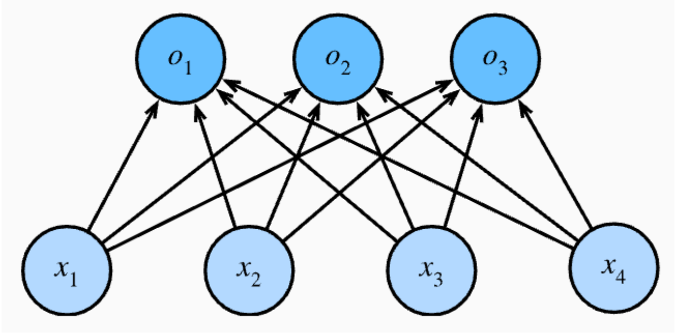
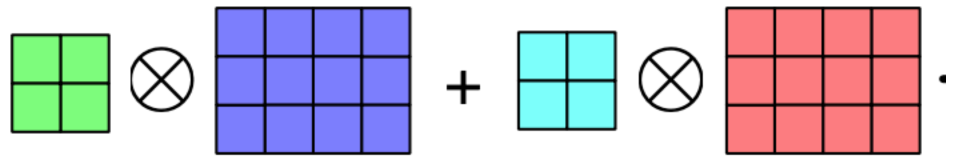
Beyond Fully-Connected Layers with Quaternions: Parameterization of Hypercomplex Multiplications with $1/n$ Parameters

Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Cheung Hui, Jie Fu

ICLR'21

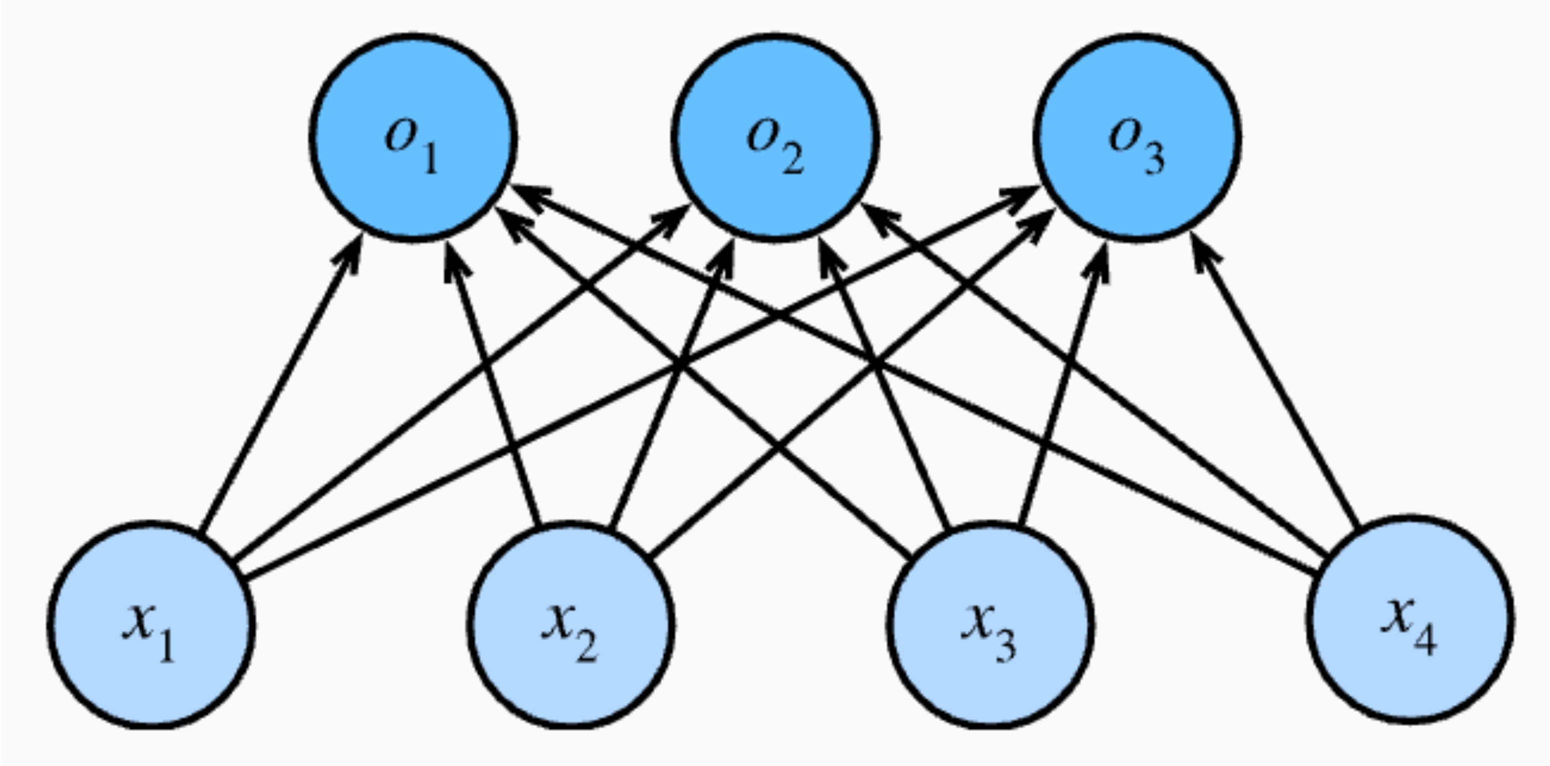




Fully-connected (FC) layers are pervasive

$$\mathbf{y} = \mathrm{FC}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \boldsymbol{b}$$

$\mathbb{R}^k$ $\mathbb{R}^d$

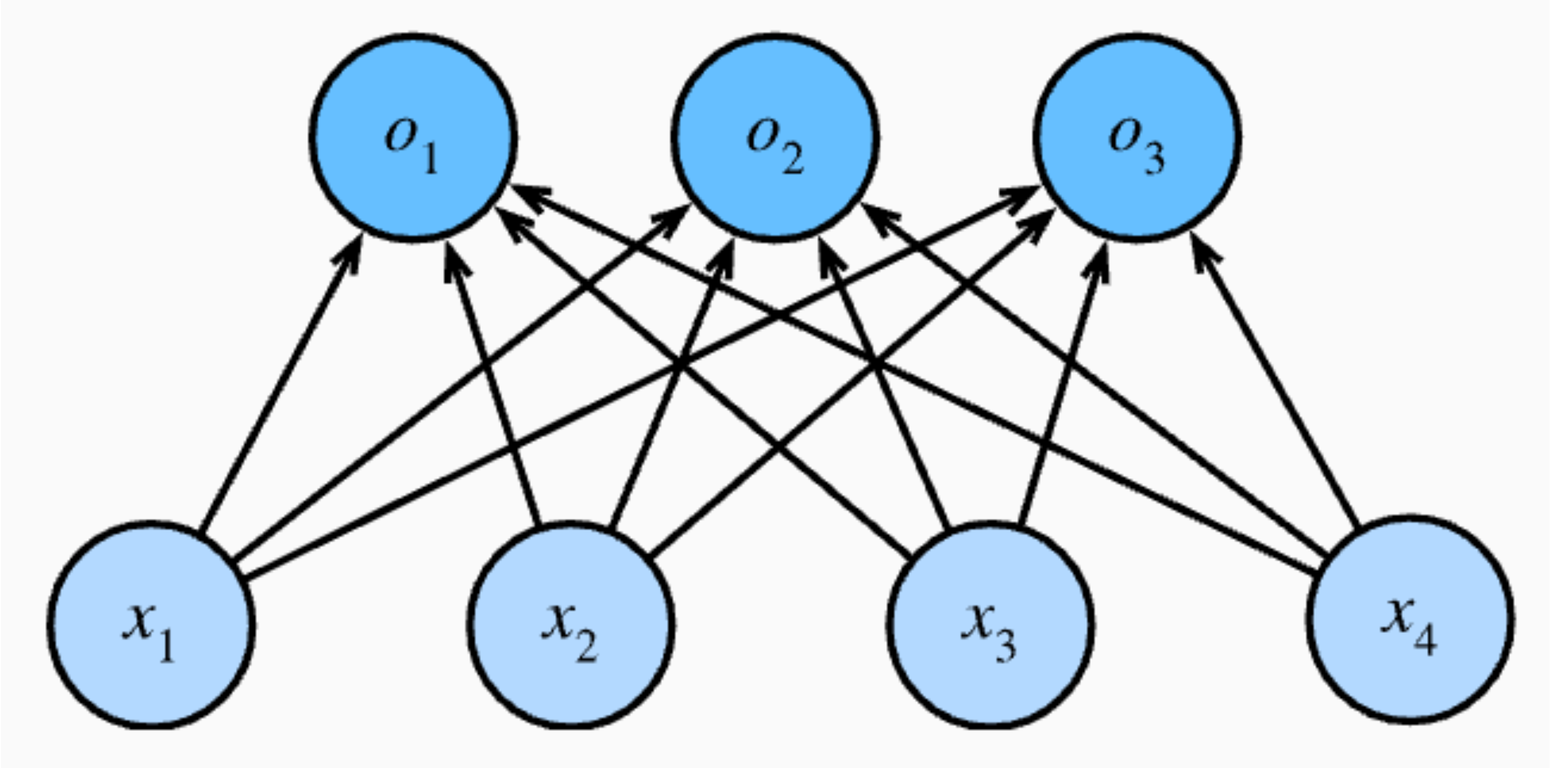


Zhang et al. Dive into Deep Learning
https://d2l.ai/chapter_linear-networks/softmax-regression.html

FC layers are heavily parameterized

$$\mathbf{y} = \mathrm{FC}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \boldsymbol{b}$$

$\mathbb{R}^k$ $\mathbb{R}^d$



Parameterization cost:

$$\mathcal{O}(kd)$$

Zhang et al. Dive into Deep Learning
https://d2l.ai/chapter_linear-networks/softmax-regression.html

FC layers with quaternions

- **Quaternion**: 4-dimensional hypercomplex number with one real component and three imaginary components

$$Q = Q_r + Q_x \mathbf{i} + Q_y \mathbf{j} + Q_z \mathbf{k}$$

- **Hamilton product**: (multiplication of two quaternions Q and P):

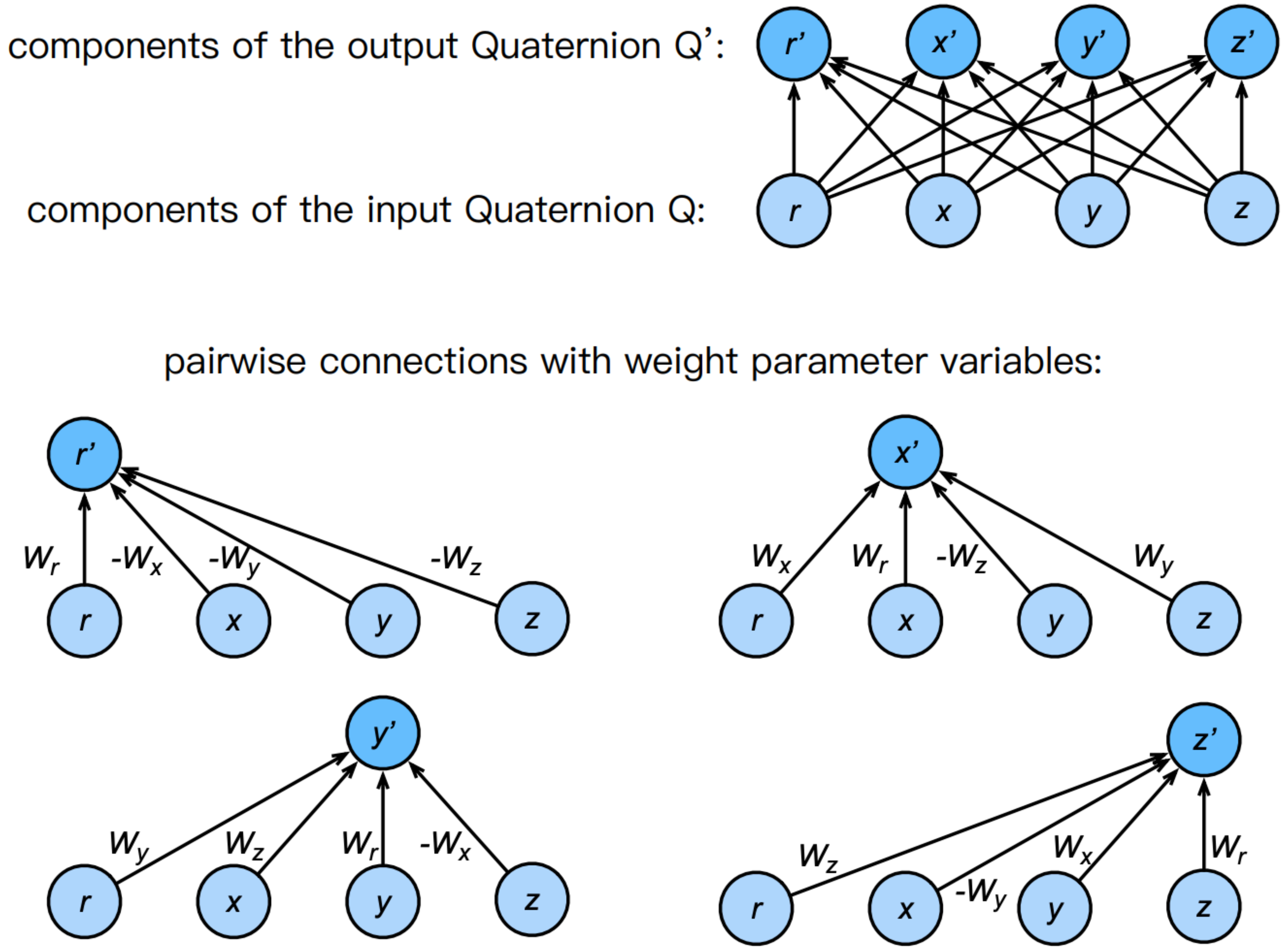
$$\begin{aligned} Q \otimes P = &(Q_r P_r - Q_x P_x - Q_y P_y - Q_z P_z) + (Q_x P_r + Q_r P_x - Q_z P_y + Q_y P_z)\,\mathbf{i} \\ &+ (Q_y P_r + Q_z P_x + Q_r P_y - Q_x P_z)\,\mathbf{j} + (Q_z P_r - Q_y P_x + Q_x P_y + Q_r P_z)\,\mathbf{k} \end{aligned}$$

- **FC layers with quaternions**: replace real-valued matrix multiplications in FC layers with Hamilton products of quaternions

FC layers with quaternions has 1/4 parameters

By re-using parameters 4 times via Hamilton Products

$$\begin{bmatrix} W_r & -\boxed{W_x} & -W_y & -W_z \\ \boxed{W_x} & W_r & -W_z & W_y \\ W_y & W_z & W_r & -\boxed{W_x} \\ W_z & -W_y & \boxed{W_x} & W_r \end{bmatrix} \begin{bmatrix} r \\ x \\ y \\ z \end{bmatrix}$$



Tay et al. Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks (ACL'19)

We propose to parameterize hypercomplex multiplications

- Hypercomplex multiplication rules only exist at very few predefined dimensions (4D, 8D, and 16D)

- We learn multiplication rules from data regardless of whether such rules are predefined

- We provide more architectural flexibility using arbitrarily $1/n$ learnable parameters compared with the FC layer counterpart

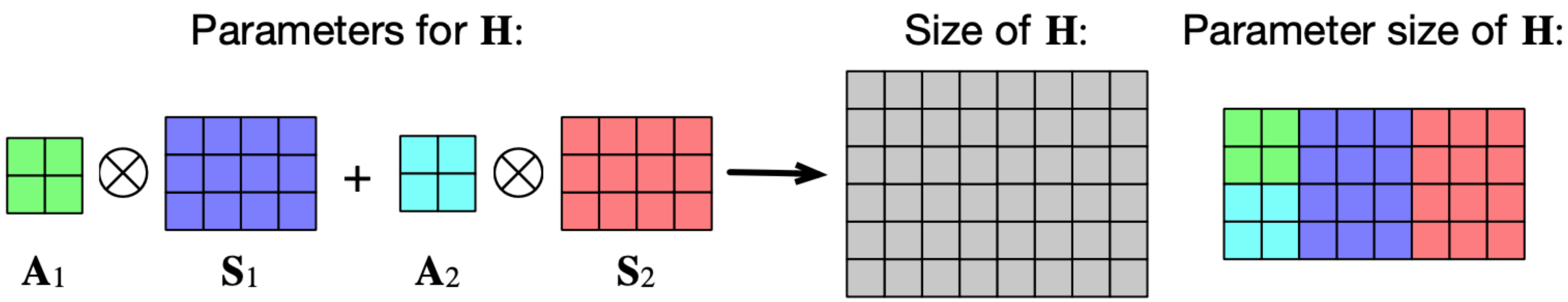
We propose parameterized hypercomplex multiplication (PHM) layers with $1/n$ parameters

$$\underset{\mathbb{R}^k}{\mathbf{y}} = \mathrm{PHM}(\underset{\mathbb{R}^d}{\mathbf{x}}) = \mathbf{Hx} + \boldsymbol{b}$$

By re-using parameters n times via sum of $\boxed{n}$ Kronecker Products

$$\mathbf{H} = \sum_{i=1}^{\boxed{n}} \mathbf{A}_i \otimes \mathbf{S}_i$$

$$\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{11}\mathbf{Y} & \dots & x_{1n}\mathbf{Y} \\ \vdots & \ddots & \vdots \\ x_{m1}\mathbf{Y} & \dots & x_{mn}\mathbf{Y} \end{bmatrix}$$



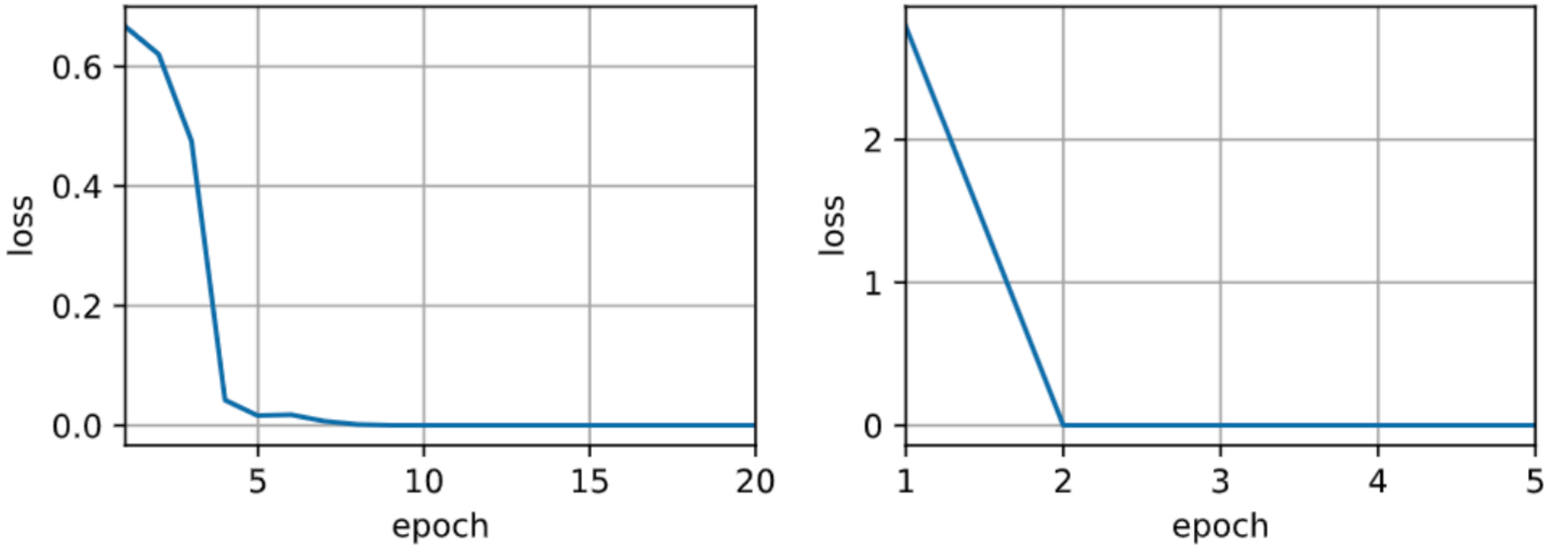
Parameterization cost: $\mathcal{O}(kd/n)$

PHM layers subsume hypercomplex multiplications and real-valued matrix multiplications

Hamilton product by PHM layers

$$\underbrace{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{\mathbf{A}_1}\otimes\underbrace{\begin{bmatrix}Q_r\end{bmatrix}}_{\mathbf{S}_1}+\underbrace{\begin{bmatrix}0&-1&0&0\\1&0&0&0\\0&0&0&-1\\0&0&1&0\end{bmatrix}}_{\mathbf{A}_2}\otimes\underbrace{\begin{bmatrix}Q_x\end{bmatrix}}_{\mathbf{S}_2}+\underbrace{\begin{bmatrix}0&0&-1&0\\0&0&0&1\\1&0&0&0\\0&-1&0&0\end{bmatrix}}_{\mathbf{A}_3}\otimes\underbrace{\begin{bmatrix}Q_y\end{bmatrix}}_{\mathbf{S}_3}+\underbrace{\begin{bmatrix}0&0&0&-1\\0&0&-1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}}_{\mathbf{A}_4}\otimes\underbrace{\begin{bmatrix}Q_z\end{bmatrix}}_{\mathbf{S}_4}$$

PHM layers can learn to perform existing multiplication rules



(a) Learning rotations in 3D real space

(b) Learning Hamilton products in Quaternion space

PHM-LSTM and PHM-transformer replace FC layers or matrix multiplications with PHM layers

PHM-LSTM

$$
\begin{aligned}
\mathbf{y}_t &= \mathrm{PHM}\left(\mathbf{x}_t\right) + \mathrm{PHM}\left(\mathbf{h}_{t-1}\right) + \boldsymbol{b} \\
\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t, \mathbf{x}'_t &= \phi(\mathbf{y}_t) \\
\mathbf{c}_t &= \sigma_s(\mathbf{f}_t)\, \mathbf{c}_{t-1} + \sigma_s(\mathbf{i}_t)\, \sigma_t(\mathbf{x}'_t) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t,
\end{aligned}
$$

PHM-transformer

Single-head self-attention:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \Phi(\mathrm{PHM}(\mathbf{X}))$$

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

Multi-head aggregation:

$$\mathbf{X} = \mathrm{PHM}([\mathbf{H}_1; \ldots; \mathbf{H}_{N_h}])$$

Position-wise FFN:

$$\mathbf{Y} = \mathrm{PHM}(\mathrm{ReLU}(\mathrm{PHM}(\mathbf{X})))$$

PHM layers can reduce parameters and improve performance with flexible choices of n for LSTMs on NLI tasks

Table 1: Experimental results of natural language inference (accuracy) on five different datasets. The PHM-LSTM reduces the parameters of the standard LSTM model and improves or partially matches performance on four out of five datasets.

Model	#Params	MNLI	QNLI	SNLI	DNLI	SciTail
LSTM	721K	**71.82** / 71.89	84.44	84.18	85.16	74.36
Quaternion LSTM	180K (-75.0%)	71.57 / **72.19**	**84.73**	84.21	86.45	75.58
PHM-LSTM ($n=2$)	361K (-49.9%)	**71.82** / 72.08	84.39	84.38	85.77	77.47
PHM-LSTM ($n=5$)	146K (-79.7%)	71.80 / 71.77	83.87	**84.58**	**86.47**	74.64
PHM-LSTM ($n=10$)	81K (-88.7%)	71.59 / 71.59	84.25	84.40	86.21	**77.84**

PHM layers can reduce parameters and improve performance with flexible choices of n for transformers on NMT tasks

Table 2: Experimental results of machine translation (BLEU) on seven different datasets. Symbol † represents re-scaling the parameters with a factor of 2 by doubling the hidden size. The PHM-transformer does not lose much performance despite enjoying parameter savings. Re-scaling can lead to improvement in performance.

Model		#Params	En-Vi	En-Id	De-En	Ro-En	En-Et	En-Mk	En-Ro
Transformer (Tm)		44M	28.43	47.40	**36.68**	**34.60**	14.17	13.96	**22.79**
Quaternion Tm		11M (-75.0%)	28.00	42.22	32.83	30.53	13.10	13.67	18.50
PHM-Tm	$n = 2$	22M (-50.0%)	29.25	46.32	35.52	33.40	**14.98**	13.60	21.73
PHM-Tm	$n = 4$	11M (-75.0%)	29.13	44.13	35.53	32.74	14.11	13.01	21.19
PHM-Tm	$n = 8$	5.5M (-87.5%)	29.34	40.81	34.16	31.88	13.08	12.95	21.66
PHM-Tm	$n = 16$	2.9M (-93.4%)	29.04	33.48	33.89	31.53	12.15	11.97	19.63
PHM-Tm†	$n = 2$	44M	**29.54**	**49.05**	34.32	33.88	14.05	**14.41**	22.18
PHM-Tm†	$n = 4$	22M (-50.0%)	29.17	46.24	34.86	33.80	14.43	13.78	21.91
PHM-Tm†	$n = 8$	11M (-75.0%)	29.47	43.49	34.71	32.59	13.75	13.78	21.43

PHM layers do not increase much computational cost in practice

Table 3: Training time (seconds per 100 steps) and inference time (seconds to decode test sets) with beam size of 4 and length penalty of 0.6 on the IWSLT'14 German-English dataset.

Model	Transformer (Tm)	Quaternion Tm	PHM-Tm ($n = 4$)	PHM-Tm ($n = 8$)
Training time	**7.61**	8.11	7.92	7.70
Inference time	336	293	299	**282**

PHM layers can reduce parameters and improve performance with flexible choices of n for transformers on more tasks

Text style transfer

Model	#Params	BLEU
Transformer (Tm)	44M	11.65
PHM-Tm $(n = 2)$	22M (-50.0%)	12.20
PHM-Tm $(n = 4)$	11M (-75.0%)	**12.42**
PHM-Tm $(n = 8)$	5.5M (-87.5%)	11.66
PHM-Tm $(n = 16)$	2.9M (-93.4%)	10.76

Subject verb agreement

Model	#Params	Acc
Transformer (Tm)	400K	94.80
Quaternion Tm	100K	94.70
PHM-Tm $(n = 2)$	200K (-50.0%)	95.14
PHM-Tm $(n = 4)$	101K (-74.8%)	95.05
PHM-Tm $(n = 8)$	56K (-86.0%)	**95.62**

Summary

- Parameterized hypercomplex multiplication (PHM) layers learn multiplication rules from data using arbitrarily $1/n$ parameters compared with the fully-connected layer counterpart

- PHM layers can improve LSTMs and transformers on multiple NLP tasks