Mastering Transformers: From Building Blocks to Real-World Applications

Transformers in Computer Vision

Prof. Alptekin Temizel
Graduate School of Informatics, Middle East Technical University
13 Sept 2023

Computer Vision Tasks

Image Classification

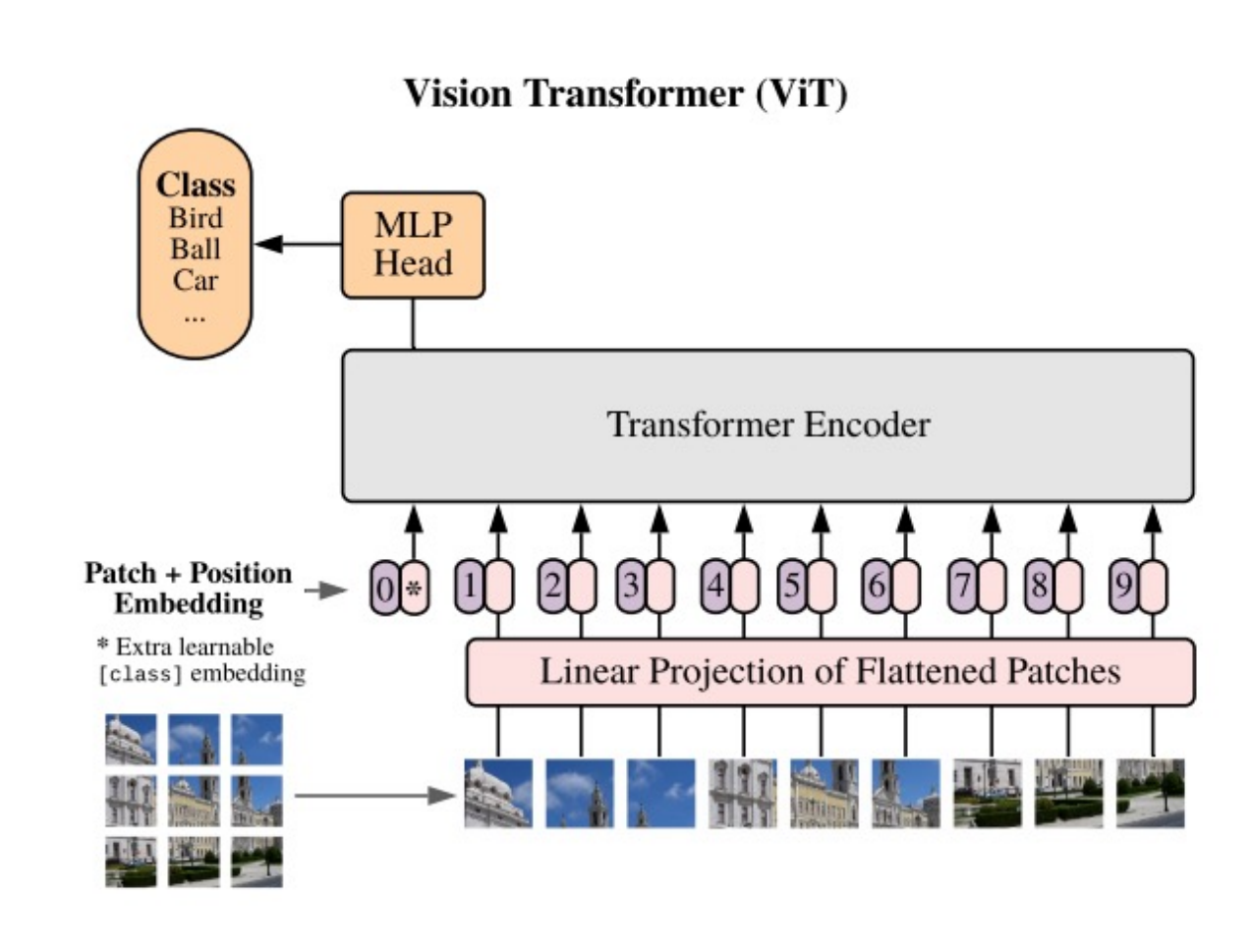
Object Detection

Image Segmentation

(Examples are from cs231n lecture from Stanford University)

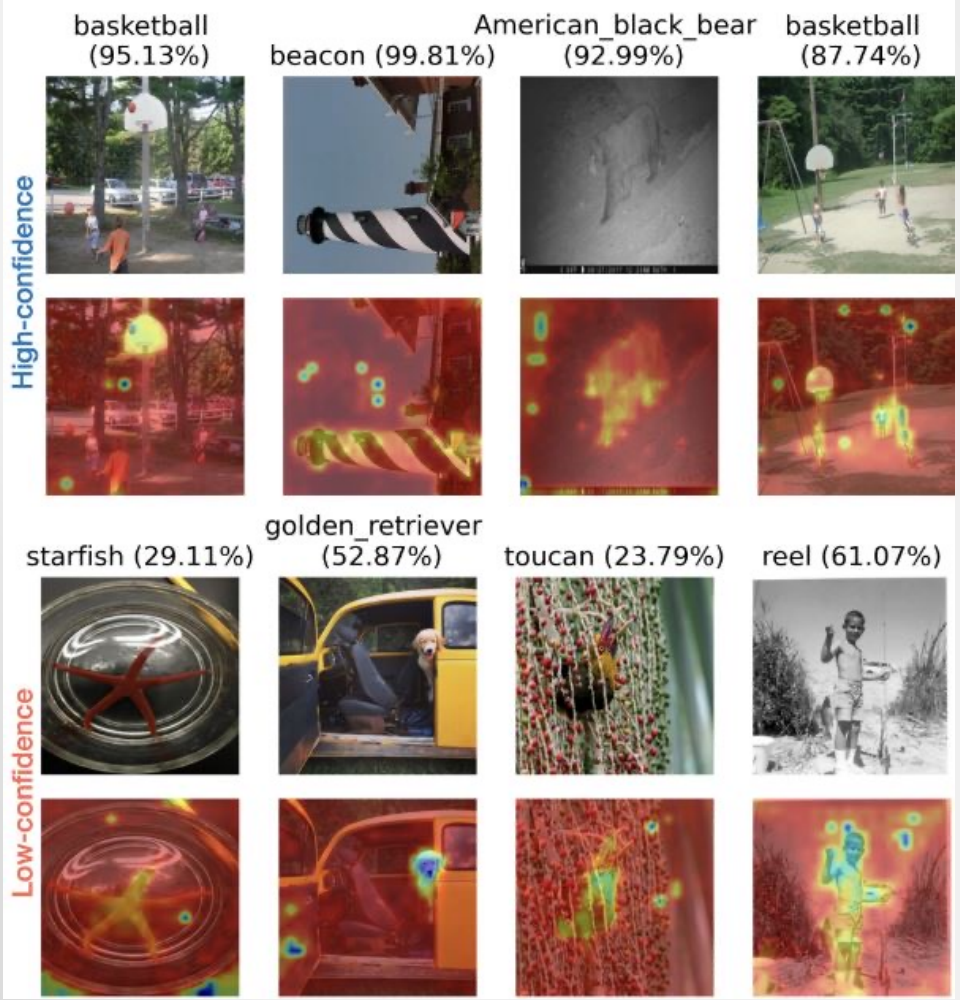
The Vision Transformer - ViT

- ViT, is a model for image classification that employs a Transformer-like architecture over patches of the image.
- An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder.
- In order to perform classification, the standard approach of adding an extra learnable “classification token” to the sequence is used.



Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

The Vision Transformer - ViT



Vision Transformers vs. CNNs

Inductive biases allow the learning solution to prioritise one solution over another, independent of the observed data.

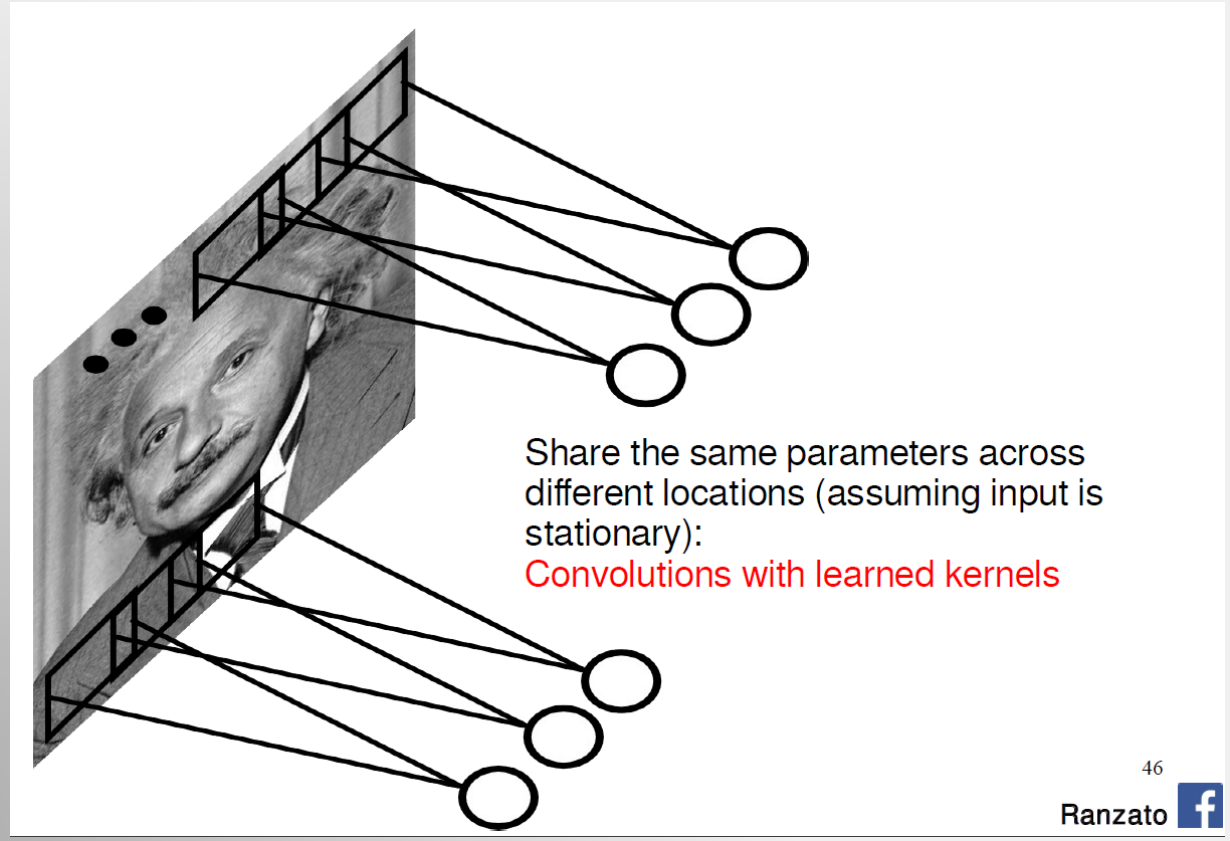
- By constraining the models, it reduces the space of solutions without sacrificing key metrics thus improving performance.
- It allows us to encode our understanding of the world to search for viable solutions. This lets us steers the model generalization the way we want.

https://medium.com/mlearning-ai/everything-you-need-to-know-about-inductive-bias-4850b77048f2

Vision Transformers vs. CNNs

In convolutional neural networks : The inductive bias is on **locality** i.e the features are generated using local pixels and then combined **hierarchically.**

Translational equivariance is another inductive bias for CNNs



Idea: statistics are similar at different locations - stationarity (Lecun 1998)

Connect each hidden unit to a small input patch and share the weight across space: convolution layer.

https://medium.com/mlearning-ai/everything-you-need-to-know-about-inductive-bias-4850b77048f2

Vision Transformers vs. CNNs

- CNNs achieve excellent results even with training based on data volumes that are not as large as those required by Vision Transformers due to **inductive biases** that can be somehow exploited by these networks to grasp more quickly the particularities of the analysed images even if, on the other hand, they end up limiting them making it more complex to grasp global relations.
- On the other hand, the Vision Transformers are free from these biases which leads them to be able to capture also global and wider range relations but at the cost of a more onerous training in terms of data.
- Vision Transformers also proved to be much more robust to input image distortions such as adversarial patches or permutations.
- Hybrid architectures combining convolutional layers with Vision Transformers have shown to provide excellent results in Computer Vision.
- On the other hand, weaker inductive bias results in increased reliance on model regularization or data augmentation when training on smaller datasets.

Coccomini, Davide, 2021, "Vision Transformers or Convolutional Neural Networks? Both!". *Towards Data Science*.

ViT Model Sizes

ViT-B (Base): 12 layers, hidden size of 768, 86M parameters.

ViT-L (Large): 24 layers, hidden size of 1024, 307M parameters.

ViT-H (Huge): 32 layers, hidden size of 1280, 632M parameters.

ViT Variants
Data-efficient Image Transformer - DeiT

The original paper concluded that transformers “do not generalize well when trained on insufficient amounts of data”, and the training of these models involve extensive computing resources.

DeiT (data-efficient image transformers) are more efficiently trained transformers for image classification, requiring far less data and far less computing.

DeiT is trained using a teacher-student strategy specific to transformers.

It relies on a distillation token ensuring that the student learns from the teacher through attention.

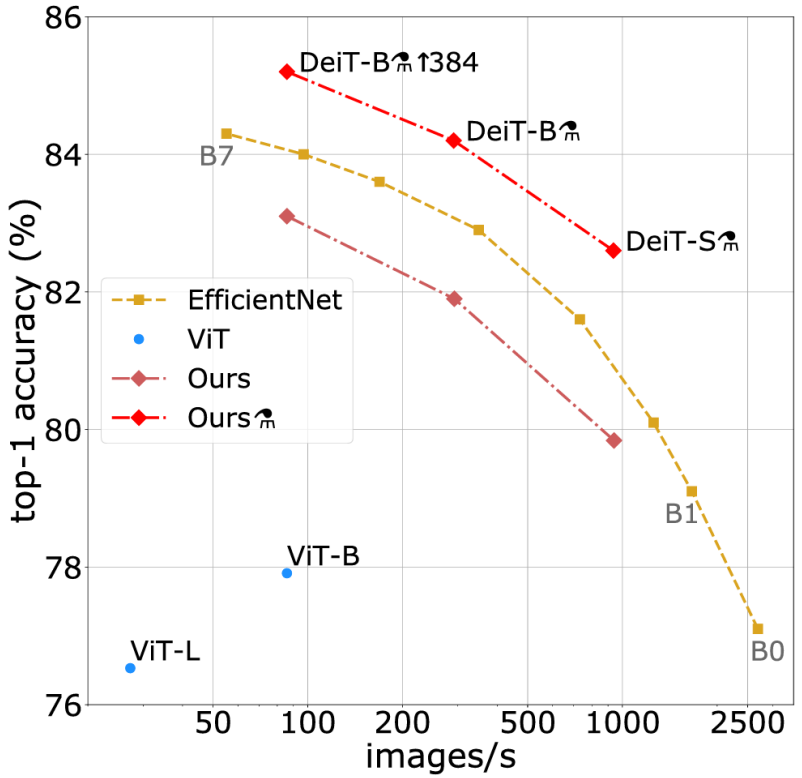


Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to VIT-B, but the training is more adapted to a data-starving regime. It is learned in a few days on one machine. The symbol ⚗ refers to models trained with our transformer-specific distillation. See Table 5 for details and more models.

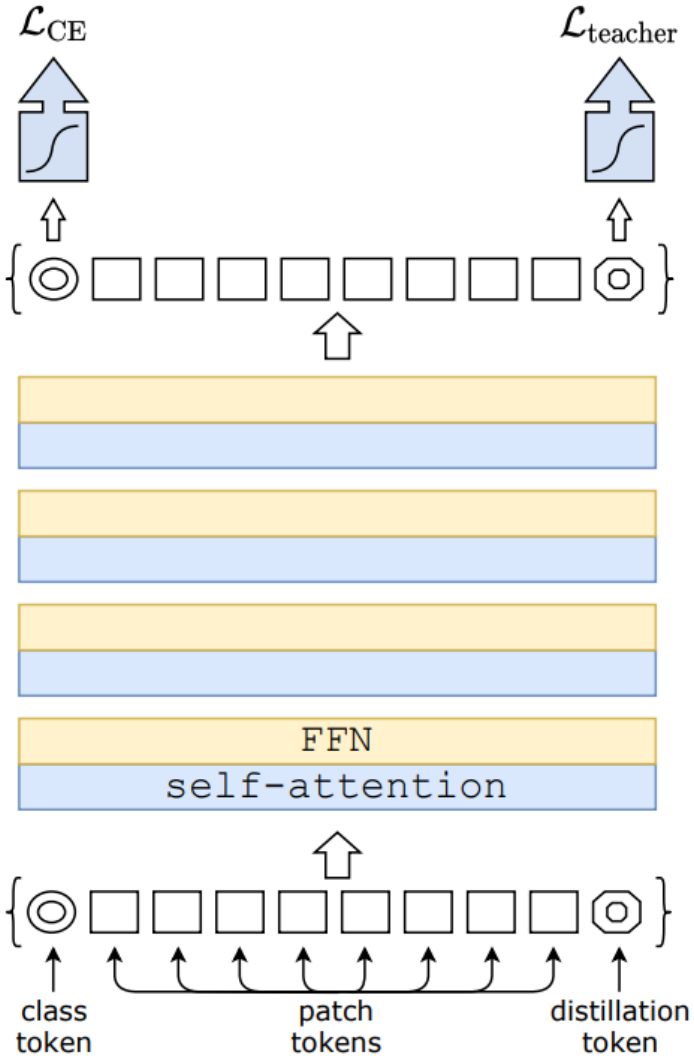
Touvron et al., Training data-efficient image transformers & distillation through attention. ICML, 2021

ViT Variants
Data-efficient Image Transformer - DeiT

Distillation token interacts with the class token and patch tokens through the self-attention layers.

It is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce the (hard)label predicted by the teacher, instead of true label.

Both the class and distillation tokens input to the transformers are learned by back-propagation.



Touvron et al., Training data-efficient image transformers & distillation through attention. ICML, 2021

ViT Variants

Swin Transformer: Hierarchical Vision Transformer Using **S**hifted **WIN**dows

Challenges in adapting Transformer from language to vision arise from differences between the two domains, such as large variations in the scale of visual entities and the high resolution of pixels in images compared to words in text.

To address these differences, a hierarchical Transformer whose representation is computed with Shifted Windows is proposed.

The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection.

Liu et al, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021.

ViT Variants

Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows

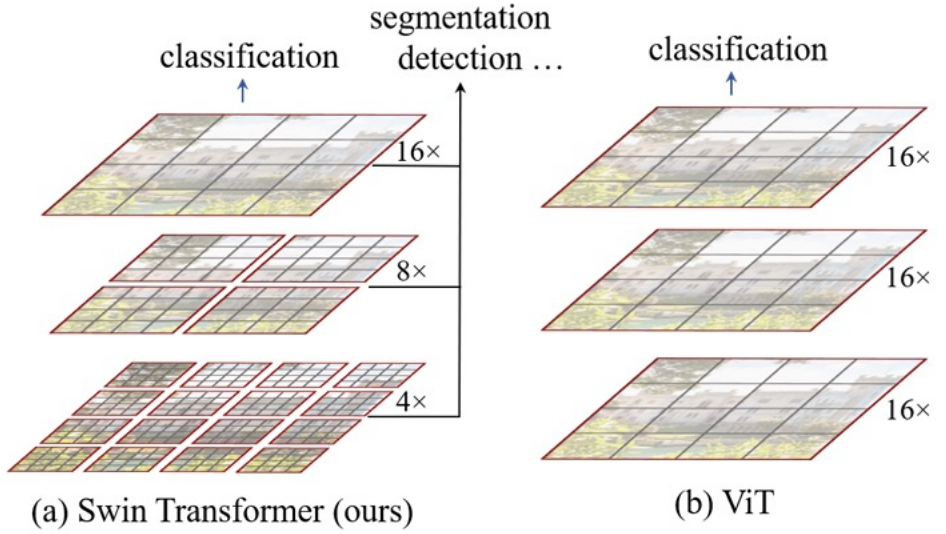


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [19] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

Liu et al, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021.

ViT Variants
DINO - Self-**DI**stillation with **NO** labels

DINO (a method for self-supervised training of Vision Transformers) by Facebook AI. Vision Transformers trained using the DINO method show very interesting properties not seen with convolutional models. They are capable of segmenting objects, without having ever been trained to do so.

Caron et al., Emerging Properties in Self-Supervised Vision Transformers, ICCV 2021.

ViT Variants
Masked AutoEncoder

- The paper shows that masked autoencoders (MAE) are scalable self-supervised learners for computer vision
- Random patches of the input image are masked and missing pixels are reconstructed.
- An asymmetric encoder-decoder architecture, with an encoder that operates only on the visible subset of patches (without mask tokens), along with a lightweight decoder that reconstructs the original image from the latent representation and mask tokens.
- Masking a high proportion of the input image, e.g., 75%, yields a nontrivial and meaningful self-supervisory task.

He et al., Masked Autoencoders Are Scalable Vision Learners, CVPR 2022.

ViT Variants
Masked AutoEncoder

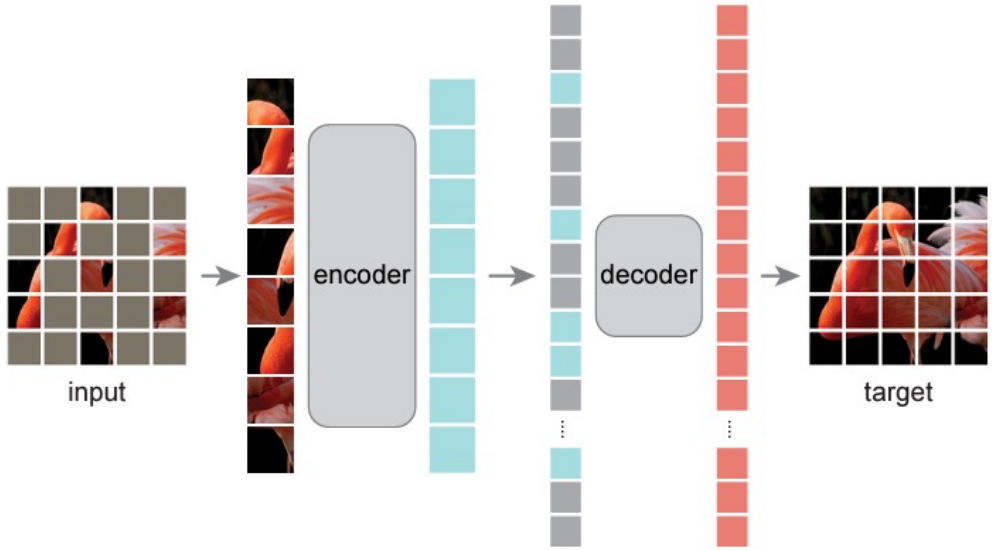
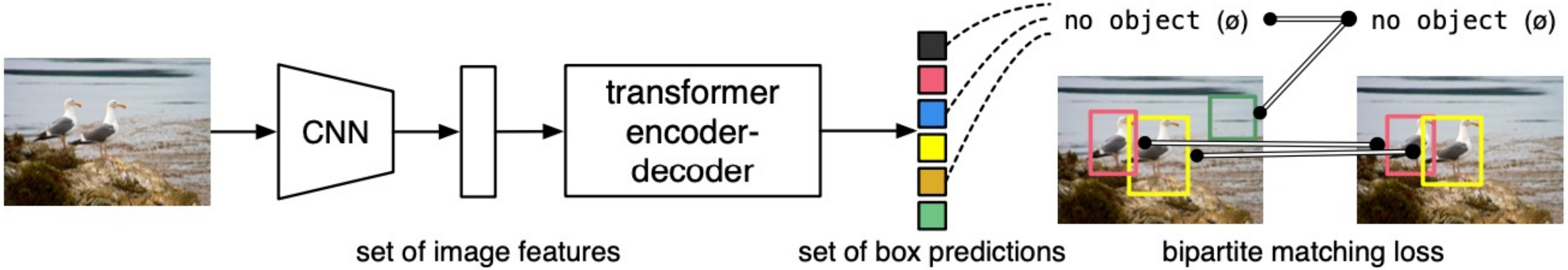


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

He et al., Masked Autoencoders Are Scalable Vision Learners, CVPR 2022.

Object Detection
DEtection TRansformer (DETR)

- DETR introduces transformers to object detection tasks by reframing detection as a set prediction problem, eliminating the need for proposal generation and post-processing steps.
- Initially, despite competitive performance, DETR suffered from slow training convergence and ineffective detection of smaller objects.



Object Detection
DEtection TRansformer (DETR)

- Initially, despite competitive performance, DETR suffered from slow training convergence and ineffective detection of smaller objects.

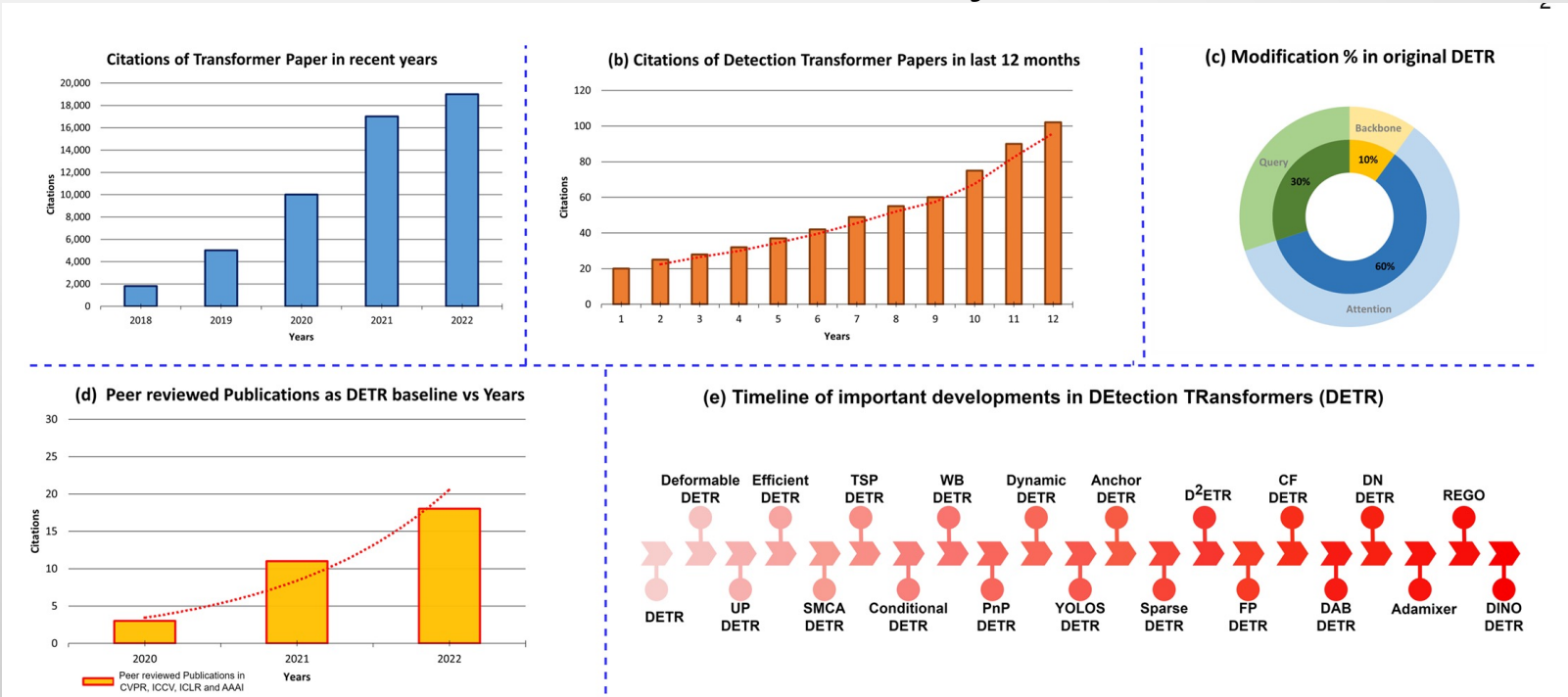


Fig. 1: Statistics overview of the literature on Transformers. (a) Number of citations per year of Transformer papers. (b) Citations in the last 12 months on Detection Transformer papers. (c) Modification percentage in original DEtection TRansformer (DETR) to improve performance and training convergence (d) Number of peer-reviewed publications per year that used DETR as a baseline. (e) A non-exhaustive timeline overview of important developments in DETR for detection tasks.

Shehzadi et al., 2D Object Detection with Transformers: A Review. *arXiv preprint arXiv:2306.04670, Jun. 2023.*

Object Detection
Deformable-DETR

Deformable DETR mitigates the slow convergence issues and limited feature spatial resolution of the original DETR by leveraging a new deformable attention module which only attends to a small set of key sampling points around a reference.

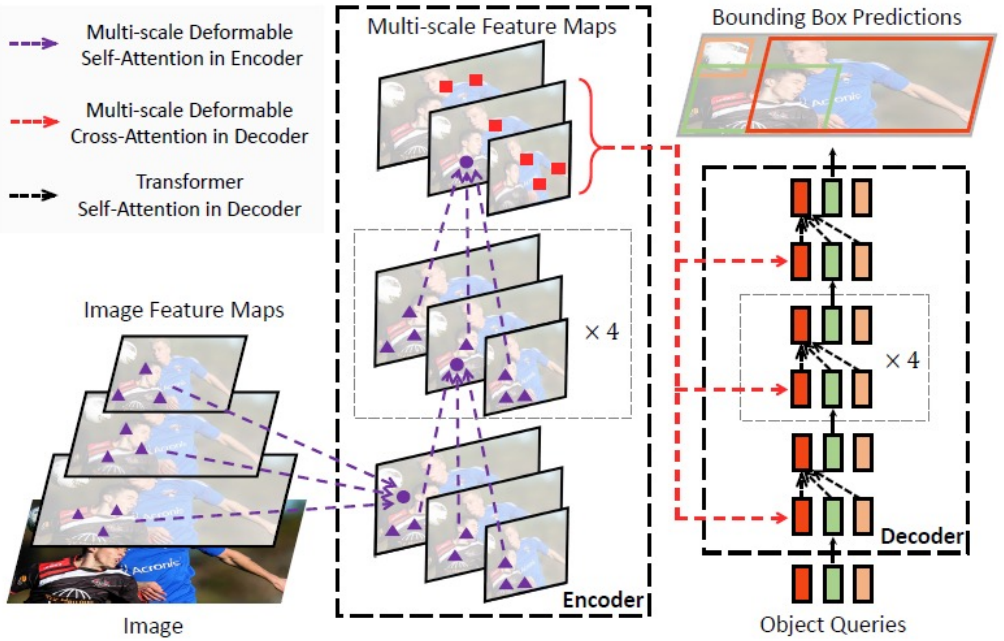


Figure 1: Illustration of the proposed Deformable DETR object detector.

Zhu et al., Deformable DETR: Deformable Transformers for End-to-End Object Detection, Arxiv Preprint, 2020

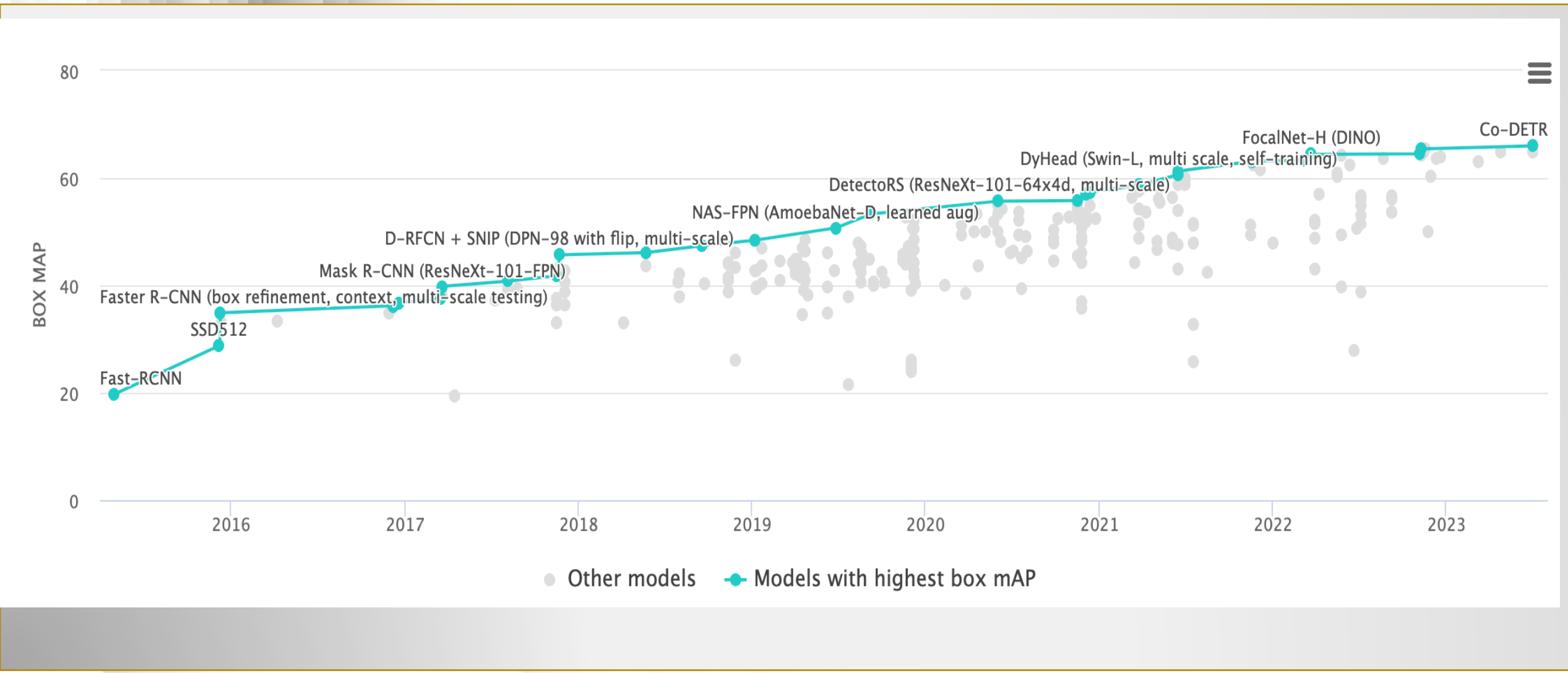
Object Detection
DINO-DETR

DETR with **I**mproved de**N**oising anch**O**r boxes (DINO) improves over previous DETR-like models in performance and efficiency by

- using a contrastive way for denoising training
- a mixed query selection method for anchor initialization
- a look forward twice scheme for box prediction.

Zhang et al., DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection, ICLR 2023

Object Detection State-of-the-Art



https://paperswithcode.com/sota/object-detection-on-coco

Object Detection
Co-Deformable-DETR

“Too few queries assigned as positive samples in DETR with one-to-one set matching leads to sparse supervision on the encoder's output which considerably hurt the discriminative feature learning of the encoder and vice visa for attention learning in the decoder.

To alleviate this, we present a novel collaborative hybrid assignments training scheme, namely Co-DETR, to learn more efficient and effective DETR-based detectors from versatile label assignment manners. This new training scheme can easily enhance the encoder's learning ability in end-to-end detectors by training the multiple parallel auxiliary heads supervised by one-to-many label assignments such as ATSS and Faster RCNN. In addition, we conduct extra customized positive queries by extracting the positive coordinates from these auxiliary heads to improve the training efficiency of positive samples in the decoder. In inference, these auxiliary heads are discarded and thus our method introduces no additional parameters and computational cost to the original detector while requiring no hand-crafted non-maximum suppression (NMS).

Zong et al., DETRs with collaborative hybrid assignments training, Oct. 2023, *ICCV*

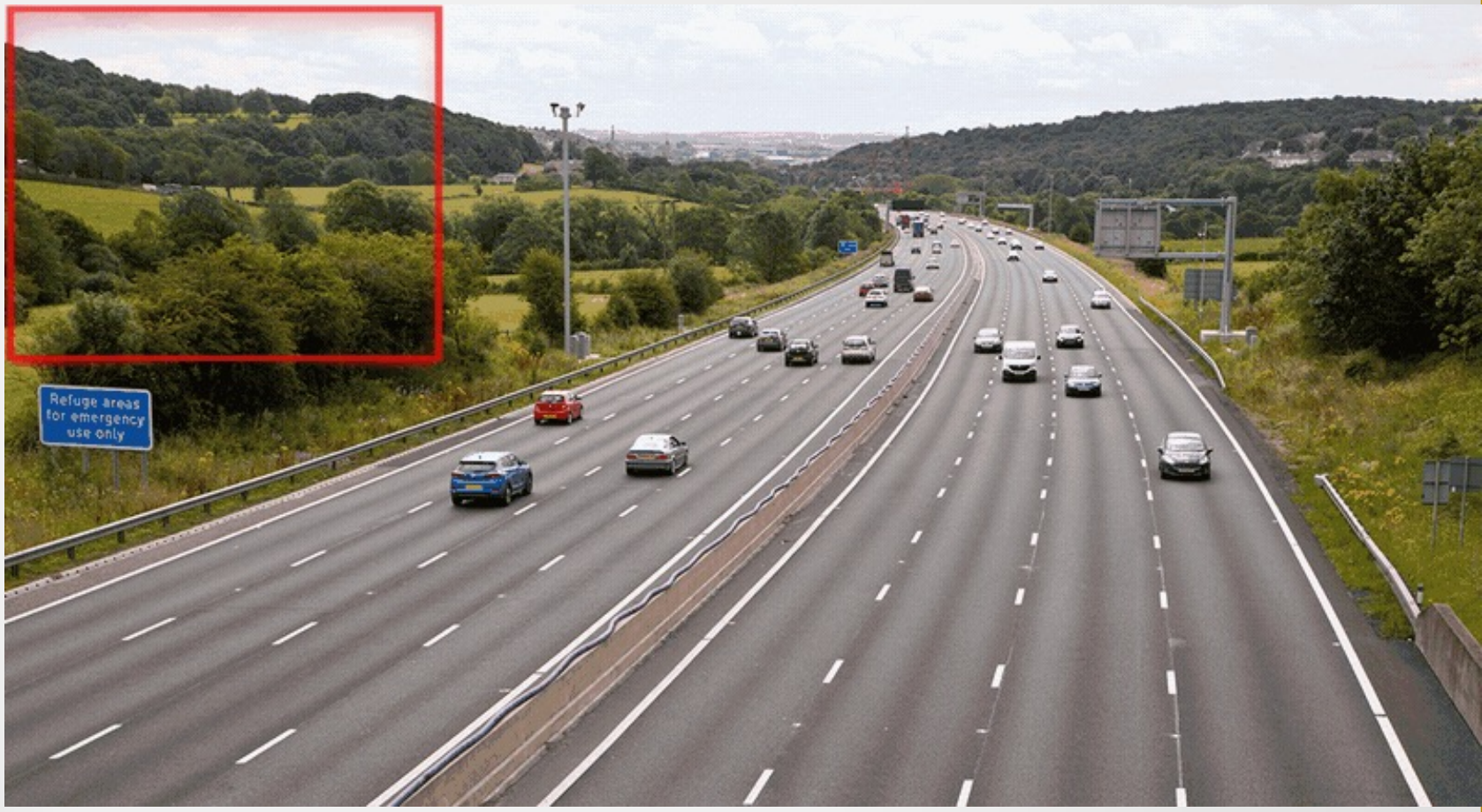
Slicing Aided Hyper Inference (SAHI)

Large scale object detection & instance segmentation

Over 3000 stars on Github

PyPI link

https://pypi.org/project/sahi



F.C. Akyon, S.O. Altinuc, A. Temizel, “Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection”, ICIP, Oct. 2022.

Slicing Aided Hyper Inference (SAHI)
DETR Integration

- Install latest version of SAHI and Huggingface transformers:

```
!pip install -U sahi
!pip install transformers timm
```

```
import os
os.getcwd()
```

- Import required modules:

```
# import required functions, classes
from sahi import AutoDetectionModel
from sahi.predict import get_sliced_prediction, predict, get_prediction
from sahi.utils.file import download_from_url
from sahi.utils.cv import read_image
from IPython.display import Image
```

You can see object detection models available at HF model hub.

```
# Select a model to use, we use a DETR model.
model_path = "facebook/detr-resnet-50"

# download test images into demo_data folder
download_from_url('https://raw.githubusercontent.com/obss/sahi/main/demo/demo_data/small-vehicles1.jpeg'
download_from_url('https://raw.githubusercontent.com/obss/sahi/main/demo/demo_data/terrain2.png', 'demo_
```

SAHI Notebook with Huggingface (HF) Transformers
https://github.com/obss/sahi/blob/main/demo/inference_for_huggingface.ipynb

Slicing Aided Hyper Inference (SAHI)
DETR Integration

- To perform sliced prediction we need to specify slice parameters. In this example we will perform prediction over slices of 512x512 with an overlap ratio of 0.2:

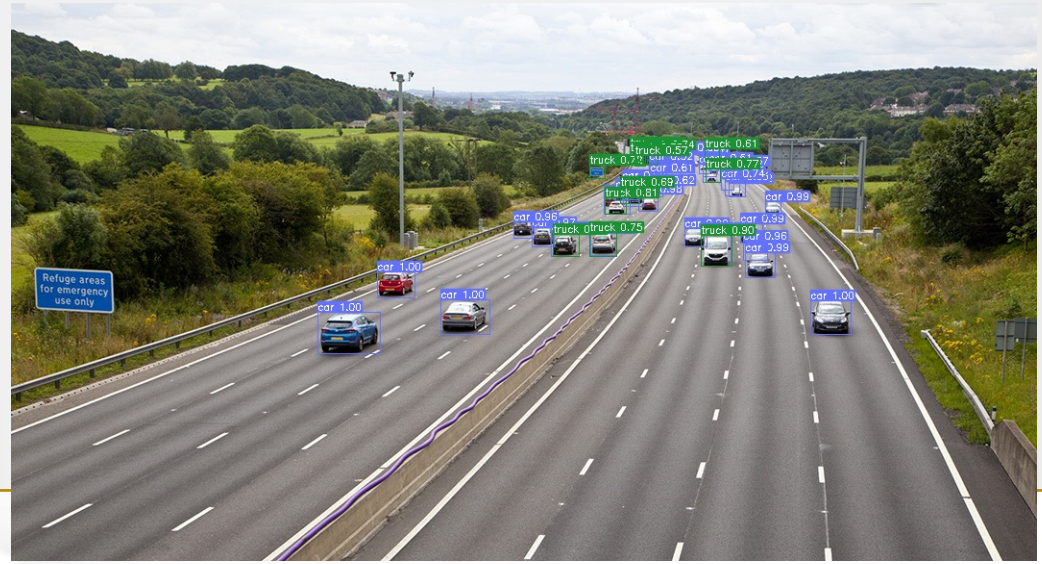
```
result = get_sliced_prediction(
    "demo_data/small-vehicles1.jpeg",
    detection_model,
    slice_height = 512,
    slice_width = 512,
    overlap_height_ratio = 0.2,
    overlap_width_ratio = 0.2,
)
```

```
Performing prediction on 6 number of slices.
```

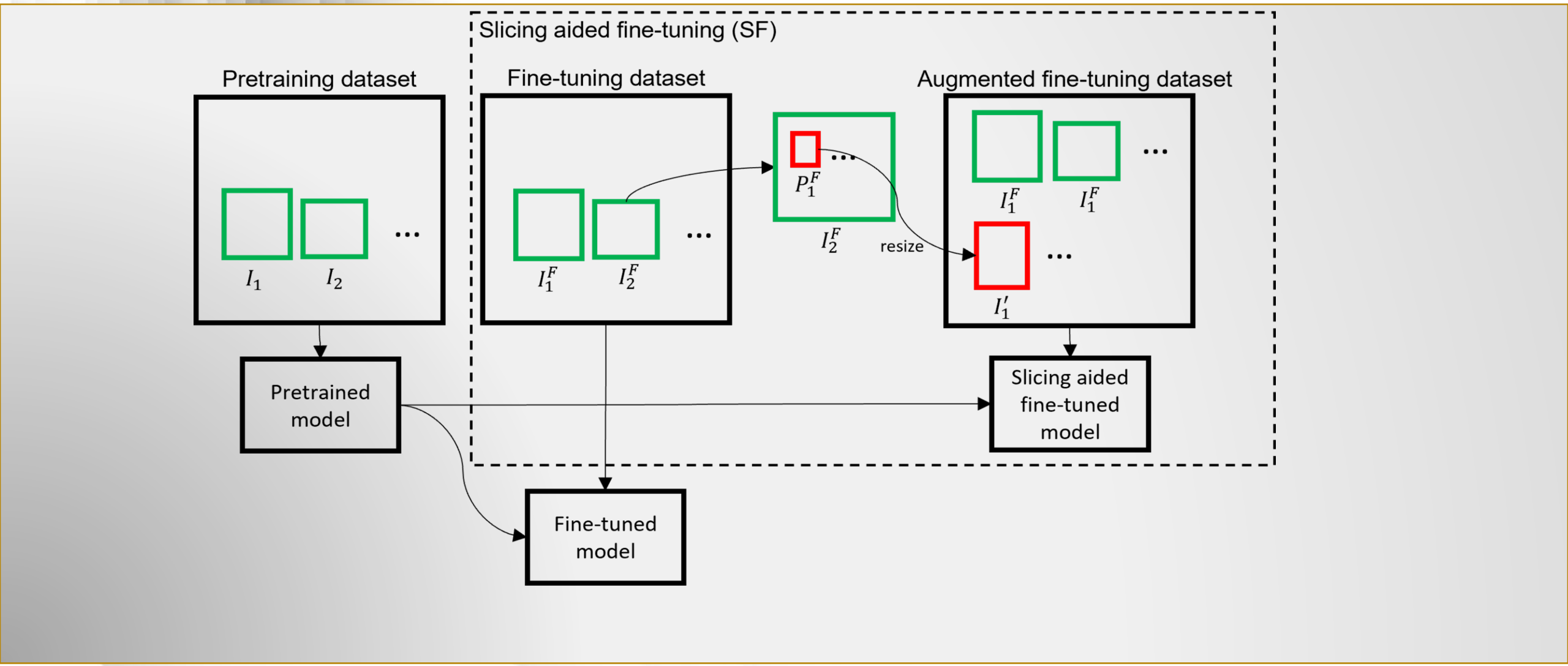
- Visualize predicted bounding boxes and masks over the original image:

```
result.export_visuals(export_dir="demo_data/")

Image("demo_data/prediction_visual.png")
```

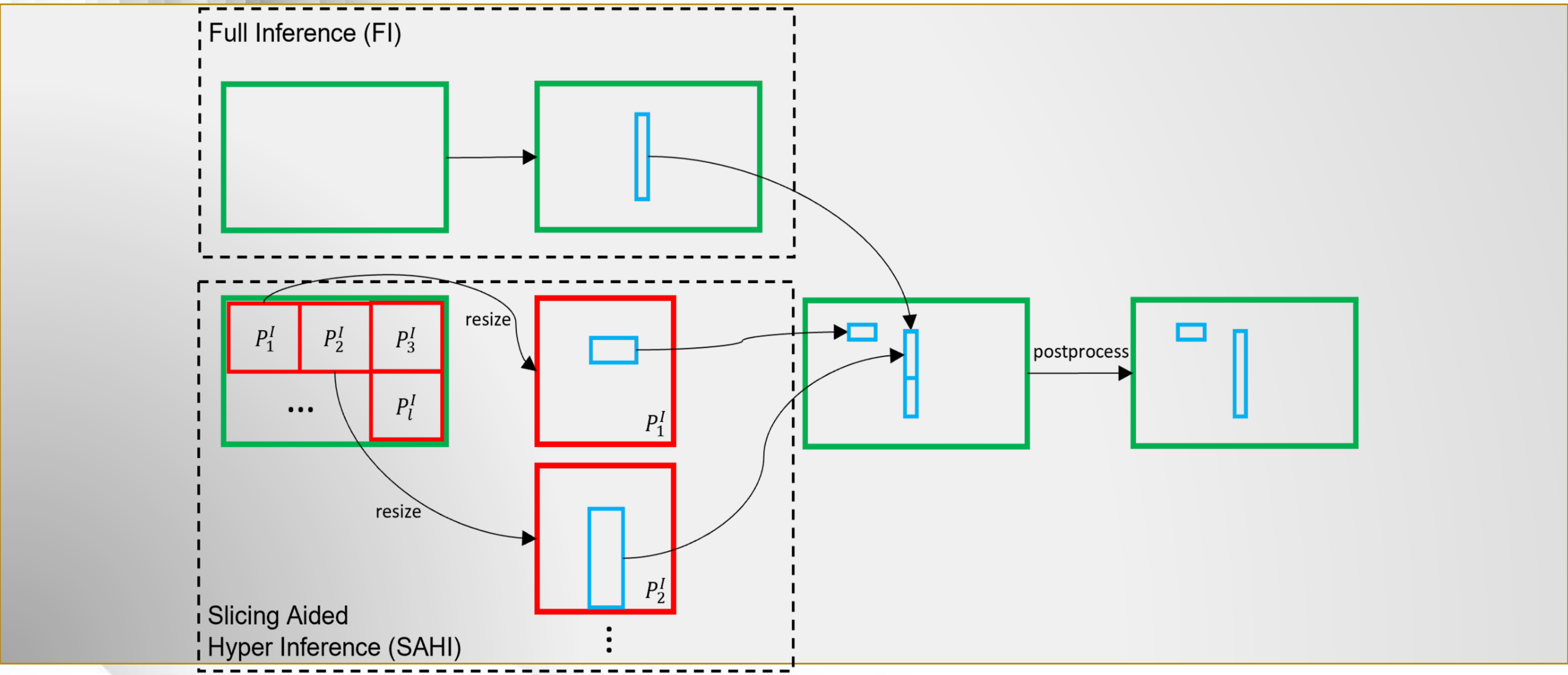


Slicing Aided Fine-tuning



F.C. Akyon, S.O. Altinuc, A. Temizel, “Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection”, ICIP, Oct. 2022.

Slicing Aided Hyper Inference (SAHI)



F.C. Akyon, S.O. Altinuc, A. Temizel, “Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection”, ICIP, Oct. 2022.

ResNet strikes back: An improved training procedure in timm

- ResNets remain the gold-standard architecture in numerous scientific publications.
- Yet there has been significant progress on best practices for training neural networks since the inception of the ResNet in 2015.
- Novel optimization methods and data augmentation have increased the effectiveness of the training recipes.
- The performance of the vanilla ResNet-50 when trained with a procedure that integrates such advances is evaluated.
- Training settings and pre-trained models are shared in the timm open-source library.

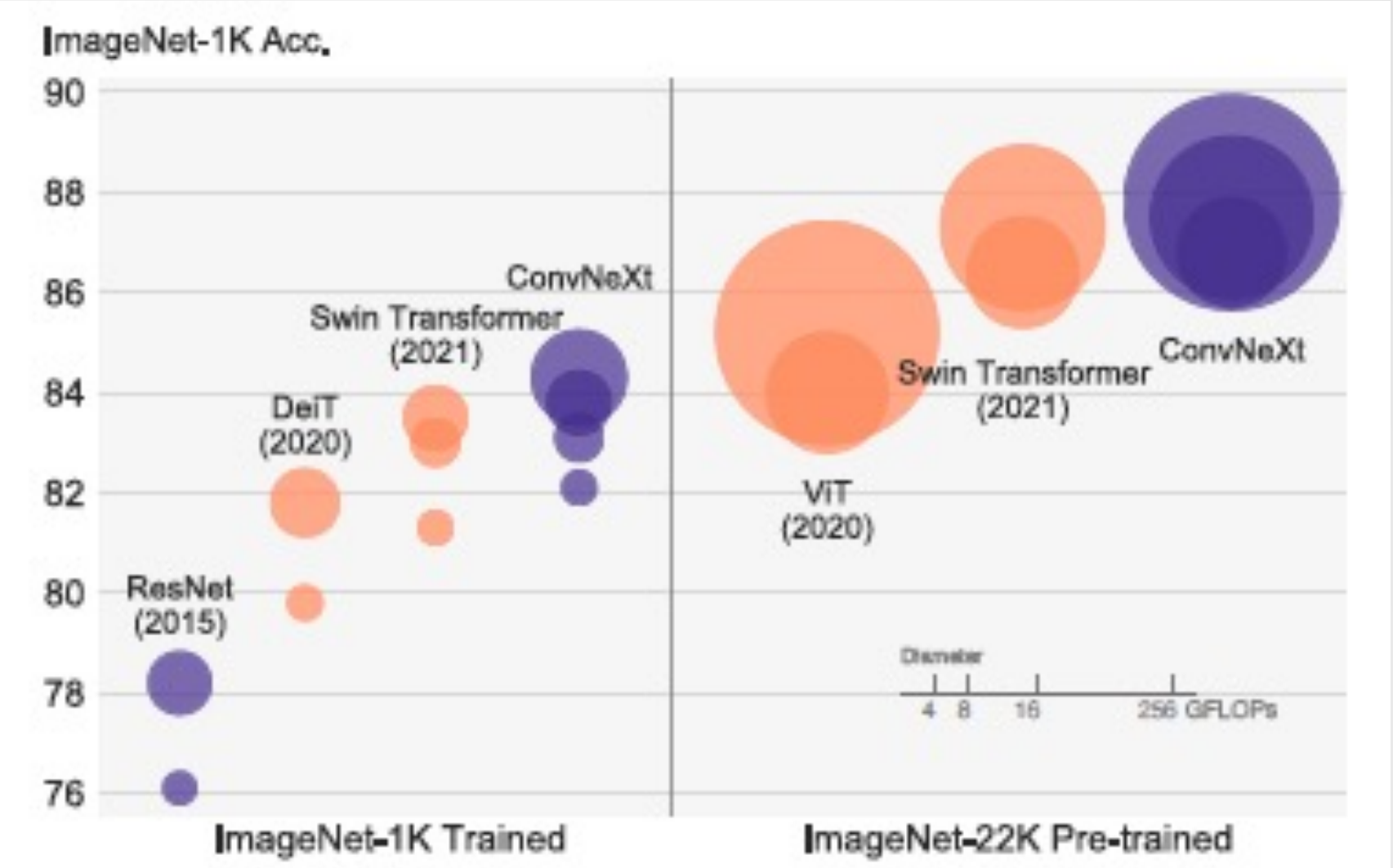
Wightman, Ross et al., 2021, "Resnet strikes back: An improved training procedure in timm." *arXiv preprint arXiv:2110.00476*.
https://github.com/rwightman/pytorch-image-models

ResNet strikes back: An improved training procedure in timm

Table 2: Ingredients and hyper-parameters used for ResNet-50 training in different papers. We compare existing training procedures with ours.

	Previous approaches					Ours		
Procedure →	ResNet	PyTorch	FixRes	DeiT	FAMS (×4)	A1	A2	A3
Reference	[13]	[1]	[48]	[45]	[10]			
Train Res	224	224	224	224	224	224	224	160
Test Res	224	224	224	224	224	224	224	224
Epochs	90	90	120	300	400	600	300	100
# of forward pass	450k	450k	300k	375k	500k	375k	188k	63k
Batch size	256	256	512	1024	1024	2048	2048	2048
Optimizer	SGD-M	SGD-M	SGD-M	AdamW	SGD-M	LAMB	LAMB	LAMB
LR	0.1	0.1	0.2	1×10^{-3}	2.0	5×10^{-3}	5×10^{-3}	8×10^{-3}
LR decay	step	step	step	cosine	step	cosine	cosine	cosine
decay rate	0.1	0.1	0.1	–	$0.02^{e/400}$	-	–	–
decay epochs	30	30	30	–	1	-	–	–
Weight decay	10^{-4}	10^{-4}	10^{-4}	0.05	10^{-4}	0.01	0.02	0.02
Warmup epochs	✗	✗	✗	5	5	5	5	5
Label smoothing ε	✗	✗	✗	0.1	0.1	0.1	✗	✗
Dropout	✗	✗	✗	✗	✗	✗	✗	✗
Stoch. Depth	✗	✗	✗	0.1	✗	0.05	0.05	✗
Repeated Aug	✗	✗	✓	✓	✗	✓	✓	✗
Gradient Clip.	✗	✗	✗	✗	✗	✗	✗	✗
H. flip	✓	✓	✓	✓	✓	✓	✓	✓
RRC	✗	✓	✓	✓	✓	✓	✓	✓
Rand Augment	✗	✗	✗	9/0.5	✗	7/0.5	7/0.5	6/0.5
Auto Augment	✗	✗	✗	✗	✓	✗	✗	✗
Mixup alpha	✗	✗	✗	0.8	0.2	0.2	0.1	0.1
Cutmix alpha	✗	✗	✗	1.0	✗	1.0	1.0	1.0
Erasing prob.	✗	✗	✗	0.25	✗	✗	✗	✗
ColorJitter	✗	✓	✓	✗	✗	✗	✗	✗
PCA lighting	✓	✗	✗	✗	✗	✗	✗	✗
SWA	✗	✗	✗	✗	✓	✗	✗	✗
EMA	✗	✗	✗	✗	✗	✗	✗	✗
Test crop ratio	0.875	0.875	0.875	0.875	0.875	0.95	0.95	0.95
CE loss	✓	✓	✓	✓	✓	✗	✗	✗
BCE loss	✗	✗	✗	✗	✗	✓	✓	✓
Mixed precision	✗	✗	✗	✓	✓	✓	✓	✓
Top-1 acc.	75.3%	76.1%	77.0%	78.4%	79.5%	80.4%	79.8%	78.1%

Vision Transformers or CNNs?



Liu, Zhuang, et al. "A ConvNet for the 2020s." *arXiv preprint arXiv:2201.03545* (2022).

Vision Transformers and CNNs

- ViT performance depends highly on dataset-specific hyperparameters, and network depth. CNNs are much easier to optimize.
- A variation on a pure transformer is to marry a transformer to a CNN stem/front end. A typical ViT stem uses a 16x16 convolution with a 16 stride. By contrast a 3x3 convolution with stride 2, increases stability and also improves accuracy*.
- The CNN translates from the basic pixel level to a feature map. A tokenizer translates the feature map into a series of tokens that are then fed into the transformer, which applies the attention mechanism to produce a series of output tokens.
- Finally, a projector reconnects the output tokens to the feature map. The latter allows the analysis to exploit potentially significant pixel-level details. This drastically reduces the number of tokens that need to be analyzed, reducing costs accordingly.

Xiao, Tete et al., 2021, "Early Convolutions Help Transformers See Better". arXiv:2106.14881