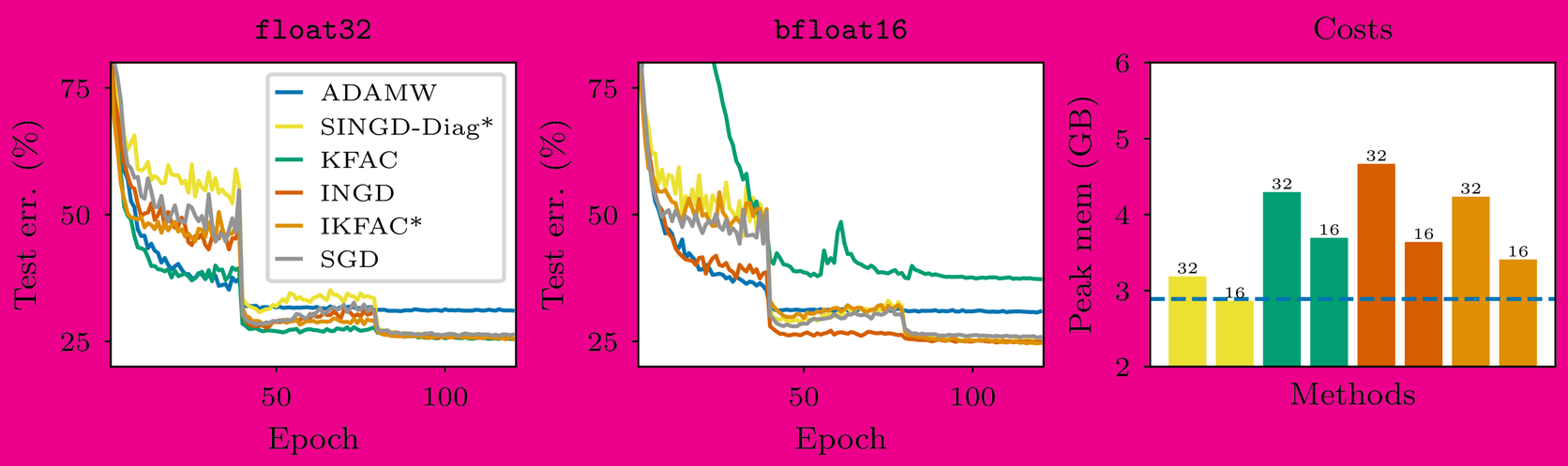
SINGD: Structured Inverse-Free Natural Gradient Descent

Wu Lin*, Felix Dangel*, Runa Eschenhagen, Kirill Neklyudov, Agustinus Kristiadi,
Richard Turner, Alireza Makhzani

UNIVERSITY OF TORONTO · UNIVERSITY OF CAMBRIDGE · VECTOR INSTITUTE



Paper

Code

We present a **KFAC-style 2nd-order method** for DL that is **memory-efficient & numerically-stable in `bfloat16`** thanks to structured Kronecker factors and an inverse-free update rule.

SINGD = Inverse-free KFAC + Structures

Goal: Approximate natural gradient $\mathbf{F}^{-1}\mathbf{g}$ with gradient $\mathbf{g}$, inverse Fisher approximation $\mathbf{F}^{-1}$, and damping λ.

$$\mathbf{F}^{-1}_{(\text{KFAC})} = (\mathbf{S}_\mathbf{K} + \lambda\mathbf{I})^{-1} \otimes (\mathbf{S}_\mathbf{C} + \lambda\mathbf{I})^{-1}$$

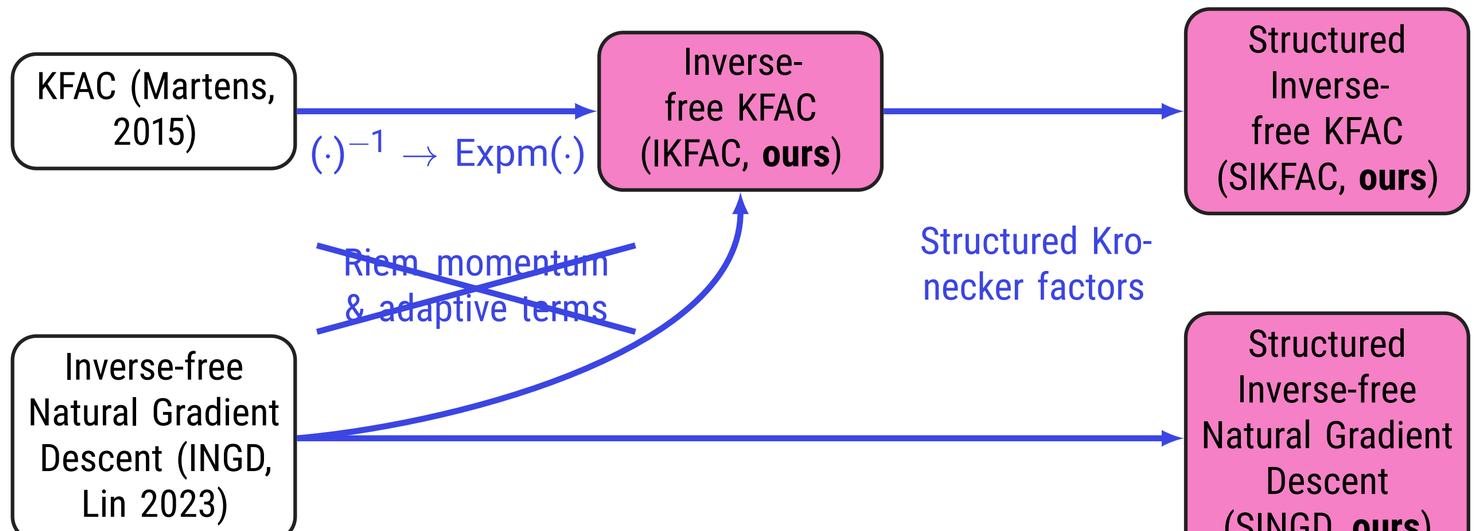
vs.

$$\mathbf{F}^{-1}_{(\text{INGD})} = \mathbf{K}\mathbf{K}^\top \otimes \mathbf{C}\mathbf{C}^\top$$

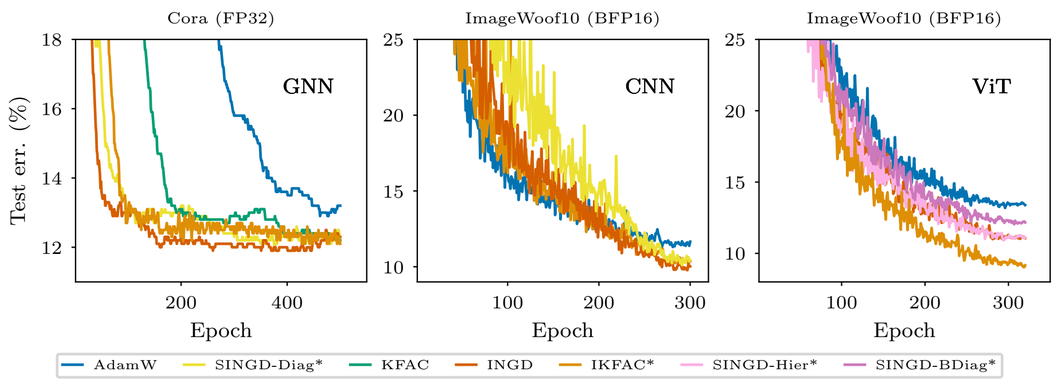
vs.

$$\mathbf{F}^{-1}_{(\text{SINGD})} = \mathbf{K}\mathbf{K}^\top \otimes \mathbf{C}\mathbf{C}^\top,$$

with structured $\mathbf{K}, \mathbf{C}$.



Performs Well on Many Problems & in `bfloat16`



Works like any Optimizer (`pip install singd`)

```python
from singd.optim.optimizer import SINGD
from torch_utils import CrossEntropyLoss, alexnet, new_batch

# 1. Setup
model, loss_func = alexnet(), CrossEntropyLoss()
X, y = new_batch()

optim = SINGD(  # Optimizer setup
    model,
    kfac_like=False,  # Use True for IKFAC
    structures=("dense", "diagonal"),  # Structure of (K, C)
)

# 2. One training step
optim.zero_grad()  # Clear previous gradients
loss_func(model(X), y).backward()  # Forward + backward pass
optim.step()  # Update pre-conditioner and parameters
```

Available Structures

Different structures capture different information.