FLASHLIGHT: PyTorch Compiler Extensions to Accelerate Attention Variants

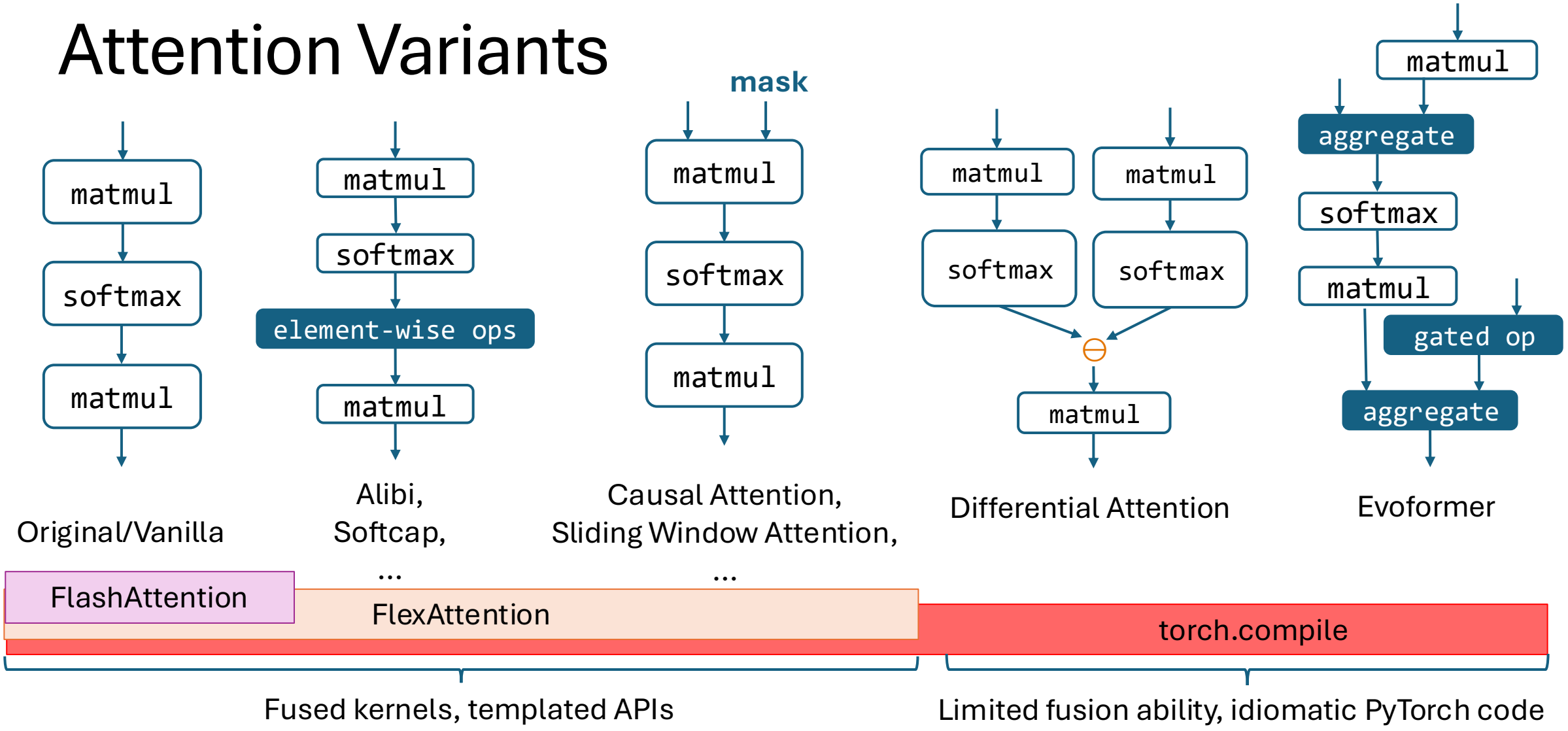
Bozhi You, Irene Wang, Zelal Su Mustafaoglu,

Abhinav Jangda, Angélica Moreira, Roshan Dathathri,

Divya Mahajan, Keshav Pingali

https://github.com/bozhiyou/flashlight

Attention Variants



FLASHLIGHT generates optimized, fused kernels from idiomatic PyTorch code

FLASHLIGHT in a Nutshell

General extensions to PyTorch compiler (torch.compile)

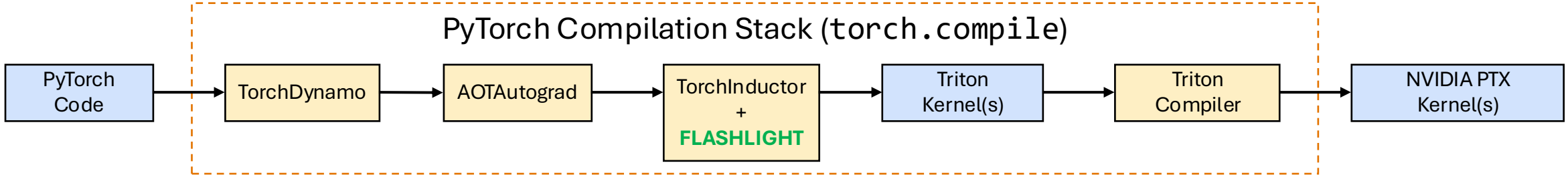
- Generates I/O-efficient Triton kernels from *idiomatic* PyTorch code
  - Even for variants not supported by FlexAttention or FlashAttention
- Moves the optimization burden from the *user* to the *compiler*

Superior performance

- Always faster than default PyTorch Compiler
- For FlexAttention-supported variants: similar performance
- Beyond FlexAttention: **5x** faster for Evoformer
  - Improves end-to-end inference latency for AlphaFold by 6 to 9%

FLASHLIGHT: An Augmented Torch

General extensions to TorchInductor, *not* a standalone compiler

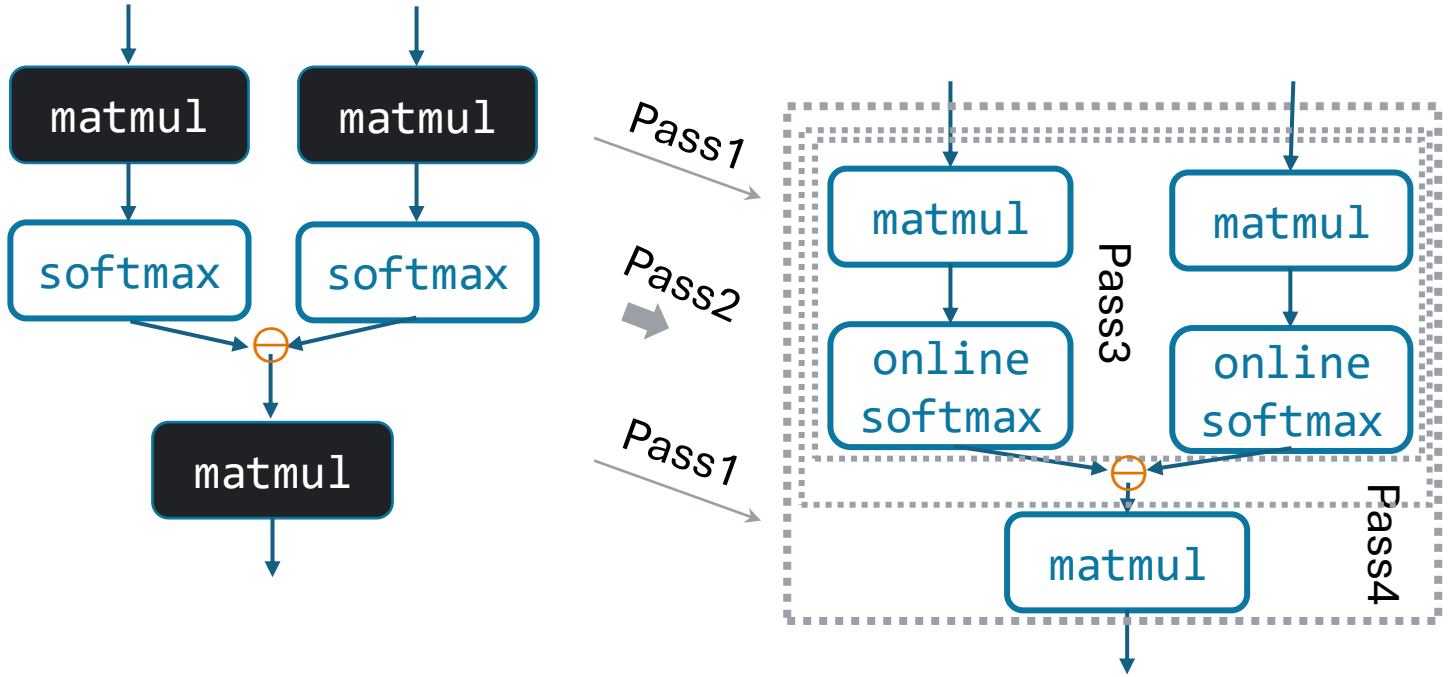


1. Unified Reduction IR

2. Algebraic Transformation

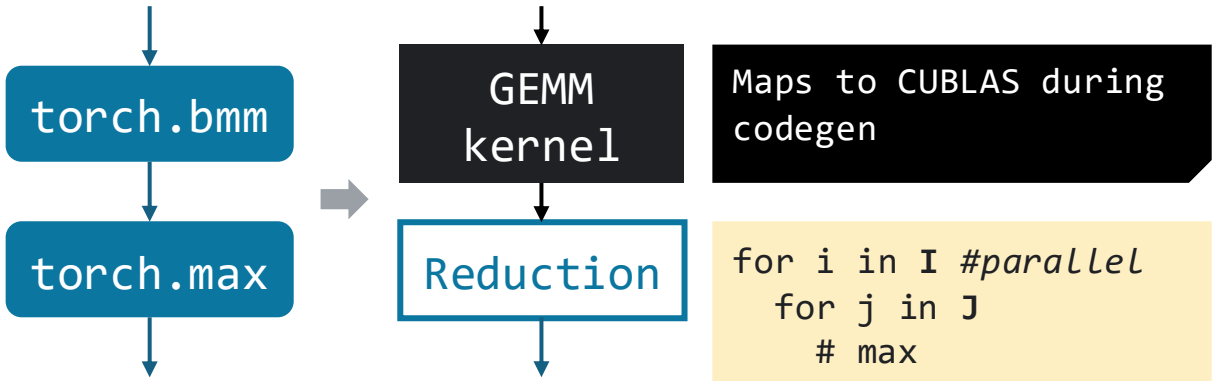
3. Dimension Demotion

4. Tiling-Aware Dimension Elimination



FLASHLIGHT (1/4): Unified Reduction IR

Before/Default

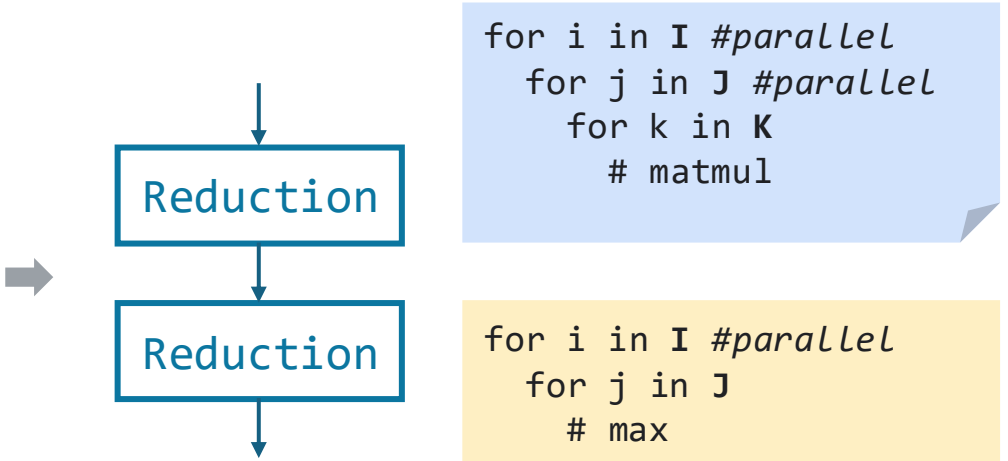


Compute Graph

Inductor IR

Optimized GEMM kernel but artificial fusion boundary

With FLASHLIGHT



Inductor IR

Enables fusing matmul with reductions (sacrifices CUBLAS performance for fusion opportunities)

FLASHLIGHT (2/4): Algebraic Transformation

Before/Default

$$\text{softmax}(x) = \frac{e^{x-\max x}}{\sum e^{x-\max x}}$$

```
m = -inf
for j in J
  m = max(m, x[p1])
```

```
s = 0
for j in J
  s = s + exp(x[p1] - m)
```

Second loop/pass depends on first => fusion boundary

With FLASHLIGHT

```
m = -inf
s = 0
for j in J
  m_ = max(m, x[p1])
  s = s * exp(m - m_)
  s = s + exp(x[p1] - m_)
  m = m_
```

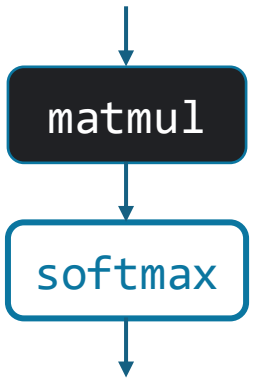
Homomorphism: $h(f(a,b)) = \mathrm{g}(h(a), h(b))$

$\exp(\text{sub}(a,b)) = \text{div}(\exp(a), \exp(b))$: $e^{a-b} = \frac{e^a}{e^b}$

Single loop/scan => enables fusion

FLASHLIGHT (3/4): Dimension Demotion

Any "parallel" dim can be treated as a "reduction" dim
=> demote fusable "parallel" dim to "reduction" dim
(sacrifices parallelism for I/O efficiency)



Pass 1

```
for i in I #parallel
  for j in J #parallel
    for k in K
      # matmul
```

Pass 2

```
for i in I #parallel
  for j in J #reduction
    # softmax
```

Pass 3

```
for i in I #parallel
  for j in J #reduction
    for k in K
      # matmul
```

```
for i in I #parallel
  for j in J #reduction
    # softmax
```

```
for i in I #parallel
  for j in J
    for k in K
      # matmul
    # softmax
```

Single fused kernel for:

$$A = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

FLASHLIGHT (4/4): Tiling-Aware Dimension Elimination

When H (head dim) is small,
set tile_size_of_H = H so that |TH| = 1 to drop TH loop
(requires logical grid dimensions)

```
for i in I #parallel
  for j in J
    for k in K
      # matmul
    # softmax
```

```
for i in I #parallel
  for h in H
    for j in J
      # matmul
```

```
for ti in TI #parallel
  for tj in TJ
    for tk in TK
      # tile of matmul
    # tile of softmax
```

```
for ti in TI #parallel
  for th in TH
    for tj in TJ
      # tile of matmul
```

```
for ti in TI #parallel
  for tj in TJ
    for tk in TK
      # tile of matmul
    # tile of softmax
```

```
for ti in TI #parallel
  # |TH| = 1
  for tj in TJ
    # tile of matmul
```

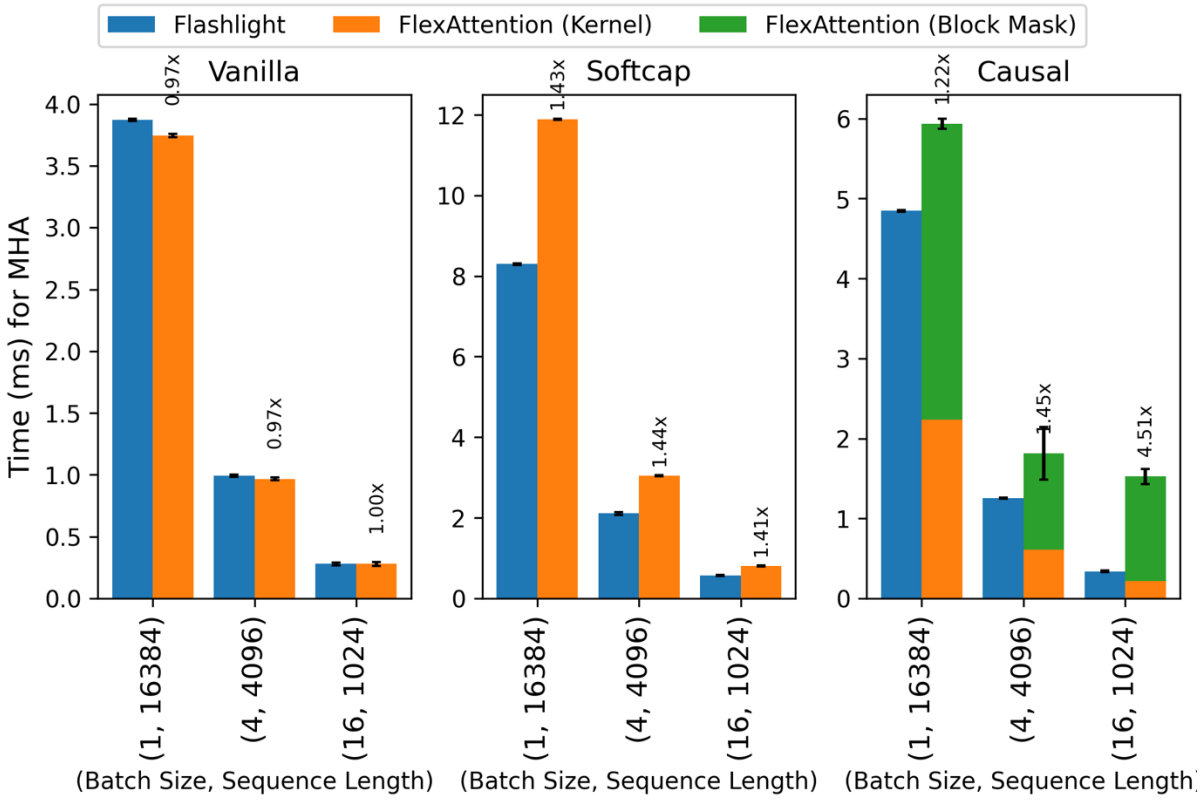
```
for ti in TI #parallel
  for tj in TJ
    for tk in TK
      # tile of matmul
    # tile of softmax
    # tile of matmul
```

Single fused kernel for:

$$A = \mathbf{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)\mathbf{V}$$

Evaluation: Flex-Supported Attention Variants on H100

How do FLASHLIGHT *generated* kernels compare to FlexAttention's *specialized* template?

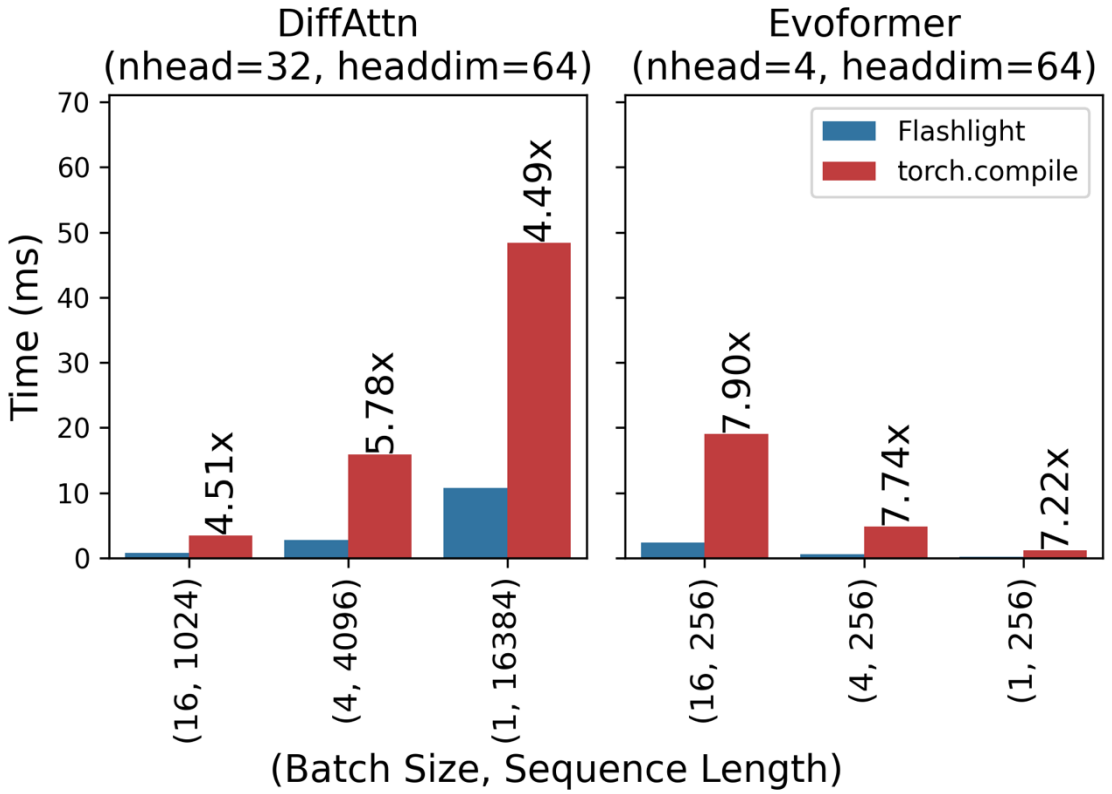


Refer to the paper for more results

FLASHLIGHT is competitive or faster than FlexAttention

Evaluation: Complex Attention Variants on H100

What about the complex attention variants that FlexAttention cannot handle?



Refer to the paper for more results

FLASHLIGHT is significantly faster than default torch.compile

FLASHLIGHT: Summary

General extensions to PyTorch compiler (torch.compile)

- Generates I/O-efficient Triton kernels from *idiomatic* PyTorch code
  - Even for variants not supported by FlexAttention or FlashAttention
- Moves the optimization burden from the *user* to the *compiler*

Superior performance

- Always faster than default PyTorch Compiler
- For FlexAttention-supported variants: similar performance
- Beyond FlexAttention: **5x** faster for Evoformer
  - Improves end-to-end inference latency for AlphaFold by 6 to 9%

https://github.com/bozhiyou/flashlight

References

Tillet, P., Kung, H. T., and Cox, D. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wu, M., Cheng, X., Liu, S., Shi, C., Ji, J., Ao, M. K., Velliengiri, P., Miao, X., Padon, O., and Jia, Z. Mirage: A {Multi-Level} superoptimizer for tensor programs. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pp. 21–38, 2025.

Ye, T., Dong, L., Xia, Y., Sun, Y., Zhu, Y., Huang, G., and Wei, F. Differential transformer, 2024. URL https://arxiv.org/abs/2410.05258.

Ye, Z., Chen, L., Lai, R., Lin, W., Zhang, Y., Wang, S., Chen, T., Kasikci, B., Grover, V., Krishnamurthy, A., and Ceze, L. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025. URL https://arxiv.org/abs/2501.01005.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Dong, J., Feng, B., Guessous, D., Liang, Y., and He, H. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

He, H., Guessous, D., Liang, Y., and Dong, J. Flexattention: The flexibility of pytorch with the performance of flashattention, Aug 2024. URL https://pytorch.org/blog/flexattention/.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde Farley, and Yoshua Bengio. Theano: A CPU and GPU Math Compiler in Python. In Stéfan van der Walt and Jarrod Millman (eds.), *Proceedings of the 9th Python in Science Conference*, pp. 18 – 24, 2010. doi: 10.25080/Majora-92bf1922-003.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Milakov, M. and Gimelshein, N. Online normalizer calculation for softmax, 2018. URL https://arxiv.org/abs/1805.02867.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation, 2022. URL https://arxiv.org/abs/2108.12409.

Reed, J., DeVito, Z., He, H., Ussery, A., and Ansel, J. torch.fx: Practical program capture and transformation for deep learning in python. In Marculescu, D., Chi, Y., and Wu, C. (eds.), *Proceedings of Machine Learning and Systems*, volume 4, pp. 638–651, 2022. URL https://proceedings.mlsys.org/paper_files/paper/2022/file/7c98f9c7ab2df90911da23f9ce72ed6e-Paper.pdf.

Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C. K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., and Chintala, S. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, pp. 929–947, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703850. doi: 10.1145/3620665.3640366. URL https://doi.org/10.1145/3620665.3640366.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 578–594, 2018.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pp. 265–283, USA, 2016. USENIX Association. ISBN 9781931971331.

Ahdritz, G., Bouatta, N., Floristean, C., Kadyan, S., Xia, Q., Gerecke, W., O'Donnell, T. J., Berenberg, D., Fisk, I., Zanichelli, N., Zhang, B., Nowaczynski, A., Wang, B., Stepniewska-Dziubinska, M. M., Zhang, S., Ojewole, A., Guney, M. E., Biderman, S., Watkins, A. M., Ra, S., Lorenzo, P. R., Nivon, L., Weitzner, B., Ban, Y.-E. A., Sorger, P. K., Mostaque, E., Zhang, Z., Bonneau, R., and AlQuraishi, M. OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization. *bioRxiv*, 2022. doi: 10.1101/2022.11.20.517210. URL https://www.biorxiv.org/content/10.1101/2022.11.20.517210.