ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 18: Vision Transformers

03/24/2025

https://deeprob.org/w25/

ROBOTICS

Today

- Feedback and Recap (5min)
- Final Project Reminder (5min)
- Vision Transformer (40min)
  - Layer Norm
  - ViT
- DETR (20min)
- Summary and Takeaways (5min)

P4 due March 30, 2025

Final Project

https://deeprob.org/w25/projects/finalproject/

April 1st, 5-min poster “lightning talk” @ CSRB 2246 (classroom)
April 22nd, final project showcase @FRB atrium
April 28th, final project (report, code, video/website) DUE

More Details on “Lightning Talk”

- **April 1, 2025** (Tuesday discussion session)
  - starting **3:30pm** EST, CSRB 2246
- Each group gets ~5min
- All group members should speak
- Poster/Slides (either are fine). Content include:
  - Topic/Title
  - Motivation/Background
  - Some Lit Review (“Gaps”)
  - Aim to reproduce/run 1-2 baseline method by then and include some baseline results
  - MUST-DO Goals/Stretch Goals: Describe what you propose to do
  - Group member names/roles/sub-tasks
  - Timeline/Milestone
  - Material (if any); Dataset/Compute resources etc.

~1min
~3min
~1min

- Images/Diagrams Highly Encouraged!
- Practice!

More Details on “Lightning Talk”

- Some References
  - Look up “3 Minute Thesis” (3MT) winner talks
  - UM Research Poster template/resources
    - https://guides.lib.umich.edu/poster
    - https://branding.med.umich.edu/design-resources/diy-templates/research-posters
    - https://lsa.umich.edu/urop/symposium/spring-symposium/fall-winter-students/poster-resources.html
    - https://branding.med.umich.edu/design-resources/diy-templates/research-posters/posters-templates

Also, Robotics seminars

- **Post all your materials to “Final_Project_LightningTalk” folder on Google Drive** by April 1
  - Note: If using external video/images, make sure it has access

5% grade:

- Content/Organization + poster/slides quality
- Clarity
- Time management, flow of presentation
- Enthusiasm/Audience awareness and connection/ communication

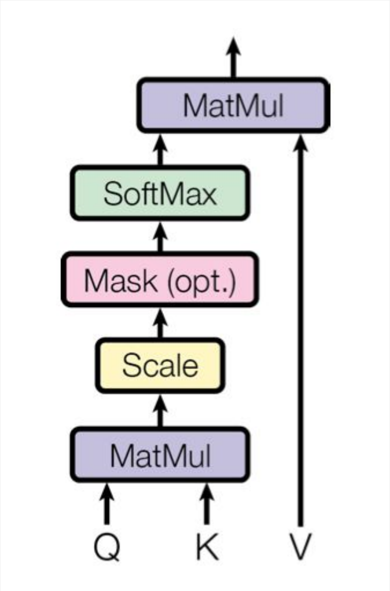
Rubrics example

- https://www.readwritethink.org/sites/default/files/30700_rubric.pdf
- https://open.umich.edu/sites/default/files/downloads/instructmethodshpe-resources-oral_presentation_evaluation_rubric_reformatted-wlicenseimage.pdf

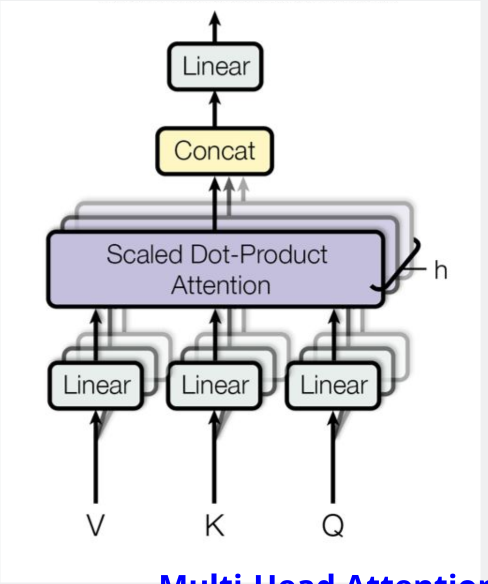
Vision Transformer

Recap: Transformers

Scaled Dot-Product Attention

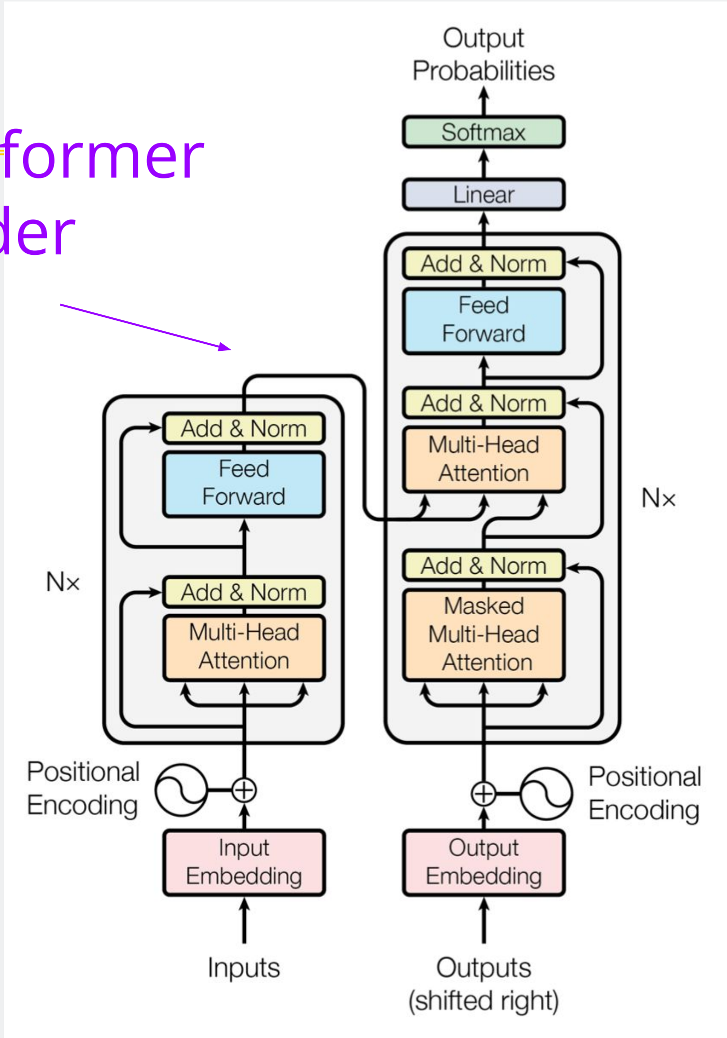


$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \in \mathbb{R}^{P \times d_v}$$



Multi-Head Attention

Transformer Encoder



Comparing RNNs to Transformer

RNNs

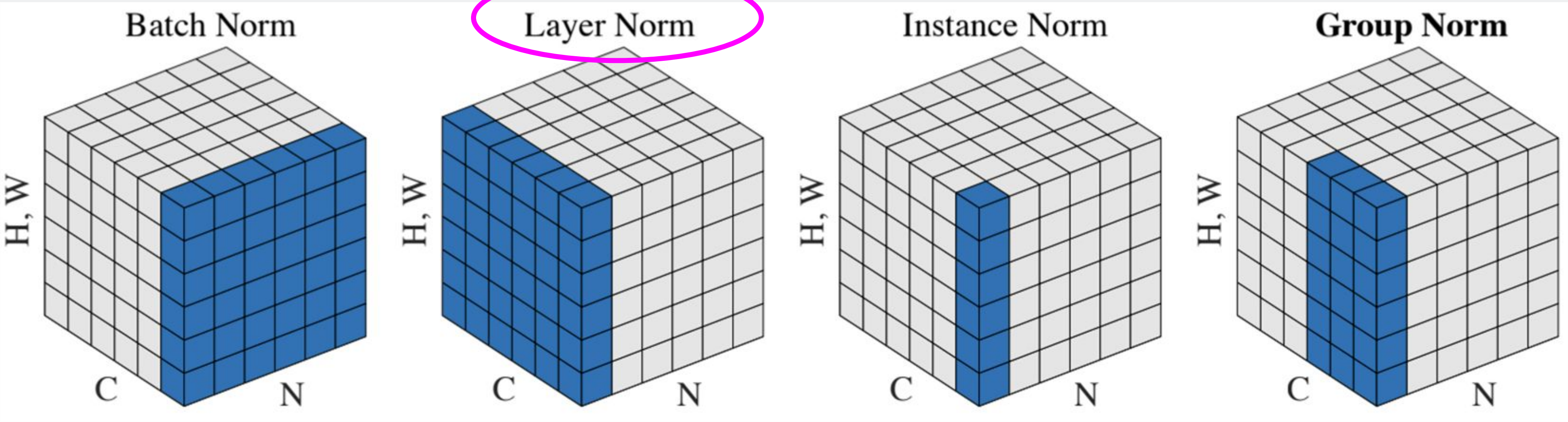
(+) LSTMs work reasonably well for long sequences.
(-) Expects an ordered sequences of inputs
(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

Transformer:

(+) Good at **long sequences**. Each attention calculation looks at all inputs.
(+) Can operate over unordered sets or **ordered** sequences with positional encodings.
(+) **Parallel** computation: All alignment and attention scores for all inputs can be done in parallel.
(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head.
(but GPUs/TPUs are getting bigger and better)

Layer Normalization (LayerNorm)

Recall: From Lecture 7



https://arxiv.org/abs/1607.06450

(Also check the supplementary materials)

Layer Normalization (LayerNorm)

Why?

"One of the challenges of deep learning is that the gradients with respect to the weights in one layer are <u>highly dependent</u> on the outputs of the neurons in the previous layer especially if these outputs change in a highly correlated way."

"covariate shift"

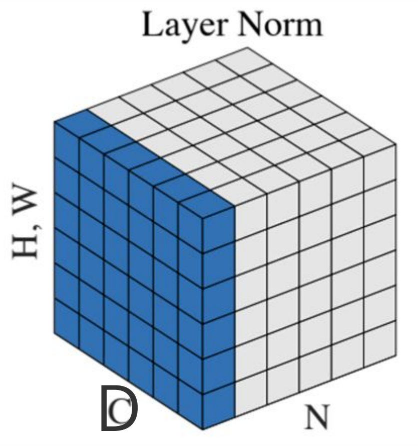
BatchNorm: over the whole data distribution

- Can be impractical
- Hard to apply to recurrent NNs

LayerNorm: Fixing the mean and variance within each layer

https://arxiv.org/abs/1607.06450

Layer Normalization (LayerNorm)



Forward:

Input: `x`: shape (*, D)

1. Compute mean and variance over the features (D) for each data point (N)
2. Normalizing the input data of shape (*, D)
3. Scale and shift using gamma and beta

   `out = gamma * div + beta`
4. Store cache needed for backward pass

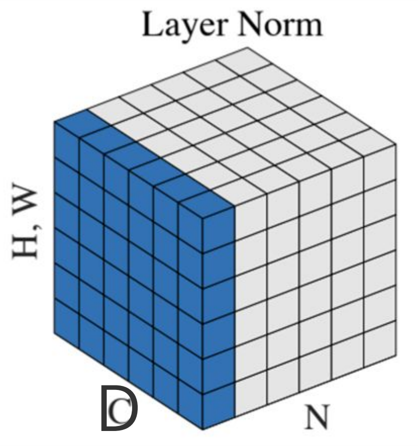
$$div = \frac{x - sampleMean}{\sqrt{sampleVar + eps}}$$

https://arxiv.org/abs/1607.06450

ROBOTICS

Layer Normalization (LayerNorm)

$$div = \frac{x - sampleMean}{\sqrt{sampleVar + eps}}$$



Backward:

Input: `dout`: Upstream derivatives, of shape (*, D)

```
out = gamma * div + beta
```

1. Compute gradients w.r.t. gamma and beta
2. Compute gradients w.r.t. input x
   a. ddiv = dout * gamma
   b. dx

```
dgamma, dbeta
```

```
return dx, dgamma, dbeta
```

https://robotchinwag.com/posts/layer-normalization-deriving-the-gradient-for-the-backward-pass/

Can we apply Transformers to Images?

Yes!

Idea: Treat the image as a set of patches of pixels

Vision Transformers

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], Lucas Beyer[*], Alexander Kolesnikov[*], Dirk Weissenborn[*], Xiaohua Zhai[*], Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

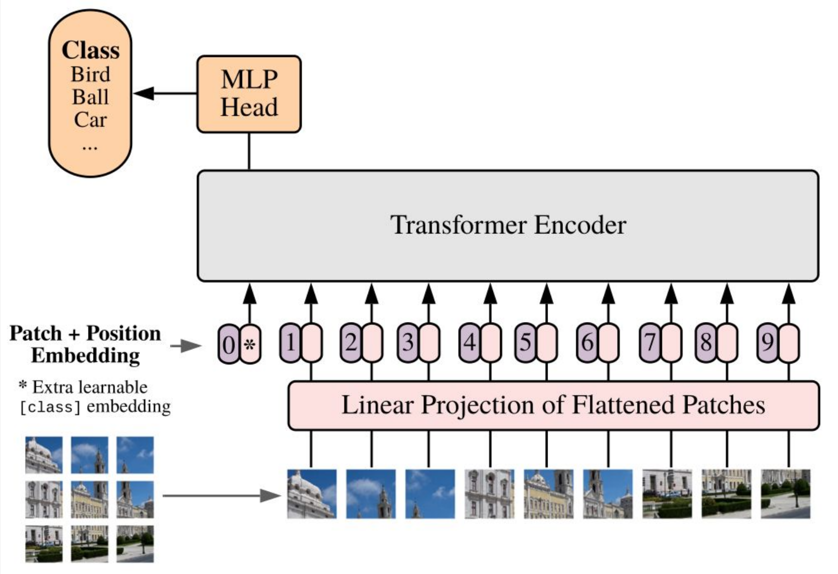
{adosovitskiy, neilhoulsby}@google.com

Dosovitskiy et al. ICLR'21

https://arxiv.org/pdf/2010.11929

Vision Transformers

- Convert image into 16x16 patches
    - E.g. (1, 240, 240, 3) -> (1, 15x15, 16x16x3)
- Apply shared linear projection to each patch
    - E.g. (1, 15x15, 16x16x3) -> (1, 15x15, 64)
- Concatenate learnable class token for classifier output
    - E.g. (1, 1+15x15, 64)
- Add position embedding to each patch
    - E.g. (1, 1+15x15, 64) + (1, 1+15x15, 64)



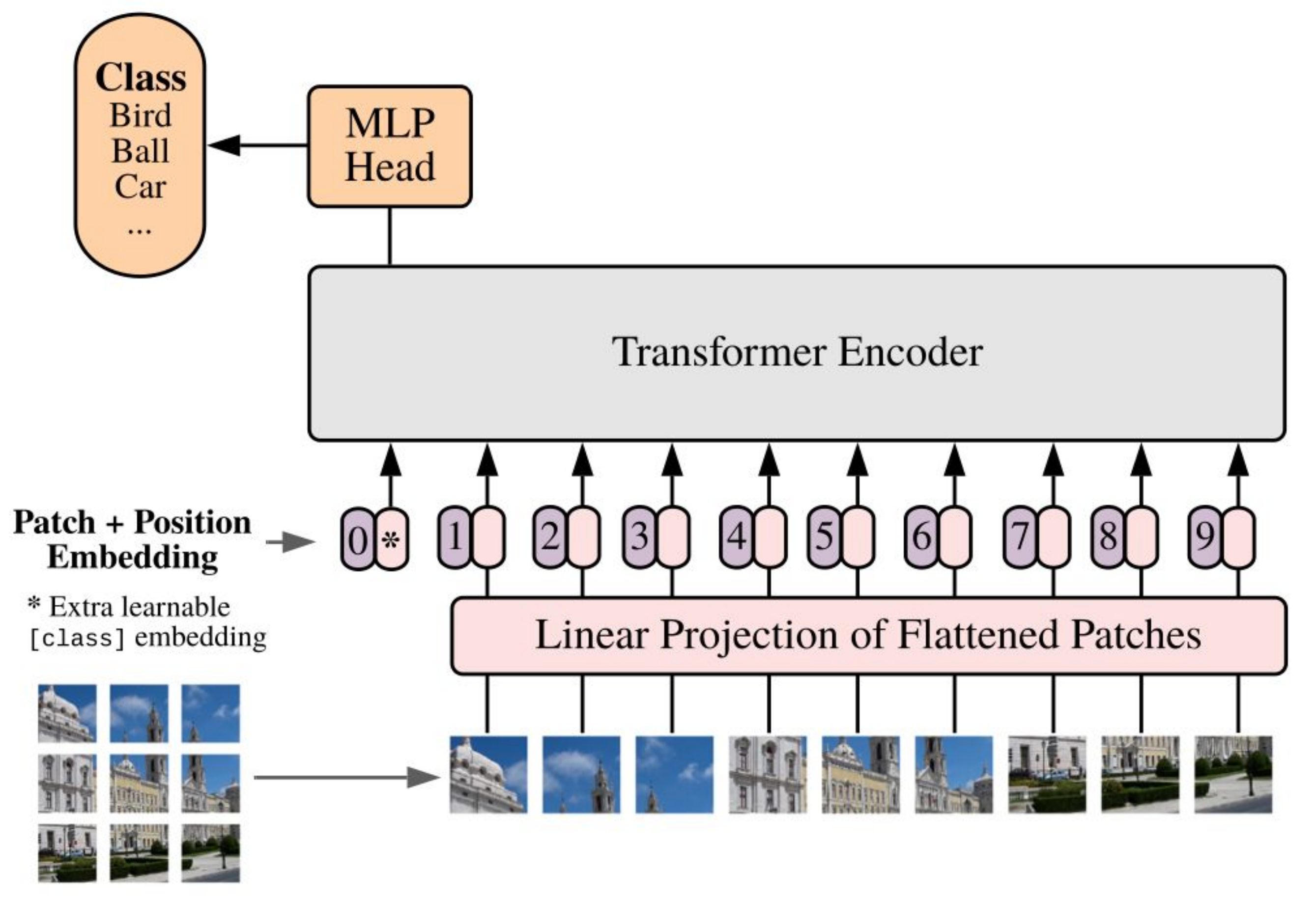
(P4) patchify

Vision Transformers - cls token

(also appears in **BERT**) https://arxiv.org/pdf/1810.04805

[cls] token

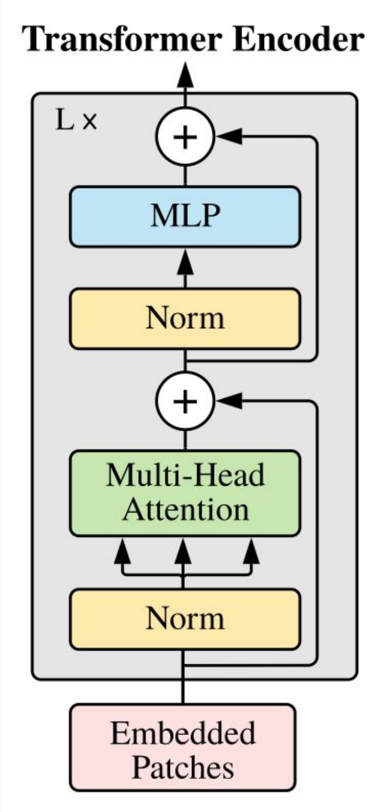
Added in front of every input example



Vision Transformer Encoder

`Class TransformerEncoder (P4)`

- Based on the Transformer encoder
- Sequence of **LNorm->MHSA->LNorm->MLP** with residual skip connections
- For input embedded patches: (1, 1+15x15, D_in)
    - Output: (1, 1+15x15, D_out)
- For final classification decision:
    - Apply MLP and softmax to the class token
    - (1, 1, D_out) -> (1, 1, N_classes)

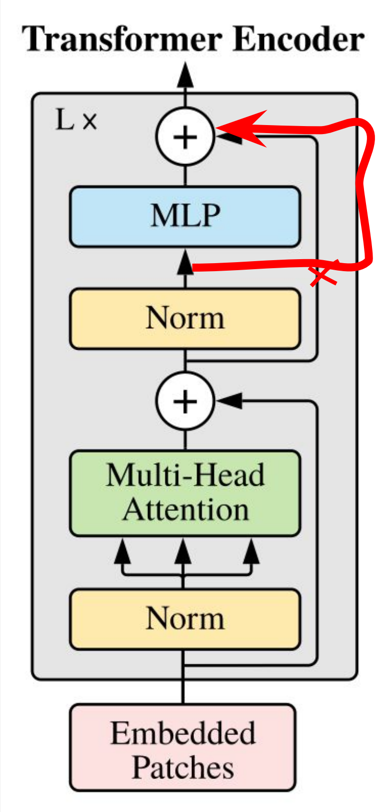


MLP: "two layers with a GELU non-linearity."
Linear -> GELU -> Linear

Vision Transformer Encoder

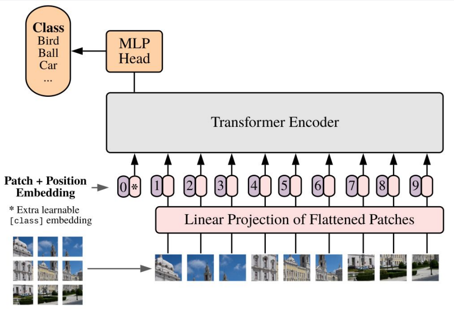
- Based on the Transformer encoder
- Sequence of **LNorm->MHSA->LNorm->MLP** with residual skip connections
- *One note on current autograder:
  - `x_norm2 + mlp_out`

    (in TransformerEncoder class)



Vision Transformer

Based on Fig.1 of ViT paper



`Class VisionTransformer (P4)`

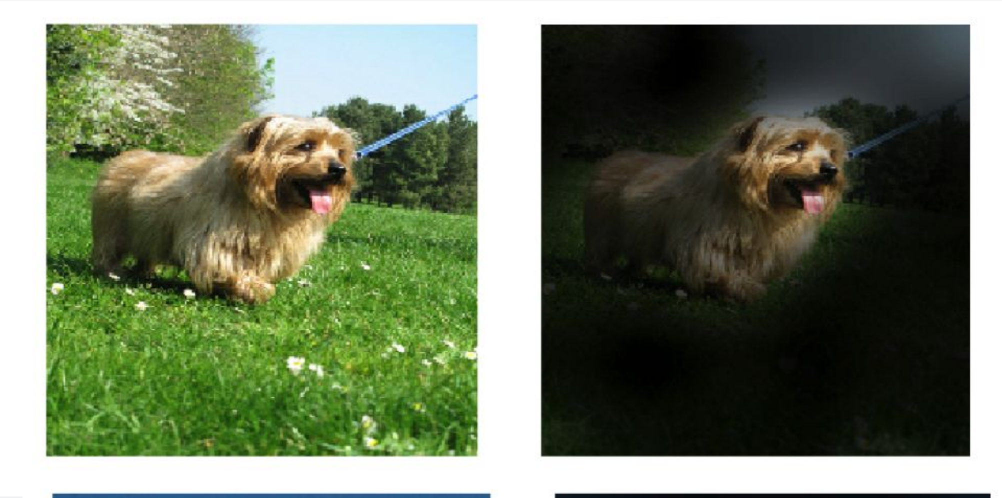
`__INIT__` part: Initialize

1. a projection (linear) layer, that takes in patches and output `embed_dim` (embeddings)
2. transformer layers, where you have `range(self.num_layers)` of `TransformerEncoder` layers.
3. the MLP head, where it is essentially a `LayerNorm` + a `Linear` layer.

`forward` part: Pass through input x through all these layers. So:

1. patchify
2. call the projection layer
3. get cls_tokens (repeat `self.cls_token` for N times)
4. concatenate outputs from 2 and 3
5. add position embedding to 4. This will be the input that goes in stacks of transformer layers
6. loop through all `self.transformer_layers`, this gives you the `out_tokens` and `attention maps`. (see `TransformerEncoder` class for more details).
7. pass the `out_tokens` from 6 through your `mlp_head` (that you initialized in step 3 in the __init__). This will be `out_cls`.

Vision Transformer - Attention Maps Visualization



https://arxiv.org/pdf/2010.11929

Figure 6: Representative examples of attention from the output token to the input space. See Appendix D.7 for details.

Vision Transformer Encoder

- Q: Why **skip connection**?

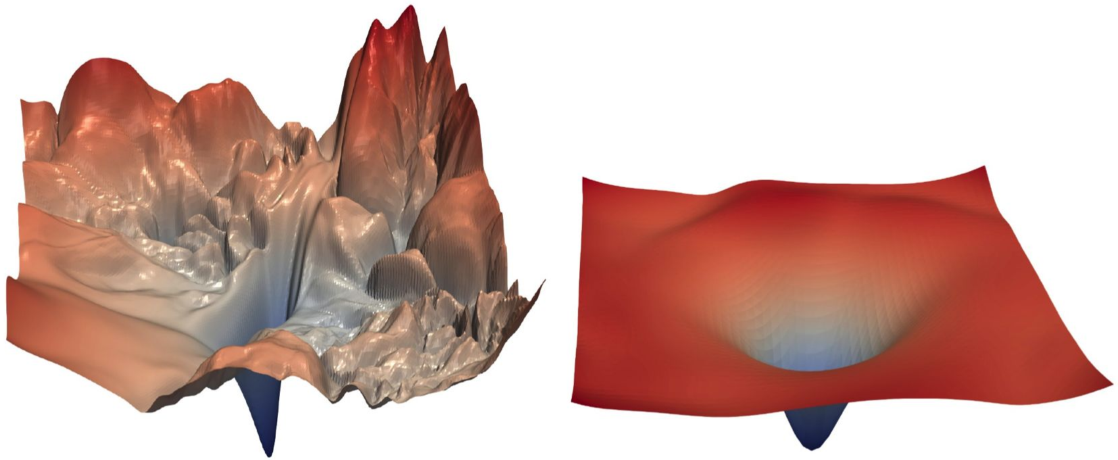
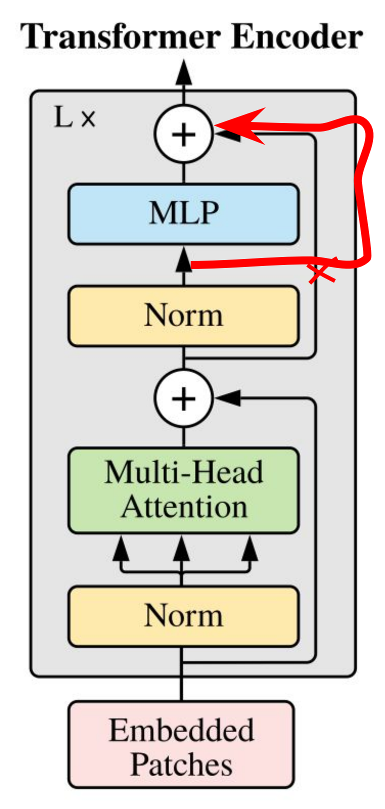


Figure 1: The loss surfaces of ResNet-56 with/without skip connections.



https://arxiv.org/pdf/1712.09913

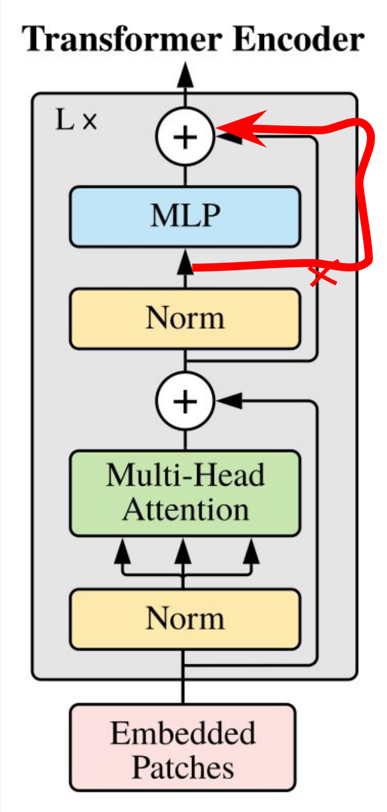
Dosovitskiy et al. ICLR'21

Vision Transformer Encoder

- Q: What is **inductive bias**?

The inductive bias (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered

- Pre-trained on larger scale (14M-300M images)
- (authors claim) Vision Transformer has much less image-specific inductive bias than CNNs



https://arxiv.org/pdf/2010.11929

Dosovitskiy et al. ICLR'21

Improving Vision Transformer

- **Regularization** for ViT models:
  - Weight Decay
  - Stochastic Depth
  - Dropout (in FFN layers of transformer)
- **Data Augmentation** for ViT models:
  - MixUp
  - RandAugment

https://arxiv.org/pdf/2106.10270

Large-scale pre-training

- **Distillation** for ViT models:
  - Teacher-Student model
  - (will discuss more on this)

https://arxiv.org/pdf/1503.02531

- Revisit **High-frequency Components**

https://arxiv.org/pdf/2204.00993

- ViT + **Evolutionary Algorithm**

https://link.springer.com/content/pdf/10.1007/s11263-024-02034-6.pdf

- **ConViT: Soft** convolutional inductive bias (gated positional self-attention)

https://proceedings.mlr.press/v139/d-ascoli21a/d-ascoli21a.pdf

M ROBOTICS

DETR

Cited by 16754

DETR: DEtection TRansformer

https://arxiv.org/pdf/2005.12872 (Facebook AI, 2020)

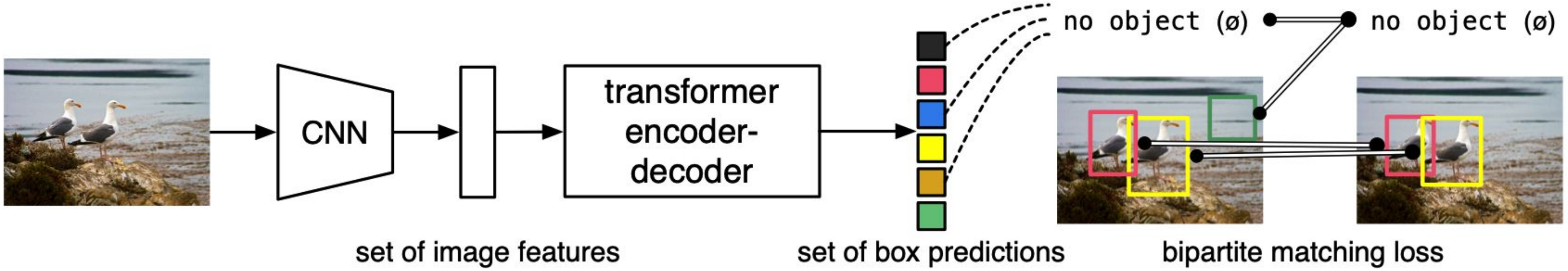


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a “**no object**” (∅) class prediction.

DETR: DEtection TRansformer

https://arxiv.org/pdf/2005.12872 (Facebook AI, 2020)

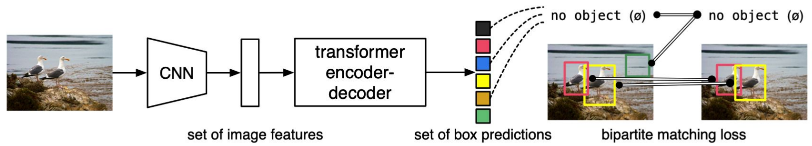


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "no object" (∅) class prediction.

Motivation:

- End-to-end
- No NMS (post-processing)
- Direct "set prediction" problem - removing duplicate

Key architecture components:

- Transformer Encoder-Decoder
- Bipartite loss - Hungarian Algorithm

*Fixed-size N (large) predictions

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

DETR: DEtection TRansformer

https://arxiv.org/pdf/2005.12872 (Facebook AI, 2020)

https://github.com/facebookresearch/detr
(Archived 2024)

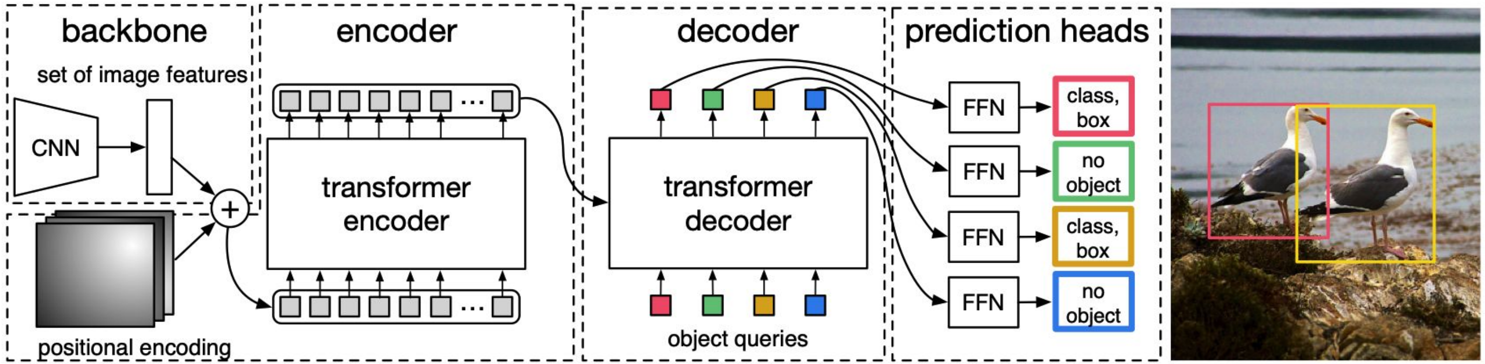
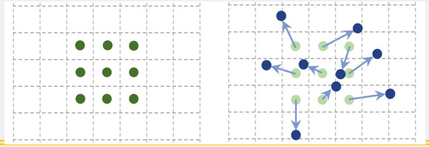


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

Deformable-DETR



https://openaccess.thecvf.com/content_ICCV_2017/papers/Dai_Deformable_Convolutional_Networks_ICCV_2017_paper.pdf

https://arxiv.org/pdf/2010.04159
(ICLR 2021)

Address two DETR issues:
(1) It requires much longer training epochs to converge than the existing object detectors

(2) DETR delivers relatively low performance at detecting small objects

➡ Deformable attention

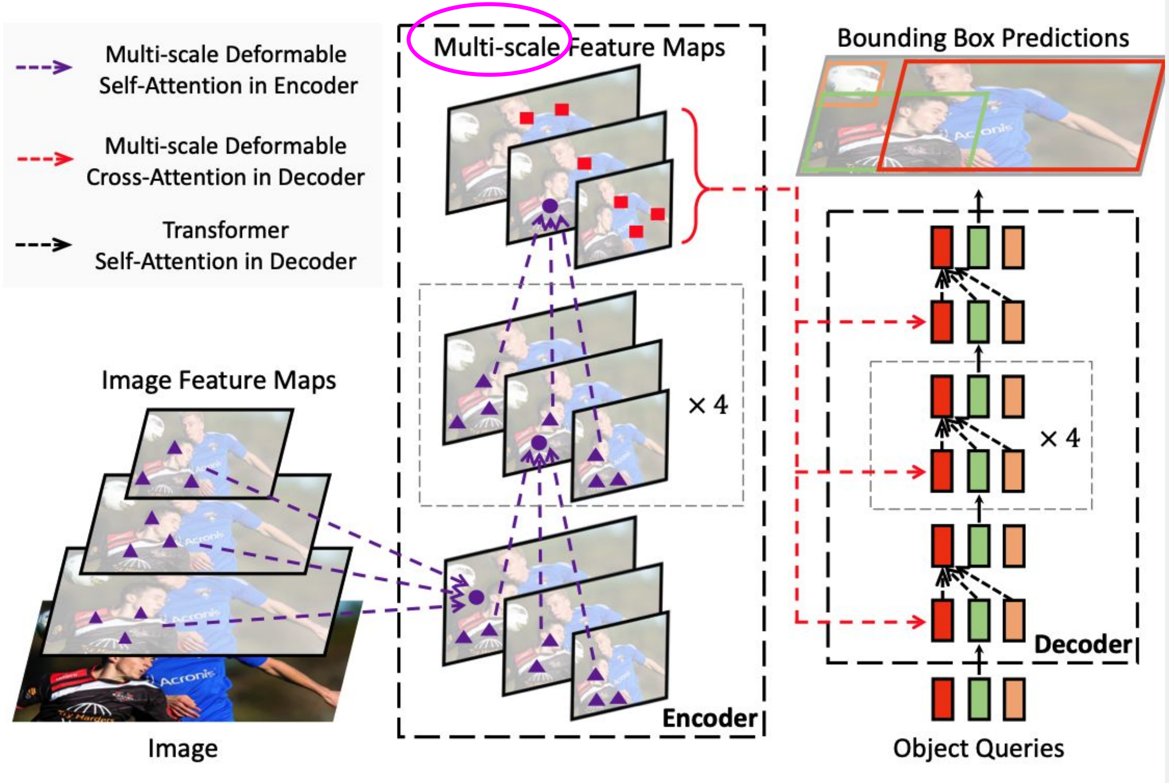
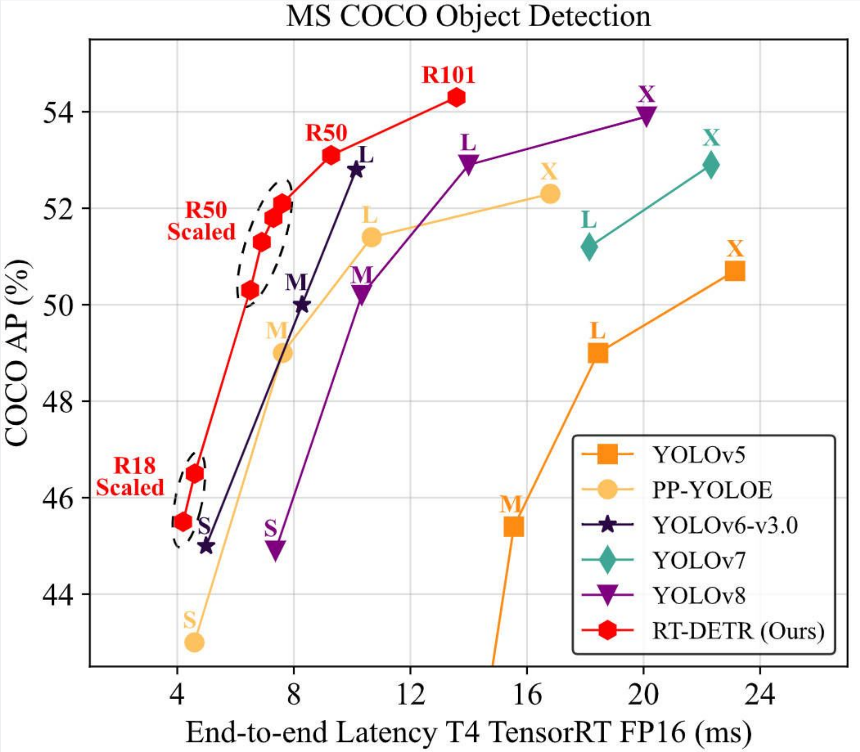
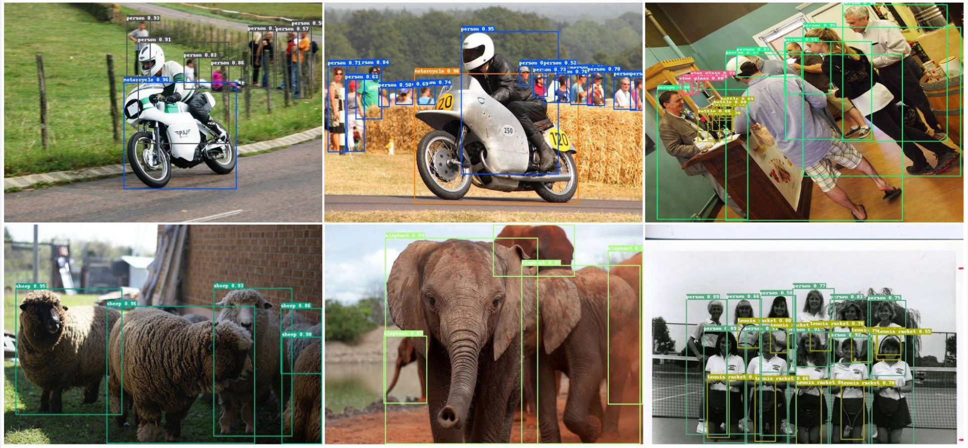


Figure 1: Illustration of the proposed Deformable DETR object detector.

RT-DETR (real-time)

https://zhao-yian.github.io/RTDETR/ (CVPR 2024)





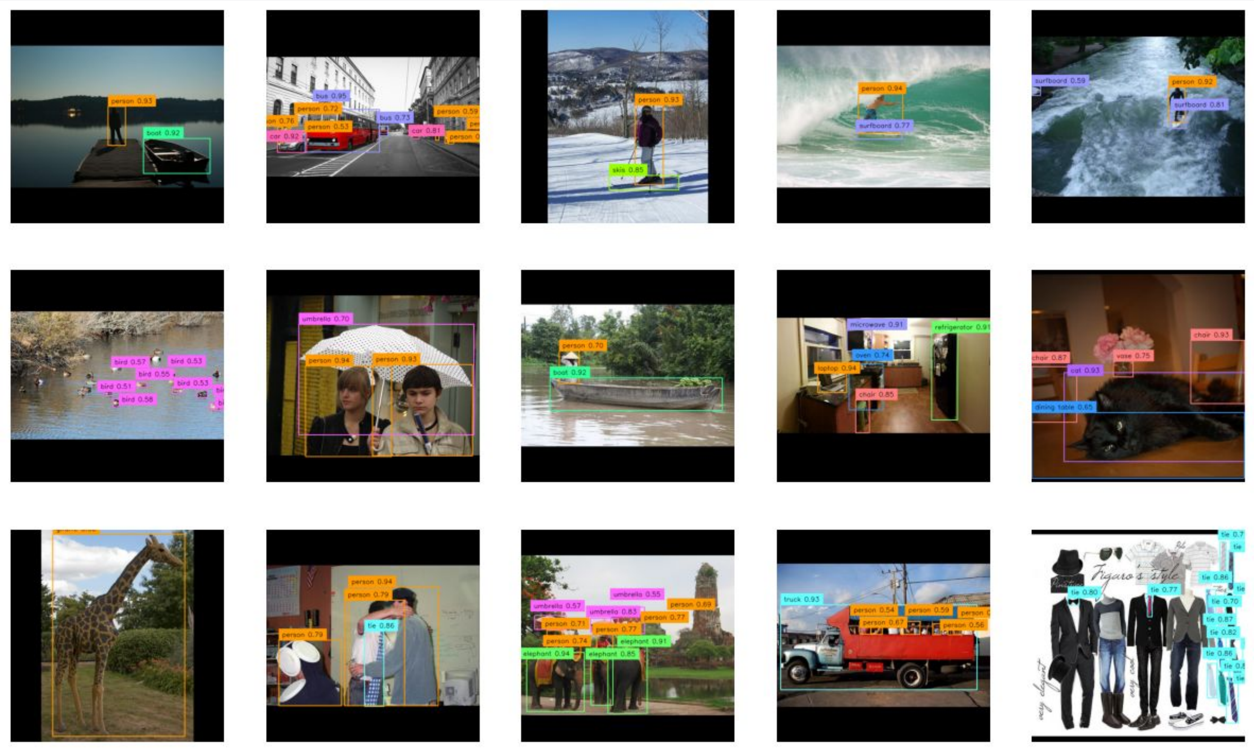
https://openaccess.thecvf.com/content/CVPR2024/papers/Zhao_DETRs_Beat_YOLOs_on_Real-time_Object_Detection_CVPR_2024_paper.pdf

ROBOTICS

RF-DETR (real-time)

https://github.com/roboflow/rf-detr (released March 20, 2025)

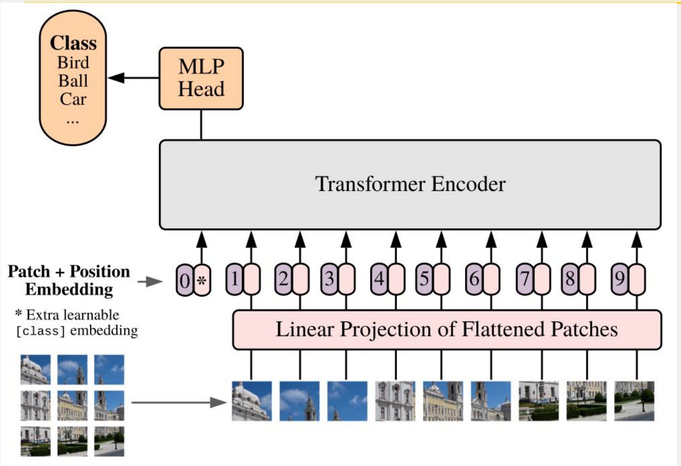
(claims >60AP)



Detectron2

https://github.com/facebookresearch/detectron2

Summary



- ViT
- DETR
- Swin Transformer
- More...

Reminder:
- **P4** Due March 30, 2025

(Lecture 13 on poseCNN,
Lecture 17, 18 on attention and ViT)
- **Final Project** "lightning talk" April 1, 2025
- **Canvas Quiz** will be released

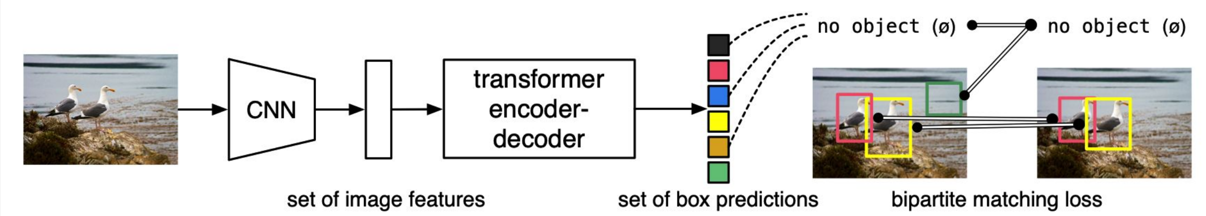


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "no object" (∅) class prediction.