NLP: Many Tasks / A Single Model

- Language Modeling Objectives (Masked / Autoregressive), often pretraining
- Classification (often fine-tuning)
- Generation: Translation, Summarization, Question Answering
- Short vs Long Inputs
- Multilinguality and X-Transfer

Architecture name *former

NLP: Many Tasks / A Single Model

- The past: LSTM-RNN, Convolutions (e.g. ByteNet, ConvSeq2Seq, SliceNet).
- Transformers are the default architecture.
- Attention & Self-Attention: Inductive Bias that tokens should dynamically interact. Query, Key, Value terminology = **content retrieval**
- Attention & Self-Attention: Do not need to wait for information to propagate.

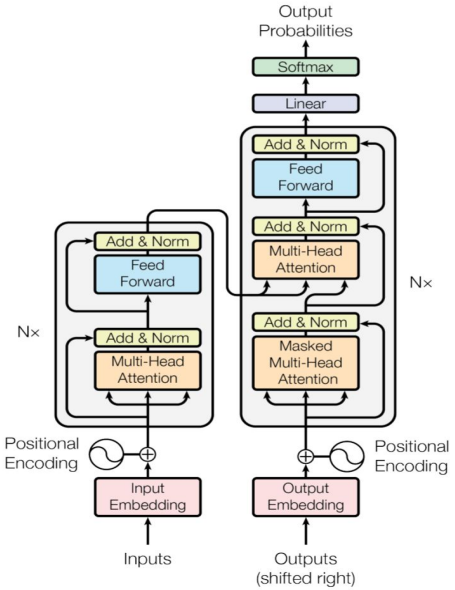


Figure 1: The Transformer - model architecture.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

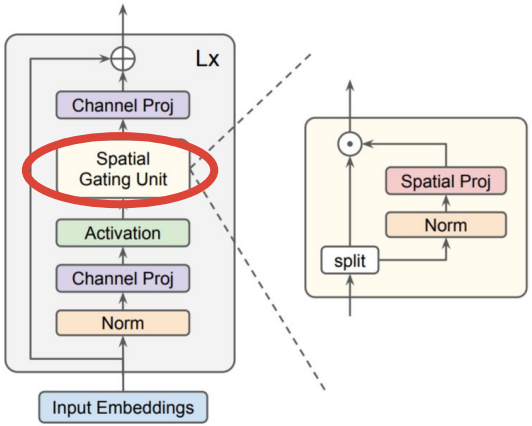
NLP: Many Tasks / A Single Model

- Let us start pessimistically about architecture papers…
- Many modifications to Transformers reported to outperform the original one, Either hard to reproduce or do not transfer across tasks. LINK

The research community has proposed copious modifications to the Transformer architecture since it was introduced over three years ago, relatively few of which have seen widespread adoption. In this paper, we comprehensively evaluate many of these modifications in a shared experimental setting that covers most of the common uses of the Transformer in natural language processing. Surprisingly, we find that most modifications do not meaningfully improve performance. Furthermore, most of the Transformer variants we found beneficial were either developed in the same codebase that we used or are relatively minor changes. We conjecture that performance improvements may strongly depend on implementation details and correspondingly make some recommendations for improving the generality of experimental results.

Pay Attention to MLPs [link]

- They propose gMLPs and show that it can match Transformers on some language and vision applications.
- NLP: On BERT pre-training they achieve parity on perplexity.
- NLP: On BERT fine-tuning worse; they investigate ideas to close the gap.
- gMLP = gating + MLP



Pseudo-code for the gMLP block

```
def gmlp_block(x, d_model, d_ffn):
  shortcut = x
  x = norm(x, axis="channel")
  x = proj(x, d_ffn, axis="channel")
  x = gelu(x)
  x = spatial_gating_unit(x)
  x = proj(x, d_model, axis="channel")
  return x + shortcut

def spatial_gating_unit(x):
  u, v = split(x, axis="channel")
  v = norm(v, axis="channel")
  n = get_dim(v, axis="spatial")
  v = proj(v, n, axis="spatial", init_bias=1)
  return u * v
```

Figure 1: Overview of the gMLP architecture with Spatial Gating Unit (SGU). The model consists of a stack of L blocks with identical structure and size. All projection operations are linear and "$\odot$" refers to element-wise multiplication (linear gating). The input and output protocols follow BERT for NLP and ViT for vision. Unlike Transformers, gMLPs do not require positional encodings, nor is it necessary to mask out the paddings during NLP finetuning.

Pay Attention to MLPs [LINK]

- Key ingredient the spatial gating unit **SGU**.
- Mixer would suggest using a spatial projection with position biases:

$$f_{W,b}(Z) = WZ + b$$

- Attention is a **3rd order** interaction: q k v; this would be **1st order**.
- They make it **2nd order**:

$$s(Z) = Z \odot f_{W,b}(Z)$$

- However a special second order by pointwise multiplication: z z.
- Better to use linear projections to create two Z's

$$s(Z) = Z_1 \odot f_{W,b}(Z_2)$$

- Strangely f needs to be initialized to a constant function: f(z) = ones_like(z)

Pay Attention to MLPs [LINK]

- Ablation study on BERT pre-training. (stdev is 0.01)
- gMLPs need bigger Feed-Forwards and need to be deeper
- BERT is L=24 and hidden=768
- Learned W is close to a 1-d convolution with kernel-width = sequence length

Table 3: MLM validation perplexities of Transformer baselines and four versions of gMLPs. f refers to the spatial linear projection in Equation (2) with input normalization. The MLP-Mixer baseline model has L=24 layers with d_{model}=768, d_{spatial}=384 and d_{ffn}=3072. Each gMLP model has L=36 layers with d_{model}=512 and d_{ffn} = 3072. No positional encodings are used for Mixer or gMLPs.

Model	Perplexity*	Params (M)
$\text{BERT}_{\text{base}}$	4.37	110
$\text{BERT}_{\text{base}}$ + rel pos	4.26	110
$\text{BERT}_{\text{base}}$ + rel pos - attn = Random Synthetizer	5.64	96
MLP-Mixer	5.34	112
Linear gMLP, $s(Z) = f(Z)$	5.14	92
Additive gMLP, $s(Z) = Z + f(Z)$	4.97	92
Multiplicative gMLP, $s(Z) = Z \odot f(Z)$	4.53	92
Multiplicative, Split gMLP, $s(Z) = Z_1 \odot f(Z_2), Z = Z_1 \| Z_2$	4.35	102

* Standard deviation across multiple independent runs is around 0.01.

Pay Attention to MLPs [link]

- As you scale up you need more than 2 times the number of layers.
- Results on fine-tuning can depend on the tasks. They scale with similar Slope; conjecture: Scaling Law asymptotically does not depend on presence of self attention.

Table 4: Pretraining and dev-set finetuning results over increased model capacity. We use the relative positional encoding scheme for Transformers which performs the best in Table 3.

Model	#L	Params (M)	Perplexity	SST-2	MNLI-m
Transformer	6+6	67	**4.91**	90.4	81.5
gMLP	18	59	5.25	91.2	77.7
Transformer	12+12	110	**4.26**	91.3	83.3
gMLP	36	102	4.35	92.3	80.9
Transformer	24+24	195	3.83	92.1	85.2
gMLP	72	187	**3.79**	93.5	82.8
Transformer	48+48	365	3.47	92.8	86.3
gMLP	144	357	**3.43**	95.1	84.6

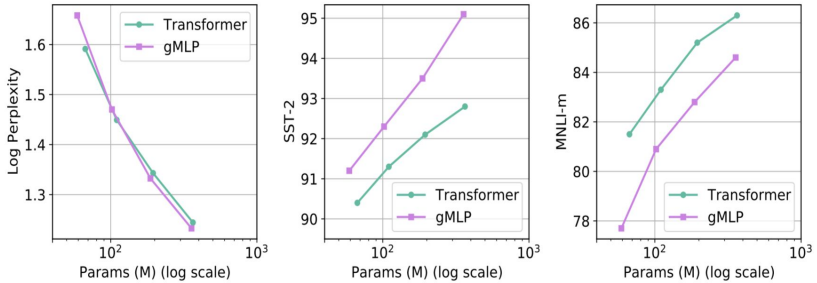


Figure 5: Scaling properties with respect to perplexity and finetuning accuracies. The figures show that for pretraining, gMLPs are equally good at optimizing perplexity as Transformers. For finetuning, the two model families exhibit comparable scalability despite task-specific offsets.

Pay Attention to MLPs [LINK]

- ***My Conjecture:*** `MLM < SST-2 < MNLI < SQUAD`
- `Then gMLP takes a hit; you need to bring back attention.`
- `Tiny attention is one-head.`

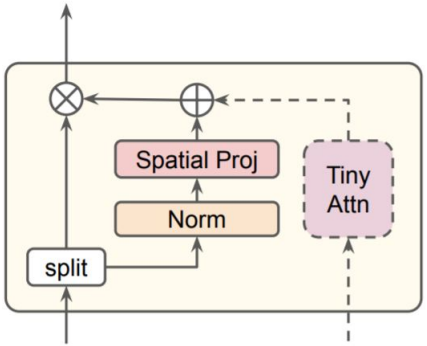


Table 6: Pretraining perplexities and dev-set results for finetuning. "ours" indicates models trained using our setup. We report accuracies for SST-2 and MNLI, and F1 scores for SQuAD v1.1/2.0.

	Perplexity	SST-2	MNLI (m/mm)	SQuAD v1.1	SQuAD v2.0	Attn Size	Params (M)
$\text{BERT}_{\text{base}}$ [2]	–	92.7	84.4/-	88.5	76.3	768 (64 × 12)	110
$\text{BERT}_{\text{base}}$ (ours)	4.17	93.8	85.6/85.7	90.2	78.6	768 (64 × 12)	110
$\text{gMLP}_{\text{base}}$	4.28	94.2	83.7/84.1	86.7	70.1	–	130
$\text{aMLP}_{\text{base}}$	3.95	93.4	85.9/85.8	90.7	80.9	64	109
$\text{BERT}_{\text{large}}$ [2]	–	93.7	86.6/-	90.9	81.8	1024 (64 × 16)	336
$\text{BERT}_{\text{large}}$ (ours)	3.35	94.3	87.0/87.4	92.0	81.0	1024 (64 × 16)	336
$\text{gMLP}_{\text{large}}$	3.32	94.8	86.2/86.5	89.5	78.3	–	365
$\text{aMLP}_{\text{large}}$	3.19	94.8	88.4/88.4	92.2	85.4	128	316
$\text{gMLP}_{\text{xlarge}}$	2.89	95.6	87.7/87.7	90.9	82.1	–	941

Pay Attention to Mlps [link]

!! These are my takeaways from the paper !!

- Cannot find a single-sentence motivation to recommend these models for NLP.
- At a theoretical level the finding is intriguing. Could one prove that gMLPs have the Universal Approximation property for sequences ?
- Transformers were first studied for Generation (MT). Maybe these models will take even a bigger hit on Generation?
- Besides the increased depth, bringing back the tiny attention might slow things down significantly at inference time compared to the Transformer.

Synthetizer [link]

- Is Self-Attention really required?
- They suggest surrogating self-alignments.
- Investigate both pretraining, fine-tuning and generation.
- Their models take a hit, discuss how to close the gap with the Transformer.
- **This paper appeared before the Mixer paper; Mixer is a special case of Synthetizer.**
- Close relationship with previous work:
    - Raganato: for Transformer encoders use fixed (non-learnable) attention patterns.
    - Lightweight and Dynamic convolutions: Attention patterns are replaced by convolutional weights; either fixed (Light) or a linear function of the current token (Dynamic)

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

w

Synthetizer [link]

- Random Synthetizer: W is a parameter (not a new idea, in Light Convolutions It is a parameter with the convolutional inductive bias) -> **1st order** interaction

$$\text{softmax}(R_{h,\ell})$$

- Dense Synthetizer: R is predicted by an MLP -> 2nd order interaction. Note with Dynamic Convolutions, R was just predicted by a linear map. Back the Idea of **2nd order**.

$$W_{2,h,\ell}(\sigma_R(W_{1,h,\ell}(X_{i,h,\ell}))$$

- Let N be the sequence length. As there is no convolutional bias… if N is large >> channel dimension then comparison with the Transformer (on parameter size) **starts to become unfavorable.**

Synthetizer [link]

- Factorized Dense Syn; N = a * b; build an R_a projecting to sequence length a; R_b to seq length b; then tile and get back sequence length N.

$$A_{h,\ell}, B_{h,\ell} = F_{A,h,\ell}(X_{i,h,\ell}), F_{B,h,\ell}(X_{i,h,\ell})$$

$$C_{h,\ell} = H_A(A_{h,\ell}) * H_B(B_{h,\ell})$$

$$\text{softmax}(C_{h,\ell})$$

- Factorized Random: Use a latent factorization of the parameter R (k=8)

$$\text{softmax}(R_{1,h,\ell} R_{2,h,\ell}^{\top})$$

Synthetizer [link]

- Why the dense one does not use the factorization idea and relies on tiling?

- You can take a mixture (even using the standard self-attention as one candidate).

$$Y_{h,\ell} = \text{softmax}(\alpha_{1,h,\ell} S_{1,h,\ell}(X_{h,\ell}) + \cdots \alpha_{N,h,\ell} S_{N,h,ell}(X_{h,\ell}))G_{h,\ell}(X_{h,\ell}).$$

Synthetizer [link]

- Machine Translation and LM: you need attention and X-attention.
- Limitation 1: One model size (~ Transformer Base)
- Limitation 2: En to close languages (De and Fr).
- Paper on Dynamic Convolutions reported that they outperformed the Transformer. Here not the case.
- Unclear how SVO / SOV languages perform. Unclear what happens at the scale of Transformer Big.

Model	NMT (BLEU) $\|\theta\|$	EnDe	EnFr	LM (PPL) $\|\theta\|$	LM
Transformer†	67M	27.30	38.10	-	-
Transformer	67M	27.67	41.57	70M	38.21
Synthesizer (Fixed Random)	61M	23.89	38.31	53M	50.52
Synthesizer (Random)	67M	27.27	41.12	58M	40.60
Synthesizer (Factorized Random)	61M	27.30	41.12	53M	42.40
Synthesizer (Dense)	62M	27.43	41.39	53M	40.88
Synthesizer (Factorized Dense)	61M	27.32	41.57	53M	41.20
Synthesizer (Random + Dense)	67M	27.68	41.21	58M	42.35
Synthesizer (Dense + Vanilla)	74M	27.57	41.38	70M	**37.27**
Synthesizer (Random + Vanilla)	73M	**28.47**	**41.85**	70M	40.05

Table 2. Experimental Results on WMT'14 English-German, WMT'14 English-French Machine Translation tasks and Language Modeling One Billion (LM1B). † denotes original reported results in (Vaswani et al., 2017).

Synthetizer [link]

- On summarization attention does not help

Model	Sum. RL	Dialogue B_4	Dialogue RL	Dialogue Met.	Dialogue CIDr
Trans.	35.77	3.20	13.38	5.89	18.94
Synthesizer Models					
R	33.10	2.25	15.00	6.42	19.57
D	33.70	**4.02**	**15.22**	**6.61**	**20.54**
D+V	**36.02**	3.57	14.22	6.32	18.87
R+V	35.95	2.28	14.79	6.39	19.09

Table 3. Experimental results on Abstractive Summarization (CNN/Dailymail) and Dialogue Generation (PersonaChat). We report on RL (Rouge-L), B4 (Bleu-4), Met. (Meteor) and CIDr.

- On MLM you need attention. (Speed comparison at training, but unclear if Convolutions were efficiently implemented on TPU)

Model	Log PPL	Steps/Sec	Params	TFLOPS
Trans.	1.865	3.90	223M	3.70
DyConv	2.040	2.65	257M	3.93
LightConv	1.972	4.05	224M	3.50
Syn (D)	1.965	3.61	224M	3.80
Syn (R)	1.972	**4.26**	254M	3.36
Syn (R+V)	1.849	3.79	292M	4.03
Syn (D+V)	**1.832**	3.34	243M	4.20

Table 4. Validation perplexity scores on C4 dataset (Raffel et al., 2019). All models are at approximately similar parameterization.

Synthetizer [link]

- On GLUE/SuperGLUE you need Attention. Harder tasks (entailment) benefit from attention (longer dependencies get connected).

Model	Glue	CoLA	SST	MRPC	STSB	QQP	MNLI	QNLI	RTE
T5 (Base)	83.5	53.1	**92.2**	**92.0/88.7**	89.1/88.9	88.2/91.2	84.7/**85.0**	91.7	76.9
T5 (Base+)	82.8	54.3	92.9	88.0/83.8	85.2/85.4	88.3/91.2	84.2/84.3	91.4	79.1
DyConv	69.4	33.9	90.6	82.6/72.5	60.7/63.1	84.2/88.2	73.8/75.1	84.4	58.1
Syn (R)	75.1	41.2	91.2	85.9/79.4	74.0/74.3	85.5/89.0	77.6/78.1	87.6	59.2
Syn (D)	72.0	18.9	89.9	86.4/79.4	75.3/75.5	85.2/88.3	77.4/78.1	86.9	57.4
Syn (D+V)	82.6	48.6	92.4	91.2/87.7	88.9/89.0	88.6/91.5	84.3/84.8	91.7	75.1
Syn (R+V)	**84.1**	**53.3**	**92.2**	91.2/87.7	**89.3/88.9**	**88.6/91.4**	**85.0**/84.6	**92.3**	**81.2**

Table 5. Experimental results (dev scores) on multi-task language understanding (GLUE benchmark) for *small* model and `en-mix` mixture. Note: This task has been co-trained with SuperGLUE.

Model	SGlue	BoolQ	CB	CoPA	MultiRC	ReCoRD	RTE	WiC	WSC
T5 (Base)	70.3	78.2	72.1/83.9	59.0	73.1/32.1	**71.1/70.3**	77.3	**65.8**	**80.8**
T5 (Base+)	70.7	79.3	81.1/87.5	60.0	75.1/34.4	71.7/70.7	80.5	64.6	71.2
DyConv	57.8	66.7	65.9/73.2	58.0	57.9/8.71	58.4/57.4	69.0	58.6	73.1
Syn (R)	61.1	69.5	54.6/73.2	60.0	63.0/15.7	58.4/57.4	67.5	64.4	66.3
Syn (D)	58.5	69.5	51.7/71.4	51.0	66.0/15.8	54.1/53.0	67.5	65.2	58.7
Syn (D+V)	69.7	79.3	74.3/85.7	64.0	73.8/33.7	69.9/69.2	78.7	64.3	68.3
Syn (R+V)	**72.2**	**79.3**	**82.7/91.1**	**64.0**	**74.3/34.9**	70.8/69.9	**82.7**	64.6	75.0

Table 6. Experimental results (dev scores) on multi-task language understanding (SuperGLUE benchmark) for *small* model and `en-mix` mixture. Note: This task has been co-trained with GLUE.

Synthetizer [LINK]

!! These are my takeaways from the paper !!

- Cannot find a single-sentence motivation to recommend these models for NLP.
- At a theoretical level the finding is intriguing. Very comprehensive investigation across tasks.
- Unclear inference speed + how efficiently operations could be implemented. (No convolution prior).
- Sometimes Random + Vanilla > Dense + Vanilla (which should be more expressive). Is there an issue with training?