Transformers
(no not that kind…)

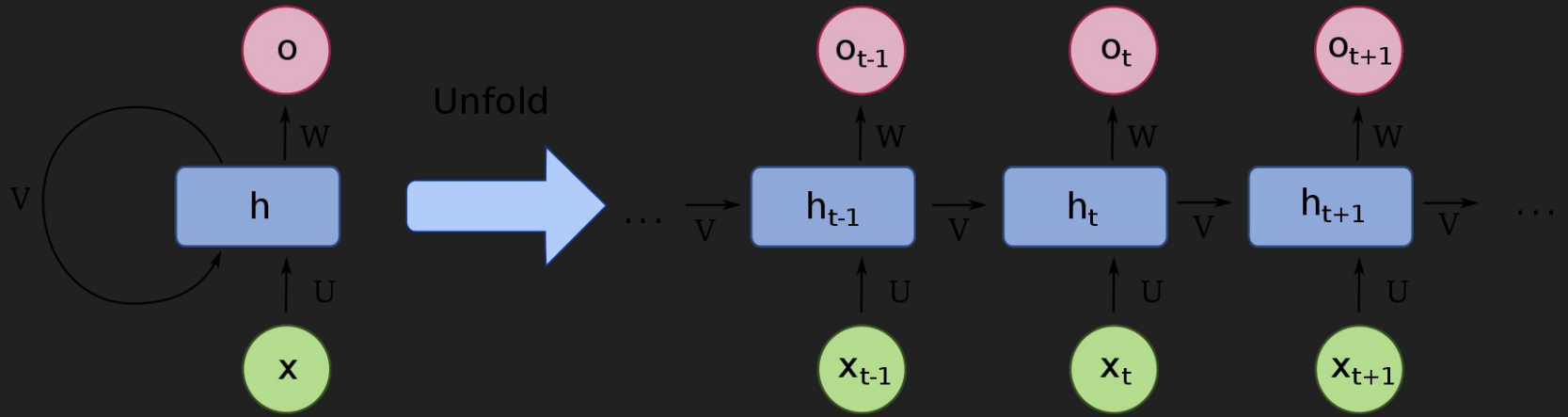
Areeb Gani, Michael Ilie, Vijay Shanmugam

Welcome!

ml.mbhs.edu

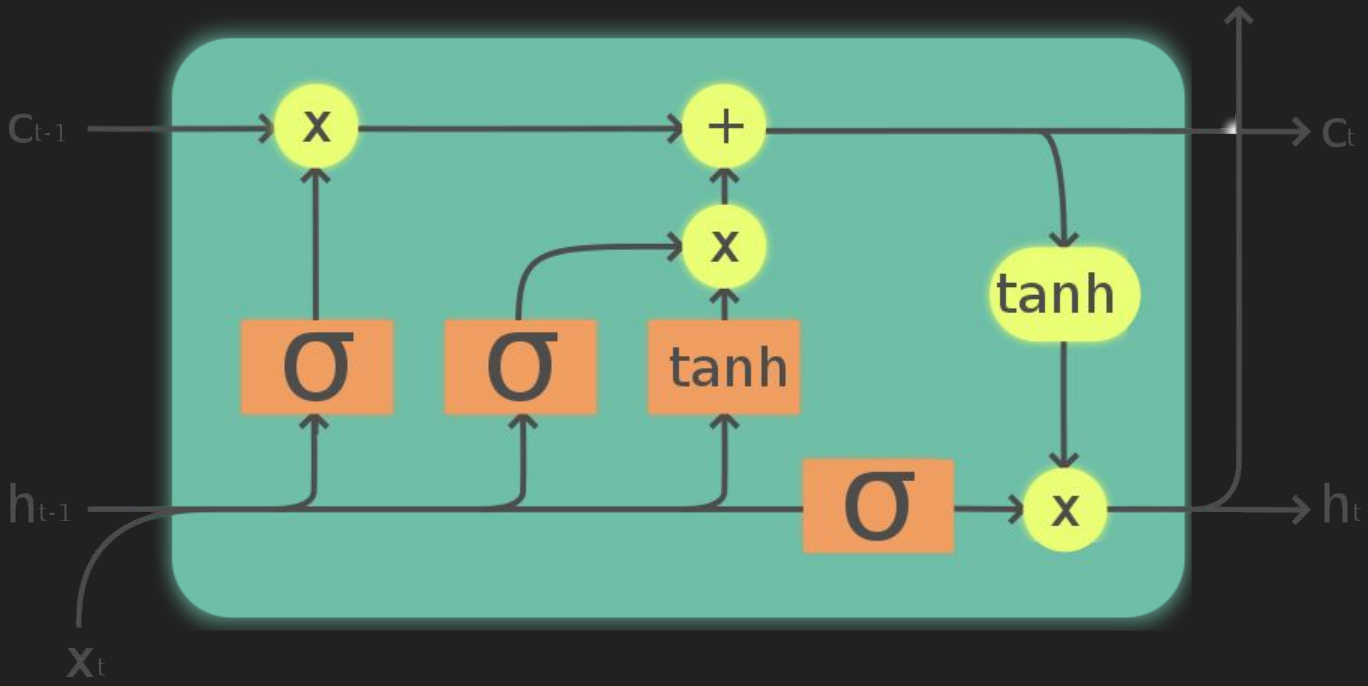
Recurrent neural network (1986)

- Language Translation
- Text Summarization
- Next Sentence Prediction



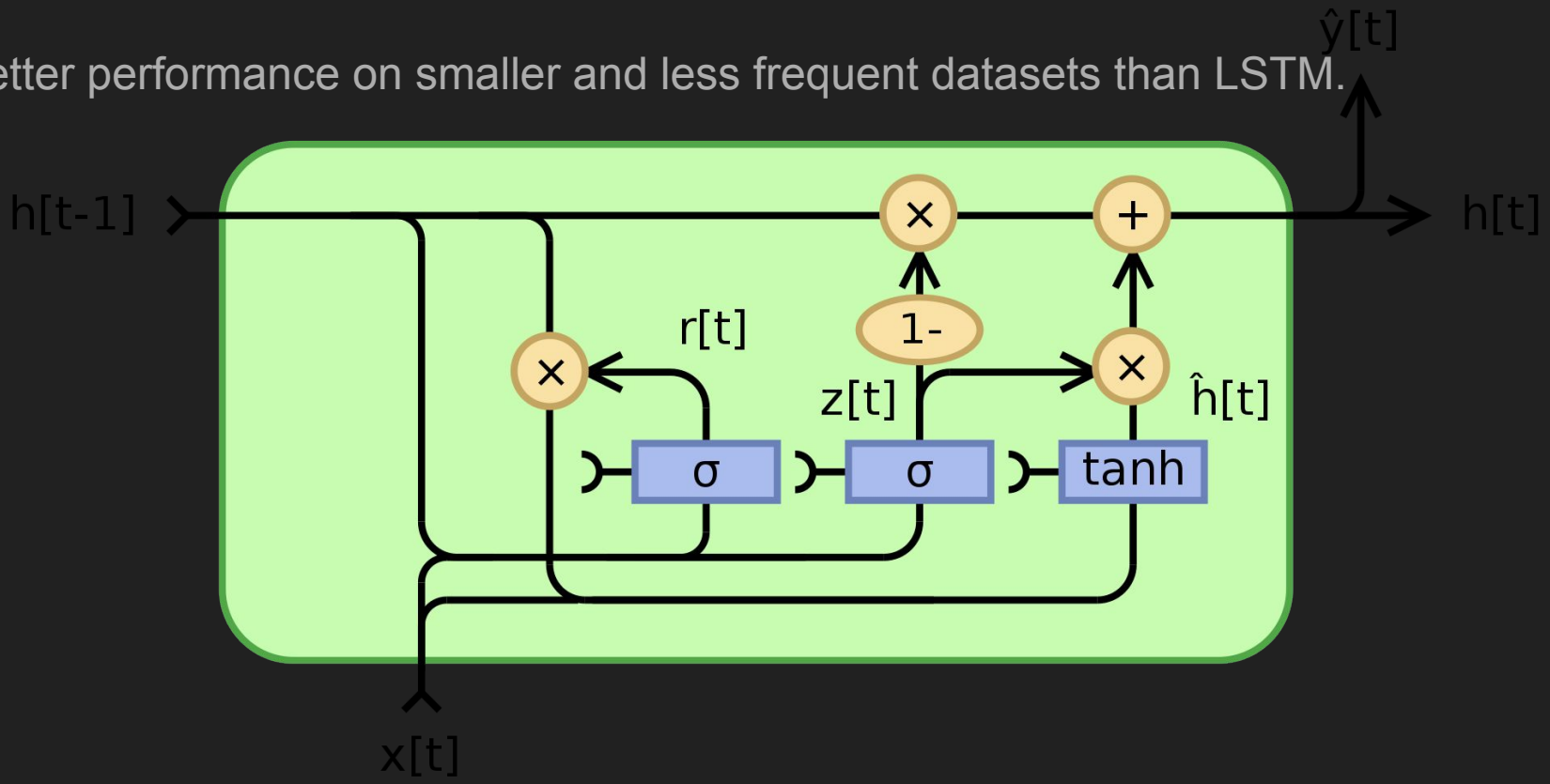
LSTM (1997)

RNNs using LSTM units partially solve the vanishing gradient problem



GRU (2014)

Better performance on smaller and less frequent datasets than LSTM.



Attention is all you need (2017)

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

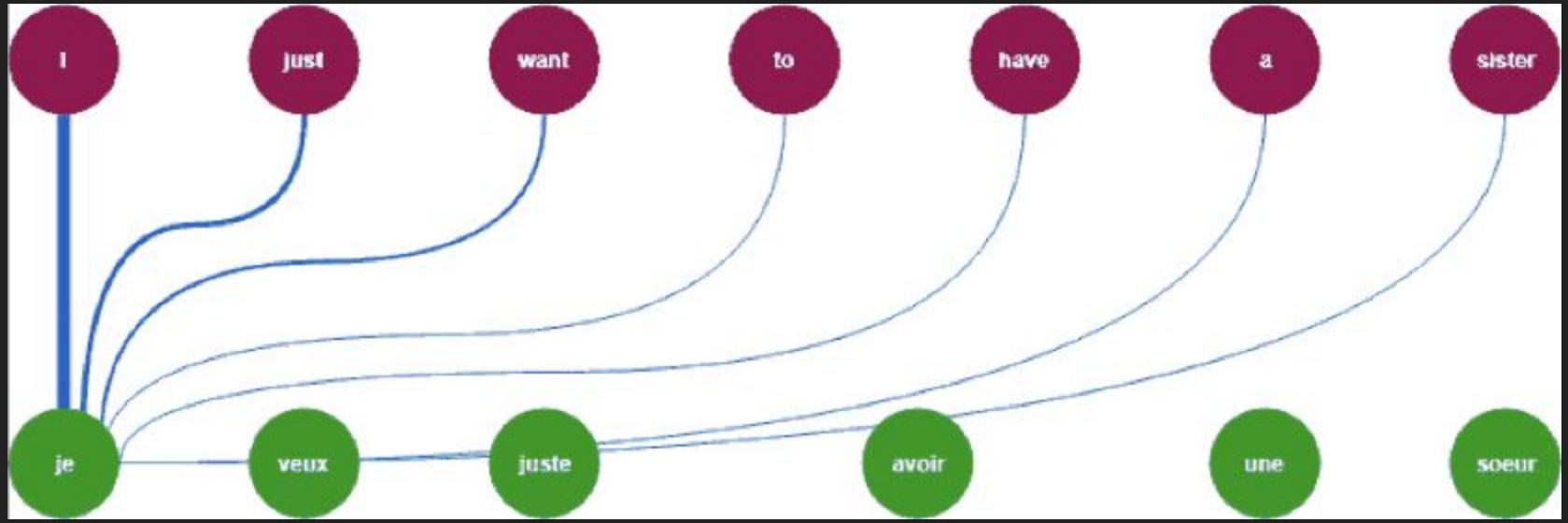
Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

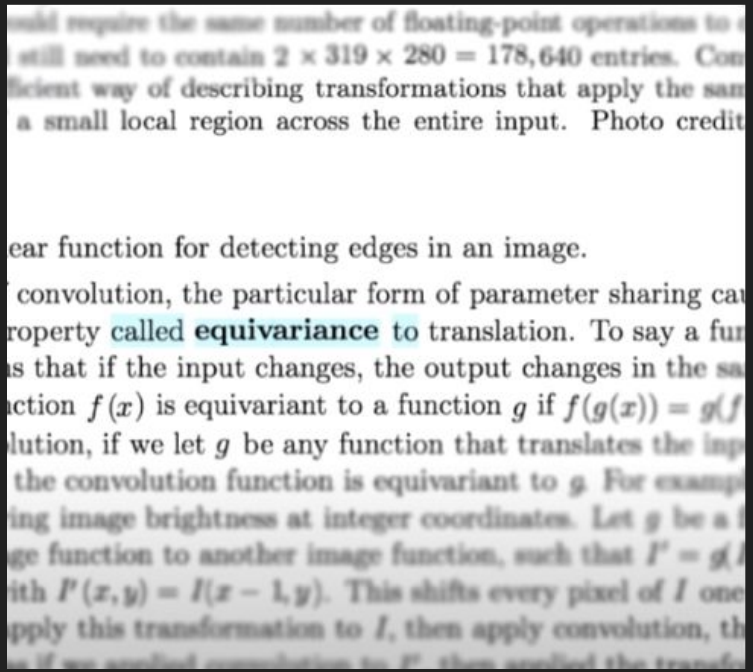
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Attention in Neural networks



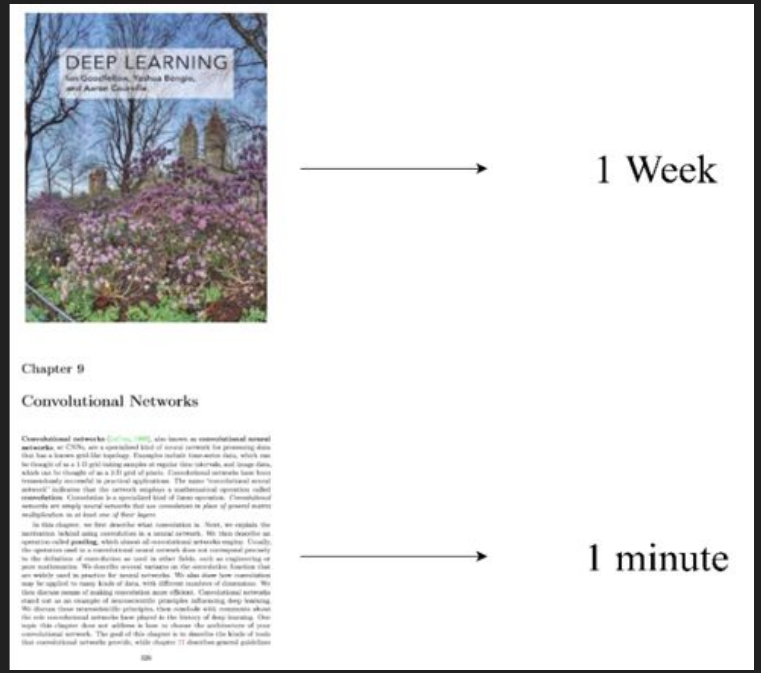
Example : Searching for “Equivariance” in a book



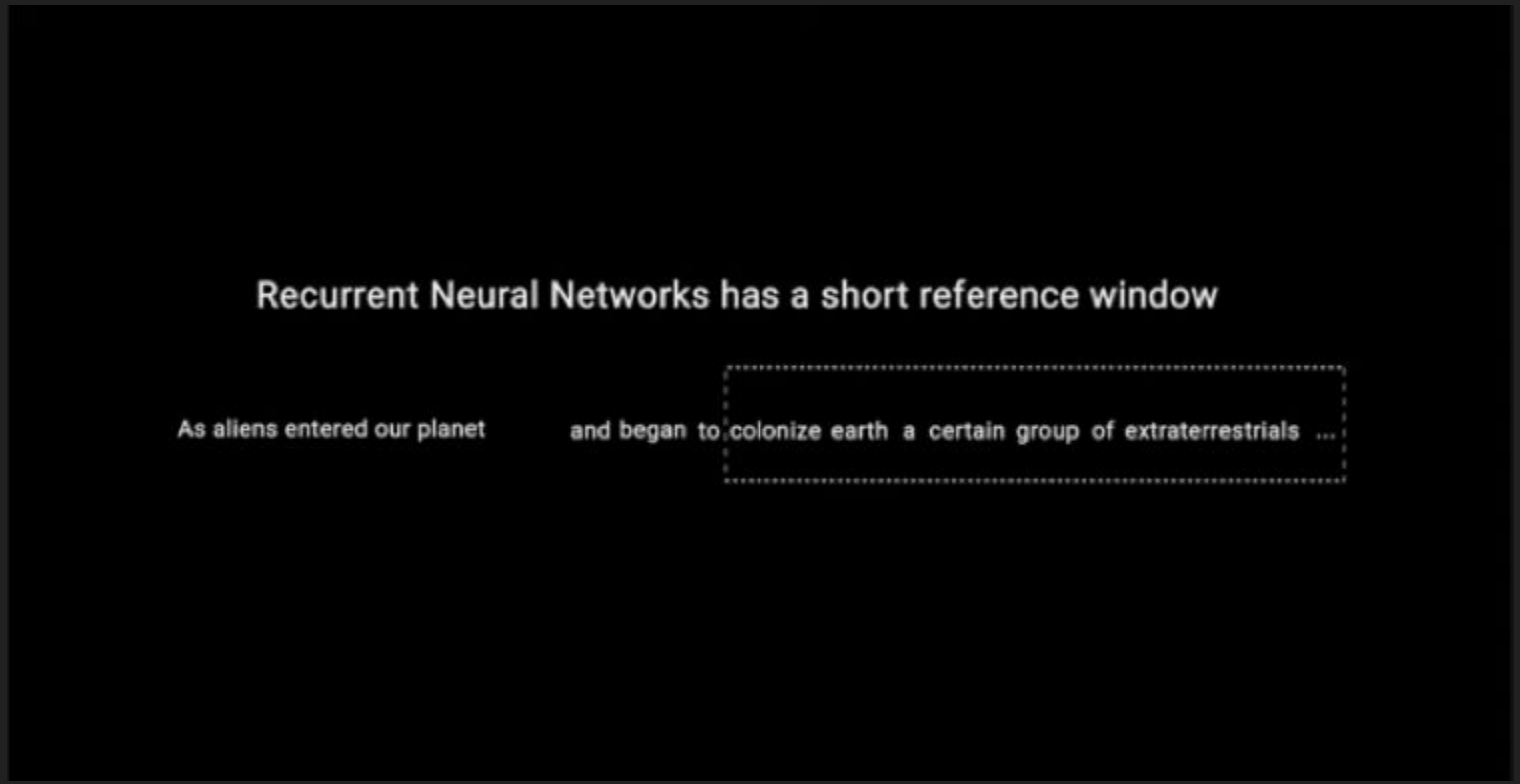
ould require the same number of floating-point operations to
still need to contain $2 \times 319 \times 280 = 178,640$ entries. Co
fficient way of describing transformations that apply the sa
a small local region across the entire input. Photo credit

ear function for detecting edges in an image.

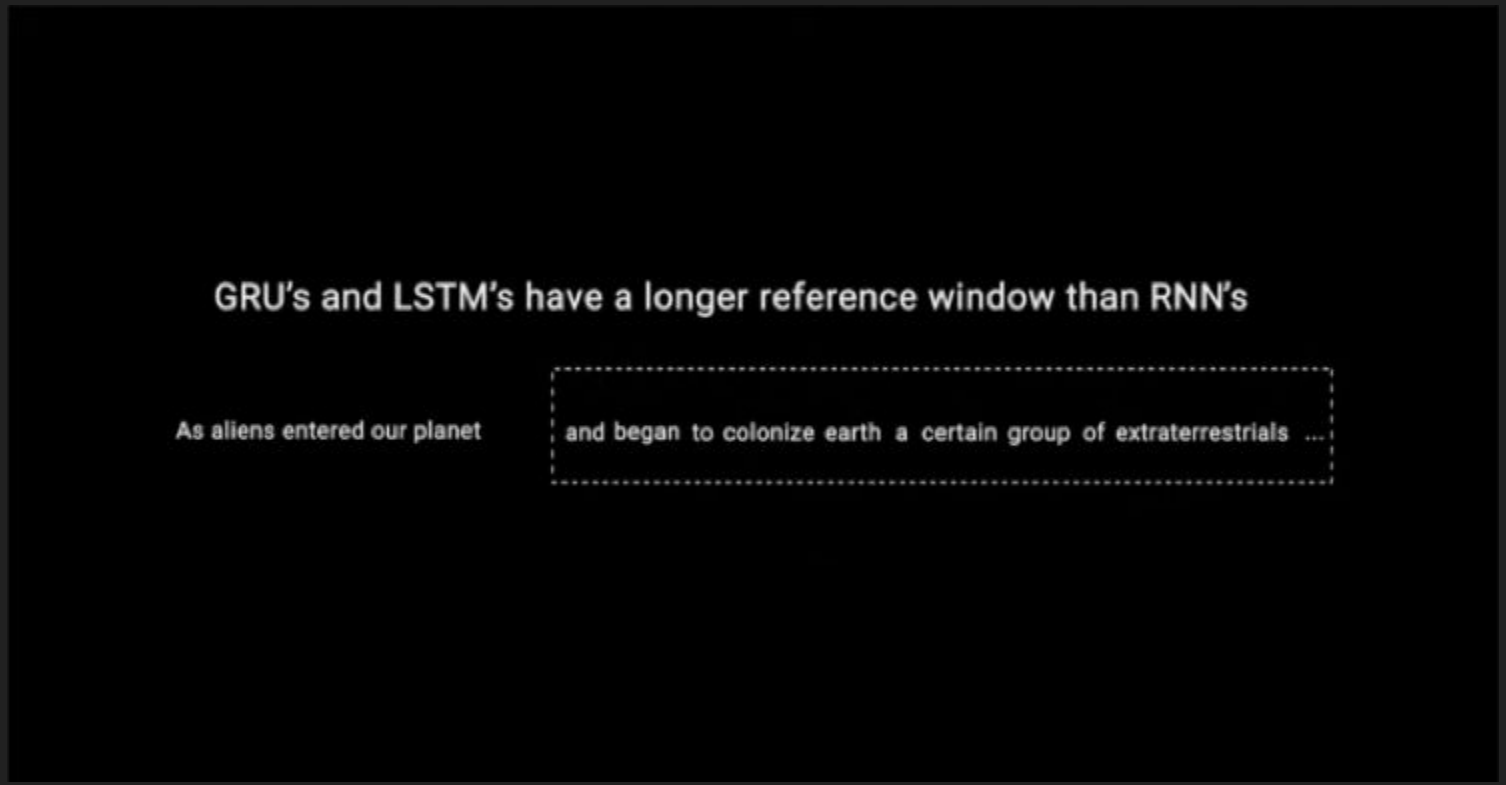
convolution, the particular form of parameter sharing ca
roperty called **equivariance** to translation. To say a fu
s that if the input changes, the output changes in the sa
nction $f(x)$ is equivariant to a function g if $f(g(x)) = g(f$
lution, if we let g be any function that translates the inp
the convolution function is equivariant to g. For exampl
ing image brightness at integer coordinates. Let g be a f
ge function to another image function, such that $I' = g(I$
ith $I'(x, y) = I(x - 1, y)$. This shifts every pixel of I one
pply this transformation to I, then apply convolution, th



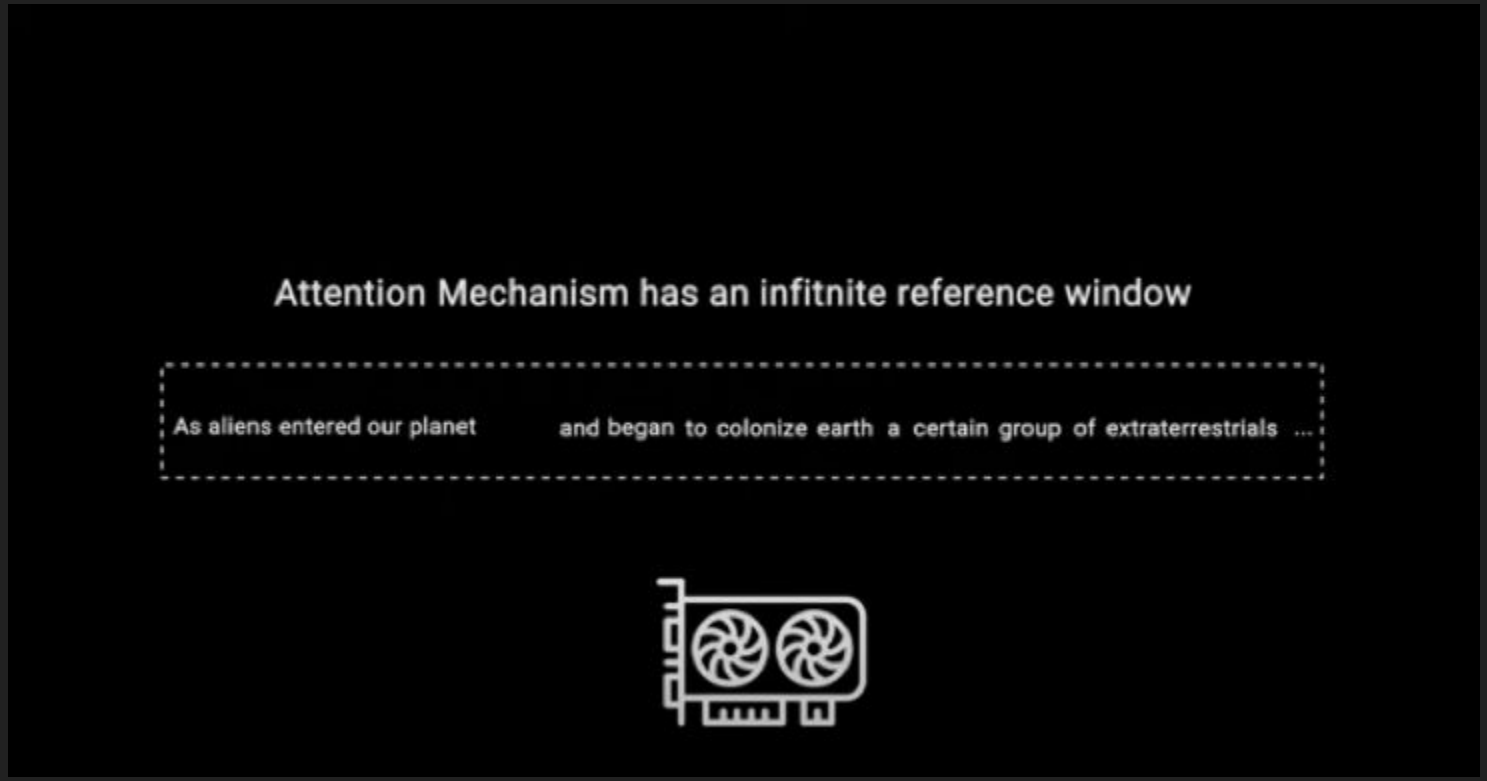
RNN vs LSTM vs Attention



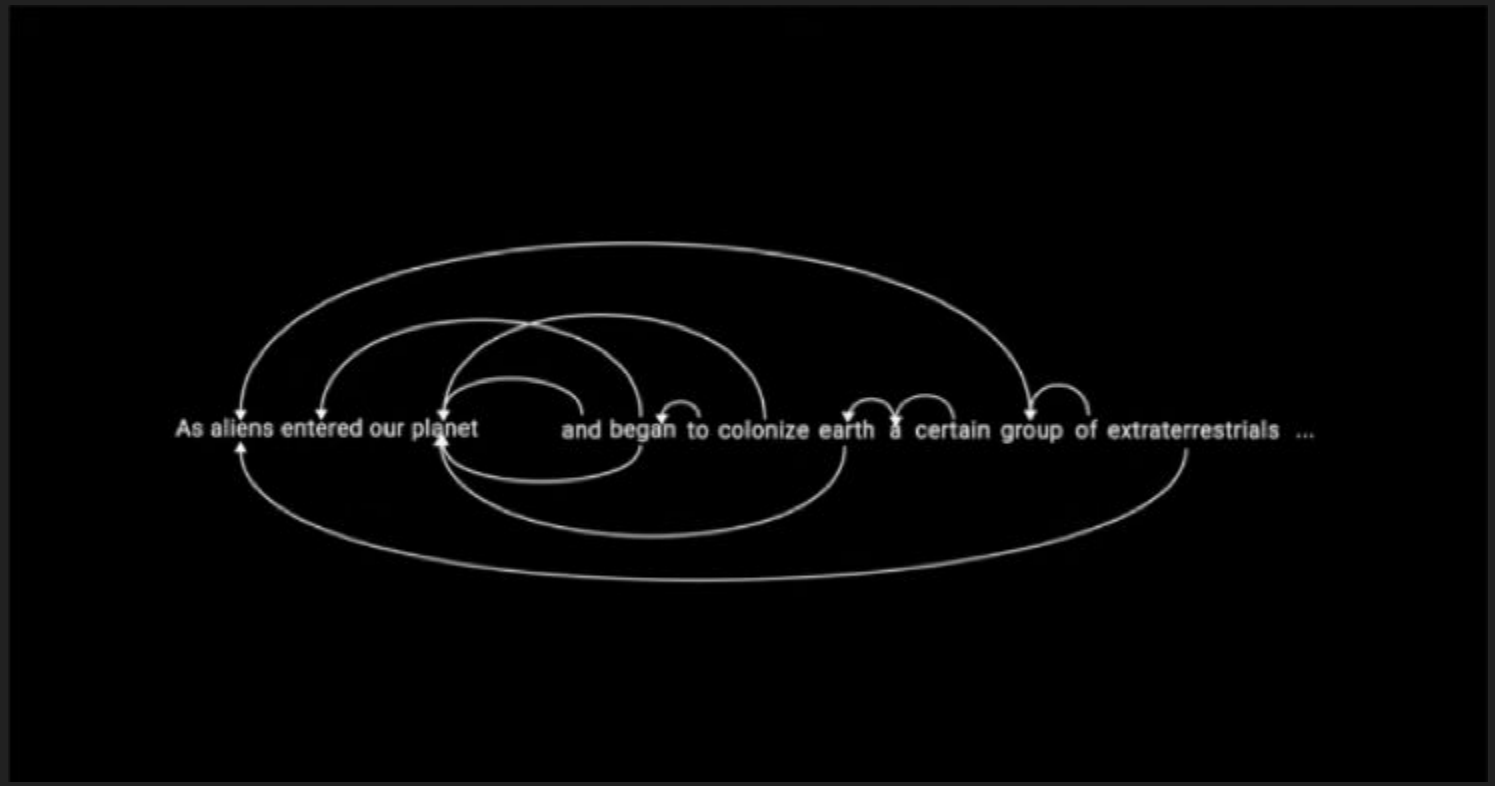
RNN vs **LSTM** vs Attention



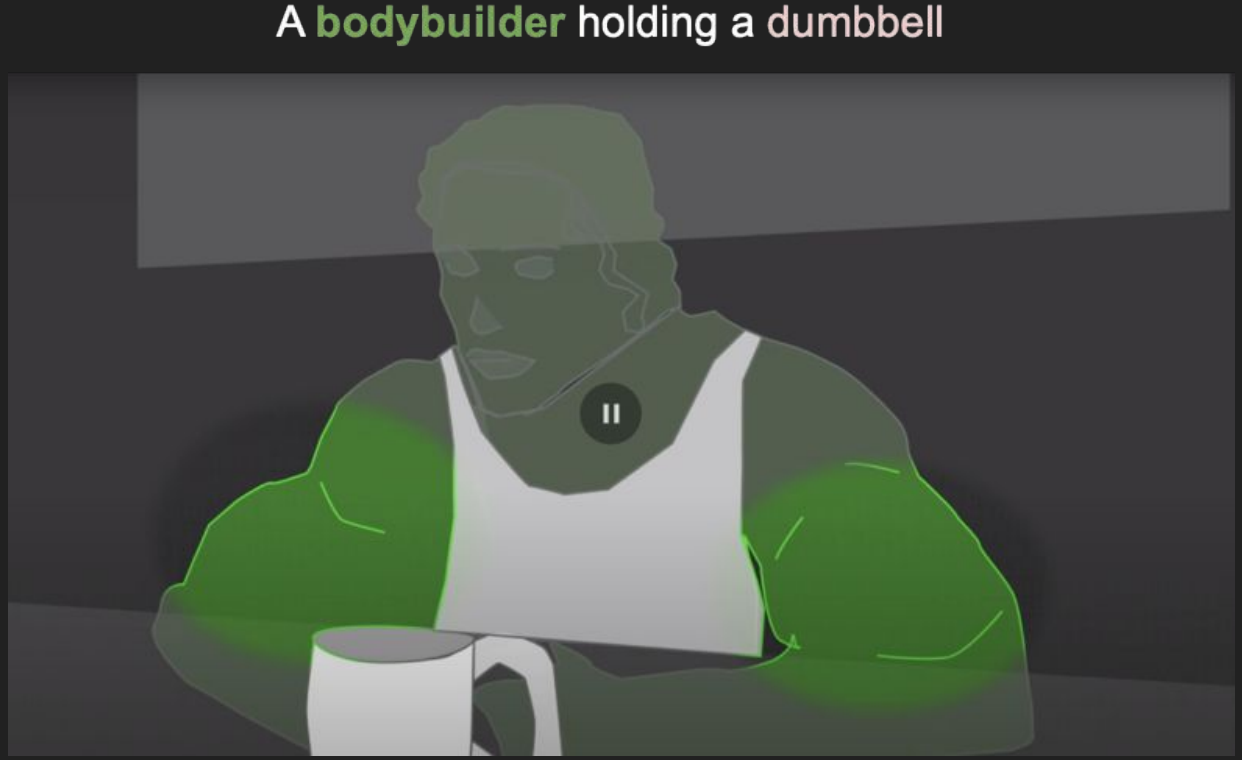
RNN vs LSTM vs **Attention**



RNN vs LSTM vs **Attention**



Attention example in images



Microsoft Attention GANs

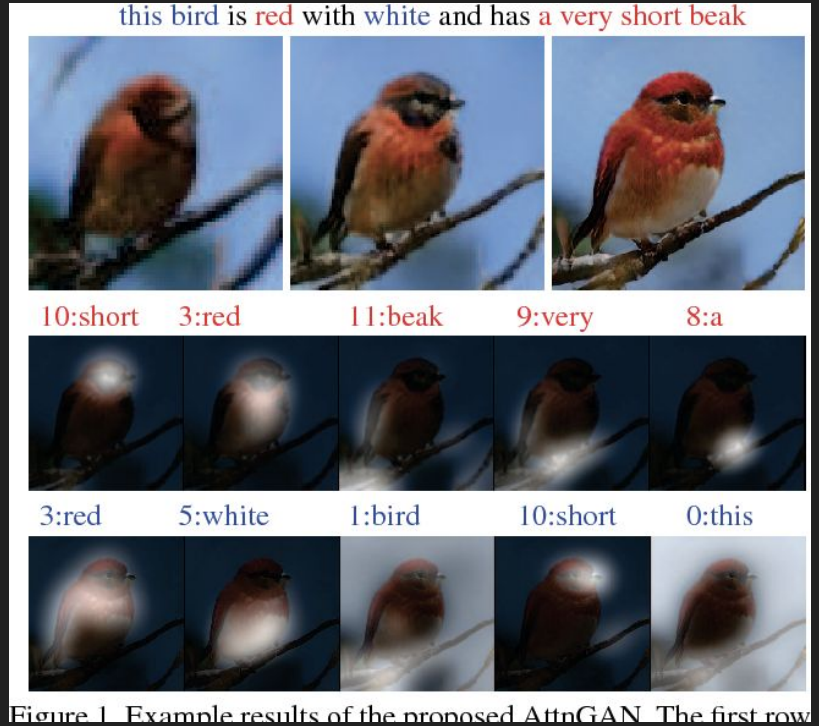


Figure 1. Example results of the proposed AttnGAN. The first row

Transformers

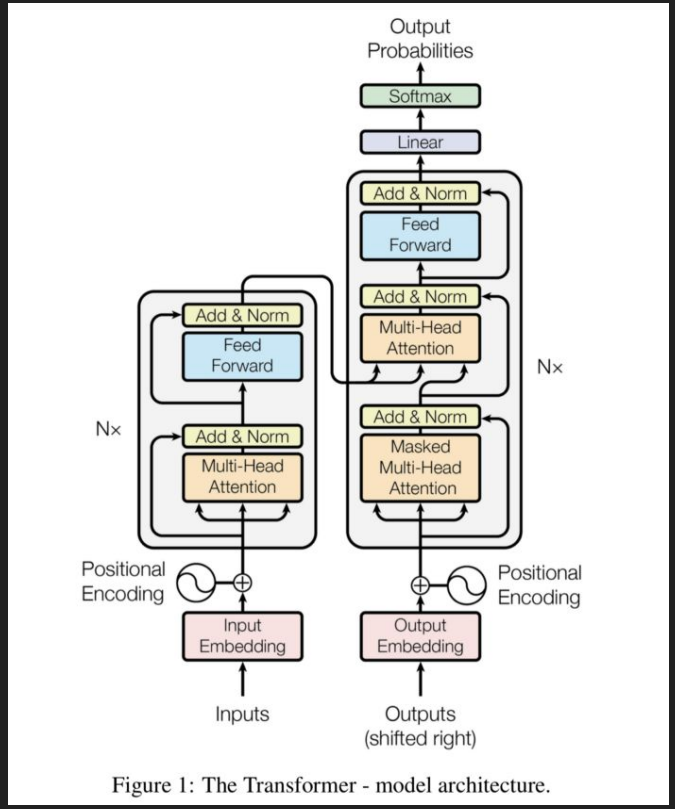
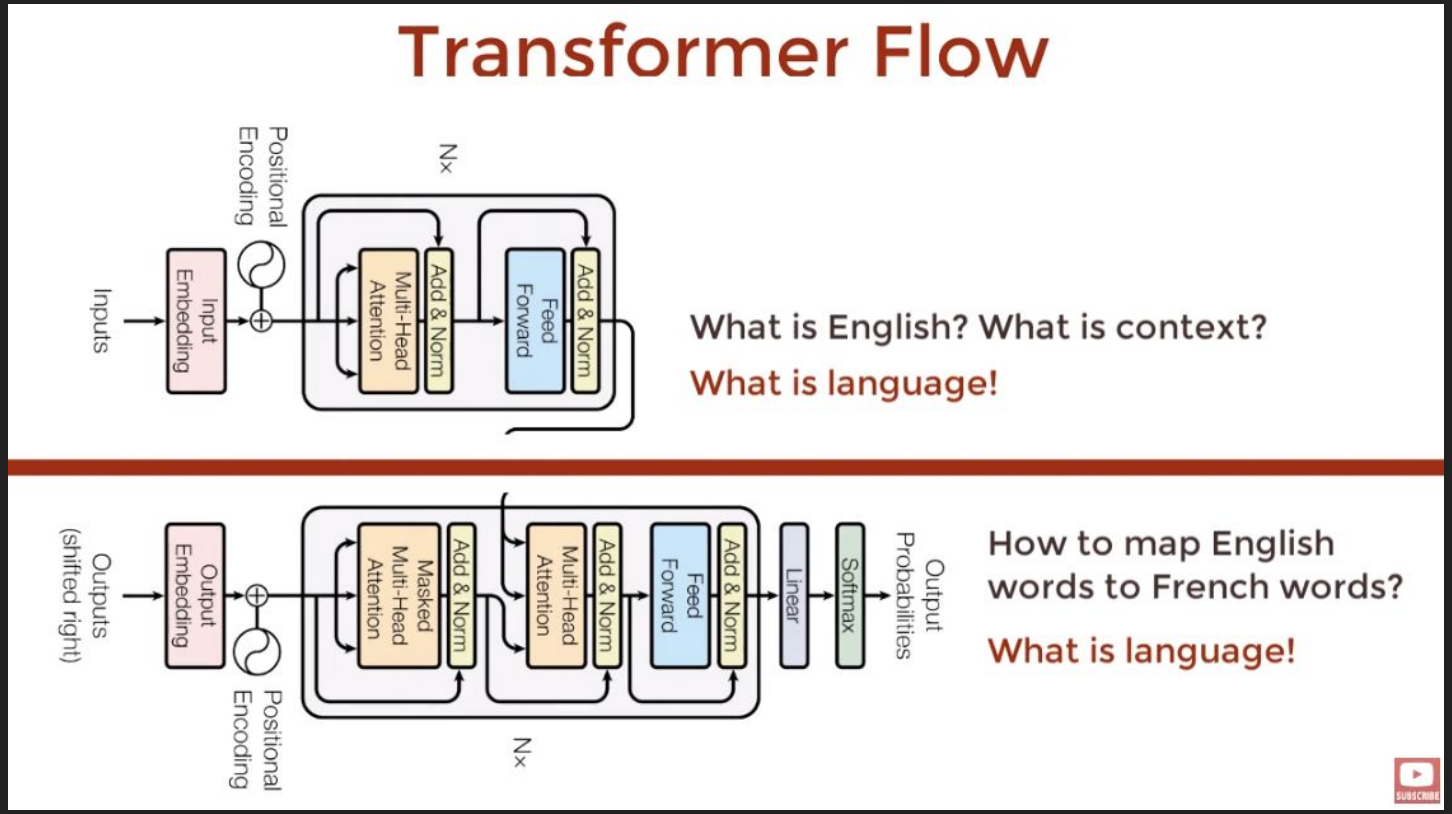


Figure 1: The Transformer - model architecture.

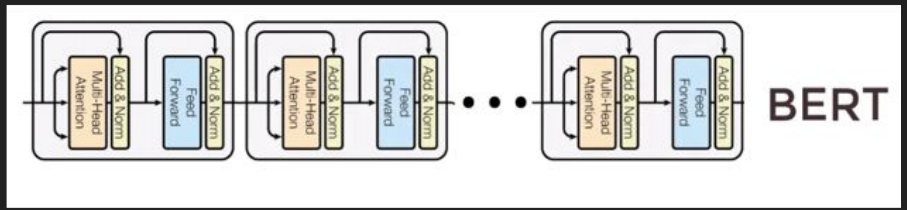
Encoder and Decoder



BERT and GPT

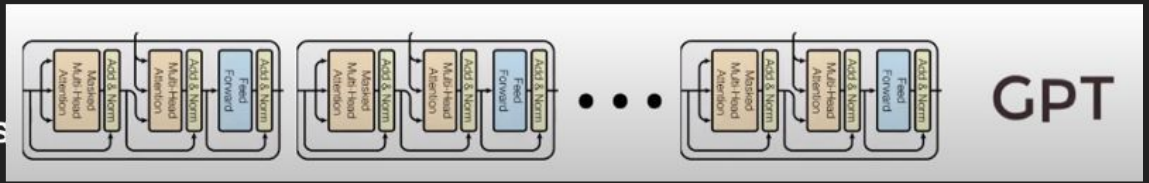
BERT (2018) by Google

A stack of Encoders



GPT (2018) by OpenAI

A stack of Decoders



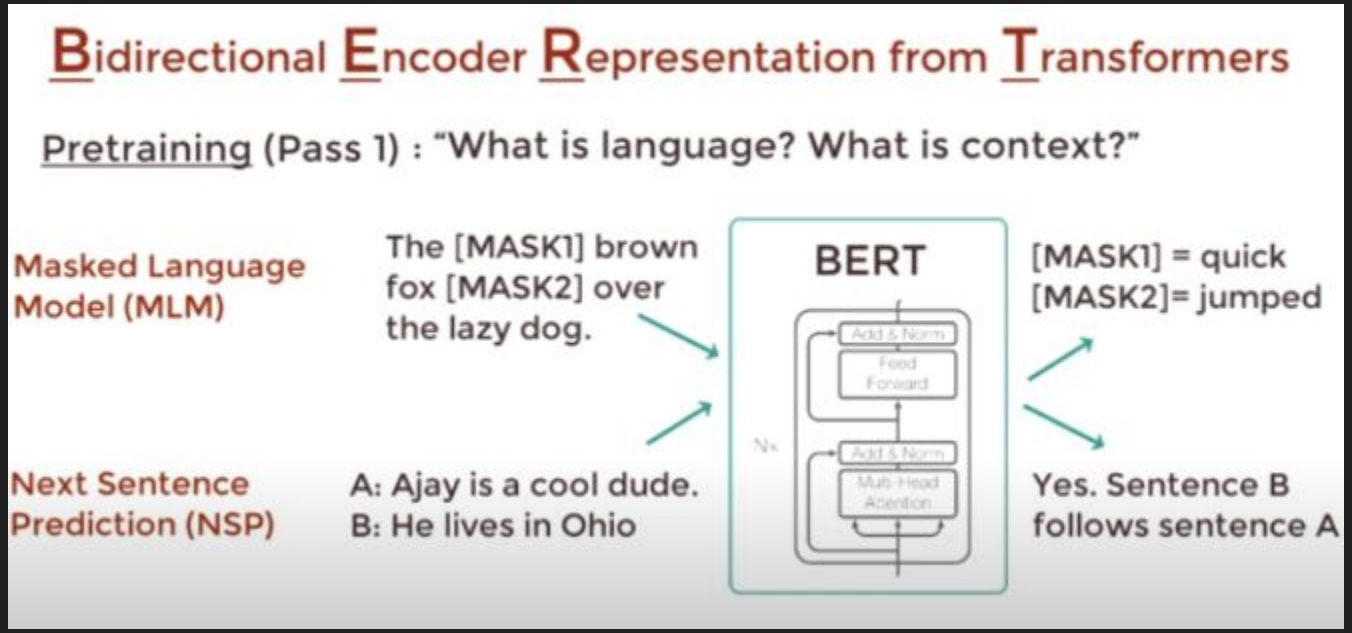
BERT(Bidirectional Encoder Representations from Transformers)

state-of-the-art performance on a number of NLU tasks

- GLUE (General Language Understanding Evaluation)
- SQuAD (Stanford Question Answering Dataset)
- SWAG (Situations With Adversarial Generations)

BERT pre-training procedure

1. Masked Language Model
2. Next Sentence Prediction



GPT-1(Generative Pre-trained Transformer)

- GPT is a "transformer" model, which uses "attention" in place of previous recurrence- and convolution-based architectures.
- It showed how a generative model of language is able to acquire world knowledge and process long-range dependencies by pre-training on a diverse corpus with long stretches of contiguous text.

GPT-2 (1.5 Billion Parameters)

Dataset for pretraining : 40 GB of text
required tens of petaflop/s-days*
translates text, answers questions, summarizes passages, and generates text output on a level that, while sometimes indistinguishable from that of humans

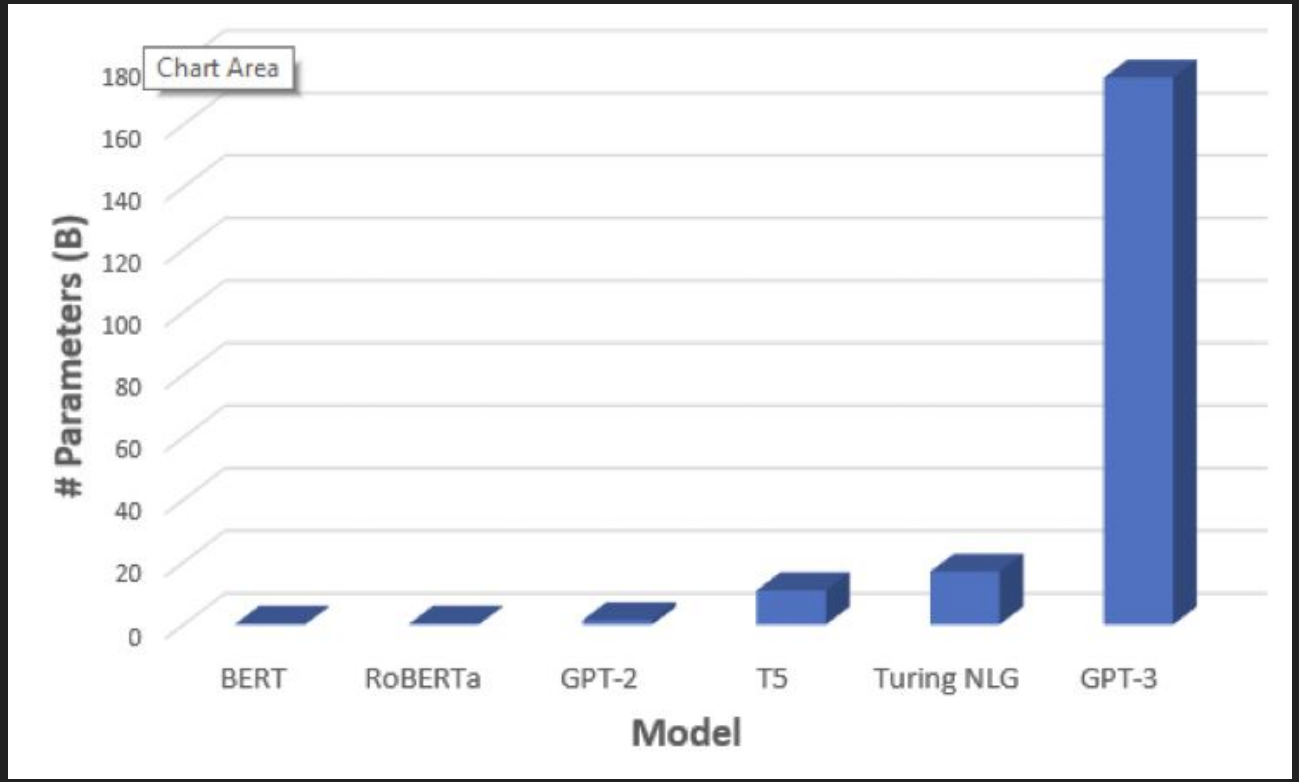
*One petaflop/s-day is approximately equal to 1020 neural net operations

GPT-3 (175 Billion Parameters)

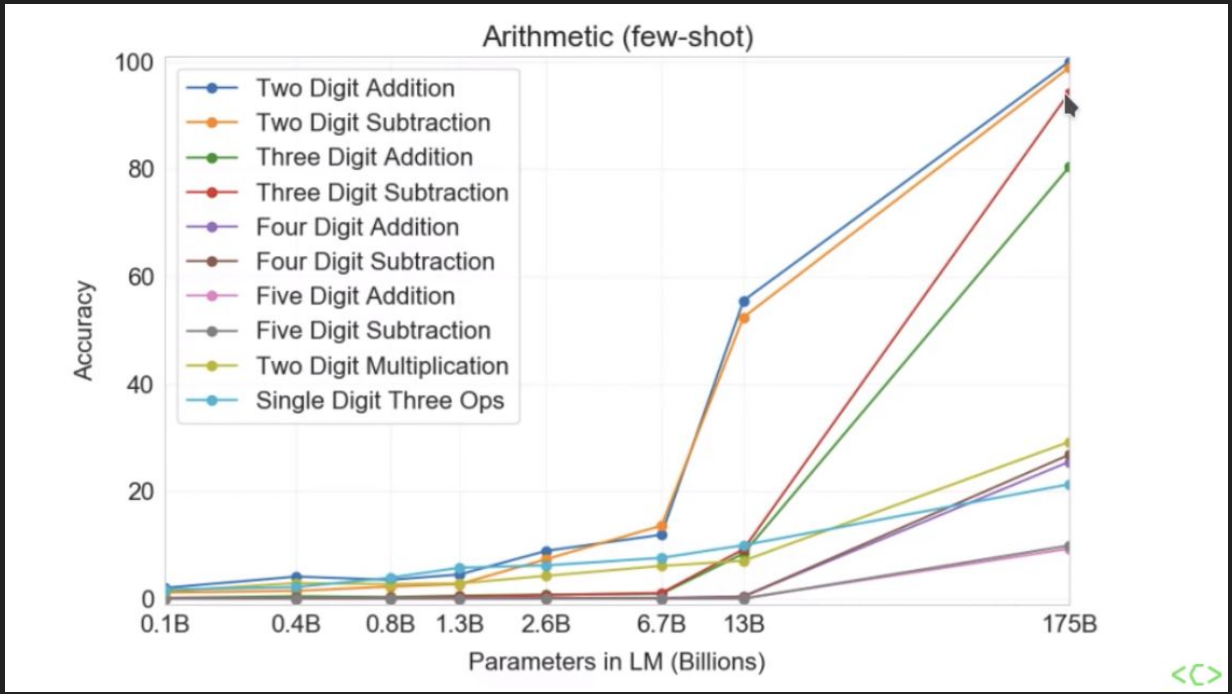
Dataset for pretraining : 570 GB of text
required several thousand petaflop/s-days*

*One petaflop/s-day is approximately equal to 1020 neural net operations

Comparison of sizes



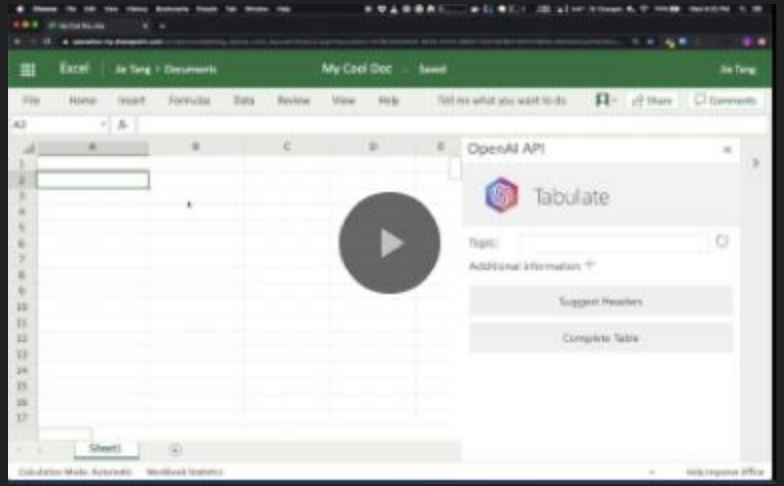
Size matters!



Some Practice Tutorials

- https://www.tensorflow.org/text/tutorials/text_generation (Tensorflow)
- https://www.tensorflow.org/text/tutorials/text_classification_rnn (Tensorflow)
- https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html (Pytorch)
- https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html (LSTM in Pytorch)

GPT-3 Demos


Excel
My Cool Doc
OpenAI API
Tabulate
Topic:
Additional information:
Suggest Headers
Complete Table
Sheet1


Wikipedia
wikipedia.org
WIKIPEDIA
The Free Encyclopedia
English
6 095 000+ articles
Español
1 604 000+ artículos
日本語
1 210 000+ 記事
Deutsch
2 441 000+ Artikel
Русский
Français

Transformers Drawbacks

- Very large models.
  - Memory and compute intensive to train
- Relatively young class of models
  - so we know less about them
- Might be worse for hierarchical data (Tran et al, ACL 2018)

Challenges

1. Transformer complexity
2. Longer sequences

Shrinking Transformer

Transformers are becoming both more accurate and larger (t5 has 11 billion parameters)
But there are ways to make them smaller without hurt performance:

1. Quantization
2. Distillation
3. Pruning
4. More specialized models

Quantization

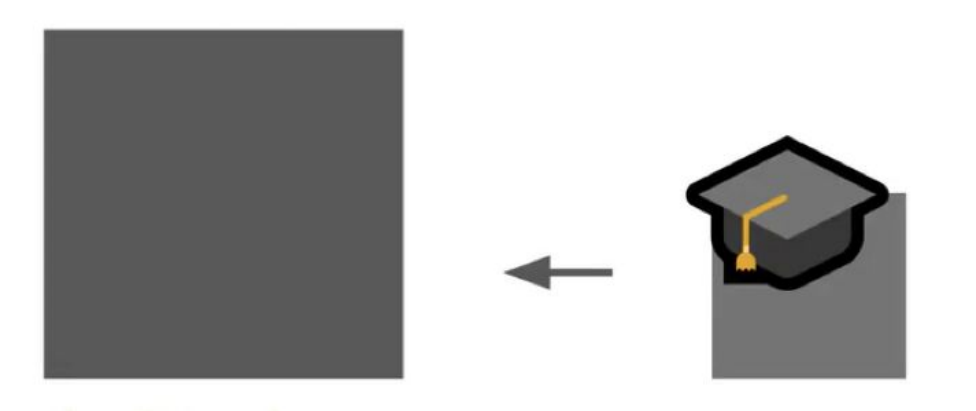
Reduced number of bit needed to store the trained parameters in model
Convert 32 bit floating point to 8 bit integer
Problem: usually hardware dependant

Distillation

A new model is trained to predict the weights of one or more layers of the larger model

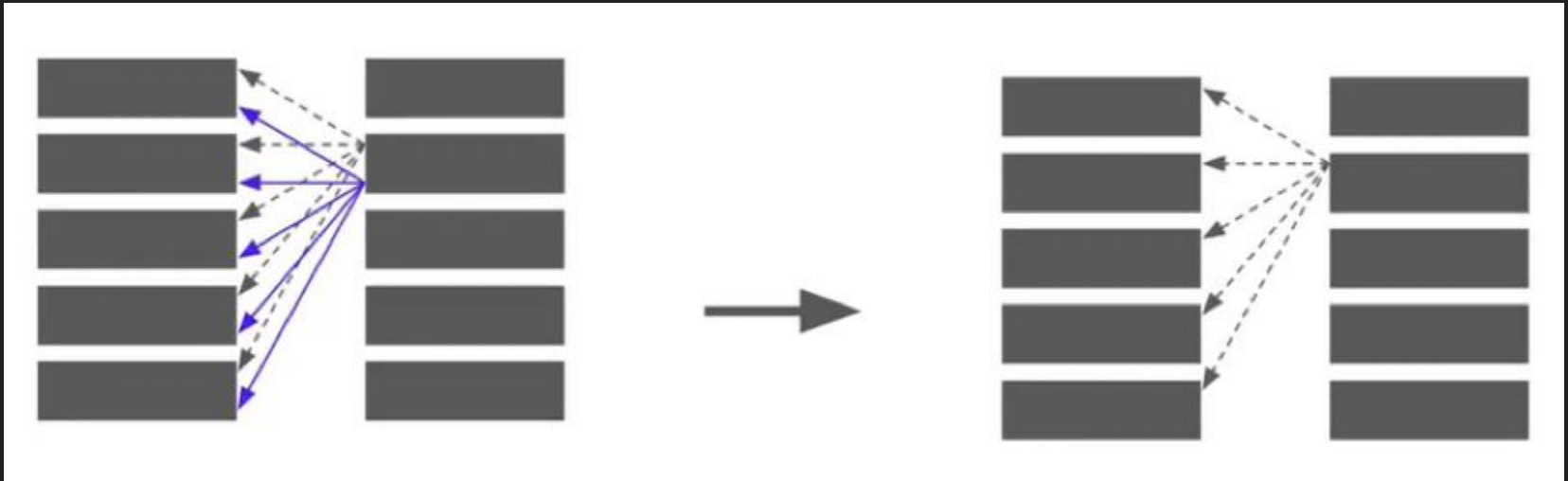
Up to 100x smaller and 15x faster

Problem: need more setup



Pruning

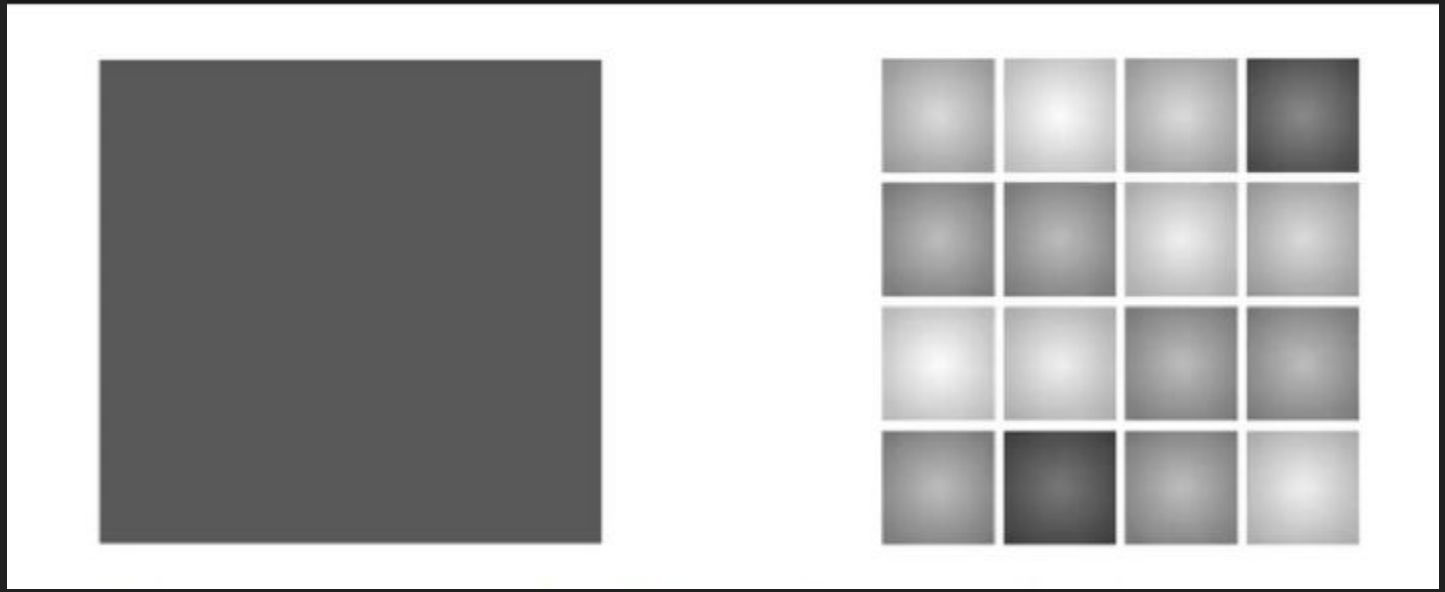
Remove attention heads based on how useful they are for a specific task
Up to 80% the heads of trained transformer heads can removed without significantly reducing accuracy



More specialized models

Train a special smaller model

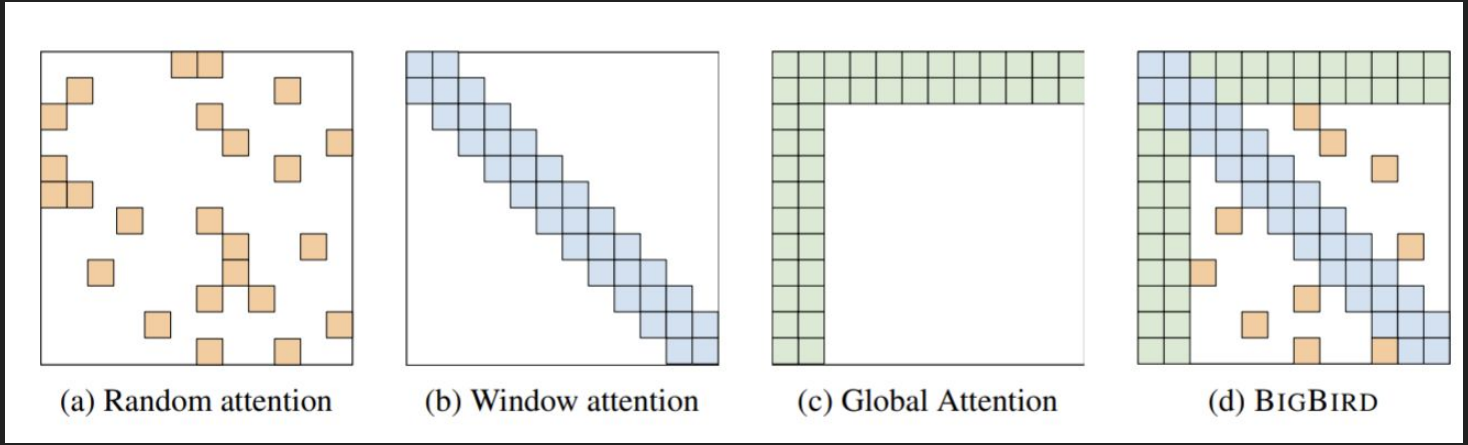
Really large nlp models (like bert & gpt) tend to be open domain



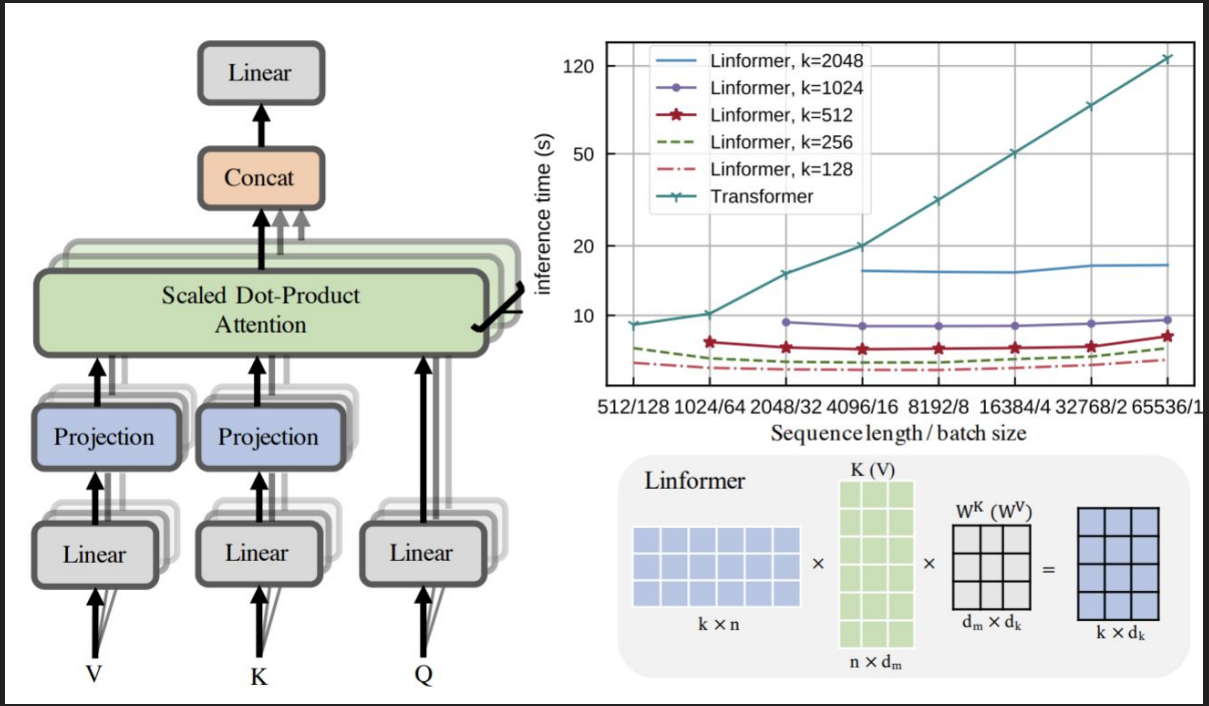
Why don’t these methods hurt performance?

1. Really large transformers are bigger than they need to be for some tasks
2. There is a lot of redundancy in these models

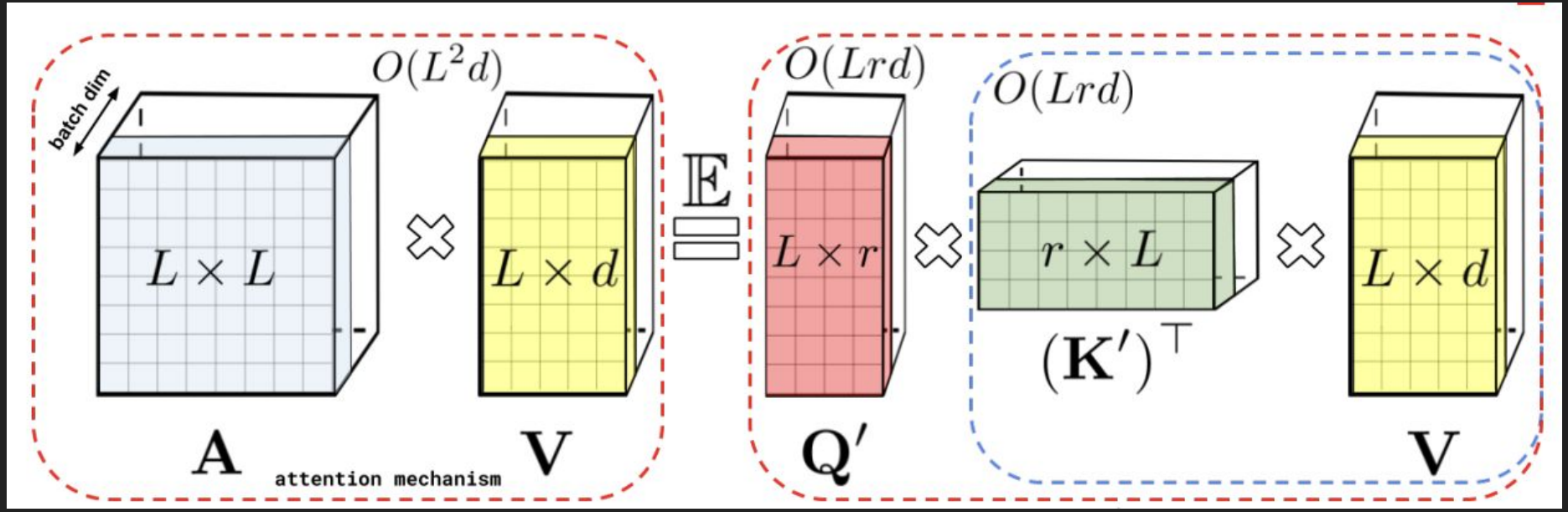
Bigbird


(a) Random attention
(b) Window attention
(c) Global Attention
(d) BIGBIRD

Linformer


Linear
Concat
Scaled Dot-Product Attention
Projection
Projection
Linear
Linear
Linear
V
K
Q
Linformer, k=2048
Linformer, k=1024
Linformer, k=512
Linformer, k=256
Linformer, k=128
Transformer
inference time (s)
120
50
20
10
512/128 1024/64 2048/32 4096/16 8192/8 16384/4 32768/2 65536/1
Sequence length / batch size
Linformer
K (V)
W^K (W^V)
k × n
n × d_m
d_m × d_k
k × d_k

Performer


batch dim
$O(L^2 d)$
$L \times L$
$L \times d$
$\mathbf{A}$
attention mechanism
$\mathbf{V}$
$\mathbb{E}$
$O(Lrd)$
$L \times r$
$\mathbf{Q}'$
$O(Lrd)$
$r \times L$
$(\mathbf{K}')^\top$
$L \times d$
$\mathbf{V}$

Join Our Groups

- Sign up for Discord (https://discord.gg/3Z5YuPqt)
- Join Deepnote (https://deepnote.com/join-team?token=af3af0284bc8497)
- Fill out our form (https://forms.gle/Fr31aFLWx8cHdtTY8)
  - Join mailing list + Github organization

- Next week: Text Analysis/NLP (Natural Language Processing)