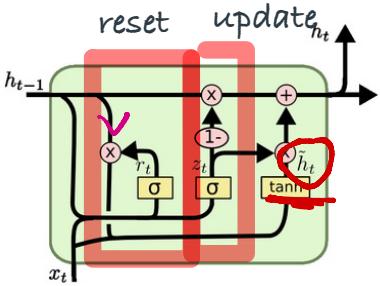
Bert + Vision Transformers

Agenda

1. GRU
2. BERT
3. BART
4. GPT
5. Vision Transformers

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Reset Gate: Determine how to combine new input with the past hidden State, i.e. ht-1 with xt

ht(1, 512) where 512 is the hidden state dimension
xt(1, 512) where 512 is embedding dimension of xt

hidden-st

ht-1 + xt

(1024, 512)

In Reset gate $[h_{t-1}; x_t]$ (1,512) (1,512) $\Rightarrow (1, 1024) \cdot W_r$

$(1,024) \times (1024, 512)$

$rt = \sigma(1, 512)$

↓

Series of on/off values

$rt \otimes h_{t-1}$

(1,512) (1,1024)

$[rt \otimes h_{t-1}; z_t] \odot W$ (1024, 512) $[---]$

↓

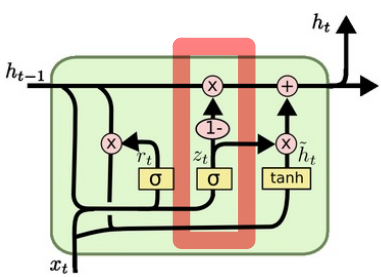
bw 0/1

candidate hidden state = $\tanh([1, 512])$

$= \tanh([r_t \circledast h_{t-1}, x_t] \circ \omega)$

determine how to combine h_{t-1} & x_t

Update gate: Decide how much of past information to keep and how much of new information to add to hidden-state



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

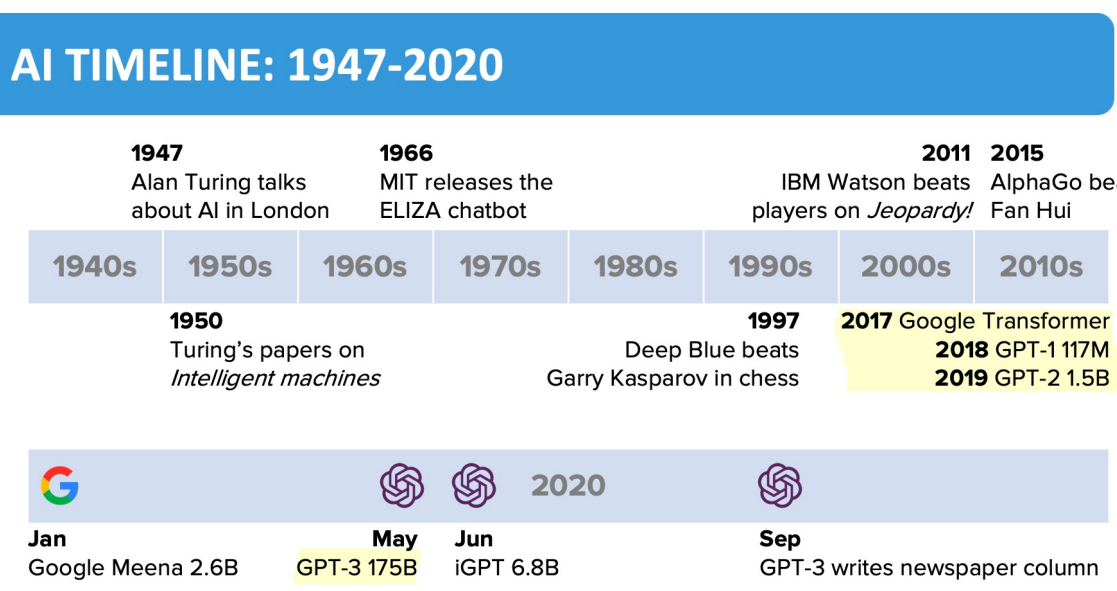
$[h_{t-1} : x_t] \longrightarrow (1, 1024) \cdot \omega_2$ $(1024, 512)$

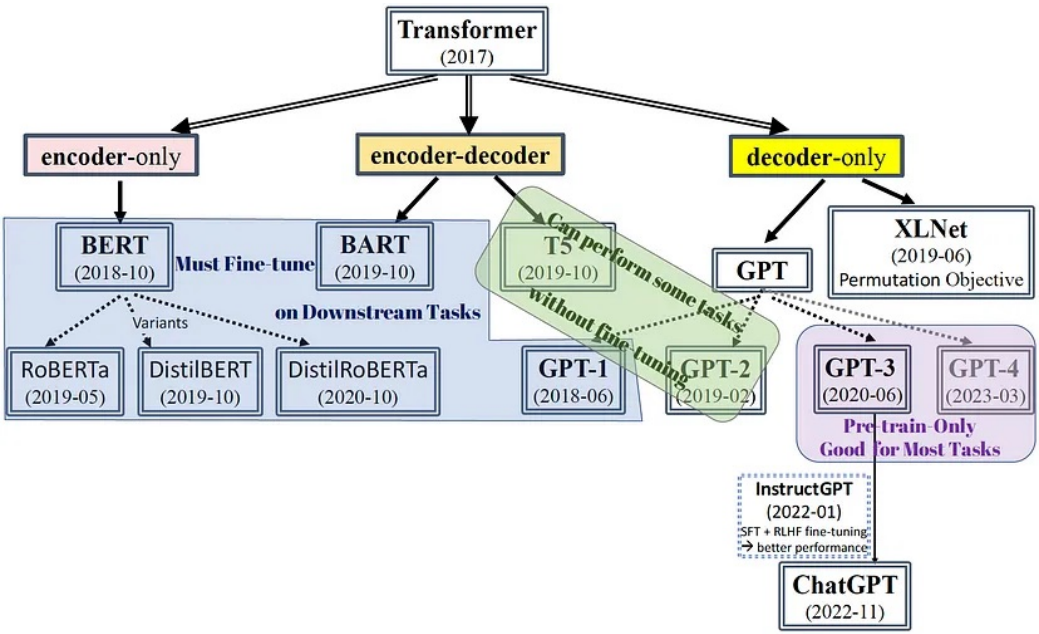
$z_t = 6(1, 512)$ blood

$$h_t = z_t \odot \tilde{h}_t + (1 - z_t) \odot h_{t-1}$$

high

High value of zt means low value for (1-zt) which means forget more about past as (1 - zt) will be small

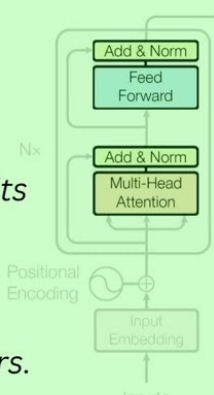


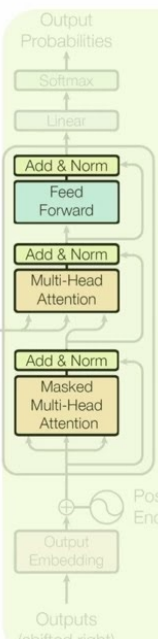


BERT

Google

use transfer learning to ***continue learning*** *from its existing data when adding user-specific tasks and layers.*





GPT

OpenAI

decodes from its massive pre-learned embeddings to present output that matches user prompts. It

Figure 1: The Transformer - model architecture.

BERT is a stack of Encoders. We train BERT in two steps, "**pre-training**" and "**fine-tuning**".

During **pre-training**, the model is trained on a large dataset in an *unsupervised learning* task where the model is trained on an unlabelled dataset like the data from Wikipedia.

During **fine-tuning** the model is trained for downstream tasks like *classification, text generation, language translation, question-answering, and so forth.*

In the **pre-training** mode, BERT is trained on 2 specific tasks: **Masked Language Model** and **Next-sentence similarity prediction**.

First, let's talk about pre-training, how do you pre-train the stack of encoders

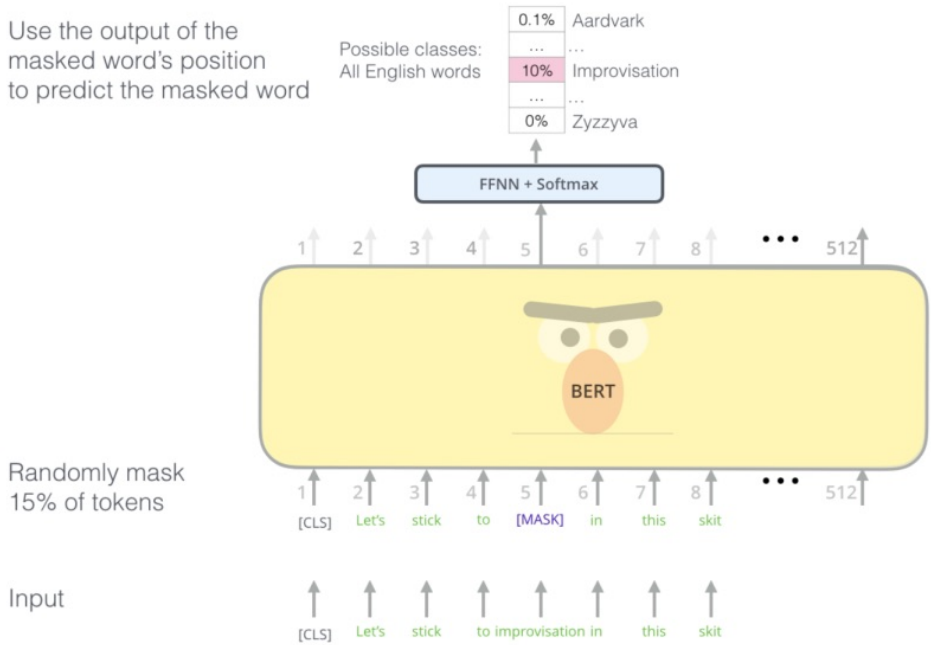
1. Masked Language Modelling
2. Next Sentence Similarity Prediction

The pre-training part, of BERT which includes above methods are UNSUPERVISED learning

We will discuss this each of above in elaborate detail

Masked Language Modelling

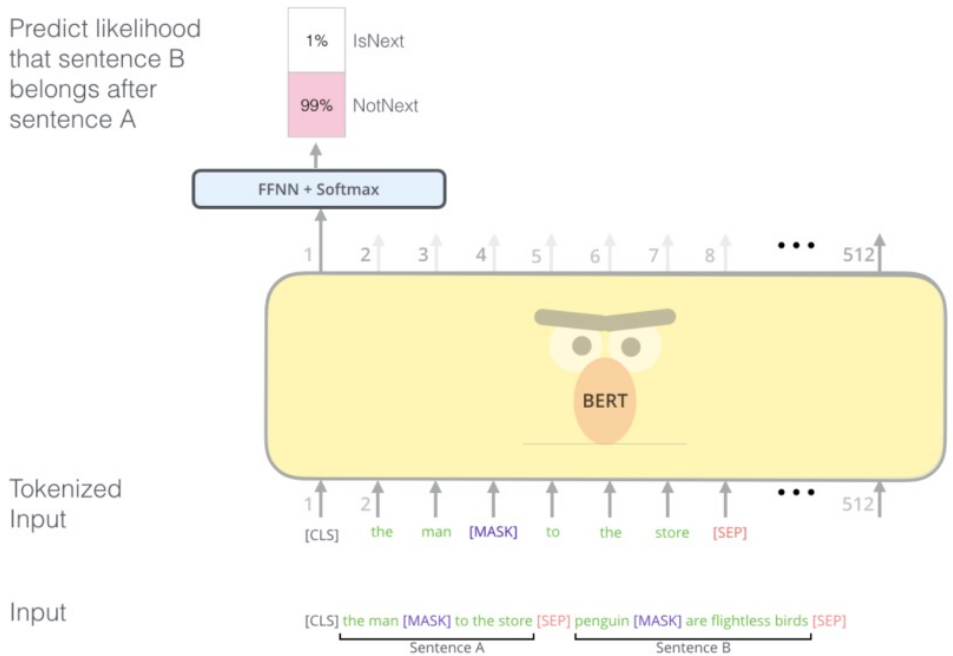
I could not visit the bank in the afternoon, as employees were on leave



Masked Language modelling uses - masked input like "the cat [MASK] the mouse" and BERT needs to predict ATE.

Basically I can use entire text corpus of internet to train this

Next Sentence Similarity Prediction



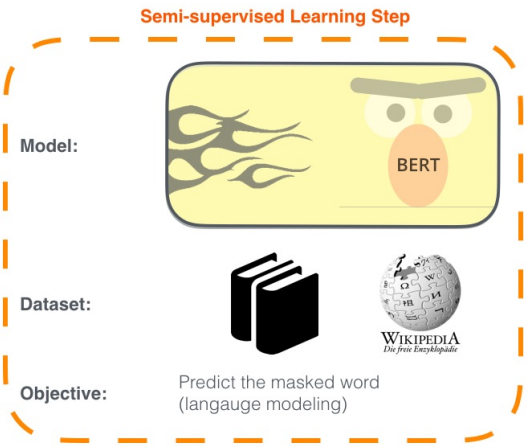
CLS Cat loves milk. Cat hates dog.

Code From Scratch: https://colab.research.google.com/drive/1V1n57ImzTLroU1K0cRX51x9XyQMe8aPD?usp=sharing

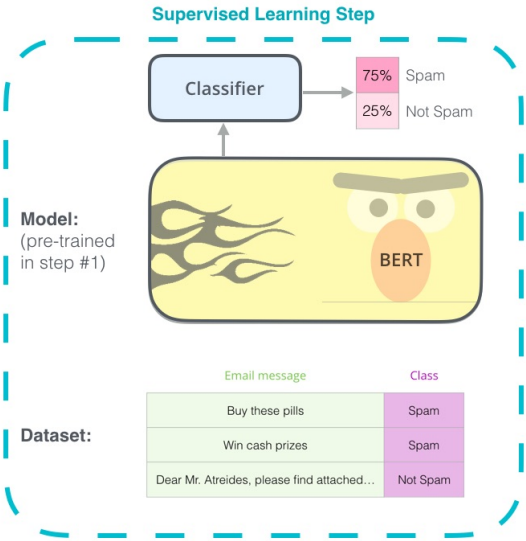
Link to Beautiful Article: https://neptune.ai/blog/bert-and-the-transformer-architecture

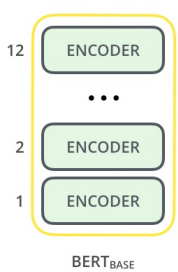
1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

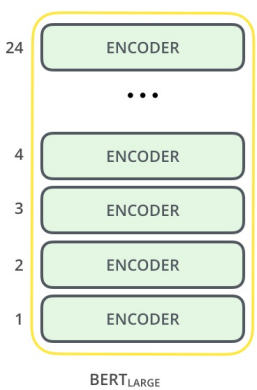
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

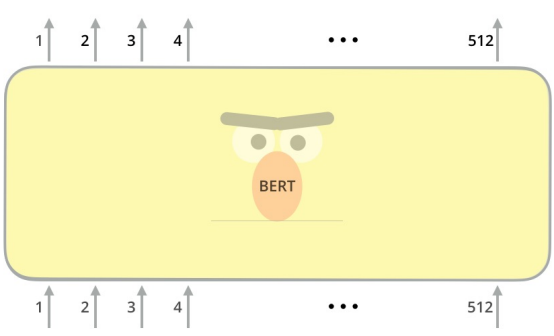


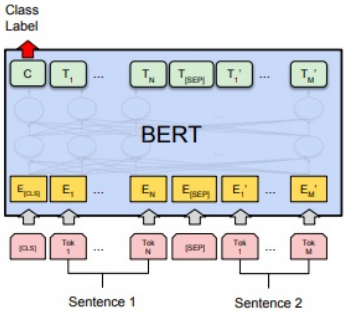
2 - Supervised training on a specific task with a labeled dataset.



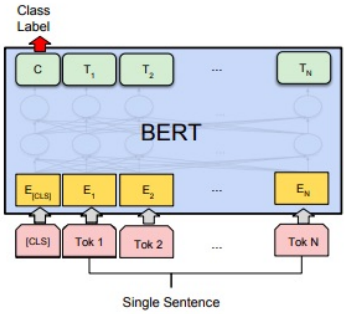




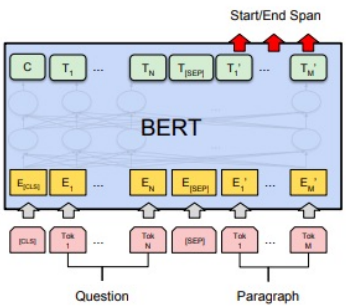




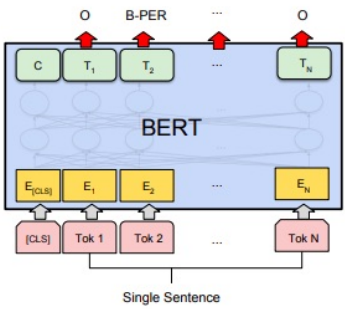
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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

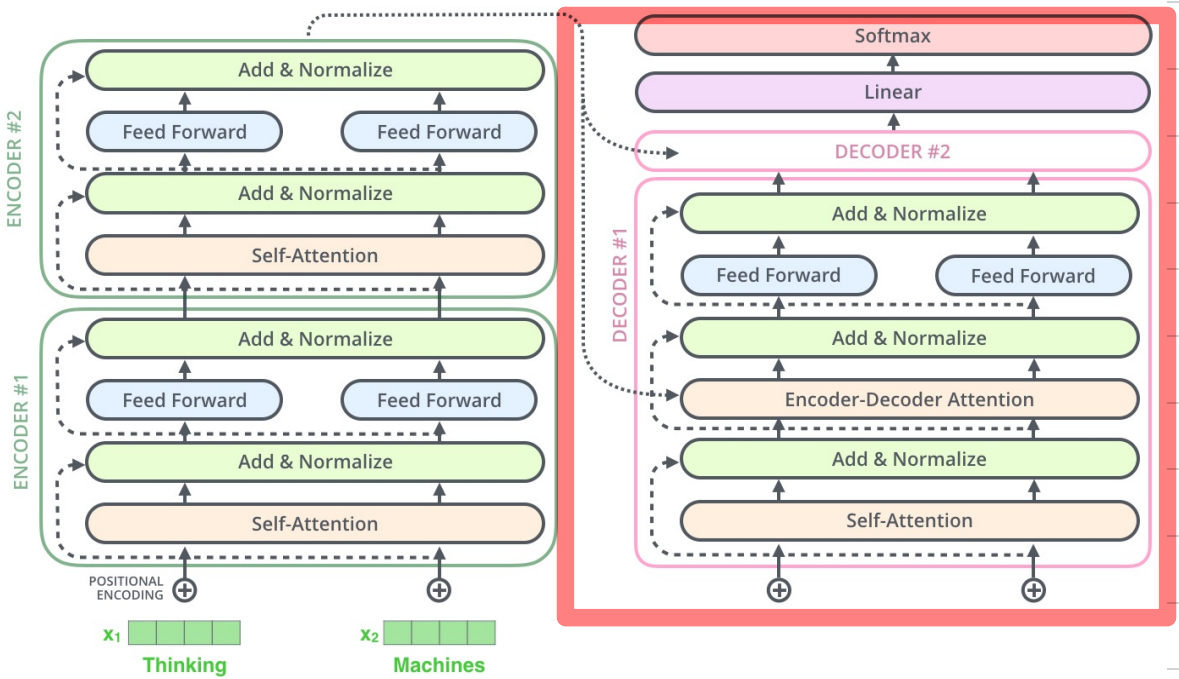
Rarely vs

BART (Bidirectional and Auto-Regressive Transformers)

1. Token masking - a random subset of the input is replaced with [MASK] token, like in BERT
2. Token deletion - random tokens are deleted from the input. The model must decide which positions are missing (as the tokens are simply deleted and not replaced with anything else)
3. Text infilling - a number of text spans (length may vary) are each replaced with a single [MASK] token.
4. Sentence Permutation - the input is split based on period (.) and the sentences are shuffled.
5. Document Rotation - a token is chosen at random, and the sequence is rotated so that it starts with the chosen token.

Corruption Scheme	Original Text	Corrupted Text	Explanation
Token Masking	ABC.DE.	A_C._E.	Both B and D are masked with a single mask token for each.
Token Deletion	ABC.DE.	A.C.E.	Both B and D are deleted (and not replaced).
Text Infilling	ABC.DE.	A_.D_E.	The span BC is replaced with a single mask token. A 0 length span is inserted between D and E.
Sentence Permutation	ABC.DE.	DE.ABC.	Split into sentences at periods (.) and shuffled.
Document Rotation	ABC.DE.	C.DE.AB	The sequence is rotated around C.

GPT-s

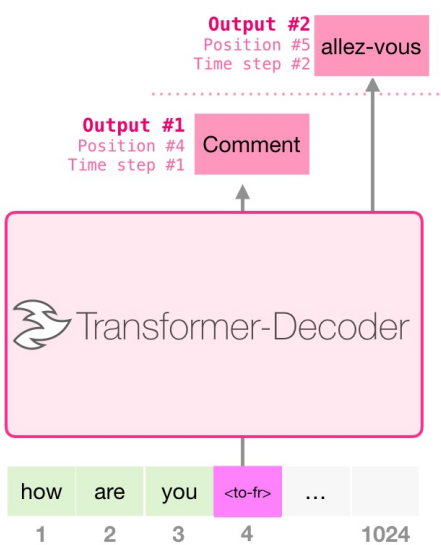


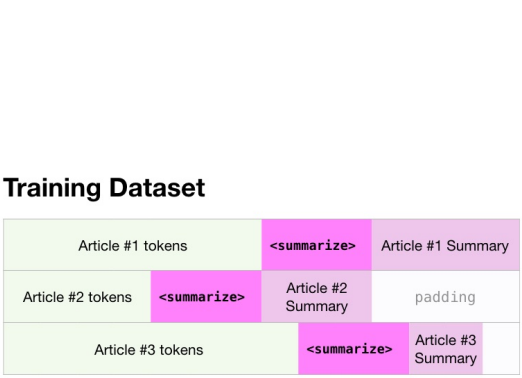
let 7
2nd
3rd

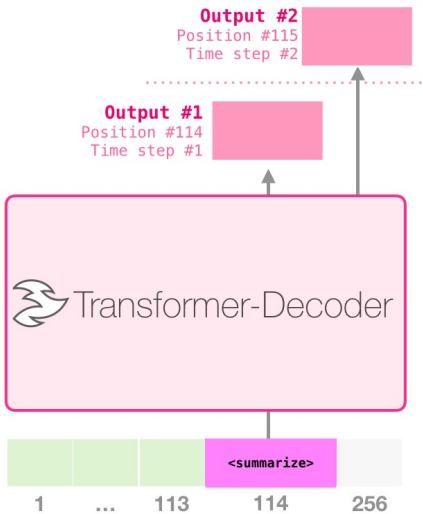
out1 → J
out1 → suis
out1 → etudia

Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				







very detailed article for GPT-2 https://jalammar.github.io/illustrated-gpt2/

GPT:

Generative Pre-trained Transformers are decoder-only transformer models. These models are pre-trained on massive amounts of data, such as books, web-pages, to generate contextually relevant and semantically coherent language.

GPT1 (117M parameters) was released in 2018 and was trained on Common CrawlLinks to an external site. and the BookCorpusLinks to an external site..

While GPT was great, it was prone to generating repetitive text and also failed to reason over multiple turns of dialogue.

GPT2:

GPT2 was released in 2019 and had 1.5B parameters. It was trained on much larger and more diverse dataset. It also struggled with tasks that required complex reasoning and understanding of context. It failed to maintain context and coherence over longer passages.

GPT3:

GPT3 was released in 2020 and has 175B parameters! It was trained on an even larger and diverse range of data sources, including BookCorpus, Common Crawl, and Wikipedia, among other things. The datasets comprise nearly a trillion words, allowing GPT3 to generate sophisticated responses on a wide range of NLP tasks, even without providing any prior example data.

One of the main improvements of GPT3 over its previous models is its ability to generate coherent text, write computer code, and even create art (SVG, etc.). Unlike the previous models, GPT3 understands the context of a given text and can generate appropriate responses which led to applications like content creation, chatbots, and language translation.

GPT3 had its flaws. The model can return biased, inaccurate, and inappropriate responses. This issue arises because it is trained on massive amounts of texts that possibly contain biased and inaccurate information.

GPT4:

GPT4 was released on March 14, 2023. It's a significant step up from GPT3. Rumors claim that GPT4 has 1.8 trillion parameters, which was estimated by the speed it was running. Some other interestingLinks to an external site. facts/rumors: GPT4

has 1.8 trillion parameters

has 120 layers (10 times more than GPT3)

Has 16 Mixtures of Expert (MoE)

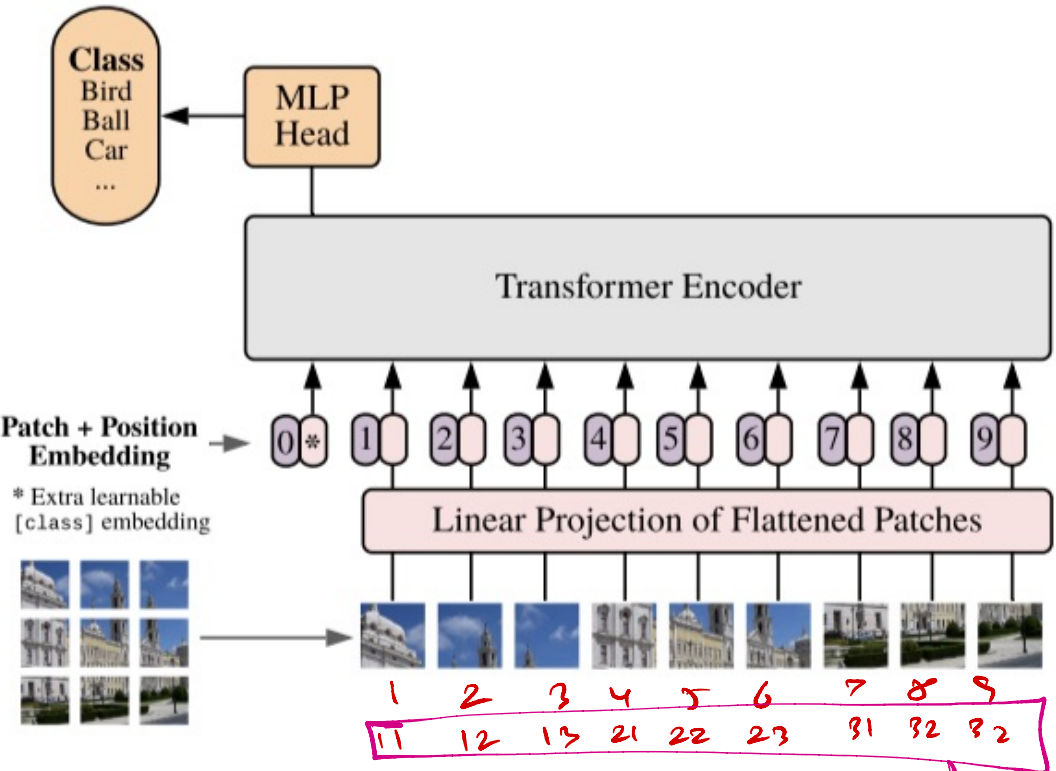
Was trained on 13 trillion tokens

Was trained on GPT3 data + twitter, youtube, reddit, and more books

Took $63M to train

Andrej Karpathy Course

Vision Transformers



Not good

Input Image → 224 × 224 × 3

224 / 14 = 16

14 × 14 = 196

16 × 16 × 3

196 Patch → 16 × 16 × 3

16 × 16 × 3

14

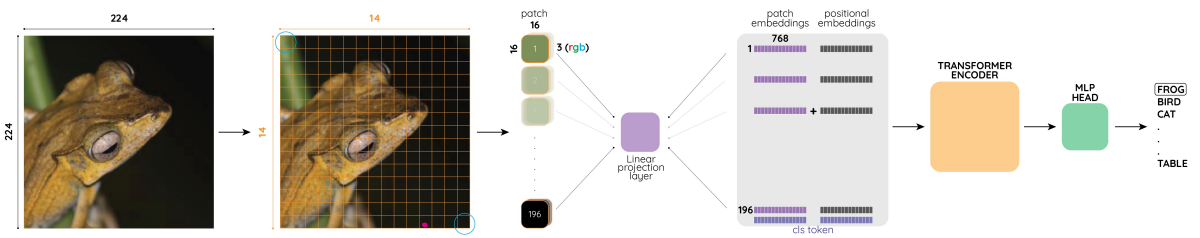
14

227

⇒ 196 × 768 + PE

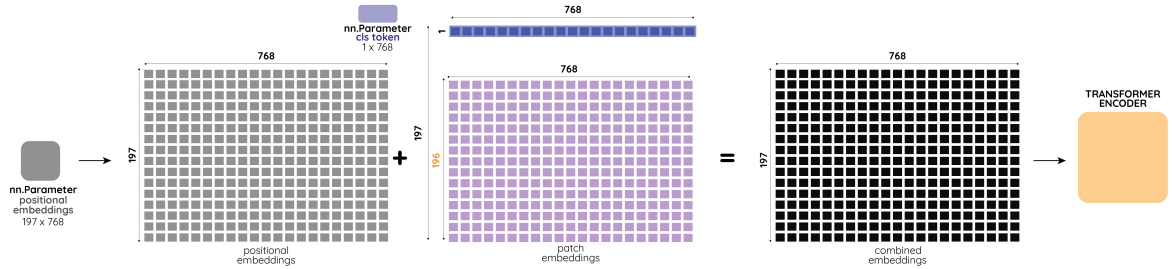
The overall architecture can be described easily in five simple steps:

1. Split an input image into patches
2. Get linear embeddings (representations) from each patch referred to as Patch Embeddings
3. Add positional embeddings and a [CLS] token to each of the Patch Embeddings.
4. Pass through a Transformer Encoder and get the output values for each of the [CLS] tokens.
5. Pass the representations of [CLS] tokens through an MLP Head to get final class predictions.

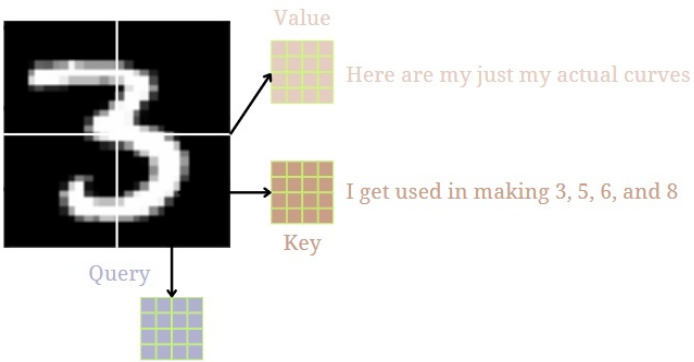


As an overall method, from the paper

We split an image into fixed-size patches, linearly embed each of them, add positional embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence.



Query Key: Value



Stride = 32

$32 \times 32 \times 3 \times 3072 \rightarrow 32 \times 32 \times 3$

$32 \times 32 \times 3$

$224 \times 224 \times 3$ ⊙ $16 \times 16 \times 3$

$16 \times 16 \times 3 \times 768$ no of Kernels

Stride of 16

output shape Convolution $14 \times 14 \times 768$

$$\frac{n_f + 2P - K}{\text{Stride}} + 1$$

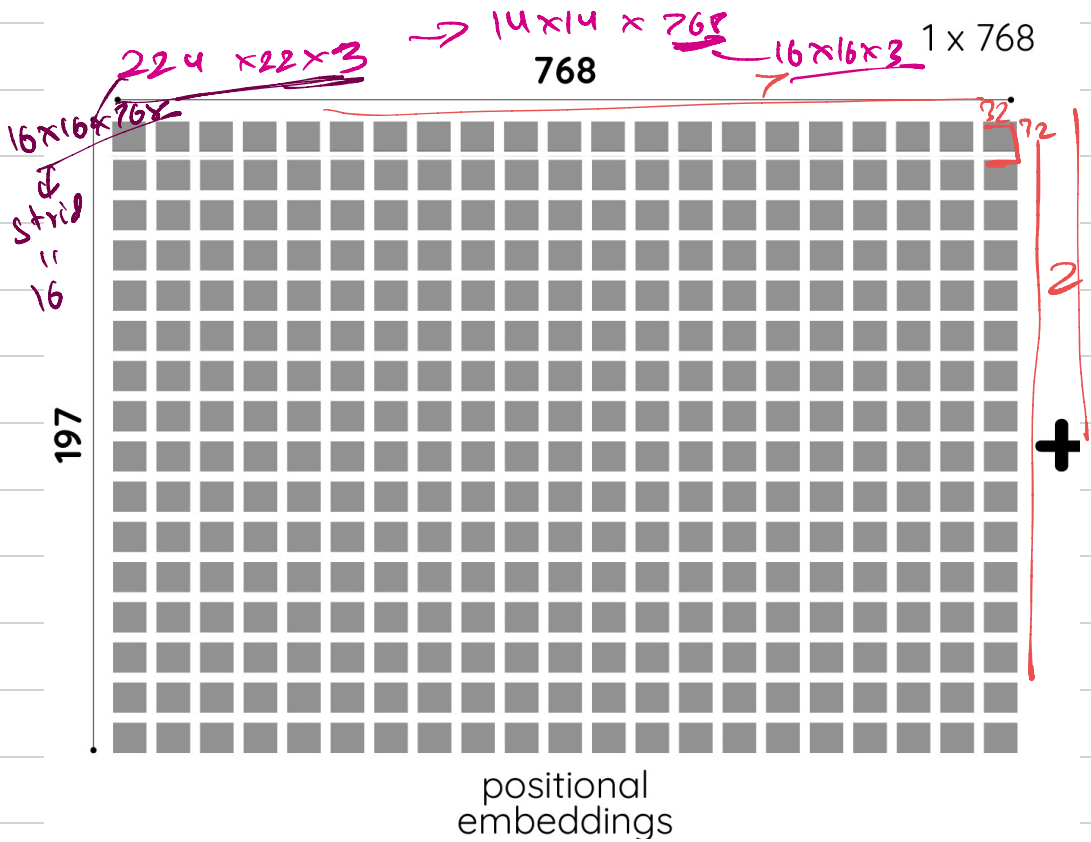
[(W−K+2P)/S]+1

$$\frac{224 + 2P - 16}{16} + 1$$

P → 0

$$\frac{208 + 2\times0 - 16}{16} + 1$$

$$\Rightarrow \frac{13}{1} + 1 = 14$$



$I_{mage} \rightarrow 224 \times 224 \times 3 \rightarrow$ 49 $\rightarrow 32 \times 32 \times 3$ (1D)

$49 \times (32 \times 32 \times 3) = 3072$ — 01 → Flattening out

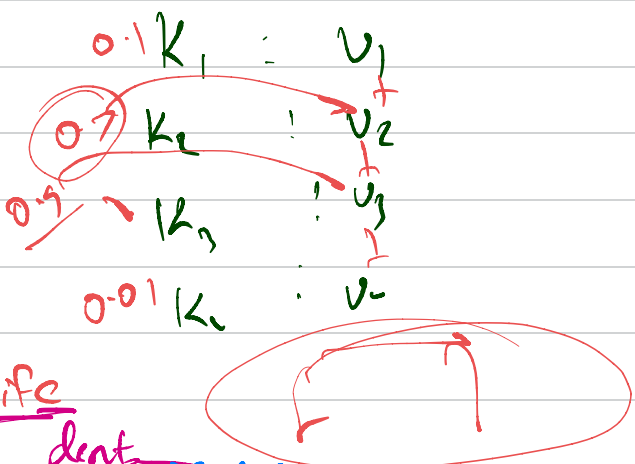
$224 \times 224 \times 3$ ⊙ $32 \times 32 \times 3 \times 3072$ ← Kernel

$\rightarrow$ $7 \times 7 \times 3072$

$\Rightarrow$ 49×3072

$\frac{224 - 2 \times 0 + 32}{32} + 1 \Rightarrow 6 + 1 = 7$

$Q_1 =$



WK

K

Best day of your life

dent

WV

V

Scdd day of y li

dinf gnP