

Supplementary Material

We provide the code [here](#). We structure the supplementary material as follows:

Appendix [A](#) presents a proposition that provides special cases of the generalized Householder product for specific choices of keys and betas and characterizes the spectrum of the product of two generalized Householders.

Appendix [B](#) characterizes the expressivity of DeltaProduct.

- [B.2](#) demonstrates how DeltaProduct can solve any group word problem in a single layer when n_h is sufficiently high, or alternatively using up to 4 layers when n_h is limited.
- [B.3](#) shows how DeltaProduct can recognize any regular language in a finite number of layers using the Krohn-Rhodes decomposition.
- [B.4](#) demonstrates how DeltaNet can solve any dihedral group in 2 layers using a specific construction.
- [B.5](#) discusses the tradeoff between expressivity and stability in linear RNNs.

Appendix [C](#) provides comprehensive details on our experiments and additional results.

Notation. Mathematical objects are typically styled as follows. Matrices: uppercase letters (e.g., $\mathbf{A}, \mathbf{H}, \mathbf{M}$). Vectors: lowercase letters (e.g., $\mathbf{k}, \mathbf{v}, \mathbf{x}$). Standard sets: \mathbb{R} for reals, \mathbb{C} for complexes, \mathbb{N} for naturals. Common symbols and operations are denoted as follows. \mathbf{I} : Identity matrix. \top : Transpose operator (e.g., \mathbf{k}^\top). $\|\mathbf{v}\|$ or $\|\mathbf{v}\|_2$: Euclidean norm of a vector \mathbf{v} . $\|\mathbf{A}\|$: the operator norm for a matrix \mathbf{A} . $|\cdot|$: Absolute value for real scalars, or modulus for complex numbers. \odot : Element-wise (Hadamard) product. $\sigma(\mathbf{A}), \rho(\mathbf{A})$: Spectrum (set of eigenvalues) and spectral radius of matrix \mathbf{A} . $\text{tr}(\mathbf{A}), \det(\mathbf{A})$: Trace and determinant of matrix \mathbf{A} . δ_{ij} : Kronecker delta (1 if $i = j$, 0 otherwise). $e_i \in \{0, 1\}^n$ is the i -th element of the canonical basis of \mathbb{R}^n .

A Spectral Properties and Simplifications of Householder Product Matrices

The following proposition characterizes conditions under which the product structure simplifies and the spectrum is real, contrasting with the general case which allows for complex spectra. It provides illustrative special cases for the more general Proposition 1 in Grazi et al. [\[16\]](#).

Proposition 1. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a matrix defined as the product of $n_h \geq 1$ generalized Householder transformations: $\mathbf{A} = \prod_{j=1}^{n_h} \mathbf{H}_j$ where each $\mathbf{H}_j = \mathbf{I} - \beta_j \mathbf{k}_j \mathbf{k}_j^\top$, with $\mathbf{k}_j \in \mathbb{R}^n$ being a unit vector ($\|\mathbf{k}_j\|_2 = 1$) and $\beta_j \in [0, 2]$. Let $\sigma(\mathbf{A}) \subset \mathbb{C}$ denote the spectrum (set of eigenvalues) of \mathbf{A} . Then, all eigenvalues $\lambda \in \sigma(\mathbf{A})$ satisfy $|\lambda| \leq 1$ and the following hold.

1. **(Identical Direction Vector)** Let $\mathbf{k} \in \mathbb{R}^n$ be nonzero and $\mathbf{k}_j = \mathbf{k} / \|\mathbf{k}\|$ for all $j = 1..m$. Then $\prod_{j=1}^m (\mathbf{I} - \beta_j \mathbf{k} \mathbf{k}^\top) = \mathbf{I} - \beta^* \mathbf{k} \mathbf{k}^\top$ for some real scalar β^* depending on $\{\beta_j\}_{j=1}^m$. The product collapses to a single effective transformation of the same form. Consequently, if \mathbf{A} is formed using only a single direction vector \mathbf{k}_1 , it is symmetric and its spectrum is real.
2. **(Orthogonal Vectors)** If the direction vectors $\{\mathbf{k}_j\}_{j=1}^{n_h}$ form an orthonormal set (i.e., $\mathbf{k}_j^\top \mathbf{k}_l = \delta_{jl}$; this requires $n_h \leq n$), then the factors \mathbf{H}_j commute, and the product simplifies to $\mathbf{A} = \mathbf{I} - \sum_{j=1}^{n_h} \beta_j \mathbf{k}_j \mathbf{k}_j^\top$. This matrix \mathbf{A} is symmetric, and its spectrum is purely real: $\sigma(\mathbf{A}) = \{1 - \beta_1, \dots, 1 - \beta_{n_h}\} \cup \{1 \text{ (multiplicity } n - n_h)\}$. When $\beta_j = 2$ for all $j \in \{1, \dots, n_h\}$ then \mathbf{A} is known as a block reflector [\[66\]](#).
3. **(Complex Spectrum via Non-orthogonal Directions)** For $n_h = 2$, \mathbf{A} has complex eigenvalues if two consecutive direction vectors, e.g. $\mathbf{k}_1, \mathbf{k}_2$ satisfy $0 < |\mathbf{k}_1^\top \mathbf{k}_2| < 1$ and their coefficients β_1, β_2 exceed a threshold $\beta^*(\theta) < 2$ dependent on the angle θ between them. Conversely, if $0 \leq \beta_1 \leq 1$ or $0 \leq \beta_2 \leq 1$, these eigenvalues from the 2D span are guaranteed to be real.

Proof. Identical Direction Vector If $m = 1$, then the statement is trivially satisfied with $\beta^* = \beta_1$. Suppose the statement is true for $m \geq 1$, i.e., $\prod_{j=1}^m (\mathbf{I} - \beta_j \mathbf{k} \mathbf{k}^\top) = \mathbf{I} - \beta^{(m)} \mathbf{k} \mathbf{k}^\top$. Multiplying by $(\mathbf{I} - \beta_{m+1} \mathbf{k} \mathbf{k}^\top)$ produces $(\mathbf{I} - \beta^{(m)} \mathbf{k} \mathbf{k}^\top)(\mathbf{I} - \beta_{m+1} \mathbf{k} \mathbf{k}^\top) = \mathbf{I} - [\beta^{(m)} + \beta_{m+1} - \beta^{(m)} \beta_{m+1}] \mathbf{k} \mathbf{k}^\top$. Hence, by induction, the product of any number of such factors remains of the form $\mathbf{I} - \beta^* \mathbf{k} \mathbf{k}^\top$. Since the resulting matrix $\mathbf{A} = \mathbf{I} - \beta^* \mathbf{k} \mathbf{k}^\top$ (where $\mathbf{k} = \mathbf{k}_1$) is symmetric, its eigenvalues are real.

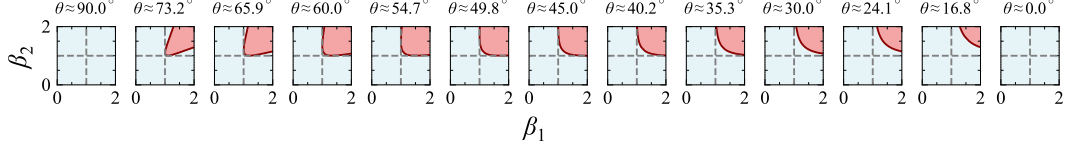


Figure 11: Visualization of complex eigenvalues of $(\mathbf{I} - \beta_1 \mathbf{k}_1 \mathbf{k}_1^\top)(\mathbf{I} - \beta_2 \mathbf{k}_2 \mathbf{k}_2^\top)$ with $\cos \theta = \mathbf{k}_1^\top \mathbf{k}_2$. Complex region in red, real in blue.

Orthogonal Vectors Assume $\{\mathbf{k}_j\}_{j=1}^{n_h}$ is an orthonormal set ($\mathbf{k}_j^\top \mathbf{k}_l = \delta_{jl}$, $n_h \leq n$). Let $\mathbf{P}_j = \mathbf{k}_j \mathbf{k}_j^\top$. Then $\mathbf{P}_j \mathbf{P}_l = \delta_{jl} \mathbf{P}_j$. The factors $\mathbf{H}_j = \mathbf{I} - \beta_j \mathbf{P}_j$ commute because $\mathbf{P}_j \mathbf{P}_l = \mathbf{0}$ for $j \neq l$. The product simplifies via induction to $\mathbf{A} = \mathbf{I} - \sum_{j=1}^{n_h} \beta_j \mathbf{P}_j$. This matrix is symmetric. Its eigenvalues are $(1 - \beta_l)$ for $l = 1, \dots, n_h$ (eigenvector \mathbf{k}_l) and 1 with multiplicity $n - n_h$ (subspace orthogonal to all \mathbf{k}_j). The spectrum is real.

Complex Spectrum via Non-orthogonal Directions and Real Subcase Let $\mathbf{k}_1, \mathbf{k}_2$ span a 2D subspace $S \subset \mathbb{R}^n$ with $\cos \theta = \mathbf{k}_1^\top \mathbf{k}_2$ such that $0 < |\cos \theta| < 1$. The product $\mathbf{A} = \mathbf{H}_2 \mathbf{H}_1$ acts as the identity on S^\perp (preserving $n - 2$ eigenvalues at 1) and non-trivially on S . The restriction \mathbf{A}_S of \mathbf{A} to S has trace $\text{tr}(\mathbf{A}_S) = 2 - \beta_1 - \beta_2 + \beta_1 \beta_2 \cos^2 \theta$ and determinant $\det(\mathbf{A}_S) = (1 - \beta_1)(1 - \beta_2)$. The discriminant of its characteristic equation is $D = [\text{tr}(\mathbf{A}_S)]^2 - 4 \det(\mathbf{A}_S)$. Complex eigenvalues arise if $D < 0$. This occurs when $(2 - \beta_1 - \beta_2 + \beta_1 \beta_2 \cos^2 \theta)^2 < 4(1 - \beta_1)(1 - \beta_2)$, a condition which can be met if β_1, β_2 are sufficiently large (e.g., $> \beta^*(\theta)$ for some $\beta^*(\theta) < 2$).

Conversely, we show that if $0 \leq \beta_1, \beta_2 \leq 1$, then $D \geq 0$. Expanding the expression of the discriminant, we get $D = (2 - \beta_1 - \beta_2)^2 + 2(2 - \beta_1 - \beta_2)\beta_1 \beta_2 \cos^2 \theta + \beta_1^2 \beta_2^2 \cos^4 \theta$. When $0 \leq \beta_1, \beta_2 \leq 1$, each term in this sum is non-negative, hence $D \geq 0$. This ensures that the eigenvalues from the 2D span S are real when $0 \leq \beta_1, \beta_2 \leq 1$. Furthermore, real eigenvalues from S are also guaranteed if one coefficient is in the range $[0, 1]$ while the other is in $(1, 2]$. Specifically, if one $\beta_i \leq 1$ (i.e., $\beta_i \in [0, 1]$) and the other $\beta_j > 1$ (i.e., $\beta_j \in (1, 2]$ for $i \neq j$), then their product $(1 - \beta_i)(1 - \beta_j) \leq 0$, which means $\det(\mathbf{A}_S) = (1 - \beta_1)(1 - \beta_2) \leq 0$. If $\det(\mathbf{A}_S) = 0$ (which occurs if the coefficient $\beta_i = 1$), then the discriminant $D = [\text{tr}(\mathbf{A}_S)]^2 \geq 0$. If $\det(\mathbf{A}_S) < 0$ (which occurs if $\beta_i \in [0, 1]$ and $\beta_j \in (1, 2]$), then the term $-4 \det(\mathbf{A}_S)$ is strictly positive. Consequently, $D = [\text{tr}(\mathbf{A}_S)]^2 - 4 \det(\mathbf{A}_S)$ must be positive, as it is the sum of a non-negative term $[\text{tr}(\mathbf{A}_S)]^2$ and a positive term. Thus, in both these subcases, $D \geq 0$ and the eigenvalues are real. This analysis confirms that complex eigenvalues can arise if and only if both $\beta_1 > 1$ and $\beta_2 > 1$, which enables rotations within the 2D subspace. Conversely, when $\beta_1 \leq 1$ or $\beta_2 \leq 1$, real eigenvalues restrict the transformations in that subspace to scaling or reflection. This clear distinction in behavior, dictated by the β values and θ , is illustrated in Figure 11. \square

B Expressivity of DeltaProduct

In this section, we characterize the expressivity of (Gated) DeltaProduct in solving group word problems and recognizing regular languages, in support of Section 4.1. The results hold in finite precision (since our constructions require a finite number of values), and take inspiration from Peng et al. [13, Appendix D] and [16]. We begin by stating and discussing our key assumptions in Appendix B.1. We then present our main results for group word problems in Appendix B.2, followed by our findings for regular languages in Appendix B.3. In Appendix B.4, we examine a result specific to dihedral groups. Finally, we explore the fundamental tradeoff between expressivity and stability of the recurrence in Appendix B.5.

B.1 Assumptions

We consider a (Gated) DeltaProduct model where each layer is structured as

$$\mathbf{H}_i = \mathbf{A}(\mathbf{x}_i) \mathbf{H}_{i-1} + \mathbf{B}(\mathbf{x}_i), \quad \hat{\mathbf{y}}_i = \text{dec}(\mathbf{H}_i, \mathbf{x}_i) \quad \text{where } i \in 1, \dots, t$$

$$\mathbf{A}(\mathbf{x}_i) = \prod_{j=1}^{n_h} g_i \left(\mathbf{I} - \beta_{i,j} \mathbf{k}_{i,j} \mathbf{k}_{i,j}^\top \right), \quad \mathbf{B}(\mathbf{x}_i) = \sum_{j=1}^{n_h} \left(\prod_{k=j+1}^{n_h} (\mathbf{I} - \beta_{i,k} \mathbf{k}_{i,k} \mathbf{k}_{i,k}^\top) \right) \beta_{i,j} \mathbf{k}_{i,j} \mathbf{v}_{i,j}^\top,$$

where g_i is only present in the gated variant and there is only one head per layer. If H heads are considered, head i will run the recurrence $\mathbf{H}_i^j = \mathbf{A}^j(\mathbf{x}_i) \mathbf{H}_{i-1} + \mathbf{B}^j(\mathbf{x}_i)$, (with different learnable

parameters from all other heads) and all the states will be passed to the decoder to get the output as $\hat{\mathbf{y}}_i = \text{dec}((\mathbf{H}_i^1, \dots, \mathbf{H}_i^H), \mathbf{x}_i)$.

We assume that for every $i, j, g_i \in [0, 1], \beta_{i,j} \in [0, 2], \mathbf{k}_{i,j} \in \mathbb{R}^d, \|\mathbf{k}_{i,j}\| = 1$ and $\mathbf{v}_{i,j} \in \mathbb{R}^d$ are arbitrary functions of \mathbf{x}_i , so that the model can set $\mathbf{A}(\mathbf{x}_i)$ as any possible product of n_h generalized Householder matrices. The function dec can be arbitrary and \mathbf{H}_0 can be set appropriately depending on the task and is of rank at most n_h . For simplicity, we also assume that the output of all layers is passed as input to all following layers: this can be achieved by using the residual connections (which would be inside dec) and by placing the output of different layers onto separate subspaces.

Beginning of sequence token. Alternatively, the assumption on \mathbf{H}_0 can be replaced by using a beginning of sequence token $x_1 = \$$ and setting $\mathbf{H}_0 = 0$, which is done in practice so that $\mathbf{H}_1 = B(\$)$ is a learnable matrix of rank at most n_h which acts as the \mathbf{H}_0 in our constructions.

Decoder implementation. In our implementation, dec is the same as in Gated DeltaNet:

$$\begin{aligned} \text{dec}((\mathbf{H}_t^1, \dots, \mathbf{H}_t^H), \mathbf{x}_t) &= \text{MLP}(\text{RMSnorm}(\mathbf{x}_t + \mathbf{o}_t)), \\ \mathbf{o}_t &= \sum_{j=1}^H \mathbf{W}_o^j \text{RMSnorm}((\mathbf{H}_t^j)^\top \mathbf{q}_t^j), \quad \mathbf{q}_t^j = \psi(\mathbf{W}_q^j \mathbf{x}_t) / \|\psi(\mathbf{W}_q^j \mathbf{x}_t)\| \end{aligned}$$

where $\mathbf{W}_o^j, \mathbf{W}_q^j$ are two learned matrices, $\psi = \text{SiLU}$, $\text{RMSnorm}(\mathbf{x}) = \mathbf{a} \odot \mathbf{x} / \sqrt{\epsilon + d^{-1} \sum_{i=1}^d x_i^2}$ corresponds (when $\epsilon = 0$) to a projection onto an axis aligned ellipse where $\mathbf{a} \in \mathbb{R}^d$ is learnable and determines the lengths of the axis and $\epsilon > 0$ is a small value set to avoid numerical errors. Meanwhile, MLP is a two layer MLP . This structure can be limiting. For instance, when $\mathbf{H}_t^j \in \mathbb{R}^d$, then $b_t = (\mathbf{H}_t^j)^\top \mathbf{q}_t \in \mathbb{R}$ and if $b_t \gg \epsilon$, then $|\text{RMSNorm}(b_t)| \approx 1$, which means that $\text{RMSNorm}(b_t) \approx \pm 1$ (binary) and would restrict the flow of information. Indeed, for all our constructions to work, we would need $|\epsilon| \approx |b_t|$ (which can happen in practice), so that the RMSnorm can retain the magnitude information and we could set $\mathbf{q}_t = \mathbf{b} / \|\mathbf{b}\|$ with $\mathbf{b} = (2b, 2b^2, \dots, 2b^d)$ with $b = a/c$ where a and c are respectively the maximum and minimum absolute value of the values that each element of \mathbf{H}_t^j can take (always a finite set in our constructions) excluding 0. When these conditions are met, we are guaranteed that $\mathbf{H}_t^j \mapsto \text{RMSnorm}((\mathbf{H}_t^j)^\top \mathbf{q}_t)$ is bijective so that the MLP can model a possibly complex function of \mathbf{H}_t . When $\mathbf{H}_t^j = \mathbf{e}_i \in \{0, 1\}^d$ (i -th element of the canonical basis), an alternative to the above construction is to replicate the recurrence onto d heads, where the j -th head has $\mathbf{W}_o^j = \mathbf{q}_t^j = \mathbf{e}_j$, so that, assuming $\mathbf{a} = (\sqrt{d}, \dots, \sqrt{d})^\top$ and $\epsilon = 0$, we get $\mathbf{o}_t = \mathbf{H}_t^j = \mathbf{e}_i$, which is the strategy used in Peng et al. [13] Appendix D]. However, for some problems using the one-hot encoding states $\mathbf{e}_1, \dots, \mathbf{e}_n$ is not very efficient. For instance, to solve the S_n word problem one would need $n!$ -dimensional one-hot vectors as states, while in our Theorem 1 we use n -dimensional vectors. Moreover, learning multiple identical heads is redundant and indeed we observe that in our synthetic experiments, the model is learning to use only one head to solve the task (see Section 5.2).

B.2 Group Word Problems

Any finite group is isomorphic to a subgroup of S_n , and the next theorem establishes that DeltaProduct can solve the word problem for the symmetric group S_n , which implies that it can also solve any group word problem.

Theorem 3 (Restatement of Theorem 1). *For any $n \in \mathbb{N}$ there exists a DeltaProduct model with one of the following configurations that can solve the word problem of the symmetric group S_n : (i) one layer with $n_h = n-1$ [16 Theorem 3] (ii) 3 layers with $n_h > 1$ (iii) 4 layers with $n_h = 1$. The construction for (ii) and (iii) requires that the MLP at the second last layer computes a lookup-table of size $2m \times (n!)^{2m}$, function of the last $2m$ input tokens and the position modulo $2m$ with $m = \lceil (n-1)/n_h \rceil$.*

Proof. One way to solve the group word problem for the symmetric group S_n is to map each element of the group $g \in S_n$ to the corresponding permutation matrix $\mathbf{P}_g \in \{0, 1\}^n$ and then for each input sequence x_1, \dots, x_t with $x_i \in S_n$ compute each element of the output sequence y_1, \dots, y_t as

$$y_i = x_i \cdot x_{i-1} \cdots x_1 = \phi(\mathbf{P}_{x_i} \cdots \mathbf{P}_{x_1} \mathbf{u}_0), \quad \mathbf{u}_0 = (1, \dots, n)^\top,$$

where ϕ is the surjective map from vectors in $\{1, \dots, n\}^n$ to the $n!$ elements of S_n , which we consider integers for simplicity, i.e. $x_i, y_i \in \{1, \dots, n!\}$.

(i). Since a permutation of n elements is a series of at most $n-1$ swaps of 2 elements, if $n_h = n-1$, then we can solve the problem with a 1-layer DeltaProduct by setting $\mathbf{H}_0 = \mathbf{u}_0$, $\text{dec}(\mathbf{H}_i, x_i) = \phi(\mathbf{H}_i)$, $\mathbf{B}(x_i) = 0$ ($\mathbf{v}_{i,j} = 0$), $\mathbf{A}(x_i) = \mathbf{P}_{x_i}$. The latter is achieved by setting for the j -th element in the product $\prod_{j=1}^{n_h} (I - \beta_{i,j} \mathbf{k}_{i,j} \mathbf{k}_{i,j}^\top)$, either $\beta_{i,j} = 2$ and $\mathbf{k}_{i,j} = (\mathbf{e}_k - \mathbf{e}_p)/\sqrt{2}$ with \mathbf{e}_i being the i -th element of the canonical basis of \mathbb{R}^n (swap element at index k with the one at index p), or $\beta_{i,j} = 0$ (identity).

(ii) and (iii). If $n_h < n-1$, then the state transition matrix is not sufficiently expressive to represent all permutations of n elements. However, we can use additional layers to overcome this issue as follows. Factorize the position $i \in \{1, \dots, t\}$ into $i = lm + \tilde{i}$ for integers $l \geq 0$ and $\tilde{i} \in \{1, \dots, m\}$.

First, consider the case when $l \geq 1$. The product $\tilde{\mathbf{P}}_l = \mathbf{P}_{x_{(l-1)m+m}} \cdots \mathbf{P}_{x_{(l-1)m+1}}$ is a permutation matrix of n elements and we can therefore factor it into $\tilde{\mathbf{P}}_l = \mathbf{G}_{l,m} \cdots \mathbf{G}_{l,1}$ where each of $\mathbf{G}_{l,1}, \dots, \mathbf{G}_{l,m}$ is a product of n_h generalized householder matrices. We fix one factorization for each possible permutation matrix. In the last layer and with enough information in \mathbf{x}_i about previous tokens, we can thus set $\mathbf{H}_0 = \mathbf{u}_0$ and

$$\mathbf{H}_i = \mathbf{G}_{l,\tilde{i}} \mathbf{H}_{i-1}, \quad \text{dec}(\mathbf{H}_i, x_i) = \phi(\mathbf{P}_{lm+\tilde{i}} \cdots \mathbf{P}_{lm+1} \mathbf{G}_{l,m} \cdots \mathbf{G}_{l,\tilde{i}+1} \mathbf{H}_i).$$

The case when $l = 0$ is handled by setting $\tilde{\mathbf{P}}_0 = \mathbf{G}_{0,m} \cdots \mathbf{G}_{0,1}$ with $\mathbf{G}_{0,i} = I$. Note that both $\mathbf{G}_{l,\tilde{i}}$ and $\text{dec}(\mathbf{H}_t, x_t)$ are functions of $\tilde{i} = i \bmod m$ and the last $m + \tilde{i}$ (in general the last $2m$) tokens. Hence, the layers before the last one are dedicated to output at each time-step i a lookup table for the possible values of $(i \bmod 2m, x_i, \dots, x_{i-2m+1})$. The first layers (2 if $n_h = 1$, 1 if $n_h > 1$) can provide $i \bmod 2m$ by using Lemma 1 with $d = 2m$. Finally, the second to last layer can output any function of the last $2m$ tokens and the position modulo $2m$ through Lemma 2 with $d = 2m$ and $\mathbf{a}_t = x_t$, by using $i \bmod 2m$ from the first layer(s). \square

Lemma 1. *The following DeltaProduct configurations can count modulo $d \in \mathbb{N}$. (i) 2 layers each with one head and $n_h = 1$ [16 Theorem 6]. (ii) 1 layer with one head and $n_h \geq 2$.*

Proof. For (i), we can use the same construction as in [16 Theorem 6], where the first layer does counting modulo 2 and the second layer computes addition modulo d . In this case, since we just want to count modulo d we can ignore input tokens and add 1 modulo d at each time-step. For (ii), note that if $n_h > 2$, we can set, for any time-step t , $\mathbf{B}(x_t) = 0$ and the state transition matrix $\mathbf{A}(x_t)$ equal to a 2D rotation with an angle of $2\pi/d$ by appropriately setting two keys, say $\mathbf{k}_{t,1}, \mathbf{k}_{t,2}$, setting $\beta_{t,1}, \beta_{t,2} = 2$ (while for the other Householders we set $\beta_{i,j} = 0$). Then, we can count modulo d by setting \mathbf{H}_0 in the span of $\mathbf{k}_{t,1}, \mathbf{k}_{t,2}$ and dec appropriately to map the d values that \mathbf{H}_t can take to the correspondent element in $\{1, \dots, d\}$. \square

Lemma 2. *A DeltaProduct layer with $n_h = 1$, receiving in its input at time-step t the tuple $(t \bmod d, a_t)$ ($t \geq 1$) where $a_t \in D \subset \mathbb{R}$ with D being a discrete set of values, can implement any function of $(t \bmod d, a_{t-d+1}, \dots, a_t)$, where for simplicity we set $a_i = a \notin D$ for $i \in \{2-d, \dots, 0\}$.*

Proof. Let $\tilde{t} = t \bmod d + 1$ and $\sigma : D \rightarrow \{0, \dots, |D|\}$ bijective and $\sigma(a) = 0$. Set $\mathbf{H}_0 = 0 \in \mathbb{R}^d$ and the recurrence update as

$$\mathbf{H}_t = (I - \mathbf{e}_{\tilde{t}} \mathbf{e}_{\tilde{t}}^\top) \mathbf{H}_{t-1} + \mathbf{e}_{\tilde{t}} a_t,$$

where \mathbf{e}_i is the i -th element of the canonical basis of \mathbb{R}^d . This can be implemented by setting $\beta_{t,1} = 1$, $\mathbf{k}_{t,1} = \mathbf{e}_{\tilde{t}}$, $\mathbf{v}_{t,1} = a_t$ and $\beta_{t,j}, \mathbf{k}_{t,j}, \mathbf{v}_{t,j} = 0$ for $j > 1$. With this choice, \mathbf{H}_t contains a_{t-d+1}, \dots, a_t . The result follows since dec can be an arbitrary function of both \mathbf{H}_t and \mathbf{x}_t and the latter contains $t \bmod d$. \square

Finally, the next results concern finite subgroups of the orthogonal and special orthogonal groups.

Theorem 4. *Let G be a group isomorphic either to a subgroup of $O(n)$, or to a subgroup of $SO(n+1)$ if n is even, then if $n_h = n$, there exist DeltaProduct model that solves the group word problem for G .*

1152 *Proof.* From the isomorphism we can map each element $g \in G$ to an orthogonal matrix G_g . For
 1153 this problem that each element of the input sequence belongs to G : $x_i \in G$ for every i .

1154 If $G_g \in O(n)$, then, since $n_h = n$ and every orthogonal $n \times n$ matrix can be written as the product
 1155 of at most n Householder matrices, we can set $H_0 = I \in \mathbb{R}^{n \times n}$ and $A(x_t) = G_{x_t}$, $B(x_t) = 0$
 1156 and $\text{dec}(H_t, x_t) = \phi(H_t)$ with $\phi : O(n) \rightarrow G$ bijective (which exists due to the isomorphism).
 1157 The Householder product structure enables $A(x_t)$ to represent general orthogonal matrices G_{x_t} ,
 1158 including rotations.

1159 If instead $G_g \in SO(n+1)$, since n is even in this case then we can still write G_g as a product of
 1160 an even number (at most n since $n+1$ is odd) of Householder matrices of dimension $n+1 \times n+1$
 1161 because the determinant of G_g is $+1$. Thus, we can set $A(x_t) = G_{x_t} \in \mathbb{R}^{n+1 \times n+1}$, $B(x_t) = 0$.
 1162 Now if we let $\bar{G} = G_{x_t} G_{x_{t-1}} \cdots G_{x_1}$ and set $H_0 = \text{diag}(1, \dots, 1, 0) \in \mathbb{R}^{(n+1) \times (n+1)}$ (we are
 1163 only allowed a rank n matrix), then $H_t = \bar{G} H_0$ will have all the first n columns equal to \bar{G} and the
 1164 last set to zero. However, the last column can be found as a function of the others since it must be
 1165 the unique unit vector orthogonal to all other columns of \bar{G} and for which $\det(\bar{G}) = +1$. Therefore,
 1166 there exists a bijective function from states to elements of the group, which can be implemented in
 1167 dec. \square

1168 B.3 Regular Languages

1169 **Definition 1** (Finite state automaton (FSA)). A finite state automaton (FSA) is a tuple
 1170 $(\Sigma, Q, q_0, \delta, F)$, where Σ is a finite set called alphabet, Q is the finite set of states, $q_0 \in Q$
 1171 is the initial state, for every $w \in \Sigma$ $\delta_w : Q \rightarrow Q$ is a state transition function and $F \subset Q$ is the set of
 1172 accepting states.

1173 **Definition 2** (Permutation-reset automaton). An FSA is permutation-reset if for every $w \in \Sigma$, δ_w is
 1174 either bijective or constant.

1175 **Definition 3** (Regular language). A regular language is a set of sequences L such that there exist an
 1176 FSA that accepts it, i.e. such that $L \subset \Sigma^*$, where Σ^* is the set of sequences with elements in Σ , and
 1177 that for every word $w = w_1 w_2 \dots w_t \in \Sigma^*$

$$\delta_w(q_0) := \delta_{w_t} \circ \delta_{w_{t-1}} \circ \cdots \circ \delta_{w_1}(q_0) \in F \iff w \in L. \quad (4)$$

1178 Notably, the computation of any FSA can be also done using only matrix and vector multiplications.
 1179 Indeed, if we let $Q = \{1, \dots, n\}$ (for simplicity), then we can map each state q to the one hot vector
 1180 e_q (element of the canonical basis of \mathbb{R}^n) and each transition δ_w to the matrix $M_w \in \{0, 1\}^{n \times n}$ with
 1181 element at row q and column q' being 1 if and only if $\delta(q') = q$. This way, by setting $r \in \{0, 1\}^n$
 1182 such that $r_q = 1$ if $q \in F$ and $r_q = 0$ otherwise, we have that for every word $w = w_1 w_2 \dots w_n \in \Sigma^*$

$$r^\top M_{w_t} M_{w_{t-1}} \cdots M_{w_1} e_{q_0} = 1 \iff w \in L.$$

1183 We observe that if δ_w is bijective and changes k states, then the corresponding M_w is a permutation
 1184 matrix that can be written as a product of $k-1$ Householder matrices, each corresponding to a swap
 1185 of two elements. Moreover, if δ_w is constant, i.e., if $\delta_w(q) = \bar{q}$ for every $q \in Q$, then $M_w e_q = e_{\bar{q}}$.
 1186 As we will see, constant transitions can be modeled by setting the gate to zero.

1187 We are now ready to state our main result.

1188 **Theorem 5** (Restatement of Theorem 2). For any regular language L and any $n_h \in \mathbb{N}$, there exists
 1189 a Gated DeltaProduct model with a finite number of layers that recognizes the language, i.e., for
 1190 every word $w \in \Sigma^*$ outputs 1 if $w \in L$ and 0 otherwise.

1191 *Proof.* Using the landmark theorem by Krohn and Rhodes [49] we can decompose the FSA corre-
 1192 sponding to the regular language L into a cascade of permutation-reset FSA. We can use a group of
 1193 at most 4 consecutive layers to represent each automaton in the cascade via Lemma 3. Then, we
 1194 can combine the different FSA in the cascade in a feedforward manner using the same construction
 1195 as the one in the proof of [16, Theorem 3], where the input of each FSA is the output concatenated
 1196 with the input of the one that comes before it in the cascade. \square

1197 **Lemma 3.** For any permutation-reset FSA with $|Q| = n$ and $|\Sigma| = s$, where each bijective state-
 1198 transition function δ_w changes at most k states, there exist a Gated DeltaProduct model with the
 1199 following configuration that can implement it, i.e., for any word $w = w_1, \dots, w_t \in \Sigma$ in input, it

1200 can output the corresponding sequence of states q_1, \dots, q_t of the FSA. (i) one layer with $n_h = k - 1$.
 1201 (ii) 3 layers with $n_h > 1$. (iii) 4 layers with $n_h = 1$. The construction for (ii) and (iii) requires that
 1202 the MLP at the second last layer computes a lookup-table of size $2m \times s^{2m}$, function of the last $2m$
 1203 input tokens and the position modulo $2m$ with $m = \lceil (k - 1)/n_h \rceil$.

1204 *Proof.* We use the matrix vector multiply construction to implement the FSA. For every time-step i
 1205 we set $x_i = w_i$ as the input to the model.

1206 (i). Set $H_0 = e_{q_0}$. When δ_{w_i} is bijective, by assumption it changes at most k states. Thus, by
 1207 setting $n_h = k - 1$ we can represent the corresponding M_{w_i} matrix using $A(w_i)$ with gate $g_i = 1$
 1208 (product of $k - 1$ generalized Householder matrices), and thus we set $B(w_i) = 0$. If instead δ_{w_i} is
 1209 constant, i.e., $\delta_i(q) = \bar{q}$ and $M_w e_q = e_{\bar{q}}$ for every $q \in Q$, then we can set the gate $g_i = 0$ so that
 1210 $A(w_i) = 0$ and $B(w_i) = k_i = e_{\bar{q}}$. Finally we set $\text{dec}(H_i, x_i) = H_i^\top (1, \dots, n)^\top$ to retrieve the
 1211 correct state at step i (for simplicity $Q = \{1, \dots, n\}$).

1212 (ii) and (iii). If $n_h < k - 1$, then the state transition matrix is not sufficiently expressive to repre-
 1213 sent all permutations of k elements. However, we can use additional layers to overcome this issue
 1214 similarly to Theorem 1.

1215 Factorize the position $i \in \{1, \dots, t\}$ into $i = lm + \tilde{i}$ for integers $l \geq 0$ and $\tilde{i} \in \{1, \dots, m\}$.
 1216 First, consider the case when $l \geq 1$. The product $\tilde{M}_l = M_{w_{(l-1)m+1}} \dots M_{w_{(l-1)m+m}}$ is either
 1217 a permutation matrix of k elements or, if for some i δ_{w_i} is constant, then there exist \bar{q} such that
 1218 $\tilde{M}_l e_q = e_{\bar{q}}$ for every $q \in Q$. Therefore, we factor \tilde{M}_l into $\tilde{M}_l = G_{l,m} \dots G_{l,1}$ where each of
 1219 $G_{l,1}, \dots, G_{l,m}$ is either a product of n_h generalized householder matrices or $G_{l,i} e_q = e_{\bar{q}}$ for every
 1220 $q \in Q$, which can be modeled setting the gate to zero as in point (i). We fix one factorization for
 1221 each possible permutation matrix. In the last layer and with enough information in x_i about past
 1222 tokens, we can thus set $H_0 = e_0$ and

$$H_i = G_{l,\tilde{i}} H_{i-1}, \quad \text{dec}(H_i, x_i) = (M_{lm+\tilde{i}} \dots M_{lm+1} G_{l,m} \dots G_{l,\tilde{i}+1} H_i)^\top (1, \dots, n)^\top$$

1223 The case when $l = 0$ is handled by setting $\tilde{M}_0 = G_{0,m} \dots G_{0,1}$ with $G_{0,i} = I$. Note that both
 1224 $M_{l,\tilde{i}}$ and $\text{dec}(H_t, x_t)$ are functions of $\tilde{i} = i \bmod m$ and the last $m + \tilde{i}$ (in general the last $2m$)
 1225 tokens. Hence, the layers before the last are dedicated to output at each time-step i a lookup table
 1226 for the possible values of $(i \bmod 2m, w_i, \dots, w_{i-2m+1})$. The first layers (2 if $n_h = 1$, 1 if $n_h > 1$)
 1227 can provide $i \bmod 2m$ by using Lemma 1 with $d = 2m$. Finally, the second last layer can output
 1228 any function of the last $2m$ tokens and the position modulo $2m$ through Lemma 2 with $d = 2m$ and
 1229 $a_t = w_t$, by using $i \bmod 2m$ from the first layer(s). \square

1230 B.4 Dihedral Groups

1231 In Grazi et al. [16] Theorem 6] it is shown that with 2 layers and the extended eigenvalue range,
 1232 DeltaNet can compute addition modulo m , which corresponds to solving the group word problem
 1233 for the cyclic group \mathbb{Z}_m , for any $m \in \mathbb{N}$.

1234 We extend Grazi et al. [16] Theorem 6] to prove that, under identical assumptions, DeltaNet
 1235 (DeltaProduct with $n_h = 1$) can solve the group word problem for the dihedral group D_m , for
 1236 any $m \in \mathbb{N}$. The dihedral group D_m represents the symmetries (both rotations and reflections) of a
 1237 regular m -sided polygon. As a notable example, D_3 is isomorphic to the symmetric group S_3 .

1238 The linear RNN construction used in this result can be implemented using a 2-layer DeltaNet Model
 1239 (DeltaProduct with $n_h = 1$) with two heads in the first layer. In the first layer, the linear RNN
 1240 will compute parity for rotations and reflections separately, i.e. it will record if the number of past
 1241 rotations (reflections) is even or odd. The recurrent state of the second layer will have $2m$ possible
 1242 values (same as the order of D_m) and each will be decoded differently based on the parity of reflec-
 1243 tions. The parity of rotations, combined with the group element, determines which reflection matrix
 1244 to use as the state transition matrix of the second layer.

1245 **Theorem 6** (Dihedral group word problems with reflections). *For any $m \in \mathbb{N}$, consider the group*
 1246 *word problem of the dihedral group D_m . There exist DeltaProduct models with the following con-*
 1247 *figurations that can solve it. (i) Two layers with $n_h = 1$ and at least two heads in the first layer and*
 1248 *one in the second layer. (ii) One layer with $n_h \geq 2$.*

1249 *Proof.* The elements of the dihedral group D_m can be divided into m rotations $\mathcal{R} = \{r_0, \dots, r_{m-1}\}$
 1250 and m reflections $\mathcal{S} = \{s_0, \dots, s_{m-1}\}$. The identity is r_0 . To be able to solve the corresponding
 1251 word problem, we would like to map sequences of group elements x_1, \dots, x_t with $x_i \in \mathcal{R} \cup \mathcal{S}$ into
 1252 sequences y_1, \dots, y_t with $y_i = x_1 \cdot x_2 \cdots x_i$ and \cdot is the group operation, that for dihedral groups is
 1253 defined as

$$r_i \cdot r_j = r_{i+j \bmod m}, \quad r_i \cdot s_j = s_{i+j \bmod m}, \quad s_i \cdot r_j = s_{i-j \bmod m}, \quad s_i \cdot s_j = r_{i-j \bmod m}. \quad (5)$$

1254 Note that a product of two rotations is commutative, while the product of two reflections or a reflection
 1255 with a rotation is not. Indeed for $m \geq 3$ D_m , is not an abelian group.

1256 **(ii)** The constructions of the two layers of DeltaProduct with $n_h = 1$ builds upon the one for the
 1257 cyclic group Z_m outlined in [16, Theorem 6]. The first layer is constructed to output parity separately
 1258 for rotations and reflections. In particular, using the following diagonal recurrence which indicates
 1259 in the first (second) coordinate whether the number of rotations (reflections) is even (0) or odd (1).

$$\begin{aligned} h_0^{(1)} &= 0, \quad h_t^{(1)} = a(x_t) \odot h_{t-1}^{(1)} + b(x_t), \quad y_t^{(1)} = \text{dec}^{(1)}(h_t, x_t) = (x_t, h_{t,1}, h_{t,2}). \\ a(x_i)_1 &= \begin{cases} -1 & \text{if } x_i \in \mathcal{R} \\ 1 & \text{if } x_i \in \mathcal{S} \end{cases} \quad a(x_i)_2 = \begin{cases} -1 & \text{if } x_i \in \mathcal{S} \\ 1 & \text{if } x_i \in \mathcal{R} \end{cases} \\ b(x_i)_1 &= \begin{cases} 1 & \text{if } x_i \in \mathcal{R} \\ 0 & \text{if } x_i \in \mathcal{S} \end{cases} \quad b(x_i)_2 = \begin{cases} 1 & \text{if } x_i \in \mathcal{S} \\ 0 & \text{if } x_i \in \mathcal{R} \end{cases} \end{aligned}$$

1260 This recurrence can be implemented also by DeltaProduct with $n_h = 1$ using 2 heads each with
 1261 scalar hidden states: one for the rotations and the other for the reflections. For the second layer, we
 1262 have instead the following constructions, which selects the appropriate reflection based on the parity
 1263 of the rotations and uses the parity of the reflections for dec.

$$\begin{aligned} h_0^{(2)} &= (1, 0)^\top, \quad h_t^{(2)} = A^{(2)}(y_t^{(1)})h_{t-1}^{(2)}, \quad y_t^{(2)} = \text{dec}^{(2)}(h_t^{(2)}, y_t^{(1)}) \\ A^{(2)}(y) &= H(\theta(y_1, y_2)) = \begin{bmatrix} \cos \theta(y_1, y_2) & \sin \theta(y_1, y_2) \\ \sin \theta(y_1, y_2) & -\cos \theta(y_1, y_2) \end{bmatrix} \\ \text{dec}^{(2)}(h, y) &= \begin{cases} r_{i^*} & \text{if } y_3 = 0 \\ s_{m-i^*} & \text{if } y_3 = 1 \end{cases}, \quad i^* = \arg \max_{i \in \{0, \dots, m-1\}} \max(\mathbf{c}_i^\top h, \mathbf{d}_i^\top h) \end{aligned}$$

1264 where $y = (y_1, y_2, y_3)^\top \in \mathcal{R} \cup \mathcal{S} \times \{0, 1\} \times \{0, 1\}$, $H(\alpha)$ is the 2×2 reflection matrix that reflects
 1265 all vectors by a line having an angle of $\alpha/2$ with the line passing from the origin and the vector
 1266 $(1, 0)^\top$ and $\theta : \mathcal{R} \cup \mathcal{S} \times \{0, 1\} \rightarrow \mathbb{R}$ determines the angle of the reflection and is defined for all
 1267 $i \in \{0, \dots, m-1\}$ as

$$\theta(r_i, 1) = \frac{(1-2i)\pi}{m}, \quad \theta(r_i, 0) = \frac{(1+2i)\pi}{m}, \quad \theta(s_j, 1) = \frac{-2j\pi}{m}, \quad \theta(s_i, 0) = \frac{(2+2j)\pi}{m}.$$

1268 Moreover, $\mathcal{C} = \{c_0, \dots, c_{m-1}\}$ and $\mathcal{D} = \{d_0, \dots, d_{m-1}\}$ are two sets of states and are defined as

$$\begin{aligned} d_0 &= h_0^{(2)} = (1, 0)^\top, \quad c_0 = H(\pi/m)d_0, \\ d_i &= R(2i\pi/m)d_0, \quad c_i = R(-2i\pi/m)c_0 \quad \text{for all } i \in \{0, \dots, m-1\}, \end{aligned}$$

1269 where $R(\beta)$ is a rotation matrix with angle $\beta \in \mathbb{R}$.

1270 Let $\alpha, \gamma \in \mathbb{R}$, the following are standard identities of products of 2D rotations and reflections.

$$\begin{aligned} R(\alpha)R(\gamma) &= R(\alpha + \gamma), & H(\alpha)H(\gamma) &= R(\alpha - \gamma), \\ R(\alpha)H(\gamma) &= H(\alpha + \gamma) & H(\gamma)R(\alpha) &= H(\gamma - \alpha). \end{aligned}$$

1271 From our choice of $d_0 = (1, 0)^\top$ and c_0 , for any $\alpha \in \mathbb{R}$ we have

$$\begin{aligned} R(\alpha)d_0 &= H(\alpha)d_0, \quad \text{and} \\ R(\alpha)c_0 &= R(\alpha)H(\pi/m)d_0 = R(\alpha)R(\pi/m)d_0 = R(\alpha + \pi/m \pm \pi/m)d_0 \\ &= H(\alpha + 2\pi/m)H(\pi/m)d_0 = H(\alpha + 2\pi/m)c_0. \end{aligned}$$

Moreover, from our choice of θ , \mathbf{d}_i and \mathbf{c}_i , using the identities above and the the fact that \mathbf{R} is a periodic function with period 2π we have that

$$\begin{aligned}\mathbf{d}_i &= \mathbf{R}(2i\pi/m)\mathbf{d}_0 = \mathbf{R}(2i\pi/m)\mathbf{H}(\pi/m)\mathbf{c}_0 = \mathbf{H}(\theta(r_i, 0))\mathbf{c}_0 \\ \mathbf{c}_i &= \mathbf{R}(-2i\pi/m)\mathbf{c}_0 = \mathbf{R}(-2i\pi/m)\mathbf{H}(\pi/m)\mathbf{d}_0 = \mathbf{H}(\theta(r_i, 1))\mathbf{d}_0 \\ \mathbf{d}_{m-i} &= \mathbf{R}(-2i\pi/m)\mathbf{d}_0 = \mathbf{H}(-2i\pi/m)\mathbf{d}_0 = \mathbf{H}(\theta(s_i, 1))\mathbf{d}_0 \\ \mathbf{c}_{m-i} &= \mathbf{R}(+2i\pi/m)\mathbf{c}_0 = \mathbf{H}((2+2i)\pi/m)\mathbf{c}_0 = \mathbf{H}(\theta(s_i, 0))\mathbf{c}_0\end{aligned}$$

for every $i \in \{0, \dots, m-1\}$. Therefore, we can write

$$\begin{aligned}\mathbf{H}(\theta(r_j, 1))\mathbf{d}_i &= \mathbf{R}(\theta(r_j, 1) - \theta(r_i, 0))\mathbf{c}_0 = \mathbf{R}(-2(i+j)\pi/m)\mathbf{c}_0 = \mathbf{c}_{i+j \bmod m}, \\ \mathbf{H}(\theta(r_j, 0))\mathbf{c}_i &= \mathbf{R}(\theta(r_j, 0) - \theta(r_i, 1))\mathbf{d}_0 = \mathbf{R}(2(i+j)\pi/m)\mathbf{d}_0 = \mathbf{d}_{i+j \bmod m}, \\ \mathbf{H}(\theta(s_j, 1))\mathbf{d}_i &= \mathbf{R}(\theta(s_j, 1) - \theta(s_{m-i}, 1))\mathbf{d}_0 = \mathbf{R}(-2(i+j)\pi/m)\mathbf{d}_0 = \mathbf{d}_{-i-j \bmod m}, \\ \mathbf{H}(\theta(s_j, 0))\mathbf{c}_i &= \mathbf{R}(\theta(s_j, 0) - \theta(s_{m-i}, 0))\mathbf{c}_0 = \mathbf{R}((2+2(i+j))\pi/m)\mathbf{c}_0 = \mathbf{c}_{-i-j \bmod m},\end{aligned}\tag{6}$$

for every $i, j \in \{0, \dots, m-1\}$. We proceed to verify that the output of the second layer is computed correctly: satisfying the product rule for the dihedral group in [5], i.e., we want to verify that

$$y_t^{(2)} = \begin{cases} r_{i+j \bmod m} & \text{if } y_{t-1}^{(2)} = r_i, x_t = r_j \\ s_{i+j \bmod m} & \text{if } y_{t-1}^{(2)} = r_i, x_t = s_j \\ s_{i-j \bmod m} & \text{if } y_{t-1}^{(2)} = s_i, x_t = r_j \\ r_{i-j \bmod m} & \text{if } y_{t-1}^{(2)} = s_i, x_t = s_j \end{cases}\tag{7}$$

Where we set $y_0^{(2)} = r_0$. First note that when $y_t^{(2)} \in \mathcal{S}$, then $y_{t,3}^{(1)} = 1$ and when $y_t^{(2)} \in \mathcal{R}$, then $y_{t,3}^{(1)} = 0$. We consider two cases.

Case 1. If $y_{t-1}^{(2)} = r_i$ and hence $y_{t-1,3}^{(1)} = 0$, then using [6] we obtain

$$\mathbf{h}_t^{(2)} = \mathbf{A}^{(2)}(\mathbf{y}^{(1)})\mathbf{h}_{t-1}^{(2)} = \begin{cases} \mathbf{H}(\theta(r_j, 1))\mathbf{d}_i = \mathbf{c}_{i+j \bmod m} & \text{if } x_t = r_j, y_{t,2}^{(1)} = 1 \\ \mathbf{H}(\theta(r_j, 0))\mathbf{c}_i = \mathbf{d}_{i+j \bmod m} & \text{if } x_t = r_j, y_{t,2}^{(1)} = 0 \\ \mathbf{H}(\theta(s_j, 1))\mathbf{d}_i = \mathbf{d}_{-i-j \bmod m} & \text{if } x_t = s_j, y_{t,2}^{(1)} = 1 \\ \mathbf{H}(\theta(s_j, 0))\mathbf{c}_i = \mathbf{c}_{-i-j \bmod m} & \text{if } x_t = s_j, y_{t,2}^{(1)} = 0 \end{cases}$$

This, together with the definition of $\text{dec}^{(2)}$ implies that

$$y_t^{(2)} = \text{dec}^{(2)}(\mathbf{h}_t^{(2)}, \mathbf{y}_t^{(1)}) = \begin{cases} r_{i+j \bmod m} & \text{if } x_t = r_j, y_{t,3}^{(1)} = 0 \\ s_{i+j \bmod m} & \text{if } x_t = s_j, y_{t,3}^{(1)} = 1 \end{cases}\tag{8}$$

Case 2. If instead $y_{t-1}^{(2)} = s_i$ and hence $y_{t-1,3}^{(1)} = 1$, then using [6] we obtain

$$\mathbf{h}_t^{(2)} = \mathbf{A}^{(2)}(\mathbf{y}^{(1)})\mathbf{h}_{t-1}^{(2)} = \begin{cases} \mathbf{H}(\theta(r_j, 1))\mathbf{d}_{m-i} = \mathbf{c}_{j-i \bmod m} & \text{if } x_t = r_j, y_{t,2}^{(1)} = 1 \\ \mathbf{H}(\theta(r_j, 0))\mathbf{c}_{m-i} = \mathbf{d}_{j-i \bmod m} & \text{if } x_t = r_j, y_{t,2}^{(1)} = 0 \\ \mathbf{H}(\theta(s_j, 1))\mathbf{d}_{m-i} = \mathbf{d}_{i-j \bmod m} & \text{if } x_t = s_j, y_{t,2}^{(1)} = 1 \\ \mathbf{H}(\theta(s_j, 0))\mathbf{c}_{m-i} = \mathbf{c}_{i-j \bmod m} & \text{if } x_t = s_j, y_{t,2}^{(1)} = 0 \end{cases}$$

This, together with the definition of $\text{dec}^{(2)}$ implies that

$$y_t^{(2)} = \text{dec}^{(2)}(\mathbf{h}_t^{(2)}, \mathbf{y}_t^{(1)}) = \begin{cases} s_{i-j \bmod m} & \text{if } x_t = r_j, y_{t,3}^{(1)} = 1 \\ r_{i-j \bmod m} & \text{if } x_t = s_j, y_{t,3}^{(1)} = 0 \end{cases}.\tag{9}$$

Note that [8] and [9] imply [7]. Setting the output of the linear RNN equal to the output of the second layer concludes the proof.

(ii) It follows from Theorem 4 since D_m is a finite subgroup of $O(2)$, the group of 2D orthogonal transformations: rotations and reflections. \square

1287 B.5 Stability vs. Expressivity of Linear RNNs

1288 In this section, we discuss the tradeoff between expressivity and stability of a linear RNN recurrence
 1289 $\mathbf{H}_i = \mathbf{A}_i \mathbf{H}_{i-1} + \mathbf{B}_i$, where $\mathbf{A}_i = \mathbf{A}(\mathbf{x}_i)$, $\mathbf{B}_i = \mathbf{B}(\mathbf{x}_i)$. We say that such a recurrence is stable if

$$\exists M \in [0, \infty) \text{ such that } \left\| \prod_{j=1}^i \mathbf{A}_j \right\| < M \quad \forall i \in \mathbb{N}, \quad (10)$$

1290 which is true if and only if $\rho(\prod_{j=1}^i \mathbf{A}_j) \leq 1$ where $\rho(\mathbf{M})$ is the spectral radius of \mathbf{M} , i.e. the
 1291 maximum modulus of its eigenvalues. When this property is not satisfied, the norm of the state will
 1292 diverge exponentially fast to infinity. An effective way to satisfy (10) with $M = 1$ is to enforce
 1293 $\|\mathbf{A}_i\| \leq 1$ for every i , since the norm of the product is less than or equal to the product of the norms
 1294 (submultiplicativity). However, this restriction excludes some boolean matrices which are useful
 1295 for recognizing regular languages. Indeed, in the construction shown in (4), all matrices involved
 1296 are $n \times n$ with entries taking values in $\{0, 1\}$ and having only a single one in each column. This
 1297 class of matrices \mathcal{B} satisfies (10) with $M = \sqrt{n}$ because it is closed under matrix multiplication, i.e.
 1298 $\forall \mathbf{B}, \mathbf{B}' \in \mathcal{B}$, we have $\mathbf{B}\mathbf{B}' \in \mathcal{B}$, and $\max_{\mathbf{B} \in \mathcal{B}} \|\mathbf{B}\| = \sqrt{n}$, which is achieved by matrices with
 1299 ones only in one row. All matrices in \mathcal{B} that are not permutations have spectral norm greater than
 1300 one and therefore cannot be expressed if we enforce $\|\mathbf{A}_i\| \leq 1$.

1301 The (Gated) DeltaProduct state transition matrix $\mathbf{A}_i = \prod_{l=1}^{n_h} g_l(I - \beta_l \mathbf{k}_l \mathbf{k}_l^\top)$ satisfies $\|\mathbf{A}_i\| \leq 1$
 1302 since $g_l \in [0, 1]$, $\beta_l \in [0, 2]$, and $\|\mathbf{k}_l\| = 1$. Thus, from the matrices in \mathcal{B} , it can represent only
 1303 permutations of up to $n_h + 1$ elements. The state-transition matrix of RWKV-7, $\mathbf{A}_i = \text{diag}(w_i) -$
 1304 $c\mathbf{k}_i(\mathbf{k}_i \odot \mathbf{a}_i)^\top$ with $c = 2$, can represent not only the identity and permutations of two elements, but
 1305 also any copy matrix, which is obtained by copying one column of the identity onto another and has
 1306 spectral norm equal to $\sqrt{2}$. However, as we show in the next theorem, even with the less expressive
 1307 $c = 1$ setup that is used in practice, the RWKV-7 recurrence is not stable unless \mathbf{a}_i is the same for
 1308 every i , which is the case studied in Peng et al. [13, Theorem 1]. Having different \mathbf{a}_i values is key
 1309 to modeling the copy matrix, since this requires a value different from that of a permutation.

1310 **Theorem 7.** Consider the RWKV-7 state transition matrix $\mathbf{A}_i = \text{diag}(w_i) - c\mathbf{k}_i(\mathbf{k}_i \odot \mathbf{a}_i)^\top$ with
 1311 $c = 1$ (as set in practice), $w_i = (1, \dots, 1)^\top \in \mathbb{R}^n$, $\mathbf{a}_i \in [0, 1]^n$, $\mathbf{k}_i \in \mathbb{R}^n$ with $\|\mathbf{k}_i\| = 1$, and
 1312 $n \geq 2$. There exists an infinite set \mathcal{M} of matrix pairs such that for every $(\mathbf{A}, \mathbf{A}') \in \mathcal{M}$, we have
 1313 $\rho(\mathbf{A}\mathbf{A}') > 1.2$, where ρ denotes the spectral radius. We can set

$$\mathbf{A}_i = \begin{cases} \mathbf{A} & \text{if } i \bmod 2 = 0 \\ \mathbf{A}' & \text{if } i \bmod 2 = 1 \end{cases}, \quad \text{which implies } \lim_{i \rightarrow \infty} \left\| \prod_{j=1}^i \mathbf{A}_j \right\| = \infty.$$

1314 *Proof.* We demonstrate this by construction for $n = 2$; the generalization to $n \geq 2$ is straightfor-
 1315 ward.

1316 Let $\theta = \pi/3$, $\mathbf{a} = (0, 1)^\top$, $\mathbf{a}' = (1, 0)^\top$. $\mathbf{k} = (\cos \theta, \sin \theta)^\top = [1/2, \sqrt{3}/2]^\top$. $\mathbf{k}' =$
 1317 $(\sin \theta, \cos \theta)^\top = (\sqrt{3}/2, 1/2)^\top$. Note that $\|\mathbf{k}\| = \|\mathbf{k}'\| = 1$. We construct \mathbf{A} and \mathbf{A}' as

$$\begin{aligned} \mathbf{A} &= I - \mathbf{k}(\mathbf{k} \odot \mathbf{a})^\top = I - \begin{pmatrix} \sqrt{3}/2 \\ 1/2 \end{pmatrix} \begin{bmatrix} 0 & 1/2 \end{bmatrix} = \begin{pmatrix} 1 & -\sqrt{3}/4 \\ 0 & 3/4 \end{pmatrix} \\ \mathbf{A}' &= I - \mathbf{k}'(\mathbf{k}' \odot \mathbf{a}')^\top = I - \begin{pmatrix} 1/2 \\ \sqrt{3}/2 \end{pmatrix} \begin{bmatrix} 1/2 & 0 \end{bmatrix} = \begin{pmatrix} 3/4 & 0 \\ -\sqrt{3}/4 & 1 \end{pmatrix} \end{aligned}$$

1318 Now, consider the product matrix

$$\mathbf{M} = \mathbf{A}\mathbf{A}' = \begin{pmatrix} 1 & -\sqrt{3}/4 \\ 0 & 3/4 \end{pmatrix} \begin{pmatrix} 3/4 & 0 \\ -\sqrt{3}/4 & 1 \end{pmatrix} = \begin{pmatrix} 15/16 & -\sqrt{3}/4 \\ -3\sqrt{3}/16 & 3/4 \end{pmatrix}$$

1319 To find the spectral radius $\rho(\mathbf{M})$, we examine its eigenvalues. The characteristic equation is $\lambda^2 -$
 1320 $\text{Tr}(\mathbf{M})\lambda + \det(\mathbf{M}) = 0$, with $\text{Tr}(\mathbf{M}) = 27/16$ and $\det(\mathbf{M}) = 9/16$. Hence, the characteristic
 1321 equation is $16\lambda^2 - 27\lambda + 9 = 0$. Thus, the eigenvalues are $\lambda_1 = \frac{27+\sqrt{153}}{32}$ and $\lambda_2 = \frac{27-\sqrt{153}}{32}$ and
 1322 the spectral radius is $\rho(\mathbf{M}) = \max\{|\lambda_1|, |\lambda_2|\} = \lambda_1 \approx 1.23$. Also, since the spectral radius is a

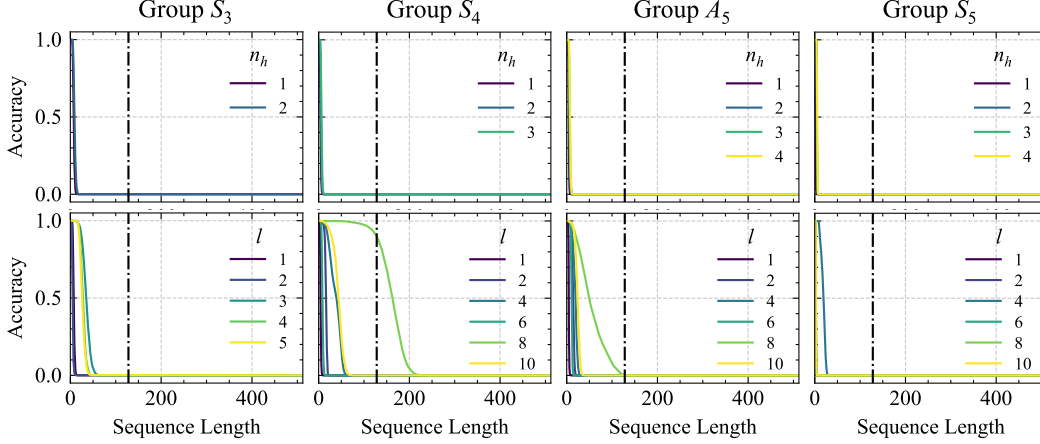


Figure 11: Results for permutation groups S_3 , S_4 , A_5 , and S_5 when limiting the eigenvalue range of the state-transition matrix to $[0, 1]$. (Top row) Varying the number of Householder products n_h for a single layer $\text{DeltaProduct}_{n_h}[0, 1]$. (Bottom row) Varying the number of layers l of $\text{DeltaProduct}_1[0, 1]/\text{DeltaNet}[0, 1]$ (single Householder). Dashed vertical line at training context length 128. Higher n_h improves extrapolation to longer sequences of permutations, e.g., S_3 can be learned with $n_h = 2$ with a single layer while three layers are required when keeping $n_h = 1$.

continuous function of the matrix entries, which are a continuous function of θ , then this means that there is an infinite set of matrices obtained by varying θ around $\pi/3$, namely \mathcal{M} , whose product has spectral radius greater than 1.2.

From our construction of A_i , we have $\prod_{j=1}^{2i} A_j = M^i$. By the definition of the spectral norm and spectral radius, $\|M^i\| \geq \|\lambda_1^i x\| = |\lambda_1|^i = \rho(M)^i$, where x is the eigenvector associated with the dominant eigenvalue λ_1 . The result follows since $\lim_{i \rightarrow \infty} \rho(M)^i = \infty$. \square

C Experiments

C.1 State-Tracking

Clarification on the isomorphisms of S_3 , S_4 , A_5 , and S_5

S_3 : The group consisting of all isometries that map an equilateral triangle onto itself, including both orientation-preserving rotations and orientation-reversing reflections, is isomorphic to S_3 .

S_4 : The rotation group of a cube is isomorphic to the symmetric group S_4 . This correspondence arises because the cube has exactly four space diagonals, and every *proper rotation*—that is, every orientation-preserving isometry of the cube about an axis through its center—permutes these diagonals in all possible ways (see Figure 6 for an example). In particular, these proper rotations include, for example, the 90° , 180° , and 270° rotations about axes passing through the centers of opposite faces, the 180° rotations about axes through the midpoints of opposite edges, and the $120^\circ/240^\circ$ rotations about axes through opposite vertices. Hence, the proper rotational symmetries of the cube correspond precisely to the permutations of its four space diagonals [67].

A_5 : Similarly, a regular dodecahedron contains exactly five special cubes symmetrically arranged within it. Each *proper rotation* of the dodecahedron—that is, every orientation-preserving rigid motion mapping the dodecahedron onto itself—rearranges these inscribed cubes by an *even permutation*. This property makes the rotation group of the dodecahedron isomorphic to the alternating group A_5 , the group of all even permutations of five elements [68].

S_5 : When both proper rotations and reflections (orientation-reversing symmetries) are considered, the full symmetry group of the dodecahedron corresponds exactly to the symmetric group S_5 , since reflections allow both even and odd permutations of the five hidden cubes [68].

Experimental Details. We used the experimental setup from Merrill et al. [3] and sampled 2,000,000 training datapoints at sequence length 128 and 500,000 test datapoints at sequence length

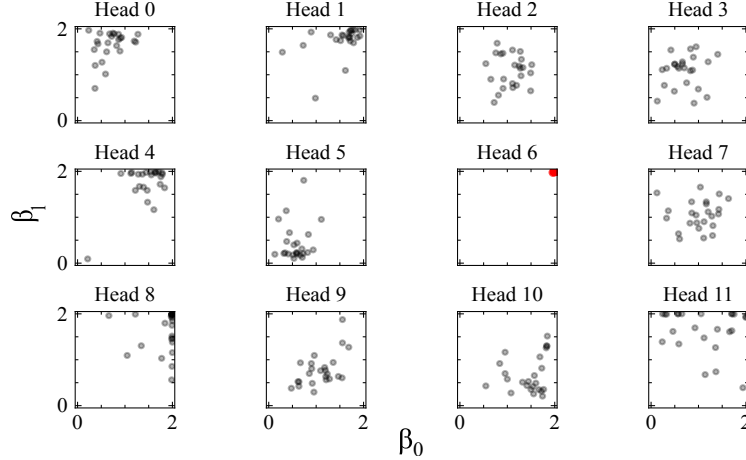


Figure 12: β_0 and β_1 values across all 24 permutations in S_4 in $\text{DeltaProduct}_2[-1, 1]$. We find that only head 6 (shown in Figure 7) learns to use both Householders as reflections ($\beta_0 \approx 2$, $\beta_1 \approx 2$) allowing it to learn the rotations to solve S_4 .

512. We did not use a curriculum over sequence length during training. The models were trained using AdamW optimizer [69] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ in PyTorch [70]. We used a learning rate of 10^{-3} with cosine annealing [71] and trained for 100 epochs with a batch size of 1024, except for the S_3 models which required a batch size of 2048 for more reliable results. All models used a single-layer DeltaProduct architecture featuring 12 heads (more heads made the results more reliable) and a head dimension of 32. We applied a weight decay coefficient of 10^{-6} . The β values were extracted from the forward pass of the trained models using NNsight [72]. We use the PCA implementation in scikit-learn [73].

C.2 Chomsky Hierarchy

Setup. We conducted experiments on selected formal language tasks originally introduced by Delétang et al. [50]. Our goal was to demonstrate the improvements in length extrapolation that can be achieved using multiple Householder matrices in the state-transition matrix compared to DeltaNet. Following Grazi et al. [16], we focus on three tasks: parity, modular arithmetic without brackets (both regular languages), and modular arithmetic with brackets (a context-free language). We trained $\text{DeltaProduct}_{n_h}$ with $n_h \in \{2, 3, 4\}$ on sequences of length 3 to 40 and tested on sequences ranging from 40 to 256 to evaluate generalization to longer inputs. We compare our results against the results obtained by Grazi et al. [16] for Transformer, mLSTM and sLSTM from Beck et al. [9], Mamba [6], and DeltaNet [10]. For both Mamba and DeltaNet, we experiment with an eigenvalue range restricted to $[0, 1]$ and extended to $[-1, 1]$.

Experimental Details. All DeltaProduct and DeltaNet models contain 3 layers with 1 head each and heads' dimensions set to 128, except for modular arithmetic with brackets, where we use 12 heads and set the heads' dimensions to 32. Both models use a causal depthwise 1D convolution with a kernel size of 4 after the query/key/value projection. For modular arithmetic, we also use a gradient clipping norm of 1.0. We train each model using AdamW [69] using a learning rate of $5e-4$, batch size of 1024, 0.1 weight decay, and a cosine annealing learning rate schedule [71] (minimum learning rate: $1e-6$) after 10% warm-up steps. We train on the modular arithmetic and parity tasks for 100k and 20k steps in total, respectively. At each training step, we make sure to generate a valid random sample from the task at hand (see below). We repeat the runs 3 times with different seeds each, and later pick the best to report in Table 2.

Considered Tasks. We empirically evaluated three tasks—parity, modular arithmetic without brackets, and modular arithmetic with brackets—spanning different levels of the Chomsky Hierarchy. Below, we provide details for each task, where $|\Sigma|$ denotes the vocabulary size and Acc_{rand} represents the accuracy of random guessing:

Table 2: Performance of $\text{DeltaProduct}_{n_h}[-1, 1]$, $n_h \in \{2, 3, 4\}$, on formal language tasks. We report the best of 3 runs. Scores are scaled accuracy, with 1.0 indicating perfect performance and 0.0 random guessing. The results for the other models were taken directly from Grazzi et al. [16].

Model	Parity	Mod. Arithm. (w/o brackets)	Mod. Arithm. (w/ brackets)	Avg.
Transformer	0.022	0.031	0.067	0.040
mLSTM	0.087	0.040	0.114	0.080
sLSTM	1.000	0.787	0.178	0.655
Mamba $[0, 1]$	0.000	0.095	0.123	0.073
Mamba $[-1, 1]$	1.000	0.241	0.116	0.452
DeltaNet $[0, 1]$	0.233	0.302	0.253	0.263
DeltaProduct ₂ $[0, 1]$	0.264	0.402	0.249	0.305
DeltaProduct ₃ $[0, 1]$	<u>0.285</u>	0.402	<u>0.288</u>	0.325
DeltaProduct ₄ $[0, 1]$	0.295	<u>0.369</u>	0.288	0.317
DeltaNet $[-1, 1]$	0.982	0.915	0.281	0.726
DeltaProduct ₂ $[-1, 1]$	0.896	0.887	0.329	0.704
DeltaProduct ₃ $[-1, 1]$	<u>0.932</u>	0.736	<u>0.330</u>	0.666
DeltaProduct ₄ $[-1, 1]$	0.982	<u>0.893</u>	0.342	0.739

- 1385 • **Parity** ($|\Sigma| = 2$, $Acc_{rand} = 0.5$). Given a binary sequence $\mathbf{x} = x_1 \dots x_t \in \{0, 1\}^t$, the parity
1386 label $y_t \in \{0, 1\}$ is 1 if the total number of ones in the sequence is odd, and 0 otherwise. This task
1387 is equivalent to computing the sum of all previous values modulo 2, i.e., $y_t = (\sum_{i=1}^t x_i) \bmod 2$.
- 1388 • **Modular Arithmetic without Brackets** ($|\Sigma| = 10$, $Acc_{rand} = 1/5$). Given a set of special
1389 tokens $\Sigma_s = \{+, -, *, =, [\text{PAD}]\}$ and a modulus $m \geq 1$, we define $\Sigma = \Sigma_s \cup \{0, \dots, m-1\}$.
1390 The label y_t corresponds to the result of evaluating the arithmetic operations in the sequence
1391 $\mathbf{x} = x_1, \dots, x_t$, computed modulo m . In our experiments, we set $m = 5$. An example is:

$$2 + 1 - 2 * 2 - 3 = \textcolor{red}{1} [\text{PAD}]$$

- 1392 • **Modular Arithmetic with Brackets** ($|\Sigma| = 12$, $Acc_{rand} = 1/5$). This task follows the same
1393 definition as modular arithmetic without brackets but includes an extended set of special tokens,
1394 $\Sigma_s = \{+, -, *, =, (,), [\text{PAD}]\}$, allowing for nested expressions. Again, we set $m = 5$. An example
1395 sequence is:

$$((1 - (-2)) + ((4) + 3)) = \textcolor{red}{0} [\text{PAD}]$$

1396 **Results.** As shown in Table 2, $\text{DeltaProduct}_{n_h}$ with $n_h \geq 2$ has better average accuracy compared
1397 to DeltaNet and other baselines. This performance improvement is particularly pronounced when
1398 using the extended eigenvalue range $[-1, 1]$, which aligns with the findings of Grazzi et al. [16].
1399 Notably, we observe the most significant improvement in the modular arithmetic with brackets task,
1400 which is also the most challenging.

1401 C.3 Language Modeling

1402 C.3.1 Experimental setup

1403 We follow the same basic training setup as in [16]. We use the training pipeline `flame` from the
1404 flash-linear-attention [52] repository. All of our models are trained on NVIDIA L40s or NVIDIA
1405 A100 40GB or NVIDIA H100 94GB GPUs. We used 16 to 32 GPUs at a time to train one model,
1406 in either a 2 to 8 node setup, depending on resource availability. We used DeepSpeed with ZeRO-
1407 2 [74] for distributed training. All models were trained for 66 758 steps with an effective batch size
1408 of 524 288 tokens, a learning rate of $3e-4$, and a training context length of 2 048 tokens. We used two
1409 steps of gradient accumulation in the 16 GPU setup. We optimized the models with AdamW [69]
1410 (0.01 weight decay) and used cosine annealing [71] for the learning rate schedule with linear warm
1411 up for 512 steps. We used a total of 10 500 GPU hours to train all of our models.

Table 3: Performance comparison of models shown in Figure 10. To account for the increased parameter count we scaled the training token budget from 19B (213M parameters) to 55B (805M parameters) on FineWeb [56]. Models were trained on 4096 token context length.

	Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	Avg. ↑
19B / 213M	DeltaNet[-1, 1]	31.96	85.36	22.5	65.2	35.4	50.8	44.7	22.4	40.17
	DeltaProduct ₂ [-1, 1]	30.87	89.23	23.1	65.4	36.5	51.2	43.5	22.4	40.35
	DeltaProduct ₃ [-1, 1]	30.85	71.52	24.1	66.3	36.1	51.6	44.1	23.9	41.02
35B / 392M	DeltaNet[-1, 1]	24.86	39.1	30.8	69.2	41.4	50.5	46.7	24.4	43.83
	DeltaProduct ₂ [-1, 1]	24.97	35.68	31.9	69.6	42.5	52.6	47.4	25.9	44.98
	DeltaProduct ₃ [-1, 1]	25.2	40.96	30.5	69.1	42.3	51.4	47.7	23.9	44.15
55B / 805M	DeltaNet[-1, 1]	20.6	21.18	38.7	71.5	48.7	52.8	51.9	25.7	48.22
	DeltaProduct ₂ [-1, 1]	20.26	17.41	40.7	72.6	50.3	53.9	52.4	24.9	49.13
	DeltaProduct ₃ [-1, 1]	19.97	17.78	40.79	72.3	50.9	52.1	53.9	26.4	49.4

Table 4: Performance comparison of models trained with 2048 context length. (SlimPajama (SPJ) reproduced from Yang et al. [10], Fine-Web (FW) ours). Results are shown for DeltaProduct and Gated DeltaProduct. We use 8 heads for each layer, unless otherwise specified.

	Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	Avg. ↑
15B tokens SPJ	340M params									
	Transformer++	28.39	42.69	31.0	63.3	34.0	50.4	44.5	24.2	41.2
	Mamba [0, 1]	28.39	39.66	30.6	65.0	35.4	50.1	46.3	23.6	41.8
	GLA [0, 1]	29.47	45.53	31.3	65.1	33.8	51.6	44.4	24.6	41.8
	DeltaNet [0, 1]	28.24	37.37	32.1	64.8	34.3	52.2	45.8	23.5	42.1
35B FW	DeltaNet[-1, 1] 340M	26.92	43.07	29.8	69.0	41.0	50.9	46.6	24.5	43.6
	DeltaNet[-1, 1] 12 heads, 392M	26.57	36.76	31.8	69.2	42.3	50.9	47.2	24.4	44.3
	DeltaProduct ₂ [-1, 1] 392M	26.43	30.66	34.0	68.9	42.4	53.1	48.9	25.9	45.5
	DeltaProduct ₃ [-1, 1] 443M	25.94	29.91	34.2	69.9	43.2	51.9	48.2	24.1	45.2
	Gated DeltaNet[-1, 1] 340M	25.97	33.57	33.1	69.5	44.1	51.1	50.9	26.7	45.9
	Gated DeltaProduct ₂ [-1, 1] 393M	25.12	<u>30.03</u>	34.2	69.1	<u>44.6</u>	55.3	<u>49.8</u>	25.3	46.4

1412 C.4 Throughput

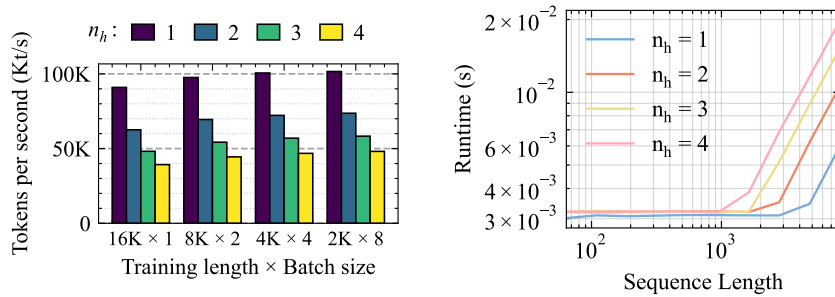


Figure 13: (Left) Throughput for different values of n_h and sequence lengths × batch sizes for 340M DeltaProduct _{n_h} on a H100-94GB. (Right) Inference latency for 340M DeltaProduct _{n_h} for sequences up to 8.192 tokens.

1413 C.4.1 Additional Benchmarks

1414 In Table 3 we report evaluations for the models in Figure 10 on tasks from lm-eval-harness [60]. In
 1415 addition, we also train and evaluate models with 2048 context length at the 340M parameter scale
 1416 and report the results in Table 4 and compare them with the results in [10] which are trained under a
 1417 comparable setup. We observe that DeltaProduct outperforms DeltaNet in terms of average accuracy
 1418 for both training setups.

1419 **Tasks Details.** We use the lm-eval-harness benchmark [60] to assess model performance. Following
 1420 Yang et al. [10], the evaluation encompasses multiple task categories: **Language Understanding**

1421 **Tasks.** The evaluation includes LAMBADA (LMB) [75] for testing text comprehension, PIQA
 1422 [76] for physical reasoning assessment, HellaSwag (Hella.) [77] for situational understanding, and
 1423 Winogrande (Wino.) [78] for commonsense reasoning evaluation. **Reasoning.** The ARC dataset
 1424 provides two distinct testing sets: ARC-easy (ARC-e) and ARC-challenge (ARC-c) [79], measuring
 1425 varying levels of scientific knowledge comprehension.

1426 C.4.2 Training behavior

1427 The training behavior of $\text{DeltaProduct}_{n_h}$ is stable as shown in Figure 14. This is also true for all
 1428 considered model sizes in Figure 10 and Appendix C.4.3

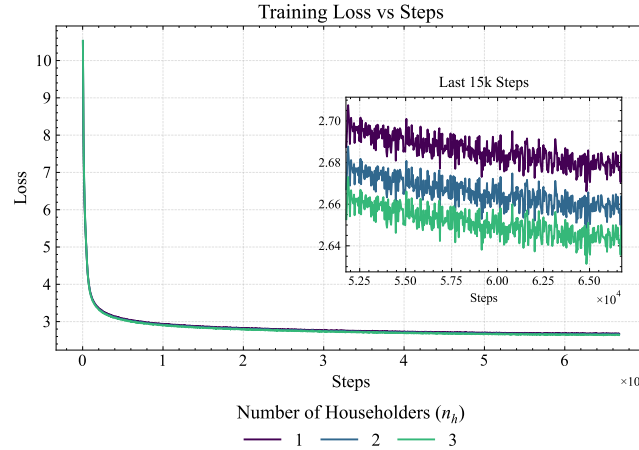


Figure 14: Training loss curves of $\text{DeltaProduct}_{n_h}[-1, 1]$. The curves demonstrate stable training behavior as n_h increases, with higher values of n_h consistently yielding lower losses throughout training and convergence. While the absolute differences in loss between different n_h values are relatively small, they correspond to significant differences in length extrapolation performance.

1429 C.4.3 Additional results on Length Extrapolation

1430 In this section we show additional plots on length extrapolation. In Figure 15 we show the length
 1431 extrapolation behavior of (Gated) $\text{DeltaProduct}_{n_h}$ scaling up n_h without adjusting any of the other
 1432 model configuration parameters. As discussed in Section 5.3, increasing n_h increases the param-
 1433 eter count of the model. Hence, Figures 16 and 17 show the per-token loss and perplexity of
 1434 $\text{DeltaProduct}_{n_h}$ at three different scales where the parameter counts are matched at the respective
 1435 scales following the configuration parameters shown in Table 5. Note that these are the same models
 1436 as shown in Figure 10

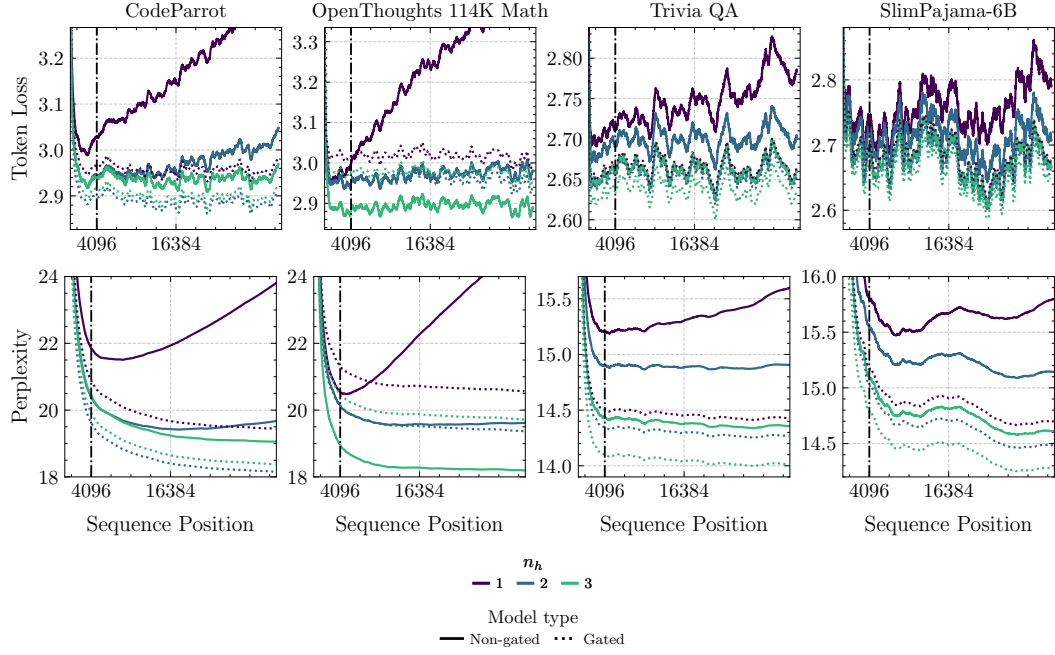


Figure 15: Per token loss and perplexity on context lengths up to 32,768 for (Gated) DeltaProduct_{n_h}. (Top) per token loss. (Bottom) Perplexity. Per token losses smoothed with a window-size of 300.

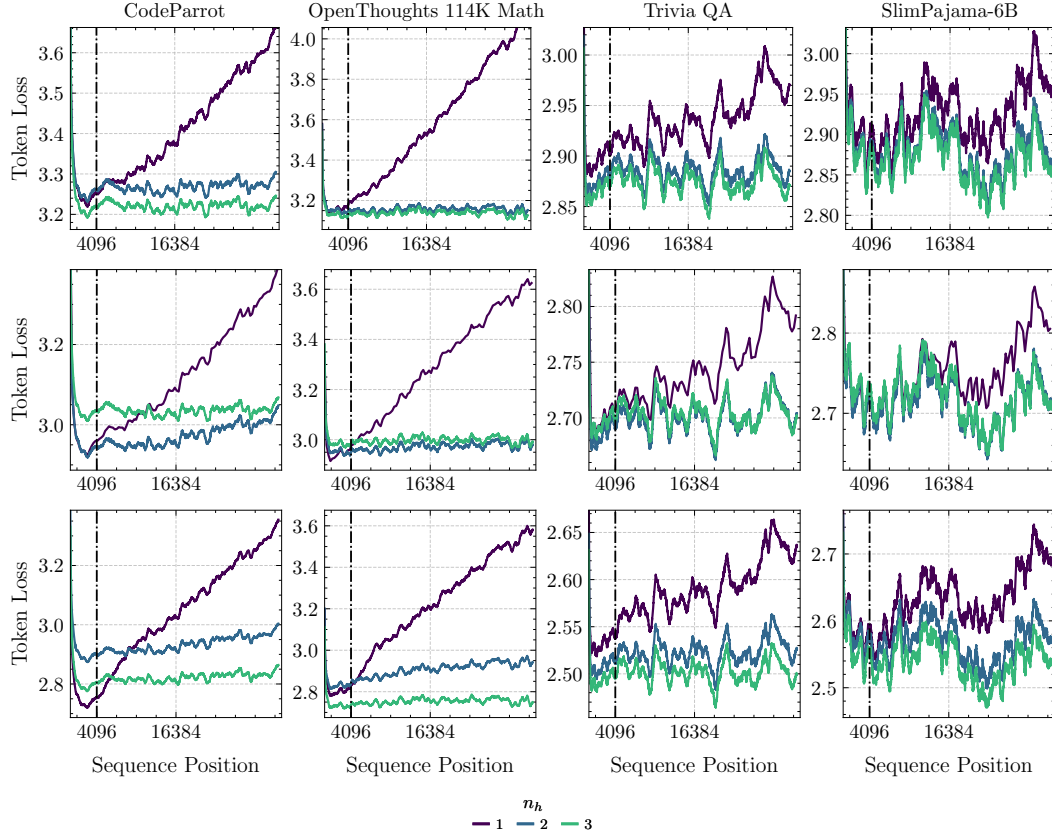


Figure 16: Per token loss of $\text{DeltaProduct}_{n_h}$ on contexts up to 32.768 at different model sizes. Models are parameter equivalent at each scale. Parameter equivalence is achieved by scaling the number of heads. Exact model configurations can be found in Table 5 (Top) 213M parameters. (Middle) 392M parameters. (Bottom) 805M parameters.

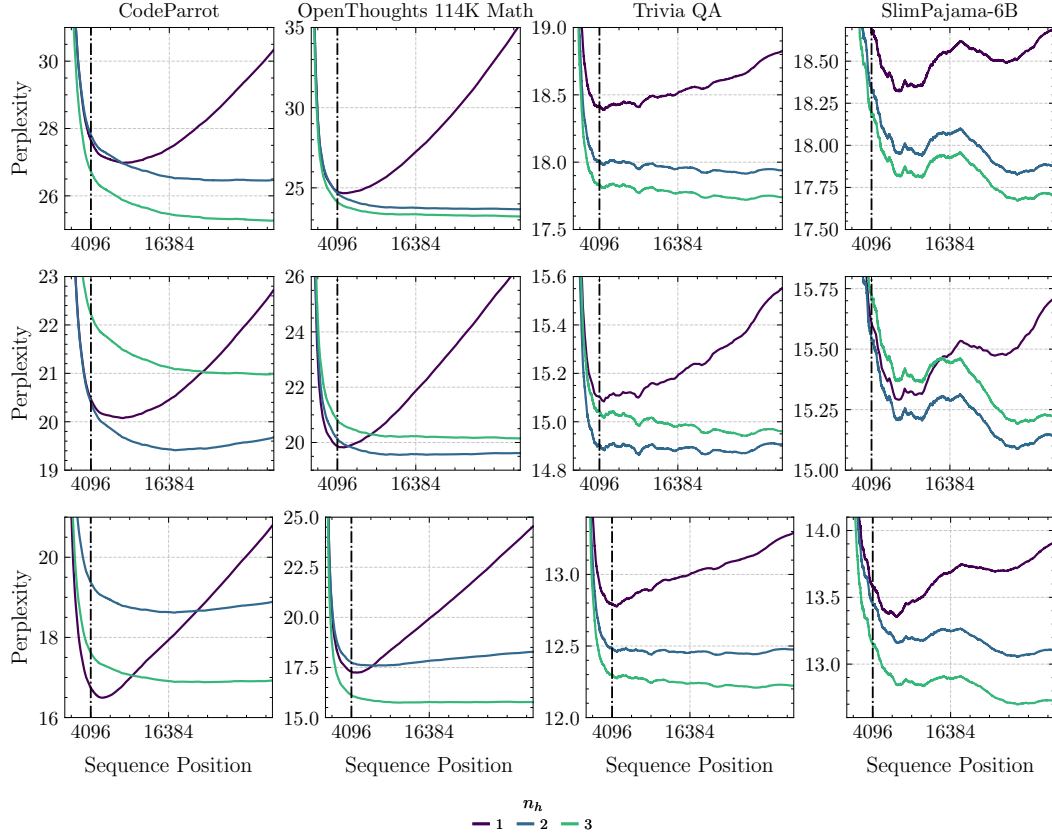


Figure 17: Perplexity analogue of Figure 16

Table 5: Model configuration parameters for models shown in Figures 16 and 17. All other configuration parameters are the same as in 16.

Model Scale	# Householders	Hidden size	# Heads
213M	1	768	8
	2	736	6
	3	768	4
392M	1	1024	12
	2	1024	8
	3	1024	6
805M	1	1536	16
	2	1468	12
	3	1536	8

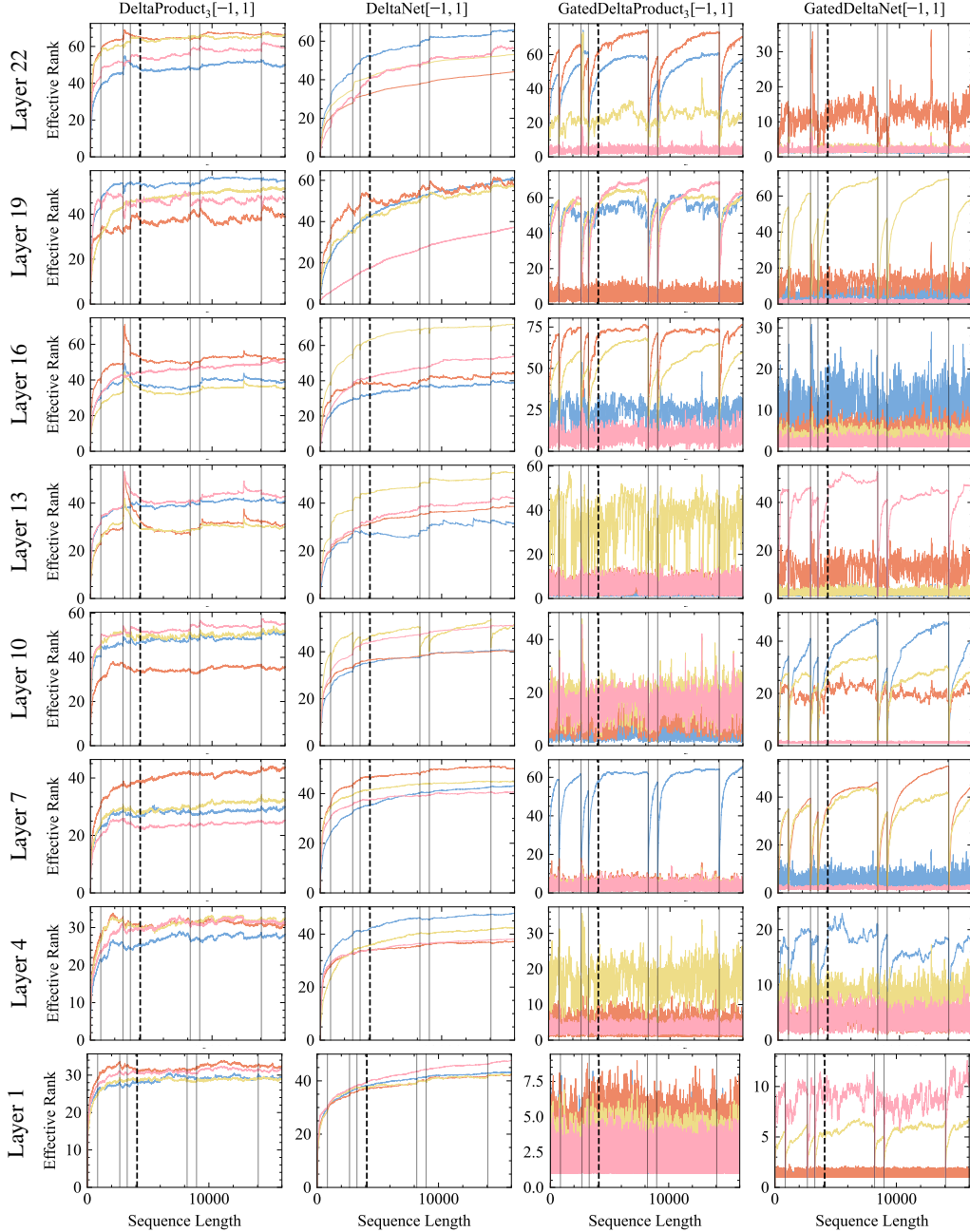


Figure 18: Effective rank of H_i for 4 of 8 heads for a selection of the layers on **CodeParrot** sequences. Solid vertical lines mark new code sequences; dashed vertical line indicates 4096-token training context length; colored lines show effective rank per head over the sequence.

1437 C.4.4 Additional Results on Scaling Behavior

1438 In Figure 10 parameter equivalence is achieved at each scale mainly by decreasing the number of
 1439 heads for models with $n_h > 1$. In Figure 20 we show perplexity of FineWeb for another set of
 1440 scaling results where parameter equivalence is reached solely by reducing the dimensionality of
 1441 each head. The result for this alternative type of scaling still shows the superiority of DeltaProduct
 1442 compared to DeltaNet. Moreover, in this case, models with higher n_h are strictly better than those
 1443 with fewer householders.

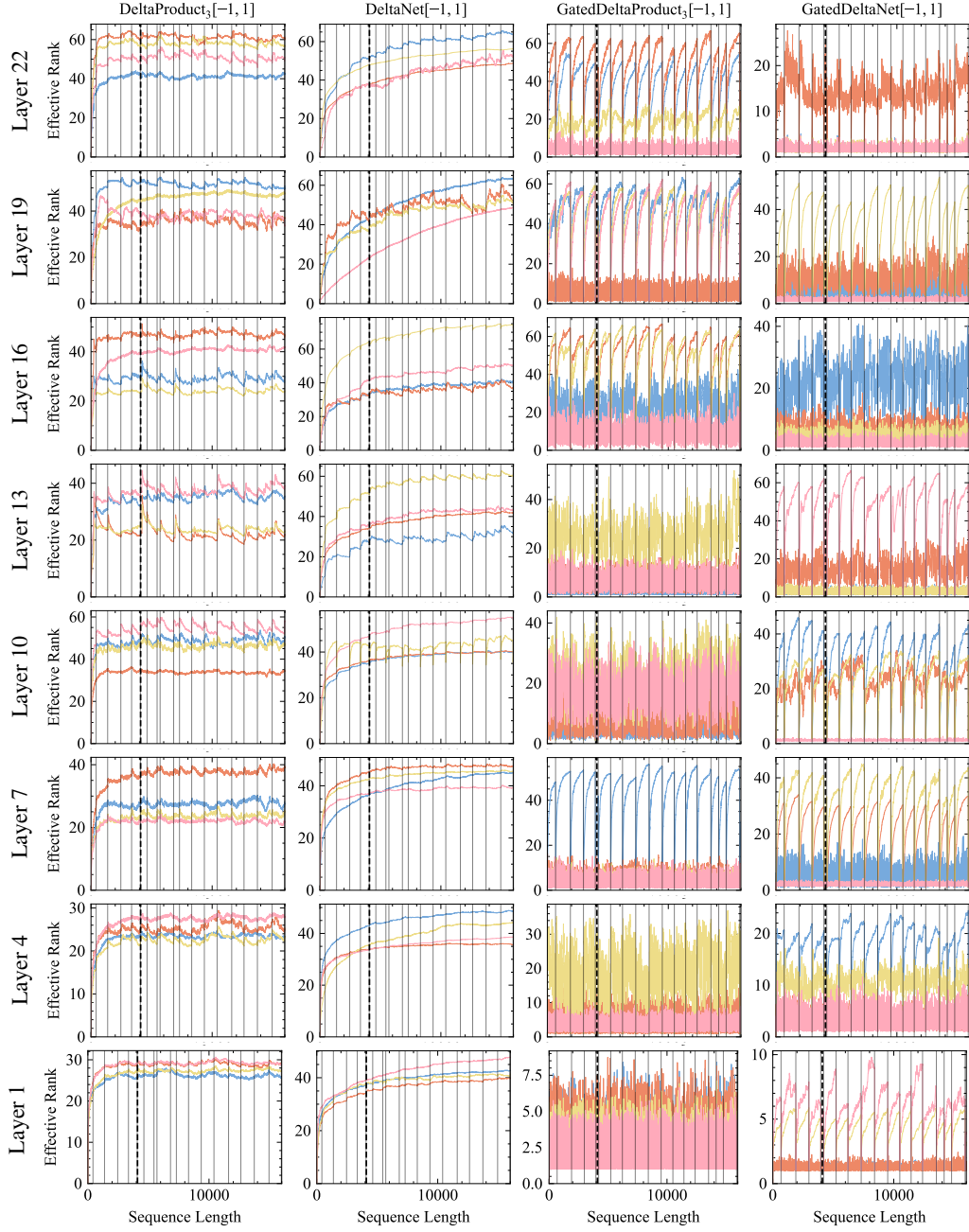


Figure 19: Effective rank of H_i for 4 of 8 heads for a selection of the layers on **TriviaQA** sequences. Solid vertical lines mark new question-answer pairs; dashed vertical line indicates 4096-token training context length; colored lines show effective rank per head over the sequence.

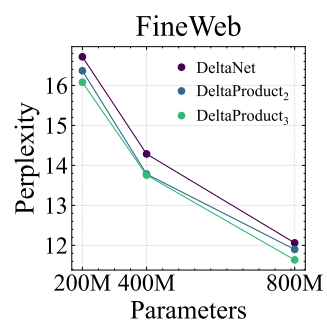


Figure 20: Scaling results on FineWeb where parameter equivalence is achieved by scaling the head dimension. Models trained at each scale with number of tokens as reported in Table 3.