

Bonus: The Fold in Machine Learning

Compressing outer products and weight matrices – and a word that means several other things

James Pustejovsky

Brandeis University
Department of Computer Science

CS 135: Computational Semantics

Fall 2026

Where This Picks Up

In the binding deck we compressed one outer product. A role bound to a filler gave a matrix $\mathbf{r}\mathbf{f}^\top$, and the **fold** squeezed its n^2 entries into n :

$$\text{fold}(M)_k = \sum_{i+j \equiv k \pmod{n}} M_{ij}, \quad \text{fold}(\mathbf{r}\mathbf{f}^\top) = \mathbf{r} \circledast \mathbf{f}$$

That was built for one job. [This deck asks whether the same operation turns up elsewhere in machine learning](#). It does, in at least four places. The word “folding” also turns up, attached to compression techniques that are *not* this operation – and telling the two apart is half the point.

Two questions to carry through every slide

- 1 What $n \times n$ object is being compressed – a *representation*, computed fresh for each input, or a set of *parameters*, learned once?
- 2 What happens to each of its cells? Summed with others, forced to share a value, dropped, copied, or rewritten?

Roadmap

- I. The fold, redrawn
- II. **Fourier coefficients**, from scratch
- III. The **adjoint**, and Plate's decoder
- IV. Two things a network can compress
- V. Compressing a representation: **tensor sketch** and compact bilinear pooling
- VI. Asymmetric relations: **holographic embeddings** of knowledge graphs
- VII. Compressing the weights: **circulant** and **hashed** layers
- VIII. Folding inside **attention**
- IX. Other things called “folding”

The running example is the one from the binding deck, $\mathbf{u} = (1, 2, 0, 1)$ and $\mathbf{v} = (2, 0, 1, 1)$, shown in the corner wherever it is used.

Part I

The Fold, Redrawn

The Fold in One Picture

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Cell (i, j) of $\mathbf{uv}^\top$ holds $u_i v_j$. Color each cell by its label $(i + j) \bmod 4$, shown in its corner:

	v_0 = 2	v_1 = 0	v_2 = 1	v_3 = 1
$u_0 = 1$	$\begin{smallmatrix} 0 \\ 2 \end{smallmatrix}$	$\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}$	$\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}$	$\begin{smallmatrix} 3 \\ 1 \end{smallmatrix}$
$u_1 = 2$	$\begin{smallmatrix} 1 \\ 4 \end{smallmatrix}$	$\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}$	$\begin{smallmatrix} 3 \\ 2 \end{smallmatrix}$	$\begin{smallmatrix} 0 \\ 2 \end{smallmatrix}$
$u_2 = 0$	$\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}$	$\begin{smallmatrix} 3 \\ 0 \end{smallmatrix}$	$\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$	$\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}$
$u_3 = 1$	$\begin{smallmatrix} 3 \\ 2 \end{smallmatrix}$	$\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}$	$\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}$	$\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}$

The fold

$$\text{fold}(M)_k = \sum_{i+j \equiv k \pmod{n}} M_{ij}$$

Add up each color.

$$\begin{aligned} 2+2+0+0 &= 4, & 0+4+0+1 &= 5, & 1+0+0+1 &= 2, \\ 1+2+0+2 &= 5 \end{aligned}$$

So $\text{fold}(\mathbf{uv}^\top) = (4, 5, 2, 5) = \mathbf{u} \circledast \mathbf{v}$, the circular convolution of the binding deck.

Three Facts We Will Lean On

1. The fold is linear

$\text{fold}(M + N) = \text{fold}(M) + \text{fold}(N)$ and $\text{fold}(cM) = c \text{fold}(M)$ – each output is a plain sum of entries.

2. On an outer product, it is a convolution

$\text{fold}(\mathbf{u}\mathbf{v}^\top) = \mathbf{u} \circledast \mathbf{v}$, so it can be computed from the two vectors alone, *without ever forming the matrix* – and, as Part II shows, fast.

3. It cannot see the transpose

Transposing moves cell (i, j) to (j, i) , and $i + j = j + i$, so every cell stays in its group:
 $\text{fold}(M^\top) = \text{fold}(M)$. Hence $\mathbf{u} \circledast \mathbf{v} = \mathbf{v} \circledast \mathbf{u}$, and the antisymmetric part A of any matrix folds to $\mathbf{0}$.

Fact 3 is the one to remember: the fold discards exactly the part of a matrix that records which argument came first.

Sixteen Numbers to Four: Two Strategies

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Any linear map from 16 numbers to 4 has to give up 12 degrees of freedom. Which twelve? Three maps, each applied to $\mathbf{uv}^\top$:

2	0	1	1
4	0	2	2
0	0	0	0
2	0	1	1

keep the diagonal
(2, 0, 0, 1)

2	0	1	1
4	0	2	2
0	0	0	0
2	0	1	1

keep one row
(4, 0, 2, 2)

2	0	1	1
4	0	2	2
0	0	0	0
2	0	1	1

fold
(4, 5, 2, 5)

The first two **select**: they keep 4 cells exactly and ignore the other 12. The fold **aggregates**: every cell is counted once, merged with three others.

Hold on to this distinction. Selection keeps a few things exactly; aggregation keeps everything, blurred. Every technique in this deck does one or the other, or makes cells share values.

Part II

Fourier Coefficients, from Scratch

A Number Whose Powers Keep Clock Time

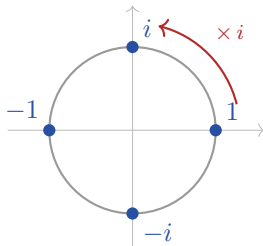
The puzzle. Convolution mixes every entry of $\mathbf{u}$ with every entry of $\mathbf{v}$. Is there a choice of coordinates in which it mixes nothing – acting on one coordinate at a time? There is, but it needs one new number.

The number i , with $i^2 = -1$. Its powers repeat with period 4:

$$i^0 = 1, \quad i^1 = i, \quad i^2 = -1, \quad i^3 = -i, \quad i^4 = 1, \quad \dots$$

So i^a depends only on $a \bmod 4$: the powers of i keep the time of a clock with four hours. Multiplying by i is a quarter turn counterclockwise; multiplying by $-i$, a quarter turn back.

Three facts are all we need. A complex number is $a + bi$; its **conjugate** is $\overline{a + bi} = a - bi$; and $(a + bi)(a - bi) = a^2 + b^2$, written $|a + bi|^2$.



Four Waves Around a Four-Step Circle

For $n = 4$, **wave** k is the vector of powers $\mathbf{f}_k = (i^{0 \cdot k}, i^{1 \cdot k}, i^{2 \cdot k}, i^{3 \cdot k})$: start at 1 and turn k quarter turns at every step.

	step 0	step 1	step 2	step 3	
$\mathbf{f}_0$	1	1	1	1	stays put
$\mathbf{f}_1$	1	i	-1	$-i$	once around the circle
$\mathbf{f}_2$	1	-1	1	-1	twice around
$\mathbf{f}_3$	1	$-i$	-1	i	three times around – once around backward

Every vector of length 4 is a combination of these four. They form a basis, like $\mathbf{e}_0, \dots, \mathbf{e}_3$, but made of *patterns* instead of *positions*.

What makes them special: rotating a wave does not scramble it. Shifting $\mathbf{f}_1 = (1, i, -1, -i)$ one place to the right gives $(-i, 1, i, -1) = -i \mathbf{f}_1$ – the same wave, times a number. In general the shift multiplies $\mathbf{f}_k$ by $(-i)^k$.

Fourier Coefficients: How Much of Each Wave

The k th **Fourier coefficient** of $\mathbf{x}$ measures how much of wave k it contains:

$$(\mathcal{F}\mathbf{x})_k = \sum_j x_j (-i)^{jk}, \quad \mathcal{F} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}$$

k	$(\mathcal{F}\mathbf{u})_k$ for $\mathbf{u} = (1, 2, 0, 1)$	$(\mathcal{F}\mathbf{v})_k$ for $\mathbf{v} = (2, 0, 1, 1)$
0	$1 + 2 + 0 + 1 = 4$	$2 + 0 + 1 + 1 = 4$
1	$1 - 2i - 0 + i = 1 - i$	$2 - 0 - 1 + i = 1 + i$
2	$1 - 2 + 0 - 1 = -2$	$2 - 0 + 1 - 1 = 2$
3	$1 + 2i - 0 - i = 1 + i$	$2 + 0 - 1 - i = 1 - i$

Coefficient 0 is the plain sum, coefficient 2 the alternating sum. For a real vector, coefficient 3 is the conjugate of coefficient 1: four real numbers in, four real numbers' worth out. **Nothing is lost** – $\mathbf{x} = \frac{1}{4} \sum_k (\mathcal{F}\mathbf{x})_k \mathbf{f}_k$ puts the waves back together.

Rotation Becomes Multiplication

Since each wave survives a rotation intact, rotating $\mathbf{x}$ one place right multiplies coefficient k by $(-i)^k$. Our $\mathbf{u}$ happens to be $\mathbf{v}$ rotated one place right, so:

$$\mathcal{F}\mathbf{v} \odot (1, -i, -1, i) = (4, (1+i)(-i), -2, (1-i)i) = (4, 1-i, -2, 1+i) = \mathcal{F}\mathbf{u}$$

Convolution is a weighted sum of rotations: $\mathbf{u} \circledast \mathbf{v} = \sum_s u_s (\mathbf{v} \text{ rotated } s \text{ places})$. Each rotation multiplies coefficient k by $(-i)^{sk}$, so the whole sum multiplies it by $\sum_s u_s (-i)^{sk}$ – which is exactly $(\mathcal{F}\mathbf{u})_k$.

The convolution theorem

$$\mathcal{F}(\mathbf{u} \circledast \mathbf{v}) = \mathcal{F}\mathbf{u} \odot \mathcal{F}\mathbf{v}$$

Check: $\mathcal{F}\mathbf{u} \odot \mathcal{F}\mathbf{v} = (16, (1-i)(1+i), -4, (1+i)(1-i)) = (16, 2, -4, 2)$, and that is $\mathcal{F}(4, 5, 2, 5)$.

In Fourier coordinates, convolution acts one coordinate at a time. That answers the puzzle.

What the Fourier Picture Buys

Speed

The **fast Fourier transform** (Cooley & Tukey 1965) computes all n coefficients in about $n \log_2 n$ steps instead of n^2 . A convolution is then two transforms, n products, and one transform back. At $n = 512$ that is about 4,600 steps per transform, against 262,144 multiplications done directly: **the fold of an outer product, without the matrix and without the quadratic cost.**

Unbinding becomes division

Given $\mathbf{w} = \mathbf{u} \circledast \mathbf{v} = (4, 5, 2, 5)$ and the role $\mathbf{u}$, divide coefficient by coefficient:

$$\mathcal{F}\mathbf{w} \oslash \mathcal{F}\mathbf{u} = \left(\frac{16}{4}, \frac{2}{1-i}, \frac{-4}{-2}, \frac{2}{1+i} \right) = (4, 1+i, 2, 1-i) = \mathcal{F}\mathbf{v}$$

then transform back: $\mathbf{v} = (2, 0, 1, 1)$, **exactly**. The vector that does this, $\mathbf{u}^\dagger = \mathcal{F}^{-1}(1 \oslash \mathcal{F}\mathbf{u}) = \frac{1}{16}(3, -1, -5, 7)$, is the **exact inverse**: $\mathbf{u} \circledast \mathbf{u}^\dagger = (1, 0, 0, 0)$.

The catch: division needs every coefficient of $\mathbf{u}$ to be nonzero, and small ones cause trouble (Part III).

To divide by $1 - i$, multiply top and bottom by its conjugate: $\frac{2}{1-i} = \frac{2(1+i)}{2} = 1 + i$.

Part III

The Adjoint, and Plate's Decoder

Running the Fold Backward: Its Adjoint

The fold takes a matrix to a vector. The natural way back is its **adjoint**: put c_k into every cell of group k .

$\text{fold}^*(1, 2, 3, 4)$

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

$$\text{fold}^*(\mathbf{c})_{ij} = c_{(i+j) \bmod n}$$

Why “adjoint”. For every M and every $\mathbf{c}$,

$$\langle \text{fold}(M), \mathbf{c} \rangle = \sum_k c_k \sum_{i+j \equiv k} M_{ij} = \sum_{i,j} M_{ij} c_{(i+j) \bmod n} = \langle M, \text{fold}^*(\mathbf{c}) \rangle$$

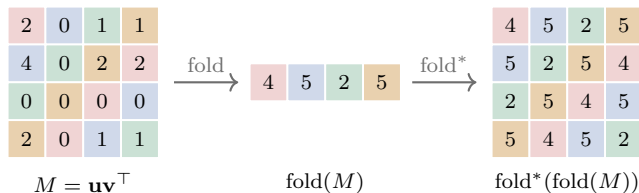
Is it an inverse? Not quite. The next two slides apply it to $\text{fold}(\mathbf{u}\mathbf{v}^\top)$ and look at what comes back.

The fold squeezes a matrix into n numbers; its adjoint builds a matrix out of n numbers. Machine learning uses both.

The Adjoint Applied to $\text{fold}(M)$, Step by Step

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Start from $M = \mathbf{u}\mathbf{v}^\top$, fold it, then apply the adjoint to the result:



Step 1. M has 16 entries. **Step 2.** The fold adds up each color: (4, 5, 2, 5). **Step 3.** The adjoint writes each sum back into every cell of that color.

The colors survive the round trip; the numbers do not. The blue cells of M held 0, 4, 0 and 1; every blue cell now holds 5, their total. Each cell carries $n = 4$ times its group's average.

What the Round Trip Keeps, and What It Loses

Divide by n and compare with the original: $P(M) = \frac{1}{4} \text{fold}^*(\text{fold}(M))$ replaces each cell by its group's average.

2	0	1	1
4	0	2	2
0	0	0	0
2	0	1	1

M

1	1.25	0.5	1.25
1.25	0.5	1.25	1
0.5	1.25	1	1.25
1.25	1	1.25	0.5

$P(M)$

The blue group in detail. M holds 0, 4, 0, 1, total 5. $P(M)$ holds 1.25 four times, total 5. The difference is $-1.25, 2.75, -1.25, -0.25$, which totals 0.

Every group's difference totals zero, so $\text{fold}(M - P(M)) = \mathbf{0}$: the fold cannot see the difference at all.

The two pieces are at right angles, so their squared lengths add: $\|M\|^2 = 36 = 17.5 + 18.5$, kept plus lost.

Undoing from one side only. $\text{fold}(\frac{1}{n} \text{fold}^*(c)) = c$ for every c , but $\frac{1}{n} \text{fold}^*(\text{fold}(M)) = P(M) \neq M$. So $\frac{1}{n} \text{fold}^*$ is the fold's **pseudo-inverse**: the best reconstruction possible from four numbers – and still not M .

From the Adjoint to Plate's Decoder

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Binding with a known role **u** is a linear map of the filler: $\mathbf{b} \mapsto \mathbf{u} \circledast \mathbf{b} = \text{fold}(\mathbf{u}\mathbf{b}^\top)$. Its adjoint spreads the trace out with fold^* , then contracts with the role: $\mathbf{c} \mapsto \text{fold}^*(\mathbf{c}) \mathbf{u}$.

Entry by entry, $(\text{fold}^*(\mathbf{c}) \mathbf{u})_j = \sum_i c_{(i+j) \bmod n} u_i = (\mathbf{u} \star \mathbf{c})_j$ – circular correlation with **u**, which is convolution with the involution $\mathbf{u}^* = (u_0, u_3, u_2, u_1) = (1, 1, 0, 2)$. That is Plate's decoder.

$$\text{fold}^*(4, 5, 2, 5) \mathbf{u} = \begin{pmatrix} 4 & 5 & 2 & 5 \\ 5 & 2 & 5 & 4 \\ 2 & 5 & 4 & 5 \\ 5 & 4 & 5 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 19 \\ 13 \\ 17 \\ 15 \end{pmatrix}$$

Compare $6\mathbf{v} = (12, 0, 6, 6)$: the answer is $6\mathbf{v}$ plus crosstalk $(7, 13, 11, 9)$. TPR decodes by applying the transpose to M itself, $M^\top \mathbf{u} = \|\mathbf{u}\|^2 \mathbf{v} = 6\mathbf{v}$, exactly; HRR can only apply it to the matrix rebuilt from the fold.

The shared star is no accident: binding with **u** is multiplication by the circulant $C(\mathbf{u})$, and $C(\mathbf{u})^\top = C(\mathbf{u}^*)$.

Inverse or Adjoint? Ask the Fourier Coefficients

The exact inverse *divides* each coefficient by $(\mathcal{F}\mathbf{u})_k$. The involution *multiplies* by its conjugate, $\overline{(\mathcal{F}\mathbf{u})_k}$.
For $\mathbf{u} = (1, 2, 0, 1)$:

k	$(\mathcal{F}\mathbf{u})_k$	exact inverse: $1/(\mathcal{F}\mathbf{u})_k$	involution: $\overline{(\mathcal{F}\mathbf{u})_k}$	ratio $ (\mathcal{F}\mathbf{u})_k ^2$
0	4	$1/4$	4	16
1	$1 - i$	$(1 + i)/2$	$1 + i$	2
2	-2	$-1/2$	-2	4
3	$1 + i$	$(1 - i)/2$	$1 - i$	2

Same directions, different sizes: the involution is the exact inverse with coefficient k reweighted by $|(\mathcal{F}\mathbf{u})_k|^2$. So decoding with it returns $\mathbf{v}$'s coefficients reweighted:

$\mathcal{F}(\mathbf{u}^* \circledast \mathbf{w}) = |\mathcal{F}\mathbf{u}|^2 \odot \mathcal{F}\mathbf{v} = (64, 2+2i, 8, 2-2i)$, which transforms back to $(19, 13, 17, 15)$.

If every $|(\mathcal{F}\mathbf{u})_k|^2$ were equal, the reweighting would be a constant, and the involution would be the inverse up to scale. Plate calls such vectors **unitary**: "Under this condition the approximate inverse is equal to the exact inverse."

Why Plate Chose the Noisy One

Plate (1995): “using the exact inverse results in a lower signal-to-noise ratio in the retrieved vector when the memory trace is noisy.” A trace holding several bound pairs *is* noisy. Random vectors, $d = 2048$, decoding the first of K superposed pairs:

pairs in the trace, K	1	2	3	5	10
cosine with the true filler: involution	0.71	0.58	0.50	0.41	0.30
cosine with the true filler: exact inverse	1.00	0.33	0.24	0.17	0.12

Averages over 300 trials. An unrelated vector scores 0.00 ± 0.02 .

Why. The exact inverse divides everything at frequency k by $(\mathcal{F}\mathbf{u})_k$ – the other pairs included. Where that coefficient is small, it multiplies their crosstalk by a large number. The involution never divides.

Why 0.71 is enough. For a random vector the $|(\mathcal{F}\mathbf{u})_k|^2$ average to a constant but vary from one k to the next, so the involution decodes one pair to cosine ≈ 0.71 , and K pairs to about $1/\sqrt{K+1}$ – still far above an unrelated vector, which is all cleanup memory needs.

Part IV

Two Things a Network Can Compress

Representations or Parameters?

A network holds $n \times n$ objects of two quite different kinds.

	a representation	a set of parameters
what it is	computed fresh for every input	learned once, reused for every input
example	a bound pair $\mathbf{r}\mathbf{f}^\top$; a bilinear feature $\mathbf{x}\mathbf{y}^\top$	a weight matrix W in a layer $\mathbf{y} = W\mathbf{x}$
compressed by	fold – squeeze n^2 numbers to n	fold* – build W from n numbers
price of a collision	a noisy comparison or retrieval	fewer distinct weights to learn

The binding deck only ever compressed representations. *Most of what is called model compression is about parameters.* The fold covers the first kind, and its adjoint the second.

Where $n \times n$ Objects Appear in Practice

setting	the $n \times n$ object	kind
fine-grained image recognition	pooled image features $\sum_{\ell} \mathbf{x}_{\ell} \mathbf{x}_{\ell}^{\top}$	representation
visual question answering	question vector $\otimes$ image vector	representation
knowledge-graph completion	one matrix W_r per relation, in $\mathbf{e}_s^{\top} W_r \mathbf{e}_o$	parameters
any fully connected layer	its weight matrix W	parameters
self-attention	keys $\otimes$ values, summed over the sequence	representation

Each of these has been compressed with some version of the fold or its adjoint. [We take them in that order.](#)

The knowledge-graph score $\mathbf{e}_s^{\top} W_r \mathbf{e}_o$ is a bilinear form, exactly as in Session 10 – with one learned matrix W_r for each relation.

Part V

Compressing a Representation: Tensor Sketch

The Problem: Bilinear Features Are Huge

Bilinear pooling (Lin, RoyChowdhury & Maji 2015). A convolutional network produces a 512-dimensional feature $\mathbf{x}_\ell$ at every image location ℓ . Pool the outer products:

$$B = \sum_{\ell} \mathbf{x}_\ell \mathbf{x}_\ell^\top \in \mathbb{R}^{512 \times 512}$$

Entry B_{ij} records how strongly features i and j fire *together*. That is why it works well for fine-grained recognition: telling 200 bird species apart depends on which textures occur with which colors.

The cost

$512 \times 512 = 262,144$ numbers per image, fed to a classifier – which needs a weight for each of them, for every class.

The question: can we keep the pairwise interactions and drop to a few thousand numbers?

First Try: Just Fold It

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

The fold is cheap: $\text{fold}(\mathbf{x}\mathbf{y}^\top) = \mathbf{x} \circledast \mathbf{y}$, by FFT. But it has two problems.

- 1. The output size is stuck at n .** A 512-dimensional feature folds to 512 numbers, no more and no fewer.
- 2. It distorts comparisons.** Compare $\mathbf{u}\mathbf{v}^\top$ with its transpose $\mathbf{v}\mathbf{u}^\top$:

	inner product	squared norm	cosine
the matrices themselves	$(\mathbf{u} \cdot \mathbf{v})^2 = 9$	$\ \mathbf{u}\ ^2 \ \mathbf{v}\ ^2 = 36$	0.25
their folds	70	70	1

Both folds are $(4, 5, 2, 5)$, so the fold reports two different matrices as identical. **The cause:** within a group, cells are added with the same sign, so $\langle \text{fold}M, \text{fold}N \rangle$ is $\langle M, N \rangle$ plus cross terms between different cells of a group – and nothing makes those cross terms small.

Fix One: Random Signs

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Give every coordinate a random sign: $\mathbf{s}_1 \in \{\pm 1\}^n$ for the first vector, $\mathbf{s}_2$ for the second. Cell (i, j) now enters its group with sign $s_1(i) s_2(j)$.

s_1	+	-	-	+	s_2
+	2	0	-1	1	
+	4	0	-2	2	
-	0	0	0	0	
-	-2	0	1	-1	

signed cells $s_1(i)s_2(j)u_iv_j$

With $\mathbf{s}_1 = (+, +, -, -)$ and $\mathbf{s}_2 = (+, -, -, +)$, adding each color gives

$$\text{fold}_{\pm}(\mathbf{u}\mathbf{v}^{\top}) = (4, 5, -2, -3)$$

and the same signs on the transpose give

$$\text{fold}_{\pm}(\mathbf{v}\mathbf{u}^{\top}) = (4, -5, -2, 3)$$

They are no longer equal. And it is still one convolution:
 $\text{fold}_{\pm}(\mathbf{u}\mathbf{v}^{\top}) = (\mathbf{s}_1 \odot \mathbf{u}) \circledast (\mathbf{s}_2 \odot \mathbf{v})$.

Why the Signs Work: Cross Terms Cancel on Average

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Take any one group. Two different cells in it never share a row or a column: if $i + j \equiv i' + j'$ and $i = i'$, then $j = j'$.

So the signs of two different cells, $s_1(i)s_2(j)$ and $s_1(i')s_2(j')$, involve four separate coin flips, and **every cross term averages to zero**:

$$\mathbb{E}[\langle \text{fold}_{\pm}(M), \text{fold}_{\pm}(N) \rangle] = \langle M, N \rangle$$

$M = \mathbf{uv}^{\top}, N = \mathbf{vu}^{\top}$	true value	plain fold	average over all 256 sign patterns
inner product	9	70	9
squared norm of M	36	70	36

Unbiased is not exact. Single sign patterns give inner products anywhere from -34 to 70 ; the averaging pays off when there are many coordinates, as there are at $n = 512$.

Fix Two: Random Buckets

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Also hash each coordinate to a bucket, $h_1, h_2 : \{0, \dots, n-1\} \rightarrow \{0, \dots, D-1\}$. Cell (i, j) goes to bucket $(h_1(i) + h_2(j)) \bmod D$, still with sign $s_1(i)s_2(j)$. **Now the output size D is ours to choose.**

h_1	1	0	2	1	h_2
0	$\begin{matrix} 1 \\ 2 \end{matrix}$	$\begin{matrix} 0 \\ 0 \end{matrix}$	$\begin{matrix} 2 \\ -1 \end{matrix}$	$\begin{matrix} 1 \\ 1 \end{matrix}$	
2	$\begin{matrix} 0 \\ 4 \end{matrix}$	$\begin{matrix} 2 \\ 0 \end{matrix}$	$\begin{matrix} 1 \\ -2 \end{matrix}$	$\begin{matrix} 0 \\ 2 \end{matrix}$	
1	$\begin{matrix} 2 \\ 0 \end{matrix}$	$\begin{matrix} 1 \\ 0 \end{matrix}$	$\begin{matrix} 0 \\ 0 \end{matrix}$	$\begin{matrix} 2 \\ 0 \end{matrix}$	
2	$\begin{matrix} 0 \\ -2 \end{matrix}$	$\begin{matrix} 2 \\ 0 \end{matrix}$	$\begin{matrix} 1 \\ 1 \end{matrix}$	$\begin{matrix} 0 \\ -1 \end{matrix}$	

colored by bucket; signs as before

With $D = 3$, $h_1 = (0, 2, 1, 2)$, $h_2 = (1, 0, 2, 1)$, and the same signs as before, the three buckets sum to

$$\text{TS}(\mathbf{u} \otimes \mathbf{v}) = (3, 2, -1)$$

The groups are no longer anti-diagonals: they are scattered, and of unequal size (6, 5, 5 cells).

Switch the randomness off – each h the identity, every sign $+$ – and the bucket of (i, j) is $(i + j) \bmod n$: the fold.

Tensor Sketch: Still One Convolution

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

The **count sketch** of a single vector (Charikar, Chen & Farach-Colton 2002) uses one hash and one sign: $\text{CS}(\mathbf{x})_b = \sum_{i: h(i)=b} s(i) x_i$.

Tensor sketch (Pham & Pagh 2013)

The bucketed, signed sum of the last slide is a circular convolution of two count sketches, in $\mathbb{R}^D$:

$$\text{TS}(\mathbf{x} \otimes \mathbf{y}) = \text{CS}_1(\mathbf{x}) \circledast \text{CS}_2(\mathbf{y})$$

Check. $\text{CS}_1(\mathbf{u}) = (1, 0, 1)$ and $\text{CS}_2(\mathbf{v}) = (0, 3, -1)$, and $(1, 0, 1) \circledast (0, 3, -1) = (3, 2, -1)$ – the same three bucket sums.

Cost. $O(n + D \log D)$ with the FFT; the $n \times n$ matrix is never formed.

Accuracy. A cancellation argument like the last one gives $\mathbb{E} \langle \text{TS}(\mathbf{x} \otimes \mathbf{y}), \text{TS}(\mathbf{x}' \otimes \mathbf{y}') \rangle = \langle \mathbf{x}, \mathbf{x}' \rangle \langle \mathbf{y}, \mathbf{y}' \rangle$.

The Fold Is Tensor Sketch with the Randomness Switched Off

	bucket of (i, j)	sign	output size	comparisons	order
fold	$(i + j) \bmod n$	+	n	biased	lost
signed fold	$(i + j) \bmod n$	random	n	unbiased	kept*
tensor sketch	$(h_1(i) + h_2(j)) \bmod D$	random	any D	unbiased	kept*

* Kept because the two factors get *independent* signs. With $s_2 = \pm s_1$ the signed fold is blind to the transpose again – for our $\mathbf{u}$ and $\mathbf{v}$, those are exactly the 32 of 256 sign patterns that lose the order.

The fold from the binding deck is the deterministic, size- n corner of a family that machine learning uses at any size. The random signs remove its bias, the random buckets free its output size, and independence between the two factors restores the order that fact 3 said was lost.

Compact Bilinear Pooling

Gao, Beijbom, Zhang & Darrell (CVPR 2016) replace the 262,144-dimensional bilinear feature with a tensor sketch of $D = 8,192$ – 32 times smaller.

CUB-200 birds, VGG-D features	dimensions	error before fine-tuning	after
full bilinear pooling	262,144	19.90%	16.0%
tensor sketch	8,192	20.50%	16.0%

They describe the compact representations as having “the same discriminative power as the full bilinear representation,” at a few thousand dimensions.

Gradients pass through the FFTs, so the whole network still trains end to end. As in any count sketch, the hashes and signs are random and fixed in advance.

A Language Example: Visual Question Answering

Fukui, Park, Yang, Rohrbach, Darrell & Rohrbach (EMNLP 2016). A question such as What color is the umbrella? is encoded by an LSTM into a 2048-dimensional vector $\mathbf{q}$; the image, by a ResNet, into a 2048-dimensional vector $\mathbf{x}$.

How should $\mathbf{q}$ and $\mathbf{x}$ be combined? Their hypothesis: adding or concatenating them is “not as expressive as an outer product of the visual and textual vectors.”

$\mathbf{q} \otimes \mathbf{x}$

a full bilinear classifier over 3,000 answers

multimodal compact bilinear pooling (MCB)

$2048 \times 2048 \approx 4.2$ million numbers

about 12.5 billion parameters (their estimate)

a tensor sketch of 16,000 numbers

At publication it outperformed the previous state of the art on the VQA challenge and on Visual7W. [It is the operation the binding deck used to bind a role to a filler, here binding a question to an image](#) – with the randomness switched on.

What the Sketch Costs

Collisions

With n^2 cells and D buckets, each bucket receives about n^2/D of them – $262,144/8,192 = 32$ cells per bucket in compact bilinear pooling. Each bucket is a noisy sum; the noise averages out across many buckets, not within one.

Fixed randomness

In the standard method the hashes and signs are part of the architecture, fixed before training rather than learned – like the stipulated role vectors of HRR.

Order, by construction

The sketch distinguishes $\mathbf{x} \otimes \mathbf{y}$ from $\mathbf{y} \otimes \mathbf{x}$, but only because the two factors were given different hashes and signs. In MCB that is harmless – a question and an image are never confused – but nothing in the sketch *learned* the difference.

Part VI

Asymmetric Relations: Holographic Embeddings

Knowledge Graphs, and a Bilinear Form per Relation

A knowledge graph stores facts as triples (subject, relation, object); the task is to score the triples that are missing. Two standard benchmarks: **WN18**, a subset of WordNet, and **FB15k**, a subset of Freebase.

dog	hypernym	canine
finger	part_of	hand
big	similar_to	large

RESCAL (Nickel, Tresp & Kriegel 2011) scores a triple with a bilinear form, exactly as in Session 10:

$$\text{score}(s, r, o) = \mathbf{e}_s^\top W_r \mathbf{e}_o, \quad W_r \in \mathbb{R}^{d \times d}$$

One full matrix per relation: at $d = 200$, that is 40,000 numbers for hypernym alone. **The puzzle:** get down to d numbers per relation without losing direction – a dog is a canine; a canine is not a dog.

First Try: Fold the Pair

The obvious compression: bind subject to object with the fold, and score the result against a relation vector $\mathbf{r} \in \mathbb{R}^d$:

$$\text{score}(s, r, o) = \mathbf{r} \cdot (\mathbf{e}_s \circledast \mathbf{e}_o) \quad - d \text{ numbers per relation}$$

But fact 3 says $\mathbf{e}_s \circledast \mathbf{e}_o = \mathbf{e}_o \circledast \mathbf{e}_s$. So for every relation vector whatsoever,

$$\text{score}(\text{dog}, \text{hypernym}, \text{canine}) = \text{score}(\text{canine}, \text{hypernym}, \text{dog})$$

Every relation comes out symmetric. That is fine for similar_to and useless for hypernym or part_of.

The fold has thrown away exactly the information a hypernym needs – the antisymmetric part of $\mathbf{e}_s \mathbf{e}_o^\top$, which records which argument came first.

HolE: Correlation Instead of Convolution

Holographic embeddings (Nickel, Rosasco & Poggio, AAAI 2016) use **circular correlation**:

$$[\mathbf{a} \star \mathbf{b}]_k = \sum_i a_i b_{(k+i) \bmod d}, \quad \text{score}(s, r, o) = \sigma(\mathbf{r} \cdot (\mathbf{e}_s \star \mathbf{e}_o))$$

In their words, circular correlation “can be interpreted as a compression of the tensor product”; and “Correlation, unlike convolution, is not commutative,” which they note “is necessary to model asymmetric relations.”

	WN18 MRR	FB15k MRR	parameters on FB15k
RESCAL, $d \times d$ per relation	0.890	0.354	32.5M ($d = 150$)
HolE, d per relation	0.938	0.524	3.3M ($d = 200$)

MRR: mean reciprocal rank of the correct entity, filtered; higher is better. Figures from the paper.

Correlation Is a Fold Too – Along the Other Diagonals

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

In $[\mathbf{a} \star \mathbf{b}]_k$, cell (i, j) of $\mathbf{ab}^\top$ contributes when $j - i \equiv k$. Color $\mathbf{uv}^\top$ that way:

	0	1	2	3
0	2 ⁰	0 ¹	1 ²	1 ³
1	4 ³	0 ⁰	2 ¹	2 ²
2	0 ²	0 ³	0 ⁰	0 ¹
3	2 ¹	0 ²	1 ³	1 ⁰

label $(j - i) \bmod 4$

The groups are now the wrapped *diagonals*, running down and to the right ($\searrow$), where the fold's ran down and to the left ($\swarrow$). Write

$$\text{fold}_{\searrow}(M)_k = \sum_{j-i \equiv k} M_{ij}, \quad \mathbf{a} \star \mathbf{b} = \text{fold}_{\searrow}(\mathbf{ab}^\top)$$

Adding each color: $\mathbf{u} \star \mathbf{v} = (3, 4, 3, 6)$.

It is the old fold after reversing the first vector:

$\mathbf{a} \star \mathbf{b} = \mathbf{a}^* \circledast \mathbf{b}$, with $a_i^* = a_{-i}$ – the binding deck's involution.

The Transpose Test

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

What happens to each fold when the matrix is transposed? Transposing sends cell (i, j) to (j, i) .

The fold, ↙

The label $i + j$ becomes $j + i$: **unchanged**.

$$\text{fold}(M^\top) = \text{fold}(M)$$

$$\mathbf{u} \circledast \mathbf{v} = \mathbf{v} \circledast \mathbf{u} = (4, 5, 2, 5)$$

Correlation, ↘

The label $j - i$ becomes $i - j$: **negated**.

$$\text{fold}_{\searrow}(M^\top)_k = \text{fold}_{\searrow}(M)_{-k}$$

$$\mathbf{u} \star \mathbf{v} = (3, 4, 3, 6), \quad \mathbf{v} \star \mathbf{u} = (3, 6, 3, 4)$$

Swapping the arguments of a correlation does not destroy anything: it reverses the output, $k \mapsto -k$, which here swaps entries 1 and 3. Which argument came first is still there to be read.

Which Half of the Matrix Survives

index	0	1	2	3
u	1	2	0	1
v	2	0	1	1

Split $\mathbf{uv}^\top = S + A$ into its symmetric and antisymmetric parts, and fold each.

	symmetric part S	antisymmetric part A	total
fold, $\swarrow$	(4, 5, 2, 5)	(0, 0, 0, 0)	(4, 5, 2, 5)
fold $\searrow$, $\swarrow$	(3, 5, 3, 5)	(0, -1, 0, 1)	(3, 4, 3, 6)

Convolution keeps only the symmetric half. Correlation keeps both, in separate places: S lands in the *even* part of the output (unchanged by $k \mapsto -k$), A in the *odd* part (negated by it).

[Back to language.](#) The difference $\mathbf{r} \cdot (\mathbf{e}_s \star \mathbf{e}_o) - \mathbf{r} \cdot (\mathbf{e}_o \star \mathbf{e}_s)$ depends only on the odd part of $\mathbf{r}$. A relation with an even $\mathbf{r}$ is symmetric – similar_to in WordNet, spouse in Freebase – while hypernym and part_of need an odd part.

Part VII

Compressing the Weights: Structured Layers

The Other Direction: Building Weights from n Numbers

A full layer $\mathbf{y} = W\mathbf{x}$ with $W \in \mathbb{R}^{n \times n}$ stores n^2 numbers and costs n^2 multiplications. Instead, build W from n numbers $\mathbf{c}$ – with the fold's adjoint, or with its close relative, the **circulant**:

c_0	c_1	c_2	c_3
c_1	c_2	c_3	c_0
c_2	c_3	c_0	c_1
c_3	c_0	c_1	c_2

fold $^*(\mathbf{c})$: entry $c_{(i+j) \bmod 4}$ constant along $\swarrow$

c_0	c_3	c_2	c_1
c_1	c_0	c_3	c_2
c_2	c_1	c_0	c_3
c_3	c_2	c_1	c_0

circulant $C(\mathbf{c})$: entry $c_{(i-j) \bmod 4}$ constant along $\searrow$

Same color, same parameter. The two differ only by reversing the input: fold $^*(\mathbf{c}) \mathbf{x} = C(\mathbf{c}) \mathbf{x}^*$ – the reversal step of the binding deck's convolution recipe, once again.

A Circulant Layer Is a Convolution

Row 0 of $C(\mathbf{c})$ is $\mathbf{c}$ reversed and rotated once, and each later row is rotated once more – exactly the strips of the binding deck. So multiplying by it convolves with $\mathbf{c}$:

$$C(\mathbf{c}) \mathbf{x} = \mathbf{c} \circledast \mathbf{x}$$

Check, with $\mathbf{c} = (1, 2, 3, 4)$ and $\mathbf{x} = (1, 0, 2, 1)$:

$$\begin{pmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 13 \\ 9 \\ 9 \end{pmatrix} = \mathbf{c} \circledast \mathbf{x}$$

Cheng, Yu, Feris, Kumar, Choudhary & Chang (ICCV 2015) replace the projection in fully connected layers with a circulant one: space falls from $O(d^2)$ to $O(d)$, and time from $O(d^2)$ to $O(d \log d)$.

Why the FFT Helps: A Circulant Is Diagonal in the Right Basis

The **convolution theorem** of Part II: the Fourier transform $\mathcal{F}$ turns convolution into coordinate-wise multiplication,

$$\mathcal{F}(\mathbf{c} \circledast \mathbf{x}) = (\mathcal{F}\mathbf{c}) \odot (\mathcal{F}\mathbf{x}) \quad \implies \quad C(\mathbf{c}) = \mathcal{F}^{-1} \text{diag}(\mathcal{F}\mathbf{c}) \mathcal{F}$$

Check, with the same $\mathbf{c}$ and $\mathbf{x}$:

$$\begin{array}{ll} \mathcal{F}\mathbf{c} & (10, -2+2i, -2, -2-2i) \\ \mathcal{F}\mathbf{x} & (4, -1+i, 2, -1-i) \\ \text{product} & (40, -4i, -4, 4i) = \mathcal{F}(9, 13, 9, 9) \end{array}$$

A dense-looking $n \times n$ block is *diagonal* once the basis is changed, and the FFT changes basis in $O(n \log n)$.

Keep this in mind for Part IX. Here “diagonal” costs nothing beyond the circulant constraint itself: no entry is set to zero, only the basis changes.

Block-Circulant Layers

A single circulant for a whole layer is a strong restriction. **CirCNN** (Ding et al., MICRO 2017) takes a middle road: cut W into $k \times k$ blocks and make *each block* circulant, with its own vector.

0	3	2	1	0	3	2	1
1	0	3	2	1	0	3	2
2	1	0	3	2	1	0	3
3	2	1	0	3	2	1	0
0	3	2	1	0	3	2	1
1	0	3	2	1	0	3	2
2	1	0	3	2	1	0	3
3	2	1	0	3	2	1	0

four circulant 4×4 blocks:
16 numbers instead of 64

Per block: k numbers instead of k^2 , and a product in $O(k \log k)$ by FFT.

Reported: weight storage in fully connected layers cut by $400\times$ to more than $4000\times$, and $6\text{--}102\times$ better energy efficiency than the prior state of the art – their figures, the upper end with very low-precision arithmetic.

The weights are trained in this form from the start, rather than compressed after training.

Random Groups: HashedNets

Chen, Wilson, Tyree, Weinberger & Chen (ICML 2015) keep a short vector $\mathbf{w}$ of real weights and define every entry of a large “virtual” matrix by hashing:

$$V_{ij} = \xi(i, j) w_{h(i, j)}$$

with h a random bucket and ξ a random sign – [the same two ingredients as tensor sketch](#), now applied to parameters instead of representations.

	bucket $h(i, j)$	sign $\xi(i, j)$
anti-circulant layer, $V = \text{fold}^*(\mathbf{w})$	$(i + j) \bmod n$	+
HashedNets	random	random

The anti-circulant layer is the special case with the randomness switched off – the same relationship as between the fold and tensor sketch, on the other side of the adjoint.

Training a Shared-Weight Layer Folds the Gradient

Each real weight w_b appears in many cells, so its gradient collects from all of them. By the chain rule,

$$\frac{\partial L}{\partial w_b} = \sum_{(i,j) : h(i,j)=b} \xi(i,j) \frac{\partial L}{\partial V_{ij}}$$

which is the gradient HashedNets computes. With $h(i,j) = (i+j) \bmod n$ and $\xi \equiv +1$, this is

$$\frac{\partial L}{\partial \mathbf{w}} = \text{fold} \left(\frac{\partial L}{\partial V} \right)$$

The forward pass uses fold^* to build the weights; the backward pass uses fold to gather their gradient. That is a general rule: backpropagating through a linear map applies its adjoint.

Checked numerically for $V = \text{fold}^*(\mathbf{c})$ and $L = \|V\mathbf{x} - \mathbf{y}\|^2$: finite differences agree with $\text{fold}(\partial L / \partial V)$ to six decimal places.

Part VIII

Folding Inside Attention

Attention's Quadratic Objects

Self-attention over a sequence of T tokens compares every query with every key:

$$\text{softmax}(QK^\top) V, \quad QK^\top \in \mathbb{R}^{T \times T}$$

Quadratic in sequence length – the bottleneck for long documents, audio, or raw byte sequences.

Linear attention (Katharopoulos, Vyas, Pappas & Fleuret, ICML 2020) replaces the softmax with a feature map ϕ and multiplies in the other order (up to a normalizing term):

$$\phi(Q) (\phi(K)^\top V), \quad \phi(K)^\top V = \sum_t \phi(\mathbf{k}_t) \mathbf{v}_t^\top \in \mathbb{R}^{H \times H}$$

Independent of T – but still a *sum of outer products*, H^2 numbers per head. [The puzzle: fold that too?](#)

HRR Attention

Hrrformer (Alam, Raff, Biderman, Oates & Holt, ICML 2023) binds each key to its value with circular convolution and superposes all T pairs into one vector:

$$\boldsymbol{\beta} = \sum_{t=1}^T \mathbf{k}_t \circledast \mathbf{v}_t \in \mathbb{R}^H$$

Since the fold is linear, $\boldsymbol{\beta} = \sum_t \text{fold}(\mathbf{k}_t \mathbf{v}_t^\top) = \text{fold}(K^\top V)$: the $H \times H$ matrix of linear attention (with ϕ the identity), folded to H numbers.

Reading it out. Each query unbinds, $\hat{\mathbf{v}}_t = \mathbf{q}_t^\dagger \circledast \boldsymbol{\beta}$, with $\mathbf{q}_t^\dagger$ the exact inverse of $\mathbf{q}_t$, computed by dividing Fourier coefficients as in Part II; then

$$a_t = \cos(\mathbf{v}_t, \hat{\mathbf{v}}_t), \quad \mathbf{w} = \text{softmax}(a_1, \dots, a_T), \quad \text{output}_t = w_t \mathbf{v}_t$$

$O(TH \log H)$ time and $O(TH)$ space: the $T \times T$ matrix is never formed.

What HRR Attention Buys, and What It Costs

Reported (Alam et al. 2023)

On the Long Range Arena benchmark: accuracy near the state of the art, up to $280\times$ faster training, and about 79% less memory than the best prior method. On EMBER malware detection: sequences of length $T = 131,072$, with a single layer.

The binding deck's cost, again

Unbinding from a superposition of T pairs returns the wanted value plus crosstalk from all the others. The softmax acts as clean-up – the collision problem, now with every token in the sequence sharing one vector.

A different shape of output

Each position keeps its *own* value, rescaled by w_t ; positions interact only through β , on which every weight depends. Standard attention instead mixes the values of other positions.

Part IX

Other Things Called “Folding”

Four Techniques, One Word

Search for “folding” in model compression and these come up. **None of them is the fold of this course;** the shared word is a coincidence of naming.

name	source	what it does	kind
batch-norm folding	standard in quantized deployment (e.g. Jacob et al., CVPR 2018)	merges a batch-norm layer's scale and shift into the preceding layer's weights and bias	exact rewrite
folding attention	Li et al. 2023, on-device speech recognition	splits each token into N sub-tokens of D/N channels, so each projection shrinks to $\frac{D}{N} \times \frac{D}{N}$	reshape
model folding	Wang, Šikić, Thiele & Saukh, ICLR 2025	clusters the rows of each weight matrix with k -means and replaces each cluster by its centroid – no data, no fine-tuning	merging
weight matrix folding	Belhaouari & Kraidia, <i>Sci. Rep.</i> 2025	replaces the query and key weights with replicated-row matrices, and keeps only the diagonal of the value weights	selection

Question 2 sorts them: what happens to each cell of the $n \times n$ object?

Weight Matrix Folding, Up Close

Proposed for compressing self-attention in large language models (Belhaouari & Kraidia, *Scientific Reports* 15:10171, 2025), and tested on GPT-2. Two parts:

Identical Row Compression – for W_Q and W_K

One row is kept and copied across the matrix, as in $W'_Q = \mathbf{q} \mathbf{1}^\top$. The paper's text and equations differ on which way the copying runs; [either way the result has rank 1](#).

Diagonal Weight Compression – for W_V

Only the diagonal is stored: $W'_V = \text{diag}(v_1, \dots, v_d)$.

Reported: transformer layers compressed by 99% and linear layers by 70%, about 70% overall, with “nearly the same accuracy” on the authors’ GPT-2 benchmarks.

In the terms of Part I, both parts *select*: they keep d of the d^2 numbers, and every other entry becomes a copy or a zero. Compare the circulant of Part VII, which is diagonal only after a change of basis.

What Rank-1 Query and Key Weights Do to Attention

If W_Q has rank 1, so does every score matrix it produces: $QK^\top = XW_QW_K^\top X^\top$ has rank at most 1. With $W'_Q = \mathbf{q}\mathbf{1}^\top$ and $W'_K = \mathbf{k}\mathbf{1}^\top$, before scaling,

$$\text{score}(t, t') = d_Q s_t r_{t'}, \quad s_t = \mathbf{x}_t \cdot \mathbf{q}, \quad r_{t'} = \mathbf{x}_{t'} \cdot \mathbf{k}$$

Each token is reduced to one number as a query (s_t) and one as a key ($r_{t'}$). Every query ranks the keys in the same order – or the reverse, when $s_t < 0$; only the sharpness of the softmax varies.

A diagonal W_V makes each output feature one input feature, rescaled: values no longer mix features.

This is a trade, and the authors report that their benchmarks tolerate it. It is selection in the sense of Part I: most of the matrix is discarded and what is kept is kept exactly – the opposite strategy from the fold, which keeps everything and blurs it.

Checked numerically under both readings of the copying: rank 1, every row of $QK^\top$ parallel.

Sorting Everything by What Happens to Each Cell

each cell of the $n \times n$ object is ...	technique	compresses
summed into its anti-diagonal group	the fold; HRR binding; HRR attention	representations
summed into a random bucket, with a random sign	tensor sketch; compact bilinear pooling; MCB	representations
summed into its diagonal group	circular correlation; HoIE	representations
forced to equal the rest of its wrapped diagonal	circulant and block-circulant layers	parameters
forced to equal $\pm$ the rest of its random group	HashedNets	parameters
averaged with the matching cells of similar rows	model folding	parameters
kept exactly, copied, or set to zero	weight matrix folding	parameters
regrouped into smaller blocks	folding attention	parameters
rewritten exactly	batch-norm folding	parameters

Aggregation keeps everything, blurred. **Selection** keeps a few things, exactly. **Sharing** ties cells together, so there are fewer numbers to learn.

Summary

- The fold is a linear map $\mathbb{R}^{n \times n} \rightarrow \mathbb{R}^n$ that sums wrapped anti-diagonals. Its adjoint builds an $n \times n$ matrix from n numbers.
- **Fourier and the adjoint.** In Fourier coordinates, convolution acts one coefficient at a time. The adjoint is not an inverse: $\frac{1}{n}\text{fold}^*$ is the fold's pseudo-inverse, and the adjoint of binding is Plate's involution – exact only for unitary roles, but stable when the trace is noisy.
- **Representations.** Tensor sketch is the fold with random buckets and signs: unbiased comparisons, any output size, and order kept. It underlies compact bilinear pooling and MCB for question answering.
- **Relations.** Correlation folds along the other diagonals and keeps the antisymmetric part – which is why HolE can tell dog hypernym canine from its reverse.
- **Parameters.** Circulant, block-circulant and hashed layers are built by the adjoint, and training them folds the gradient.
- **Attention.** HRR attention stores the fold of linear attention's $H \times H$ state.
- **Namesakes.** Batch-norm folding, folding attention, model folding and weight matrix folding are different operations. Ask what happens to each cell.

Bonus Exercises

- 1 Find a nonzero 4×4 matrix M with $\text{fold}(M) = \mathbf{0}$, so that the round trip $\frac{1}{4}\text{fold}^*(\text{fold}(M))$ returns the zero matrix. How many independent such matrices are there?
- 2 The involution of $\mathbf{u} = (1, 2, 0, 1)$ is $\mathbf{u}^* = (1, 1, 0, 2)$. Compute $\mathcal{F}\mathbf{u}^*$ by hand and check that each coefficient is the conjugate of $(\mathcal{F}\mathbf{u})_k$. Then compute $\mathbf{u}^* \circledast (\mathbf{u} \circledast \mathbf{v})$ in Fourier coordinates and compare it with $6\mathbf{v}$.
- 3 Take $s_1 = s_2 = s$. Show that the signed fold then satisfies $\text{fold}_\pm(M^\top) = \text{fold}_\pm(M)$. Why does tensor sketch give the two factors independent signs?
- 4 Call $\mathbf{r} \in \mathbb{R}^d$ *even* if $r_k = r_{-k \bmod d}$ for every k . Show that HolE then scores (s, r, o) and (o, r, s) identically. Which WordNet relations would you expect to have an even $\mathbf{r}$?
- 5 Show that if W_Q has rank 1, every query ranks the keys in the same order or in the reverse order.
- ★6. For a hashed layer with $h(i, j) = (i + j) \bmod n$ and $\xi(i, j) = s_1(i) s_2(j)$, write $\partial L / \partial \mathbf{w}$ using the signed fold.

Bonus Exercises: Answers

1. Any matrix whose four groups each sum to zero – for instance a 1 at $(0,1)$ and a -1 at $(1,0)$, both in group 1; or the antisymmetric part of $\mathbf{u}\mathbf{v}^\top$. The fold has 16 inputs and 4 independent outputs, so there are $16 - 4 = 12$.
2. $\mathcal{F}\mathbf{u}^* = (4, 1+i, -2, 1-i)$, the conjugates of $(4, 1-i, -2, 1+i)$. Times $\mathcal{F}\mathbf{w} = (16, 2, -4, 2)$ this gives $(64, 2+2i, 8, 2-2i)$, which transforms back to $(19, 13, 17, 15)$: $6\mathbf{v} = (12, 0, 6, 6)$ plus crosstalk.
3. Cell (i, j) gets sign $s(i)s(j)$ and cell (j, i) gets $s(j)s(i)$, the same, and both lie in the same group. With independent signs, $s_1(i)s_2(j)$ and $s_1(j)s_2(i)$ generally differ – which is what lets the sketch keep order.
4. $\mathbf{e}_o \star \mathbf{e}_s$ is $\mathbf{e}_s \star \mathbf{e}_o$ with $k \mapsto -k$, and an even $\mathbf{r}$ cannot tell the difference. `similar_to` is a natural candidate; `hypernym` and `part_of` are not.
5. $\text{score}(t, t') = d_Q s_t r_{t'}$: for each query, the scores over the keys are one fixed vector $(r_{t'})$ scaled by s_t . A positive s_t keeps its order; a negative one reverses it.
6. $\partial L / \partial w_k = \sum_{i+j \equiv k} s_1(i)s_2(j) \partial L / \partial V_{ij} = \text{fold}_\pm(\partial L / \partial V)_k$.

Reading, 1: Fourier, Sketches, and Relations

- Cooley & Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation, 1965 – the FFT.
- Plate, *Holographic reduced representations*, IEEE Transactions on Neural Networks, 1995 – the involution, the exact inverse, and unitary vectors.
- Charikar, Chen & Farach-Colton, *Finding frequent items in data streams*, ICALP 2002 – the count sketch.
- Pham & Pagh, *Fast and scalable polynomial kernels via explicit feature maps*, KDD 2013 – tensor sketch.
- Lin, RoyChowdhury & Maji, *Bilinear CNN models for fine-grained visual recognition*, ICCV 2015.
- Gao, Beijbom, Zhang & Darrell, *Compact bilinear pooling*, CVPR 2016.
- Fukui, Park, Yang, Rohrbach, Darrell & Rohrbach, *Multimodal compact bilinear pooling for visual question answering and visual grounding*, EMNLP 2016.
- Nickel, Tresp & Kriegel, *A three-way model for collective learning on multi-relational data*, ICML 2011 – RESCAL.
- Nickel, Rosasco & Poggio, *Holographic embeddings of knowledge graphs*, AAAI 2016 – HoE.

Reading, 2: Structured Weights, Attention, and the Namesakes

- Cheng, Yu, Feris, Kumar, Choudhary & Chang, *An exploration of parameter redundancy in deep networks with circulant projections*, ICCV 2015.
- Ding et al., *CirCNN: accelerating and compressing deep neural networks using block-circulant weight matrices*, MICRO 2017.
- Chen, Wilson, Tyree, Weinberger & Chen, *Compressing neural networks with the hashing trick*, ICML 2015 – HashedNets.
- Katharopoulos, Vyas, Pappas & Fleuret, *Transformers are RNNs: fast autoregressive transformers with linear attention*, ICML 2020.
- Alam, Raff, Biderman, Oates & Holt, *Recasting self-attention with holographic reduced representations*, ICML 2023 – Hrrformer.
- Jacob et al., *Quantization and training of neural networks for efficient integer-arithmetic-only inference*, CVPR 2018.
- Li et al., *Folding attention: memory and power optimization for on-device transformer-based streaming speech recognition*, arXiv 2023.
- Wang, Šikić, Thiele & Saukh, *Forget the data and fine-tuning! Just fold the network to compress*, ICLR 2025.
- Belhaouari & Kraidia, *Efficient self-attention with smart pruning for sustainable large language models*, *Scientific Reports* 15:10171, 2025.