습 (CIFAR 10 = 500개, 10000개, imagenet = 50개)

1. What is the distribution of scores with respect to the choice of seed? — *1. 시드선택에 대한 score 분포?*
2. Are there *black swans*, *i.e.*, seeds that produce radically different results? — *2. 근본적으로 다른결과를 내는 Balck Swan 존재?*
3. Does pretraining on larger datasets mitigate variability induced by the choice of seed? — *3. 대규모 dataset 에 대한 pretaraining 이 시드 선택에 의해 가변성을 완화시키는가?*

- **연구 limitation**
  - 한정된 실험 및 dataset
- **Variational Inference ⇒ random seed?**

실험 No.	Count of Seeds	Dataset	Model	Training
1	500	CIFAR 10	Resnet 9	Scratch
2	10000	CIFAR 10	Resnet 9	Scratch

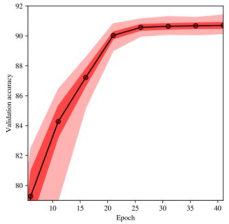


Figure 1: Validation accuracy variation on CIFAR 10 for the Resnet9 architecture against number of training epochs. he solid line represents the mean over 500 seeds, the dark red area corresponds to one standard deviation and the light red corresponds to the maximum and minimum values.

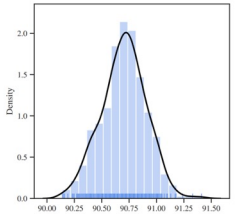


Figure 2: Histogram and density plot of the final validation accuracy on CIFAR 10 for the Resnet9 architecture over 500 seeds. Each dash at the bottom corresponds to one run.

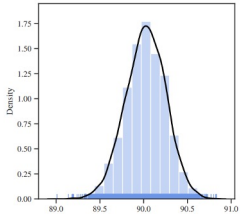


Figure 3: Histogram and density plot of the final validation accuracy on CIFAR 10 for the Resnet9 architecture over 10^4 seeds. Each dash at the bottom corresponds to one run.

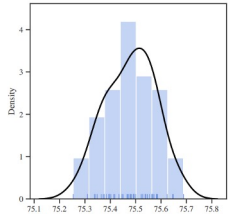


Figure 4: Histogram and density plot of the final validation accuracy on Imagenet for a pretrained ResNet50. Each dash at the bottom corresponds to one run.

Training mode	Accuracy mean ± std	Minimum accuracy	Maximum accuracy
long	90.70 ± 0.20	90.14	91.41
short	90.02 ± 0.23	89.01	90.83

Table 1: Results on cifar 10 for the long and the short training setups.

- 25 epoch 이후, 모델은 기본적으로 수렴 (fig.1)
- 수렴되기 전 차이가 좁혀지지만, [90.5-91.0] 0.5% 효과 차이 (fig.2, table 1 (1))

- 10000개의 random seed 선택 (fig.3, table 1 (1))
  [89.01-90.83] 1.82 % 효과 차이 (short 보다 성능 좋음,, 당연..?)

실험 No.	Count of Seeds	Dataset	Model	Training
3	50	ImageNet	Supervised Resnet 50 SSL Resnet 50 SSL ViT	Pretrain

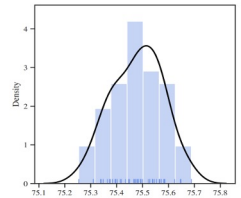


Figure 4: Histogram and density plot of the final validation accuracy on Imagenet for a pretrained ResNet50. Each dash at the bottom corresponds to one run.

Pretrained Resnet50

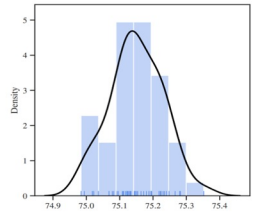


Figure 5: Histogram and density plot of the final validation accuracy on Imagenet for a self-supervised pretrained ResNet50. Each dash at the bottom corresponds to one run.

SSL Resnet50

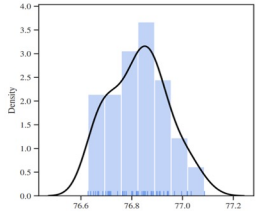


Figure 6: Histogram and density plot of the final validation accuracy on Imagenet for a self-supervised pretrained Visual Transformer. Each dash at the bottom corresponds to one run.

VIT SSL

ImageNet 실험 (pretrained)

- 50개 random seed
- gap 0.5%

Conclusion

1. 시드 선택에 대한 accuracy 분포?

- 해당 분포는 pointly 하게 집중, 이는 평균 주위에 상당히 집중되어 있음을 의미
- model 수렴시 이 분포는 상대적으로 안정적이므로 일부 seed가 본질적으로 다른 seed보다 우수함

2. 근본적으로 다른 결과를 내는 black swans가 있는가?

- Yes, 10^4 개의 seed를 추적관찰 한 결과, accuracy의 최대와 최소의 gap이 일반적으로 컴퓨터 비전 커뮤니티에서 중요하다가 간주되는 임계치인 2%에 가까운 gap을 보임.
- 그래서 black swans사용하는게 효과적

3. 대규모 dataset에 대한 preratining이 시드 선택에 의해 유발된 가변성을 완화시키는가?

- 대규모 데이터 셋에 대해 사전 훈련된 모델이 랜덤 시드로 인한 차이를 줄이더라도 가변성을 제거하지는 못한다.
- imageNet에서 최대와 최소 정확도의 gap이 약 0.5%의 차이를 발견 했는데, 이는 일반적으로 해당 dataset 커뮤니티에서 중요하게 받아들여지고 있다.

UNETR: Transformers for 3D Medical Image Segmentation

Ali Hatamizadeh, NVIDIA
Yucheng Tang, Vanderbilt University
Vishwesh Nath, NVIDIA
Dong Yang, NVIDIA
Andriy Myronenko, NVIDIA
Bennett Landman, Vanderbilt University
Holger R. Roth, NVIDIA
Daguang Xu, NVIDIA

https://arxiv.org/pdf/2103.10504.pdf

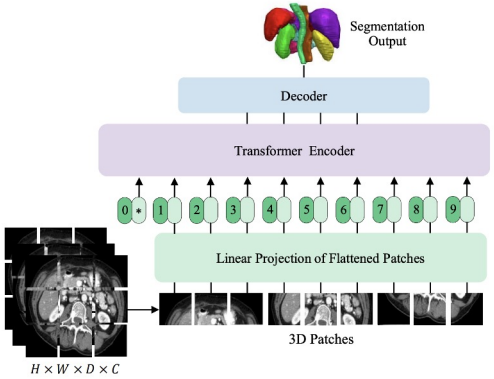


Figure 1. Overview of UNETR. Our proposed model consists of a transformer encoder that directly utilizes 3D patches and is connected to a CNN-based decoder via skip connection.

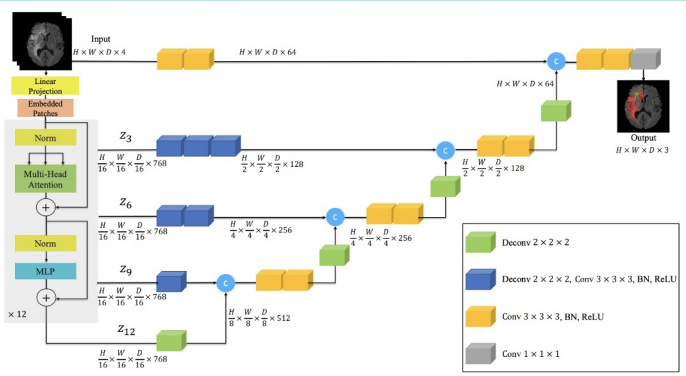


Figure 2. Overview of UNETR architecture. A 3D input volume (e.g. $C = 4$ channels for MRI images), is divided into a sequence of uniform non-overlapping patches and projected into an embedding space using a linear layer. The sequence is added with a position embedding and used as an input to a transformer model. The encoded representations of different layers in the transformer are extracted and merged with a decoder via skip connections to predict the final segmentation. Output sizes are given for patch resolution $P = 16$ and embedding size $K = 768$.

- **Research Group: NVIDIA**

- Reformulate the task of volumetric (3D) medical image segmentation as a sequence-to-sequence prediction problem.
- UNETR utilizes a contracting-expanding pattern consisting of a stack of transformers as the encoder which is connected to a decoder via skip connections.

- Novel architecture
  - UNEt TRansformers (UNETR)
    - **encoder 로써 transformer 사용** : input volume 에 대한 sequence representation 학습과, global multi-scale information capture
    - **encoder-decoder : U-shaped network design,** 다른 resolution 에서 final semantic segmentation output 을 계산하기 위해, **skip connections** 을 통해 **transformer encoder 와 decoder 가 directly connected.**

1) **3D input volume (W * H * D * C) -> 1D sequence of a 3D input volume**
   즉, 3D input volume 이미지를 sequence of uniform non-overlapping patches 로 나눔 (transformer encoder 에서 3D patches 바로 사용)
1) patches 를 linear layer 를 통해서 k dimensional **embedding space에 projection.**
   이 때, **sequence 에 position embedding 추가함 (추출된 patches 에서 spatial information 보존을 위함) → 이를 transformer model 에 input.**
1) transformer encoder 각각의 layer representations 을 추출 후, skip connections 을 통해 decoder 에 합침 (to predict final segmentation)

$$\mathbf{z}_0 = [\mathbf{x}_v^1\mathbf{E}; \mathbf{x}_v^2\mathbf{E}; \ldots; \mathbf{x}_v^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad (1)$$

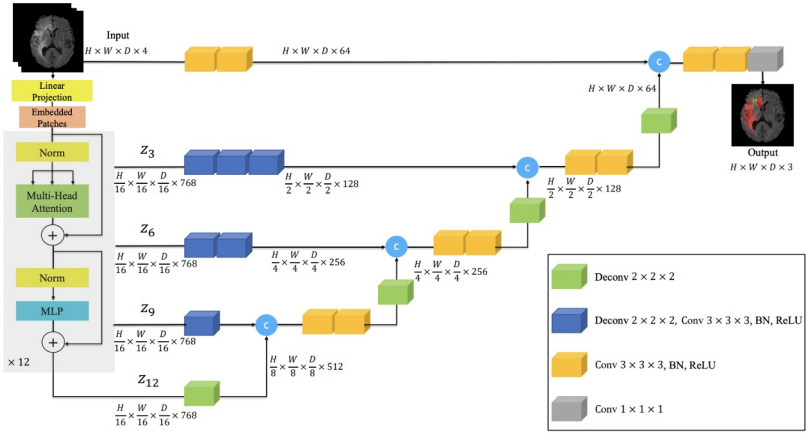


Figure 2. Overview of UNETR architecture. A 3D input volume (e.g. $C = 4$ channels for MRI images), is divided into a sequence of uniform non-overlapping patches and projected into an embedding space using a linear layer. The sequence is added with a position embedding and used as an input to a transformer model. The encoded representations of different layers in the transformer are extracted and merged with a decoder via skip connections to predict the final segmentation. Output sizes are given for patch resolution $P = 16$ and embedding size $K = 768$.

- **Loss function**
  - soft dice loss + cross-entropy loss

$$\mathcal{L}(G,Y) = 1 - \frac{2}{J}\sum_{j=1}^{J}\frac{\sum_{i=1}^{I} G_{i,j}Y_{i,j}}{\sum_{i=1}^{I} G_{i,j}^2 + \sum_{i=1}^{I} Y_{i,j}^2} - \\ -\frac{1}{I}\sum_{i=1}^{I}\sum_{j=1}^{J} G_{i,j}\log Y_{i,j}. \quad (7)$$

- **Skip connection**
  - U-Net
    - encoder 의 multiple resolutions feature 를 decoder 에 merge
  - The encoded representations of different layers in the transformer are extracted and merged with a decoder via skip connections to predict the final segmentation.
  - transformer 에서 sequence representation z{i} 추출 → reshape them into a size $\frac{H \times W \times D}{P^3} \times K$
    (reshaped 된 후, is in embedding space as an output of the transformer)

- output of transformer's last layer (at the bottleneck of encoder) 에서, deconvolutional layer 적용.
  ⇒ resized feature map 과 이전 transformer output concatenate (original input resolution 까지 모두 적용)
  ⇒ final output is fed into a 1*1*1 convolutional layer with a softmax activation function to generate voxel-wise semantic predictions.

- **Experiments**
  - datasets (three different segmentation tasks in CT and MRI imaging modalities)
    - BTCV (CT), MSD (MRI/CT)

Methods	Spl	RKid	LKid	Gall	Eso	Liv	Sto	Aor	IVC	Veins	Pan	AG	Avg.
SETR NUP [52]	0.931	0.890	0.897	0.652	0.760	0.952	0.809	0.867	0.745	0.717	0.719	0.620	0.796
SETR PUP [52]	0.929	0.893	0.892	0.649	0.764	0.954	0.822	0.869	0.742	0.715	0.714	0.618	0.797
SETR MLA [52]	0.930	0.889	0.894	0.650	0.762	0.953	0.819	0.872	0.739	0.720	0.716	0.614	0.796
nnUNet [21]	0.942	0.894	0.910	0.704	0.723	0.948	0.824	0.877	0.782	0.720	0.680	0.616	0.802
ASPP [10]	0.935	0.892	0.914	0.689	0.760	0.953	0.812	0.918	0.807	0.695	0.720	0.629	0.811
TransUNet [7]	0.952	**0.927**	0.929	0.662	0.757	0.969	0.889	0.920	0.833	0.791	0.775	0.637	0.838
CoTr w/o CNN encoder [47]	0.941	0.894	0.909	0.705	0.723	0.948	0.815	0.876	0.784	0.723	0.671	0.623	0.801
CoTr* [47]	0.943	0.924	0.929	0.687	0.762	0.962	0.894	0.914	0.838	**0.796**	**0.783**	0.647	0.841
CoTr [47]	0.958	0.921	0.936	0.700	0.764	0.963	0.854	**0.920**	0.838	0.787	0.775	0.694	0.844
UNETR	**0.968**	0.924	**0.941**	**0.750**	**0.766**	**0.971**	**0.913**	0.890	**0.847**	0.788	0.767	**0.741**	**0.856**
RandomPatch [39]	0.963	0.912	0.921	0.749	0.760	0.962	0.870	0.889	0.846	0.786	0.762	0.712	0.844
PaNN [53]	0.966	0.927	0.952	0.732	0.791	0.973	0.891	0.914	0.850	0.805	0.802	0.652	0.854
nnUNet-v2 [21]	0.972	0.924	**0.958**	0.780	0.841	0.976	0.922	0.921	0.872	0.831	0.842	0.775	0.884
nnUNet-dys3 [21]	0.967	0.924	0.957	0.814	0.832	0.975	0.925	0.928	0.870	0.832	**0.849**	0.784	0.888
UNETR	**0.972**	**0.942**	0.954	**0.825**	**0.864**	**0.983**	**0.945**	**0.948**	**0.890**	**0.858**	0.799	**0.812**	**0.891**

Table 1. Quantitative comparisons of segmentation performance in BTCV test set. Top and bottom sections represent the benchmarks of Standard and Free Competitions respectively. Our method is compared against current state-of-the-art models. All SETR [52] baselines use ViT-B-16 [14] backbone. Note: Spl: spleen, RKid: right kidney, LKid: left kidney, Gall: gallbladder, Eso: esophagus, Liv: liver, Sto: stomach, Aor: aorta IVC: inferior vena cava, Veins: portal and splenic veins, Pan: pancreas, AG: adrenal gland. All results obtained from BTCV leaderboard.

Task/Modality	Spleen Segmentation (CT)		Brain tumor Segmentation (MRI)							
Anatomy	Spleen		WT		ET		TC		All	
Metrics	Dice	HD95	Dice	HD95	Dice	HD95	Dice	HD95	Dice	HD95
UNet [36]	0.953	4.087	0.766	9.205	0.561	11.122	0.665	10.243	0.664	10.190
AttUNet [34]	0.951	4.091	0.767	9.004	0.543	10.447	0.683	10.463	0.665	9.971
SETR NUP [52]	0.947	4.124	0.697	14.419	0.544	11.723	0.669	15.192	0.637	13.778
SETR PUP [52]	0.949	4.107	0.696	15.245	0.549	11.759	0.670	15.023	0.638	14.009
SETR MLA [52]	0.950	4.091	0.698	15.503	0.554	10.237	0.665	14.716	0.639	13.485
TransUNet [7]	0.950	4.031	0.706	14.027	0.542	10.421	0.684	14.501	0.644	12.983
TransBTS [43]	-	-	0.779	10.030	0.574	9.969	0.735	8.950	0.696	9.650
CoTr w/o CNN encoder [47]	0.946	4.748	0.712	11.492	0.523	9.592	0.698	12.581	0.6444	11.221
CoTr [47]	0.954	3.860	0.746	9.198	0.557	9.447	0.748	10.445	0.683	9.697
UNETR	**0.964**	**1.333**	**0.789**	**8.266**	**0.585**	**9.354**	**0.761**	**8.845**	**0.711**	**8.822**

Table 2. Quantitative comparisons of the segmentation performance in brain tumor and spleen segmentation tasks of the MSD dataset. WT, ET and TC denote Whole Tumor, Enhancing tumor and Tumor Core sub-regions respectively.

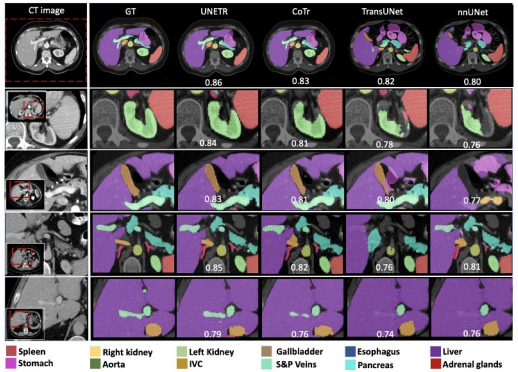


Figure 3. Qualitative comparison of different baselines in BTCV cross-validation. The first row shows a complete representative CT slice. We exhibit four zoomed-in subjects (row 2 to 5), where our method shows visual improvement on segmentation of kidney and spleen (row 2), pancreas and adrenal gland (row 3), gallbladder (row 4) and portal vein (row 5). The subject-wise average Dice score is shown on each sample.

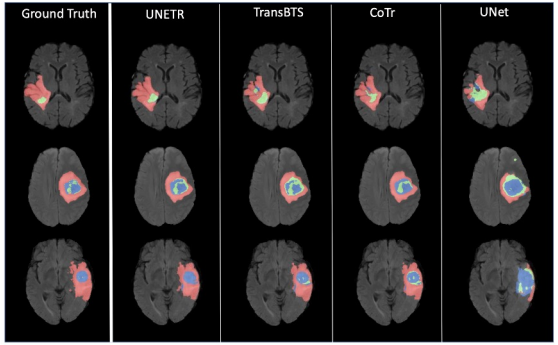


Figure 4. UNETR effectively captures the fine-grained details in segmentation outputs. The Whole Tumor (WT) encompasses a union of red, blue and green regions. The Tumor Core (TC) includes the union of red and blue regions. The Enhancing Tumor core (ET) denotes the green regions.

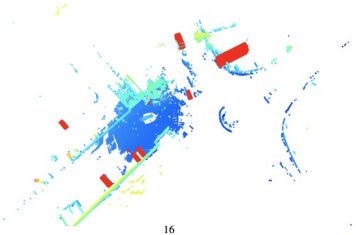
https://arxiv.org/pdf/2110.06923.pdf

Object DGCNN:
3D Object Detection using Dynamic Graphs

Yue Wang
Massachusetts Institute of Technology
yuewang@csail.mit.edu

Justin Solomon
Massachusetts Institute of Technology
jsolomon@mit.edu

https://arxiv.org/pdf/2110.06923.pdf



16

Figure 3: Visualizations. Our model makes a sparse set of predictions (boxes in red) without using NMS.

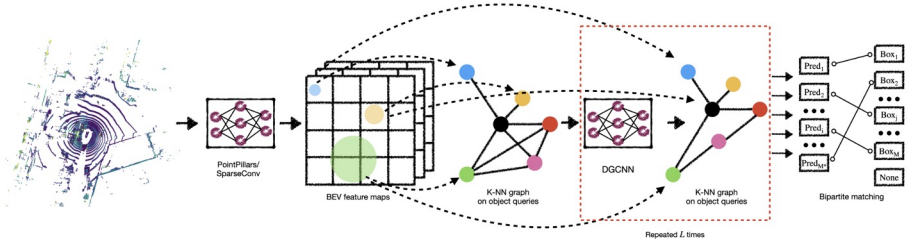
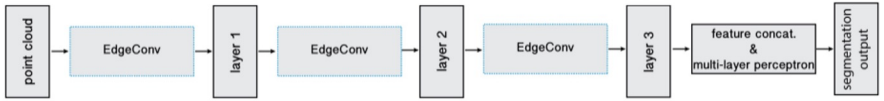


Figure 1: Overview. Point cloud features are learned in BEV, followed by L DGCNNs to model object relations. We predict a set of bounding boxes and compute loss in a one-to-one manner.



<Abstract>

- DGCNN ⇒ 본 논문에서는 이 DGCNN의 post-processing을 제거한 방법 제시
- 3D object detection architecture on point clouds ⇒ 근데 약간 3D image segmentation 느낌이 강함
- **remove the necessity of post-processing via object confidence aggregation or non maximum suppression (robust to whether to use NMS)**
- To facilitate object detection from sparse point clouds, we also propose **a set-to-set distillation** approach customized to 3D detection
  - predict a set of query points and attention weights
  - collect BEV features from keypoints determined by the queries
  - model object-object interactions via DGCNN.
- Github: https://github.com/WangYueFt/detr3d

https://arxiv.org/pdf/2110.06923v1.pdf

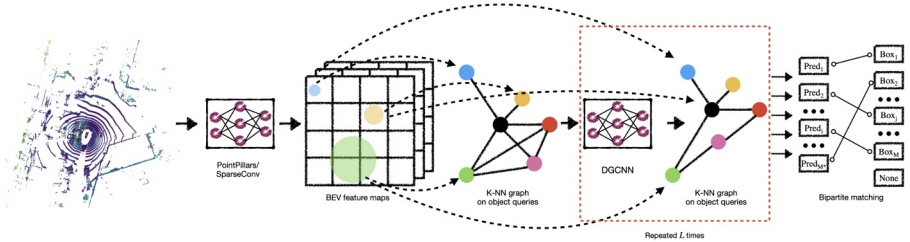


Figure 1: Overview. Point cloud features are learned in BEV, followed by L DGCNNs to model object relations. We predict a set of bounding boxes and compute loss in a one-to-one manner.

- BEV (bird-eye view; 조감도) feature extraction network
  - **3D object detection model 은 포인트를 BEV pillars or 3D voxel 로 흩뿌림 (scatter)**
    ⇒ 그 후, cnn 을 사용하여 extract features on a grid.
  - 조감도 형태의 특징맵 추출
  - 본 과정을 통해, large point clouds 에서 object detection 을 더 빠르게 수행할 수 있음.

Object DGCNN uses L layers that follow a series of set-based computations to produce bounding box predictions from the BEV feature maps. Each layer employs the following steps (Figure 1):

1. predict a set of query points and attention weights;
2. collect BEV features from keypoints determined by the queries; and
3. model object-object interactions via DGCNN.

Set-to-set loss. After L Object DGCNN layers as described above, we are left with a set of M^* queries $\mathcal{Q}_L$ used to predict our bounding boxes. For each query $\boldsymbol{q}_{Li}$, we use a classification network to predict a categorical label $\hat{c}_i$ and a regression network to predict bounding box parameters $\hat{\boldsymbol{b}}_i$. Our final task is to assign the predictions to the ground-truth boxes and compute a set-to-set loss.

Most object detection models minimize a loss $\mathcal{L}_{\text{od}}$ given by

$$\mathcal{L}_{\text{od}} = \sum_{j=1}^{\hat{M}} -\log \hat{p}_{\hat{\sigma}(j)}(\hat{c}_j) + 1_{\{c_{\hat{\sigma}(j)} \neq \varnothing\}} \mathcal{L}_{\text{box}}(\hat{\boldsymbol{b}}_j, \boldsymbol{b}_{\hat{\sigma}(j)}), \tag{7}$$

https://arxiv.org/pdf/2110.06923v1.pdf

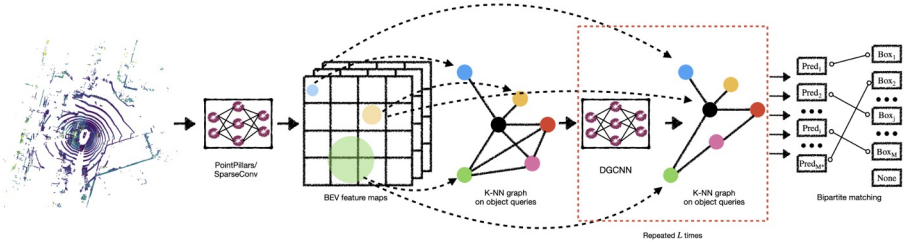


Figure 1: Overview. Point cloud features are learned in BEV, followed by L DGCNNs to model object relations. We predict a set of bounding boxes and compute loss in a one-to-one manner.

- set-to-set knowledge distillation
  - 기존 3D object detection 모델의 경우, final performance 가 NMS 에 크게 의존하고 있기 때문에 ⇒ KD 가 효과적이지 못한 방법이었음.
  - 그러나, 본 모델의 경우 pre-processing 인 NMS 에 의존하지 않게 되어 (NMS-free)
    we can easily distill the information between models.

$$\sigma_d^* = \arg\min_{\sigma_d \in \mathcal{P}} \sum_j^N -\log p_{\sigma_d(j)}(c_j^{\mathcal{T}}) + \mathcal{L}_{\text{box}}(\boldsymbol{b}_j^{\mathcal{T}}, \boldsymbol{b}_{\sigma_d(j)}^{\mathcal{S}}). \tag{11}$$

Then, the optimal matching's KD loss is given by

$$\mathcal{L}_{\text{distill}} = \sum_j^N -\log \hat{p}_{\sigma_d^*(j)}(c_j^{\mathcal{T}}) + \mathcal{L}_{\text{box}}(\boldsymbol{b}_j^{\mathcal{T}}, \boldsymbol{b}_{\sigma_d^*(j)}^{\mathcal{S}}). \tag{12}$$

So the overall loss during KD is $\mathcal{L} = \alpha\mathcal{L}_{\text{sup}} + \beta\mathcal{L}_{\text{distill}}$, where α and β balance the supervised loss and distillation loss. In practice, we use $\alpha = \beta = 1$.

https://arxiv.org/pdf/2110.06923v1.pdf

Table 1: Comparisons to recent works. Our method is robust to whether to use NMS. ∗: implementations with the same PointPillars backbone. ‡: implementations with the same SparseConv backbone.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NMS
PointPillars [4]	53.3	40.0	-	-	-	-	-	✓
SSN [63]	54.83	41.56	-	-	-	-	-	✓
FreeAnchor [64]	55.3	43.7	-	-	-	-	-	✓
RegNetX-400MF-SECFPN [65]	55.2	41.2	-	-	-	-	-	✓
Pillar-OD [5]	56.84	44.41	-	-	-	-	-	✓
CenterPoint (pillar) [6] ∗	59.56	47.48	**31.27**	**25.81**	33.78	32.25	20.20	✓
CenterPoint (pillar) [6] ∗	55.08	40.27	35.14	26.44	36.75	32.66	19.55	
CenterPoint (voxel) [6] ‡	**64.19**	**54.99**	**29.83**	**25.71**	32.56	26.08	**18.89**	✓
CenterPoint (voxel) [6] ‡	57.00	45.32	31.66	27.14	40.47	37.23	20.14	
Ours (pillar) ∗	**62.97**	**53.31**	34.62	26.56	**31.61**	**26.02**	**18.71**	✓
Ours (pillar) ∗	62.80	53.20	34.62	26.56	31.62	26.07	19.10	
Ours (voxel) ‡	**66.10**	58.73	33.31	**26.32**	**28.80**	**25.11**	19.08	✓
Ours (voxel) ‡	**66.04**	58.62	33.33	26.34	28.80	**25.11**	**19.06**	

table1. 기존 모델 비교, 주로 CenterPoint와 비교 함

- 주요 point: CenterPoint에서 NMS 사용 유무에 관해 성능 차이와 DGCNN에서 NMS 사용유무에 관한 성능차이 비교

- table2. 기존 distill과 본 논문에서 제시한 distill 비교
- table3. set-to-set의 2단계에서 "voxel→voxel", "pillar→pillar" 방법이 좋다고 자랑

Table 2: Comparisons of different distillation approaches.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
Baseline (without distillation)	62.80	53.20	34.62	26.56	31.62	26.02	19.10
Feature distill (voxel→pillar)	62.84	53.29	34.55	26.78	31.61	26.11	19.02
Pseudo labels (voxel→pillar)	63.18	53.31	34.58	26.55	31.29	25.99	18.87
Set-to-set distill (voxel→pillar)	63.37	53.89	34.34	26.25	31.01	25.57	18.77

Table 3: Comparisons of self-distillation versus baselines.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
Baseline (pillar, without distillation)	62.80	53.20	34.62	26.56	31.62	26.02	19.10
Self-distillation (pillar→pillar)	63.41	53.89	34.21	26.19	31.11	25.67	18.54
Baseline (voxel, without distillation)	66.04	58.62	33.33	26.34	28.80	25.11	19.06
Self-distillation (voxel→voxel)	66.45	59.25	31.17	25.77	30.73	25.72	18.77

Table 4: Self-distillation with privileged information.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
Sparse→sparse (pillar)	42.12	38.89	44.01	27.78	64.01	144.01	39.21
Dense→sparse (pillar)	42.79	39.10	43.89	27.77	64.01	143.97	39.11
Sparse→sparse (voxel)	59.55	49.84	31.17	25.77	33.73	32.22	20.22
Dense→sparse (voxel)	59.89	50.12	31.11	25.76	33.70	32.19	20.11

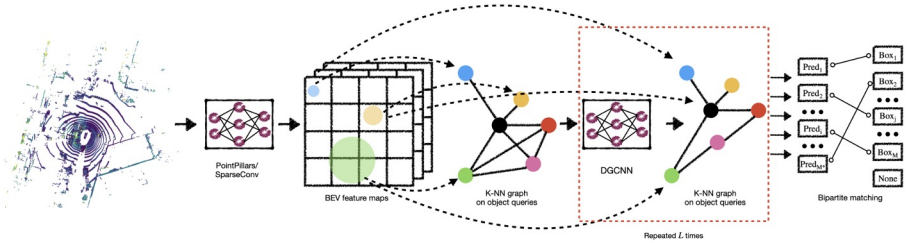


Figure 1: Overview. Point cloud features are learned in BEV, followed by L DGCNNs to model object relations. We predict a set of bounding boxes and compute loss in a one-to-one manner.

First, we distill a teacher model with a SparseConv backbone to a student model with a PointPillars backbone ("voxel→pillar")

Second, we perform self-distillation ("voxel→voxel" and "pillar→pillar"), where the teacher and the student are identical and take the same point clouds as input.

Noisy Labels Can Induce Good Representations

https://arxiv.org/pdf/2012.12896.pdf

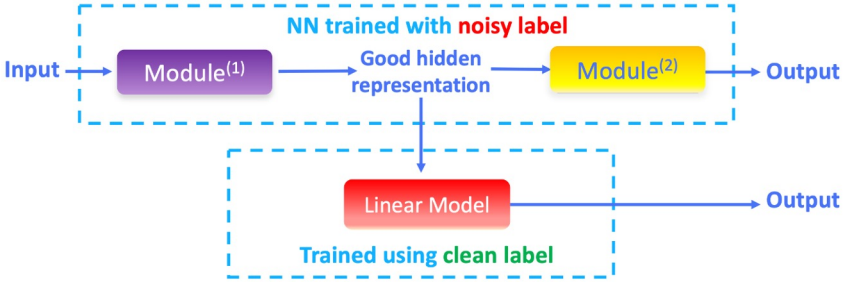
Jingling Li [†] Mozhi Zhang [‡] Keyulu Xu [§]
John P. Dickerson [¶] Jimmy Ba [‖]

To explain these mixed results, we investigate the *hidden representations* induced by noisy label training. We find that **when the network architecture is appropriate for learning the signal, noisy label training induces good hidden representations, even when the test error is large**.

Noisy Label로 학습시켜도 초반부 레이어의 Representations은 잘 학습된다.
후반부 레이어가 문제이니 이 레이어를 linear classifier로 갈아치우고 clean label에 대해서 학습시킨다는 아이디어.

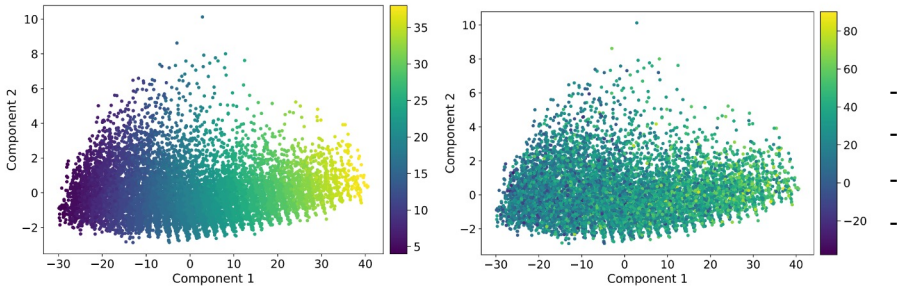
- 현재 deep learning은 large-scale labeled dataset에 대해서는 잘 작동,
- high-quality annotations을 수집하는데 많은 시간이 소요되지만 noisy annotation의 경우 좀더 affordable.
- 이전 연구, noisy labels data가 섞인 data를 학습 한 결과에 대한 보고가 많음: random하게 학습할 수 있지만, noisy labels로 학습 한 후 generalize할 수 있음.
- **해당 puzzle을 설명하기 위해, noisy labels이 어떤 영향을 미치는지 연구.**
- **풀고자 하는 task의 architecture가 적합한 경우, noisy labels를 사용하는 것은 useful**, (even, 모델이 일반화 되지 않은 경우에도)
- **i.e, 모델의 마지막 몇 계층은 노이즈가 있는 레이블에 의해 더 부정적인 영향을 받기도 함.**
- 이에, 최종 dense layer를 linear model로 대체하고, 해당 weights는 clean data에서 학습.
- CNN, GNN, MLP 세 가지 아키텍처 // graph algorithmc task, image classification 2 가지 task 를 통해 검증

All training labels are corrupted by adding a Gaussian noise drawn from $\mathcal{N}(10, 15)$.



- **our methods (inspired by transfer learning)**
1. small subset of data (clean labels) + noisy labels
2. After training the entire network on noisy data, replace the final few layers with linear model
3. whose weights are learned from the clean subset

Figure 2: **Overview of our method.** We divide a network into two modules: Module$^{(1)}$ (e.g., convolutional layers) that can easily learn the signal and Module$^{(2)}$ (e.g., final dense layers) that maps the learned representations to the final prediction. Module$^{(1)}$ can learn good representations from noisy labels. Thus, after training both modules with noisy labels, we fix Module$^{(1)}$ and replace Module$^{(2)}$ with a linear model trained on a small set of clean data.



(a) Representations colored with true labels

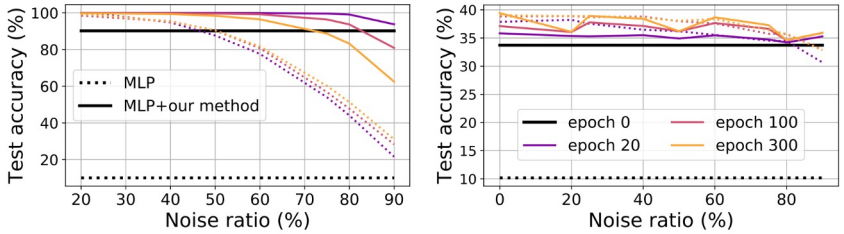
(b) Representations colored with noisy labels

- PCA visualization of hidden representations for a GNNs tranined under noisy labels.
- graph algorithic tasks with different types of label noise.
- (a, b) final MLP using training data
- 좌 (Good Representations:useful patterns), 우 (Worse Representations)

transfer learning

Transferring information from one machine learning task to another. For example, in multi-task learning, a single model solves multiple tasks, such as a **deep model** that has different output nodes for different tasks. Transfer learning might involve transferring knowledge from the solution of a simpler task to a more complex one, or involve transferring knowledge from a task where there is more data to one where there is less data.

Most machine learning systems solve a *single* task. Transfer learning is a baby step towards artificial intelligence in which a single program can solve *multiple* tasks.



(a) MLP is an appropriate architecture

(b) MLP is not an appropriate architecture

Figure 3: **Our method only helps with appropriate architectures.** We compare the test accuracy before (dotted) and after (solid) applying our method to the same architecture (MLP) trained on two different tasks. The left plot is on predicting a linear function where MLP can easily learn, and the right plot is on an image classification task where MLP is less suitable (detailed descriptions of both tasks are in Section 4.2). Each line indicates the test accuracy at a specific epoch (darker colors denote earlier epochs, and brighter colors denote later epochs). Across different epochs, our method (using 10% clean data) greatly improves the test accuracy when MLP is good at learning the task (left) and has good hidden representations. In contrast, our method barely helps when MLP is less suitable for the task (right) as the learned hidden representations are less useful.

4.1 Datasets and Noises

CIFAR-10 and **CIFAR-100** [44] come with clean labels. Thus, we simulate the noisy settings following existing methods to generate two types of noisy labels: (1) **symmetric noise** generated by randomly replacing the true labels with all possible labels; and (2) **asymmetric noise** generated by replacing labels between similar classes (e.g., deer↔horse, dog↔cat) on CIFAR-10 [48], or flipping labels to the next class on CIFAR-100 [67].

Clothing1M [96] has real-world noisy labels with an estimated 38.5% noise ratio. The dataset has a small human-verified training data, which we use as clean data. Following recent method [48], we use 1000 mini-batches in each epoch to train models on Clothing1M.

WebVision [51] also has real-world noisy labels with an estimated 20% noise ratio. It shares the same 1000 classes as ImageNet [17]. For a fair comparison, we follow [40] to create a mini WebVision dataset with the top 50 classes from the Google image subset of WebVision. We train all

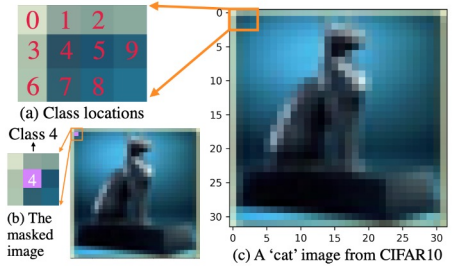


Figure 7: **Our designed labels for CIFAR-Easy.** For each image from CIFAR-10, we mask a pixel at the top left corner with a particular color (e.g., pink). Then the synthetic label is the location of the mask (Class 4 in the example).

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

모델 신뢰성 검증:
오차가 예측값에서 차지하는 정도를 나타내는 지표

- 오차 평균의 크기가 차이나는 모델을 비교하는 경우,
  오차 평균의 크기가 더 작은 모델을 좋은 모델로 평가

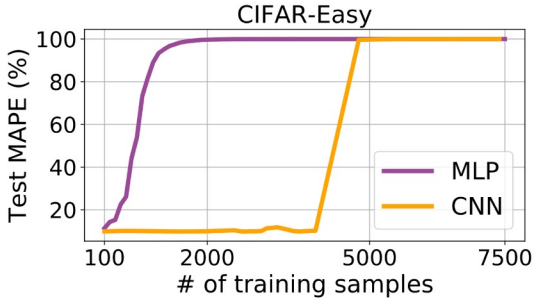


Figure 8: **Sample complexity of MLP and CNN on CIFAR-Easy.** Given enough samples, both MLP and CNN can achieve 100% test accuracy, but MLP needs far fewer samples than CNN and thus is more sample-efficient.

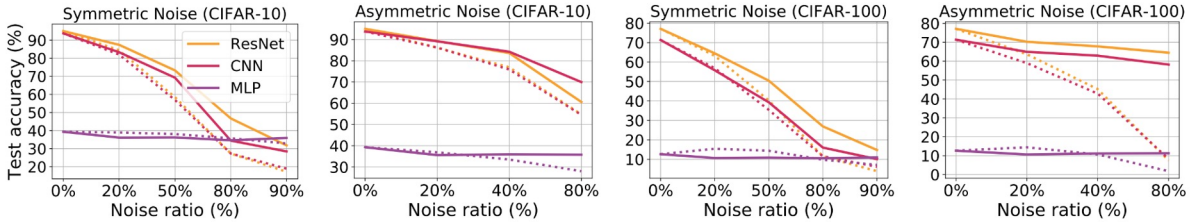


Figure 9: **CIFAR-10/100 with symmetric and asymmetric noise.** Each line indicates the test accuracy before (dotted) and after (solid) applying our method to models trained with standard procedures (i.e., vanilla training). Across both datasets and both types of label noise, our method consistently improves the test accuracy of CNNs-based models (purple/red lines) but barely helps MLPs (yellow lines). The results suggest that for image classification tasks, noisy training can induce better representations on CNN-based models than MLPs, which matches with the common intuition that CNN-based models are more appropriate for image classification tasks than MLPs.

- **symmetric and asymetric noise data에 따른 accuracy**
- **dotted : before / solid : our method**
- **our method : CNN > MLP 효과적**
- **asymetric > symmetric**
- **CNN일 경우 높은 Noise ratio일 경우 poor**

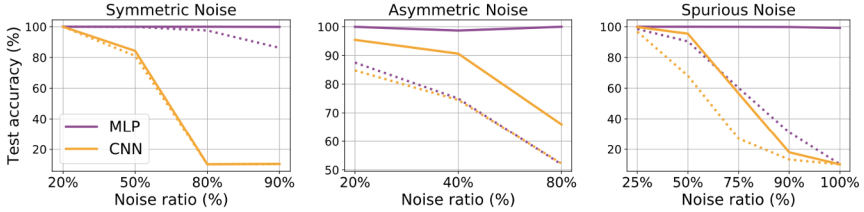
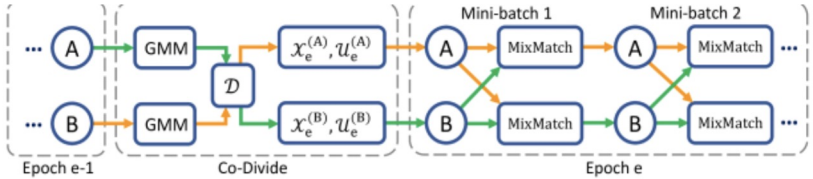


Figure 10: **CIFAR-10 with our designed labels (CIFAR-Easy).** Each line indicates the test accuracy before (dotted) and after (solid) applying our method to models trained with standard procedures (i.e. vanilla training) under different noise types and noise ratios. Our method achieves better test accuracies on MLPs than CNNs across all three types of label noise, which suggests that the learned hidden representations of MLPs are much more useful than those of CNNs, even when both models have similarly poor test performance (e.g., dotted lines in asymmetric noise and spurious noise).

Method	# clean per class	Noise ratio					
		0%	20%	40%	50%	80%	90%
Cross-Entropy	-	95.0	84.4	67.9	58.5	27.3	17.2
MentorNet [40]	500	96.0	92.0	89.0	-	49.0	-
L2R [75]	100	96.1	90.0 ± 0.4	86.9 ± 0.2	-	73.0 ± 0.8	-
IEG [112]	10	94.4	92.9 ± 0.2	92.5 ± 0.5	-	85.6 ± 1.1	-
M-correction [2]	-	93.4	93.8	92.0	91.9	86.6	68.7
DivideMix [48]	-	95.0	95.7	94.4*	94.4	92.9	75.4
DivideMix+Ours	10	95.0	**96.0 ± 0.1**	**94.5 ± 0.1**	**94.7 ± 0.0**	**93.2 ± 0.1**	74.6 ± 0.4
DivideMix+Ours	100	95.0	**96.1 ± 0.1**	**94.5 ± 0.1**	**94.8 ± 0.1**	**93.4 ± 0.1**	**76.7 ± 0.4**
DivideMix+Ours	500	95.0	**96.1 ± 0.1**	**94.6 ± 0.1**	**94.9 ± 0.1**	**93.6 ± 0.1**	**80.4 ± 0.4**

Table 2: Test accuracy (%) on **CIFAR-10 with symmetric noise**. Our method consistently improves DivideMix. * indicates results reproduced with released code.

DivideMix

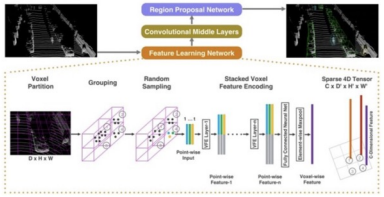


DivideMix: Learning with Noisy Labels as Semi-supervised Learning

Junnan Li, Richard Socher, Steven C.H. Hoi
Salesforce Research
{junnan.li,rsocher,shoi}@salesforce.com

https://arxiv.org/pdf/2110.06923v1.pdf

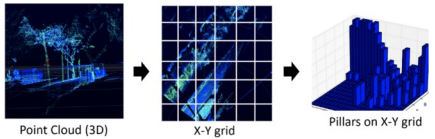
VoxelNet



[그림 7] Apple 사에서 발표한 VoxelNet의 구조
(출처 : Yin Zhou, et al. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection" 논문에서 발췌)

VoxelNet[2]은 Point Cloud 데이터로부터 Voxel Feature를 추출한 다음 이를 해석해 물체를 검출하는 3D Object Detection 모델이다. Voxel Feature는 데이터의 3차원 공간을 Voxel 단위로 나눈 후 각 Voxel 안의 점들을 Voxel Feature Encoding Layer라는 딥러닝 네트워크를 거쳐 Feature Map을 얻어낸 것이다. Point Cloud 데이터를 단순히 Voxel 형태로 전처리하는 것이 아니라 딥러닝 네트워크를 통해 Voxel 단위의 Feature Map을 만들어낸 것이 특징이다. 이러한 방법으로 얻은 데이터는 기존 방법보다 딥 러닝 네트워크를 통한 해석이 용이함이 실험을 통해 증명되었다.

PointPillars



[그림 9] PointPillar 생성 과정
(출처 : Anjul Tyagi. "PointPillars — 3D point clouds bounding box detection and tracking" Medium 글에서 발췌)

PointPillars[4]는 Pillar라는 새로운 형태의 Point Cloud Encoder를 사용해 Point Cloud 데이터로부터 격자 형태의 Feature Map을 얻은 다음, 이를 해석해 물체를 검출하는 3D Object Detection 모델이다. VoxelNet이 3D Voxel 단위의 Feature Map을 얻어냈다면, PointPillar는 Point Cloud 데이터를 특정 시점에서 투영시킨 다음 2D 격자 단위의 Feature Map을 얻어낸 것이 특징이다. 이렇게 얻어낸 Feature Map은 이미지 데이터와 동일하게 2D CNN을 통한 데이터 해석이 가능해지므로 처리 속도가 상당히 빠르다.

Y_{clean}: let C denote the number of classes in the training data, then

$$I(Y_{\text{clean}}, Y_{\text{noisy}}) = \sum_{y_1=1}^{C} \sum_{y_2=1}^{C} P_{Y_{\text{clean}}, Y_{\text{noisy}}}(y_1, y_2) \log \left(\frac{P_{Y_{\text{clean}}, Y_{\text{noisy}}}(y_1, y_2)}{P_{Y_{\text{clean}}}(y_1) P_{Y_{\text{noisy}}}(y_2)} \right). \tag{8}$$

Intuitively, mutual information quantifies the "amount of information" in the noisy training data to learn the original task. Using mutual information as the metric, we can fairly compare symmetric and asymmetric noise across different noise ratios. For example, on CIFAR-100, the mutual information shared between the original training data and the training data with 40% asymmetric noise is around

$$100 \times (0.004 * \log(40) + 0.006 * \log(60)) \approx 5.67;$$

while the mutual information shared between the original data and the data with 40% symmetric noise is around

$$100 \times (0.006 * \log(60)) \approx 3.54.$$

Notice that the mutual information decreases as the noise ratio increases, and the positive correlation in Figure 11 indicates that higher mutual information can help induce better representations (for appropriate architectures).

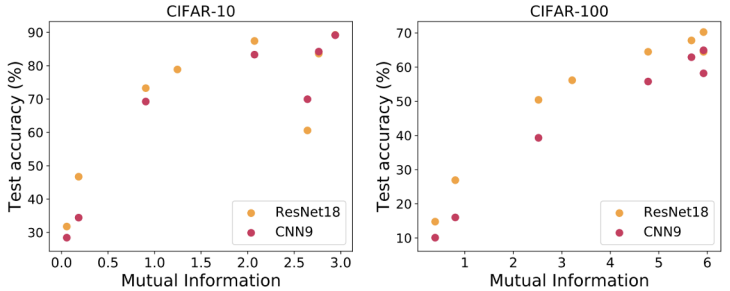


Figure 11: **Mutual information vs. Test accuracy (after applying our method to appropriate architectures).** The test accuracy after applying our method to vanilla-trained models on the noisy data has a strong positive correlation with the mutual information between the noisy data and the original clean training set. This positive correlation indicates that higher mutual information can help induce better representations.

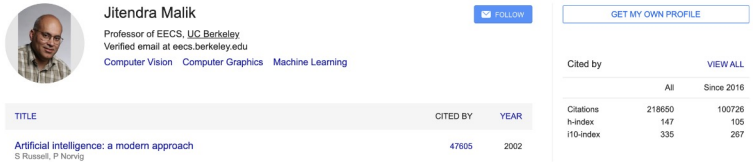
Ego4D: Around the World in 3,000 Hours of Egocentric Video

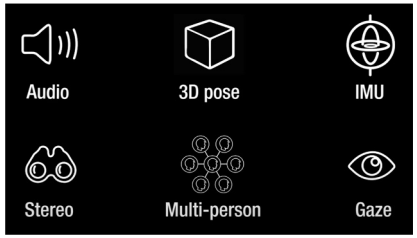
Ego4D: Around the World in 3,000 Hours of Egocentric Video

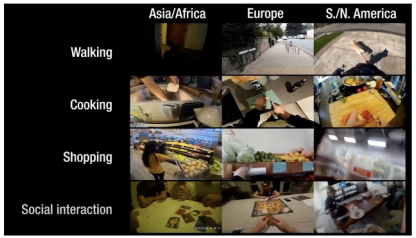
Kristen Grauman[1,2], Andrew Westbury[1], Eugene Byrne*[1], Zachary Chavis*[3], Antonino Furnari*[4], Rohit Girdhar*[1], Jackson Hamburger*[1], Hao Jiang*[5], Miao Liu*[6], Xingyu Liu*[7], Miguel Martin*[1], Tushar Nagarajan*[1,2], Ilija Radosavovic*[8], Santhosh Kumar Ramakrishnan*[1,2], Fiona Ryan*[6], Jayant Sharma*[3], Michael Wray*[9], Mengmeng Xu*[10], Eric Zhongcong Xu*[11], Chen Zhao*[10], Siddhant Bansal[17], Dhruv Batra[1], Vincent Cartillier[1,6], Sean Crane[7], Tien Do[3], Morrie Doulaty[13], Akshay Erapalli[13], Christoph Feichtenhofer[1], Adriano Fragomeni[9], Qichen Fu[7], Christian Fuegen[13], Abrham Gebreselasie[12], Cristina González[14], James Hillis[5], Xuhua Huang[7], Yifei Huang[15], Wenqi Jia[6], Weslie Khoo[16], Jachym Kolar[13], Satwik Kottur[13], Anurag Kumar[5], Federico Landini[13], Chao Li[5], Yanghao Li[1], Zhenqiang Li[15], Karttikeya Mangalam[1,8], Raghava Modhugu[17], Jonathan Munro[9], Tullie Murrell[1], Takumi Nishiyasu[15], Will Price[9], Paola Ruiz Puentes[14], Merey Ramazanova[10], Leda Sari[5], Kiran Somasundaram[5], Audrey Southerland[6], Yusuke Sugano[15], Ruijie Tao[11], Minh Vo[5], Yuchen Wang[16], Xindi Wu[7], Takuma Yagi[15], Yunyi Zhu[11], Pablo Arbeláez†[14], David Crandall†[16], Dima Damen†[9], Giovanni Maria Farinella†[4], Bernard Ghanem†[10], Vamsi Krishna Ithapu†[5], C. V. Jawahar†[17], Hanbyul Joo†[1], Kris Kitani†[7], Haizhou Li†[11], Richard Newcombe†[5], Aude Oliva†[18], Hyun Soo Park†[3], James M. Rehg†[6], Yoichi Sato†[15], Jianbo Shi†[19], Mike Zheng Shou†[11], Antonio Torralba†[18], Lorenzo Torresani†[1,20], Mingfei Yan†[5], Jitendra Malik[1,8]

[1]Facebook AI Research (FAIR), [2]University of Texas at Austin, [3]University of Minnesota, [4]University of Catania, [5]Facebook Reality Labs, [6]Georgia Tech, [7]Carnegie Mellon University, [8]UC Berkeley, [9]University of Bristol, [10]King Abdullah University of Science and Technology, [11]National University of Singapore, [12]Carnegie Mellon University Africa, [13]Facebook, [14]Universidad de los Andes, [15]University of Tokyo, [16]Indiana University, [17]International Institute of Information Technology, Hyderabad, [18]MIT, [19]University of Pennsylvania, [20]Dartmouth

https://arxiv.org/pdf/2110.07058v1.pdf

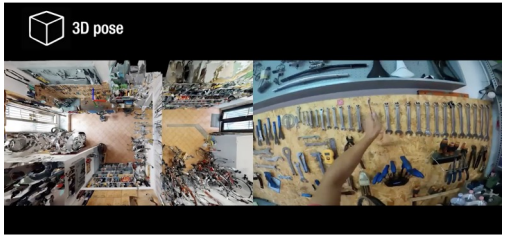




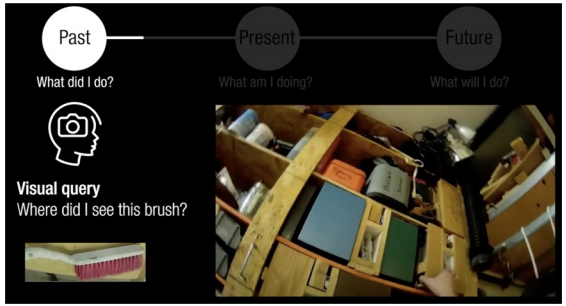


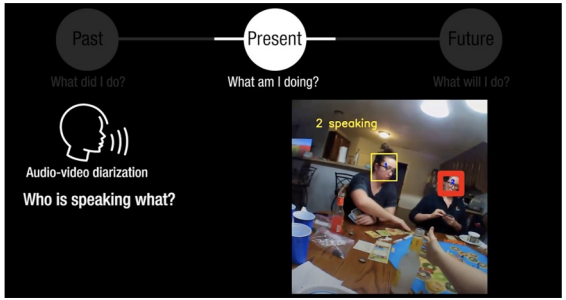
- multimodal dataset

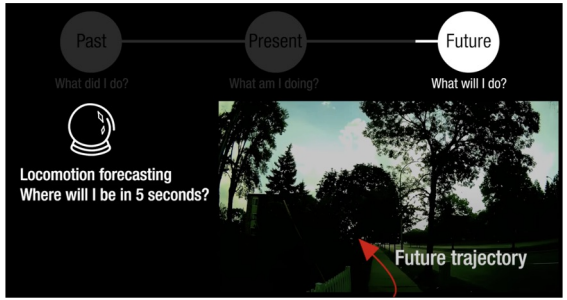
https://ego4d-data.org/



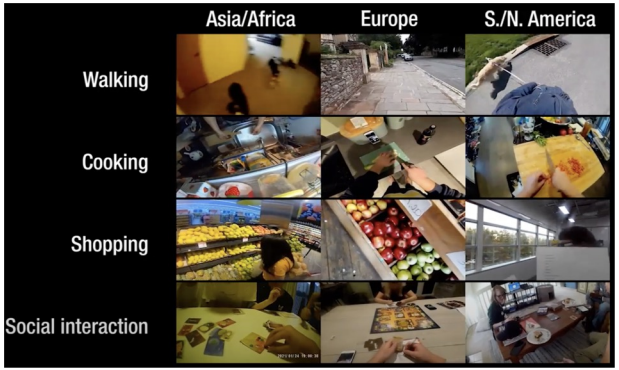
- **Ego4D**
  - 1인칭 시점 비디오 : a massive-scale egocentric video dataset and benchmark suite
  - 전 세계 9개 국가, 74개 지역, 855 명의 고유 카메라 사용자가 촬영한 수백개의 시나리오 (가정, 야외, 직장, 여가 등) 에 걸친 3,025 시간의 일상 활동 비디오 제공
  - 일부 비디오의 경우, 동일한 이벤트에 대한 여러 개의 동기화된 1인칭 시점 제공
    - audio, 3D meshes of the environment, eye gaze, stereo
  - new benchmark challenges
    - visual experience in the past (querying an episodic memory), present (analyzing hand-object manipulation, audio-visual conversation, and social interactions), and future (forecasting activities).

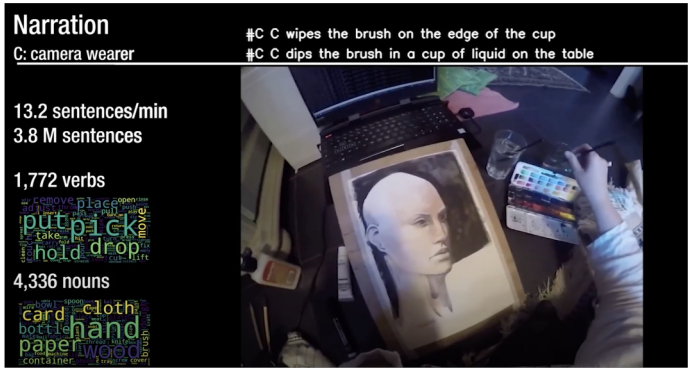




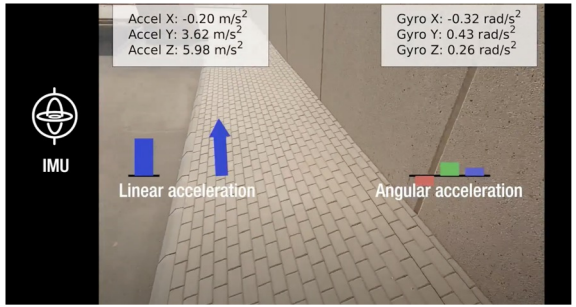


- past - present - future





- 각기 다른 국가에서, 동일한 action 에 대한 1인칭 시점 비디오
- 각각의 비디오는 행동에 대한 나레이션이 포함되어 있음



- IMU (관성 측정 장치)
- Stereo video
- 동기화된 multiple video (captured social interactions)

- eye tracking

End of the Document