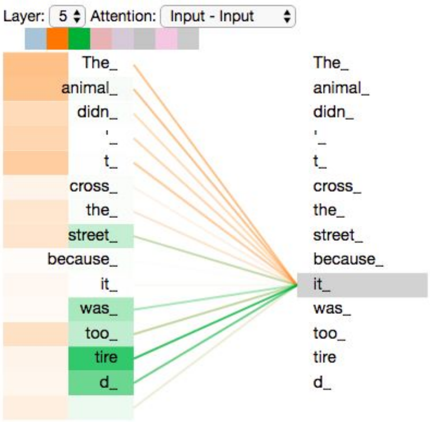
FNet

Mixing Tokens with Fourier Transform

What it’s all about

- Transformer magic lies in Self-Attention
- Self-attention is expensive
  - Quadratic cost, O(N^2) where N is the size of the sequence
  - (Remember perceiver)



What’s the attention of “it”?

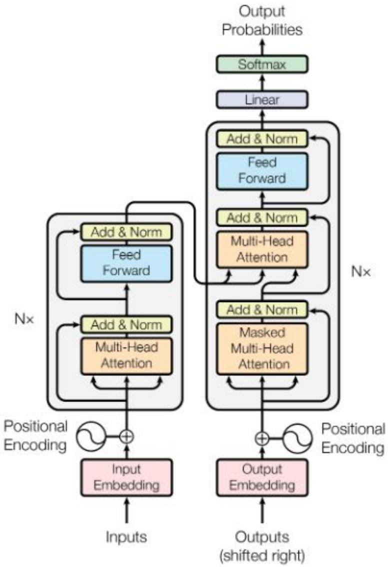
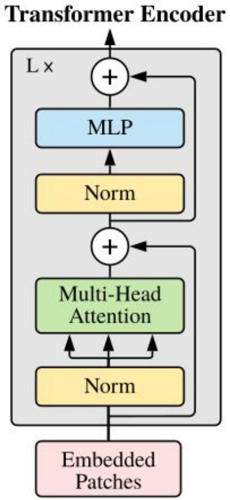


Figure 1: The Transformer - model architecture.

(from Vaswani et. al Attention is all you need)

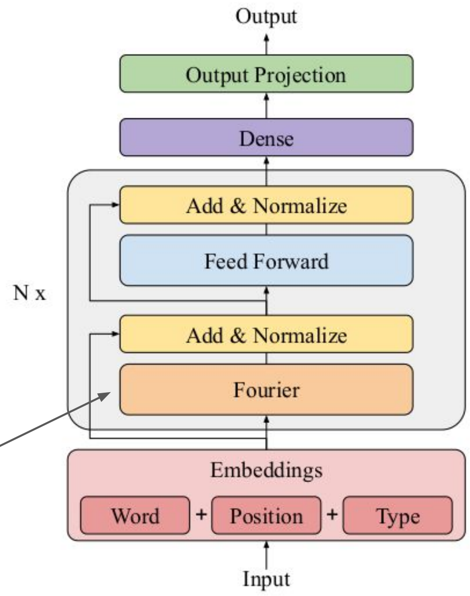


(from Dosovitskiy et. al An image is worth 16x16 words.)

What it’s all about

- Transformer magic lies in Self-Attention
- Self-attention is expensive
  - Quadratic cost, O(N^2) where N is the size of the sequence
  - (Remember perceiver)
- Can we do better?
  - Kind of

Attention?

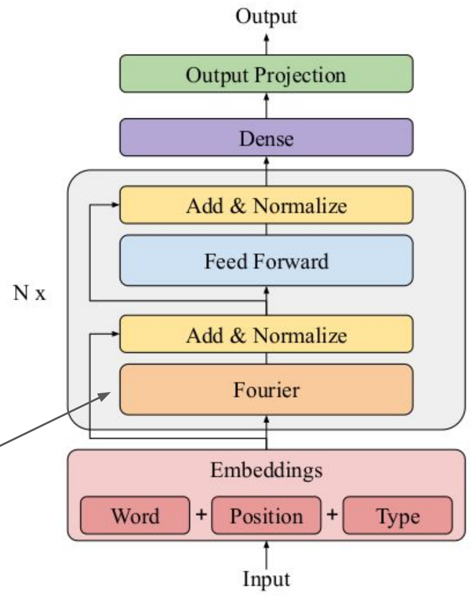


FNet

Fourier Transform? (I)

- Where is the learning part?
  - No parameters
- Does it work?
  - Yes
- Is it more efficient?
  - A lot! (obviously, no attention cost)
- Is FT all we need?
  - No

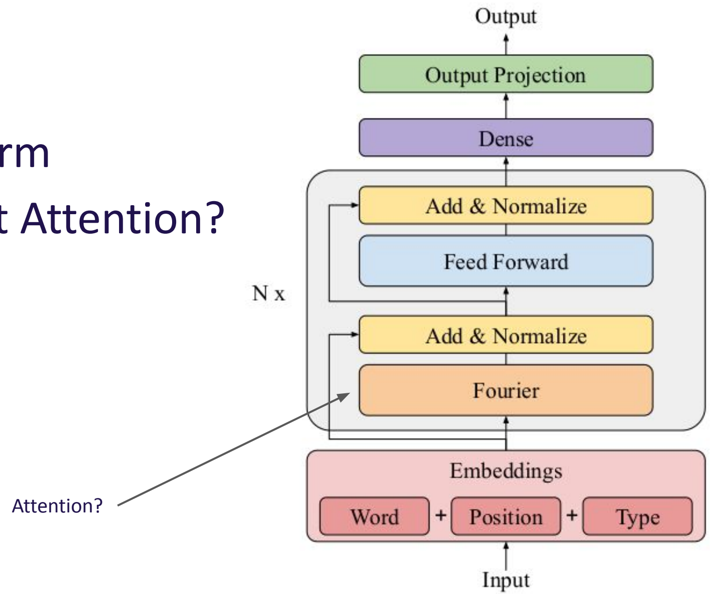
Attention?



FNet

Fourier Transform? (II)

- Fourier transform works better
  - but that's not the point, probably
- Fast implementations of Fourier Transform
- The real question is: Can we live without Attention?
  - We'll see



FNet

What's in the box?! (Fourier Version)

- How does this translates to our problem?
  - We don't have signals as input
- In this case we have a sequence of tokens
  - and tokens have embeddings
- 2 Fourier transforms.
  - One sequence wise
  - One along the hidden dimensions

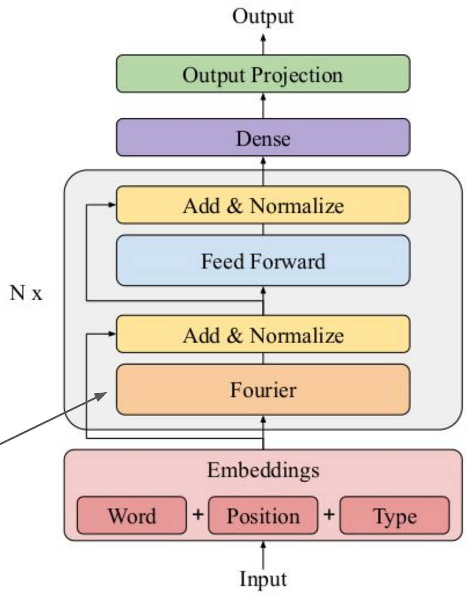
$$y = \Re\left(\mathcal{F}_{\text{seq}}\left(\mathcal{F}_{\text{h}}(x)\right)\right).$$

Input

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - \cancel{i \cdot \sin\left(\frac{2\pi}{N}kn\right)}\right], \qquad \textbf{(Eq.1)}$$

Discrete fourier transform reminder. (We drop the imaginary part)

Attention?

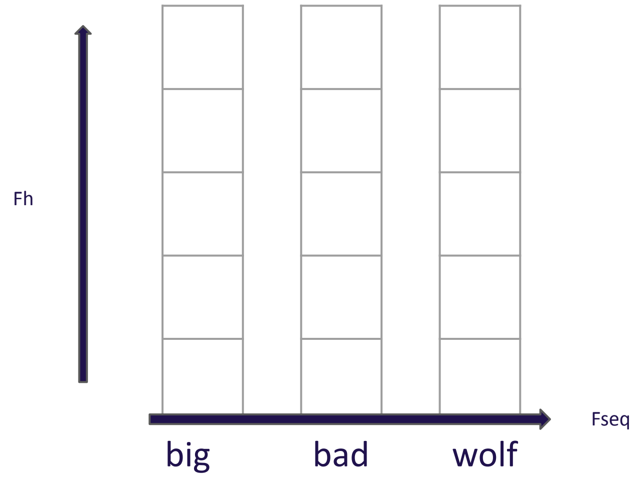


FNet

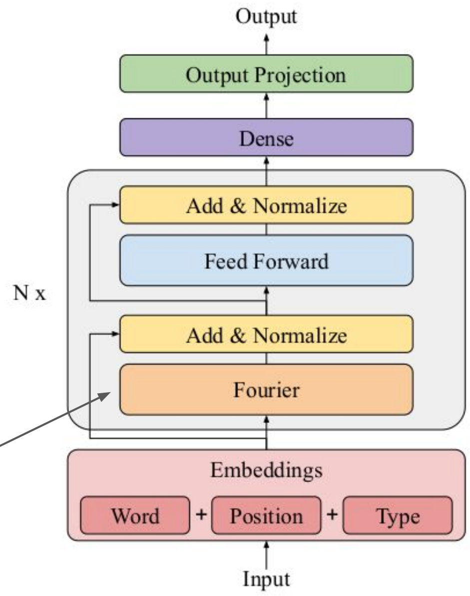
What's in the box?! (Fourier Version)

- 2 Fourier transforms.
  - One sequence wise
  - One along the hidden dimensions

$$y = \Re\left(\mathcal{F}_{\text{seq}}\left(\mathcal{F}_{\text{h}}(x)\right)\right).$$



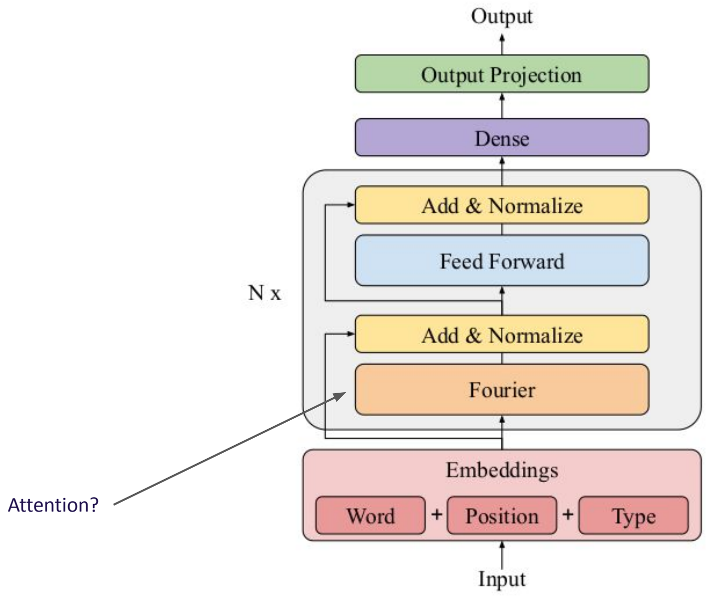
Attention?



FNet

Token mixing

- The success of the method is most likely caused by the mixing of tokens.
- Attention is a fancier way of doing that by learning what matters for what.



FNet

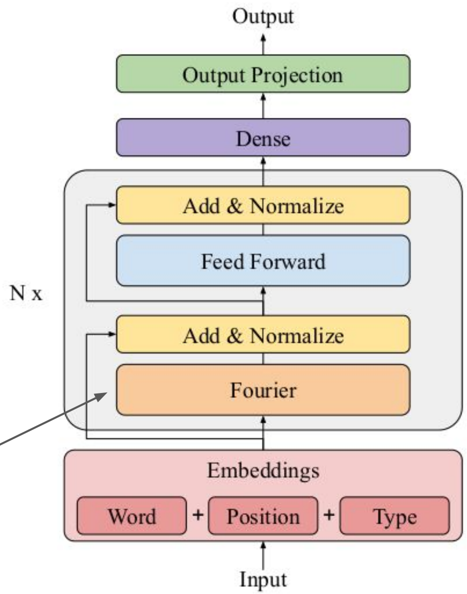
Token mixing - Other flavours

- Linear encoder: we replace each self-attention sublayer with a two learnable, dense, linear sublayers, one applied to the hidden dimension and one applied to the sequence dimension.
- Random encoder: we replace each self-attention sublayer with a two constant random matrices, one applied to the hidden dimension and one applied to the sequence dimension.
- Feed Forward-only (FF-only) encoder: we remove the self-attention sublayer from the Transformer layers; this model has no token mixing.

Hey bro!

MLP-Mixer: An all-MLP Architecture for Vision

Attention?



FNet

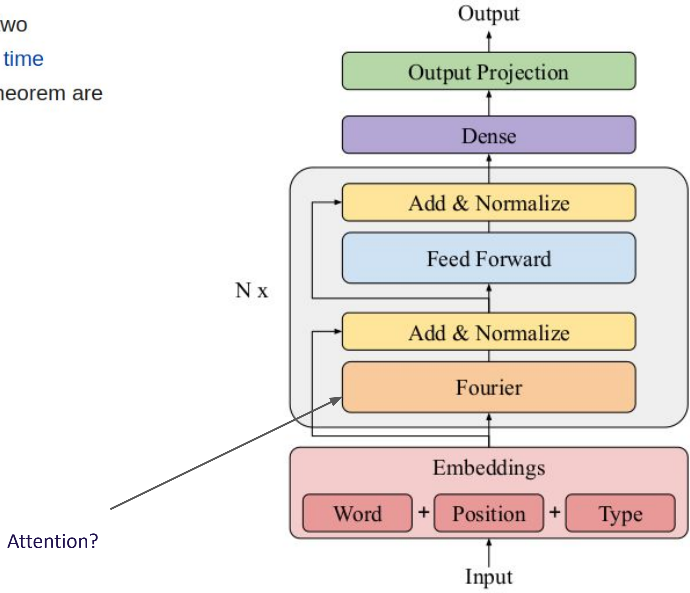
Another all MLP network?

Convolution theorem

From Wikipedia, the free encyclopedia

In mathematics, the **convolution theorem** states that under suitable conditions the Fourier transform of a convolution of two functions (or signals) is the pointwise product of their Fourier transforms. More generally, convolution in one domain (e.g., time domain) equals point-wise multiplication in the other domain (e.g., frequency domain). Other versions of the convolution theorem are applicable to various Fourier-related transforms.

- In our case we constantly move from one domain to another in every layer →
- Convolutions kind of exist in FNet



FNet

Experiments

Table 1: GLUE Validation results on TPUs, after finetuning on respective tasks. We report the mean of accuracy and F1 scores for QQP and MRPC, Spearman correlations for STS-B and accuracy scores for all other tasks. The MNLI metrics are reported by the match/mismatch splits. Average scores exclude any failure cases. After controlling for batch size and training steps, the GPU metrics (not shown) are similar.

Model	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
BERT-Base	**84/81**	**87**	**91**	93	73	**89**	**83**	**69**	**83.3**
Linear-Base	74/75	84	80	94	67	67	83	69	77.0
FNet-Base	72/73	83	80	**95**	69	79	76	63	76.7
Random-Base	51/50	70	61	76	67	4	73	57	56.6
FF-only-Base	34/35	31	52	48	67	FAIL	73	54	49.3
FNet-Hybrid-Base	78/79	85	88	94	**76**	86	79	60	80.6
BERT-Large	**88/88**	**88**	**92**	**95**	71	**88**	86	66	**84.7**
Linear-Large	35/36	84	80	79	67	24	73	60	59.8
FNet-Large	78/76	85	85	94	**78**	84	**88**	**69**	81.9

Bye