Banya AI 3

Orthogonal Data-Operation Tokens and Convolutional Scale-Depth Structure: Label-Free Emergence of Operation-as-Weight

Hyukjin Han (bokkamsun@gmail.com) · ORCID 0009-0007-4132-1707

Banya Research Series, Part 3

Abstract

We view language as an orthogonal product of data (content: what a thing resembles) and operation (transformation: how it changes things). In this view, a single token exists as an orthogonal pair of two embeddings, a data plane and an operation plane. We call this orthogonal-pair token a bi-token. A bi-token is a unit in which one token simultaneously carries two orthogonal components, a data plane and an operation plane; its defining feature is that which plane is data and which is operation is never labeled in advance—only geometry (orthogonality) and positional asymmetry (an add slot versus a multiply slot) are enforced. The data plane is content added to the residual stream, and the operation plane is a multiplicative gate that opens and closes the channel gates of the next token. Without labeling which token is data and which is operation, enforcing only geometry (orthogonality) and positional asymmetry (add slot versus multiply slot) makes the operation plane emerge as a consistent operator that acts in a consistent direction on any data. We measure this and show the following. The directional consistency of operators is 10.1 times that of random; 72.5 percent of distinct operations are orthogonal to each other; and turning the operation plane on lowers the cross-entropy of next-word prediction by 2.79 (from 2.99 to 0.20, over 200 windows of the bundled self-made elementary dialogue corpus). This contribution is an observation within the training distribution: it stays at +0.3 to +0.6 on same-style holdout texts and does not appear on adult natural-language style that the curriculum has not yet reached (Section 6). The distribution of token operation ratios has an empty middle band, and trained tokens lie in the operation-dominant band. Channel mixing uses a circulant filterbank instead of a dense matrix, forming a filter ladder whose scale varies with depth, in which shallow layers become broad brushes and deep layers fine pens. This sequential composition carries the syntagmatic relation (order) of language, and order is handled by the mixing heads. All 16 mixing heads look at the same distance at the start of training and, by the end of training, differentiate to divide up different distances from 39 to 66 positions in the past. Finally, we measure the assembly discrimination of stem-ending combinations never seen in training using minimal pairs. On 198 unseen combinations, accuracy is 86.9% under the reversal distractor and 87.7% under the ending-internal swap distractor, far above the 50% chance level, and it holds even while a syllable-bigram baseline collapses to 47.7% on the hard distractor. The causal contribution of orthogonality was confirmed with a same-size control model retrained side by side with orthogonality removed, on the same seed and corpora. The

control model's discrimination on unseen combinations falls from 86.9 to 76.3 percent, below the bigram baseline (83.3 percent), while its prediction cross-entropy stays close to the original (0.28 versus 0.21) — removing orthogonality specifically harms compositional structure, not prediction in general (Section 6).

Measured

All quantitative figures in this paper are measured values obtained in the writing environment (frozen elementary-stage model; environment in Appendix B). Values that follow directly from the architecture, such as parameter counts, are stated as facts. The theoretical background of the orthogonal view is in the separate axiom document [1]; the claims of this paper come from measurement, not from deduction from those axioms.

Contents

Terminology Legend

Terms are defined in the tables below. One concept goes by one name only, and the text and figures of all seven parts follow these tables. The in-code column gives the identifier that implements the concept in the enclosed code, and the standard-term column gives the closest name in common use; cells are filled only where they differ.

Terminology legend 1. Engine and credit

Term	Meaning	In code	Standard term
exact adjoint	The backward computation derived by hand as closed forms for every operation; the proper name of this engine and its credit chain	`banya_core.py`	manual back-propagation
adjoint	The partner operation of the forward pass; it carries the gradient exactly backward	`bwd` kernels	adjoint
credit	The loss gradient at a hidden state; the responsibility a parameter bears for the outcome	delta (δ)	gradient
credit chain	The skeleton that walks the cache in reverse, applying each adjoint to carry credit back to the input	`backward`	back-propagation
adjoint agreement	The check that the adjoint formulas produce the same values as finite differences	per-part probes	gradient check
circulant-matrix channel mixing	Channel mixing with a circulant matrix, computed as componentwise products under rfft		circular convolution
relative-distance mixing	Mixing positions with shared coefficients that depend only on distance	`m_mat_w_lag`	Toeplitz mixing
temporal mixing	The umbrella name for any path by which other positions' information flows into the current one		token mixing
mixing heads	The number of parallel heads a mixing operation is split into		attention heads
filter ladder	The convolutional channel transform whose filter scale varies with depth, from a few wide filters to many narrow ones	`m_mat_w_filter`	convolutional filterbank
window	The token positions the model sees at once; as a measurement unit, a fixed-length text slice	`T`	context window
probe	The verification script that replays every measured figure together with its target value	`probe` folder	reproduction script
frozen model	A developmental-stage model whose training is stopped and fixed	`model` npz	frozen checkpoint
holdout	Evaluation text never used in training		held-out set

Terminology legend 2. Bi-token and the gate

Term	Meaning	In code	Standard term
bi-token	The structure that gives one token an orthogonal pair of embeddings, a data plane and an operation plane	`USE_BITOKEN`	
data plane	The embedding plane carrying a token's content; it enters at the add slot	`m_mat_w_data_axis`	token embedding
operation plane	The embedding plane carrying a token's operation instruction; it enters only at the multiply slot	`m_mat_w_operate_axis`	
add slot, multiply slot	The residual-addition position where the data plane lands and the gate-multiplication position where the operation plane acts		residual connection, gating
gate	The valve that sets each channel's pass-through between 0 and 1; the previous token's operation plane is injected into it	`m_mat_w_gate`	gating
one-step shift	The one-position displacement by which the previous token's operation plane is injected into the next position's gate		
operation-as-weight	The phenomenon in which operation-plane embeddings emerge as weights without any labels		
consistent operator	An operator that acts in a consistent direction on any data		
direction consistency	The mean cosine of the directions in which one operation bends different data; the quantitative sign of operator emergence	`p_direction_consistency`	
rotation-operator gate	The gate that reads the operation plane as Euler rotation angles; abbreviated in the text as the rotation gate	`paper7_rotation.py`	
quaternion gate	The gate that reads four operation-plane components as a unit quaternion and left-multiplies blocks of four channels	`install_quaternion`	
primitive operation	An operation the gate can express directly in a single-hop instruction		primitive
depth-excluded discrimination	The deliberately biased measurement that restricts depth to one block so the gate is the only remaining variable	Part 7 probe	
self-feedback	The scoring mode that feeds the model's own output back as input and scores the final state		autoregressive rollout

Terminology legend 3. Rumination and development

Term	Meaning	In code	Standard term
Rumination	Self-organizing learning that pulls embeddings toward their neighbors without computing any gradient	Part 2 probes	self-organization
same-kind group	The set of close-in-meaning concepts gathered from a seed by nearest-cosine traversal		cluster
centroid pull, decay	The two forces of Rumination: pulling toward the mean and restoring toward the original; equilibrium emerges from their balance	`ALPHA`, decay rate	
World-first	The curriculum order that develops sensory, spatial, and logical axes before language	`banya_world_data`	curriculum learning
developmental stage	A checkpoint stage defined by step count and named after human growth		
axis	A property scale formed as a direction inside the embedding: sensory, ordinal, categorical		representation axis

Terminology legend 4. Vocabulary and metacognition

Term	Meaning	In code	Standard term
syllable atom	The syllable-level token forming the bottom layer of the vocabulary	`AtomTokenizer`	character-level token
bundle	A concept token made by binding syllable atoms into the dictionary; used only as a vocabulary unit		subword
cache-dictionary tokenization	The two-layer vocabulary system that puts a bundle dictionary on top of the atom layer	Part 5 probes	contrast with subword tokenization
certain, vague, unknown	The three knowing states separated by the model's own output distribution	pmax, entropy	uncertainty estimation
routing	Assigning an answer's processing path according to the state		routing
reinterpretation	The procedure that re-runs a vague hidden state together with its context to lift it to certain	Part 6 probes	
similarity consolidation	The axis along which concepts are organized by data-plane similarity		
inference traversal	The axis along which inference is carried forward on the operation plane		
self-utterance	The target state that weaves similarity consolidation and inference traversal so the model continues speech by itself		
Banya framework	The fifteen-axiom system forming the series' background; not the ground of the claims in the text		

Symbol Legend

We define the symbols shared by the equations and the text of this paper. Local symbols used only within a single equation are given in the symbol line beneath each equation.

Symbol	Name	Definition and meaning	Type
d	data-plane embedding	The embedding carrying the token's content (what it resembles); added directly to the residual stream. Code name mat_w_data_axis	vector
o	operation-plane embedding	The embedding carrying the token's transformation (how it changes things); opens and closes the channel gate and starts at identity (initial value 0). Code name mat_w_operate_axis	vector
g	channel gate	The valve between 0 and 1 that sets how much of each component passes. Written g_t with a position subscript in Eq. 4-1	vector
x_t	hidden state at position t	Hidden vector at context position t; its dimension is the channel width H	vector
t	context position index	The subscript pointing to the context position of hidden states, gates, and operation planes; $t-1$ is the position one step earlier (the previous token)	integer
H	channel width	Dimension of the hidden vector. Default 1024 (Table A-1). Section 5.2 shows that a circulant kernel needs only H entries per block and layer	integer
b_g	gate bias	The gate's bias; starting at 2.0, the gate starts out nearly open (Table A-1)	vector
o_{t-1}	previous token's operation plane	The previous token's operation-plane embedding, shifted one step, is added inside the gate at the current position and regulates how much of other tokens' data passes; if 0 the gate is identity	vector
m_t	time-mixing input	Other tokens' data arriving at position t through time mixing; modulated by the element-wise product with the gate g_t	vector
n	number of pairs	The number of valid pairs in each bundle of the minimal-pair assembly task (33 seen, 198 unseen, etc.)	integer
p	binomial-test p-value	One-sided binomial test value against random 50%; the probability of scoring this well by coin flips	scalar

1. Introduction

Linguistics distinguishes two kinds of relations: paradigmatic (things that can substitute for one another—a bear and a rabbit are both animals) and syntagmatic (things linked by order—"the bear" is followed by "is an animal"). Standard embeddings collapse both into a single vector. This paper splits a token into two orthogonal planes. The data plane holds what the token resembles (paradigmatic, content), and the operation plane holds how it transforms (transformation, multiplication). Their sequential composition then produces the syntagmatic relation. This is a view of language as an orthogonal product of data and operation, and the formal background of this view is in the separate axiom document [1]. This paper verifies by measurement what the view actually produces in training.

The core question is a single one: without labels for which token is an operator, does enforc-

ing geometry alone make the operation plane split off as a consistent operator? The measurements say it does. The operation plane starts from identity (multiplying changes nothing), splits off without labels, emerges as a consistent operator that acts in a consistent direction on any data (10.1 times random), and, when turned on, helps prediction by 2.79.

1.1 Contributions

- Orthogonal data-operation tokens : we present a structure that splits a token into two orthogonal embeddings, a data plane (additive) and an operation plane (multiplicative gate) (Section 3).
- Label-free operator emergence : we measure, via directional consistency, orthogonality, and prediction contribution, that the operation plane emerges as a consistent operator without labels (Section 5).
- Convolutional scale structure : channel mixing is a circulant filterbank forming a filter ladder whose scale varies with depth; we measure its parameter savings and per-layer roles (Section 5).
- Order differentiation of mixing heads : we measure with distance kernels that causal mixing heads differentiate, without labels, to cover different past distances and thus divide up order (the syntagmatic relation) (Section 5).
- Honest measurement of compositional discrimination : we measure the assembly discrimination of stem-ending combinations unseen in training with minimal pairs and a bigram baseline, and delimit the scope of the claims (Section 5).

2. Related Work

Factorized embeddings (splitting an embedding into the product of two small tables) reduce parameters but do not assign distinct semantic roles to the two factors. FiLM (feature-wise linear modulation) [4] and gating modulate features with a separate conditioning signal, but that signal comes from outside rather than being an embedding owned by the same token. In this paper, the same token owns a data embedding and an operation embedding simultaneously, and the operation plane starts from identity and splits into an operator without labels. Quantifying this emergence by directional consistency, and forming a filter ladder by using a circulant filterbank instead of a dense matrix for channel mixing, are the points of difference. Training of the exact adjoint of the circulant operation is provided by the engine of Part 1 of this series [2].

The idea of giving one token both a content representation and a transformation representation is not new here. One model gave every word a vector (content) and a matrix (an operator that changes its neighbors) [5], and another learned a face composed by addition and a face composed by multiplication together [6]. The device in which the previous token, shifted by one position, opens and closes the channels of the next token by multiplication also already exists in a recurrent-style model [7]. This paper differs in four points. First, the origin of the

idea is different. The design comes not from these precedents but from the axiom literature [1], published before this series. Its first axiom writes every change as the orthogonal sum of a data bracket and an operator bracket, and splits them as the place where state is recorded (data) versus the place where operations execute (operator). The bi-token of this paper carries that axiom over to language as it stands: the place where state is recorded becomes the add slot (the data plane written into the residual), the place where operations execute becomes the multiply slot (the operation plane driving the gate), and the orthogonality of the two planes corresponds to the orthogonality of the two brackets. The idea that language too has orthogonal components, so a token should be received as an orthogonal pair, came from there. The axioms are background only, not evidence for this paper's claims; the evidence is measurement. Second, the precedents decide in advance, by shape (matrix versus vector), which side is the operator, whereas this paper gives two identically shaped embeddings only orthogonality and a position (an add slot versus a multiply slot) and lets training itself decide which plane hardens into the operator. Third, the operator here is a vector, not a matrix. Giving every word a matrix makes one operator as expensive as the square of the channel width, whereas here a single vector acts through a shared gate, so the cost of one operator stays at the channel width. The one-step shift of [7] mixes representations derived from the same embedding and carries no operator identity separate from content, whereas here a token owns an operation-embedding table separate from its content. Fourth, the precedents report task scores, whereas this paper measures the geometry of the operators themselves: direction consistency (10.1 times random), the fraction of mutually orthogonal operators (72.5 percent), an on/off contrast (cross-entropy 2.99 versus 0.20), and the distribution of token operation ratios with an empty middle band.

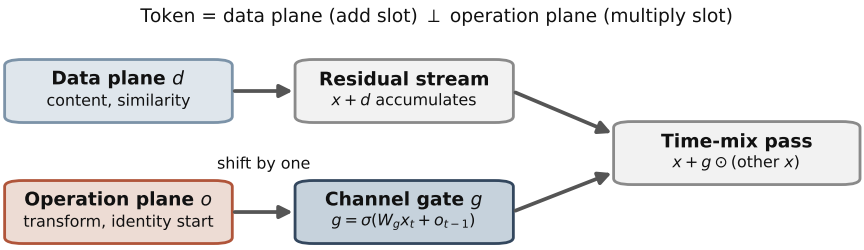
3. Notation and Structure

Table 3-1. Symbol legend

Symbol	Name	Definition and meaning	Type
d	data-plane embedding	The token's content (what it resembles). Added to the residual. Code name mat_w_data_axis	vector
o	operation-plane embedding	The token's transformation (how it changes things). Opens and closes the gate. Starts at identity (0). Code name mat_w_operate_axis	vector
g	channel gate	A 0-to-1 valve for how much of each component passes	vector
x_t	hidden state at position t	Hidden vector at context position t	vector
H	channel width	Dimension of the hidden vector. 1024	integer

A single token carries both a data plane d and an operation plane o. The data plane is content added directly to the residual stream. The operation plane of the previous token is injected into the channel gate of the next token, regulating how much of other tokens' data passes through. This one-step shift is where the syntagmatic relation (order) lives. The overall structure is shown in Figure 3-1.

Figure 3-1. Structure of the orthogonal data-operation token



The data plane is added (content); the previous token's operation plane multiplies into the next gate (transform). The two slots are orthogonal.

The token's data plane is content added to the residual stream (top), and the operation plane of the previous token is shifted one position and injected into the channel gate of the next token to regulate throughput (bottom). Only the asymmetry between the add slot and the multiply slot is enforced; which token is data and which is operation is never labeled.

3.1 Circulant filterbank channel mixing and the filter ladder

The channel-mixing part uses a circulant filterbank instead of a dense matrix. Shallow layers hold a few wide filters (broad brushes) and deep layers hold many narrow filters (fine pens), so the scale descends like a ladder. The exact adjoint of this circulant operation is provided by the engine of Part 1 [2], so training proceeds without automatic differentiation.

4. Method

The operation plane starts at identity (all zeros; multiplying changes nothing). As training proceeds, the credit of the operation plane is extracted at the gate, flows back to the token one position earlier, and is updated without labels through its exact adjoint. The data plane is pure content added to the residual and does not pass through the gate. This asymmetry is the only enforcement that separates the two planes.

Eq. 4-1. Operation-plane gate injection and modulation

$$g_t = \sigma\big(W_g\, x_t + b_g + o_{t-1}\big), \qquad x_t^{\text{out}} = x_t + g_t \odot m_t$$

Formula

Symbols: σ is the sigmoid (a function squashing values into 0 to 1), W_g is the gate weight, b_g is the gate bias (starting at 2.0, nearly open), o_{t-1} is the previous token's operation plane, m_t is the other tokens' data arriving through time mixing, and $\odot$ is the element-wise product.

How it works: the previous token's operation plane o_{t-1} is added inside the gate, regulating how much of other tokens' data the current token lets through. If the operation plane

is 0, the gate remains as it was (identity); if nonzero, it changes the throughput.
Actual behavior: the operation plane acts through the gate in the “multiply slot,” while the data plane rides on the residual in the “add slot.” Because the two actions occupy different slots, they are orthogonal. Even without labels, this difference of slots separates the two planes.

5. Results

5.1 The operation plane emerges as a consistent operator

Measured

Results measured on the frozen elementary-stage model.

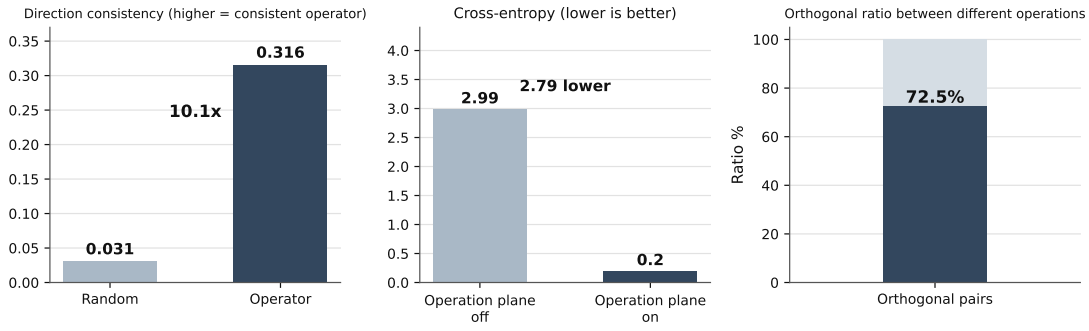
- Directional consistency : the average similarity of the directions in which the same operation bends different data is 0.316 , which is 10.1 times the random baseline of 0.031. That is, an operator acts in a consistent direction on any data (a consistent operator).
- Operation orthogonality : 72.5 percent of distinct operation pairs are nearly orthogonal (similarity below 0.15), and same-direction pairs (above 0.5) are 0 percent. Each operation is distinct.
- Prediction contribution : turning the operation plane on lowers next-word prediction cross-entropy from 2.99 to 0.20, a drop of 2.79 (200 windows of the bundled self-made elementary dialogue corpus). The operation plane substantially helps prediction. Its transfer outside the training distribution is covered in Section 6.
- Token identity distribution : in the distribution of operation ratios (the operation-plane norm divided by the sum of the data-plane and operation-plane norms), the middle (0.45 to 0.55) holds 0 percent and the extremes (below 0.15 or above 0.85) hold 34 percent. The composition of the extremes, however, is not symmetric. The data-side pole, 33.6 percent (915 tokens), consists entirely of tokens that never appeared in training (pad, control characters, unused jamo). Among the 1809 trained tokens, the data-side pole holds 0 tokens, the operation-side pole holds 8 (0.4 percent), and all the rest lie in the operation-dominant band (0.55 to 0.85). What this distribution shows is not a symmetric split into two identities but an empty middle, meaning no token is a half-and-half mixture, together with the operation dominance of trained tokens.
- Within-token orthogonality : for each of the 1809 trained tokens, the absolute cosine between its own data-plane and operation-plane embeddings averages 0.027 with a maximum of 0.121, so every token falls in the nearly orthogonal range (below 0.15), at the level of the random baseline of 0.031. No projection enforcing orthog-

onality exists in the code, so this is an observation that the two planes developed no correlation with each other even while trained by the same prediction loss.

- Operator identity : being an operator is not the property of a grammatical class. For 30 functional syllables (particles and endings) versus 30 noun-like syllables, the mean operation ratios are 0.630 versus 0.622, a weak lean with a pairwise win rate of 63.4 percent, and directional consistency shows no class difference: 0.324 for functional, 0.316 for noun-like, and 0.313 for verb-adjective stems (random 0.031). The operator role is distributed across the vocabulary, and which token acts as an operator is decided by the multiply slot (the wiring that feeds the previous token's operation plane into the next position's gate) and by combination. What this paper enforces is the operator's slot; what occupies that slot is left to training and context.

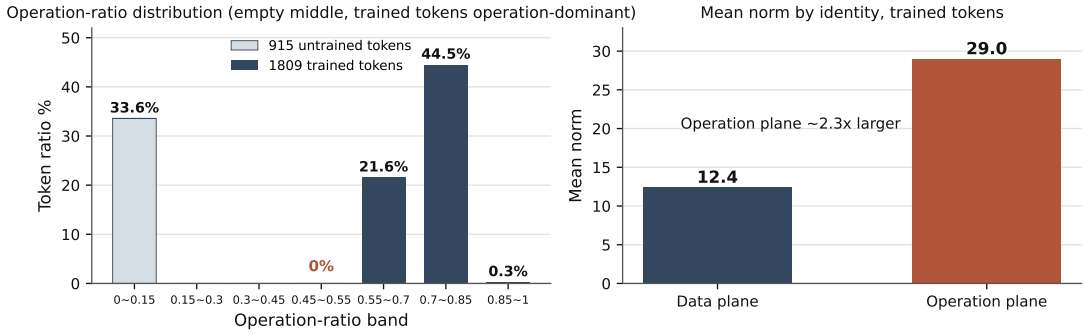
The orthogonality drawn by these measurements is two-fold. In representation, a token's data plane and operation plane are orthogonal to each other (within-token orthogonality); in processing, the filterbank reads only the data plane (Section 5.2) and the operation plane enters only through the gate (Section 4). The difference in organizing principles — the data plane gathers by resemblance while the operation plane separates by use — is treated in Part 2.

Figure 5-1. Operator directional consistency and prediction contribution



The directional consistency of operations is 10.1 times random, showing that the operation plane is a consistent operator acting in a consistent direction on any data (left). Turning the operation plane on lowers prediction cross-entropy from 2.99 to 0.20, a drop of 2.79 (right). The operation plane is not decoration; it genuinely contributes to prediction.

Figure 5-2. Distribution of token operation ratios: an empty middle and operation dominance of trained tokens



In the distribution of token operation ratios, the middle (half-and-half mixtures) holds 0 percent. The 33.6 percent at the data-side pole consists entirely of untrained tokens that never appeared in training, and all 1809 trained tokens lie in the operation-dominant band (0.55 and above), with 8 tokens at the operation-side pole. Among trained tokens, the mean operation-plane norm is 29.0, about 2.3 times the data-plane mean of 12.4, and no trained token has a zero operation-plane norm.

5.2 The convolutional filter ladder

Using a circulant filterbank for channel mixing greatly reduces parameters. A circulant kernel needs only H entries per block and layer, so at $H = 1024$ it is about 1024 times smaller than the roughly 1.05 million entries of a dense matrix (the same structural fact measured in Part 1 [2]).

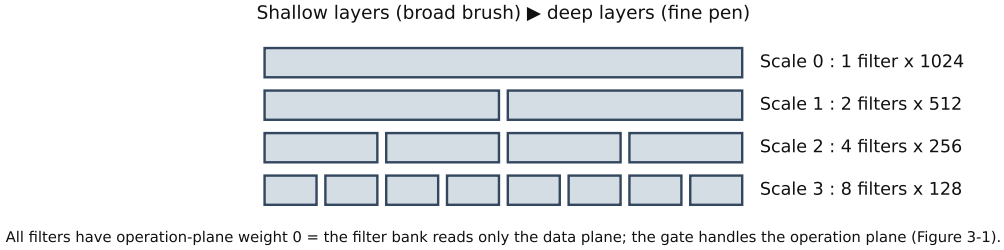
Measured

Measuring the filter ladder shows that, from shallow to deep layers, filters descend from wide and few (broad brushes) to narrow and many (fine pens).

Depth (block-layer)	Scale	Filter count	Filter length	Data-plane weight	Operation-plane weight
0-0	0	1	1024	3.49	0.00
0-1	1	2	512	4.59	0.00
1-1	2	4	256	17.57	0.00
2-1	3	8	128	8.59	0.00
3-0	3	8	128	11.71	0.00

The scale descends with depth like a ladder (1024x1, 512x2, 256x4, 128x8). And the operation-plane weight of every filter is 0. That is, the circulant filterbank reads only the data plane (content), and the operation plane is handled by the gate. The two pathways are structurally separated.

Figure 5-3. Filter ladder by depth



All filters have operation-plane weight 0 = the filter bank reads only the data plane; the gate handles the operation plane (Figure 3-1).

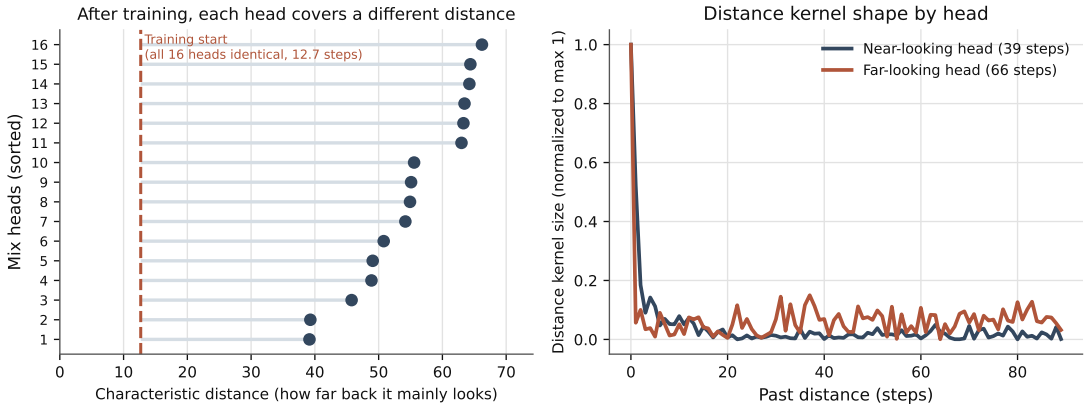
As depth increases, filters descend from wide and few to narrow and many (the filter ladder). Since the operation-plane weight of every filter is 0, the circulant filterbank mixes only data-plane content, and the operation plane is handled separately by the gate. Channel mixing (convolution) and transformation (operation plane) are structurally separated.

5.3 The mixing heads carry order

If the data plane covers the paradigmatic relation (similarity) and the operation plane covers transformation (inference), where does the syntagmatic relation—order—live? In this model, order is handled by the mixing heads. Each block has several mixing heads with a causal mask (a lower-triangular mask that sees only the past and hides the future), and each head learns a distance kernel (a weight per distance for how much to look at tokens that many positions back). When heads cover different distances, together they read the front-to-back order of a sentence at multiple widths.

We measured the distance kernels of the trained model's mixing heads. All 16 heads start training from the same distance kernel (characteristic distance 12.7 positions). By the end of training the heads' assigned distances split, ranging from a head that mainly looks 39 positions back to one that looks 66 positions back (inter-head standard deviation 8.6 positions within one block; 6 to 9 positions across all blocks). That is, the heads start identical and differentiate to divide up different intervals of order. Order is placed onto the mixing heads without labels.

Figure 5-4. Mixing heads divide up different distances of order



Left: the 16 mixing heads of one block split their assigned distances through training. All start at 12.7 positions (red vertical line, training start) and differentiate to 39 to 66 positions in the past. Right: the distance-kernel shapes of a near-looking head and a far-looking head. The near-looking head concentrates weight on short distances, while the far-looking head spreads weight out to long distances. Order (the syntagmatic relation) is divided among the mixing heads.

Reproduce

The figures in Sections 5.1 and 5.3 are reproduced with the following one-liner after completing the preparation in Appendix C.

```
cd probe && python3 paper3_bitoken_probe.py
```

In the output, directional consistency 10.1 times, nearly orthogonal pairs 72.5%, cross-entropy off 2.99 versus on 0.20, and mixing-head differentiation from a characteristic distance of 12.7 positions to 39~66 positions correspond to the figures in the text.

5.4 Assembly discrimination of unseen combinations

If the orthogonal pair structure carries composition, then even stem-ending combinations never seen in training should be separable into correct and incorrect assemblies. We measure this with a minimal-pair task (placing side by side two candidates that differ in only one element from the same material).

We build a grid of 240 combinations from 20 stems and 12 endings that attach without any morphological change regardless of final-consonant status. We exhaustively scan the token sequences of the entire corpus that this checkpoint's training accessed, classifying combinations whose surface form appeared at least once as seen and those with zero occurrences as unseen. The scanned list is the 23 self-made mixed corpora bundled in the reproduction package. External text corpora (fairy-tale materials) and dictionaries that training may have accessed are excluded from the package and the scan because they contain original copyrighted works, so the possibility remains that some unseen combinations were exposed through that path. This limitation is reflected in the interpretation below. For discrimination, we place the correct assembly and a distractor after each of five subject frames (naneun, geuneun, urineun, dongsaengi, agiga), compare the log-probability sums, and count it correct if the correct side is higher. Distractors come in two stages. Stage 1 reverses the order of stem and ending (meokdaga vs. dagameok); 9 pairs whose reversed sequences actually occur in the corpus and thus carry no discriminative power (e.g., jiga, 5,834 occurrences) are excluded. Stage 2 swaps syllables inside the ending (meokdaga vs. meokgada); the swapped neighboring sequences (gada, nida) are actually common in the corpus, making these items hard to solve by neighbor statistics alone. The valid pairs are 33 seen combinations and 198 unseen combinations.

There are two comparison baselines: random 50%, and a syllable bigram baseline (frequency of adjacent two-character sequences) built from the same 23 corpora. The bigram baseline

solves the same minimal pairs using only a frequency table with Laplace smoothing (a correction adding 1 to every count). This baseline shows how far surface transition statistics alone go on this task.

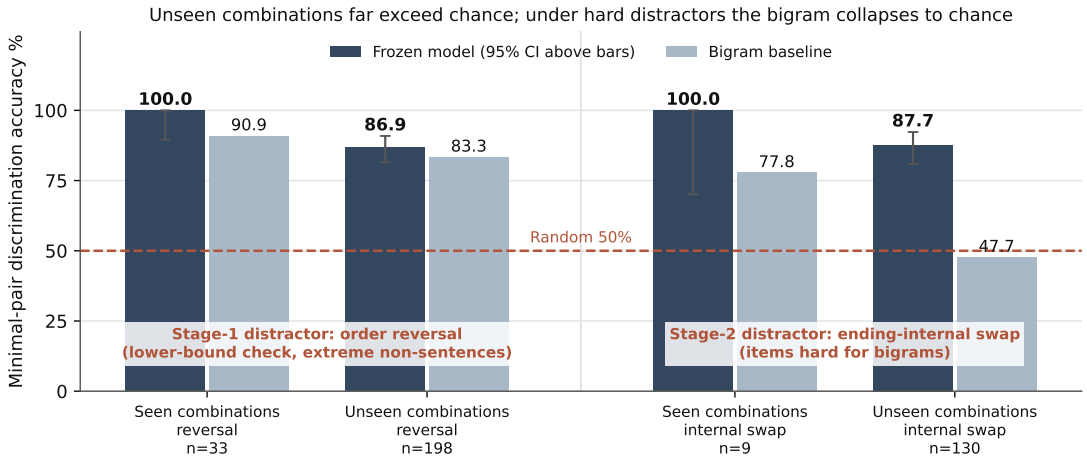
Measured

Measured results. p is the one-sided binomial test value against random 50% (the probability of scoring this well by coin flips).

Distractor	Bundle	n	Model accuracy	Bigram baseline	Model p
Reversal	seen	33	100.0%	90.9%	1.2e-10
Reversal	unseen	198	86.9%	83.3%	6.7e-28
Internal swap	seen	9	100.0%	77.8%	2.0e-03
Internal swap	unseen	130	87.7%	47.7%	1.0e-19

Unseen combinations far exceed chance under both distractors (86.9% and 87.7%, p at or below 10 to the minus 19). There exists assembly discrimination that whole-form memorization alone cannot explain. The easy stage-1 distractor is largely solved by the bigram baseline as well (83.3%), but the model leads. On the hard stage-2 distractor, the bigram collapses to 47.7%, chance level, and its log-probability margin flips to -0.50, while the model holds 87.7% with a margin of +8.10. On this distractor, the pairs where the bigram is wrong but only the model is right number 60 among unseen combinations, versus 8 for the reverse.

Figure 5-5. Compositional generalization, model versus bigram baseline



In all four bundles the model (dark bars) far exceeds random 50% (dashed line). On the easy reversal distractor the bigram baseline (light bars) is also high, but on the hard internal-swap distractor the baseline collapses to chance level while only the model holds. Error bars are Wilson 95% confidence intervals.

We state the interpretive limits of this measurement. Unseen means the entire surface form has zero occurrences relative to the 23 bundled corpora; the possibility of exposure via the external text path excluded from the scan remains, and most of the constituent bigrams were seen

many times in training. The seen judgment uses substring matching, so it is a superset that may include homographs (nouns such as gaji and bogo), and the sample of 33 is small. What this measurement shows extends to: the absence of a whole-form memorization advantage, assembly discrimination far above chance, and discrimination that holds on the hard distractor even where surface statistics collapse. Causal confirmation that the orthogonal structure is the cause was carried out through a side-by-side retraining comparison with a same-size control model with orthogonality removed; the results are in Section 6. On this discrimination the control model falls below the bigram baseline.

Reproduce

The assembly discrimination of this subsection is reproduced with the following one-liner after completing the preparation in Appendix C.

```
cd probe && python3 paper3_composition_probe.py
```

In the output, 33 seen combinations at 100.0%, 198 unseen combinations at 86.9%, the stage-2 distractor model 87.7% versus bigram 47.7%, and the diagnostic subset correspond to the figures in the text.

6. Limitations and Next Measurement Tasks

The reference class of the baseline is a single developer, a self-built exact-adjoint backpropagation engine, and one consumer graphics processing unit. The following are not defects but the next measurement tasks.

- Standard-transformer baseline (measured) : a standard transformer of 104M parameters (attention and dense transforms, AdamW) was trained on the same world curriculum for the same 170,000 steps with the same batch of 32, and measured side by side on identical windows (in-distribution elementary dialogue, 128-token windows, 200 per seed, four seed sets). Table 6-1 gives the results and the trade balance. The reading is as follows. The standard model's slight prediction edge is the price bought by all-pairs comparison quadratic in the window and dense transforms quadratic in the channels; this engine stays within that 10 percent without paying those costs and runs 70 percent faster per step, so at equal execution-time time it takes 70 percent more steps (a time-matched comparison remains a next task). At window 128 the quadratic cost of attention is still small, so we do not attribute the 0.020 solely to all-pairs comparison; optimizer-practice differences (AdamW versus this engine's update accounting) are also mixed in. The claim of this paper is not a prediction advantage but the structure observed on top of this same-league baseline. As a side observation, the standard model's free-generation sample did not form sentences despite its lower prediction CE; that the prediction scale and generation quality can diverge is left as a separate measurement task.

Table 6-1. Standard-transformer baseline and the trade balance

Scale	Bi-token 104M	Standard 104M	Trade
Prediction CE (same 800 windows)	0.207	0.186	standard lower by 0.020 (about 10% relative)
Step speed (batch 32, window 128)	80.5 ms	136.7 ms	this engine 70% faster
Position-mixing cost	relative-distance mixing (log-linear)	attention (quadratic in window)	advantage shifts to this engine as the window grows
Channel-transform cost	filter ladder, circulant (linear to log-linear)	dense transform (quadratic in channels)	advantage shifts to this engine as channels grow

- Orthogonality-removed control (measured) : a same-size model with orthogonality removed (no operation plane) was retrained side by side on the same seed, the same corpora, and the same 170,000 steps, and measured under the same protocol as Section 5.4. The control model falls to 90.9 percent on seen combinations (original 100.0) and 76.3 percent on unseen combinations (original 86.9), dropping below the bigram baseline of 83.3 percent, which the original exceeds. On the hard stage-2 distractor it reaches 80.0 percent (original 87.7). Its prediction cross-entropy, in contrast, stays close to the original (0.28 versus 0.21): retraining compensates almost all of general prediction but fails to compensate the compositional knowledge beyond surface statistics. The orthogonal structure therefore contributes specifically to compositional generalization. Limits: the retraining is a single run (one seed) under GPU nondeterminism, so confirming the variance with repeated retraining is a next task.

- Out-of-distribution transfer : the prediction contribution of +2.79 in Section 5.1 is an observation within the training distribution. Repeating the same measurement (200 random windows, four window samples) with the same frozen model while changing only the text, the contribution decreases with distribution distance. On the bundled holdout texts that were not used in training but share the same style (30 pieces each of the middle-school and high-school sets in eval_sets.json), it stays positive: +0.32 to +0.46 on the middle-school set and +0.40 to +0.60 on the high-school set. On adult natural language of a different style (75 random Korean Wikipedia articles, 225 thousand characters, atom-encoded after Unicode normalization and Hangul-and-ASCII-preserving preprocessing, zero unknown tokens), cross-entropy is 4.58 to 4.69 with the plane off versus 4.74 to 4.86 with it on, a slightly negative contribution of -0.13 to -0.25. The reading is as follows: what the operation-plane gate has learned is gate instructions tuned to the constructions of the training style, so it transfers to unseen text of the same style, whereas on a different style the instructions misfire and end up slightly worse than the off setting. Geometric measurements such as directional consistency and orthogonality are internal properties of the model, independent of the input text, and stand as they are. This checkpoint is the product of a developmental curriculum that has reached only the elementary stage, so adult style is a register it has not yet been taught; the decrease is not a defect but

the expected order. We confirmed this by actually continuing the curriculum. From the published checkpoint we trained a stage that adds middle-school spoken language [8] and a Korean dictionary [9] (15,000 steps), then a stage that adds a self-made adult-topic corpus (20,000 steps), and measured the same evaluation ladder again. Directly trained styles rose sharply on their holdouts (middle-school spoken holdout +0.46 to +1.69; adult-topic holdout +0.34 to +3.34). Elementary dialogue, which kept receiving rehearsal, held at +2.79 to +3.05. Adult natural language (Wikipedia) turned positive, -0.13 to +0.10, at the middle-school stage where the dictionary share was large, then returned to -0.42 at the next stage where the dictionary share shrank and the adult-topic corpus dominated: the training material closest to encyclopedic style is identified as definitional style (the dictionary). Styles left out of rehearsal declined stage by stage (self-made knowledge-QA holdout +0.37 to -0.99), an interference in which gate instructions are reallocated toward the newest material, meaning a style to be kept must be placed in the rehearsal share. Throughout these shifts the geometry kept its frame (directional consistency 10.1 to 9.2 times, orthogonal pairs 72.5 to 73.4 percent, head differentiation 39~66 to 40~66 positions): the basis of the decomposition is already formed, and continued training reallocates the instructions on top of it. The continued training is a single run (one seed), and the next tasks toward encyclopedic style are keeping the definitional-style share and adding world-knowledge (news) corpora.

- Convolution versus dense : a comparison placing the circulant filterbank side by side with same-size dense channel mixing is the next measurement task (the relevant probe reads only the default path, so its execution mode needs supplementing).
- Comparison with modulation methods : a side-by-side comparison with FiLM or gating is a next task.
- Expressive limits of the gate : the gate's range of 0 to 1 expresses only passing and blocking, so negation (a sign flip) cannot be expressed in one hop, and the operation-plane components act diagonally, each adjusting only its own channel, so there is no rotation. The extension that reads the operation plane as Euler rotation angles, and one rung up as unit quaternions, is the subject of a subsequent paper (Part 7).

7. Conclusion

This paper presented a structure that splits a token into two orthogonal embeddings, a data plane and an operation plane, and showed by measurement that enforcing geometry alone, without labels, makes the operation plane emerge as a consistent operator. The directional consistency of operations is 10.1 times random, 72.5 percent of distinct operations are orthogonal, turning the operation plane on lowers prediction cross-entropy by 2.79 within the training distribution (out-of-distribution transfer is covered in Section 6), and the distribution of token operation ratios has an empty middle band with trained tokens in the operation-dominant band. Channel mixing uses a circulant filterbank to form a filter ladder whose scale varies with depth while reading only the data plane, structurally separating convolution from transformation. Order is handled by the causal mixing heads: we measured that 16 heads start identical and

differentiate to different distances from 39 to 66 positions in the past. We also measured that assembly discrimination of stem-ending combinations unseen in training far exceeds chance and holds on the hard distractor even where the bigram baseline collapses; its interpretation follows the qualifications of Section 5.4. In the orthogonality-removed control retraining this discrimination falls below the bigram baseline, confirming the causal contribution of the orthogonal structure (Section 6). The orthogonality of data and operation, and their sequential composition, correspond respectively to the paradigmatic and syntagmatic relations of language.

8. Series Preview

Series preview

- Part 4 — World-first developmental curriculum learning : how the frozen model of this paper was trained; measures the process in which a training order that develops sensory, spatial, and logical axes before language forms axis structure in the embeddings.
- Part 5 — Cache-dictionary tokenization and dynamic embedding : treats the vocabulary layer on which the data plane and operation plane ride as a two-tier structure of syllable atoms and a bundle dictionary.
- Part 6 — Control signals emerging from self-distribution and geometry monitoring : treats, with measured statistics, the control signals the model extracts by reading its own output distribution and embedding geometry.
- Part 7 — The rotation-operator gate and the quaternion gate : starts from the two expressive limits of this paper's gate (Section 6). Reading the operation plane as Euler rotation angles expresses negation as 180 degrees and the composition of operations as angle addition, and the adjoint of a rotation is the conjugate rotation, closed form. Angle addition is commutative, so composition whose order changes the result demands the next rung: the quaternion gate, which reads four operation-plane components as a unit quaternion. On a depth-stripped one-block small setting, an ablation contrast that swaps only the gate treats both discriminations, a commutative task (mod 3) and a non-commutative one (the triangle symmetries).
- Final part (unfinished) — Combining similarity consolidation and inference traversal : self-utterance weaving data-plane similarity consolidation (Part 2 [3]) with operation-plane inference traversal. Not yet covered, as the final assembly is incomplete.

References

[1] Han, H. (2026). Banya Framework: An Axiom-Based Science Mining Engine for Deriving Fundamental Physical Constants. Zenodo. https://doi.org/10.5281/zenodo.20301304 . (Background natural philosophy of this series; prior axiom document)

[2] Han, H. (2026). A Language-Model Training Engine Using Hand-Derived Closed-Form Exact Adjoints Without Automatic Differentiation. Banya Research Series, Part 1. Zenodo. https://doi.org/10.5281/zenodo.21385447

[3] Han, H. (2026). Gradient-Free Self-Organization of Concept Embeddings by Frequency-Proportional Rumination. Banya Research Series, Part 2. Zenodo. https://doi.org/10.5281/zenodo.21385516

[4] Perez, E., Strub, F., de Vries, H., Dumoulin, V., Courville, A. (2018). FiLM: Visual Reasoning with a General Conditioning Layer. AAAI. arXiv:1709.07871.

[5] Socher, R., Huval, B., Manning, C. D., Ng, A. Y. (2012). Semantic Compositionality through Recursive Matrix-Vector Spaces. EMNLP-CoNLL.

[6] Mai, F., Galke, L., Scherp, A. (2019). CBOW Is Not All You Need: Combining CBOW with the Compositional Matrix Space Model. ICLR. arXiv:1902.06423.

[7] Peng, B. et al. (2023). RWKV: Reinventing RNNs for the Transformer Era. Findings of EMNLP. arXiv:2305.13048.

[8] National Institute of Korean Language (2021). NIKL Spoken Corpus (v.1.2). https://kli.korean.go.kr/corpus

[9] National Institute of Korean Language. Standard Korean Language Dictionary. KOGL Type 1. https://stdict.korean.go.kr

Appendix A. Default Configuration

Table A-1. Bi-token configuration

Item	Value	Item	Value
Channel width H	1024	Operation-plane initial value	0 (identity)
Gate bias initial value	2.0 (nearly open)	Operation-plane injection	one-step shift
Channel mixing	circulant filterbank	Scale lower bound	filter length 32

Appendix B. Measurement Environment

Table B-1. Computer specifications used for measurement

Category	Specification
Graphics processing unit (GPU)	NVIDIA GeForce RTX 3090 Ti (24 GB)
Operating system	Ubuntu 24.04.4 LTS
Software	Python 3.12.3, numpy 2.5.1, scipy 1.18.0, cupy 14.1.1, CUDA 13.3.1
Measurement target	Data plane, operation plane, and filters of the frozen elementary-stage model (about 170,000 steps)

The measured figures in Section 5 were obtained in the environment above.

Appendix C. Reproduction Guide

The measured figures in this paper are reproduced with the public reproduction package. Follow the five steps below in order.

1. Requirements. An NVIDIA graphics processing unit (GPU) with CUDA and Python 3.12 are required. Install the Python packages from the unzipped folder with the following. For cupy, use the distribution matching the installed CUDA version (e.g., cupy-cuda13x for CUDA 13).

```
pip install -r requirements.txt
```

2. Get the code. Download and unzip the archive (zip). To get it as a repository, use the github.com/ubmscoin/banya/banya_ai_paper_code folder.

```
wget https:
//ubmscoin.github.io/banya/banya_ai_paper_code/banya_ai_paper_code.zip
unzip banya_ai_paper_code.zip && cd banya_ai_paper_code
```

3. Get the models. The three frozen models (471MB total) are on Zenodo at doi.org/10.5281/

zenodo.21383724. The following one-liner performs both the download and the integrity verification (sha256 file-fingerprint check). To do it by hand, download the three files from the record above, place them in the model folder, and verify with `sha256sum -c checksums.sha256`.

```
cd model && bash download.sh && cd ..
```

4. Build the corpora. The corpora are produced by generators without distributing original texts. The following one-liner generates all 23 corpora into the banya_world_data folder.

```
cd corpus_build && bash build_all.sh && cd ..
```

5. Reproduce the measurements. The probes in the probe folder reproduce the figures of each section. The figures in this paper are covered by the following probes. Target values are printed alongside the output, so they can be checked directly against the text.

Probe	Figures reproduced	Output to check
paper3_bitoken_probe.py	5.1 consistent operator and token identity distribution, 5.3 head differentiation	consistency 10.1 times, orthogonal pairs 72.5%, cross-entropy 2.99 vs. 0.20, extremes 34% (trained tokens only 0.4%), within-token orthogonality mean 0.027, class consistency 0.324 vs. 0.316, heads from 12.7 positions to 39~66 positions
paper3_composition_probe.py	5.4 assembly discrimination	seen 33 at 100.0%, unseen 198 at 86.9%, stage-2 model 87.7% vs. bigram 47.7%
paper3_baseline_probe.py	Section 6, Table 6-1 standard baseline	cross entropy of bi-token vs. standard on the same 800 windows, milliseconds per step

The trainer of the standard baseline, banya_bp_pytorch.py, is bundled as well; it is not specific to Part 3 but the common contrast model of the series. Scoring the standard side of Table 6-1 requires the standard checkpoint model/banya_bp_pytorch.pt, produced by the bundled trainer or downloaded. The three tale corpora are not bundled for copyright reasons, so a bundle-only retraining differs slightly from the published run in the last-stage mixture.

The package's folder layout and per-file descriptions are on the index page.

Banya Research Series, Part 3 (orthogonal tokens, convolutional scale). All figures were measured in the environment of Appendix B. The background natural philosophy is cited as the prior axiom document [1] but is not the basis of this paper's claims. Part 1 (engine) and Part 2 (rumination) are cited. The developmental curriculum, cache-dictionary tokenization, and emergent control signals are treated in Parts 4 through 6.