ICLR'23 Potpourri

Mert Kilickaya

Notes

ICLR 2023 is in May 2023 | 5000 submissions, 1000 accepted

Not depth-first, but bread-first

Initial Selection of ~150 papers, then down to 10 (dense prediction, learning to learn, new directions…)

I select some papers based on high reviewer scores: Link

Dense Prediction

UNIFIED-IO: A UNIFIED MODEL FOR VISION, LANGUAGE, AND MULTI-MODAL TASKS

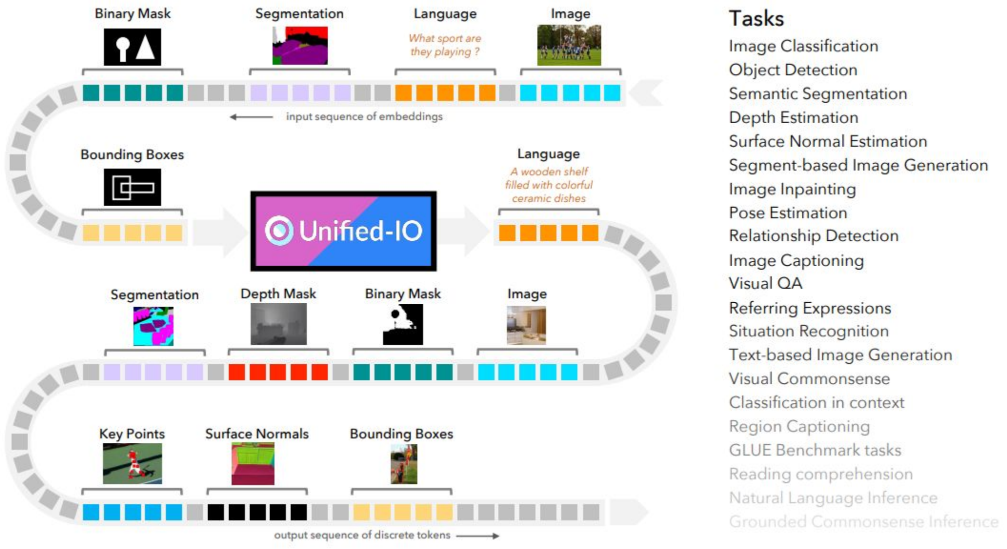


Figure 1: UNIFIED-IO is a single sequence-to-sequence model that performs a variety of tasks in computer vision and NLP using a unified architecture without a need for either task or modality-specific branches. This broad unification is achieved by homogenizing every task's input and output into a sequence of discrete vocabulary tokens. UNIFIED-IO supports modalities as diverse as images, masks, keypoints, boxes, and text, and tasks as varied as depth estimation, inpainting, semantic segmentation, captioning, and reading comprehension.

What:

UNIFIED-IO: A UNIFIED MODEL FOR VISION, LANGUAGE, AND MULTI-MODAL TASKS

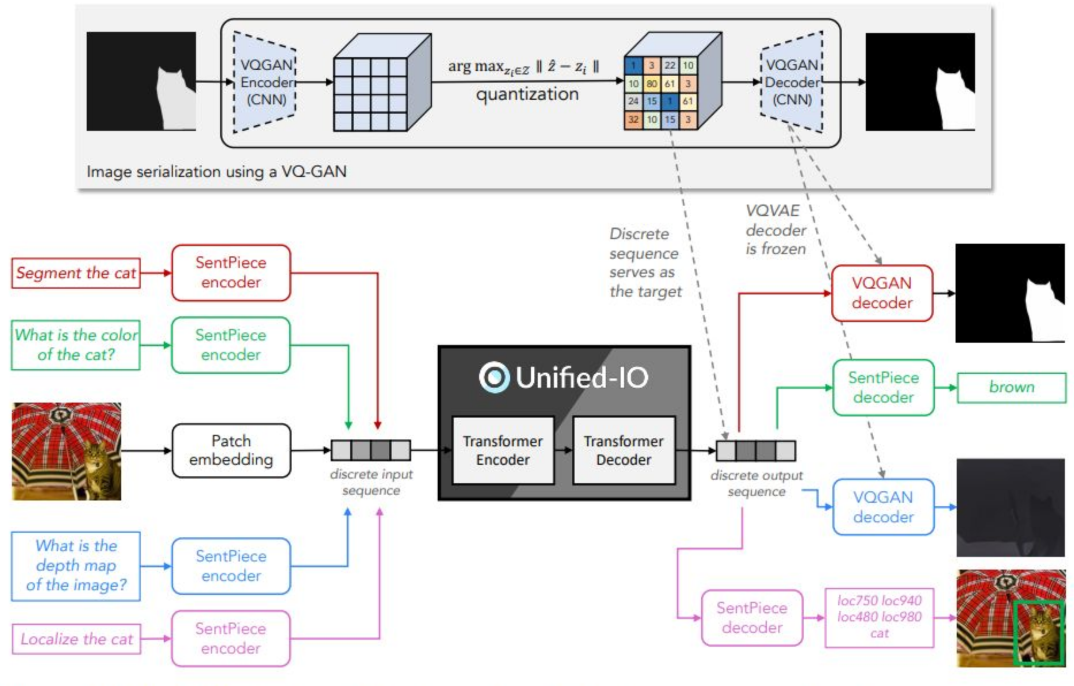
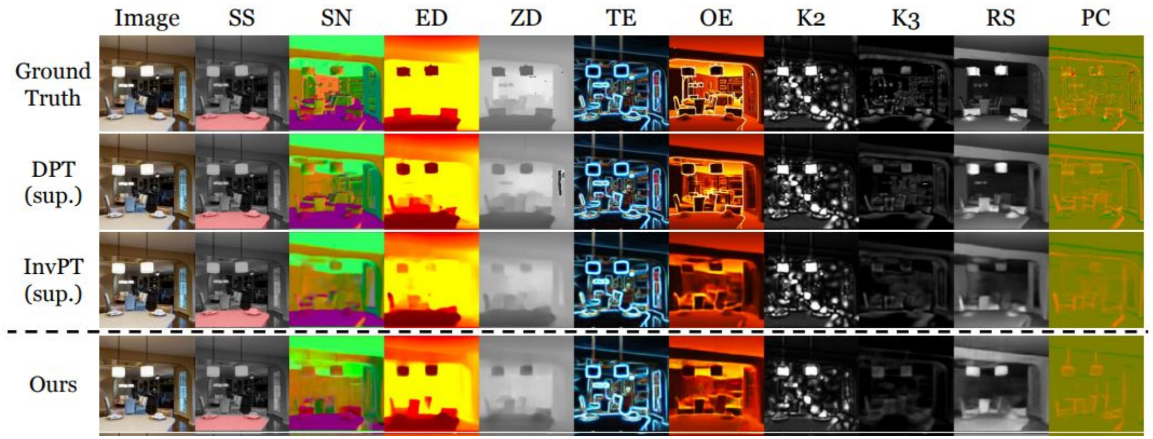


Figure 2: **Unified-IO.** A schematic of the model with four demonstrative tasks: object segmentation, visual question answering, depth estimation and object localization.

How:

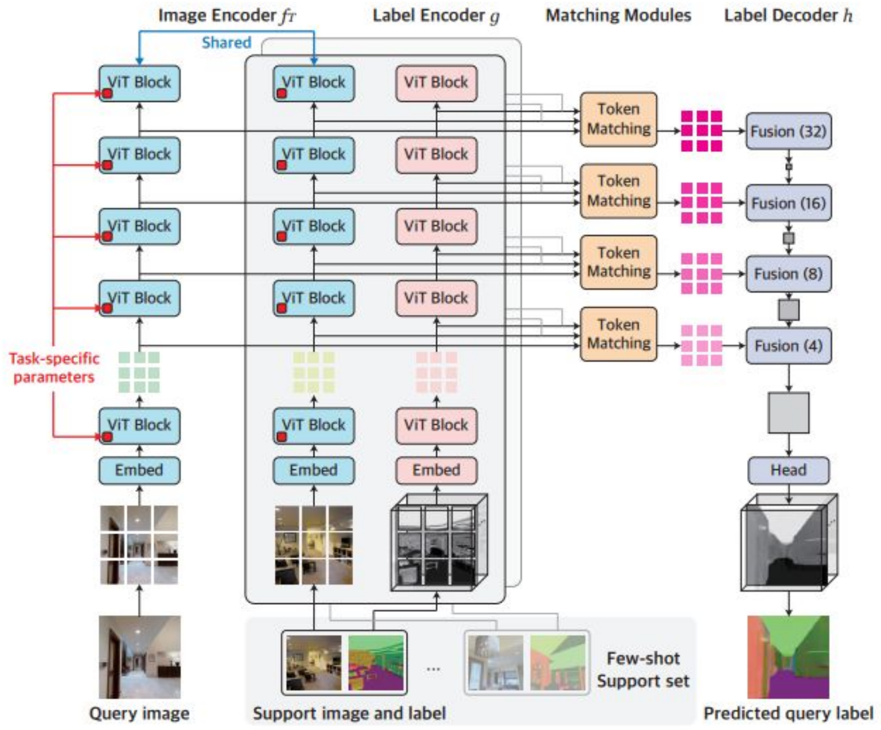
UNIVERSAL FEW-SHOT LEARNING OF DENSE PREDICTION TASKS

SS: Semantic Segmentation	**TE:** Texture Edge	**K2:** Keypoints
SN: Surface Normal	**OE:** Occlusion Edge	**RS:** Reshading



What: Train a single model to solve 10 dense prediction tasks simultaneously, with only few-shots

UNIVERSAL FEW-SHOT LEARNING OF DENSE PREDICTION TASKS



How: Train an image encoder + label encoder | Learn to match image patches to label patches (tokens).

VISION TRANSFORMER ADAPTER FOR DENSE PREDICTIONS

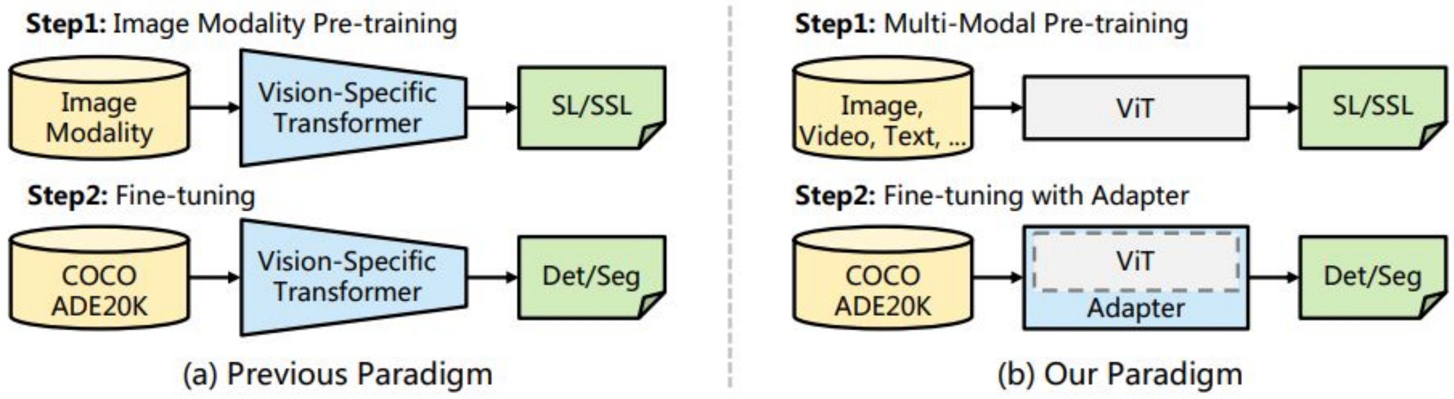

Step1: Image Modality Pre-training
Image Modality
Vision-Specific Transformer
SL/SSL
Step2: Fine-tuning
COCO ADE20K
Vision-Specific Transformer
Det/Seg
(a) Previous Paradigm
Step1: Multi-Modal Pre-training
Image, Video, Text, ...
ViT
SL/SSL
Step2: Fine-tuning with Adapter
COCO ADE20K
ViT
Adapter
Det/Seg
(b) Our Paradigm

Figure 1: **Previous paradigm *vs.* our paradigm.** (a) Previous paradigm designs vision-specific models and pre-trains on large-scale image datasets via supervised or self-supervised learning and then fine-tunes them on downstream tasks. (b) We propose a pre-training-free adapter to close the performance gap between plain ViT (Dosovitskiy et al., 2020) and vision-specific transformers (*e.g.*, Swin (Liu et al., 2021b)) for dense prediction tasks. Compared to the previous paradigm, our method preserves the flexibility of ViT and thus could benefit from advanced multi-modal pre-training.

What: Do not train separate transformers for recognition (ViT)/localization(Swin), adapt a plain ViT instead.

VISION TRANSFORMER ADAPTER FOR DENSE PREDICTIONS

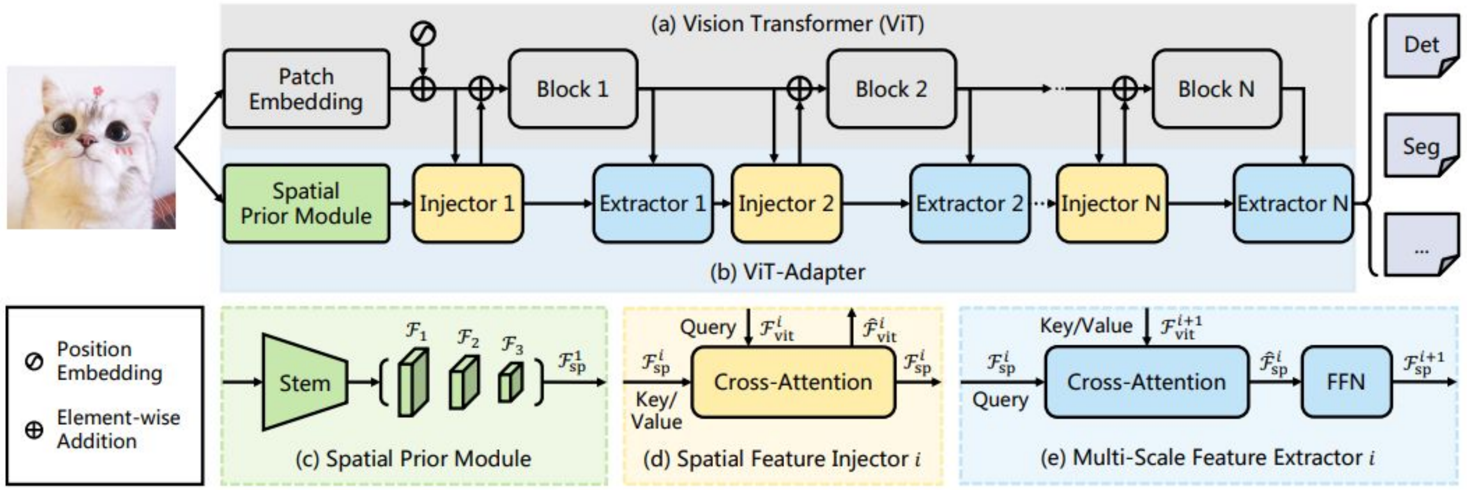
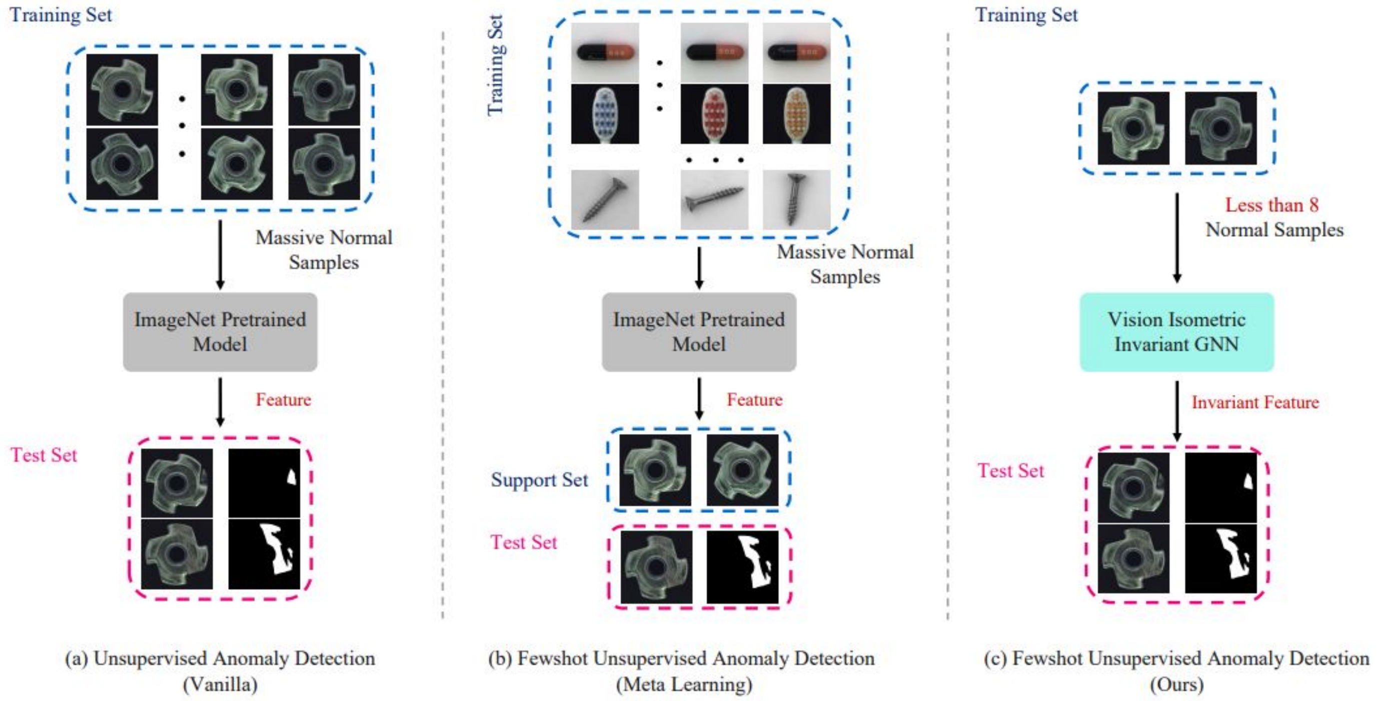


Figure 4: **Overall architecture of ViT-Adapter.** (a) The ViT, whose encoder layers are divided into N (usually $N = 4$) equal blocks for feature interaction. (b) Our ViT-Adapter, which contains three key designs, including (c) a spatial prior module for modeling local spatial contexts from the input image, (d) a spatial feature injector for introducing spatial priors into ViT, and (e) a multi-scale feature extractor for reorganizing multi-scale features from the single-scale features of ViT.

How: Include several blocks to plain ViT (spatial prior module, Extractor/Injector) to perform localization.

PUSHING THE LIMITS OF FEW-SHOT ANOMALY DETECTION IN INDUSTRY VISION: GRAPHCORE


Training Set
Massive Normal Samples
ImageNet Pretrained Model
Feature
Test Set
(a) Unsupervised Anomaly Detection (Vanilla)
Training Set
Massive Normal Samples
ImageNet Pretrained Model
Feature
Support Set
Test Set
(b) Fewshot Unsupervised Anomaly Detection (Meta Learning)
Training Set
Less than 8
Normal Samples
Vision Isometric Invariant GNN
Invariant Feature
Test Set
(c) Fewshot Unsupervised Anomaly Detection (Ours)

What: For visual anomaly detection, reduce the need for high-volume of normal (non-defect) examples.

PUSHING THE LIMITS OF FEW-SHOT ANOMALY DETECTION IN INDUSTRY VISION: GRAPHCORE

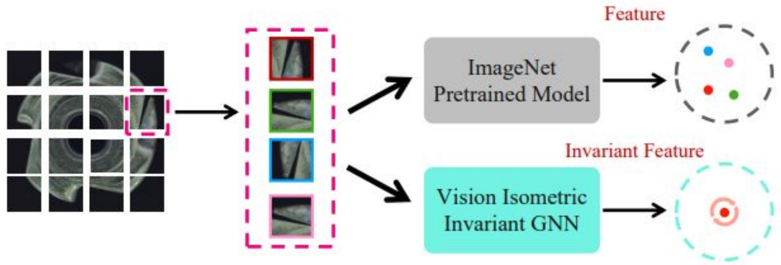


Figure 3: Convolution feature VS vision isometric invariant feature.

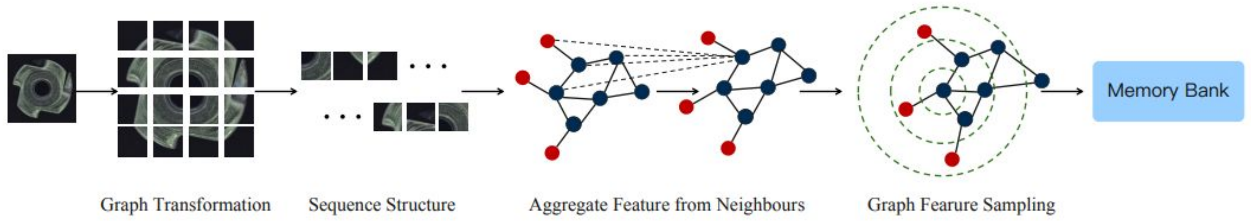
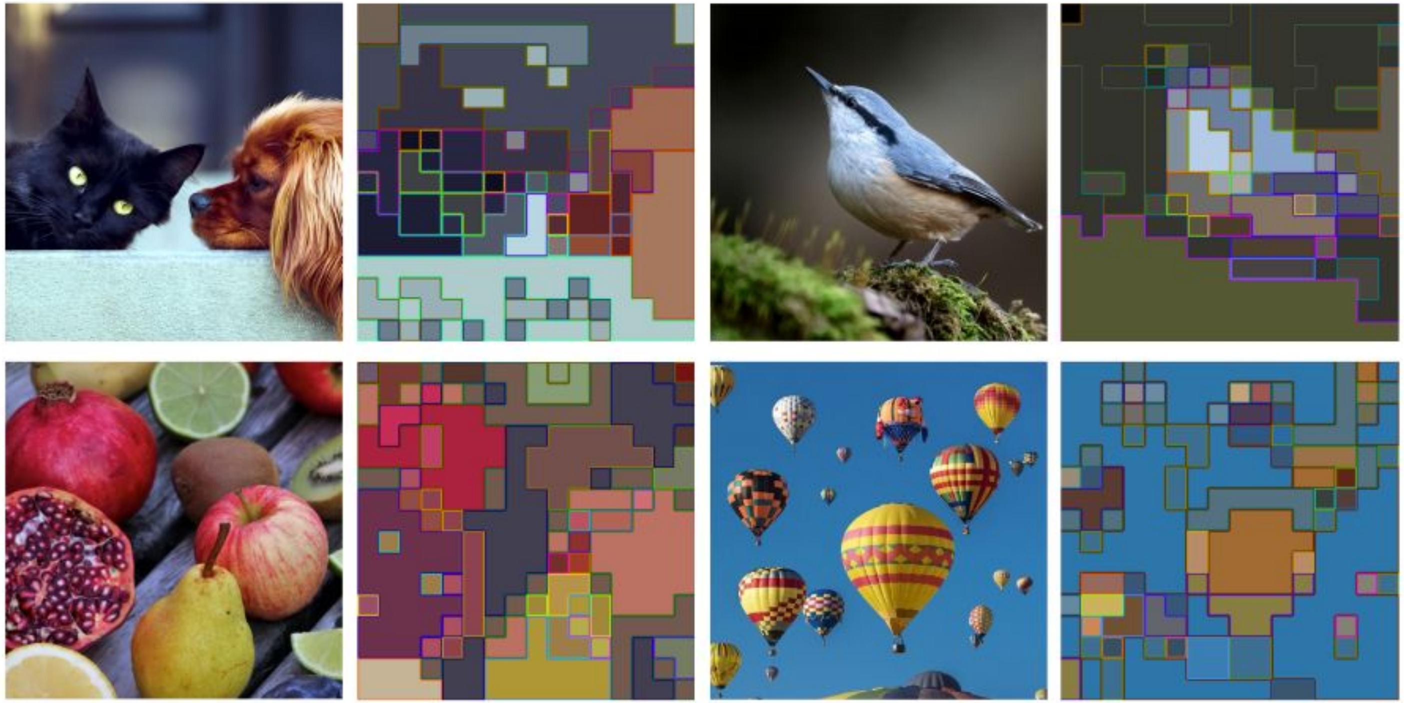


Figure 4: Vision isometric invariant GNN pipeline.

How: Isometric Invariant GNN is strongly invariant to different rotations of the same patch.

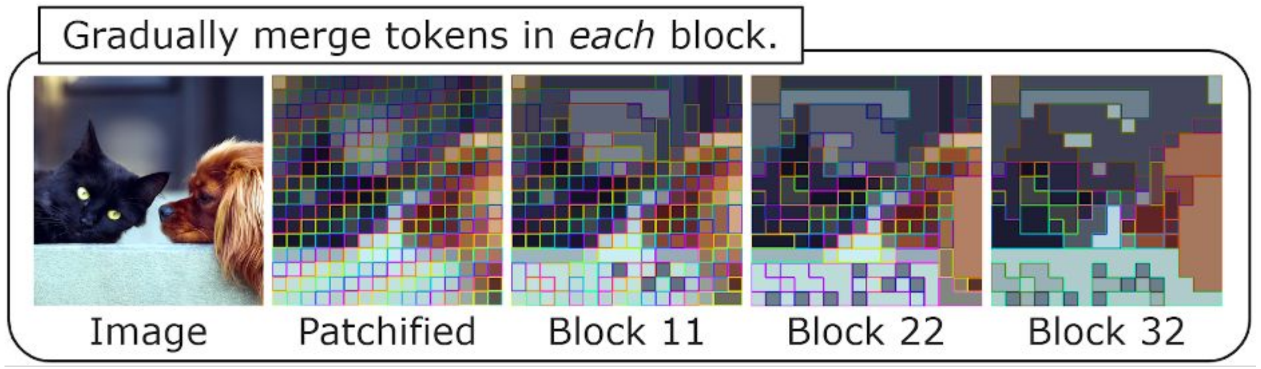
Learning to Learn and Adapt

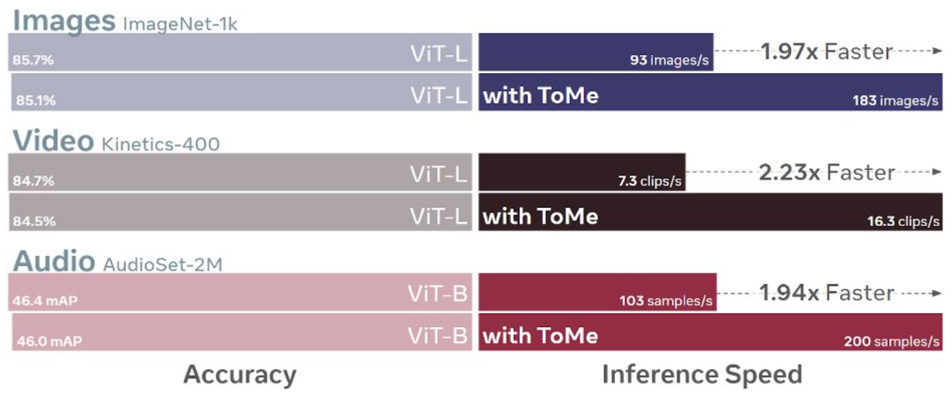
TOKEN MERGING: YOUR VIT BUT FASTER



What: Learn to group similar tokens in a pre-trained ViT to save inference time *without any further training*.

TOKEN MERGING: YOUR VIT BUT FASTER





How: Measure pairwise similarities across patches -> Merge those with similar features.

LEARNING TO GROW PRETRAINED MODELS FOR EFFICIENT TRANSFORMER TRAINING

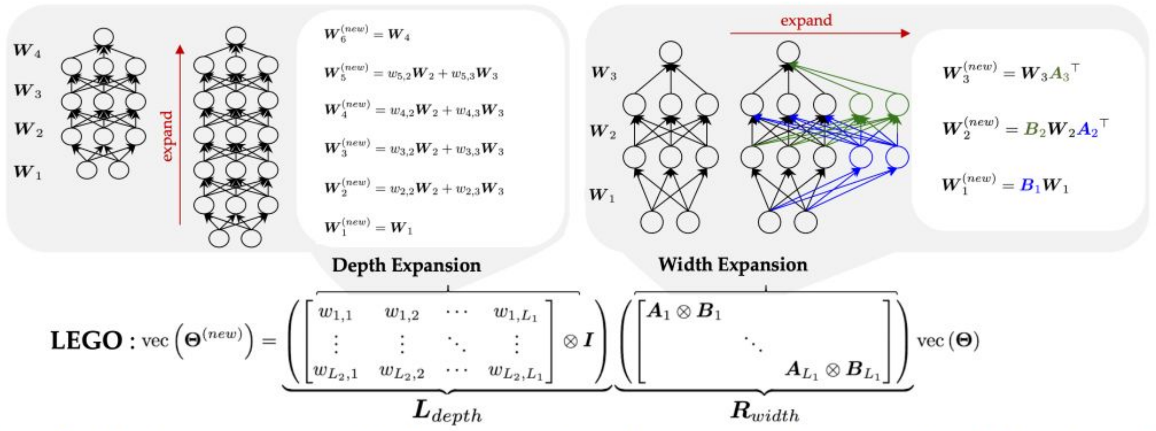

W_4
W_3
W_2
W_1
expand
$W_6^{(new)} = W_4$
$W_5^{(new)} = w_{5,2}W_2 + w_{5,3}W_3$
$W_4^{(new)} = w_{4,2}W_2 + w_{4,3}W_3$
$W_3^{(new)} = w_{3,2}W_2 + w_{3,3}W_3$
$W_2^{(new)} = w_{2,2}W_2 + w_{2,3}W_3$
$W_1^{(new)} = W_1$
Depth Expansion
expand
$W_3^{(new)} = W_3 A_3^\top$
$W_2^{(new)} = B_2 W_2 A_2^\top$
$W_1^{(new)} = B_1 W_1$
Width Expansion
$\text{LEGO}: \text{vec}\left(\Theta^{(new)}\right) = \underbrace{\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,L_1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L_2,1} & w_{L_2,2} & \cdots & w_{L_2,L_1} \end{bmatrix} \otimes I\right)}_{L_{depth}} \underbrace{\left(\begin{bmatrix} A_1 \otimes B_1 & & \\ & \ddots & \\ & & A_{L_1} \otimes B_{L_1} \end{bmatrix}\right)}_{R_{width}} \text{vec}(\Theta)$

Figure 1: Our learning to grow (LEGO) framework accelerates training by using the weights of a smaller model Θ to initialize the weights of the larger model $\Theta^{(new)}$. The LEGO operator is parameterized as a sparse linear map M that can be decomposed into width- and depth-expansion operators. The width-operator R_{width} and depth-operator L_{depth} are structured matrices obtained from Kronecker products of smaller matrices which encode architectural knowledge by grouping parameters into layers and neurons. While we show the expansion operators for simple multi-layer perceptrons for illustrative purposes, in practice we apply LEGO to enable faster training of transformer networks. In our approach, we learn the LEGO matrix M with a 100 steps of SGD, use this to initialize the larger model, and then continue training as usual. Best viewed in color.

What: Learning to initialize a bigger Transformer with much smaller Transformer (both in depth/width).

LEARNING TO GROW PRETRAINED MODELS FOR EFFICIENT TRANSFORMER TRAINING

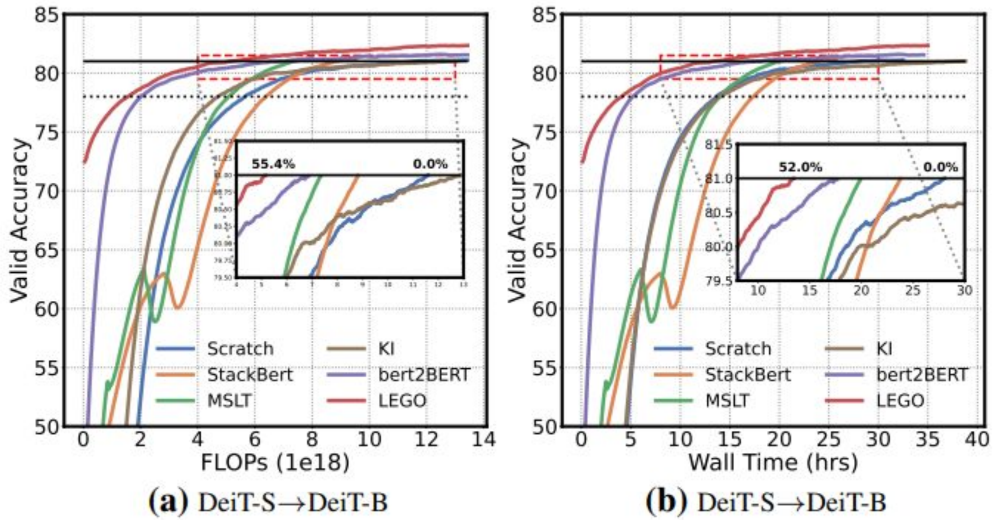


Figure 4: Results on DeiT. (a) accuracy vs. flops and (b) accuracy vs. wall time, for training DeiT-B. LEGO saves flops and wall time by more than 50% over training from scratch on ImageNet.

How: With this learned initialization, a bigger model can be trained 50% faster (12 hours vs. 24 hours).

LEARNING TO PREDICT PARAMETER FOR UNSEEN DATA

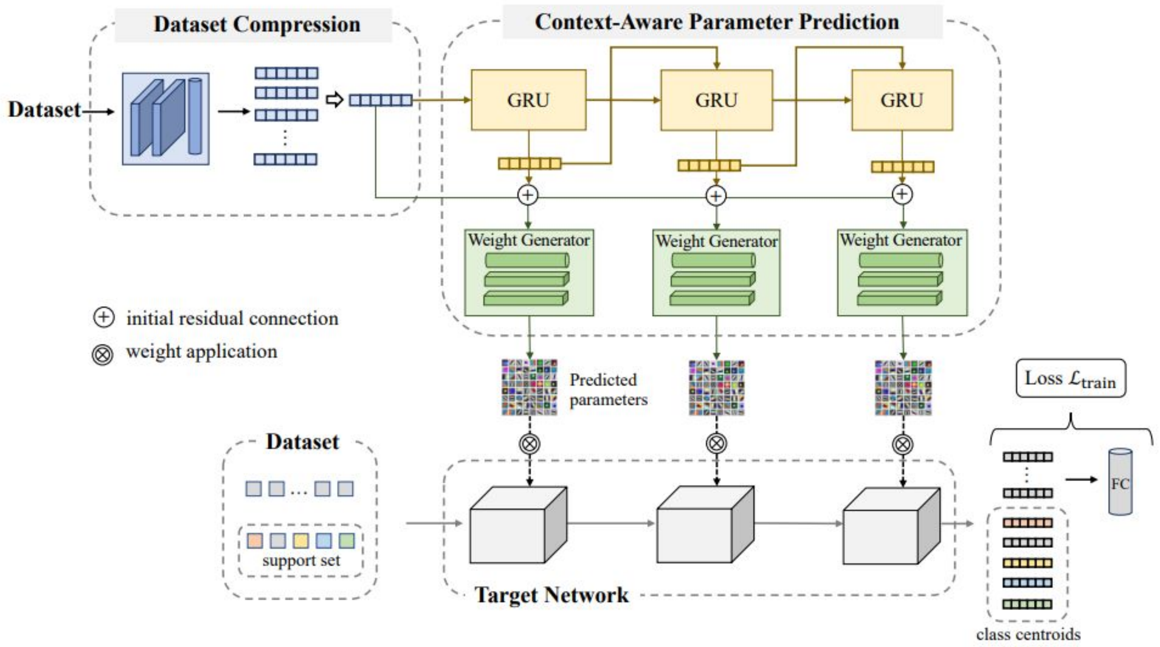


Figure 2: Overview of our proposed PudNet. PudNet first compresses each dataset into a sketch with a fixed size, and then utilizes the hypernetwork to generate parameters of a target network based on the sketch. Finally, PudNet is optimized based on a support set in a meta-learning based manner.

What: Train a hyper-network to generate the weights of another network based on the incoming dataset.

Novel Ideas

UNDERSTANDING SELF-SUPERVISED PRETRAINING WITH PART-AWARE REPRESENTATION LEARNING

Contrastive Self-Supervised Learning

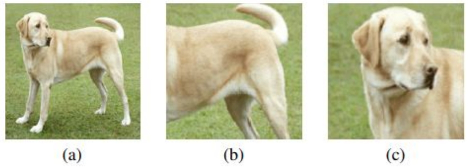
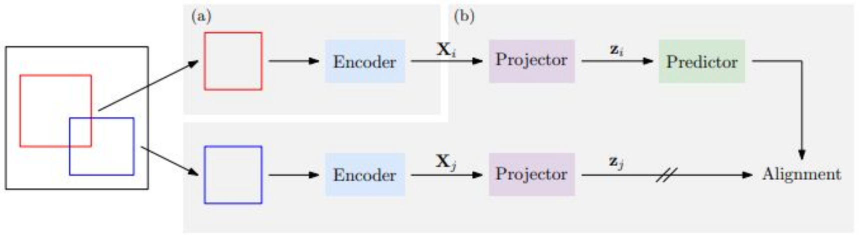


Figure 1: (a) original image, (b-c) two random crops, ar



Masked Image Modelling for Self-Supervised Learning

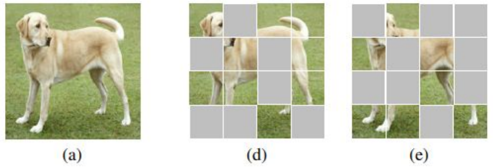
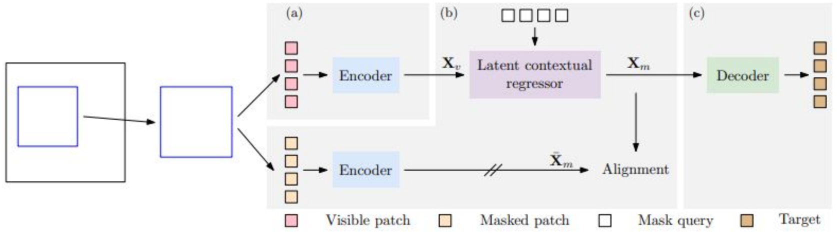
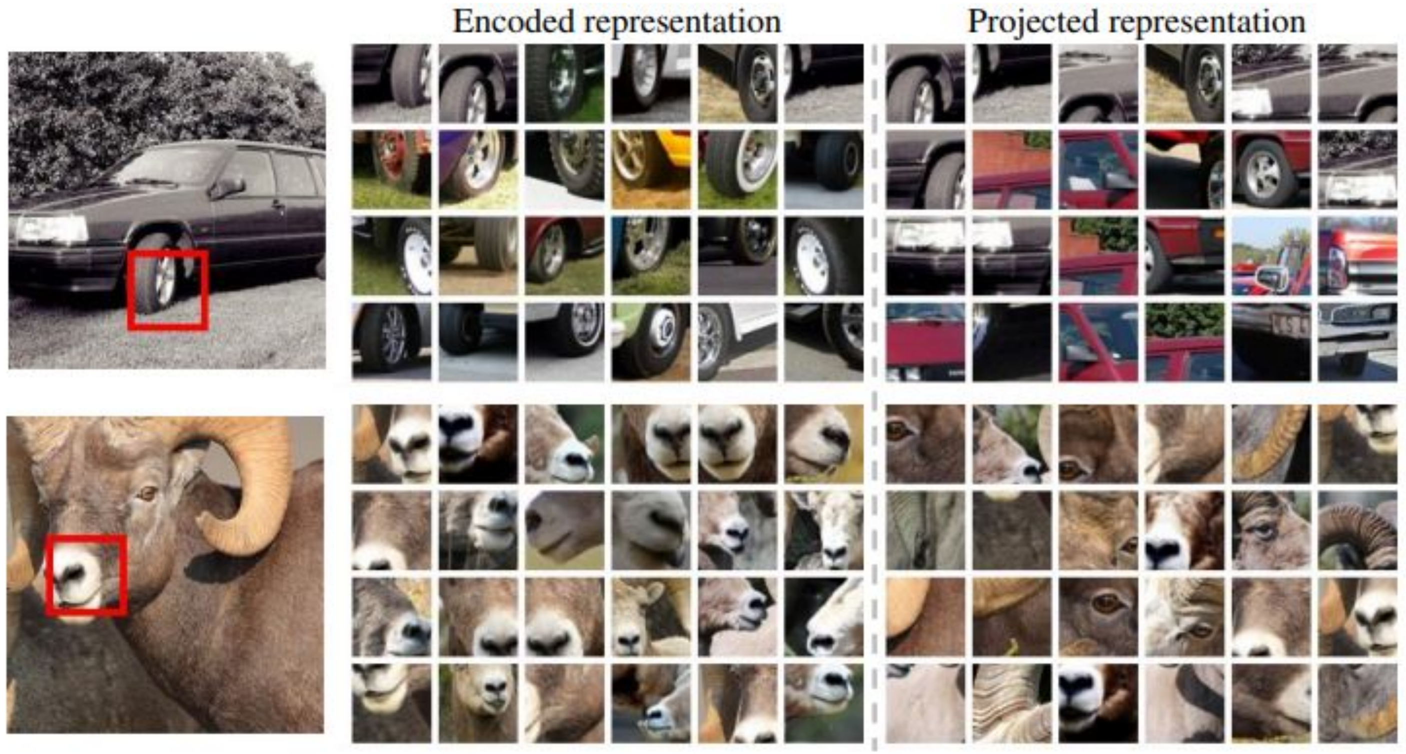


Figure 1: (a) o d (d-e) masked and visible patches.



What: Self-supervised learning models either: 1) Part-to-whole, 2) Whole-to-part representations.

UNDERSTANDING SELF-SUPERVISED PRETRAINING WITH PART-AWARE REPRESENTATION LEARNING



How: See how encoded representation focuses on the same part/projected representation other parts.

LANGUAGE MODELLING WITH PIXELS

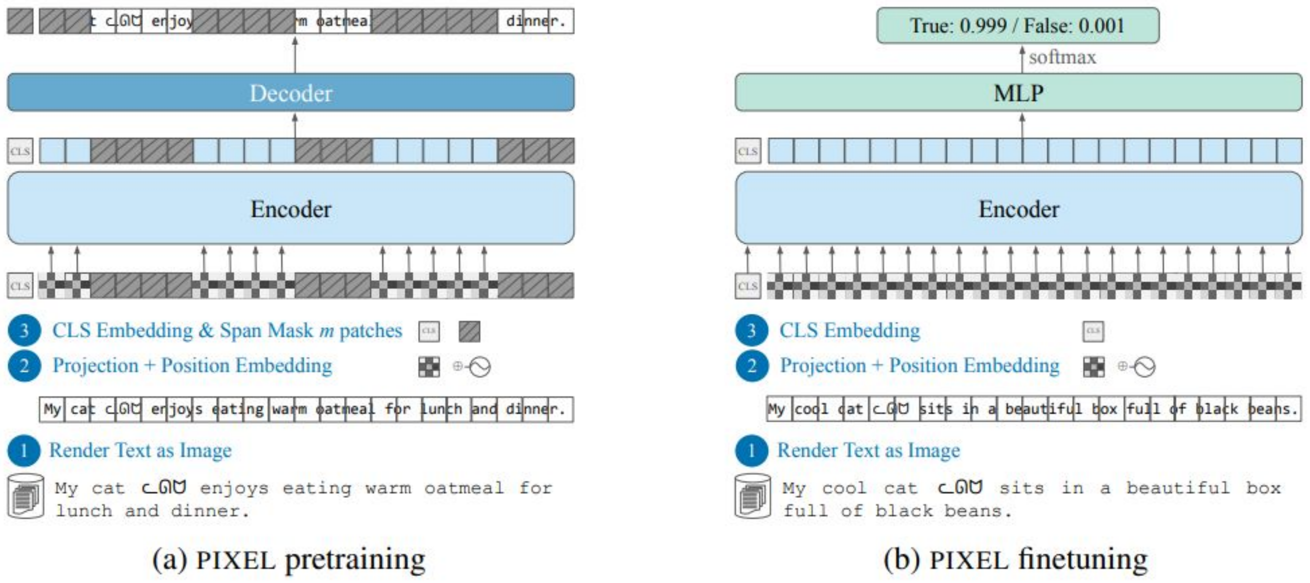
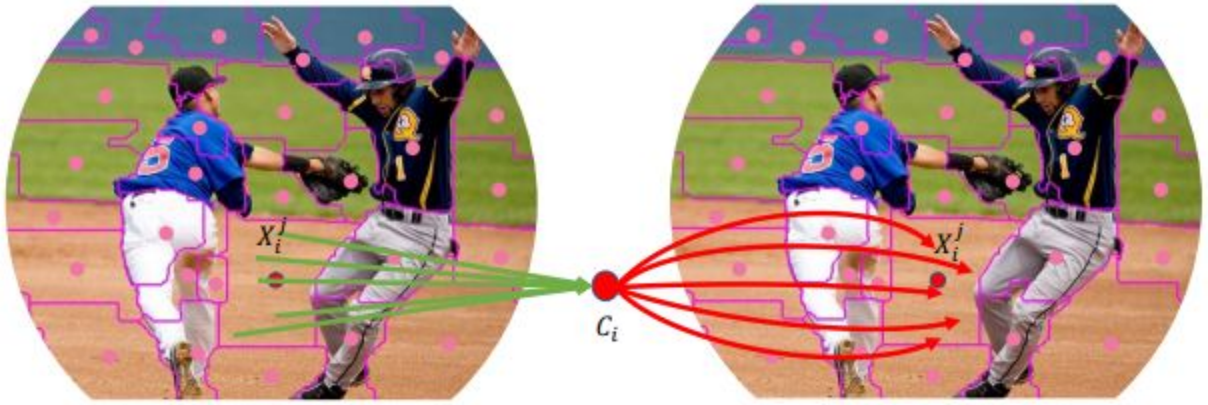

Decoder
CLS
Encoder
3 CLS Embedding & Span Mask m patches
2 Projection + Position Embedding
My cat ᓚᘏᗢ enjoys eating warm oatmeal for lunch and dinner.
1 Render Text as Image
My cat ᓚᘏᗢ enjoys eating warm oatmeal for lunch and dinner.
(a) PIXEL pretraining
True: 0.999 / False: 0.001
softmax
MLP
3 CLS Embedding
My cool cat ᓚᘏᗢ sits in a beautiful box full of black beans.
(b) PIXEL finetuning

Figure 1: Overview of PIXEL's architecture. Following He et al. (2022), we use a masked autoencoder with a ViT architecture and a lightweight decoder for pretraining (left). At finetuning time (right), the decoder is replaced by a task-specific classification head that sits on top of the encoder.

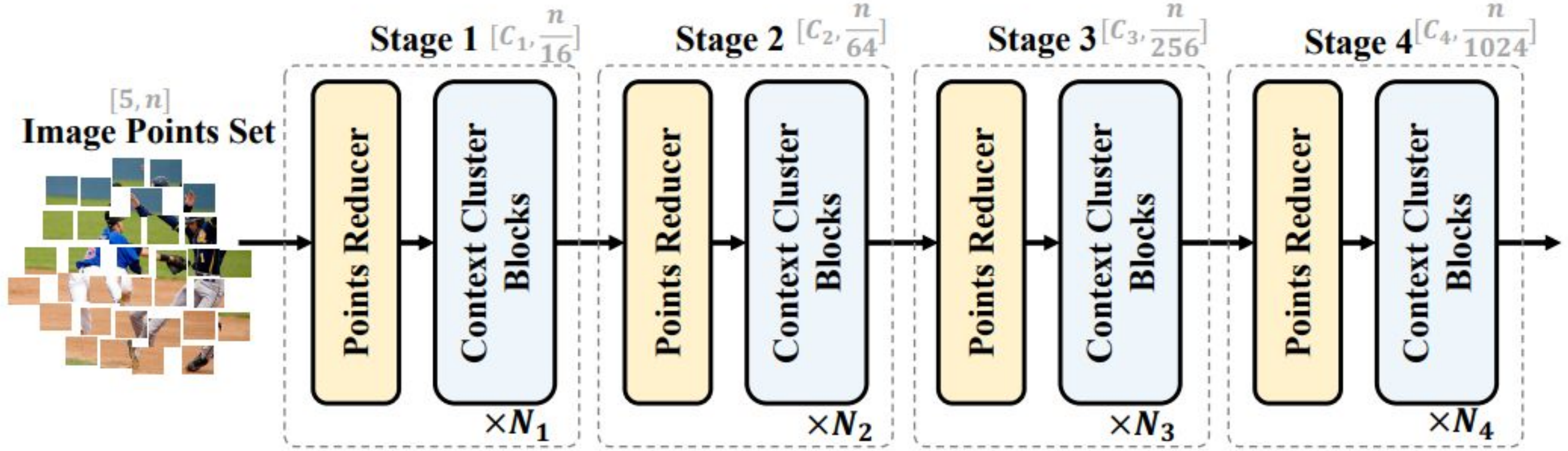
What: Instead of encoding language as distinct word tokens, just turn them into an image.

Image as Set of Points



What: Instead of processing an input image point-by-point, group similar pixels, and jointly process groups.

Image as Set of Points

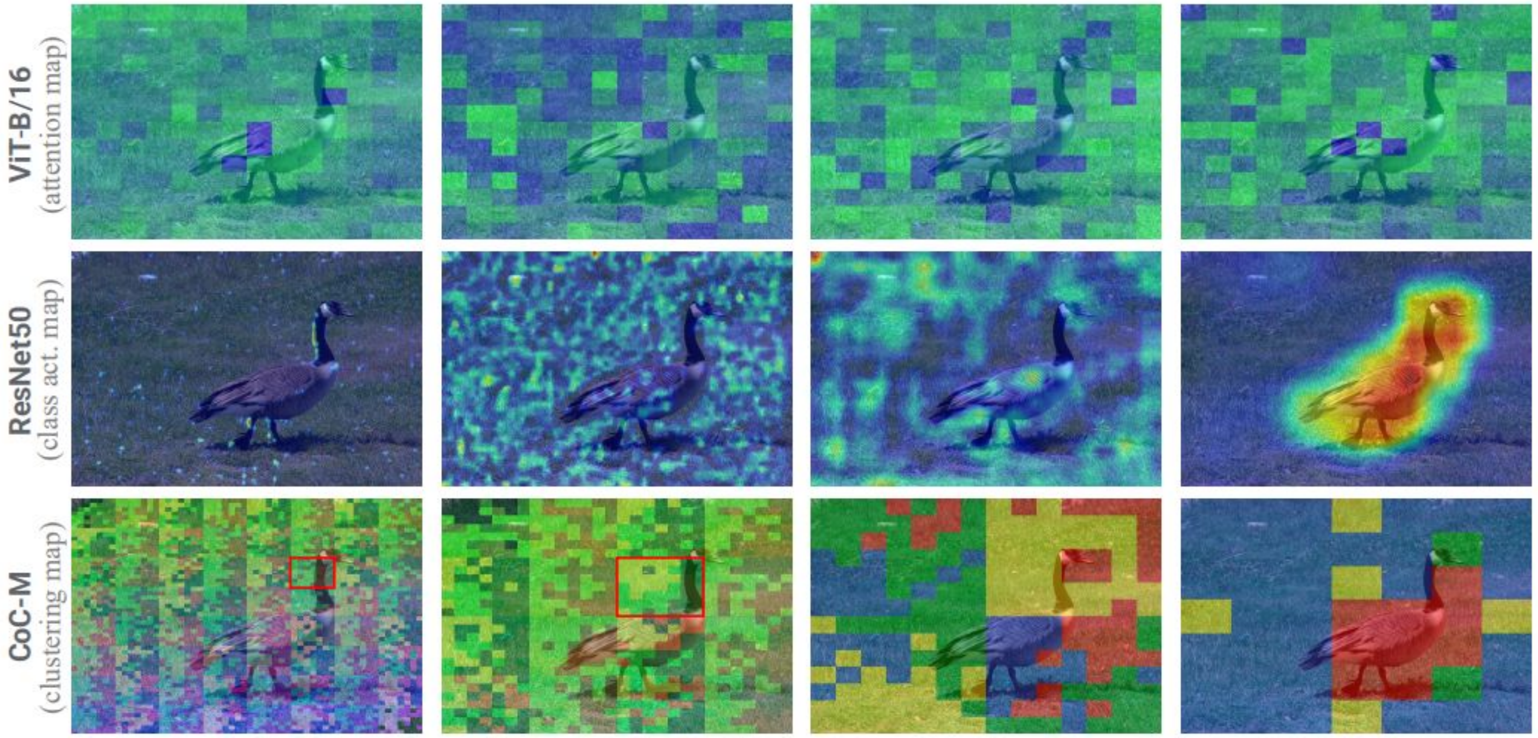


Stage 1 $[C_1, \frac{n}{16}]$
Stage 2 $[C_2, \frac{n}{64}]$
Stage 3 $[C_3, \frac{n}{256}]$
Stage 4 $[C_4, \frac{n}{1024}]$
$[5, n]$
Image Points Set
Points Reducer
Context Cluster Blocks
$\times N_1$
$\times N_2$
$\times N_3$
$\times N_4$

Figure 3: Context Cluster architecture with four stages. Given a set of image points, Context Cluster gradually reduces the point number and extracts deep features. Each stage begins with a points reducer, after which a succession of context cluster blocks is used to extract features.

How: No convolution | No attention | Only clustering blocks + MLP layers | On par performance.

Image as Set of Points



How: See how CoC learns to group (cluster) similar patches together (duck-to-duck, grass-to-grass, etc.)

Discussion

Unification of tasks/models: Converging to single model for all/many tasks?

Converging to Transformer-like architectures?

Training networks to generate (data-specific) networks/weights rather than directly tackling tasks?

Check out my ICLR’22 Potpourri also (time goes too fast): Link

Reach out for: Clarifications, Research ideas, Anything: kilickayamert@gmail.com, https://kilickaya.github.io/