Matrix decomposition as an alternative to attention

Paper from ICLR 2021 ~DL seminar ~ Ruard van Workum

Attention has become the most widely adopted global context module

But how irreplaceable is it?

Matrix Decomposition/Factorization as noise-removal

Using classic techniques of matrix decomposition, we can factorize a matrix into a low-rank reconstruction and noise.

Suppose that the given data are arranged as the columns of a large matrix $\boldsymbol{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$. A general assumption is that there is a low-dimensional subspace, or a union of multiple subspaces hidden in $\boldsymbol{X}$. That is, there exists a dictionary matrix $\boldsymbol{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_r] \in \mathbb{R}^{d \times r}$ and corresponding codes $\boldsymbol{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_n] \in \mathbb{R}^{r \times n}$ that $\boldsymbol{X}$ can be expressed as

$$\boldsymbol{X} = \overset{\xleftarrow{\textit{generation}}}{\underset{\xrightarrow[\textit{decomposition}]{}}{\bar{\boldsymbol{X}} + \boldsymbol{E} = \boldsymbol{D}\boldsymbol{C}}} + \boldsymbol{E}, \tag{1}$$

where $\bar{\boldsymbol{X}} \in \mathbb{R}^{d \times n}$ is the output low-rank reconstruction, and $\boldsymbol{E} \in \mathbb{R}^{d \times n}$ is the noise matrix to be discarded. Here we assume that the recovered matrix $\bar{\boldsymbol{X}}$ has the low-rank property, such that

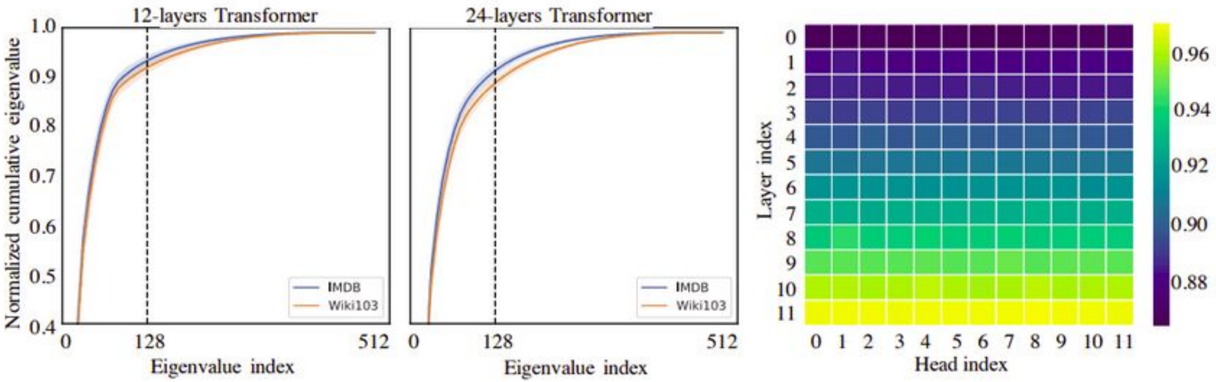
$$\operatorname{rank}(\bar{\boldsymbol{X}}) \le \min(\operatorname{rank}(\boldsymbol{D}), \operatorname{rank}(\boldsymbol{C})) \le r \ll \min(d, n). \tag{2}$$

Modelling Global context / long-range dependencies

Observation 1: long-range dependencies can be modelled through low-rank

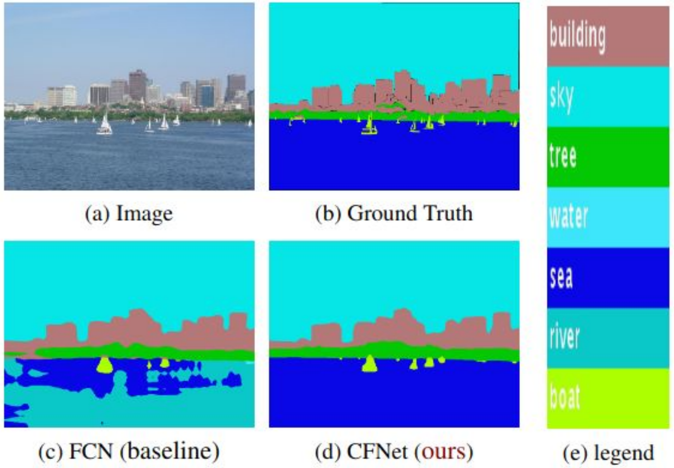
- Long-range dependencies -> correlation -> low-rank
- Interestingly, self-attention matrices are also known to have a low-rank structure:

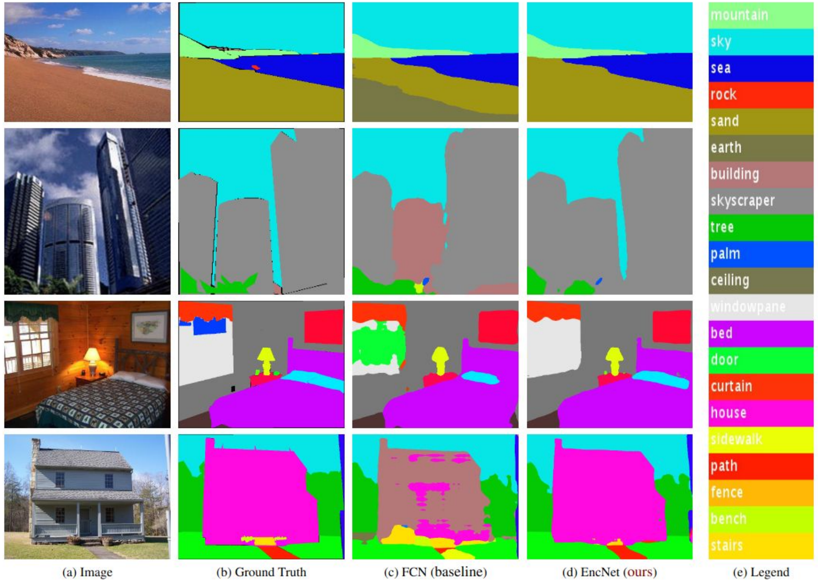
$$P = \mathrm{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)$$



Modelling Global context / long-range dependencies

Observation 2: “Vanilla” CNN’s extract features that are too local for tasks requiring global context

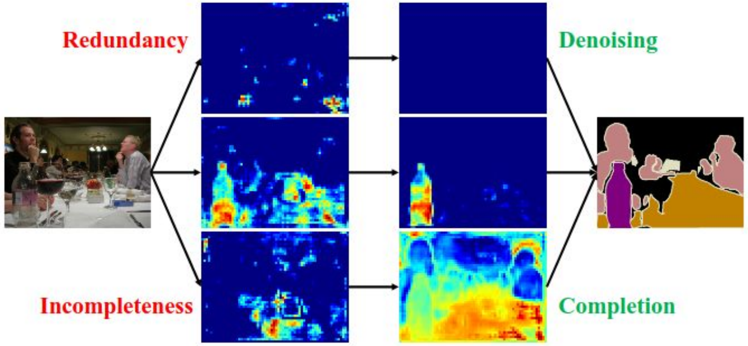




Key Idea

"Learn global context by extracting the (low-rank) clean signal subspace of inputs using classic matrix decomposition, which filters out the redundancy and incompleteness (of vanilla CNN's) at the same time."

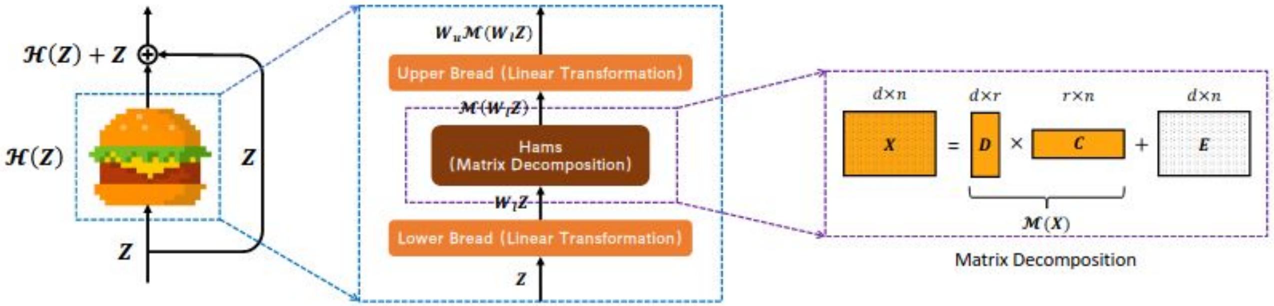
- By-product of the low-rank assumption is efficiency in terms of computation and memory
- This is implemented through the "Hamburger" architecture



How does that look like?

How does that look like?

Turn image tensors $x \in \mathbb{R}^{C \times H \times W}$ into hyper-pixels $x \in \mathbb{R}^{C \times HW}$



notion of Sec. 2.1, the general objective function of matrix decomposition is

$$\min_{\boldsymbol{D},\boldsymbol{C}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{DC}) + \mathcal{R}_1(\boldsymbol{D}) + \mathcal{R}_2(\boldsymbol{C}) \tag{4}$$

where $\mathcal{L}$ is the reconstruction loss, $\mathcal{R}_1$ and $\mathcal{R}_2$ are regularization terms for the dictionary $\boldsymbol{D}$ and the codes $\boldsymbol{C}$. Denote the optimization algorithm to minimize Eq. (4) as $\mathcal{M}$. $\mathcal{M}$ is the core architecture we deploy in our global context module. To help readers further understand this modeling, We also provide a more intuitive illustration in Appendix G.

Thus, we’re modelling the global context as a low-rank recovery problem with matrix decomposition as its solution.

Matrix decomposition part (Ham)

$$\min_{\boldsymbol{D},\boldsymbol{C}} \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C}\|_F \quad \text{s.t. } \mathbf{c}_i \in \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_r\},$$

$$\min_{\boldsymbol{D},\boldsymbol{C}} \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C}\|_F \quad \text{s.t. } \boldsymbol{D}_{ij} \geq 0, \boldsymbol{C}_{jk} \geq 0.$$

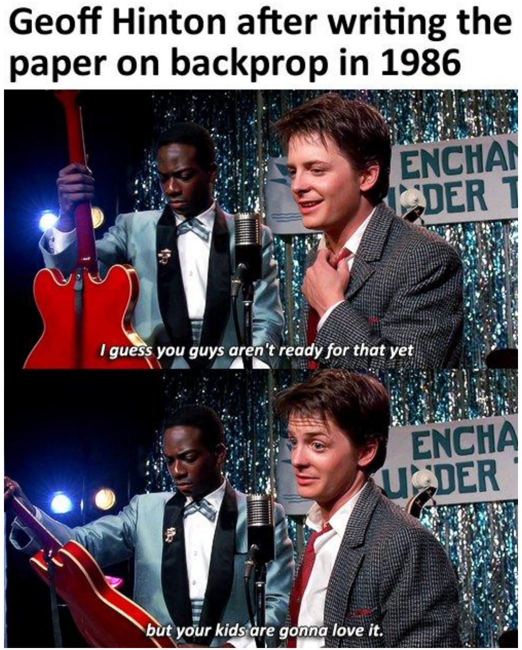
Algorithm 1 Ham: Soft VQ

Input $\boldsymbol{X}$. Initialize $\boldsymbol{D}, \boldsymbol{C}$.
for k from 1 to K **do**
  $\boldsymbol{C} \leftarrow softmax(\frac{1}{T} cosine(\boldsymbol{D}, \boldsymbol{X}))$
  $\boldsymbol{D} \leftarrow \boldsymbol{X}\boldsymbol{C}^\top diag(\boldsymbol{C}\mathbf{1}_n)^{-1}$
end for
Output $\bar{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{C}$.

Algorithm 2 Ham: NMF with MU

Input $\boldsymbol{X}$. Initialize non-negative $\boldsymbol{D}, \boldsymbol{C}$
for k from 1 to K **do**
  $\boldsymbol{C}_{ij} \leftarrow \boldsymbol{C}_{ij} \frac{(\boldsymbol{D}^\top \boldsymbol{X})_{ij}}{(\boldsymbol{D}^\top \boldsymbol{D}\boldsymbol{C})_{ij}}$
  $\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_{ij} \frac{(\boldsymbol{X}\boldsymbol{C}^\top)_{ij}}{(\boldsymbol{D}\boldsymbol{C}\boldsymbol{C}^\top)_{ij}}$
end for
Output $\bar{\boldsymbol{X}} = \boldsymbol{D}\boldsymbol{C}$.

Sounds cool, but how do I compute gradients?



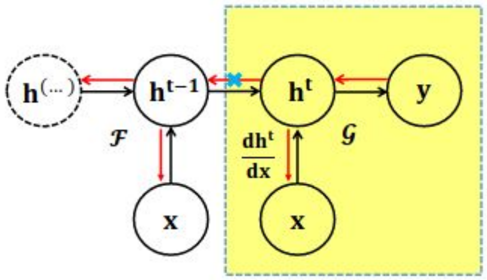
Approximation to BPTT

Imagine we have some iterative process with input **x**, output **y** and intermediate output **h_i**, and 2 operators/functions F and G:

$$\mathbf{h}^{i+1} = \mathcal{F}(\mathbf{h}^i, \mathbf{x}), \quad i = 0, 1, \cdots, t-1. \qquad \mathbf{y} = \mathcal{G}(\mathbf{h}^t).$$

In the BPTT algorithm the Jacobian matrix then follows from the Chain rule:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \sum_{i=0}^{t-1} \frac{\partial \mathbf{y}}{\partial \mathbf{h}^t} \left(\prod_{t \geq j > t-i} \frac{\partial \mathbf{h}^j}{\partial \mathbf{h}^{j-1}} \right) \frac{\partial \mathbf{h}^{t-i}}{\partial \mathbf{x}}.$$



Approximation to BPTT

Big problem: Gradients can become very ill-behaved in such scheme. (specifically when t becomes large)

The problematic scale of the gradients arise from the multiplication term and summation of a potentially infinite series when t -> infinity.

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \sum_{i=0}^{t-1} \frac{\partial \mathbf{y}}{\partial \mathbf{h}^t} \left(\prod_{t \geq j > t-i} \frac{\partial \mathbf{h}^j}{\partial \mathbf{h}^{j-1}} \right) \frac{\partial \mathbf{h}^{t-i}}{\partial \mathbf{x}}.$$

$$\left\{ \frac{\partial \mathbf{y}}{\partial \mathbf{h}^t} \left(\prod_{t \geq j > t-i} \frac{\partial \mathbf{h}^j}{\partial \mathbf{h}^{j-1}} \right) \frac{\partial \mathbf{h}^{t-i}}{\partial \mathbf{x}} \right\}_i,$$

Solution: View the summation as a series & only consider the first term (last step of optimization)

$$\widehat{\frac{\partial \mathbf{y}}{\partial \mathbf{x}}} = \frac{\partial \mathbf{y}}{\partial \mathbf{h}^*} \frac{\partial \mathcal{F}}{\partial \mathbf{x}}.$$

(one-step gradient)

Approximation to BPTT

Terms should vanish as t increases:

Proposition 1 $\{\mathbf{h}^i\}_t$ has linear convergence.

Proposition 2 $\lim_{t\to\infty} \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{h}^*}(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \mathbf{h}^*})^{-1}\frac{\partial \mathcal{F}}{\partial \mathbf{x}}$.

Proposition 3 $\lim_{t\to\infty} \|\frac{\partial \mathbf{y}}{\partial \mathbf{h}^0}\| = 0, \lim_{t\to\infty} \|\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\| \leq \frac{L_{\mathcal{G}} L_x}{1-L_h}$.

Besides, they reach higher scores on tasks using One-Step &

it reduce complexity from O(t) -> O(1)

Method	**One-Step**	**BPTT**
VQ	77.7(77.4)	76.6(76.3)
CD	78.1(77.5)	75.0(74.6)
NMF	78.3(77.8)	77.4(77.0)

Experiments

Ablation Study on PASCAL VOC dataset:

Table 2: Ablation on components of Hamburger with NMF Ham.

Method	mIoU(%)	Params
baseline	75.9(75.7)	32.67M
basic	78.3(77.8)	+0.50M
- ham	75.8(75.6)	+0.50M
- upper bread	77.0(76.8)	+0.25M
- lower bread	77.3(77.2)	+0.25M
only ham	77.0(76.8)	+0M

Experiments

Screening latent dimensions r, d & optimization iterations K:

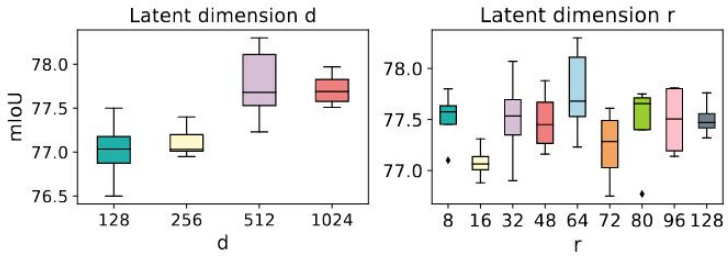


Figure 3: Ablation on d and r

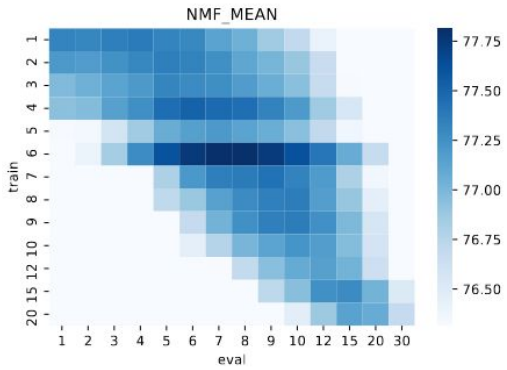


Figure 4: Ablation on K

note: Attention -> O(n^2d), MD -> O(ndr)

Experiments

SOTA performance on PASCAL VOC dataset

Table 4: Comparisons with state-of-the-art on the PASCAL VOC test set w/o COCO pretraining.

Method	mIoU(%)
PSPNet (Zhao et al., 2017)	82.6
DFN* (Yu et al., 2018)	82.7
EncNet (Zhang et al., 2018)	82.9
DANet* (Fu et al., 2019)	82.6
DMNet* (He et al., 2019a)	84.4
APCNet* (He et al., 2019b)	84.2
CFNet* (Zhang et al., 2019b)	84.2
SpyGR* (Li et al., 2020)	84.2
SANet* (Zhong et al., 2020)	83.2
OCR* (Yuan et al., 2020)	84.3
HamNet	**85.9**

Table 5: Results on the PASCAL-Context Val set.

Method	mIoU(%)
PSPNet (Zhao et al., 2017)	47.8
SGR* (Liang et al., 2018)	50.8
EncNet (Zhang et al., 2018)	51.7
DANet* (Fu et al., 2019)	52.6
EMANet* (Li et al., 2019a)	53.1
DMNet* (He et al., 2019a)	54.4
APCNet* (He et al., 2019b)	54.7
CFNet* (Zhang et al., 2019b)	54.0
SpyGR* (Li et al., 2020)	52.8
SANet* (Zhong et al., 2020)	53.0
OCR* (Yuan et al., 2020)	54.8
HamNet	**55.2**

The end


Nice paper
GitHub Link
Written in your favourite framework
Runs smoothly on your system without error or dependency issues