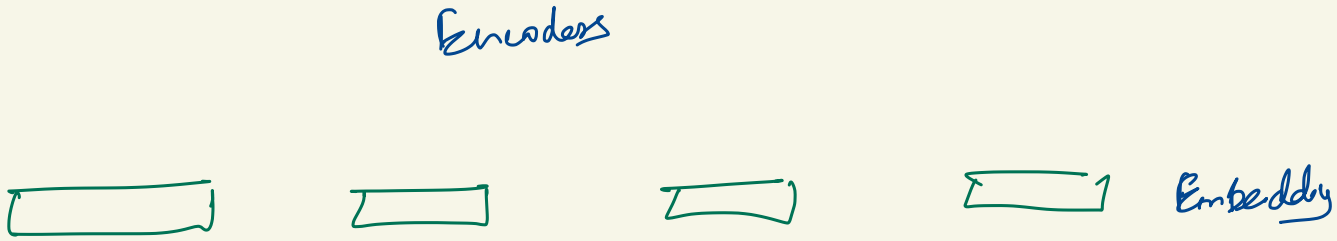
Vision Transformer (ViT), Pretrained Transformers

CS60010

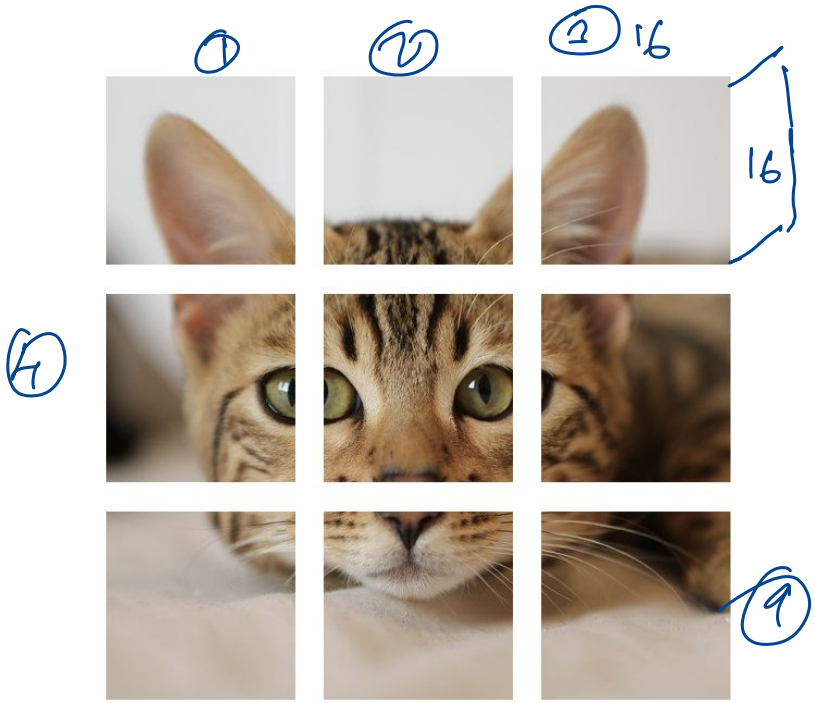
Encoding Images? Vision Transformer

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021


Contextual
Representations


Encoders
Embeddings

Encoding Images? Vision Transformer



16 × 16 × 3

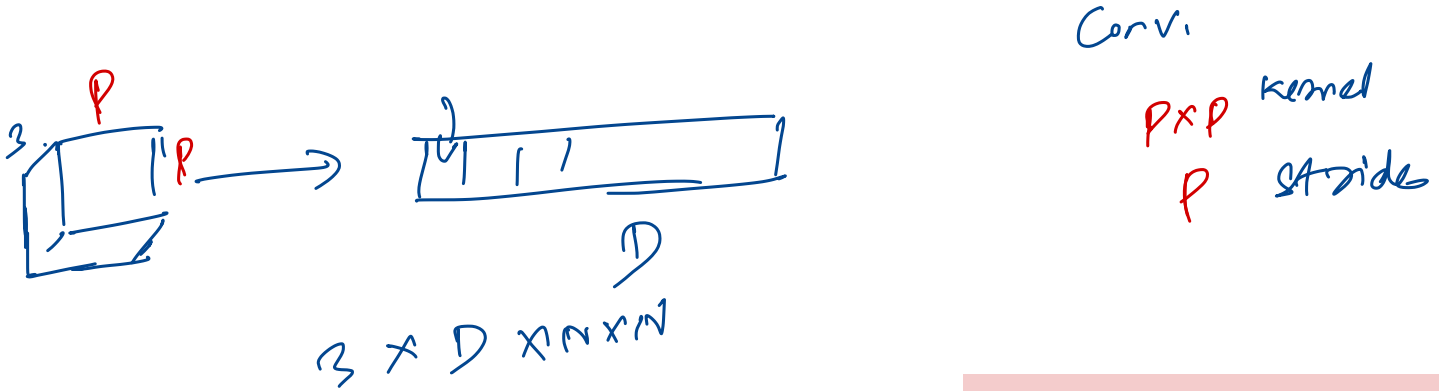
An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

Encoding Images? Vision Transformer

N input patches, each of shape 3x16x16

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

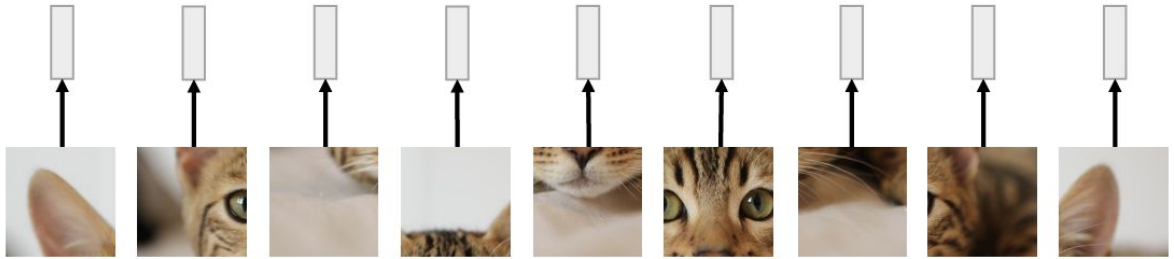
Encoding Images? Vision Transformer



What is this layer?

Linear projection to D-dimensional vector

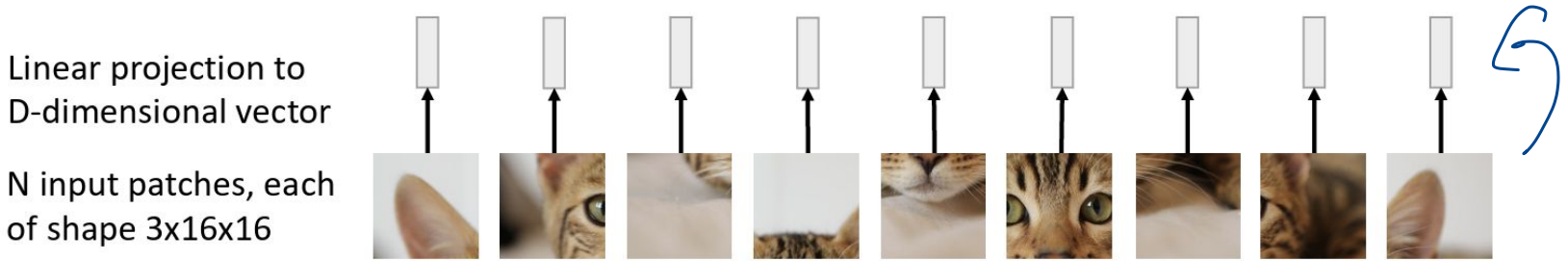
N input patches, each of shape 3x16x16



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

Encoding Images? Vision Transformer

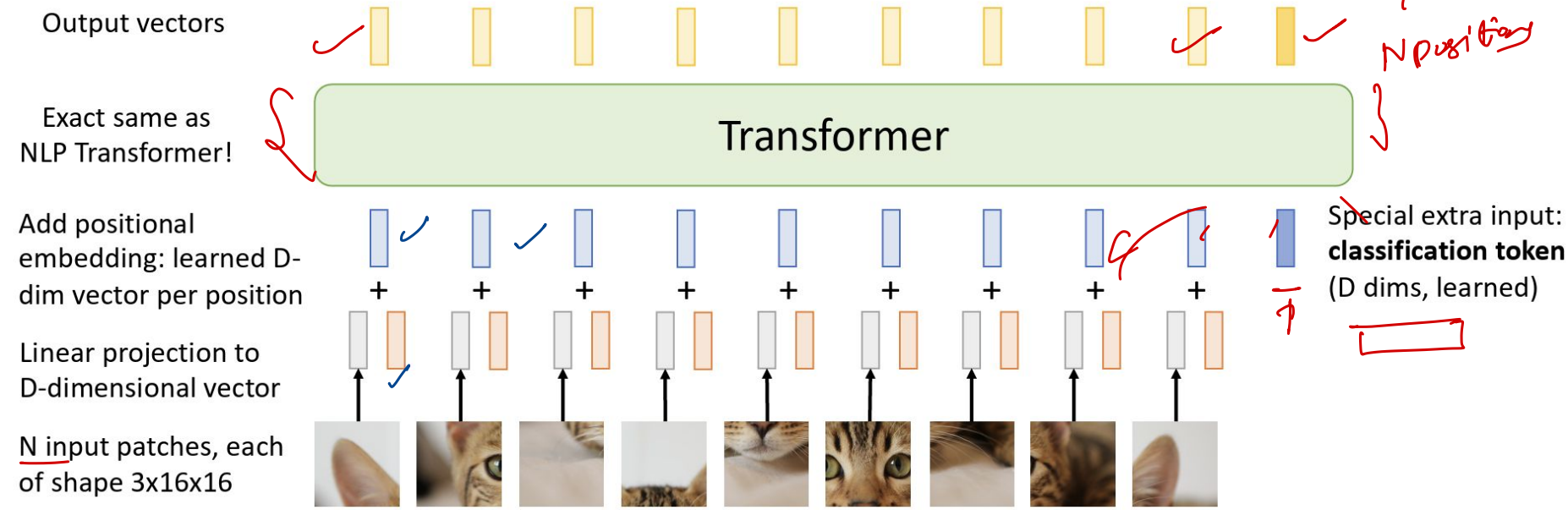
With patch size p, this layer is Conv2D (filter size: *pxp*, Input channels: *3*, Output channels: *D*, stride=*p*)



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

Encoding Images? Vision Transformer

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos} \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$$



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

Encoding Images? Vision Transformer

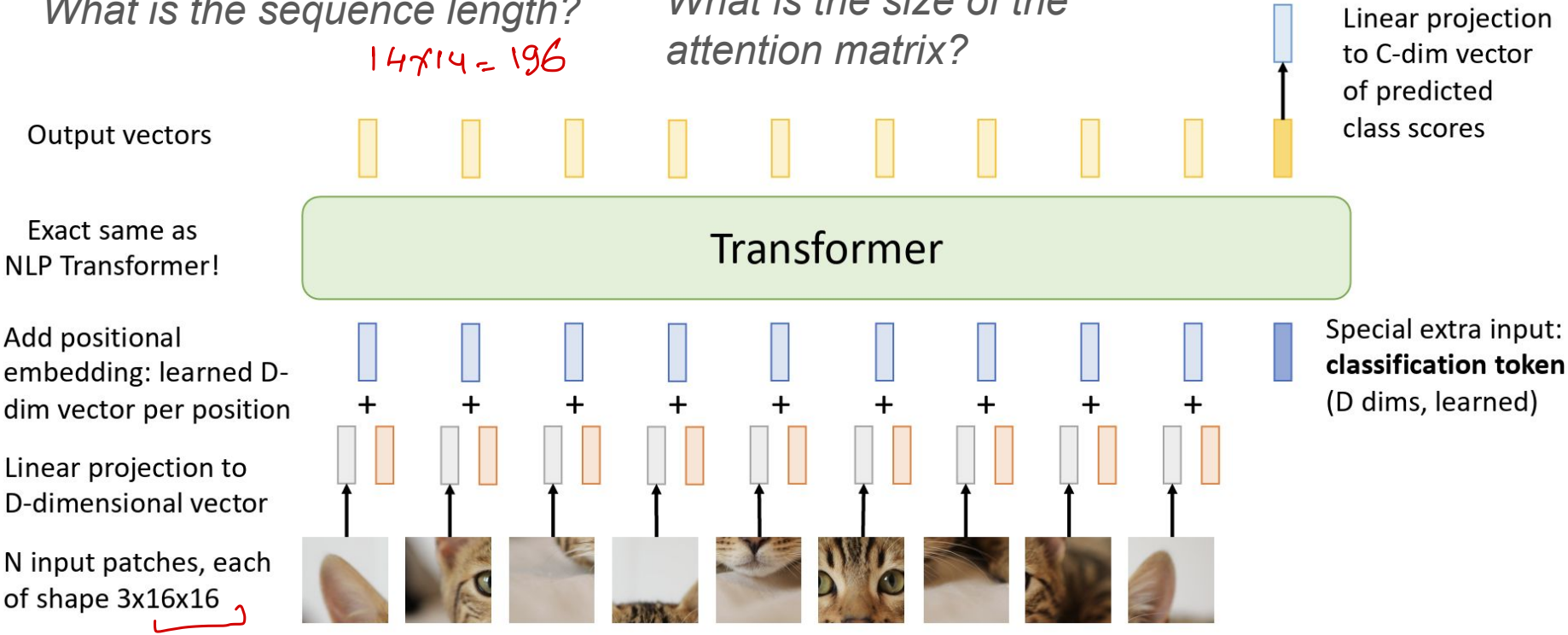
224x224

What is the sequence length?

14x14 = 196

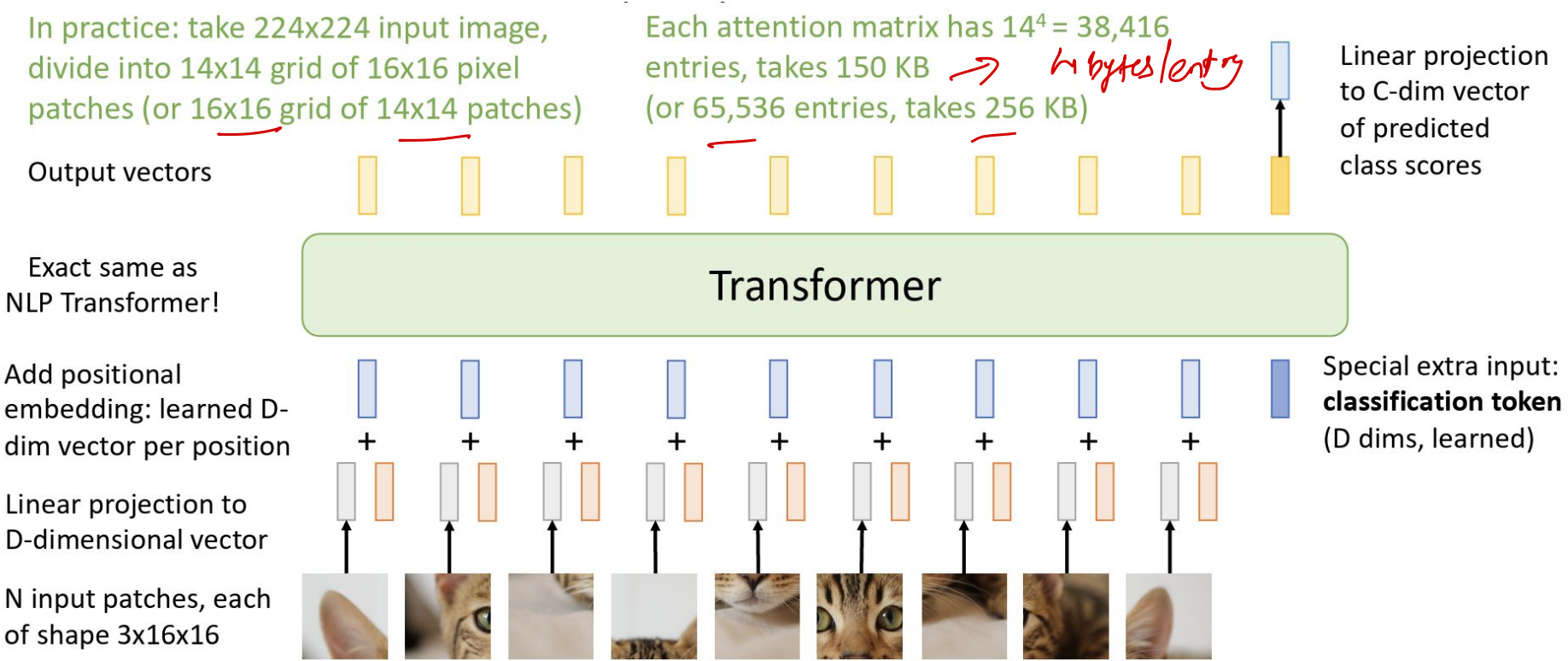
What is the size of the attention matrix?

14^4



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

Encoding Images? Vision Transformer



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale: ICLR 2021

Vision Transformer Model Variants

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

Can you make sense of the number of parameters?

model dim

n

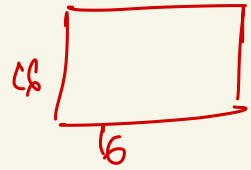
ViT Base

$12 \times d^2 \times L$

$= \boxed{12 \times (768)^2 \times 12} \approx 84\text{-}85\,M$

I/p → embedds

224×224 image



196 seq length

P.E. 196 × 768

Conv. 3 × 768 × 16 × 16

= 768 × (196 + 768)

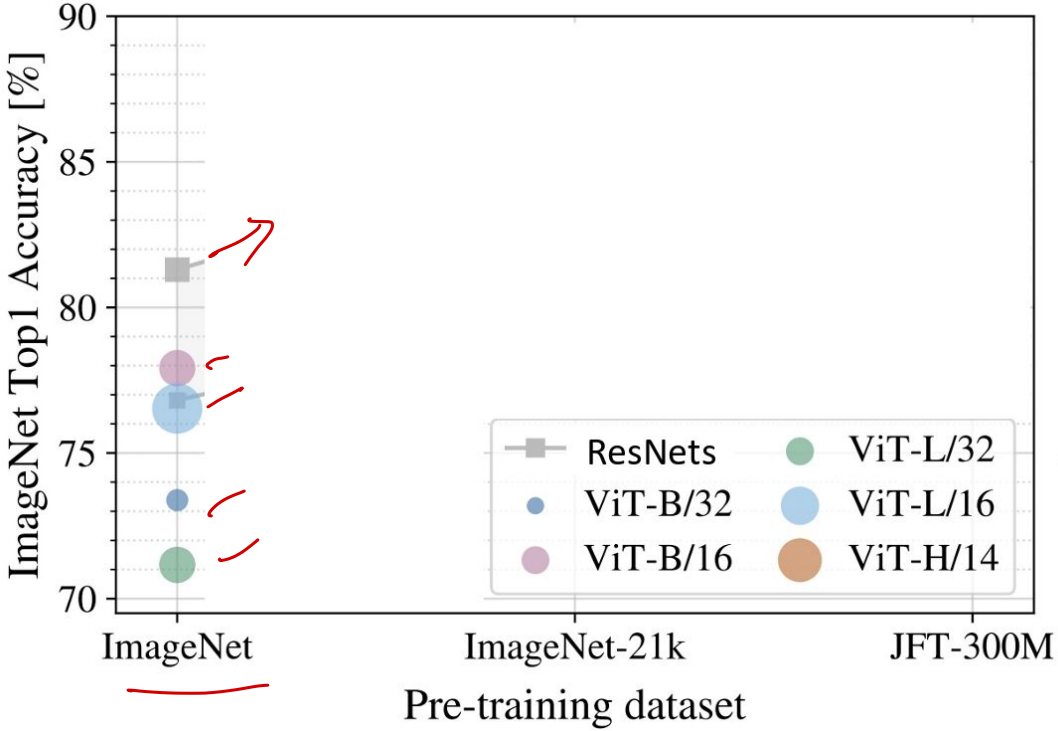
≈ 0.75 M

o/p to class 768 × 1000 ≈ 0.75 M

Performance Comparison: ViTs vs ResNets

Recall: ImageNet dataset has 1k categories, 1.2M images

When trained on ImageNet, ViT models perform worse than ResNets



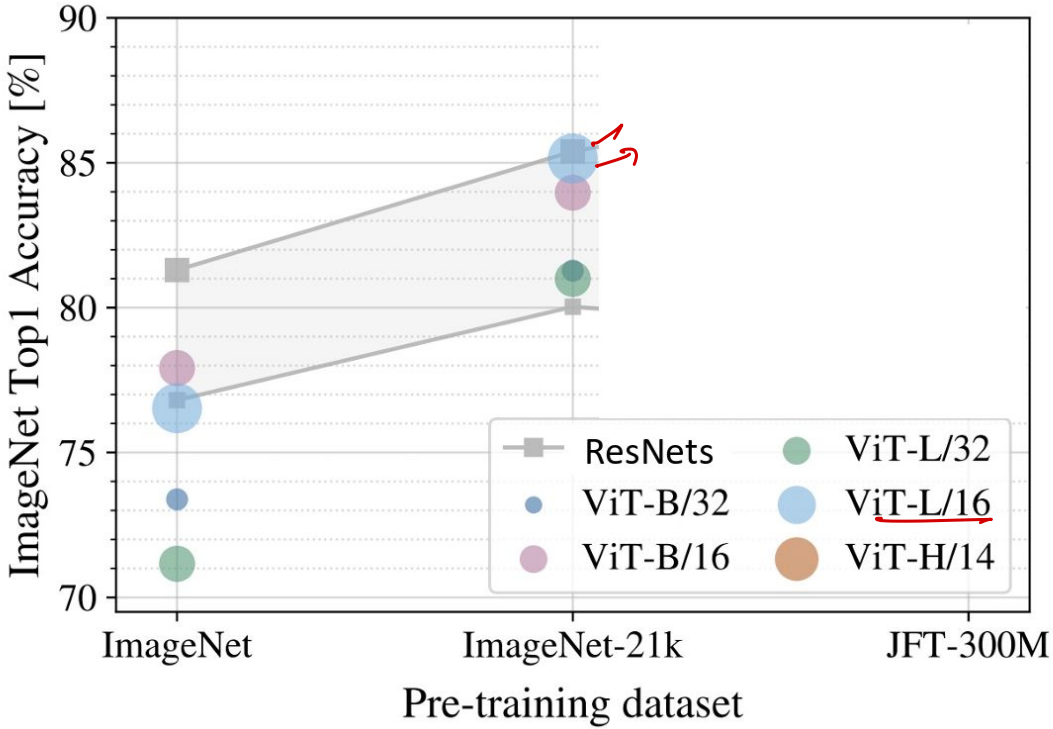
B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

Performance Comparison: ViTs vs ResNets

ImageNet-21k has 14M images with 21k categories

If you pretrain on ImageNet-21k and fine-tune on ImageNet, ViT does better: big ViTs match big ResNets



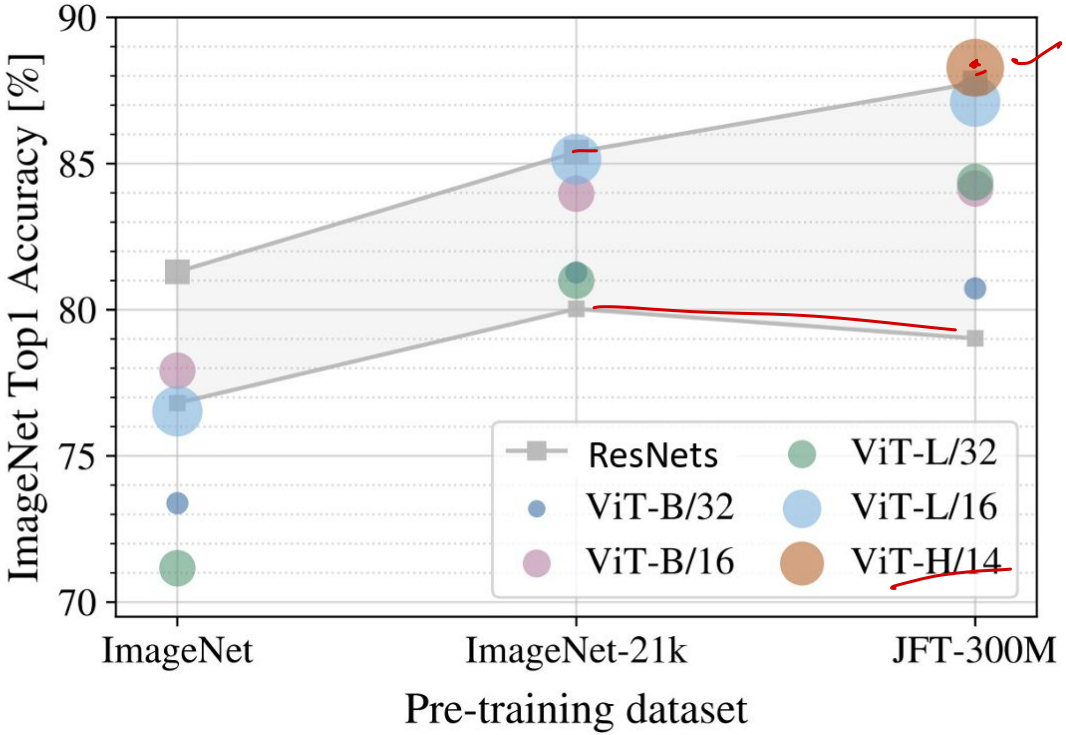
B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

Performance Comparison: ViTs vs ResNets

JFT-300M is an internal Google dataset with 300M labeled images

If you pretrain on JFT and finetune on ImageNet, large ViTs outperform large ResNets



B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

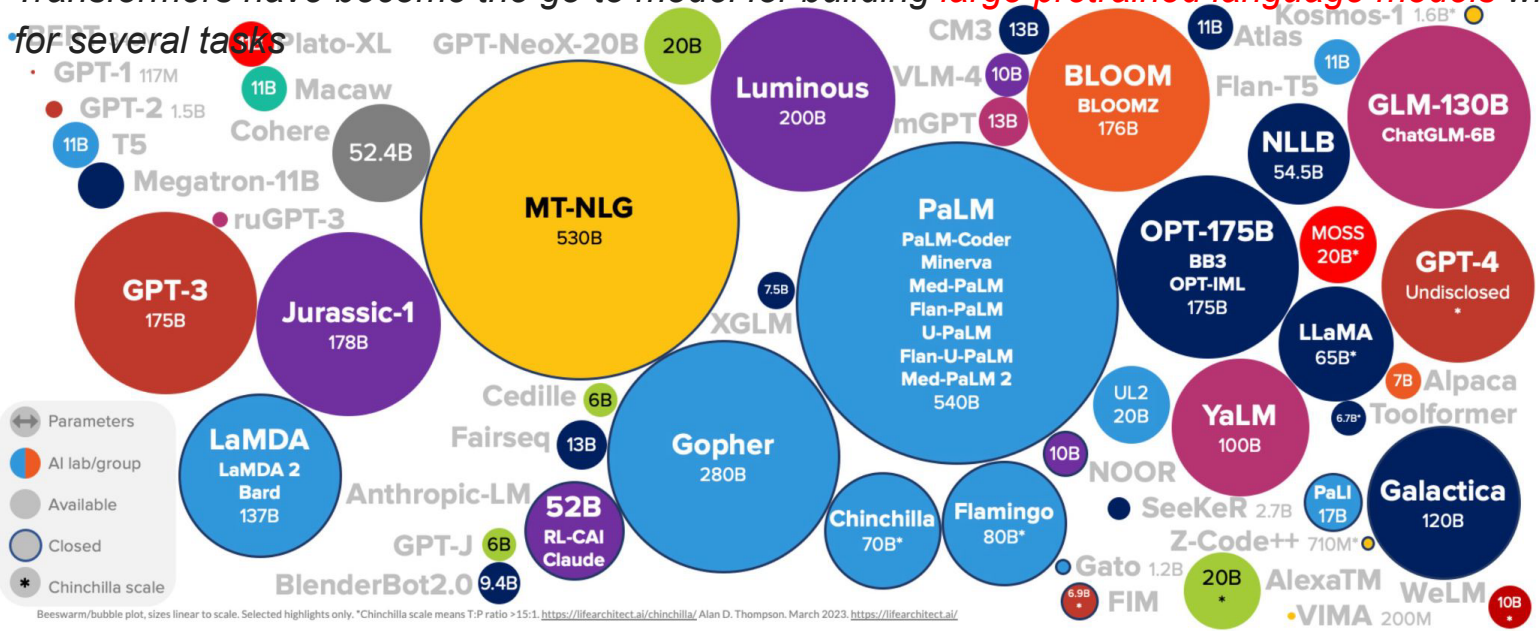
What is pretraining? Let’s go back to text

The idea of transfer learning

Transformers are not always better than RNNs by themselves.

However, what has made Transformers popular is that they can be combined with the idea of transfer learning.

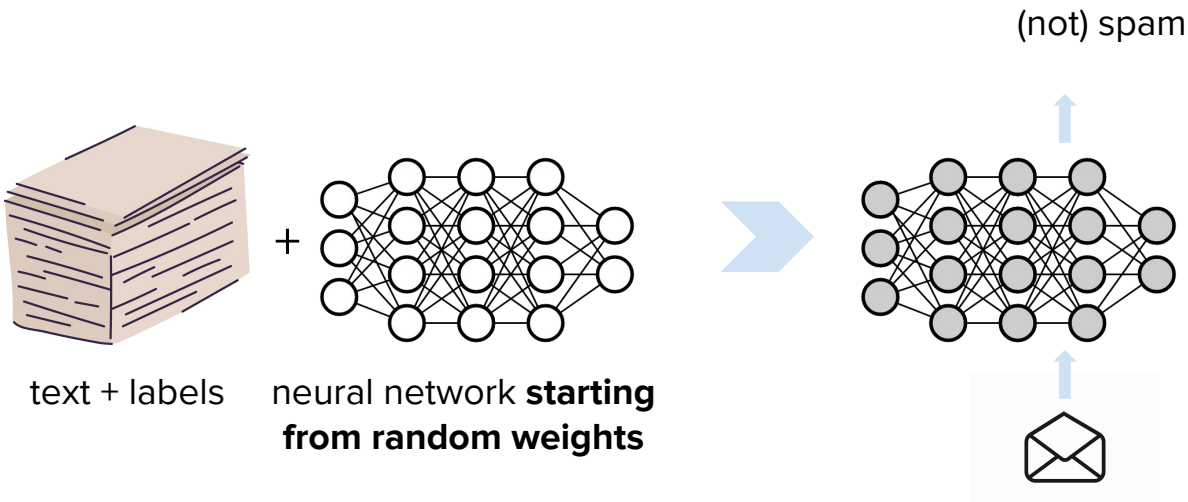
Transformers have become the go-to model for building large pretrained language models which can be adapted for several tasks



RNNs
↓
Transformers

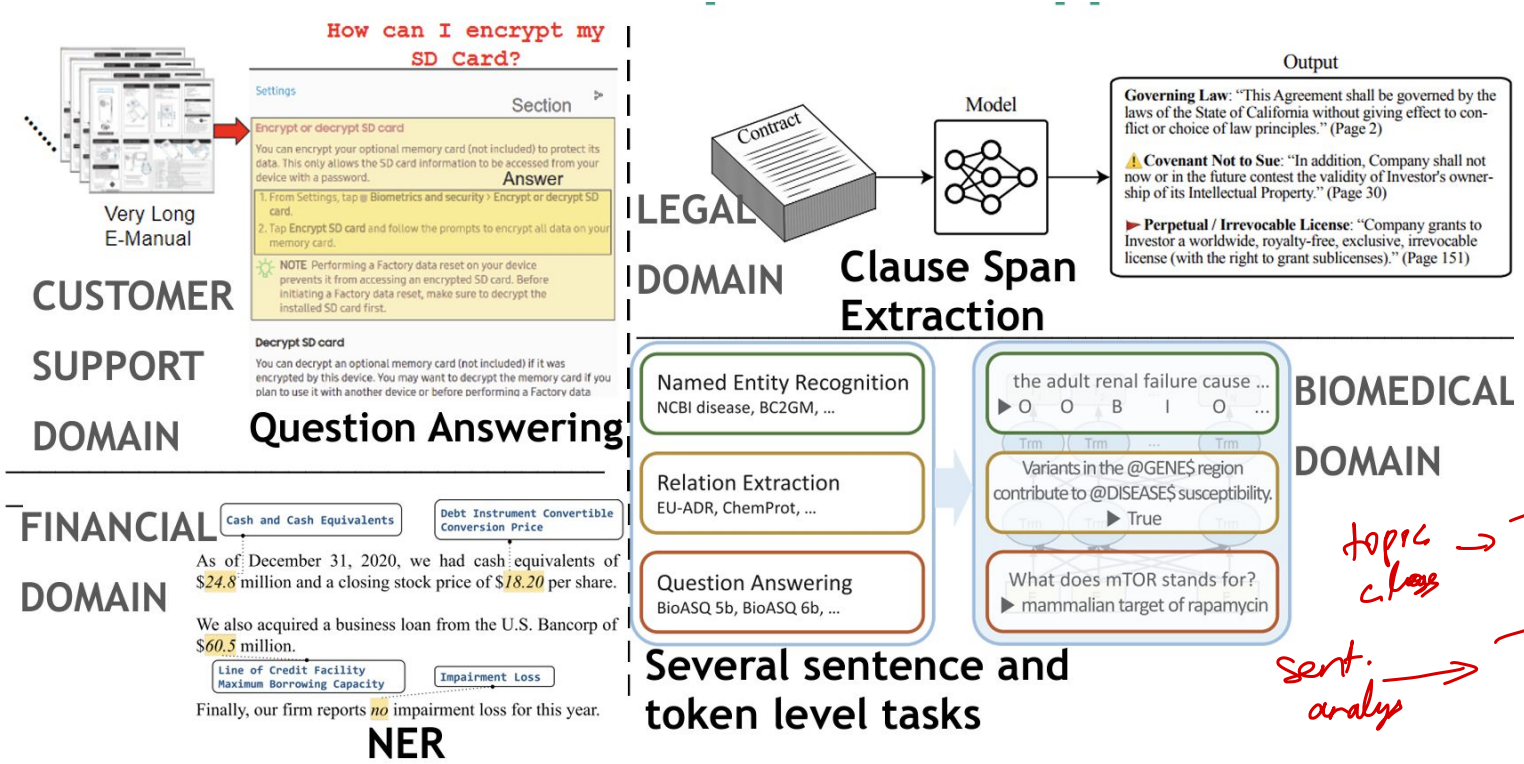
https://rycolab.io/classes/llm-s24/

Standard Supervised Deep Learning



https://utah-cs6340-nlp.notion.site/Natural-Language-Processing-bd1a2ca290fc44f69556908ad8d25c70

Abundance in domain-specific NLP Applications

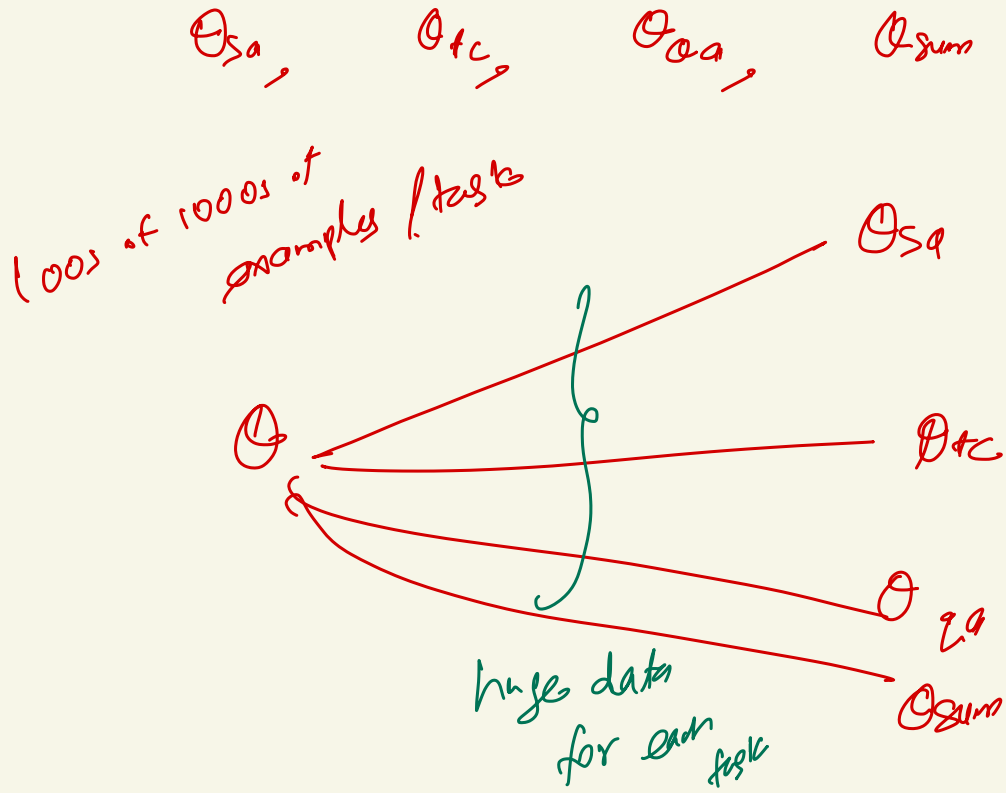


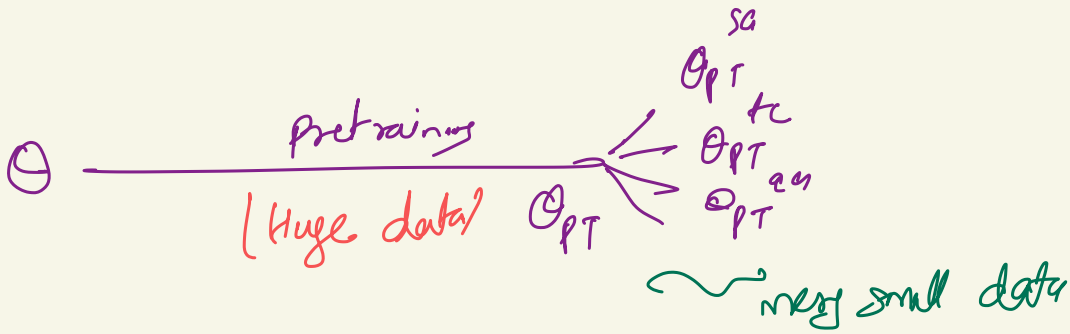
Languages

topic class → Transformers → Ote

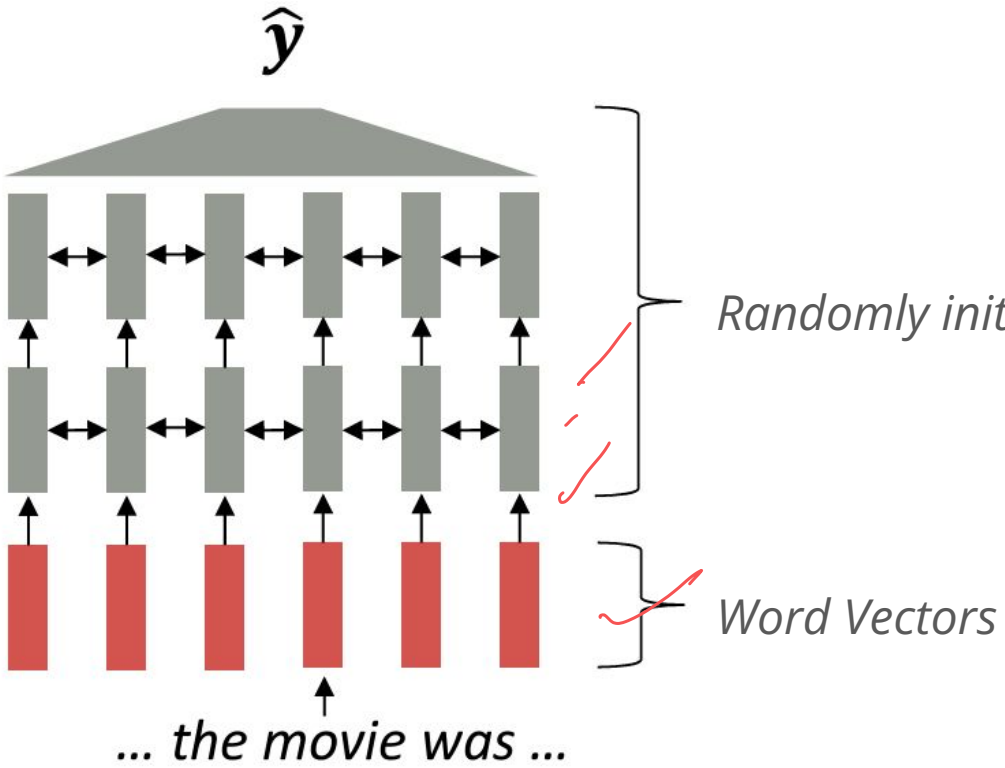
sent. analys → Transformers → Osa

Very difficult to get sufficient labeled data for each task


θ_{sa}, θ_{tc}, θ_{qa}, θ_{sum}
100s of 1000s of examples / tasks
θ_{sa}
θ
θ_{tc}
θ_{qa}
huge data for each task
θ_{sum}


pretraining
Huge data
θ_{PT}
SA
θ_{PT}
tc
θ_{PT}
qa
θ_{PT}
very small data

Why is this an issue

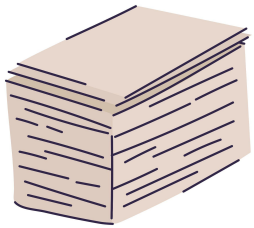


The training data we have for our downstream task (like question answering) must be sufficient to teach all contextual aspects of language.

Pretrain-then-Finetune (from 2018)

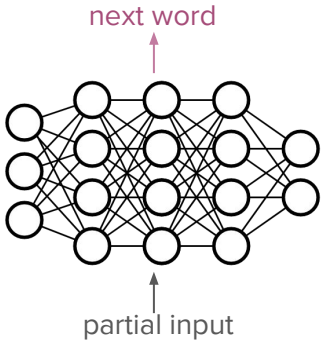
Stage 1:
Pretrain a model

neural network **starting from <u>random</u> weights**



text

+

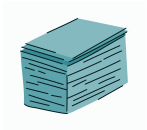


pretrained weights

Objective: generate next or masked word
(does not require that people label the next word)

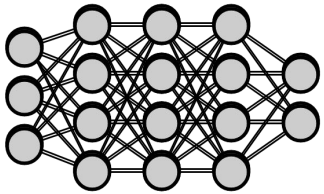
Stage 2:
Finetune the model

neural network **starting from <u>pretrained</u> weights**



text + **labels**

+



Objective: standard supervised training

https://utah-cs6340-nlp.notion.site/

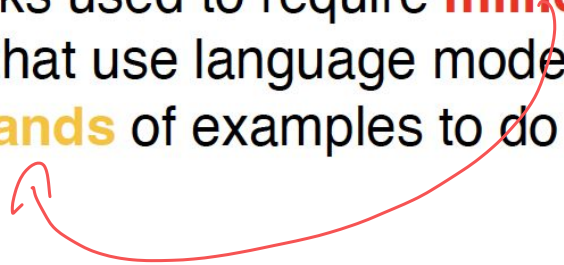
Transfer Learning with Language Modeling

Language modelling is the popular choice of pretraining in NLP.

The **key idea** is to **train a neural network language model (on vast text corpora).** And **then, use it to transfer that knowledge to any target task in NLP** we care about.

This is a **very successful** idea in NLP:

- Supervised systems for various NLP tasks used to require **millions of examples** to learn tasks. NLP systems that use language models as a starting point, need much fewer - **thousands** of examples to do so!



https://rycolab.io/classes/llm-s24/

What does a model learn from pretraining?

- There are canines everywhere! One dog in the front room, and two dogs
- It wasn't just big it was enormous
- The author of "A Room of One's Own" is Virginia Woolf
- The doctor told me that he
- The square root of 4 is 2

- *Text contains enormous amounts of knowledge*
- *Pretraining on lots of text with all that knowledge is what gives language models their ability to do so much*

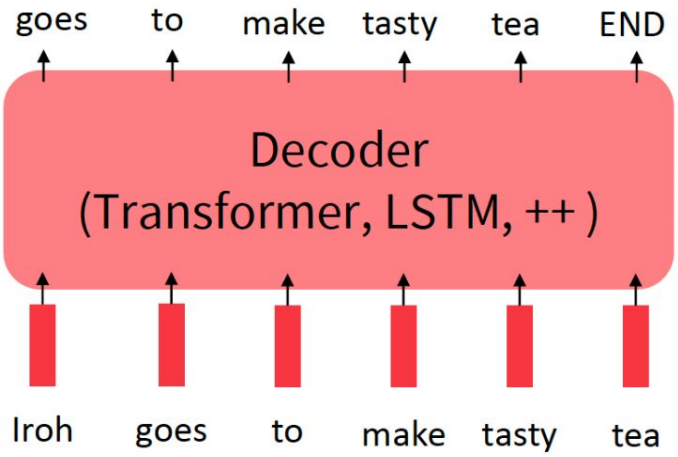
https://web.stanford.edu/~jurafsky/slp3/

The Pretraining / Finetuning paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

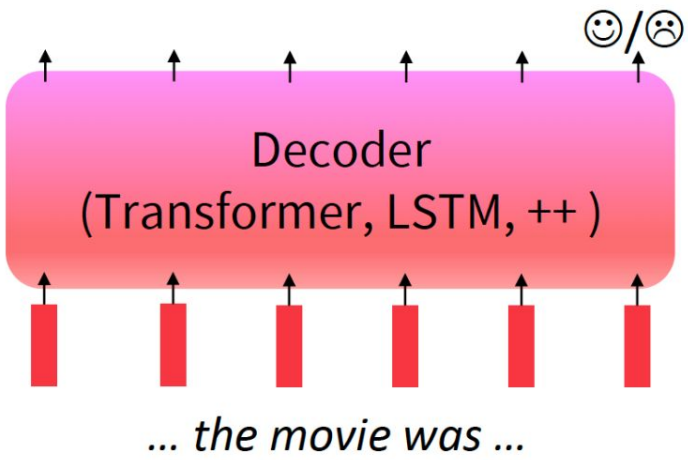
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!

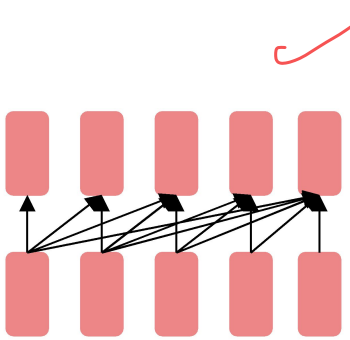


Step 2: Finetune (on your task)

Not many labels; adapt to the task!

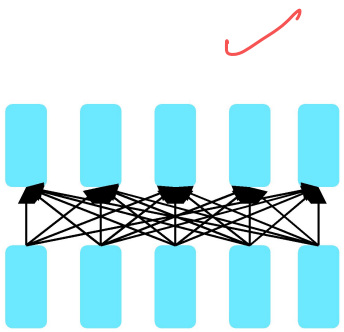


Three architectures for large language models



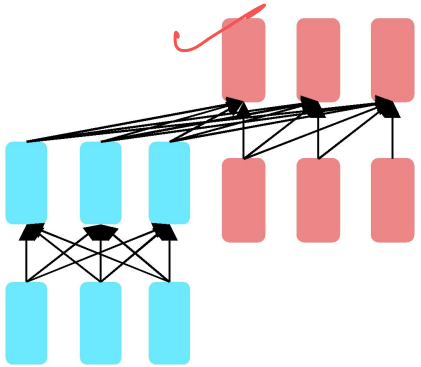
Decoders

GPT, Claude,
Llama, Mixtral



Encoders

BERT family



Encoder-decoders

Flan-T5
BART

Pretrain Decoder

predict next word

The

Pretrain Encoders

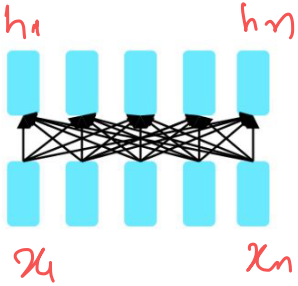
↓

Can't use

next word

prediction

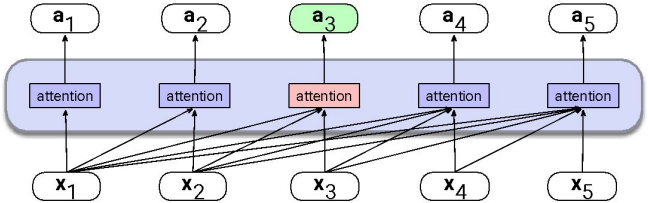
Pretraining Encoders



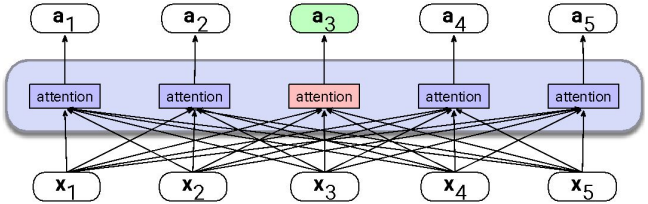
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?

- map sequences of input embeddings (x1,...,x*n*)
- to sequences of output embeddings of the same length (h1,...,h*n*),
- where the output vectors have been contextualized using information from the entire input sequence.



a) A causal self-attention layer



b) A bidirectional self-attention layer

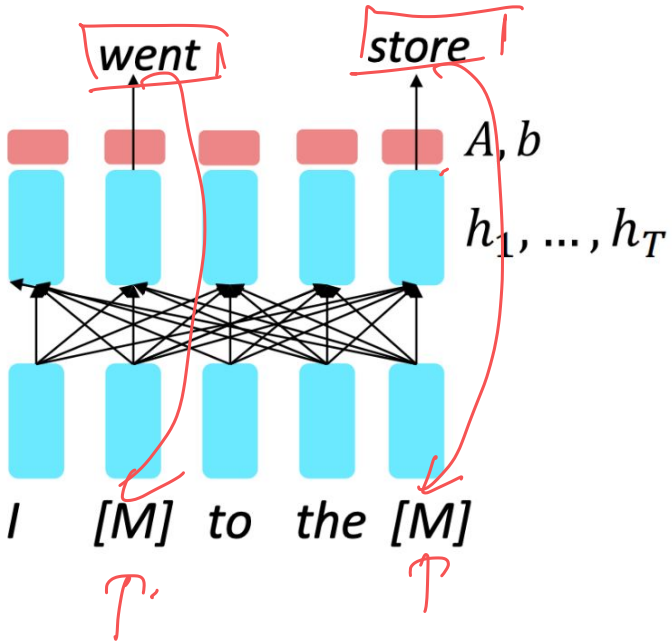
https://web.stanford.edu/~jurafsky/slp3/

Solution: Use Masks

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$

Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$. Called **Masked LM**.



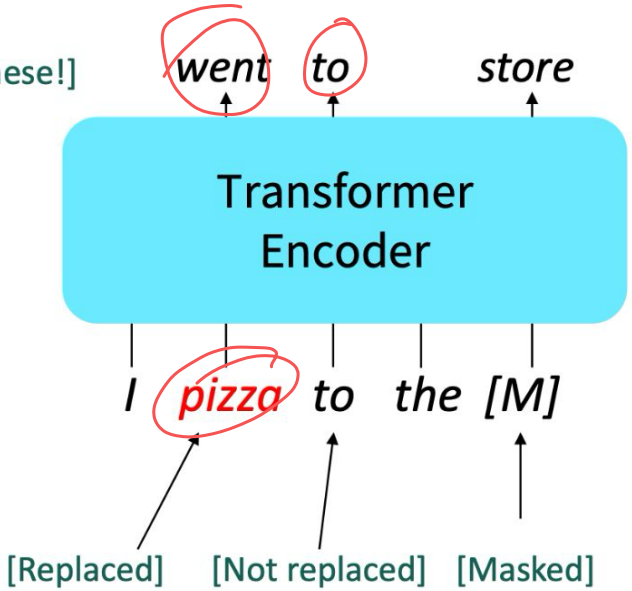
https://web.stanford.edu/class/cs224n/

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

Some more details about Masked LM for BERT:

- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



More Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - "Pretrain once, finetune many times."