# [ArXiv2020]Linformer: Self-Attention with Linear Complexity

Facebook AI, Seattle, WA

- Category
  - 벤다이어그램 : Linformer , Linera Transformer, Performer $\in$ Low Rnak / Kernels
  - 그래프 : Linformer, MCA $\in$ Memory compress

# 1. Introduction

| Model Architecture | Complexity per Layer | Sequential Operation |
| --- | --- | --- |
| Recurrent | $O(n)$ | $O(n)$ |
| Transformer, (Vaswani et al., 2017) | $O(n^2)$ | $O(1)$ |
| Sparse Tansformer, (Child et al., 2019) | $O(n\sqrt{n})$ | $O(1)$ |
| Reformer, (Kitaev et al., 2020) | $O(n\log(n))$ | $O(\log(n))$ |
| Linformer | $O(n)$ | $O(1)$ |

# 2. Related Work

- Transformer and self-attention

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) = \underbrace{\text{softmax}\left[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d_k}}\right]}_{P} VW_i^V$$

**Sparse Attention** (Child et al., 2019): This technique improves the efficiency of self-attention by adding sparsity in the context mapping matrix $P$. For example, the Sparse Transformer (Child et al., 2019) only computes $P_{ij}$ around the diagonal of matrix $P$ (instead of the all $P_{ij}$). Meanwhile, blockwise self-attention (Qiu et al., 2019) divides $P$ into multiple blocks and only computes $P_{ij}$ within the selected blocks. However, these techniques also suffer a large performance degradation, while having only limited additional speed-up, i.e., 2% drop with 20% speed up.

**LSH Attention** (Kitaev et al., 2020): Locally-sensitive hashing (LSH) attention utilizes a multi-round hashing scheme when computing dot-product attention, which in theory reduces the self-attention complexity to $O(n \log(n))$. However, in practice, their complexity term has a large constant $128^2$ and it is only more efficient than the vanilla transformer when sequence length is extremely long.

# 3. Self-Attention is Low Rank

## 기본 세팅

- We use two pretrained transformer models, RoBERTa-base (12-layer stacked transformer) and RoBERTa-large (24-layer stacked transformer) on two tasks
- two tasks: masked-language-modeling task on Wiki103 and classification task on IMDB
- (RoBERTa: A Robustly Optimized BERT Pretraining Approach)

## Empirical view

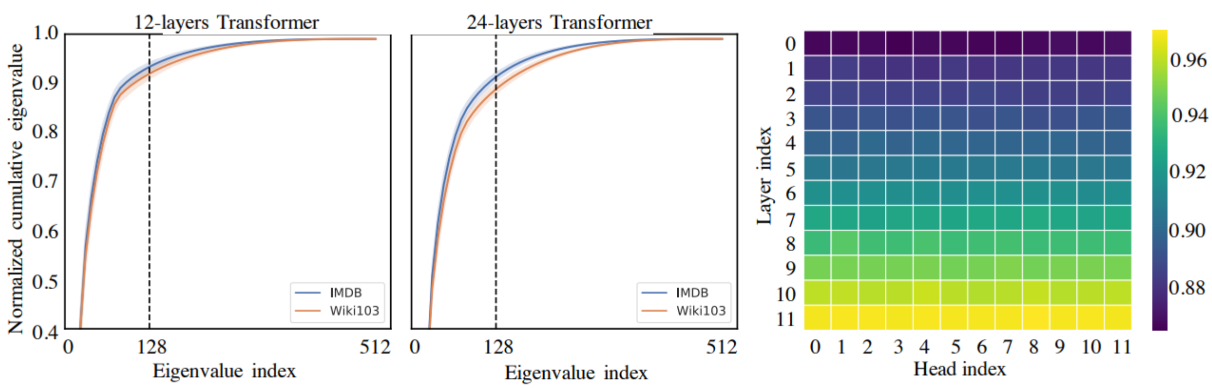



Figure 1: Left two figures are spectrum analysis of the self-attention matrix in pretrained transformer model (Liu et al., 2019) with $n = 512$. The Y-axis is the normalized cumulative singular value of context mapping matrix $P$, and the X-axis the index of largest eigenvalue. The results are based on both RoBERTa-base and large model in two public datasets: Wiki103 and IMDB. The right figure plots the heatmap of normalized cumulative eigenvalue at the 128-th largest eigenvalue across different layers and heads in Wiki103 data.

# [Theroem 1] self-attention is low rank

**Theorem 1.** (self-attention is low rank) *For any $Q, K, V \in \mathbb{R}^{n\times d}$ and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d}$, for any column vector $w \in \mathbb{R}^n$ of matrix $VW_i^V$, there exists a low-rank matrix $\tilde{P} \in \mathbb{R}^{n\times n}$ such that*

$$\Pr(\|\tilde{P}w^T - Pw^T\| < \epsilon\|Pw^T\|) > 1 - o(1) \text{ and } \operatorname{rank}(\tilde{P}) = \Theta(\log(n)), \tag{3}$$

*where the context mapping matrix $P$ is defined in (2).*

- Proof

$$P = \text{softmax}\underbrace{\left[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}}\right]}_{A} = \exp(A) \cdot D_A^{-1},$$

▼ Johnson–Lindenstrauss lemma (JL lemma) - from stanford cs369m lecture1.pdf

### 3 Johnson-Lindenstrauss Lemma

**Lemma** For any $0 < \epsilon < 1$ and any interger n let k be a possitive interger such that

$$k \geq \frac{24}{3\epsilon^2 - 2\epsilon^3}\log n \tag{2}$$

then for any set $A$ of $n$ points $\in \Re^d$ there exists a map $f : \Re^d \to \Re^k$ such that for all $x_i, x_j \in A$

$$(1-\epsilon)||x_i - x_j||^2 \leq ||f(x_i) - f(x_j)||^2 \leq (1+\epsilon)||x_i - x_j||^2 \tag{3}$$

  - $\tilde{P} = PR^TR$
    - Let R be an k × n matrix, 1 ≤ k ≤ n, with i.i.d. entries from N(0, 1/k)

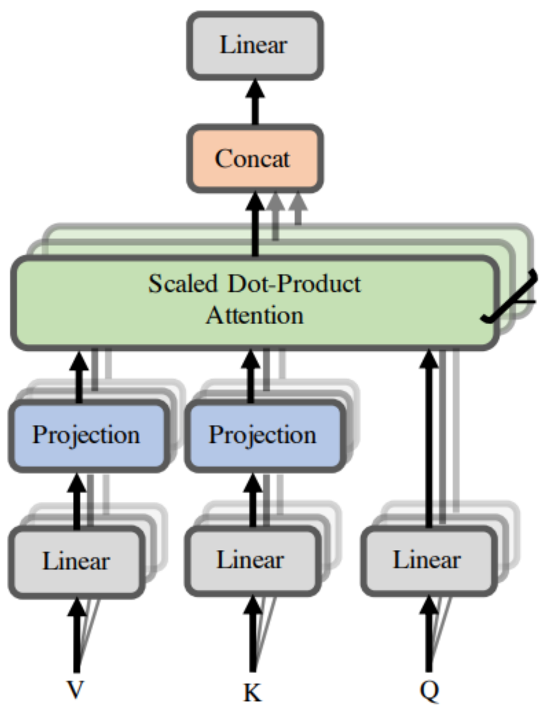
---

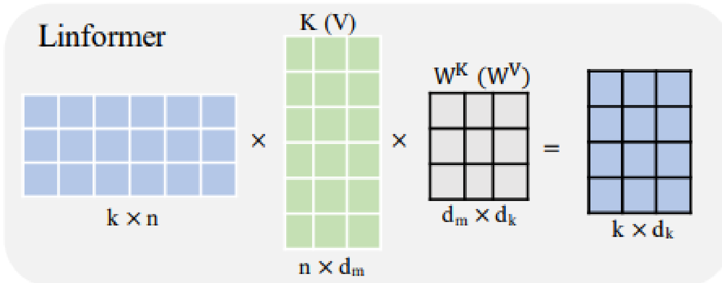
- **Given the low-rank property of the context mapping matrix P,** one straightforward idea is to use singular value decomposition (SVD) to approximate P with a low-rank matrix $P_{low}$

$$P \approx P_{\text{low}} = \sum_{i=1}^{k}\sigma_i u_i v_i^T = \underbrace{\begin{bmatrix} u_1, \cdots, u_k \end{bmatrix}}_{,} \operatorname{diag}\{\sigma_1, \cdots, \sigma_k\} \left.\begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix}\right\} k \tag{6}$$

- One can use $P_{low}$ to approximate self-attention (2) with  error and O(nk) time and space complexity.
- However, this approach **requires performing an SVD decomposition in each self-attention matrix**, which adds
  additional complexity.
  - ▼ SVD time complexity
    - SVD : $O(m^2n + mn^2 + n^3)$
    - truncated SVD : $O(mn^2)$
    - reference : https://mathoverflow.net/questions/161252/what-is-the-time-complexity-of-truncated-svd
- Therefore, we propose **another approach for low-rank approximation** that avoids this added complexity.

# 4. Model






- **Add two linear projection matrices $E_i, F_i \in R^{n \times k}$** when computing key and value : **(n x d) → (k x d)**

$$
\begin{aligned}
\overline{\text{head}_i} &= \text{Attention}(QW_i^Q, E_iKW_i^K, F_iVW_i^V) \\
&= \underbrace{\text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n \times k} \cdot \underbrace{F_iVW_i^V}_{k \times d},
\end{aligned}
$$

- only require O(nk) time and space complexity
- if k << n, then we can significantly reduce the memory and space consumption.
- The following theorem states that, when **k = O(d/2) (independent of n)**, one can approximate $P \cdot VW_i^V$ using linear self-attention with error.

---

## [Theorem 2] Linear self-attention

**Theorem 2.** (Linear self-attention) *For any $Q_i, K_i, V_i \in \mathbb{R}^{n \times d}$ and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d}$, if $k = \min\{\Theta(9d\log(d)/\epsilon^2), 5\Theta(\log(n)/\epsilon^2)\}$, then there exists matrices $E_i, F_i \in \mathbb{R}^{n \times k}$ such that, for any row vector $w$ of matrix $QW_i^Q(KW_i^K)^T/\sqrt{d}$, we have*

$$
\Pr\left(\|\text{softmax}(wE_i^T)F_iVW_i^V - \text{softmax}(w)VW_i^V\| \le \epsilon\|\text{softmax}(w)\|\|VW_i^V\|\right) > 1 - o(1) \quad (8)
$$

- Proof

*Proof.* Define $E = \delta R$ and $F = e^{-\delta} R$, where $R \in \mathbb{R}^{n \times k}$ with i.i.d. entries from $N(0, 1/k)$, $\delta$ is a constant with $\delta = 1/2^n$. We will first prove that for any row vector $x \in \mathbb{R}^n$ of matrix $QK^T$ and column vector $y \in \mathbb{R}^n$ of matrix $V$,

$$\Pr\left(\| \exp(xE^T)Fy^T - \exp(x)y^T \| \leq \epsilon \| \exp(x)y^T \|\right) > 1 - 2e^{-(\epsilon^2 - \epsilon^3)k/4}. \tag{21}$$

---

- Parameter sharing between projections

  (1) Headwise sharing : $E_i = E$ and $F_i = F$ across all heads $i$

  (2) Key-value sharing : $E_i = F_i = E$ across all heads $i$

  (3) Layerwise sharing : $E$

  ex) in a 12-layer, 12-head stacked Transformer model, headwise sharing, key-value sharing and layerwise sharing will introduce 24, 12, and 1 distinct linear projection matrices, respectively.

- Nonuniform projected dimension
  - As shown in figure 1(right), heads in higher layer tends towards a more skewed distributed spectrum (lower rank).
  - This implies one can choose a smaller projected dimension k for higher layers.

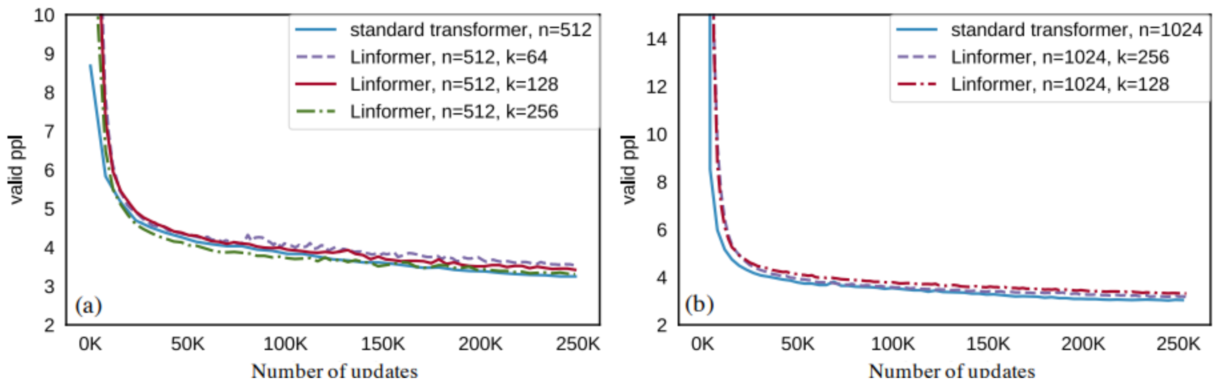
# 5. Experiments

## 5.1 Pretraining Perplexities

- Compare the pretraining performance of our proposed architecture against RoBERTa based on the Transformer
- we use BookCorpus + English Wikipedia as our pretraining set (3300M words).
- All models are pretrained with the masked-language-modeling (MLM) objective
- parallelized across 64 Tesla V100 GPUs with 250k updates.

## (1) Effect of projected dimension

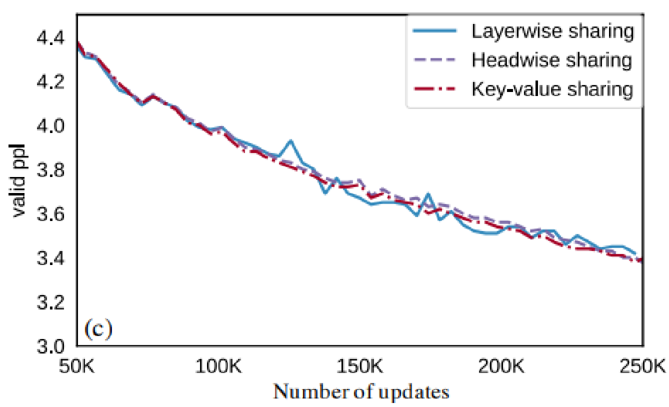
- different k & n=512/1024




- As expected, the Linformer performs better as projected dimension k increases
- However, even at k = 128 for n = 512 and k = 256 for n = 1024, Linformer's performance is already nearly on par with the original Transformer

## (2) Effect of Sharing projections

- n=512

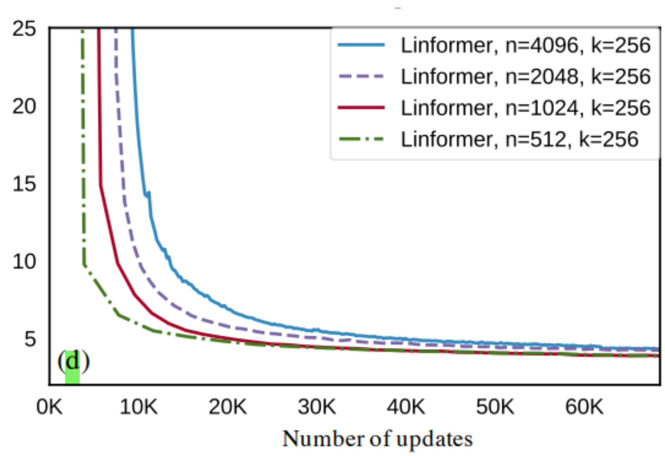



- when we use just a single projection matrix (i.e. for layerwise sharing), the resulting Linformer model's validation perplexity almost matches that of the the non-shared model

→ we can decrease the number of additional parameters in our model, and consequently, it’s memory consumption, without much detriment to performance.

### (3) Effect of longer sequences

- $n \in \{512, 1024, 2048, 4096\}$ & k fixed at 256.




- As sequence length increases, even though our projected dimension is fixed, the final perplexities after convergence remain about the same.
- This further empirically supports our assertion that the Linformer is linear-time.

## 5.2 Downstrean Results

- We finetune our Linformer on
  - IMDB (sentiment classification)
  - SST-2 (sentiment classification)
  - QNLI (natural language inference)
  - QQP (textual similarity)

| $n$ | Model | SST-2 | IMDB | QNLI | QQP | Average |
|---|---|---|---|---|---|---|
| 512 | Liu et al. (2019), RoBERTa-base | 93.1 | 94.1 | 90.9 | **90.9** | 92.25 |
| | Linformer, 128 | 92.4 | 94.0 | 90.4 | 90.2 | 91.75 |
| | Linformer, 128, shared kv | **93.4** | 93.4 | 90.3 | 90.3 | 91.85 |
| | Linformer, 128, shared kv, layer | 93.2 | 93.8 | 90.1 | 90.2 | 91.83 |
| | Linformer, 256 | 93.2 | 94.0 | 90.6 | 90.5 | 92.08 |
| | Linformer, 256, shared kv | 93.3 | 93.6 | 90.6 | 90.6 | 92.03 |
| | Linformer, 256, shared kv, layer | 93.1 | 94.1 | **91.2** | 90.8 | **92.30** |
| 512 | Devlin et al. (2019), BERT-base | 92.7 | 93.5 | 91.8 | 89.6 | 91.90 |
| | Sanh et al. (2019), Distilled BERT | 91.3 | 92.8 | 89.2 | 88.5 | 90.45 |
| 1024 | Linformer, 256 | 93.0 | 93.8 | 90.4 | 90.4 | 91.90 |
| | Linformer, 256, shared kv | 93.0 | 93.6 | 90.3 | 90.4 | 91.83 |
| | Linformer, 256, shared kv, layer | 93.2 | **94.2** | 90.8 | 90.5 | 92.18 |

Table 2: Dev set results on benchmark natural language understanding tasks. The RoBERTa-base model here is pretrained with same corpus as BERT.

- n = 512, k = 128 : comparable to RoBERTa
- n = 512, k = 256 : even slightly outperforms RoBERTa
  - although the Linformer's layerwise sharing strategy shares a single projection matrix across the entire model, it actually exhibits the best accuracy
- (n = 1024, k = 256) $\sim$ (n = 512, k = 256)
  - empirically supports that the performance of Linformer model is mainly **determined by the projected dimension k instead of the ratio n/k.**

## 5.3 Inference-time Efficiency Results

- benchmark both models' inference speed and memory on a 16GB Tesla V100 GPU card.
- We randomly generate data up to some sequence length n and perform a full forward pass on a multiple batches.
- We also choose batch size based on the maximum batch size that can fit in memory

| length $n$ | projected dimensions $k$ | | | | |
|---|---|---|---|---|---|
| | 128 | 256 | 512 | 1024 | 2048 |
| 512 | 1.5x | 1.3x | - | - | - |
| 1024 | 1.7x | 1.6x | 1.3x | - | - |
| 2048 | 2.6x | 2.4x | 2.1x | 1.3x | - |
| 4096 | 3.4x | 3.2x | 2.8x | 2.2x | 1.3x |
| 8192 | 5.5x | 5.0x | 4.4x | 3.5x | 2.1x |
| 16384 | 8.6x | 7.8x | 7.0x | 5.6x | 3.3x |
| 32768 | 13x | 12x | 11x | 8.8x | 5.0x |
| 65536 | 20x | 18x | 16x | 14x | 7.9x |

| length $n$ | projected dimensions $k$ | | | | |
|---|---|---|---|---|---|
| | 128 | 256 | 512 | 1024 | 2048 |
| 512 | 1.7x | 1.5x | - | - | - |
| 1024 | 3.0x | 2.9x | 1.8x | - | - |
| 2048 | 6.1x | 5.6x | 3.6x | 2.0x | - |
| 4096 | 14x | 13x | 8.3x | 4.3x | 2.3x |
| 8192 | 28x | 26x | 17x | 8.5x | 4.5x |
| 16384 | 56x | 48x | 32x | 16x | 8x |
| 32768 | 56x | 48x | 36x | 18x | 16x |
| 65536 | 60x | 52x | 40x | 20x | 18x |

Table 3: Inference-time efficiency improvements of the Linformer over the Transformer, across various projected dimensions $k$ and sequence lengths $n$. Left table shows time saved. Right table shows memory saved.

- We also plot inference times of both Linformer and Transformer on the 100 data samples in the top right of Figure 2