Deep learning

13.3. Transformer Networks

François Fleuret

https://fleuret.org/dlc/

UNIVERSITÉ DE GENÈVE

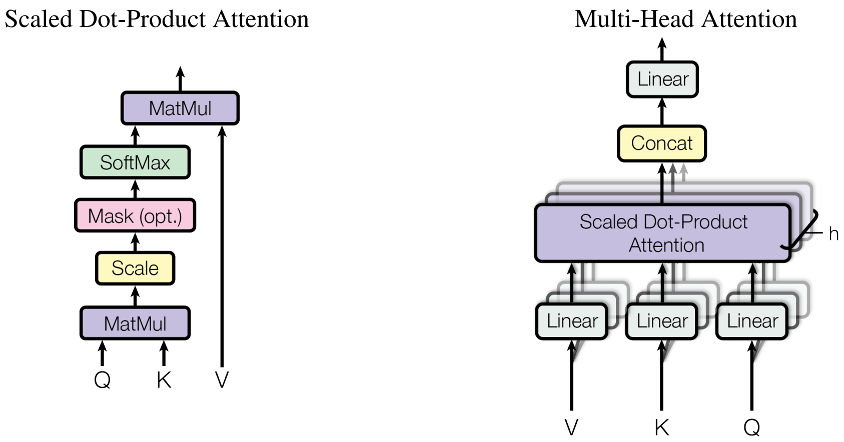
Vaswani et al. (2017) proposed to go one step further: instead of using attention mechanisms as a supplement to standard convolutional and recurrent operations, they designed a model composed of attention layers only.

They designed this "transformer" for a sequence-to-sequence translation task, but it is currently key to state-of-the-art approaches across NLP tasks.

Notes

The standard practice is to train a transformer in a non-supervised manner on large unlabeled datasets such as Wikipedia–or re-use a pre-trained transformer–and then fine tune it in a supervised manner for tasks which require a ground truth such as sentiment analysis.

They first introduce a multi-head attention module.



(Vaswani et al., 2017)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}\,(H_1, \dots, H_h)\,W^O$$

$$H_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right), \; i = 1, \dots, h$$

with

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, \; W_i^K \in \mathbb{R}^{d_{model} \times d_k}, \; W_i^V \in \mathbb{R}^{d_{model} \times d_v}, \; W^O \in \mathbb{R}^{hd_v \times d_{model}}$$

Notes

The "scaled dot-product attention" (left) is very close to the attention module we saw in lecture 13.2. "Attention Mechanisms", with the addition of an optional masking (in pink). This may be useful when such a module is used for a generative auto-regressive operation and the attention should be causal, looking only to the past.

The attention is a function of the keys, queries, and values. The only difference with what was seen in the previous course is that the attention matrix is rescaled with the dimension of the embedding, which matters quite a lot.

In the multi-head attention, each head h has its own processing of the input keys, queries, and values through respectively W_i^K, W_i^Q, and W_i^V. And there is one final processing W^O applied on the concatenated results of the multiple heads.

Their complete **Transformer** model is composed of:

- An **encoder** that combines $N = 6$ modules, each composed of a multi-head attention sub-module, and a [per-token] one hidden-layer MLP, with residual pass-through and layer normalization.
- A **decoder** with a similar structure, but with causal attention layers to allow for regression training, and additional attention layers that attend to the encoder final keys and values.

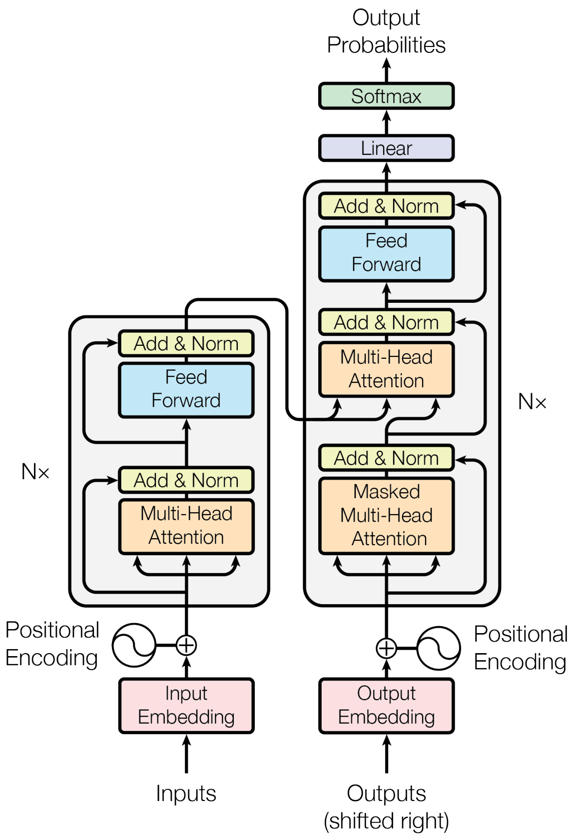
Positional information is provided through an **additive** positional encoding of same dimension d_{model} as the internal representation, and is of the form

$$PE_{t,2i} = \sin\left(\frac{t}{10{,}000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{t,2i+1} = \cos\left(\frac{t}{10{,}000^{\frac{2i+1}{d_{model}}}}\right).$$

Notes

Contrary to what we previously saw with the concatenated binary positional encoding, here the position is provided as additive encoding, where t is the position in the sequence, and $2i$ and $2i + 1$ the dimension.



"Original" Transformer (Vaswani et al., 2017).

Notes

This is a depiction of the standard transformer architecture for sequence-to-sequence translation. It consists of an encoder (left part) and a decoder (right part). Both are a stack of $N = 6$ modules. Each token (subword) of the input sequence is encoded with a look-up table to get its embedding of dimension d, so that the input is a tensor of size $T \times d$. Then the positional encoding of same size is added to it.

Each of the N modules of the encoder is composed of a multi-head self-attention operation followed by a "feed forward" operation that applies a one hidden layer perceptron at every position of the sequence separately. This can be implemented with 1×1 convolutions. Both the self-attention and the feed-forward are combined with residual pass-through.

The decoder is an auto-regressive model, and each of its module has a multi-head self-attention operation, then an attention that attends to the encoder, and a feed-forward operation. The self-attention is masked to make it causal, i.e. it takes into account only the part of the sequence already generated. The attention to the encoder is not masked but its keys and values are functions of the outputs of the corresponding module in the encoding stack.

The architecture is tested on English-to-German and English-to-French translation using the standard WMT2014 datasets.

- English-to-German: 4.5M sentence pairs, 37k tokens vocabulary.
- English-to-French: 36M sentence pairs, 32k tokens vocabulary.
- 8 P100 GPUs (150 TFlops FP16), 0.5 day for the small model, 3.5 days for the large one.

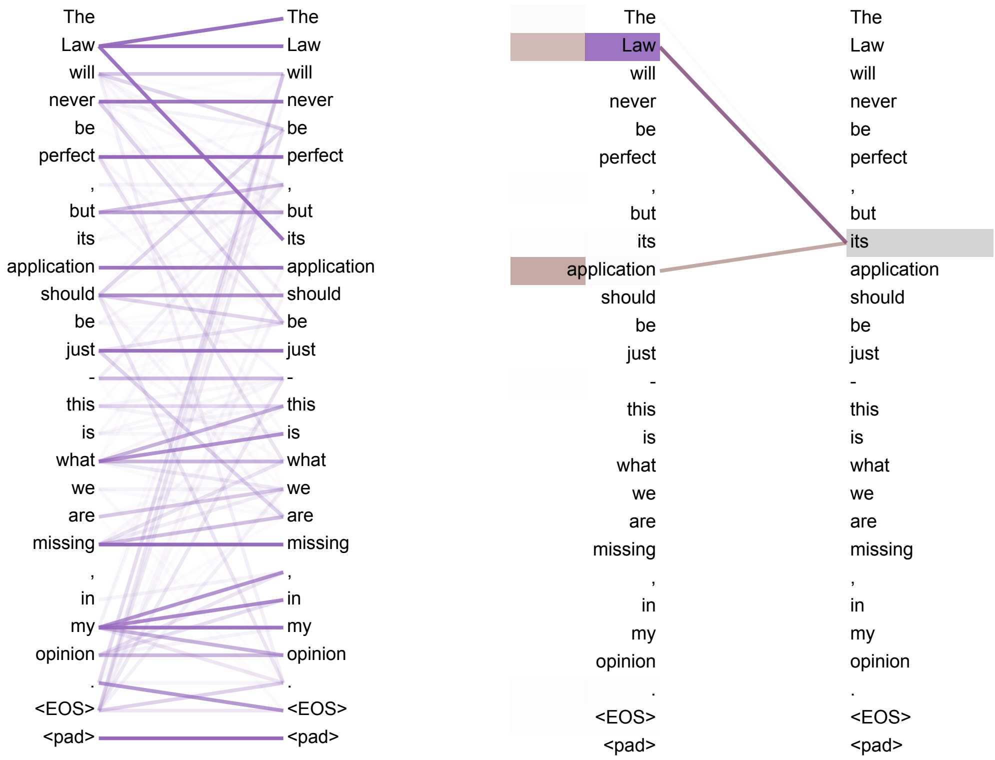
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

(Vaswani et al., 2017)

Notes

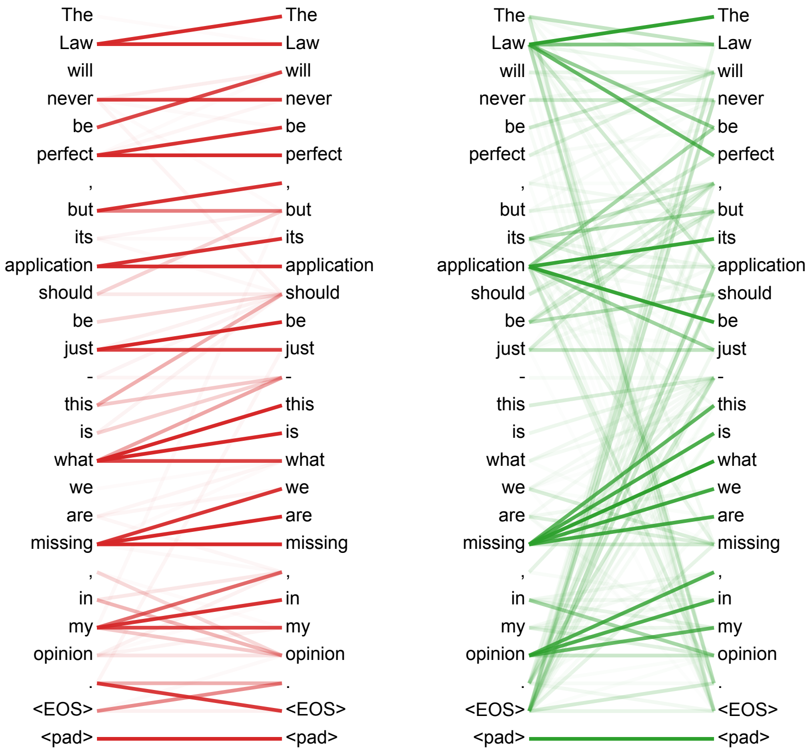
The standard metric in natural language processing is the Bilingual Evaluation Understudy Score (BLEU) score which aims at evaluating a generated sequence to a reference sentence. The BLEU score ranges between 0 (perfect mismatch) and 1 (perfect match).



(Vaswani et al., 2017)

Notes

On the left is a visualization of the attention as computed by one head of the layer 5 of the encoder
On the right the attention given by the word "its" for two different heads is on "law" and "application" which provides help for gender and grammatical issues.



(Vaswani et al., 2017)

Notes

Two other heads also in layer 5.

Standard transformers now combine differently the residual connection and the normalization (Wang et al., 2019).

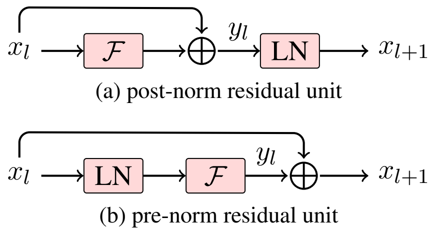


Figure 1: Examples of pre-norm residual unit and post-norm residual unit. $\mathcal{F}$ = sub-layer, and LN = layer normalization.

(Wang et al., 2019)

Transformer self-training and fine-tuning for NLP

The transformer networks were introduced for translation, and trained with a supervised procedure, from pairs of sentences.

However, as for word embeddings, they can be trained in an unsupervised manner, for auto-regression or as denoising auto-encoders, from very large data-sets, and fine-tuned on supervised tasks with small data-sets.

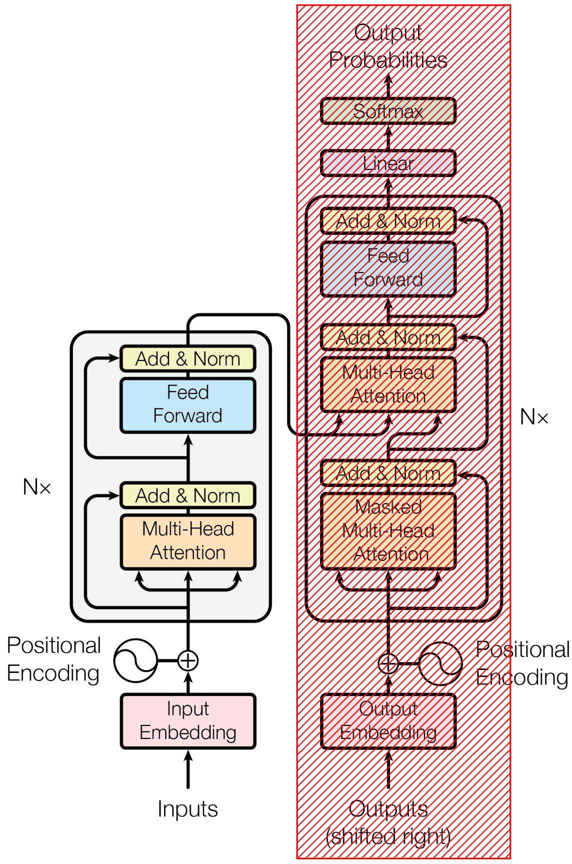
Notes

A transformer [pre-]trained in a unsupervised manner for the task of predicting a token: for auto-regression, the input is the sentence up to the token to predict, for mask language modeling, the input is a full sentence with some tokens replaced by a "mask" token. No ground truth is required for those tasks.
As for word embedding, training a transformer model like this allows to capture statistical structures in the text and provide an extremely good representation for more sophisticated tasks which can only be trained in a supervised manner with only small datasets available.

BERT (Bidirectional Encoder Representation from Transformers, Devlin et al., 2018) is an encoder of a transformer pre-trained with:

- Masked Language Model (MLM), that consists in predicting [15% of] words which have been replaced with a "MASK" token.
- Next Sentence Prediction (NSP), which consists in predicting if a certain sentence follows the current one.

It is then fine-tuned on multiple NLP tasks.



BERT (Devlin et al., 2018)

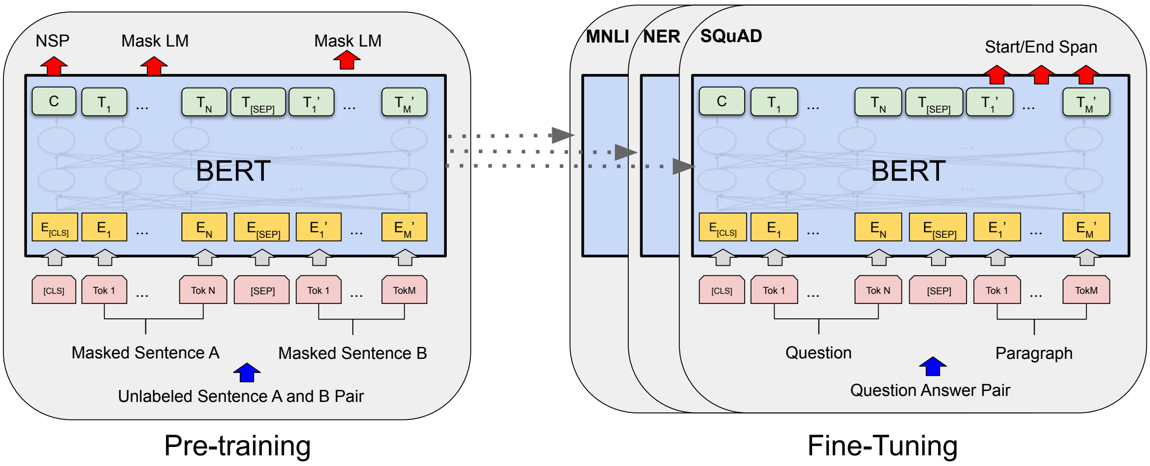
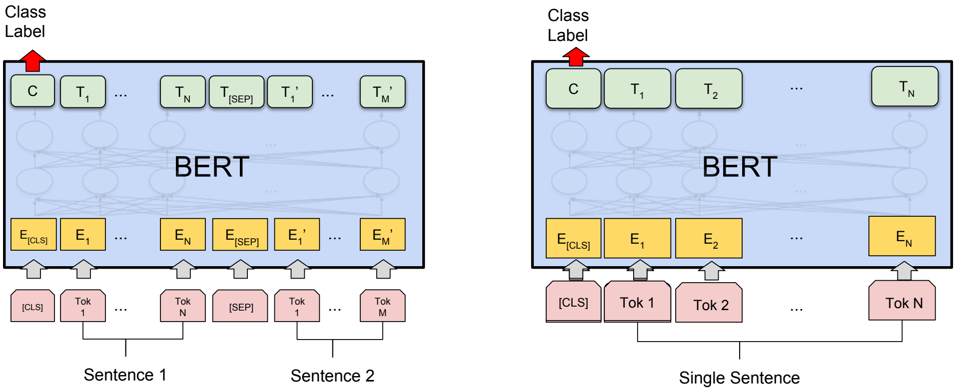


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

(Devlin et al., 2018)

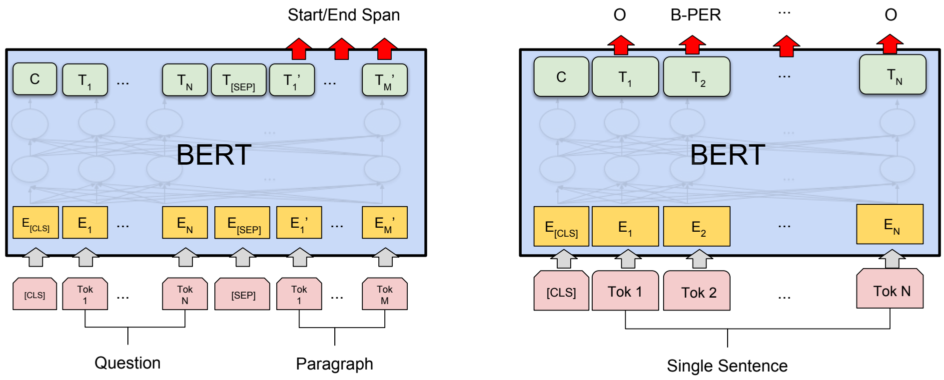
Notes

BERT is trained in an unsupervised manner to do two tasks at the same time: predicting missing words and predicting whether two sentences follow each other in the corpus.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

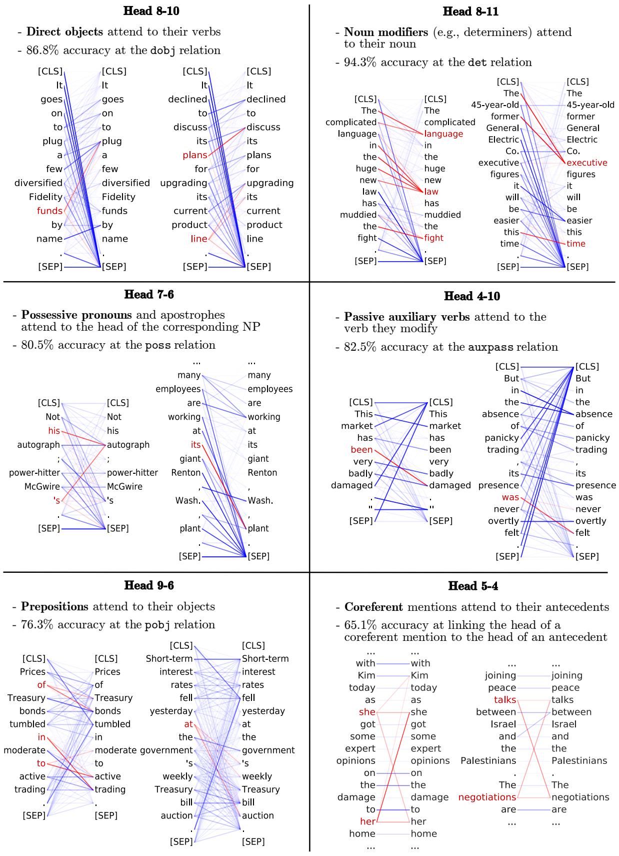
(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

(Devlin et al., 2018)



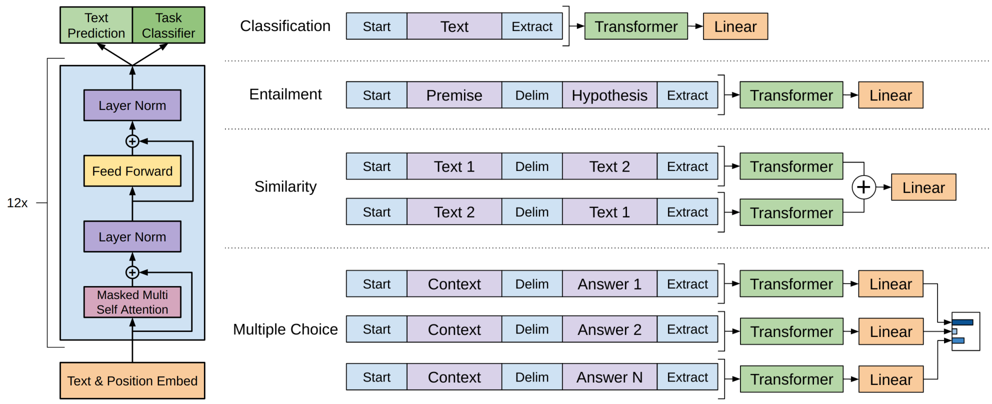
(Clark et al., 2019)

Notes

Once again, visualizing the attention matrices show that it connects a word with other words that help to get its meaning.

Large Language Models

GPT (Generative Pre-Training, Radford, 2018) is a decoder of a transformer trained for auto-regressive text generation.

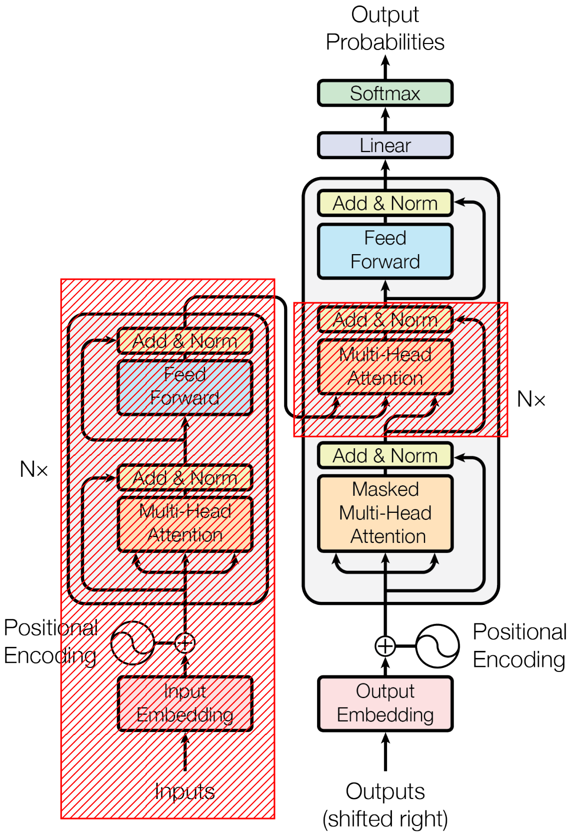


(Radford, 2018)

Notes

Note that GPT model is inherently causal, so only carries information forward, and consists of 12 modules as opposed to 6 for the original transformer.
The tasks GPT can be fine-tuned on are:

- Classification: for instance for sentiment analysis, when the input is a comment, and the task is to predict whether it is positive or negative.
- Entailment: given a premise and a hypothesis, the task is to predict whether the hypothesis is implied by the premise.
- Similarity: the task is to predict if two pieces of text have the same meaning.
- Multiple choice: the task is to predict the correct answer.



GPT (Radford, 2018)

"GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages. GPT-2 is trained with a simple objective: predict the next word, given all of the previous words within some text. The diversity of the dataset causes this simple goal to contain naturally occurring demonstrations of many tasks across diverse domains. GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data."

(Radford et al., 2019)

We can use HuggingFace's pre-trained models (https://huggingface.co/).

```
import torch

from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')
model.eval()

tokens = tokenizer.encode('Studying Deep-Learning is')

for k in range(100): # no more than 100 tokens
    outputs = model(torch.tensor([tokens])).logits
    next_token = torch.argmax(outputs[0, -1])
    tokens.append(next_token)
    if tokenizer.decode([next_token]) == '.': break

print(tokenizer.decode(tokens))
```

prints

```
Studying Deep-Learning is a great way to learn about the world around you.
```


Notes

HugginFace is a company that develops and distributes open-source implementations of language models. The transformers can be installed using the pip Python packets management system with `pip install transformers`

The piece of code in the slide loads a GPT-2 model and generates the end of a sentence given "Studying Deep-Learning is" as beginning.

`tokenizer` can take as input a sequence of strings to produce a sequence of token, or the opposite, take as input a sequence of tokens and produces the corresponding sequence of words.

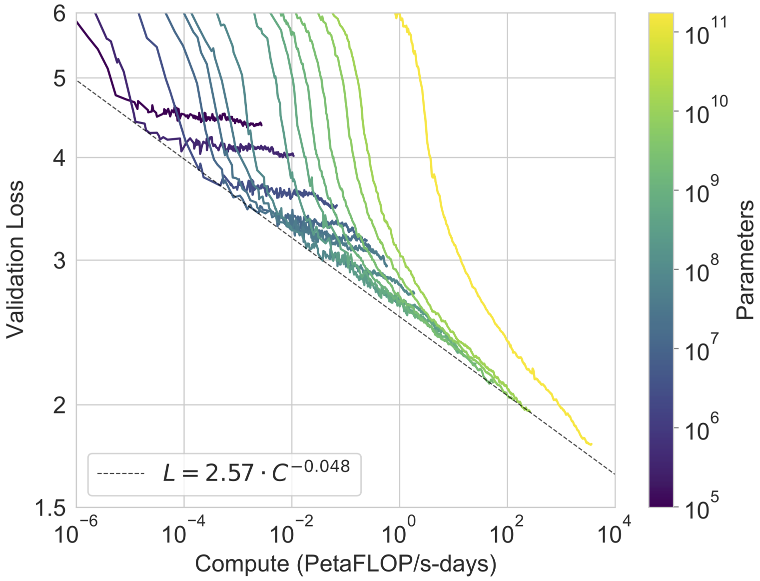
In this example, the generative procedure picks at each iteration the word with the maximum probability, but stochastic sampling could also be used.

The generated sentence could make sense in other situations (e.g. replacing "Deep-Learning" with another topic of studies), but is grammatically correct and consistent with "studying".

Large GPT have been shown to exhibit some "few shot learning" capabilities when they are properly "primed" (Brown et al., 2020).

For instance using Hugging Face's `gpt2` model with 120M parameters, we can get these sentence completions, where the generated parts are in bold:

I: I love apples, O: positive, I: music is my passion, O: positive, I: my job is boring, O: negative, I: frozen pizzas are awesome, O: **positive,**
I: I love apples, O: positive, I: music is my passion, O: positive, I: my job is boring, O: negative, I: frozen pizzas taste like cardboard, O: **negative,**
I: water boils at 100 degrees, O: physics, I: the square root of two is irrational, O: mathematics, I: the set of prime numbers is infinite, O: mathematics, I: gravity is proportional to the mass, O: **physics,**
I: water boils at 100 degrees, O: physics, I: the square root of two is irrational, O: mathematics, I: the set of prime numbers is infinite, O: mathematics, I: squares are rectangles, O: **mathematics,**



(Brown et al., 2020)

The GPT-3 model has 175B parameters and is trained on 300B tokens from various sources (Brown et al., 2020). The Pathways Language Model (PaLM) has 540B parameters and is trained on 780B tokens (Chowdhery et al., 2022).

Context →	`Q: What is 98 plus 45?` `A:`
Completion →	`143`

Figure G.44: Evaluation example for Arithmetic 2D+

Context →	`Q: What is 95 times 45?` `A:`
Completion →	`4275`

Figure G.45: Evaluation example for Arithmetic 2Dx

Context →	`Q: What is 509 minus 488?` `A:`
Completion →	`21`

Figure G.46: Evaluation example for Arithmetic 3D-

Setting	2D+	2D-	3D+	3D-	4D+	4D-	5D+	5D-	2Dx	1DC
GPT-3 Zero-shot	76.9	58.0	34.2	48.3	4.0	7.5	0.7	0.8	19.8	9.8
GPT-3 One-shot	99.6	86.4	65.5	78.7	14.0	14.0	3.5	3.8	27.4	14.3
GPT-3 Few-shot	100.0	98.9	80.4	94.2	25.5	26.8	9.3	9.9	29.2	21.3

(Brown et al., 2020)

```python
X = torch.randn(1, 3, 768, 768)
# Print all values of X higher than its median
print(X[X > X.median()])
```

```python
class Downsample(nn.Module):
    def __init__(self, factor, channel_out, drop_prob):
        super().__init__()
        self.downsample_conv = nn.Conv2d(64, 64, kernel_size=factor, stride=factor, groups=64)
        self.enlarge_conv = nn.Sequential(
            nn.Dropout2d(drop_prob),
            nn.Conv2d(64, channel_out, kernel_size=3, padding=1),
            norm_layer(channel_out),
            nn.ReLU(inplace=True),
        )

    def forward(self, x):
        x = self.downsample_conv(x)
        x = self.enlarge_conv(x)
        return x
```

OpenAI/Github's Copilot

Explaining a Joke

I will explain these jokes:
(1) The problem with kleptomaniacs is that they always take things literally.
Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't fully understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.
(2) Always borrow money from a pessimist. They'll never expect it back.
Explanation: Most people expect you to pay them back when you borrow money, however a pessimist is someone who always assumes the worst, so if you borrow money from them, they will expect that you won't pay them back anyways.

Input: I was supposed to start writing the paper at 5:00 PM. But then I started playing with this cool new language model for 10 minutes. 10 minutes later, it's suddenly 9:30 PM!
Model Output: This joke is about how time flies when you're having fun. The person was supposed to start writing a paper at 5:00 PM, but they started playing with a language model instead. They thought they were only playing with it for 10 minutes, but it was actually 4.5 hours.

Input: I think I understand why me and my partner are such a good match. I get excited when I smell food and become super alert when the doorbell rings. My girlfriend Shelley becomes ecstatic when she sees a person who she just saw 5 hours ago.
Model Output: This joke is about dogs. Dogs get excited when they smell food and when the doorbell rings. The girlfriend is excited to see someone she just saw 5 hours ago, which is a trait of dogs.

Input: I was going to fly to visit my family on April 6th. My mom said "Oh great, your step dad's poetry reading is that night!" So now I'm flying in on April 7th.
Model Output: The joke is that the speaker's mother is trying to get them to go to their step dad's poetry reading, but the speaker doesn't want to go, so they are changing their flight to the day after the poetry reading.

Input: I tried 10,000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.
Model Output: This joke is a pun. A neural network is a computer program that can learn from data. A "seed" is a number that is used to initialize a random number generator. A "good seed" is a number that produces a good random number generator. The phrase "no good deed goes unpunished" means that if you do something good, you will be punished for it.

Input: Reading "War and Peace" doesn't scare me. I once had to review the CL for my intern's whole project on his final day!
Model Output: 'War and Peace' is a very long book, so reading it is a daunting task. However, the speaker is saying they had to review the code for their intern's project on their final day, which is even more daunting.

Input: What's the difference between a zebra and an umbrella? One is a striped animal related to horses, another is a device you use to stop the rain from falling on you.
Model Output: This joke is an anti-joke. The joke is that the answer is obvious, and the joke is that you were expecting a funny answer.

(Chowdhery et al., 2022)

Inference Chaining

```
I will explain how to make logical inferences:
(1) John wakes to the sound of his children giggling outside the door. They bust in and say "Happy Father's Day!" with a plate full of food. 2022 has been a great year. Can we infer that John was born before the inauguration of Barack Obama?
Answer: We can infer that John is a father because of "his children". We can infer that the current year is 2022. Barack Obama was inaugurated in 2009, so it has been about 13 years since Obama was inaugurated. John's children are old enough to speak, so if John were born after Obama's inauguration, he would have been younger than 10 when he had children. So, we can infer that John was born before Obama's inauguration. The answer is "yes".
(2) Sam sees a piano in the subway stations. Laughing, his friends prod him to play, saying "Remember how amazing your rendition of Chopsticks was at the Christmas party?" Can we infer that John majored in piano at Juilliard?
Answer: Juilliard is a top school for music, so someone who majored in piano at Juilliard would be an expert pianist. "Chopsticks" is a very simple song to play on the piano. His friends were therefore likely being sarcastic when they told him to play, which would imply that Sam was not an expert piano player. So it is unlikely that Sam majored in piano at Juilliard. The answer is "no".
```

Input: When I found out my grandma was in the hospital I felt a particular color. When someone cut me off in traffic I felt a different color. What is the most likely color I would see if I combined these two colors?
Model Output: The color that I felt when I found out my grandma was in the hospital was likely a sad color, like blue. The color that I felt when someone cut me off in traffic was likely an angry color, like red. If I combined these two colors, I would likely see purple. The answer is "purple".

Input: Michael is at that really famous museum in France looking at its most famous painting. However, the artist who made this painting just makes Michael think of his favorite cartoon character from his childhood. What was the country of origin of the thing that the cartoon character usually holds in his hand?
Model Output: The most famous painting in the Louvre is the Mona Lisa. The artist who made the Mona Lisa is Leonardo da Vinci. Leonardo da Vinci is also the name of the main character in the cartoon Teenage Mutant Ninja Turtles. Leonardo da Vinci is from Italy. The thing that Leonardo da Vinci usually holds in his hand is a katana. The country of origin of the katana is Japan. The answer is "Japan".

Input: Guido was writing a Python program. He keeps getting confused by variable scope and looking it up on StackOverflow. Is it possible that the Guido in the prompt refers to Guido van Rossum?
Model Output: Guido van Rossum is the creator of Python. He would not need to look up variable scope on StackOverflow. The answer is "no".

Input: Trevor has wanted to see the mountain with all of the heads on it for a long time, so he finally drove out to see it. What is the capital of the state that is directly east of the state that Trevor is currently in?
Model Output: The mountain with all of the heads on it is Mount Rushmore. Mount Rushmore is in South Dakota. The state directly east of South Dakota is Minnesota. The capital of Minnesota is St. Paul. The answer is "St. Paul".

Input: Jennifer looked out her window and sees a really cool cloud below her. She unbuckles her seatbelt and heads to the bathroom. Is Jennifer probably traveling more than 300 miles per hour relative to the earth?
Model Output: 300 miles per hour is about 480 km/h. This is about the speed of a commercial airplane. Clouds are usually below airplanes, so Jennifer is probably on an airplane. The answer is "yes".

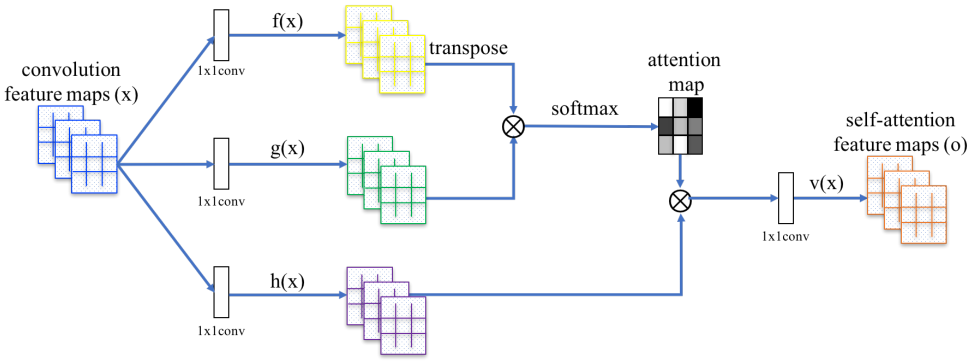
(Chowdhery et al., 2022)

Summary:

- The original Transformer (Vaswani et al., 2017) combines an encoder composed of blocks each made of a self-attention layer, and a per-token MLP, and a decoder composed of blocks each made of a causal self-attention layer, a cross attention layer, and a per-token MLP.
- BERT (Devlin et al., 2018) has the structure of the Transformer's encoder.
- GPT (Radford, 2018; Radford et al., 2019) has the structure of the Transformer's decoder without cross-attention.
- A model can be self-trained to predict masked words (BERT), or for auto-regression (GPT), and fine-tuned on downstream tasks.
- Special tokens can separate parts of inputs (e.g. question / answer) or indicate the output token used for prediction (e.g. sentiment analysis).
- These models scale extremely well to 100s of billions of tokens and parameters (Kaplan et al., 2020)
- Auto-regressive language models can be primed to solve with remarkable accuracy zero-shot learning tasks (Brown et al., 2020; Chowdhery et al., 2022).

Vision Transformers

As in NLP, attention mechanisms in vision allow models to leverage long-term dependencies that would require many convolutional layers, e.g. for Self-Attention Generative Adversarial Networks (SAGANs):



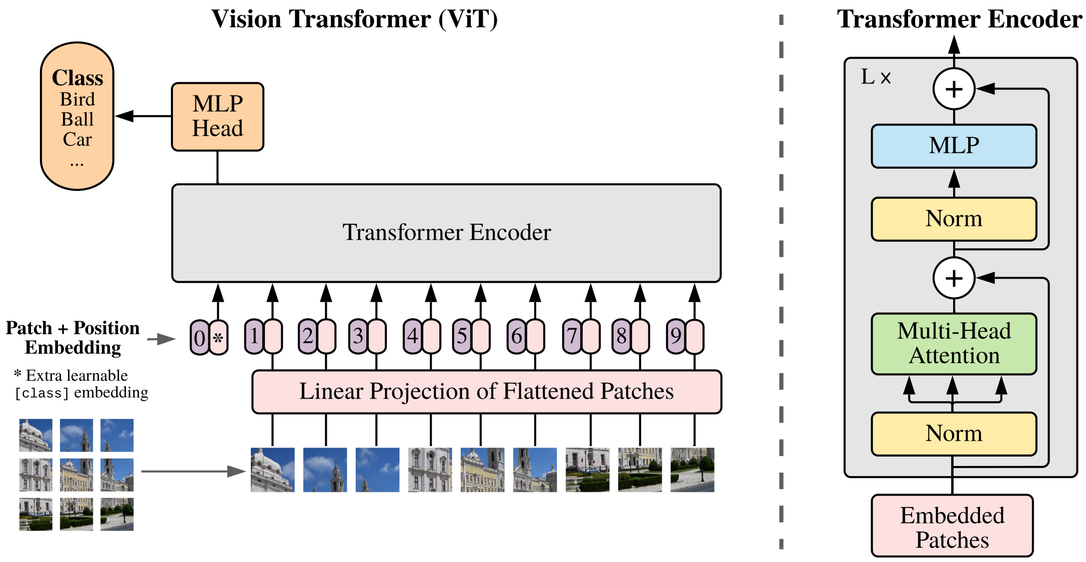
> "The self-attention module is complementary to convolutions and helps with modeling long range, multi-level dependencies across image regions. Armed with self-attention, the generator can draw images in which **fine details at every location are carefully coordinated with fine details in distant portions of the image.**"

(Zhang et al., 2018)

The Vision Transformer (ViT, Dosovitskiy et al. 2020) is a very simple architecture for image classification.

> "Inspired by the Transformer scaling successes in NLP, we experiment with applying a standard Transformer directly to images, with the fewest possible modifications. To do so, we split an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer. Image patches are treated the same way as tokens (words) in an NLP application. We train the model on image classification in supervised fashion."

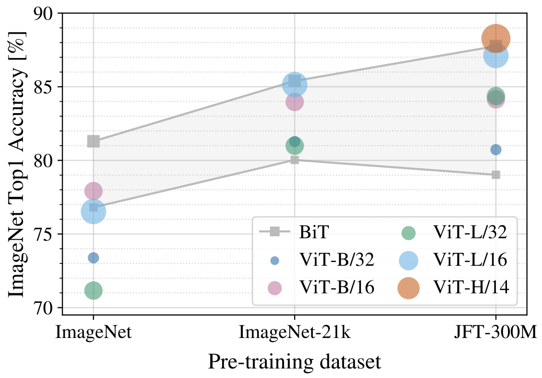
(Dosovitskiy et al., 2020)

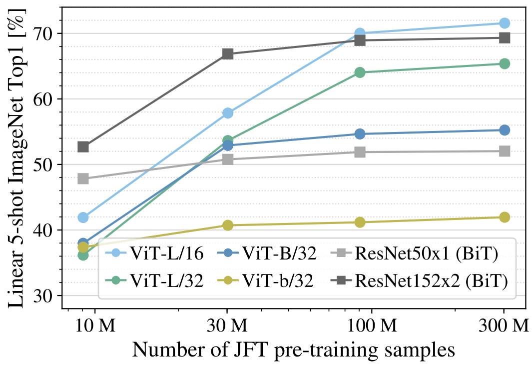


(Dosovitskiy et al., 2020)

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

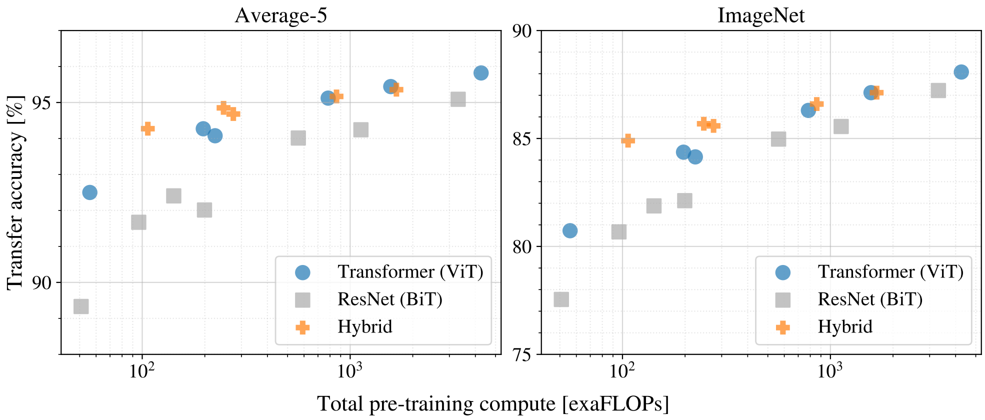




(Dosovitskiy et al., 2020)

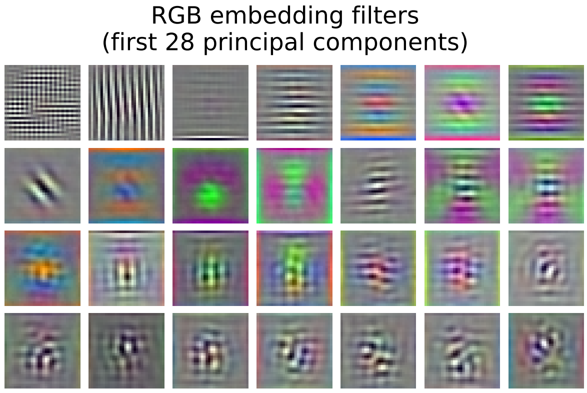
Notes

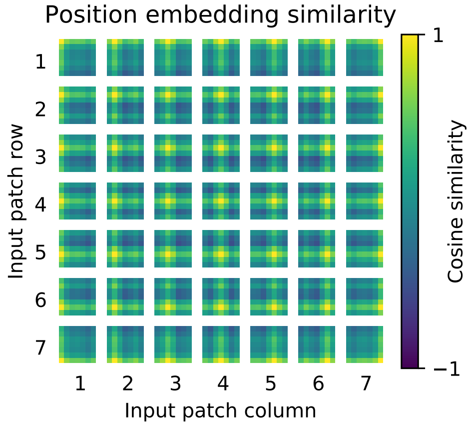
With large pre-training, Vision Transformers (ViT) performs better than Resnets adapted to "Big Transfer" (BiT).



1 exaFLOPs $\simeq$ 1h RTX 3090

(Dosovitskiy et al., 2020)

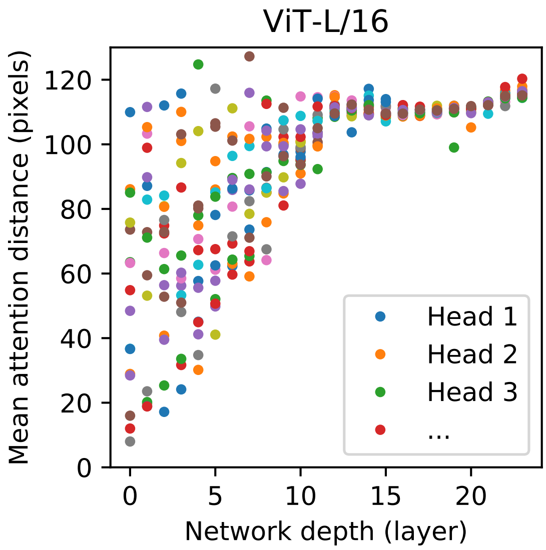




(Dosovitskiy et al., 2020)

Notes

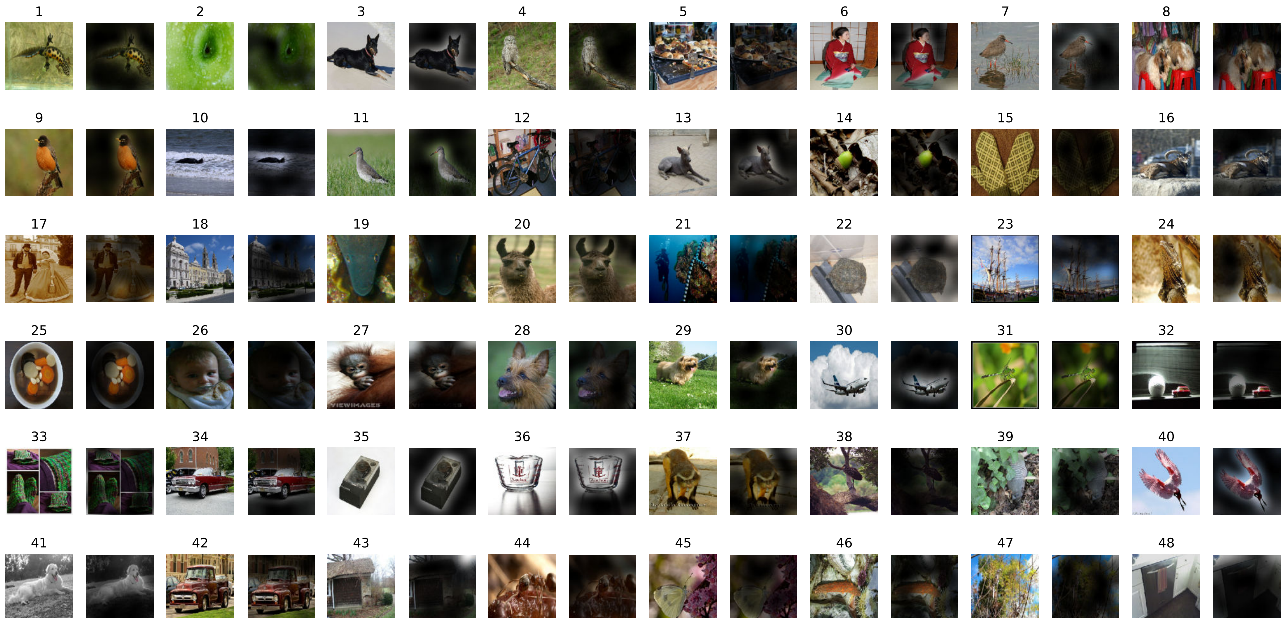
The left figure shows filters learned to encode the patches. They exhibit the expected structure of an image linear basis.
The right image shows for every one of the 7×7 patch positions the similarity of its positional encoding with the positional encodings of all other patches. The learned encodings reflect the rows and columns, and more generally the 2d structure of the image lattice.



(Dosovitskiy et al., 2020)

Notes

This scatter plot shows for every layer and every head the average distance in pixels between a patch and the patches it attends too.
Early layers have very diverse "attention distances", while the last one tend to have long distance attention only.

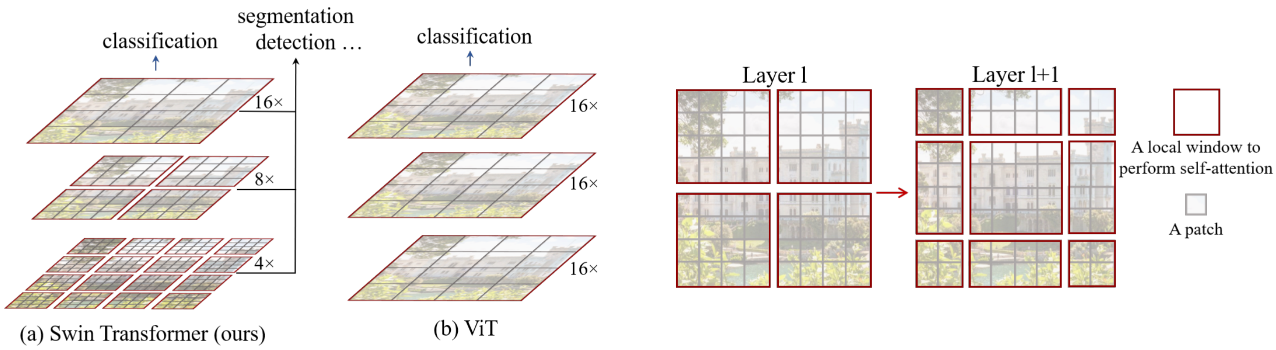


(Dosovitskiy et al., 2020)

Notes

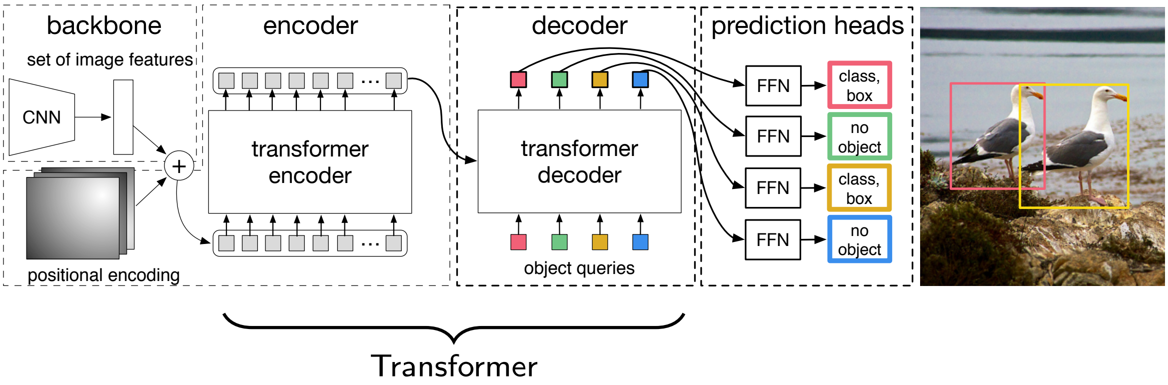
This picture is obtained by looking at the attention of the output token and then going backward through layers, multiplying by the attention averaged across heads for each.

The Swin Transformer (Liu et al., 2021) improves the ViT architectures through the use of hierarchical representation with local attention in shifting windows.



(Liu et al., 2021)

The DETR algorithm (Carion et al., 2020) combines a CNN and a transformer for object detection.



$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

(Carion et al., 2020)

Notes

A CNN converts the original image into a tensor of $1/32$ the size, which is flatten as a sequence of tokens.

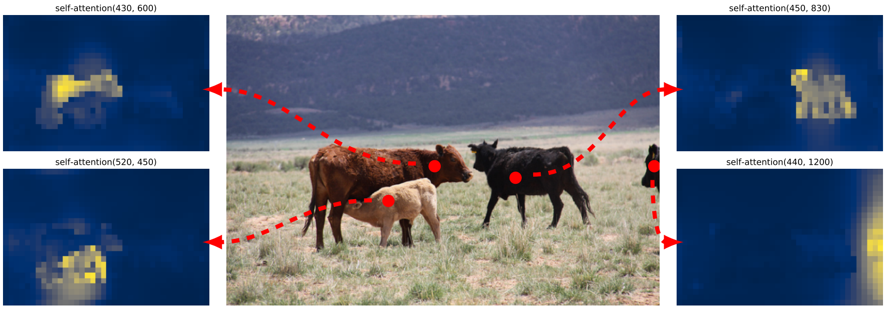
The sequence if then fed into a Transformer with a non-causal feed-forward decoder (as opposed to the standard auto-regressive one). The maximum number of detections is specified by the number initial "object queries" given to the decoder.

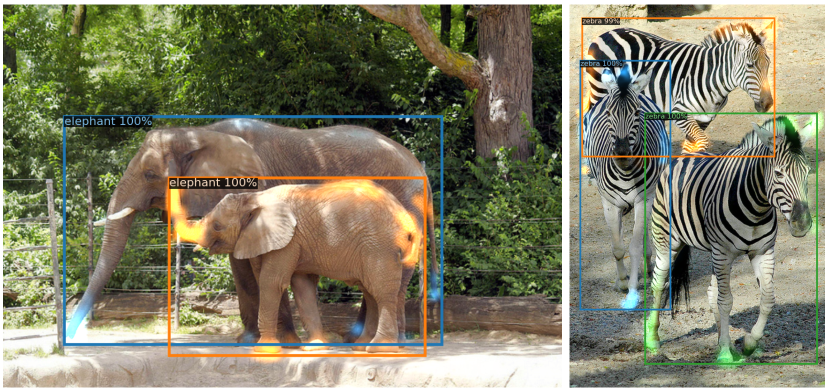
The final read-out is done with a per-token MLP that maps the internal feature dimension to the box coordinates, and a linear layer that maps the internal feature representation to the class (+ "background") logits.

The loss is computed for an optimal matching σ computed with the Hungarian algorithm between the ground truth and the N predicted detections.

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

(Carion et al., 2020)


self-attention(430, 600)
self-attention(450, 830)
self-attention(520, 450)
self-attention(440, 1200)


elephant 100%
elephant 100%
zebra 99%
zebra 100%

References

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. **Language models are few-shot learners**. CoRR, abs/2005.14165, 2020.

N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. **End-to-end object detection with transformers**. CoRR, abs/2005.12872, 2020.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. Dai, T. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. **PaLM: Scaling language modeling with pathways**. CoRR, abs/2204.02311, 2022.

K. Clark, U. Khandelwal, O. Levy, and C. Manning. **What does BERT look at? An analysis of BERT's attention**. CoRR, abs/1906.04341, 2019.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. **BERT: Pre-training of deep bidirectional transformers for language understanding**. CoRR, abs/1810.04805, 2018.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. **An image is worth 16x16 words: Transformers for image recognition at scale**. CoRR, abs/2010.11929, 2020.

J. Kaplan, S. McCandlish, T. Henighan, T. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. **Scaling laws for neural language models**. CoRR, abs/2001.08361, 2020.

Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. **Swin transformer: Hierarchical vision transformer using shifted windows**. CoRR, abs/2103.14030, 2021.

A. Radford. **Improving language understanding with unsupervised learning**. web, June 2018. https://openai.com/blog/language-unsupervised/.

A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever. **Better language models and their implications**. web, February 2019. https://blog.openai.com/better-language-models/.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. **Attention is all you need**. CoRR, abs/1706.03762, 2017.

Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. Wong, and L. Chao. **Learning deep transformer models for machine translation**. CoRR, abs/1906.01787, 2019.

H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. **Self-attention generative adversarial networks**. CoRR, abs/1805.08318, 2018.