FNet: Mixing Tokens with Fourier Transforms - James Lee-Thorp et al. (Google Research)

Presentation Author
Mathieu Ravaut

Nanyang Technological University

Table of content

Introduction

- The **Transformer** architecture (Vaswani et al. [2017]) has become ubiquitous.
  - Initially designed in Machine Translation.
  - SOTA on many NLP tasks.
  - Now SOTA in tasks beyond NLP (vision, speech).
- It all relies on the **self-attention** mechanism.

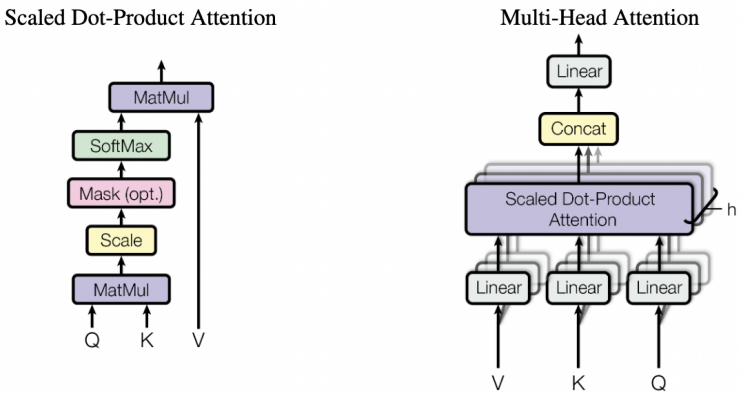

FIGURE – Multi-head scaled dot-product attention introduced in the Transformer.

Introduction

- Limits of Transformer-based architectures include :
  - Scaling to **long input sequences** (crucial in summarization).
  - **Training time**. Several days on multiple GPUs to pre-train models like BERT (Devlin et al. [2018]).
  - **Inference time**.
  - **Memory** footprint. Needs GPUs with enough RAM.
  - Relative **complexity** of the underlying self-attention mechanism.

Introduction

- FNet replaces the attention sublayer by a **Fourier transformation**, which is linear.
- Excellent compromise :
  - Almost all the accuracy is preserved
  - Scales very well to long input sequences
  - Faster at training and inference on GPUs, and on TPUs for shorter sequences
  - Lighter memory footprint

Model

Model :

- Background on Fourier transform
- FNet
- Implementation

Fourier transform

The Fourier transform (Joseph Fourier, 1822) decomposes a signal into its frequencies.

- Let $\omega = e^{-i\frac{2\pi}{n}}$.

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \xrightarrow{y_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} x_k \omega^{jk}} \text{Discrete Fourier Transform} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

- In matrix form,

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} a_0 + ib_0 \\ a_1 + ib_1 \\ a_2 + ib_2 \\ \vdots \\ a_{n-1} + ib_{n-1} \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \cdots & \omega^{n-1} \\ \omega^0 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \omega^0 & \omega^3 & \omega^6 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

FIGURE – Discrete Fourier Transform (DFT). Source : CE7453

Fourier transform

2 ways to compute the DFT :

- Matrix-vector multiplication. **Complexity :** $\mathcal{O}(n^2)$.
- **Fast-Fourier Transformation** (FFT) :

$$F_{2n}[x_0, x_1, x_2, x_3, \cdots, x_{2n-2}, x_{2n-1}]_j = F_n[x_0, x_2, \cdots, x_{2n-2}]_j + e^{-i\frac{j\pi}{n}} F_n[x_1, x_3, \cdots, x_{2n-1}]_j$$

$$F_{2n}[x_0, x_1, x_2, x_3, \cdots, x_{2n-2}, x_{2n-1}]_{n+j} = F_n[x_0, x_2, \cdots, x_{2n-2}]_j - e^{-i\frac{j\pi}{n}} F_n[x_1, x_3, \cdots, x_{2n-1}]_j$$

FIGURE – Fast Fourier Transform (DFT). Source : CE7453

Complexity : $\mathcal{O}(n \log(n))$.

FNet

FNet architecture :

- Replace the attention sublayer with a 2D Fourier transform on the (sequence length, hidden dimension) input.
- Apply 2 consecutive 1D Fourier transforms.
- Take the real part, **after applying both Fourier transforms**.

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

FIGURE – Fourier transform module in FNet.

FNet

How to apply the 2 consecutive 1D DFT :

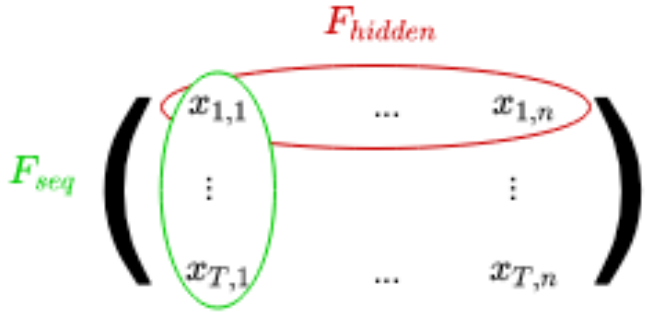

FIGURE – FNet consecutive 1D DFTs.

FNet

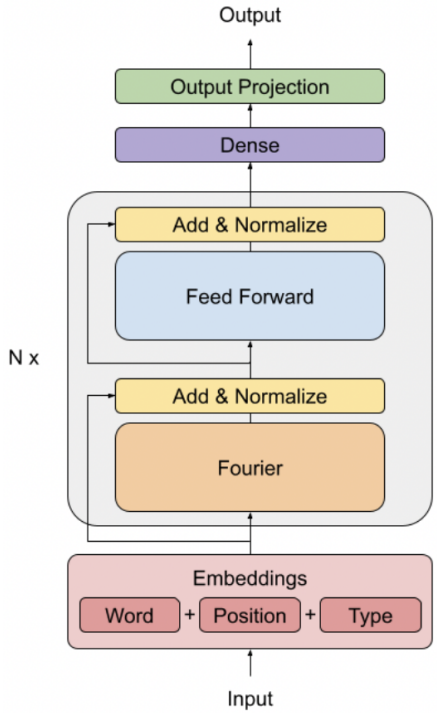

FIGURE – FNet architecture.

FNet

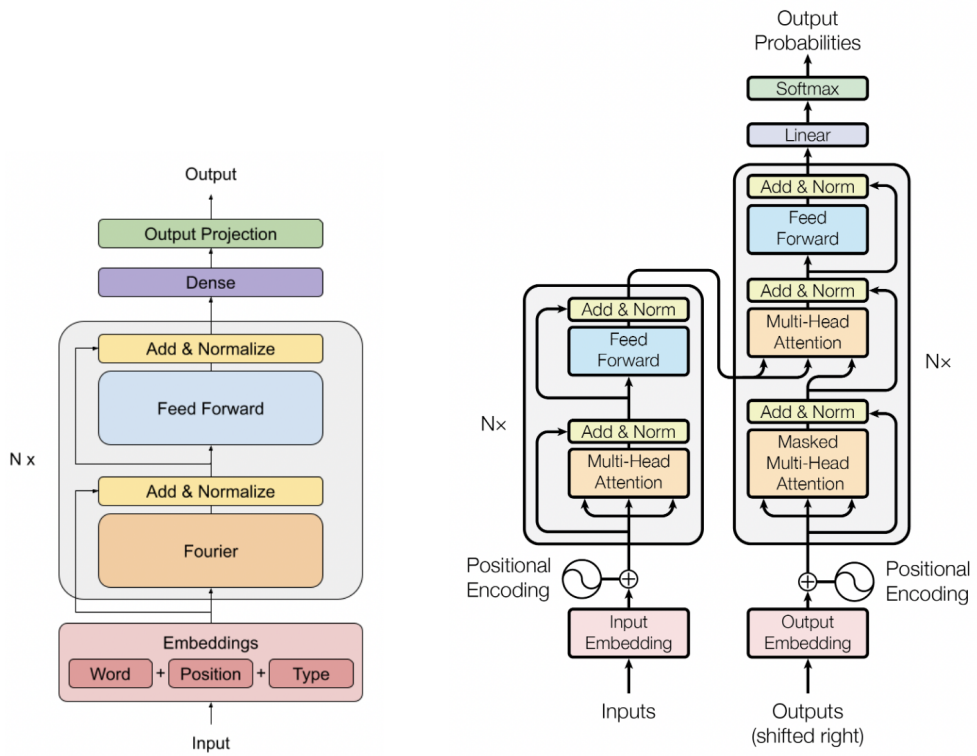

FIGURE – Left : FNet. Right : original Transformer.

Implementation

In practice, the way FNet computes the DFT depends on the hardware :

- On GPUs : always use FFT as it is faster
- On TPUs (very optimized for matrix multiplication) :
  - For sequence lengths ≤ 8192 : pre-compute the DFT matrix, then do matrix multiplication.
  - For sequence lengths ≥ 8192 : FFT.

Results

Results :

- Baselines
- Pre-training
- Fine-tuning
- Long input sequences
- Speed
- Memory footprint

Baselines

The authors compare FNet to BERT and to some other baseline models :

- **Linear** : replace attention sublayer with 2 Fully Connected layers.
- **Random** : replace attention sublayer with 2 constant random matrices.
- **FF** : remove the attention sublayer (and don't relace it).
- **Hybrid** : replace the final 2 Fourier sublayers with attention sublayers.

Baselines

FNet follows the BERT-base and BERT-large architectures.

Model	Mixing layer ops (forward pass)	Learnable parameters (millions)		
		Mixing layer	Encoder	Pre-training
BERT-Base	$nd_{hidden}\,(2n + 4d_{hidden})$	2.36	111	136
Linear-Base	$nd_{hidden}\,(n + d_{hidden})$	0.85	93	118
FNet-Base (matmul)	$nd_{hidden}\,(n + d_{hidden})$	0	83	108
FNet-Base (FFT)	$nd_{hidden}\,(\log(n) + \log(d_{hidden}))$	0	83	108
Random-Base	$nd_{hidden}\,(n + d_{hidden})$	0	83	108
FF-only-Base	0	0	83	108

FIGURE – Comparison of the ops and the number of parameters for BERT and FNet, and different baselines.

- FNet has much lower number of ops.
- Same number of parameters.

Baselines

Name	Dimensions			Parameters (millions)		
	d_{hidden}	d_{ff}	Layers	BERT	Linear	FNet
Large	1024	4096	24	372	302	271
Base	768	3072	12	136	118	108
	512	2048	12	72	72	72
	512	2048	8	59	59	59
"Mini"	512	2048	4	46	46	46
	256	1024	4	20	20	20
"Micro"	256	1024	2	18	18	18
	128	512	2	9	9	9

FIGURE – Different model sizes for each architecture.

Pre-training

Model	Loss			Accuracy	
	Total	MLM	NSP	MLM	NSP
BERT-Base	**1.76**	**1.48**	**0.28**	**0.68**	**0.86**
Linear-Base	2.12	1.78	0.35	0.62	0.83
FNet-Base	2.45	2.06	0.40	0.58	0.80
Random-Base	5.02	4.48	0.55	0.26	0.70
FF-only-Base	7.54	6.85	0.69	0.13	0.50
FNet-Hybrid-Base	2.13	1.79	0.34	0.63	0.84
BERT-Large	**1.49**	**1.23**	**0.25**	**0.72**	**0.88**
Linear-Large	1.91	1.60	0.31	0.65	0.85
FNet-Large	2.11	1.75	0.36	0.63	0.82

FIGURE – Pre-training results : validation loss and accuracy on Masked Language Modeling and Next Sentence Prediction for BERT and FNet.

- FNet reaches slightly lower accuracy on both tasks.

Fine-tuning

Model	MNLI (m/mm)	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[4]
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

FIGURE – Fine-tuning results on GLUE.

- FNet-base is 6.6 points behind BERT-base
- FNet-Hybrid-base 2.6 points behind.
- FNet-large 2.8 points behind BERT-large.

Long-range

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.
Transformer	36.06	61.54	59.67	41.51	80.38	FAIL	55.83
Linear	33.75	53.35	58.95	41.04	83.69	FAIL	54.16
FNet	35.33	65.11	59.61	38.67	77.80	FAIL	55.30

FIGURE – Accuracy results on the Long-Range Arena benchmark (Tay et al. [2020]).

- FNet has almost no performance drop compared to Transformer.
- The Linear baseline is very strong !

Speed

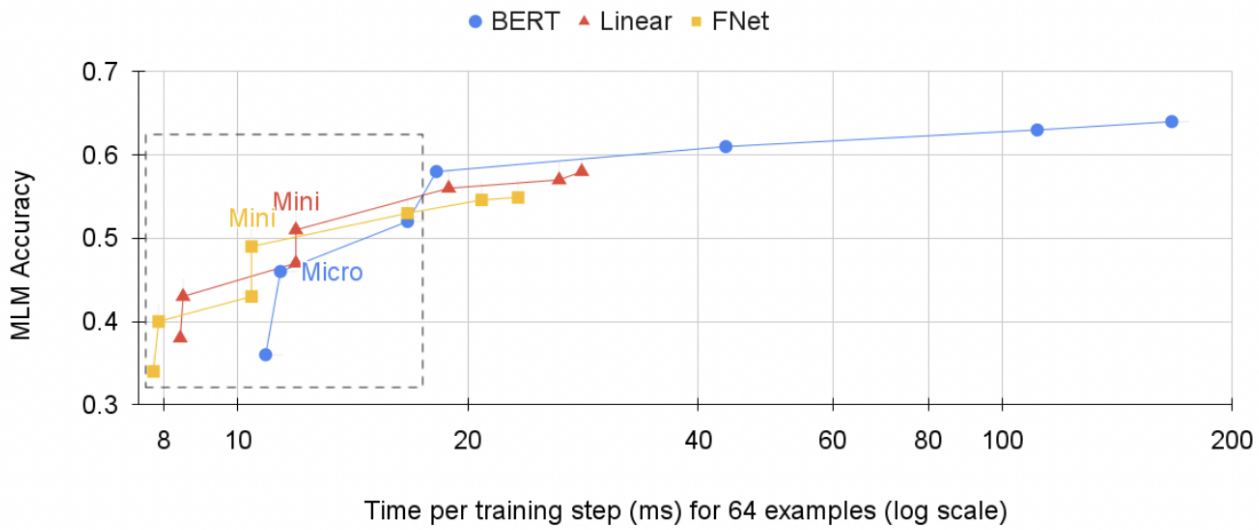

FIGURE – Pre-training speed on C4 (Raffel et al. [2019]) on GPU.

- FNet trains 7 times faster than BERT (see first point from the right).

Speed

Seq. length	512	1024	2048	4096	8192	16386
			GPU			
Transformer	26	11	4	FAIL	FAIL	
Linear	49 (1.9x)	23 (2.0x)	11 (2.6x)	4	FAIL	
FNet (FFT)	**60 (2.3x)**	**30 (2.7x)**	**16 (3.9x)**	**8**	**4**	
Performer	32 (1.3x)	19 (1.6x)	10 (2.3x)	5	2	
			TPU			
Transformer	43	16	5	1	FAIL	FAIL
Linear	78 (1.8x)	**62 (3.8x)**	**28 (5.7x)**	**12 (9.9x)**	4	FAIL
FNet (matmul)	**92 (2.1x)**	61 (3.8x)	26 (5.4x)	11 (8.8x)	4	1
FNet (FFT)	36 (0.8x)	25 (1.5x)	13 (2.7x)	7 (5.4x)	3	1
Performer	59 (1.4x)	42 (2.6x)	23 (4.6x)	**12 (9.9x)**	**6**	**3**

FIGURE – Training speed on long-range Text Classification on GPU and TPU.

- 2-4 speedup on GPUs, better than all other models including Performer.

Speed

Seq. length	512	1024	2048	4096	8192	16386
	GPU					
Transformer	5.8	21.3	67.6	227.3	FAIL	FAIL
Linear	5.4 (1.1x)	6.1 (3.5x)	22.6 (3.0x)	63.7 (3.6x)	181.8	FAIL
FNet (FFT)	**5.3 (1.1x)**	**5.2 (4.1x)**	**15.6 (4.3x)**	**35.8 (6.3x)**	**74.6**	**147.1**
Performer	5.8 (1.0x)	10.9 (2.0x)	24.8 (2.7x)	53.5 (4.3x)	109.9	227.3
	TPU					
Transformer	4.6	7.3	34.6	126.6	476.2	FAIL
Linear	5.1 (0.9x)	5.5 (1.3x)	**4.8 (7.2x)**	**14.6 (8.7x)**	49.5 (9.6x)	FAIL
FNet (matmul)	**4.7 (1.0x)**	**5.3 (1.4x)**	9.3 (3.7x)	35.0 (3.6x)	131.6 (3.6x)	454.5
FNet (FFT)	5.7 (0.8x)	11.8 (0.6x)	28.6 (1.2x)	58.8 (2.2x)	119.0 (4.0x)	232.6
Performer	5.1 (0.9x)	5.4 (1.3x)	6.2 (5.6x)	18.5 (6.8x)	**26.5 (18.0x)**	**55.9**

FIGURE – Inference speed on long-range Text Classification on GPU and TPU.

- Speedup increasing with sequence length for FNet with FFT.
- Scales to extremely long input sequences (16k !).

Memory footprint

Seq. length	512	1024	2048	4096	8192	16386
	GPU					
Transformer	1.6	4.0	12.2	FAIL	FAIL	
Linear	0.9	1.6	2.8	6.9	FAIL	
FNet (FFT)	**0.8**	**1.3**	**2.2**	**3.9**	**7.4**	
Performer	1.1	1.9	3.1	5.5	10.4	
	TPU					
Transformer	1.1	2.1	5.8	9.1	FAIL	FAIL
Linear	0.9	1.1	1.9	4.9	14.8	FAIL
FNet (matmul)	**0.8**	**0.9**	**1.3**	2.2	4.8	11.9
FNet (FFT)	**0.8**	**0.9**	**1.3**	**2.0**	**3.5**	**6.3**
Performer	1.0	1.3	1.8	3.0	5.1	9.6

FIGURE – Peak memory usage in GB when training on long-range Text Classification on GPU and TPU.

- Training on 8k-long input sequences still fits into a standard, 11GB GPU.

Conclusion

Key takeaways :

- Linear token-mixing transformations + fully-connected layers with nonlinearities are sufficient to model diverse semantic relationships in text. **Questions the universal, unquestioned use of the self-attention sublayer**.
- Using Fourier transform, FNet achieves 92% of BERT's accuracy on GLUE while training 7 times faster.
- Almost no accuracy reduction on Long Range Arena.
- FNet can scale to 16k-long input sequences.
- Memory footprint is dramatically reduced (almost 6 times with 2048-long sequences).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv :1910.10683*, 2019.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. *arXiv preprint arXiv :2011.04006*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia

Polosukhin. Attention is all you need. *arXiv preprint arXiv :1706.03762*, 2017.