Reparameterized Multi-Resolution Convolutions for Long Sequence Modelling

Harry Jake Cunningham[1], Giorgio Giannone[1,2],

Mingtian Zhang[1], Marc Deisenroth[1]

University College London[1], *Amazon*[2]

NeurIPS 2024

Global convolution sequence models:

✔ **Effective general-purpose sequence models**

- *FlexConv, S4, S4D, SGConv, Hyena*

✔ **Efficient computation via FFTs**

- *FlashFFTConv*

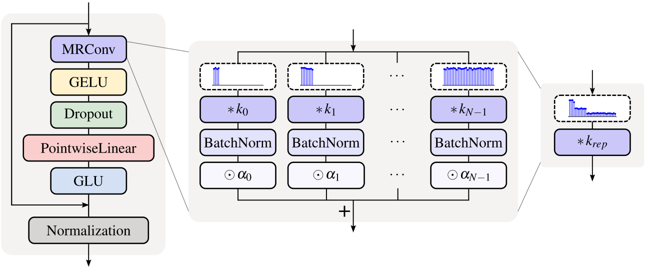
✗ **Difficult to train**

- *Explicitly parameterized kernels are prone to overfitting*
- *Implicit kernel parameterization, regularization, composition of sub kernels*

✗ **Hand-crafted inductive biases**

- *Fixed kernel decay*

MRConv: Multi-Resolution Convolutions



(a) Block (b) Training (c) Inference

1. **Multi-Resolution Convolutions**
   - *Introduces learnable kernel decay*
2. **Causal Structural Reparameterization**
   - *Improves training by introducing training-time non-linearity*
3. **Low-Rank Kernel Parameterizations**
   - *Explicitly parameterized kernels are prone to overfitting*

Multi-Resolution Convolutions

We define **multi-resolution convolutions** as the weighted sum of normalized convolutions of different length

$$y = \alpha_0 \mathrm{BN}_0(k_0 * u) + \alpha_1 \mathrm{BN}_1(k_1 * u) + \cdots + \alpha_{N-1} \mathrm{BN}_{N-1}(k_{N-1} * u) \tag{1}$$

- At each resolution the kernel k_i is of length $2^i l_0$
- Weighted sum of multi-resolution implicitly learns kernel decay
- BatchNorm required for learning weighted sum due to impact of kernel size on output statistics

Causal Structural Reparameterization

We can merge multiple causal convolutions into one as,

$$y = \underbrace{\sum_{n=0}^{N-1}(u * k_n)}_{\text{Sum of convolutions}} = \underbrace{\left(u * \left(\sum_{n=0}^{N-1} k_n\right)\right)}_{\text{Convolution of sum}} = (u * k_{rep}), \tag{2}$$

But what about BatchNorm?

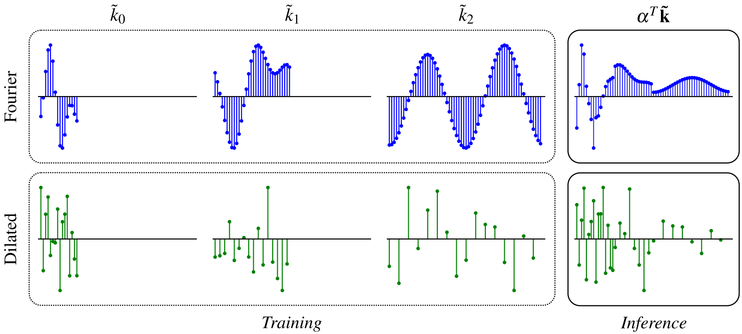
✗ Non-linear during training → Cannot Merge

$$y = \underbrace{\alpha_0 \mathrm{BN}_0(k_0 * u)}_{\text{Sum of convolutions}} + \underbrace{\alpha_1 \mathrm{BN}_1(k_1 * u)}_{\text{Sum of convolutions}} + \cdots + \underbrace{\alpha_{N-1} \mathrm{BN}_{N-1}(k_{N-1} * u)}_{\text{Sum of convolutions}} \tag{3}$$

✔ Linear during inference → Merge

$$y = u * \underbrace{(\alpha_0 \mathrm{BN}_0(k_0) + \alpha_1 \mathrm{BN}_1(k_1) + \cdots + \alpha_{N-1} \mathrm{BN}_{N-1}(k_{N-1}))}_{\text{Convolution of sum}} \tag{4}$$

Low-Rank Kernel Parameterizations



1. **Dilated Kernels** $y[t] = (u * k_{dilated})[t] = \sum_{\tau=0}^{l-1} k[\tau]u[t - p\tau]$

2. **Fourier Kernels** $k_{fourier}[t] = \text{IFFT}[\text{ZeroPad}(\hat{k}, L - m)])[t]$

3. **Sparse Kernels** $k_{sparse}[t] = \delta_{t \in \mathcal{T}} \cdot k_t$

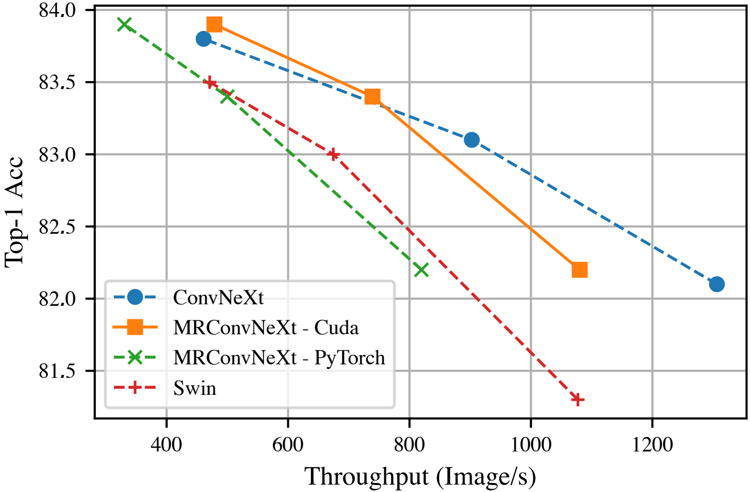
Experiments: Long Range Arena

MRConv is competitive with other sub-quadratic complexity models, including SSMs and linear-time transformers.

Model (Input length)	ListOps (2,048)	Text (4,096)	Retrieval (4,000)	Image (1,024)	Pathfinder (1,024)	Path-X (16,384)	Avg.
Transformer	36.37	64.27	57.46	42.44	71.40		53.66
Linear-Time Transformers:							
MEGA-Chunk	58.76	**90.19**	90.97	85.80	94.41	93.81	85.66
State Space Models:							
S4D-LegS	60.47	86.18	89.46	88.19	93.06	91.95	84.89
S4-LegS	59.60	86.82	90.90	88.65	94.20	96.35	86.09
Liquid-S4	**62.75**	89.02	91.20	89.50	94.8	96.66	87.32
S5	62.15	89.31	91.40	88.00	95.33	**98.58**	**87.46**
Convolutional Models:							
CCNN	43.60	84.08	-	88.90	91.51		-
Long Conv	62.2	89.6	91.3	87.0	93.2	96.0	86.6
SGConv	61.45	89.20	91.11	87.97	95.46	97.83	87.17
MRConv	62.40	89.26	**91.44**	**90.37**	**95.55**	97.82	**87.81**

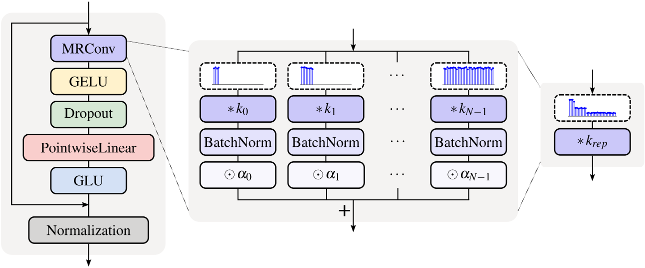
Experiments: ImageNet Classification

Using optimized CUDA kernels for 1D FFT convolutions, we close the gap between theoretical and empirical throughput.



Summary

Thank you for listening!



(a) Block (b) Training (c) Inference

More in our paper:

- More experiments
- More ablations
- More implementation details