Self-Supervised Training

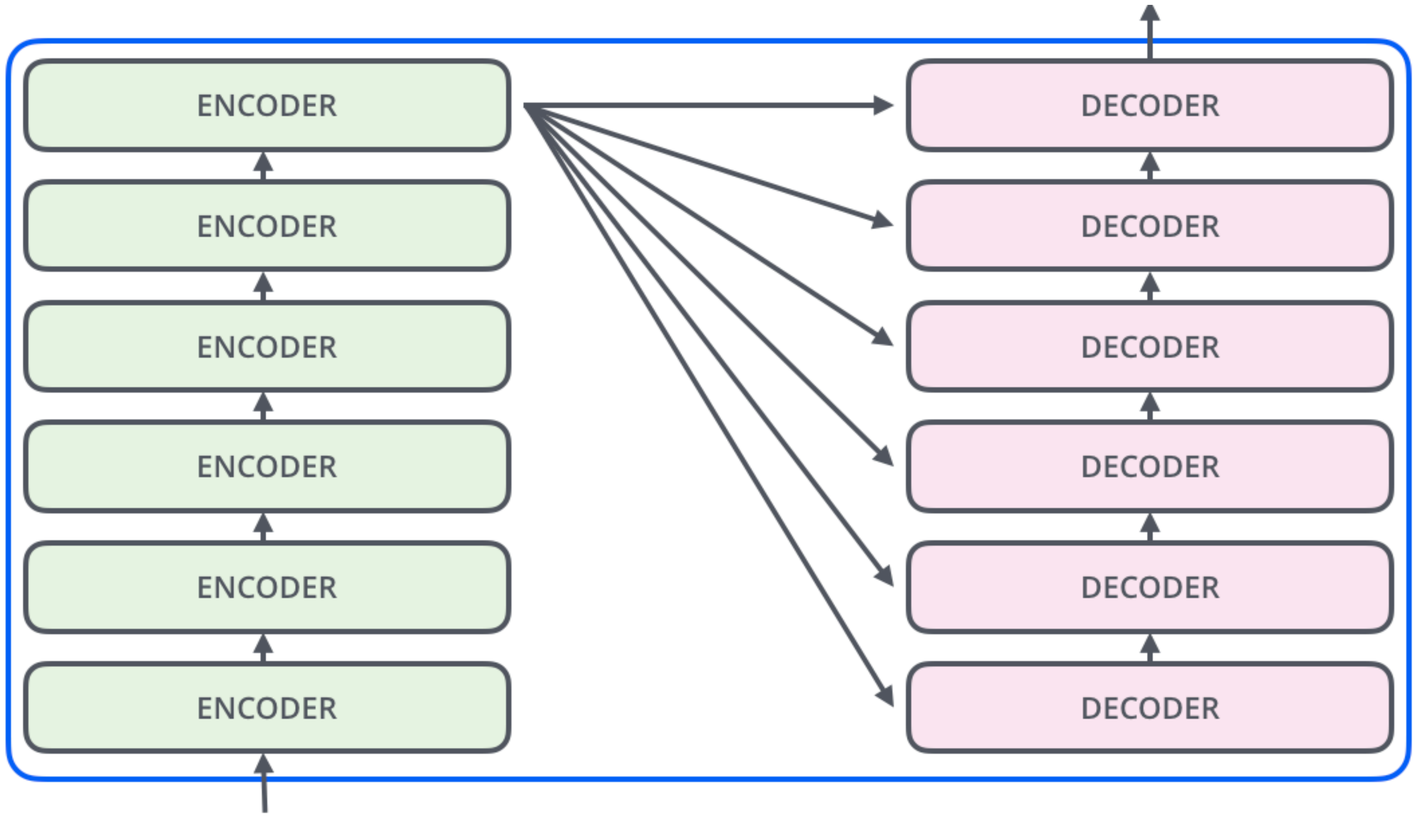
Principles of Deep Learning (AI502/IE408/IE511)

Sungbin Lim (UNIST AIGS & IE)

Contact: ai502deeplearning@gmail.com

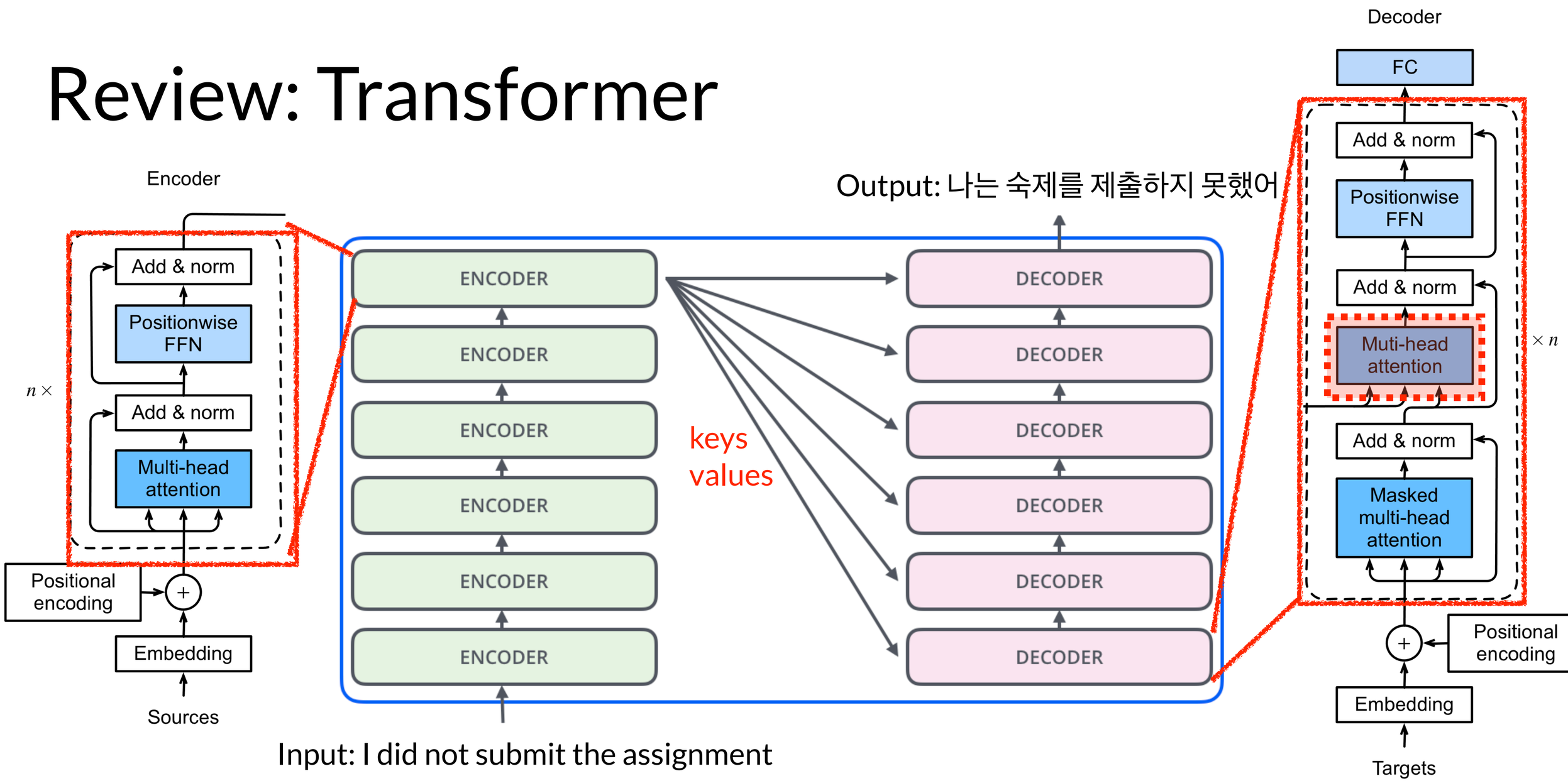
Review: Transformer

Output: 나는 숙제를 제출하지 못했어

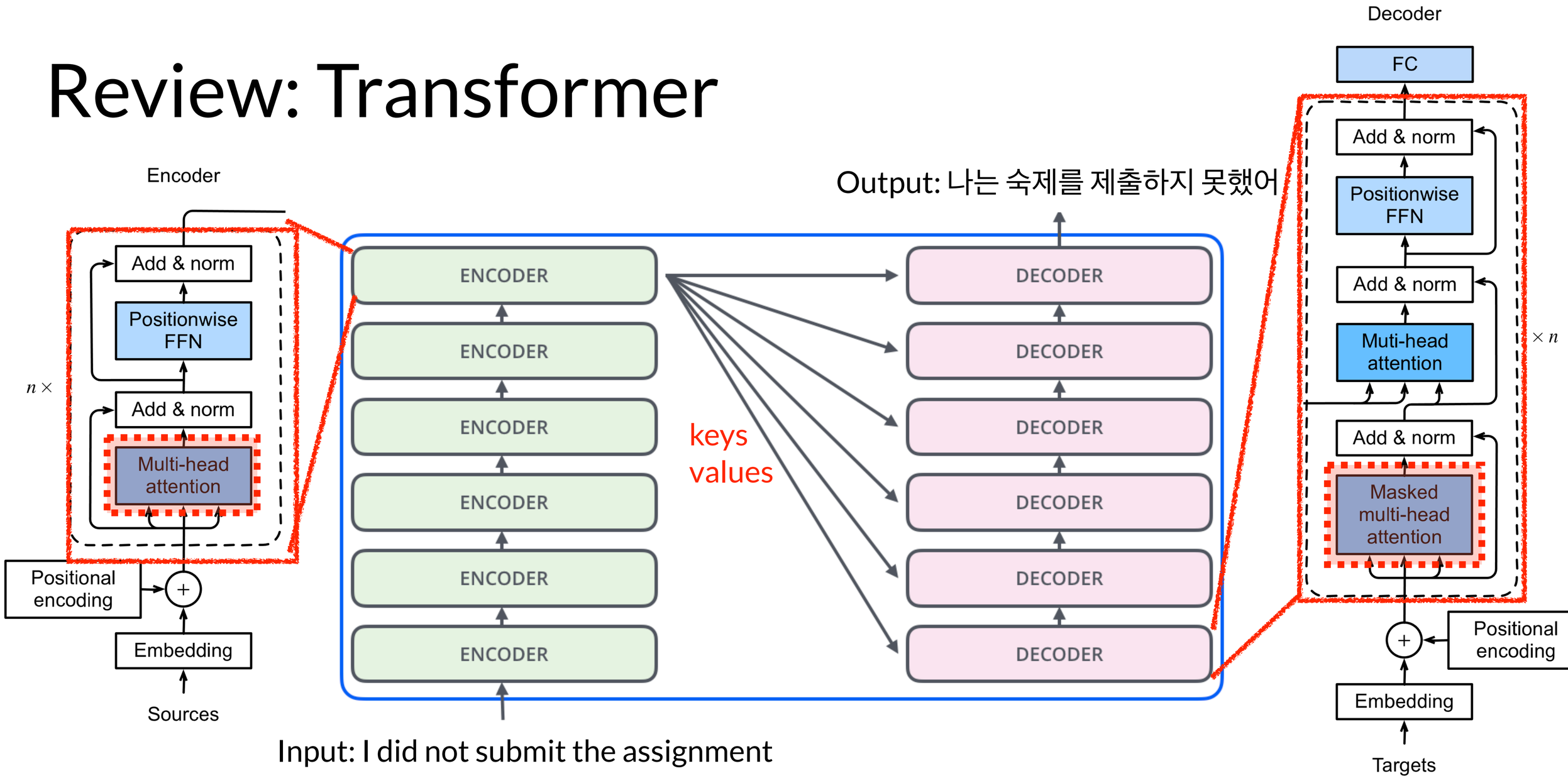


Input: I did not submit the assignment

Review: Transformer

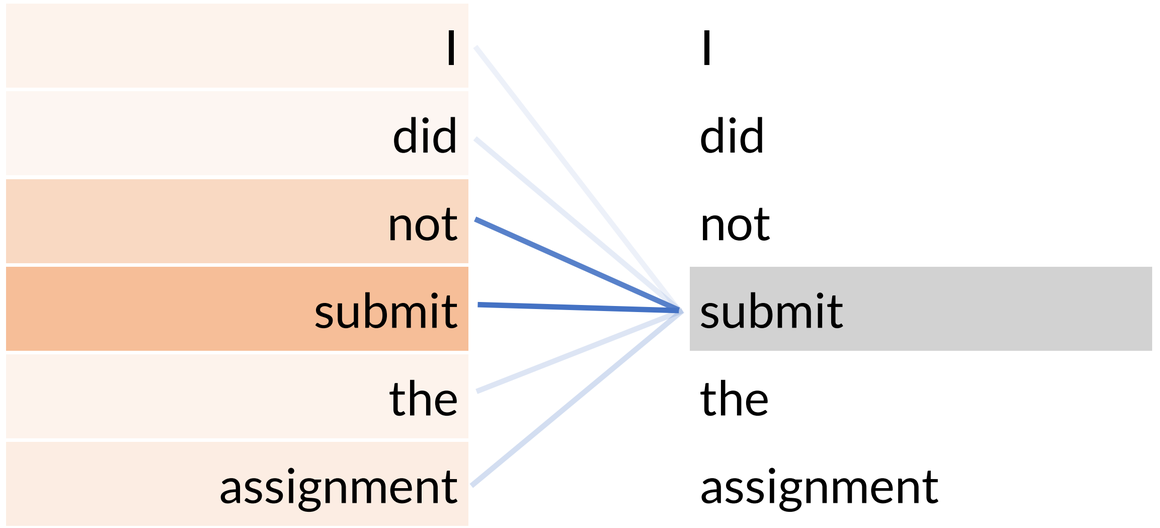


Review: Transformer



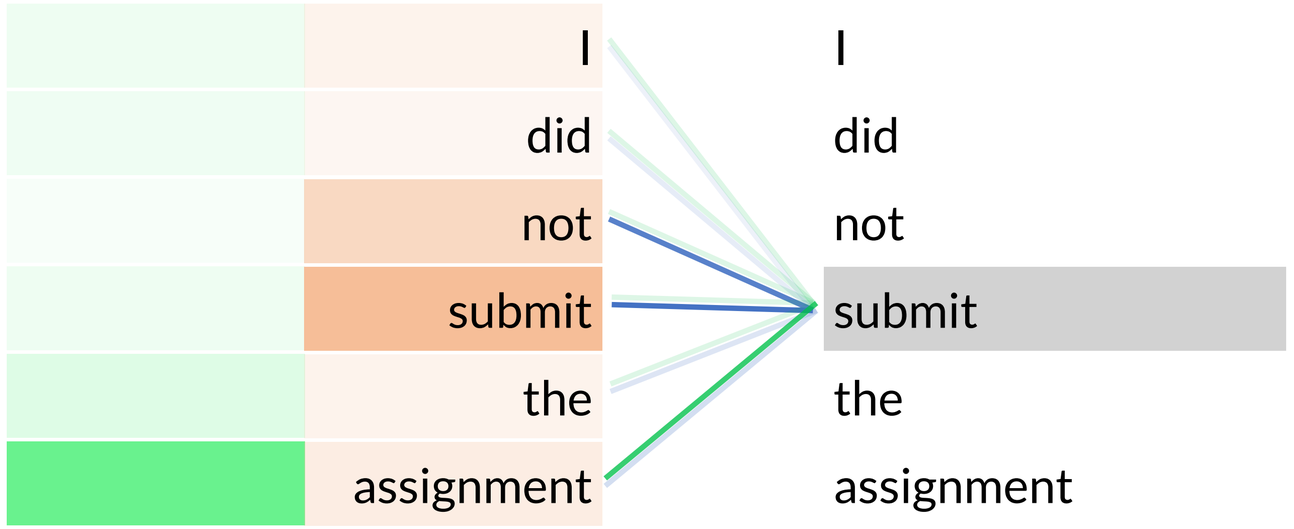
Self-Attention in Transformer

$$f(\mathbf{x}, \{(\mathbf{x}_i, \mathbf{x}_i)\}_{i=1}^{n}) = \sum_{i=1}^{n} \alpha(\mathbf{x}, \mathbf{x}_i)\mathbf{x}_i$$

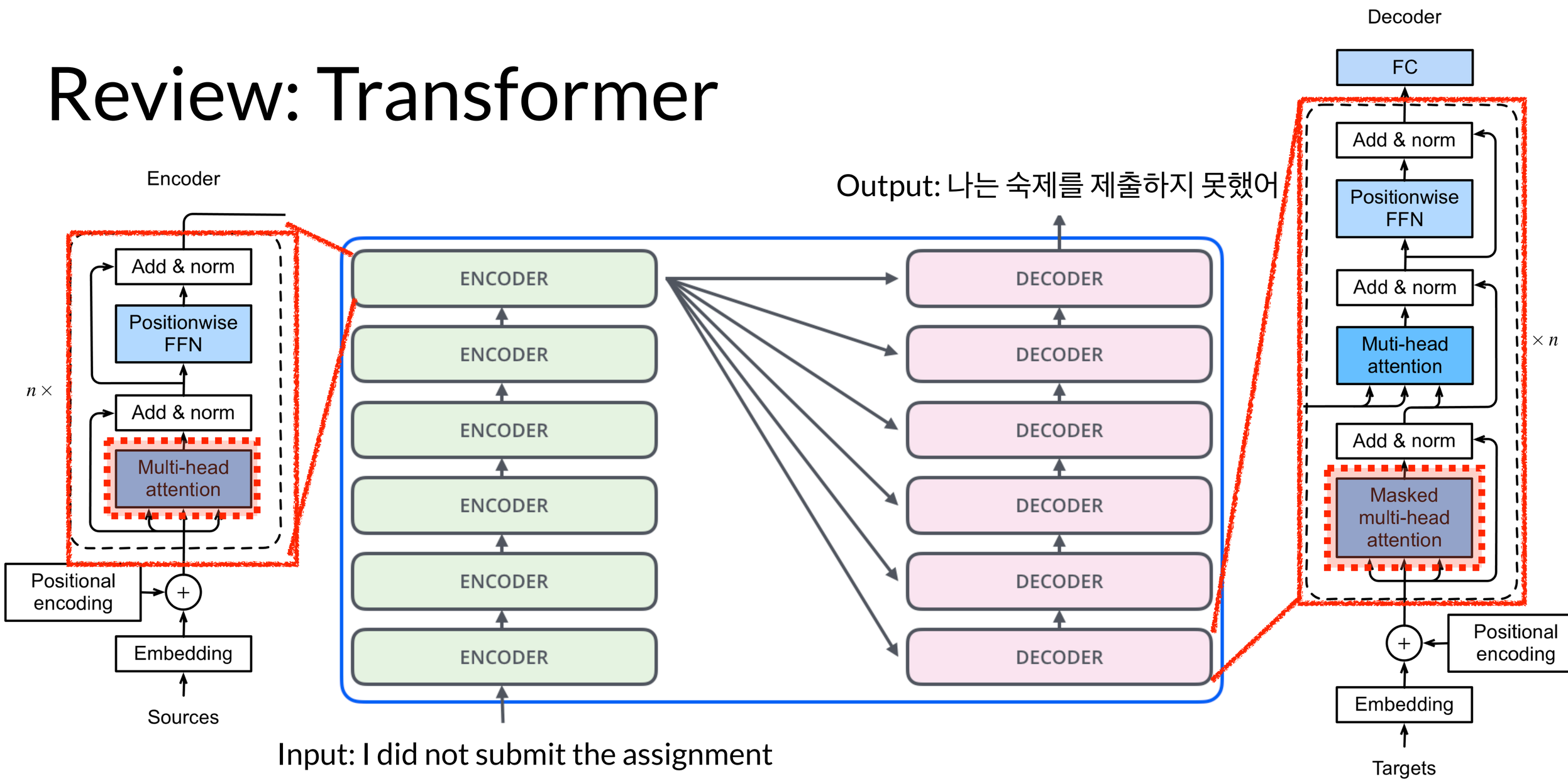


Multi-Head Self-Attentions in Transformer

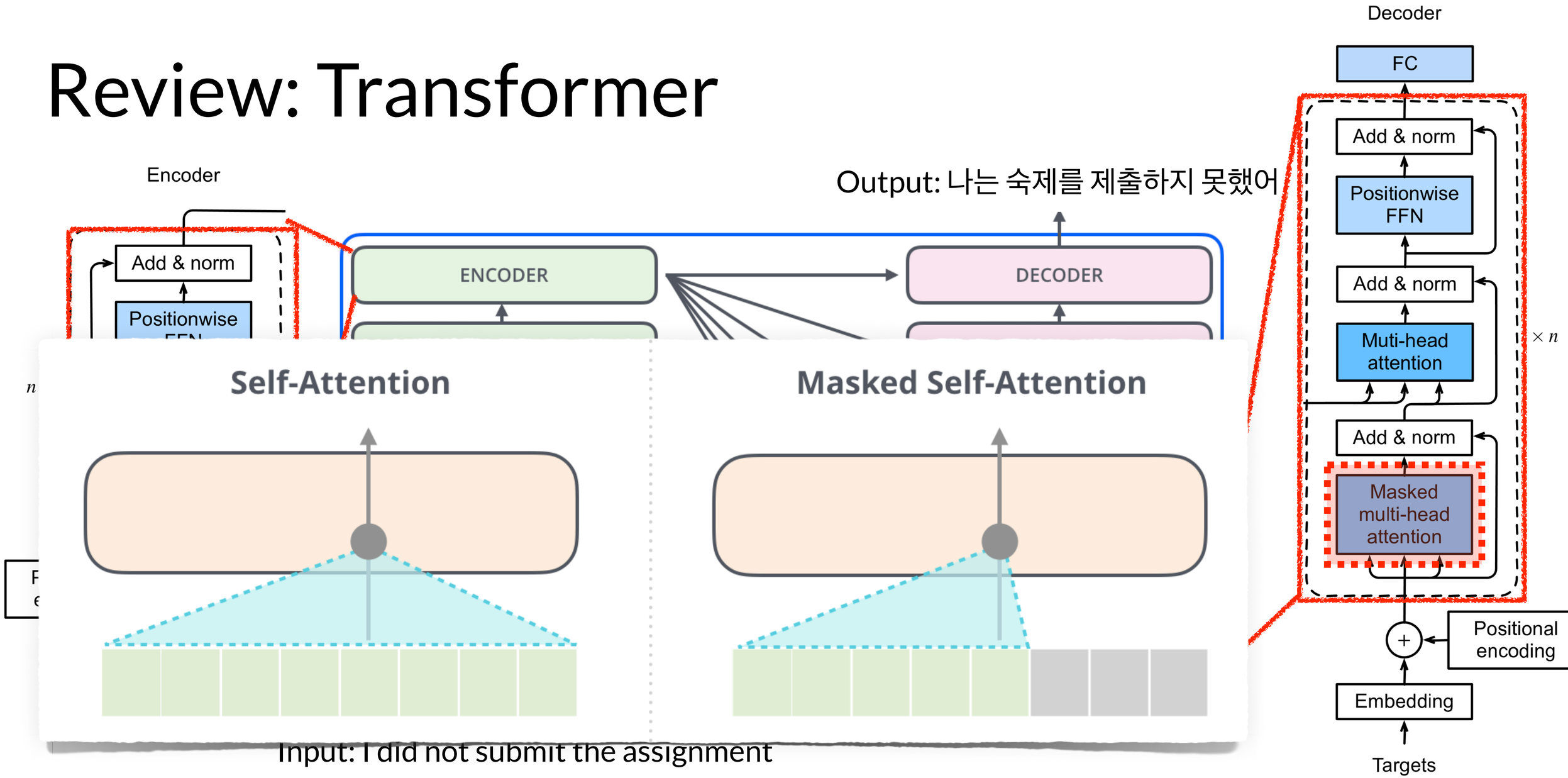
$$\mathbf{h}_m = f(\mathbf{W}_m^{(q)}\mathbf{x}, \{\mathbf{W}_m^{(k)}\mathbf{x}_i, \mathbf{W}_m^{(v)}\mathbf{x}_i\}_{i=1}^n) = \sum_{i=1}^n \alpha(\mathbf{W}_m^{(q)}\mathbf{x}, \mathbf{W}_m^{(k)}\mathbf{x}_i)\mathbf{W}_m^{(v)}\mathbf{x}_i$$



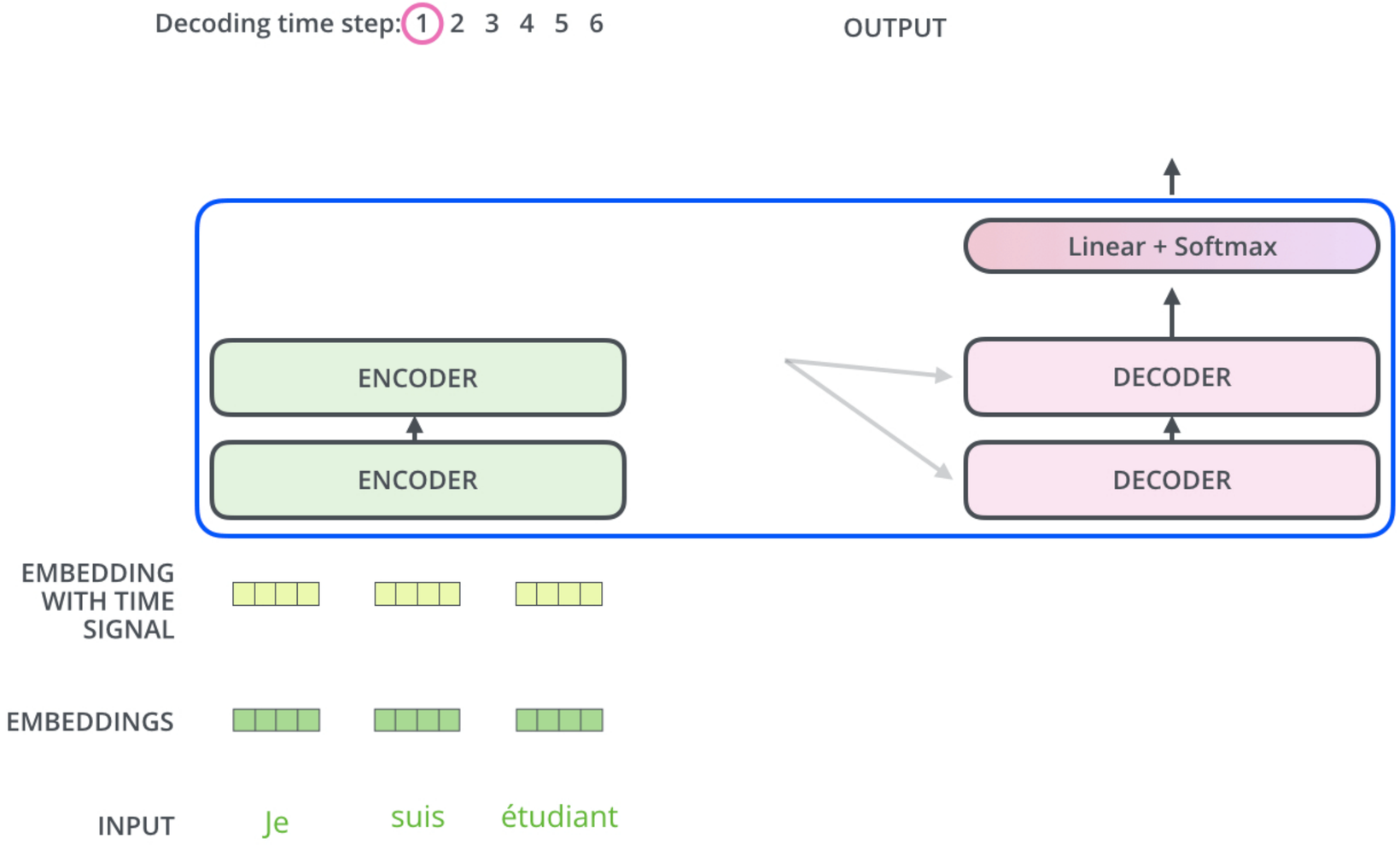
Review: Transformer



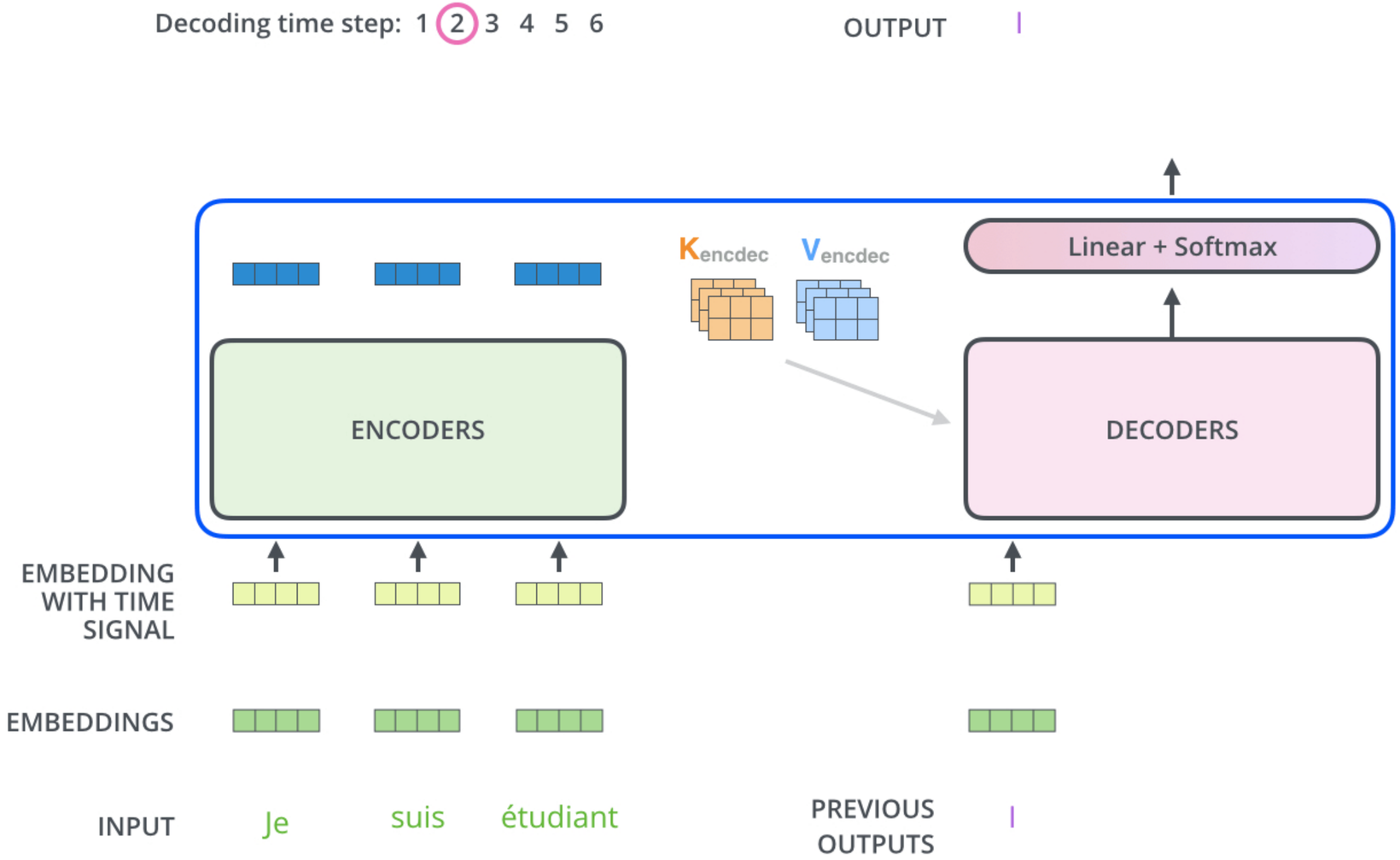
Review: Transformer



Review: Transformer



Review: Transformer



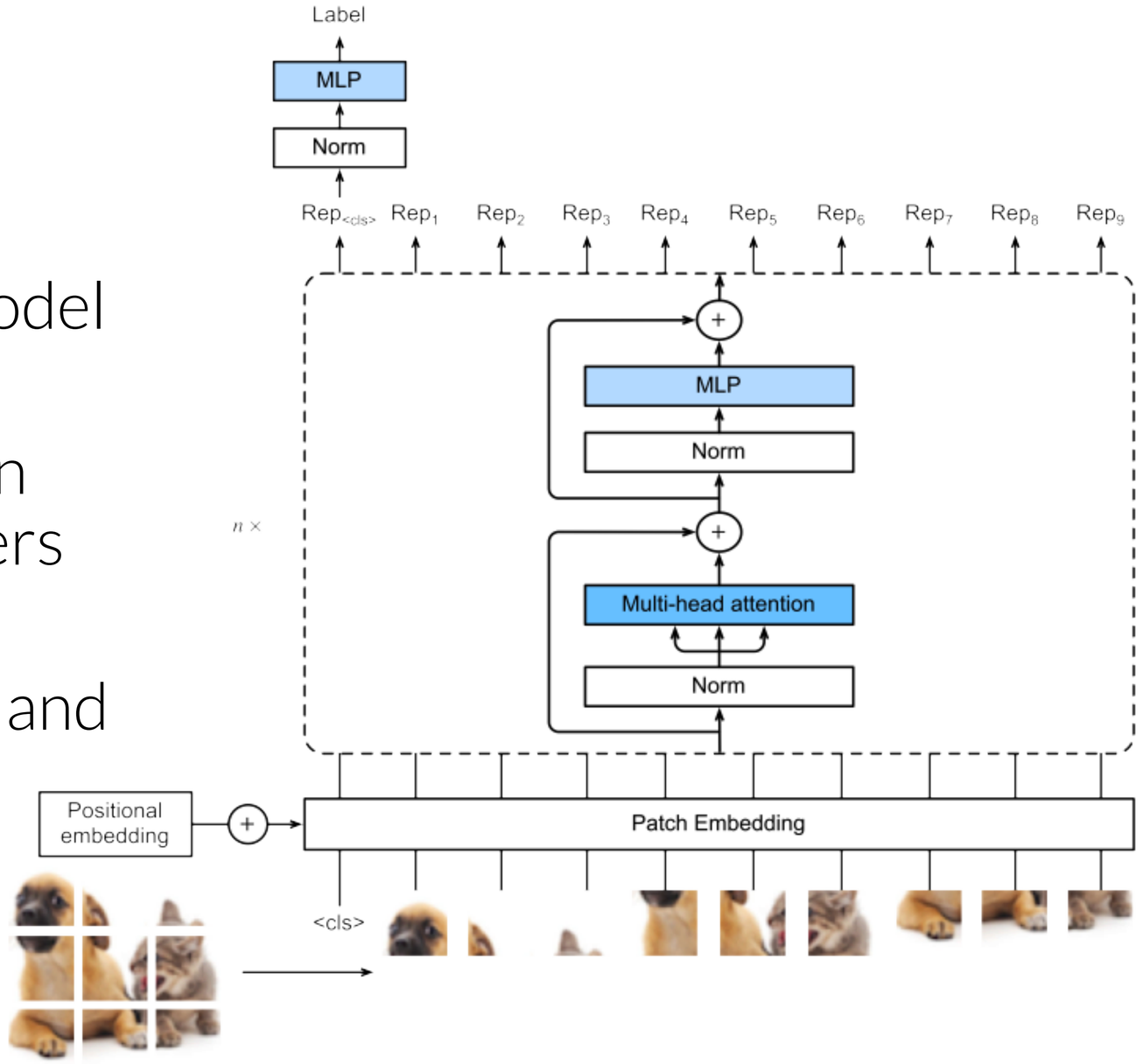
Advanced Transformers

- Encoder-Decoder Transformer
  - Classical Transformer (Vaswani et al., NeurIPS 2017)
- Encoder-only Transformer
  - BERT (Devlin et al., ACL 2019)
  - ViT (Dosovitskiy., ICLR 2021)
  - TokenGT (Kim et al., NeurIPS 2022)
- Decoder-only Transformer
  - GPT (Radford et al., 2019; Brown et al., NeurIPS 2020)

Vision Transformer

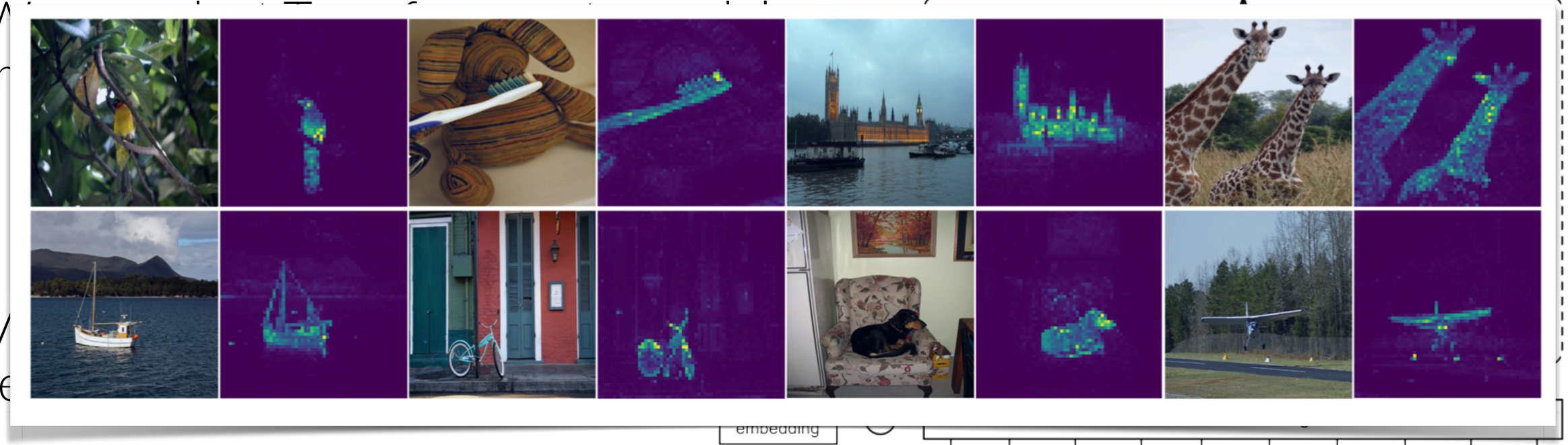
- We can adapt Transformers to model image data instead of CNNs
  - theoretically, self-attention can express **any** convolutional layers (Cordonnier et al., ***ICLR*** 2020)
- ViTs extract patches from images and feed them into a Transformer

Transformers became a game changer in computer vision!



Vision Transformer

- W
  in
- V
  fe

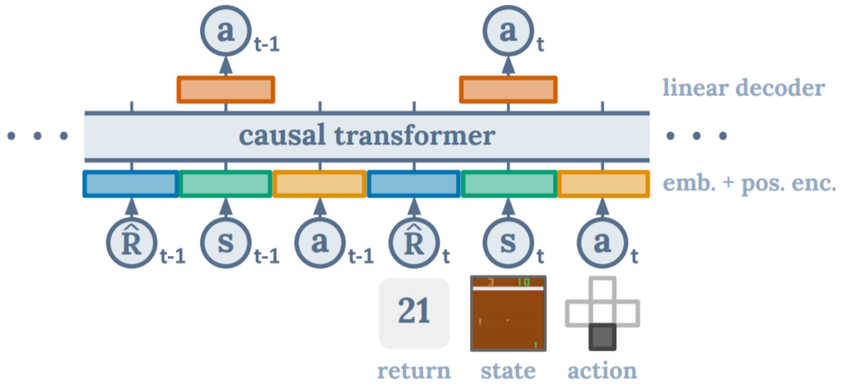


Self-attention of the [CLS] token on the heads the last layer without labels

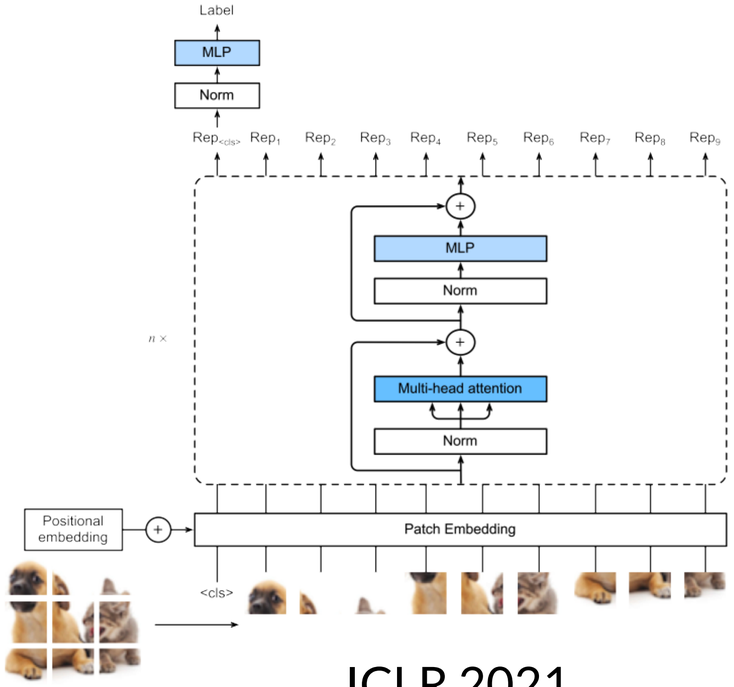
The Era of Transformer

we are rewriting textbooks…

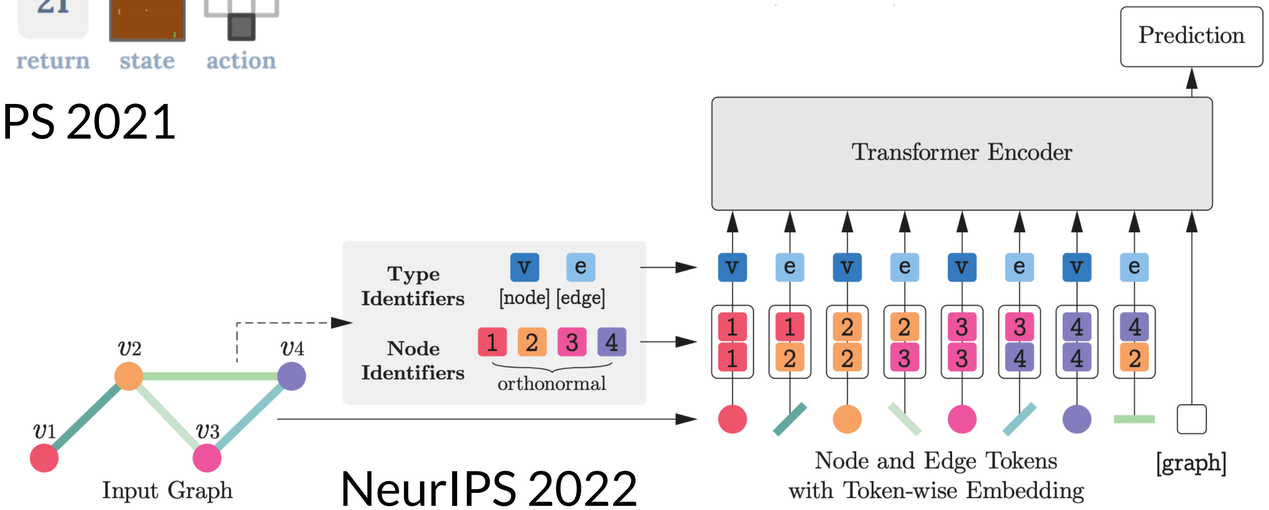
- NLP
  - BERT/GPT
- Computer Vision
  - Vision Transformer
- Graph
  - Graph Transformer
- Reinforcement Learning
  - Decision Transformer



NeurIPS 2021



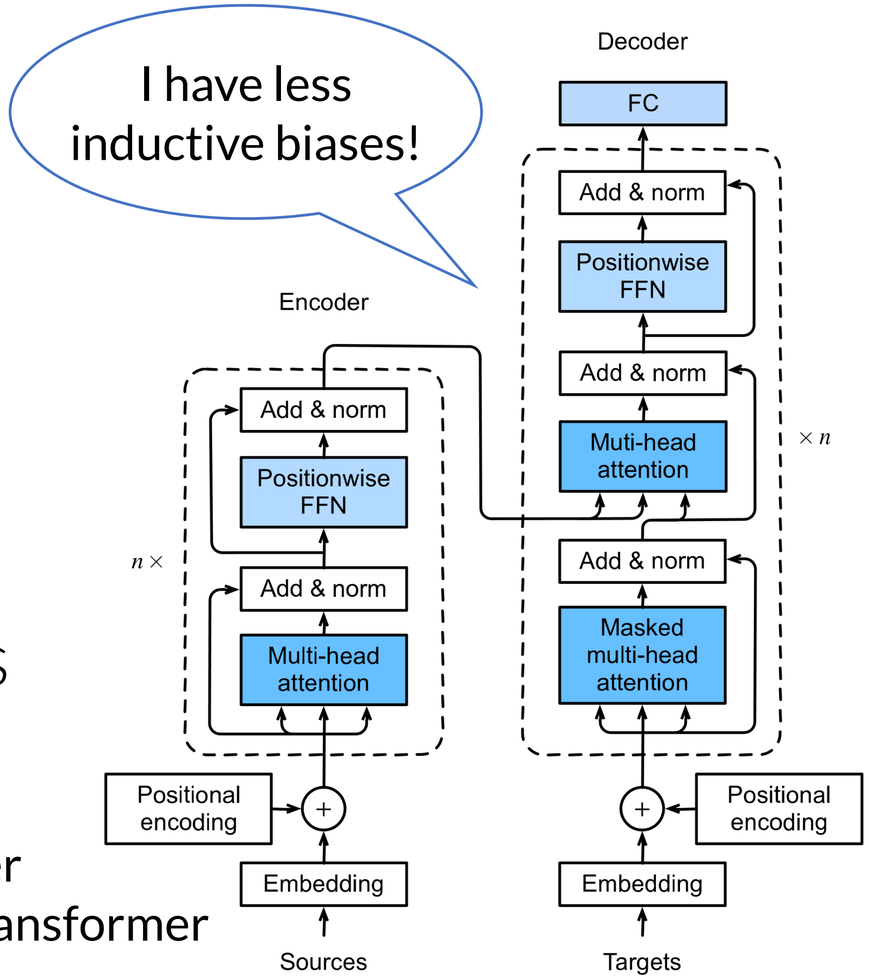
ICLR 2021



NeurIPS 2022

Design Principles in Deep Learning

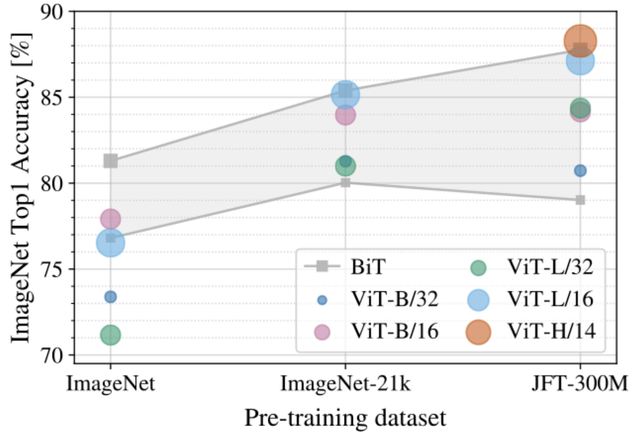
- Inductive Biases in Convolutional Networks
  - Locality Principle
  - Spatial Invariance
- Inductive Biases in Graph Neural Networks
  - Permutation Invariance
- Inductive Biases in Recurrent Neural Networks
  - Sequentiality
  - Temporal Invariance



Note that MLP has fewer inductive biases than Transformer

How Do Transformers Learn?

- For small datasets (ex: MNIST, ImageNet), Transformers do not outperform previous deep learning models
  - this is because **Transformers lack design principles** in CNNs
- However, when we train larger models on larger datasets (300M images), Transformers outperform previous approaches by a large margin

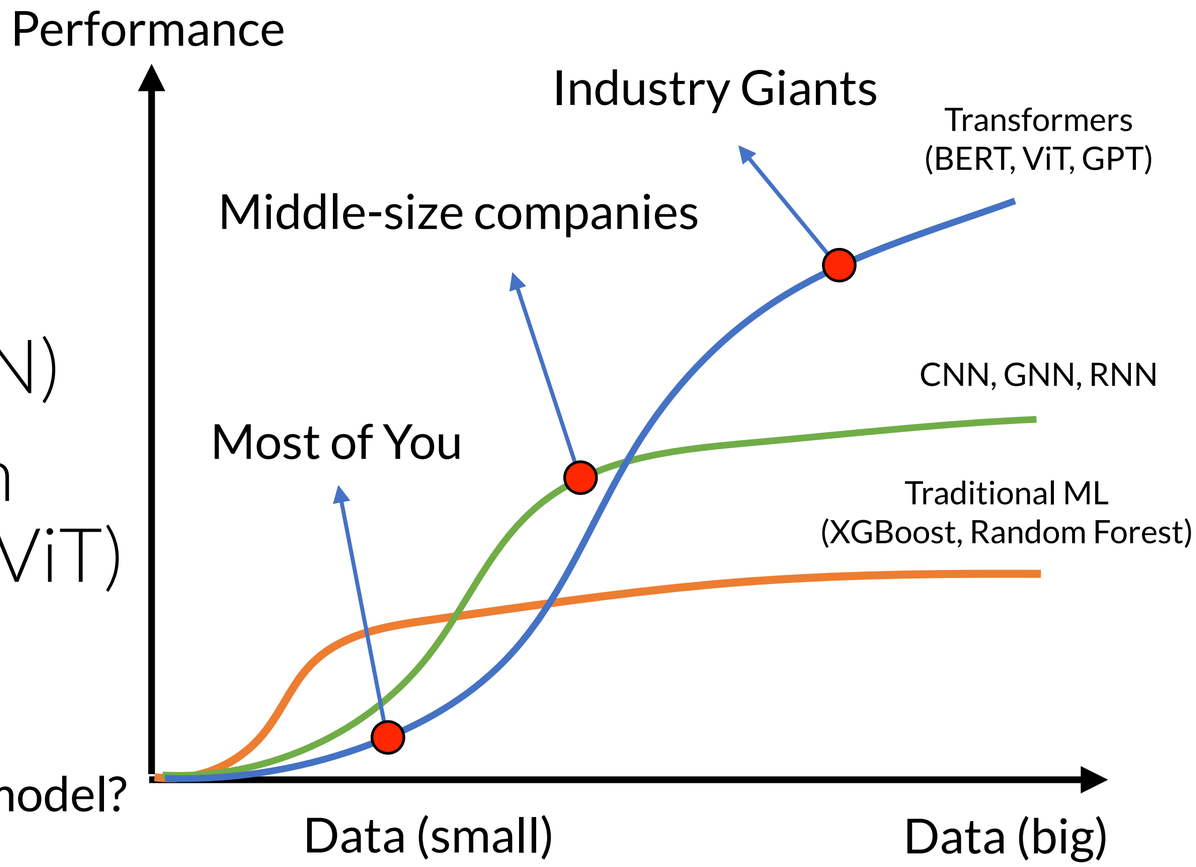


	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Expressive models require more data

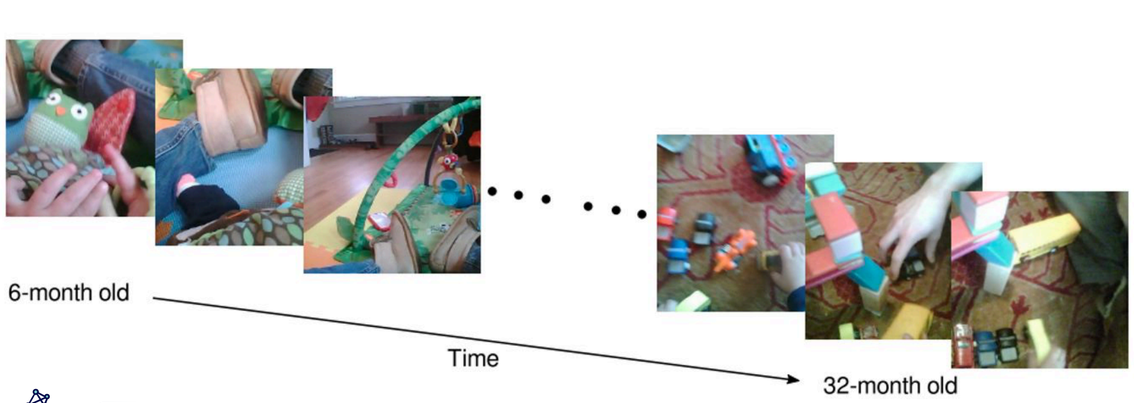
- If you have small number of data, then traditional ML performs well
- If you have more data, then you can try deep learning with well-designed inductive biases (e.g. CNN, GNN, RNN)
- If you have large number of data, then you can try Transformer-type (BERT, ViT) with (self-)supervised learning

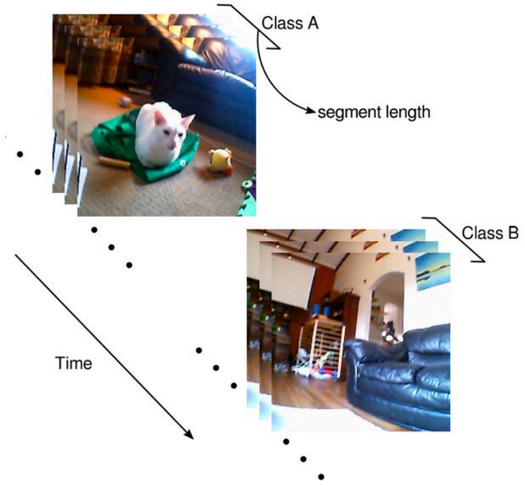
How can we obtain good data to train DL model?



Self-Supervised Training

- Transformers show impressive performance on various tasks
  - much of their success stems not only from their scalability bus also from large scale ***self-supervised (pre-)training***





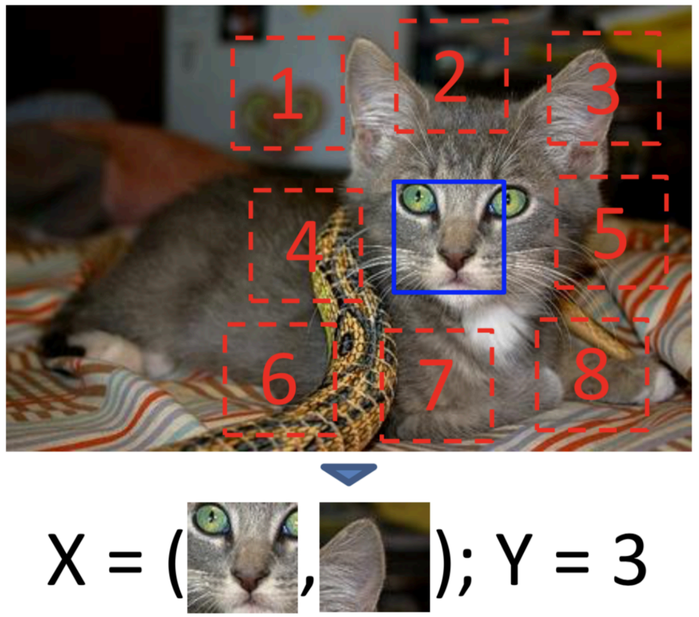
Children can learn high-level visual representations without strong supervision

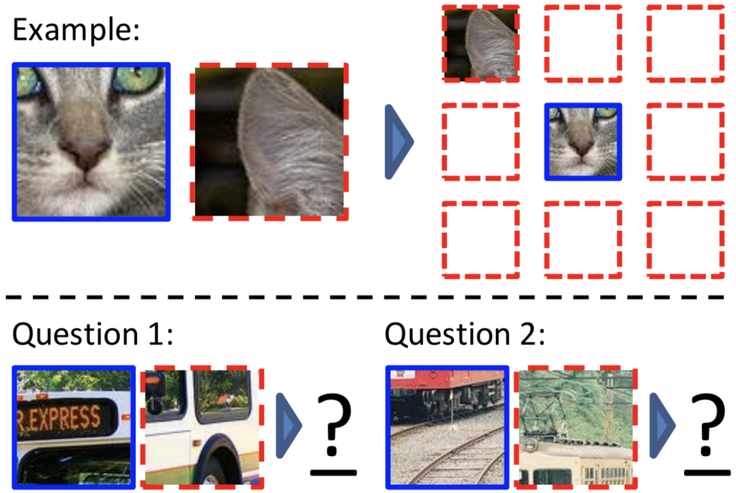
Self-Supervised Training

- Transformers show impressive performance on various tasks
  - much of their success stems not only from their scalability bus also from large scale ***self-supervised (pre-)training***
- Self-Supervised Training
  - a form of **unsupervised training** where the data (not the human) provides the supervision signal
  - we usually define a pretext task for which the model is forced to learn
  - the representations learned on the pretext task are subsequently used for a different downstream task

Example 1: position prediction

We train model to predict the relative position between random patches

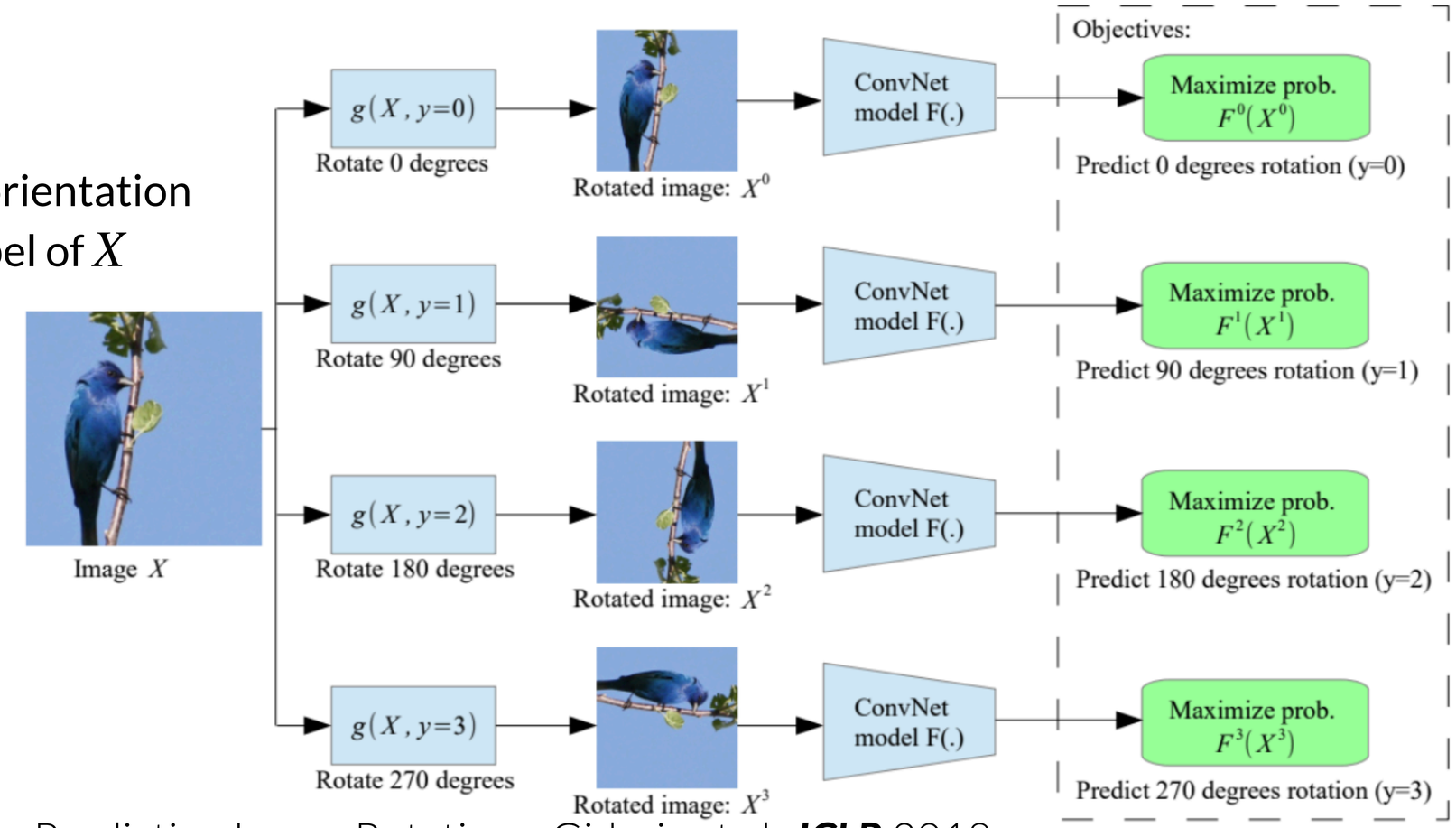




Unsupervised Visual Representation Learning by Context Prediction, Doersch et al., ***ICCV*** 2015

Example 2: rotation prediction

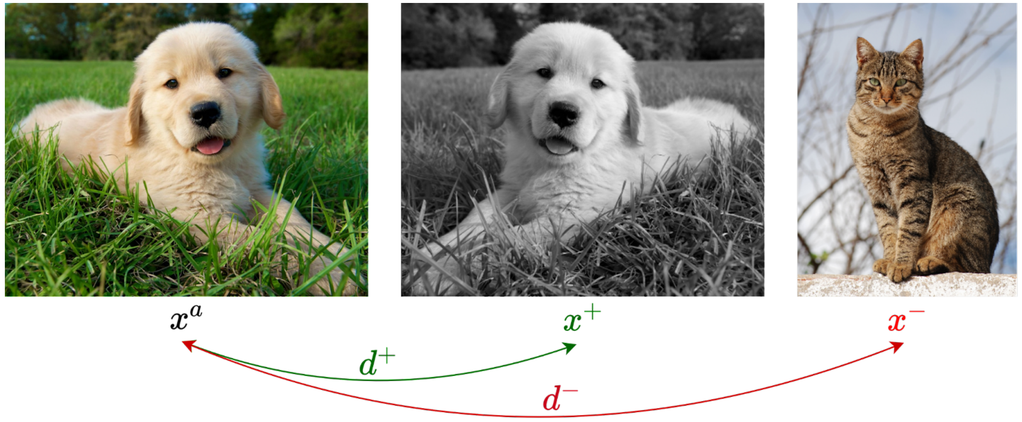
Our model can predict the orientation of the image without the label of X

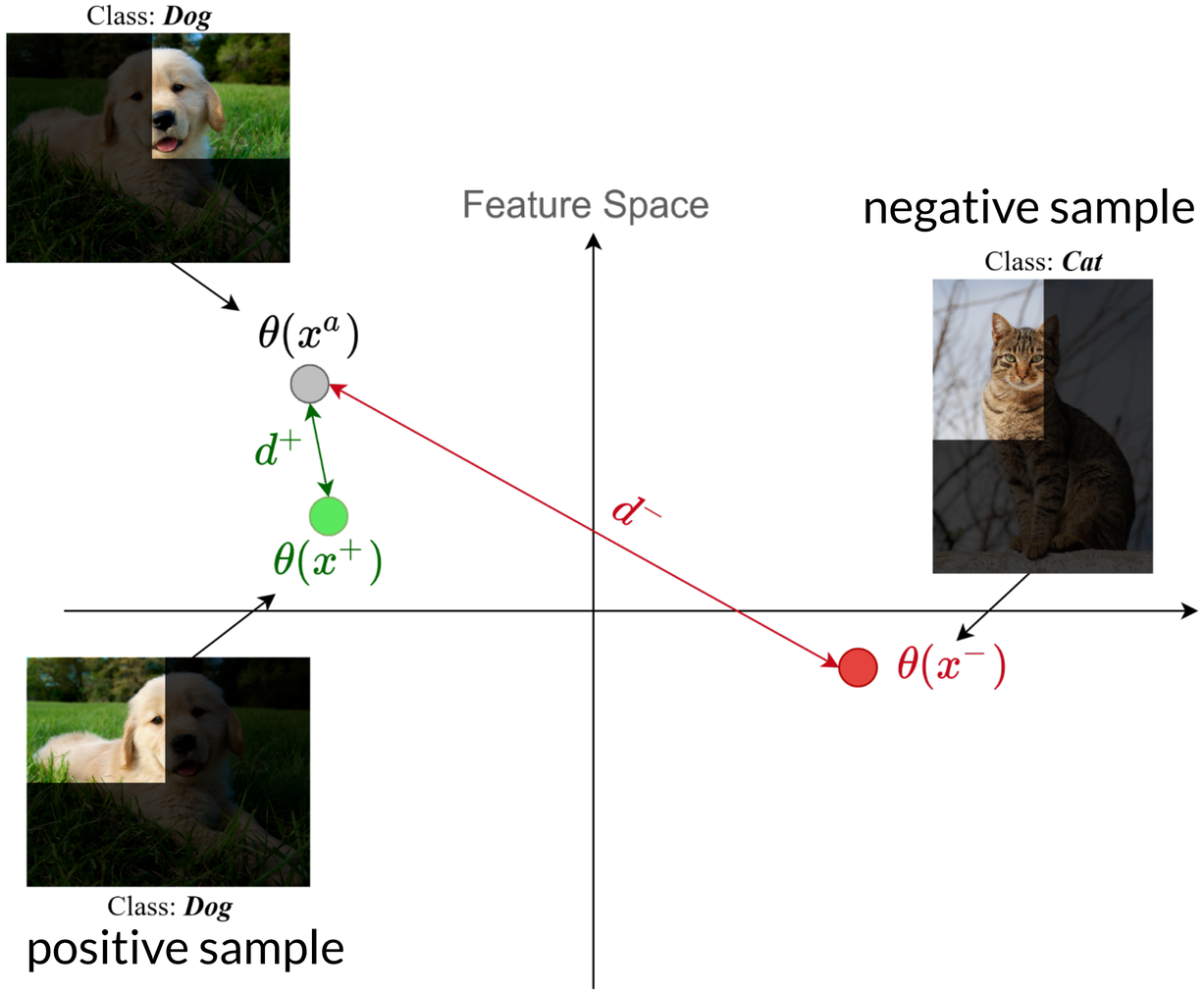


Unsupervised Representation Learning by Predicting Image Rotations, Gidaris et al., ***ICLR*** 2018

Example 3: contrastive learning

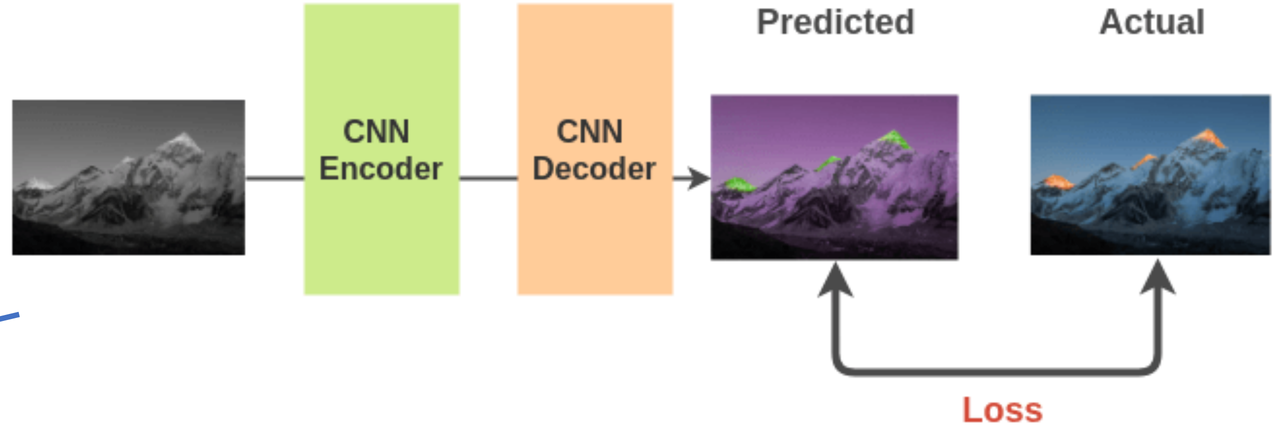
Our model can be trained by contrasting
an input with positive and negative examples

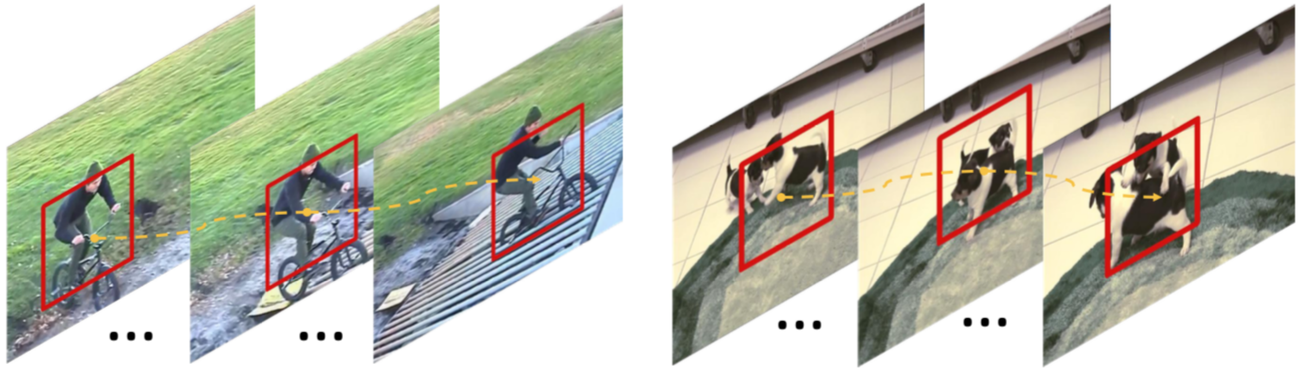


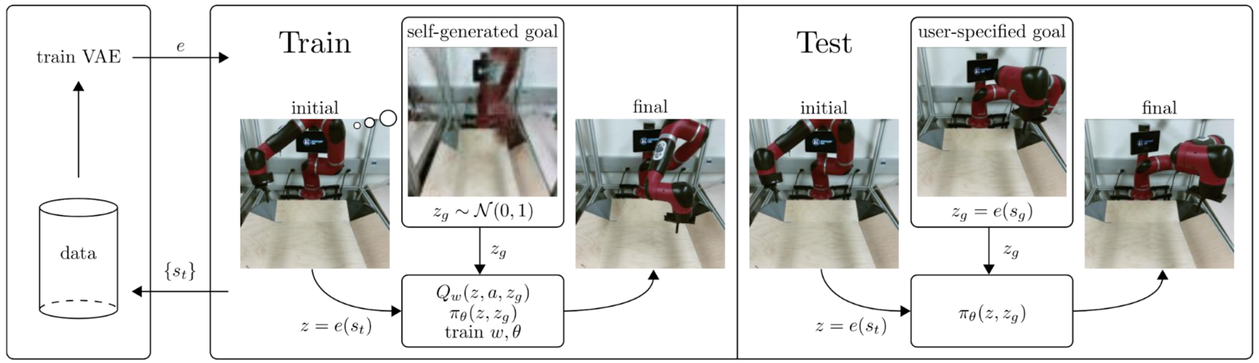


Other pretext tasks

- Image-based
  - colorization
  - generative modeling
- Video-based
  - tracking
  - frame sequence
- Control-based
  - goal generation

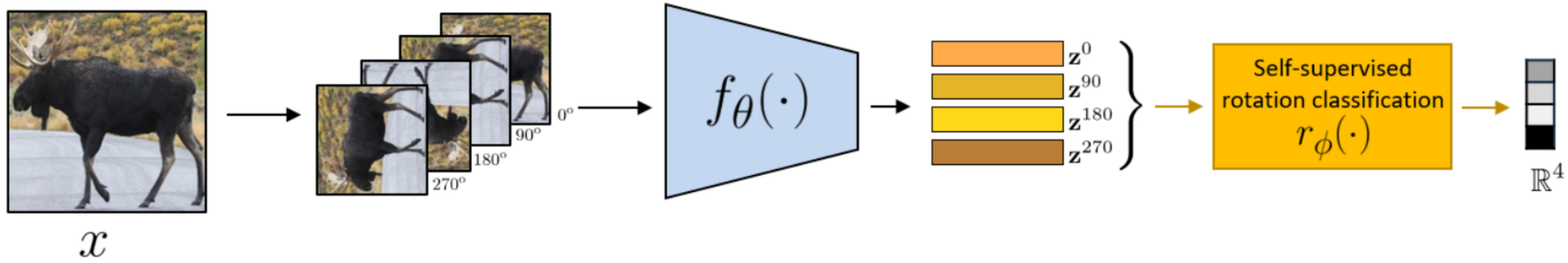






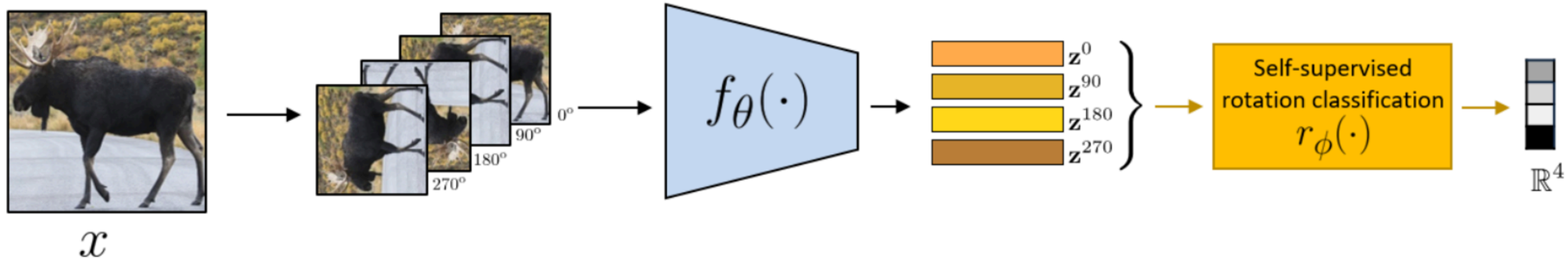
Self-Supervised Training Pipeline

- Stage 1: pre-train network on pretext task (without human labels)



Self-Supervised Training Pipeline

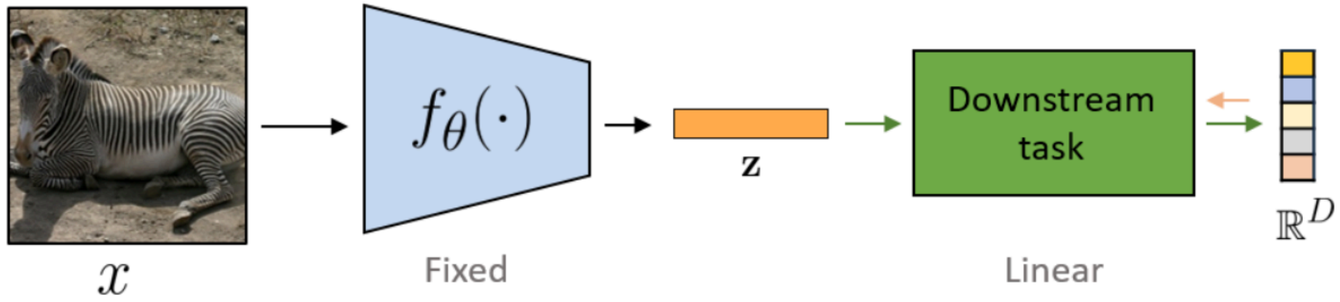
- Stage 1: pre-train network on pretext task (without human labels)



- Stage 2: **probe** classifier on learned features for new task with fewer labels

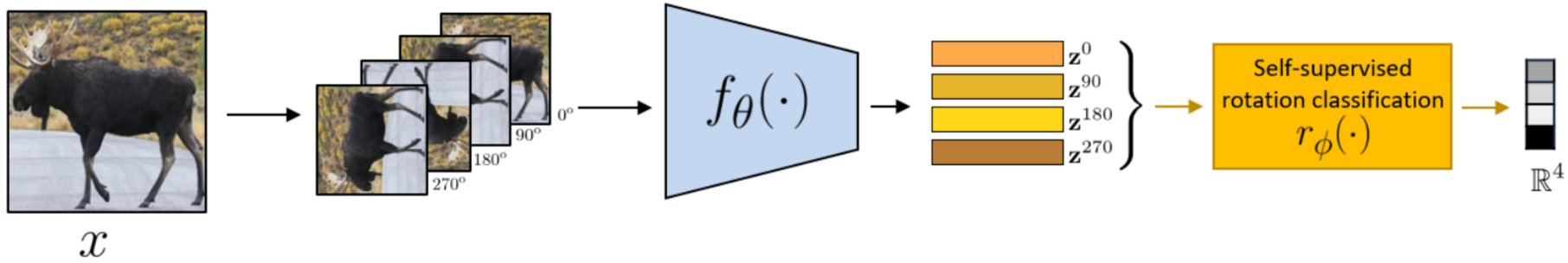
you have to freeze the pre-trained parameters

(Linear Probing)



Self-Supervised Training Pipeline

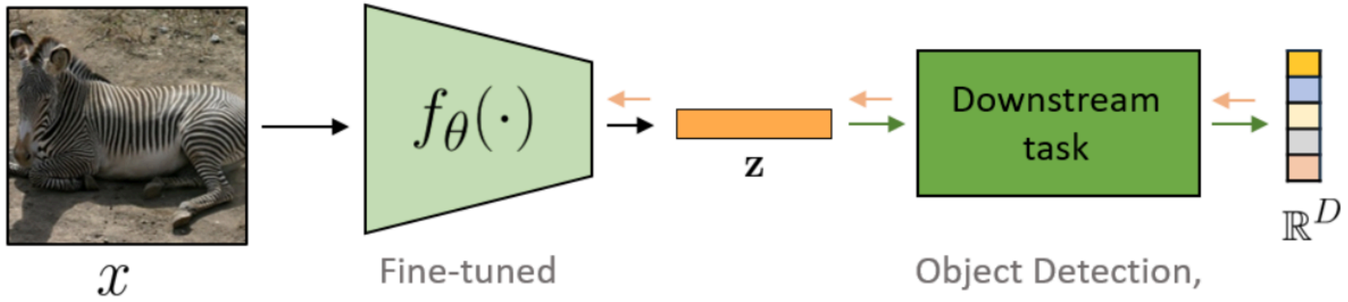
- Stage 1: pre-train network on pretext task (without human labels)



- Stage 2: **fine-tune** network for new task with fewer labels

fine-tuning requires well-tuned classifier

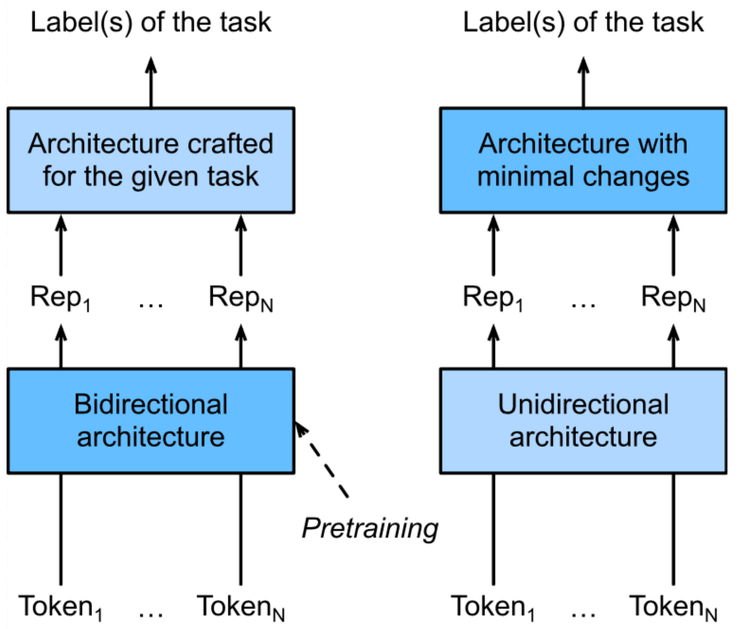
(Fine-Tuning)



Pretraining for Representation Learning in NLP

- From context-independent to context-sensitive representation
  - ELMo uses pre-trained ***bidirectional*** LSTM but task-specific architectures
  - GPT is task-agnostic but encodes context left-to-right → ***unidirectional***
- While ELMo freezes parameters of the pre-trained model, GPT fine-tunes all the parameters in the pre-trained Transformer decoder

both methods recorded SOTA in many NLP tasks



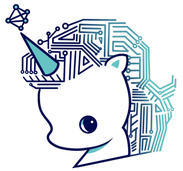
ELMo (2018) (LSTM)

GPT (2018) (Transformer)

Bidirectional RNN

- We often need sequential models which can deal with information of the subsequent data
- It is possible to have a mechanism of ***bidirectional*** path in RNN

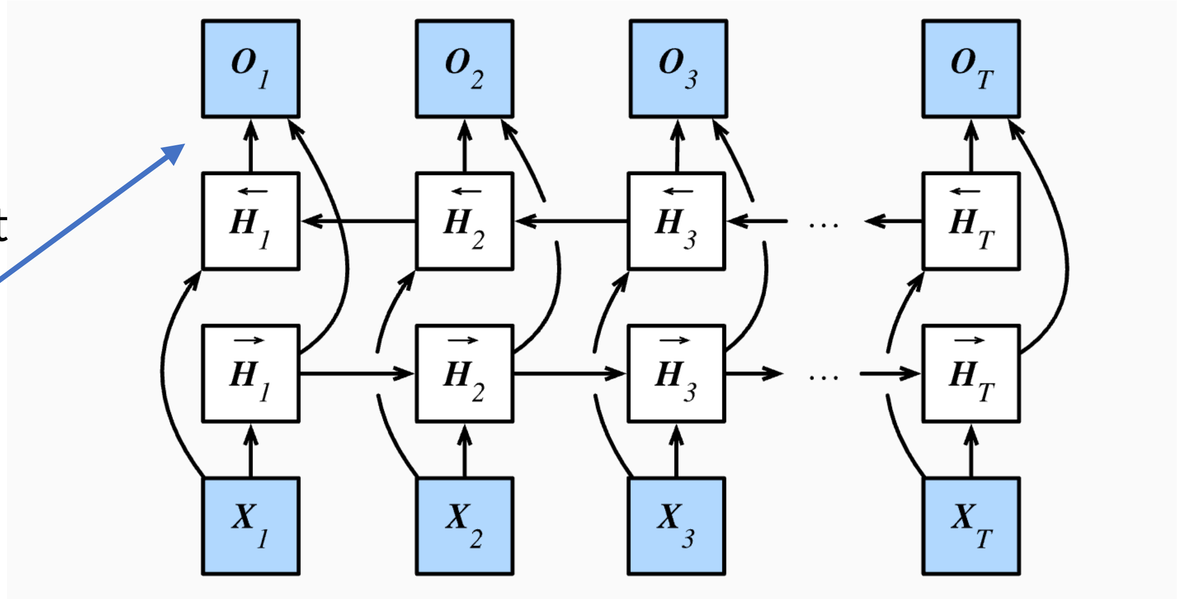
$$\overrightarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(f)} + \overrightarrow{\mathbf{H}}_{t-1} \mathbf{W}_{hh}^{(f)} + \mathbf{b}_h^{(f)}),$$
$$\overleftarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(b)} + \overleftarrow{\mathbf{H}}_{t+1} \mathbf{W}_{hh}^{(b)} + \mathbf{b}_h^{(b)}).$$



we use concatenation for combining both hidden variables

```
1. I am _____
2. I am _____ very hungry.
3. I am _____ very hungry, I could eat half a pig.
```

concat



An architecture of bidirectional RNN

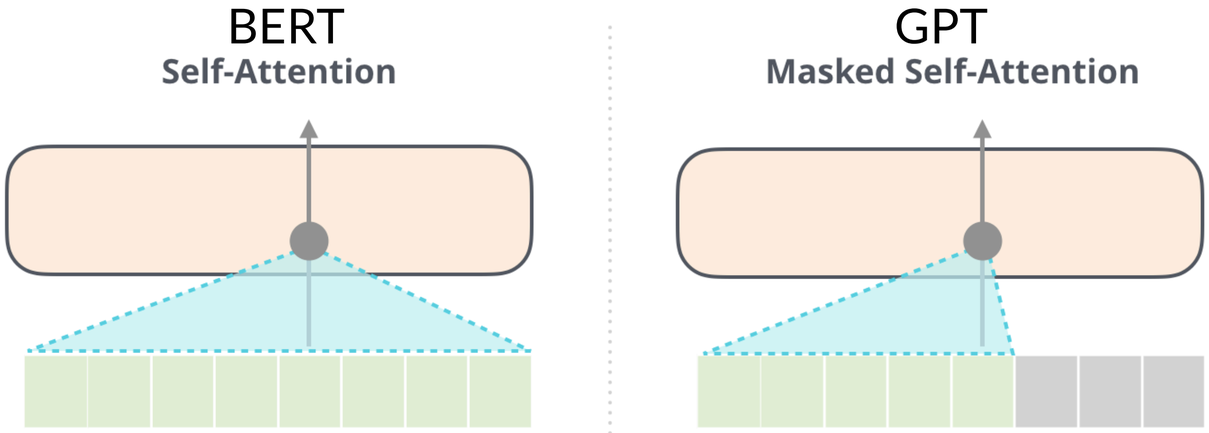
When we use bidirectional RNN?

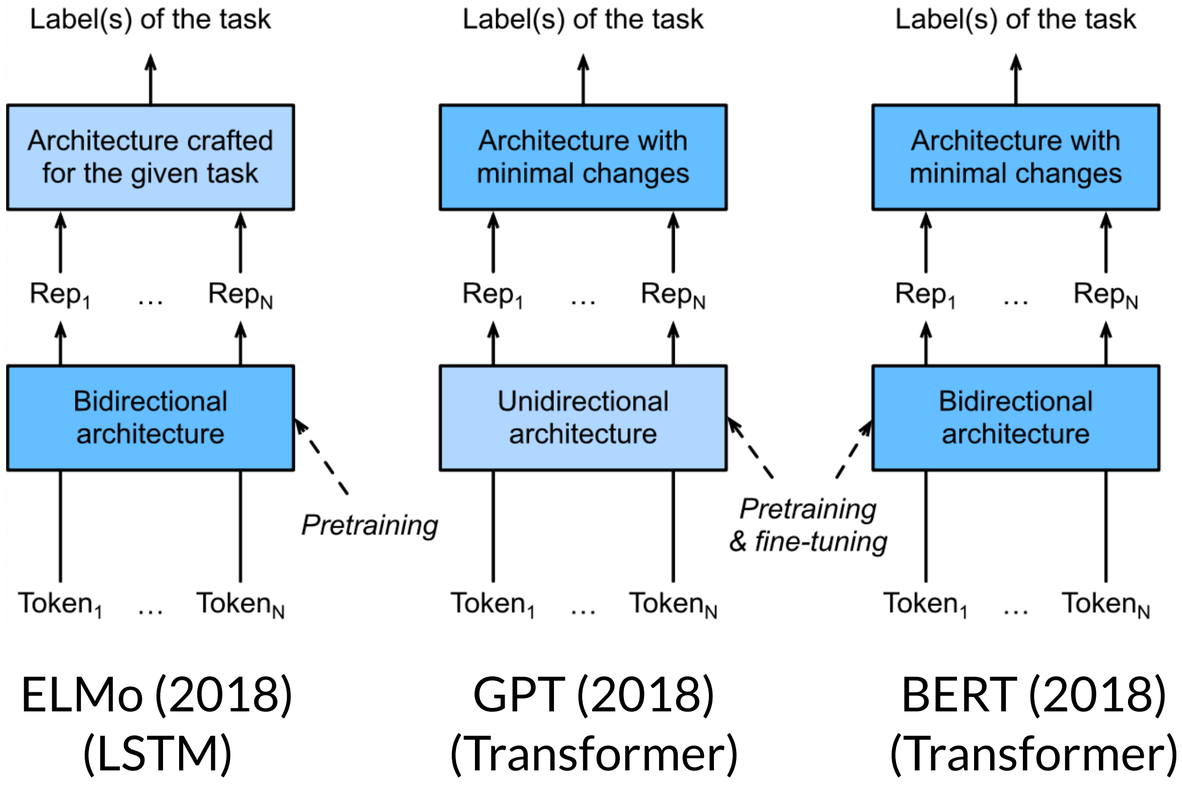
- We use bidirectional RNN (or LSTM) when we need information from both ends of the sequence
  - we use information from both future and past observation
  - if we apply bidirectional RNN naively, we would get bad performance (+ exceedingly ***slow*** training)
- In practice, bidirectional layers are used carefully
  - etc) filling in missing words, annotating tokens, translation

BERT

ELMo, GPT, and BERT have revolutionized solutions to various NLP tasks

- BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers)
  - uses a pre-trained Transformer hence requires minimal changes
  - utilizes Transformer encoder to get bidirectional context





ELMo (2018) (LSTM)

GPT (2018) (Transformer)

BERT (2018) (Transformer)

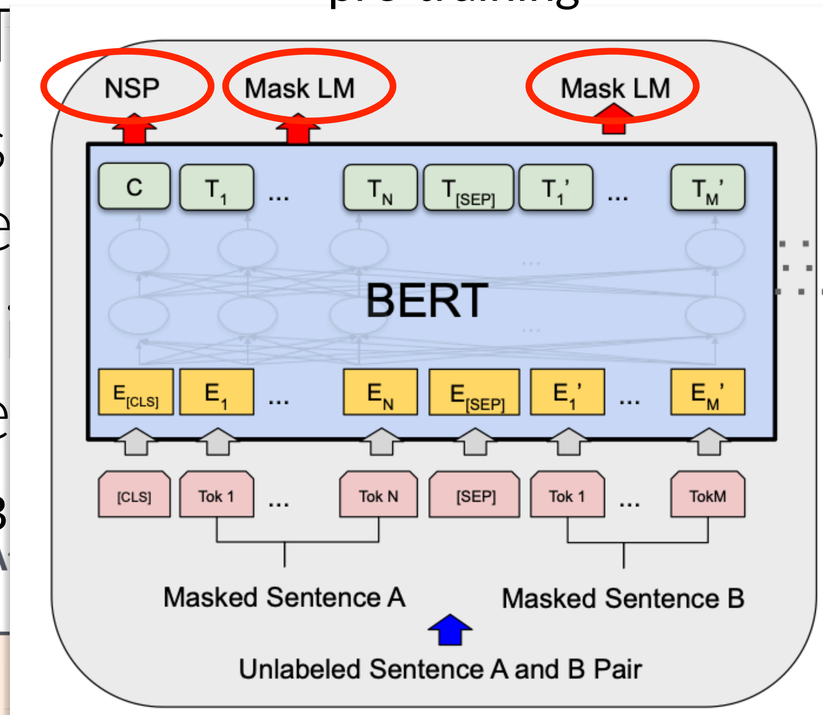
BERT

BERT also use fine-tuning as GPT but proposes a different type of pre-training

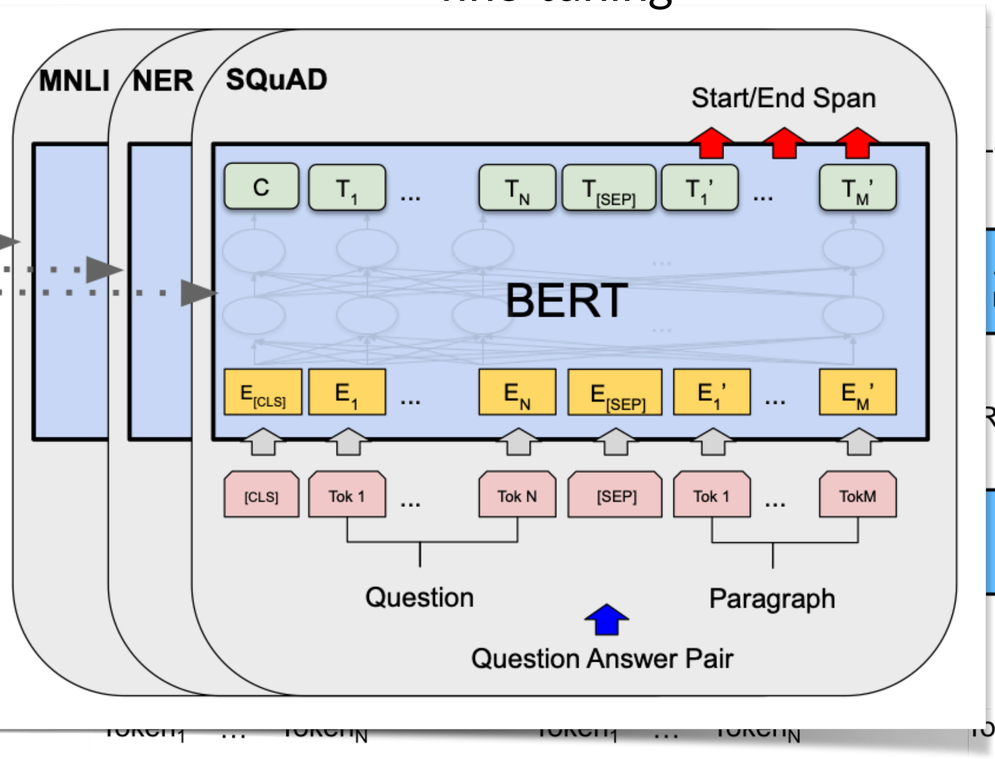
- BERT
  - us
    he
  - ut
    ge

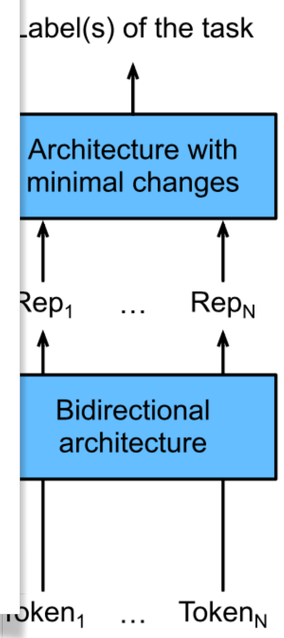
B
Self-A

pre-training



fine-tuning





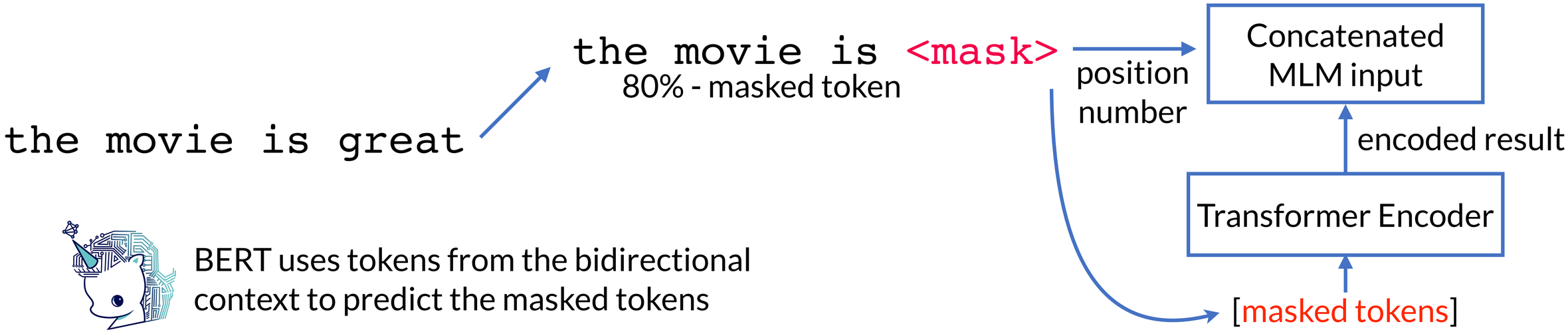
ELMo (2018)
(LSTM)

GPT (2018)
(Transformer)

BERT (2018)
(Transformer)

How to *pre-train* BERT?

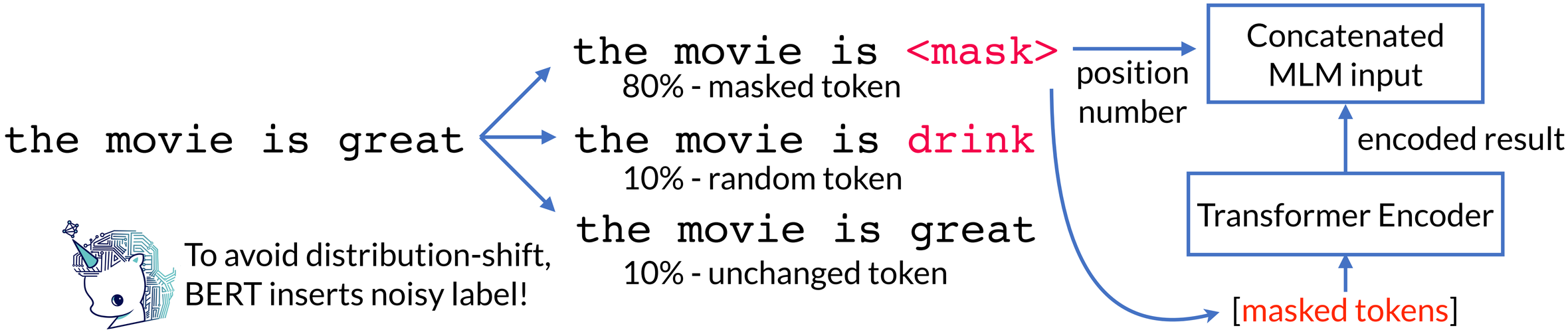
- Masked Language Modeling **(MLM) → encodes context bidirectionally**
  - input text with randomly masked tokens is fed into a Transformer encoder to predict the ***masked tokens*** in a self-supervised fashion



BERT uses tokens from the bidirectional context to predict the masked tokens

How to *pre-train* BERT?

- Masked Language Modeling **(MLM) → encodes context bidirectionally**
  - input text with randomly masked tokens is fed into a Transformer encoder to predict the ***masked tokens*** in a self-supervised fashion



How to *pre-train* BERT?

- Masked Language Modeling **(MLM) → encodes context bidirectionally**
  - input text with randomly masked tokens is fed into a Transformer encoder to predict the ***masked tokens*** in a self-supervised fashion
- Next Sequence Prediction **(NSP) → understanding logical relations**
  - to explicitly model the logical relationship between text pairs, BERT considers a ***binary classification*** as another pre-training task
  - for half of the time consecutive sentences with the label `True` are generated, while for the other half of the time the second sentence is randomly sampled from the corpus with the label `False`.

How to *pre-train* BERT?

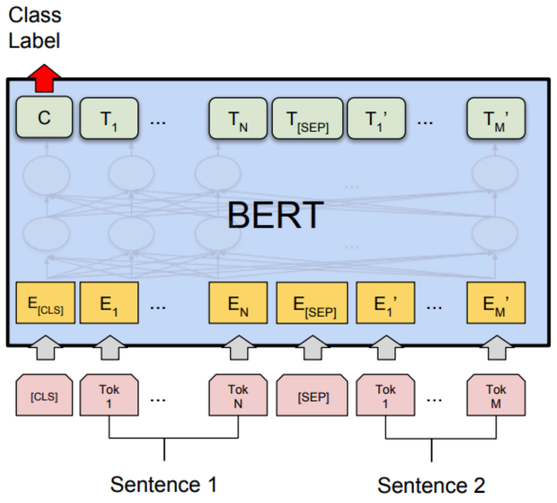
we combine loss functions for pre-training → multi-task learning!

- Masked Language Modeling **(MLM)** → **encodes context bidirectionally**
  - input text with randomly masked tokens is fed into a Transformer encoder to predict the ***masked tokens*** in a self-supervised fashion
- Next Sequence Prediction **(NSP)** → **understanding logical relations**
  - to explicitly model the logical re
    considers a ***binary classificatio***
  - for half of the time consecutive
    generated, while for the other h
    randomly sampled from the co

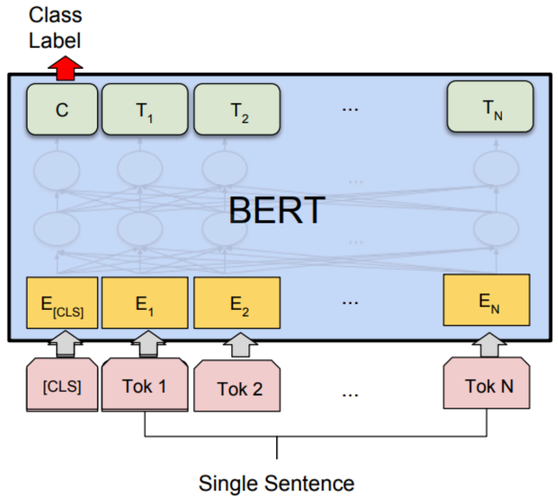
Tasks		Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	NSP	84.4	88.4	86.7	92.7	88.5
No NSP	MLM	83.9	84.9	86.5	92.6	87.9
LTR & No NSP		82.1	84.3	77.5	92.1	77.8
+ BiLSTM		82.1	84.1	75.7	91.6	84.9

How to *fine-tune* BERT?

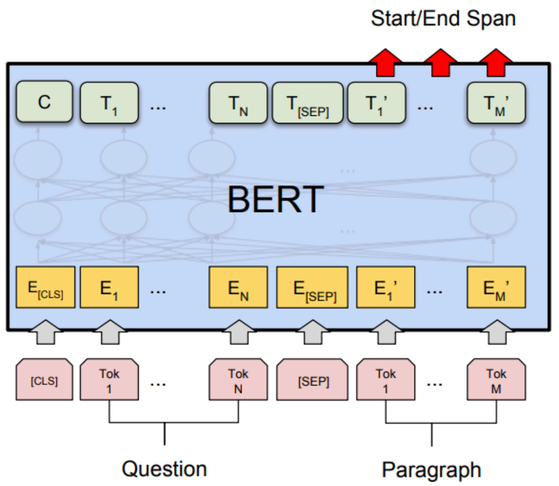
- Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks
  - for each task, we simply plug in the task specific inputs and outputs into BERT and fine-tune all the parameters end-to-end



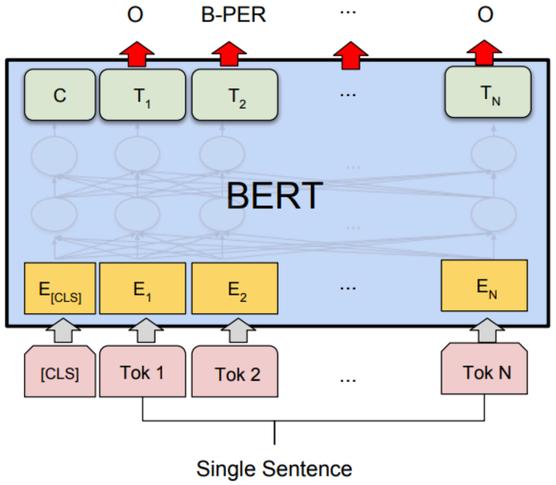
Sentence pair classification



Single sentence classification



Question & Answering

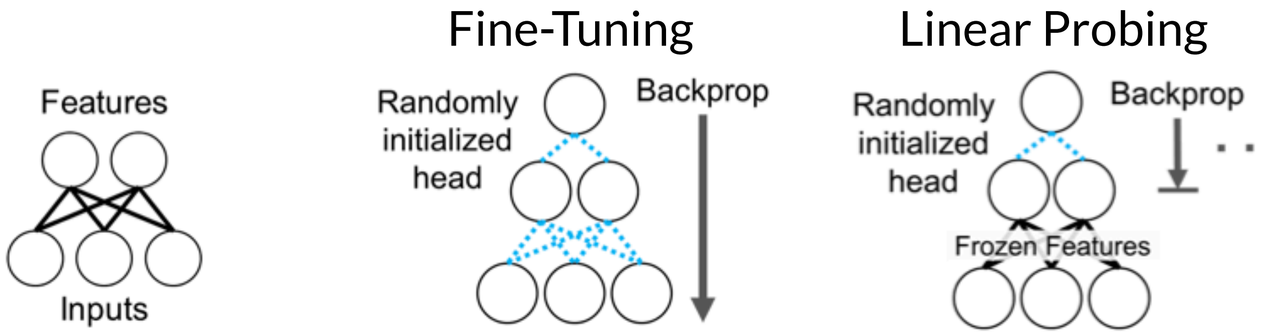


Single sentence tagging

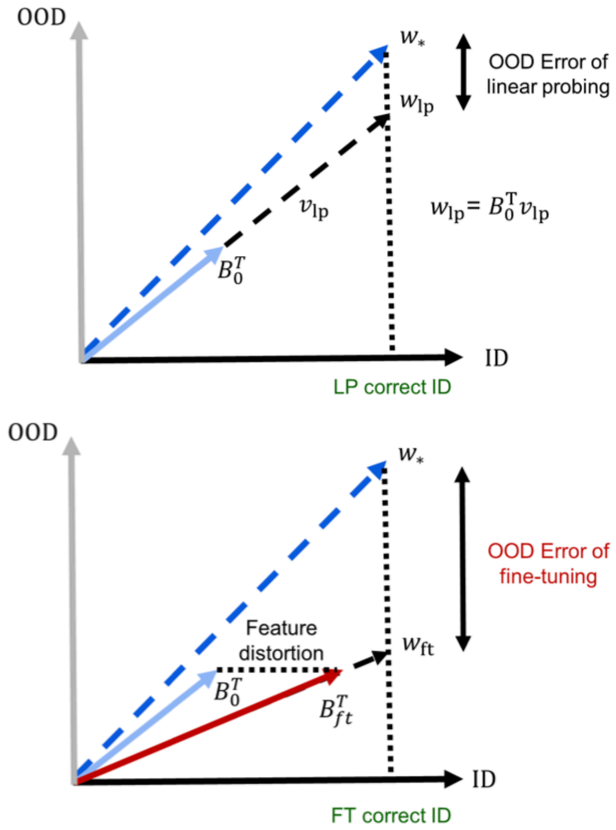
Linear Probing vs Fine-Tuning

how can we combine the two methods?

- FT can do worse than LP when ***distribution shift*** occurs



ID test	85.1%	82.9%
OOD test	59.3%	66.2%

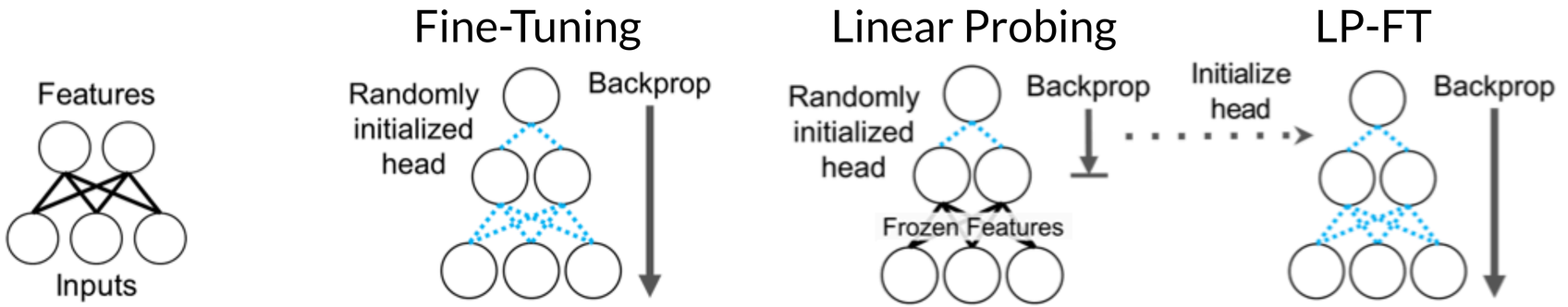


Fine-Tuning ,can Distory Pretrained Features and Underperform Out-of-Distribution, Kumar et al., ***ICLR*** 2022

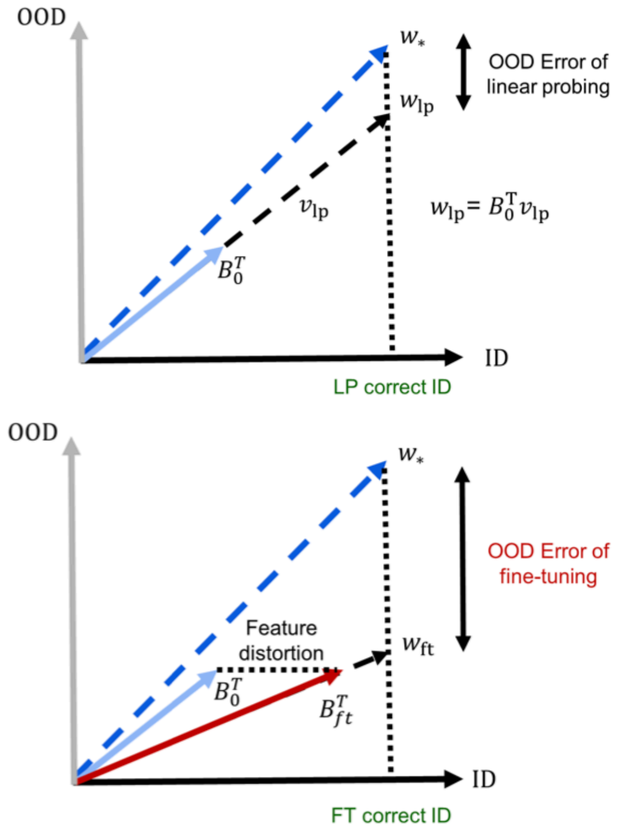
Linear Probing *then* Fine-Tuning

very simple technique
but works quite well!

- FT can do worse than LP when ***distribution shift*** occurs
- LP-FT works better than fine-tuning and linear probing



ID test	85.1%	82.9%	**85.7%**
OOD test	59.3%	66.2%	**68.9%**



Fine-Tuning ,can Distory Pretrained Features and Underperform Out-of-Distribution, Kumar et al., ***ICLR*** 2022

Q&A