Automated Neural Network Acceleration on Edge:
from Convolutional Networks to Transformers

Di Niu

Professor
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Canada
Huawei Edmonton Research Center

Research Areas:
AI： Deep learning acceleration, AutoML, Computer Vision, NLP
Distributed Systems: Federated Learning, Edge and Cloud computing, Parallel Computing Systems
Data Mining: graph mining and reasoning, GNNs

PhD students （11）
Qikai Lu, Keith Mills, Adel Ameri, Jerry Chen, Shengyao Lu, Jiuding Yang, Yakun Yu, Liyao Jiang, Linxuan Zhang, Ruichen Chen, Amirhosein Ghasemabadi

MSc students （5）
Ruiqing Tian, Mohammadali Shakerdargah, Hongxuan Liu, Juxin Fa, Mohammadamin Khoshko

DNNs are widely used in AI Applications on various Edge Devices

Semantic Segmentation
Photo Credit: Ambarella AI Vision Processor for edge applications

Face Recognition for Unlocking/Payment

Gesture Control

Assistant with Bard

Magic Editor

Video Boost

Best Take

DNN is most widely used on edge devices, including Phones, Cameras, Cars, etc.

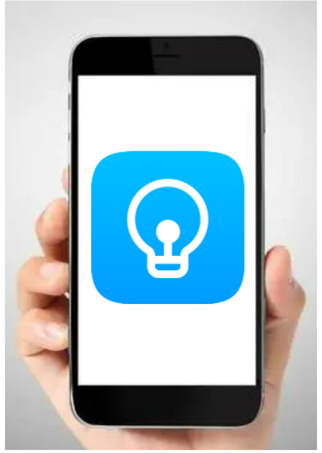
Generative AI has changed Cellphone Usage

SmartPhone -> Mobile Internet | **Generative AI -> a New Era**

Smart Phone → AI Phone

- 5hours per day
- Content acquisition
  - Short video/Long video
  - News / Search
  - Listening to music
- Shoppers
- Social Networking
- Games

- >5hours per day
- Understanding:
  - Dealing with unstructured data
  - Diversified interaction modes
  - Natural language instead of programming language
- Generation:
  - Expert level beyond ordinary humans
- Reasoning: self-learning



京东

Bard

文心一言

通义千问
阿里巴巴

Stable Diffusion

Midjourney

Tencent 腾讯

文心一格
AI艺术和创意辅助平台

Search → Push → GenAI → ?

Example: SnapDragon 8 Gen3 Improve AI Performance by 98%

SnapDragon8Gen3

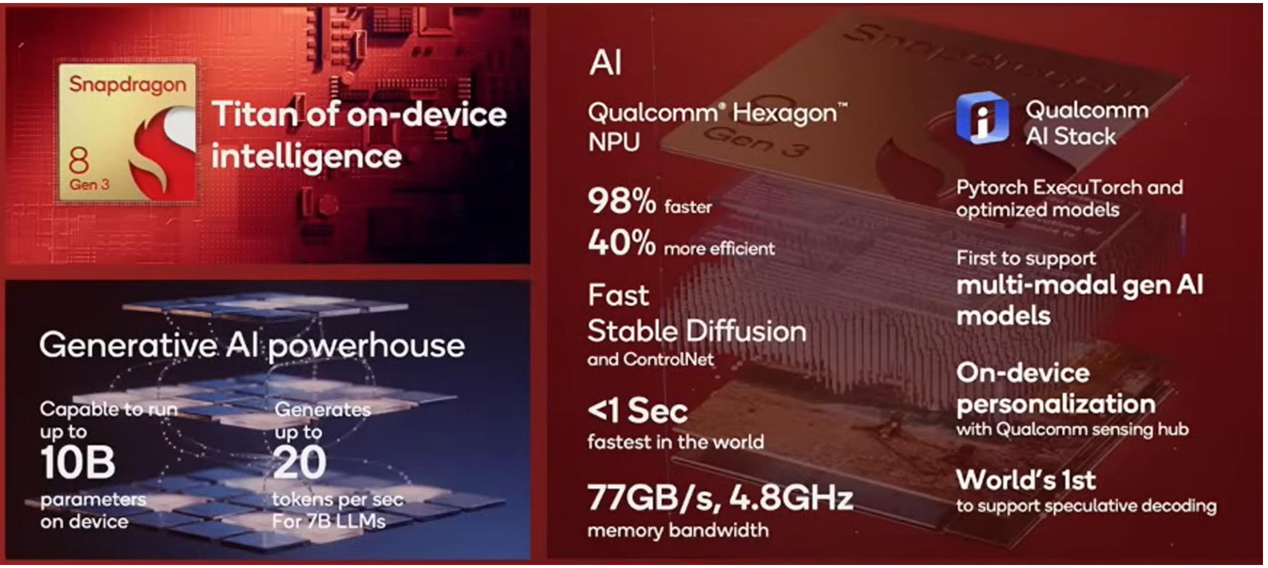
AI Stack supports more than 20 different models and various AI frameworks such as Pytorch

- Process Technology: TSMC 4nm
- CPU: Kryo
- CPU: 8 cores1+3+2+2, 3.3 GHz
- GPU: Ardeno 750
- NPU: Hexagon@45 TOPS
- AI Capabilities:
    - Micro-tile inference
    - CPU+NPU Heterogeneous for LLAMA2
    - Can run up to 10B LLM
    - 20 tokens/s for 7B LLM
    - <1 second Stable Diffusion
    - Outpainting

SnapDragon XElite for PC

Expand generative AI application scenarios and apply edge AI technologies to PC

- Process Technology: TSMC 4nm
- CPU: Oryon
- CPU: 12 cores, 3.8-4.3 GHz
- GPU: Ardeno for X Elite
- NPU: Hexagon@45 TOPS
- AI Capability: 75 TOPS
    - Micro-tile inference
    - CPU+NPU Heterogeneous for LLAMA2
    - Can run up to 13B LLM
    - 30 tokens/s for 7B LLM
    - <1 second Stable Diffusion

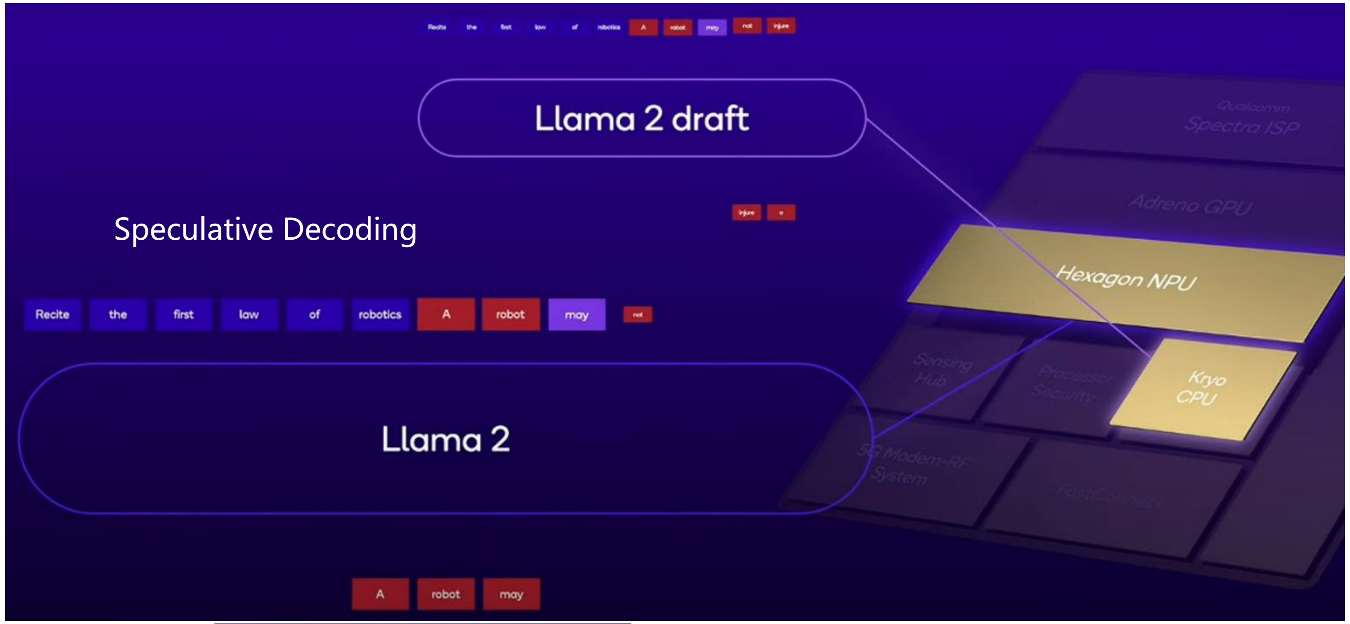


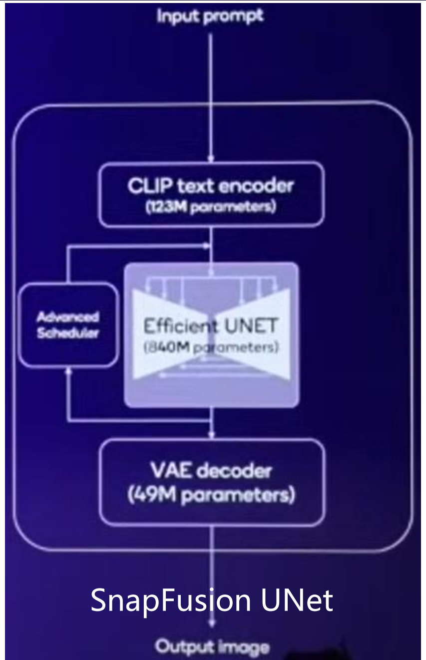
Example: SnapDragon 8 Gen 3 Key Technologies for Edge AI

Snapdragon 8 Gen3 AI Capabilities

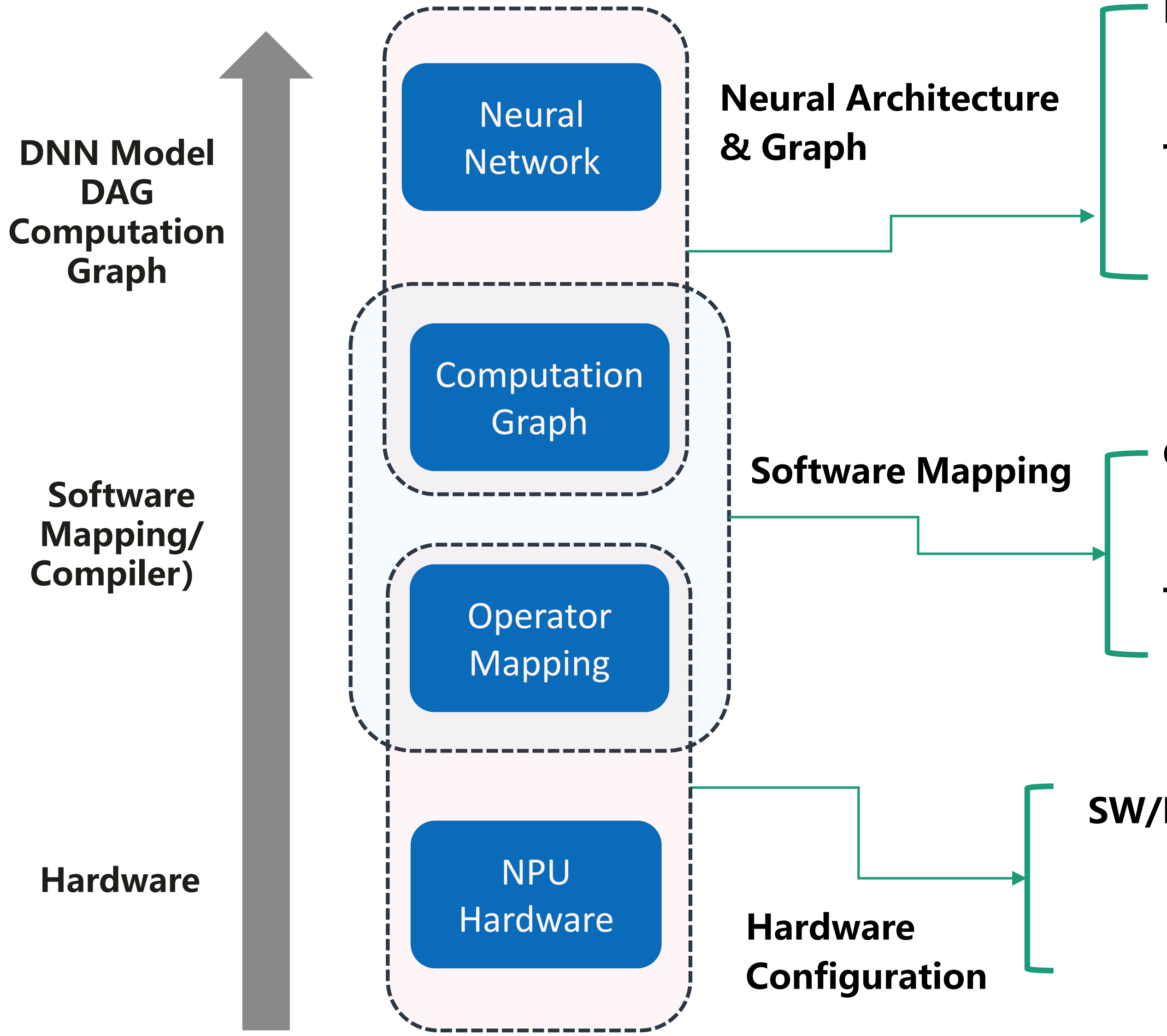
- Run LLAMA2 and Baichuan model on the device and implement LLM voice assistant function on device
- Run Stable Diffusion Text-to-Image model on the device and generate a 512x512 picture in 0.57s.
- Stable Diffusion Outpainting on the device, 8-steps, ~8s

	8Gen3
SD	Text to image 0.57s
LLM	20 tokens/s LLAMA2
LLM quantization	INT4 Weights, INT8 Act
SD quantization	INT4 Weights, INT8 Act
Software mapping	Micro-tile inference
NAS for SD	SnapFusion: NAS+Robust Training for Unet，NAS for VAE decoder
SD Step Distillation	Progressive Step Distillation
Heterogeneous Computing	**Speculative Decoding**，CPU run small model to predict k tokens，NPU run big model for verification, enhance token rate by 2x





Our Contributions to Full-Stack DNN Acceleration 2020-2023



Neural Architecture and Graph Optimizer (AutoGO)

Automates Graph mutation with frequent subgraphs for lightweight deployment

Training optimization

Increase training performance without additional inference cost

Graph Fusion

Fusing ops/layers

Depth First execution

Tensor Optimization

Tiling, loop ordering, resource binding

SW/HW co-optimization method for accelerator design (UNICO)

Efficient SW/HW co-search algorithm

Generalize HW configurations to multiple DNN workloads

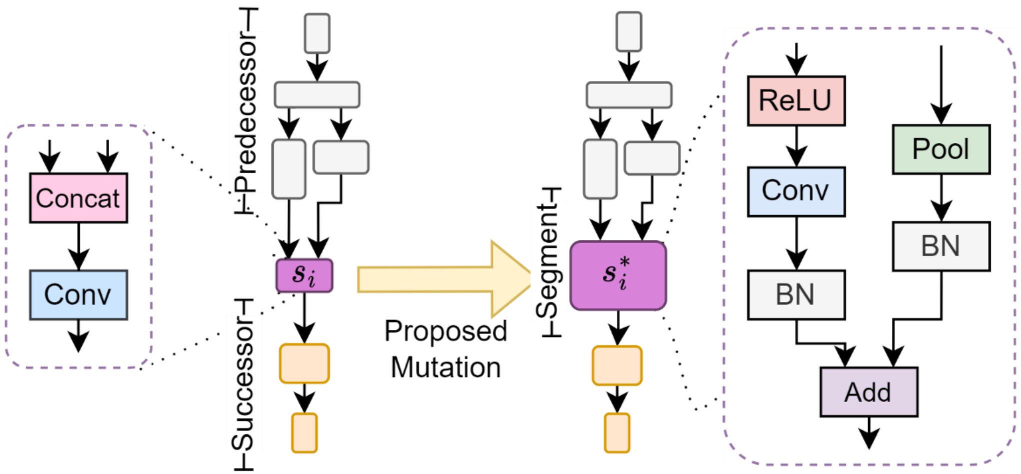
Computation Graph Optimization

AutoGO: Automated Computation Graph Optimization for Neural Network Evolution

Mohammad Salameh[1]*, Keith G. Mills[1,2]*†, Negar Hassanpour[1], Fred X. Han[1],
Shuting Zhang[3], Wei Lu[1], Shangling Jui[3], Chunhua Zhou[3], Fengyu Sun[3], Di Niu[2]
[1]Huawei Technologies Canada. [2]Dept. ECE, University of Alberta. [3]Huawei Kirin Solution, China.

NeurIPS 2023

- **Replaces manual optimization efforts by ML engineers for deploying ML on (edge) devices** for hardware friendliness and higher task accuracy
  - NOT by searching in a large design space
  - But given a **computation graph** of an existing NN, mutating it by subgraph replacement.
- Maintain a database of "well performing" subgraphs and segments (prior knowledge management)
- Leverages a pretrained neural architecture capacity predictor for different tasks
- Can explore 1000 mutated architectures in 15 minutes



Our Recent Publications on Computation Graph Optimization for Neural Architecture enhancement/compression:

- Mohammad Salameh, Keith G. Mills, Negar Hassanpour, Fred X. Han, Shuting Zhang, Wei Lu, Shangling Jui, Chunhua Zhou, Fengyu Sun, Di Niu. "**AutoGO: Automated Computation Graph Optimization for Neural Network Evolution**," in Proceedings of **NeurIPS 2023**.
- Keith G. Mills, Di Niu, Mohammad Salameh, Weichen Qiu, Fred X. Han, Puyuan Liu, Jialin Zhang, Wei Lu, Shangling Jui. "**AIO-P: Expanding Neural Performance Predictors Beyond Image Classification**," in Proceedings of **AAAI 2023**.
- Keith G. Mills, Fred X. Han, Jialin Zhang, Fabian Chudak, Ali Safari, Mohammad Salameh, Wei Lu, Shangling Jui, Di Niu. "**GENNAPE: Towards Generalized Neural Architecture Performance Estimators**," in Proceedings of **AAAI 2023**.

AutoGO: Automated Computation Graph Optimization

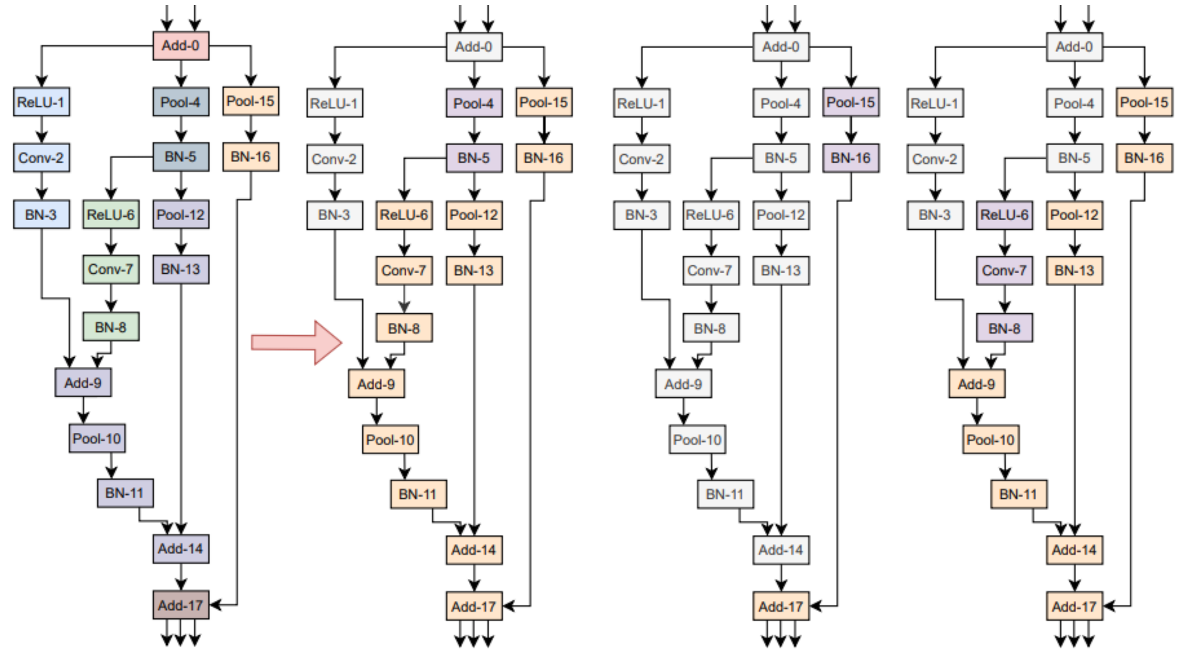
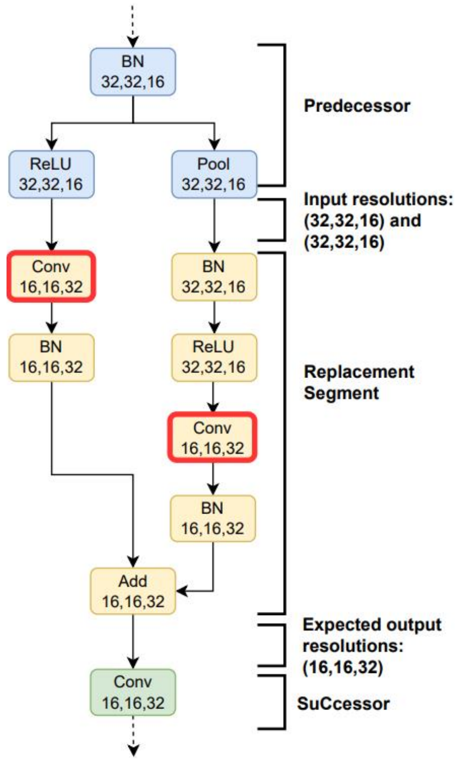


Figure 7: Example of how the BPE-segmented graph in Figure 6(b) is partitioned into Predecessor, Segment and suCcessor subgraphs based on the selected segment. Specifically, we highlight nodes of the selected segment in purple, the predecessor in grey and the successor in yellow.



Key Innovations

Build a database of frequent subgraphs using a NLP tokenization technique (BPE) based on several architectural families of >400k+ architectures including
- NB101, NB201, Inception, Two-path, HiAML benchmarks, etc.

Pretrain a P-S-C predictor to estimate the reward of each segment mutation

Use Transfer Learning (**AIO-P** and **GENNAPE in AAAI 2023**) to transfer accuracy predictor to different CV tasks

CG Optimizer:
- Segment Mutation
- Channel Resolution Propagation
- Topology Optimizer

(a) Convert a model to ONNX format (Computation Graph)

(b) Partition CG into Predecessor (P), Segment (S), suCcessor (C)

(c) Mutate the graph with a segment from a database and perform resolution propagation

(d) Evaluate Task Accuracy with a pretrained P-S-C predictor

AIO-P: Pretraining NN Performance Predictor for Various CV Tasks

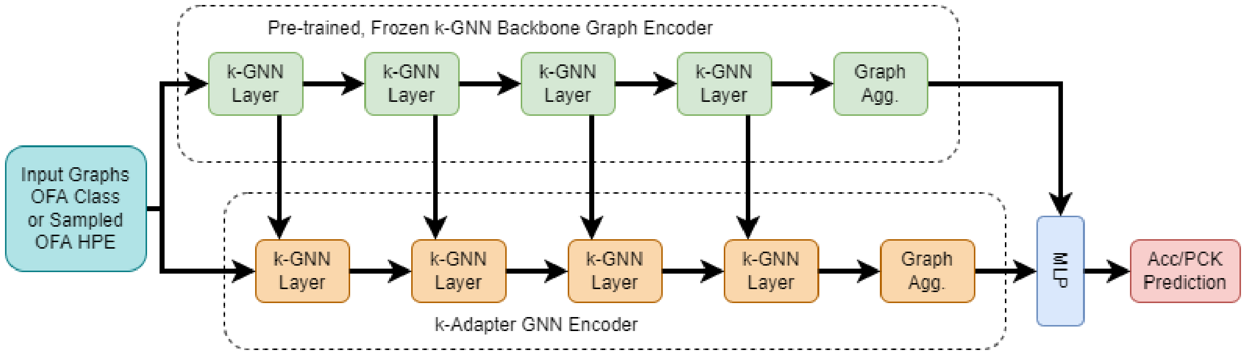
- **Current Issues**: Predictors in NAS are task (ImageClassification), dataset (CIFAR10) and metric (accuracy) dependent
- **Goal**: provide a generalizable predictor that
  - learns representations of graph structures of a neural network
  - transfer the learning among different CV tasks, datasets and metric
- **Challenges**: scarcity in NAS benchmark datasets for Dense CV Prediction tasks
  - Human Pose Estimation, SuperResolution, Image Segmentation, …

Generalizable to different CV tasks with k-Adapter

- First, pretrain a predictor on Image Classification architectures and labels
- We infuse the pretrained predictor with knowledge from a new task with k-adapters layers
- For example, we infuse PCK of models for Human Pose estimation to enable the predictor to handle new HPE architectures.

Performance prediction for ProxylessNAS architectures for Panoptic Segmentation on MS-COCO.

Scheme	Zero-Shot Inference	With Fine-tuning (20 samples)
Baseline *k*-GNN Predictor	MAE: 56.19% SRCC: 0.562	MAE: 0.53% SRCC: 0.741
+ Label Scaling	MAE: 0.76% SRCC: 0.119	MAE: 0.62% SRCC: 0.297
+ Double k-Adapter (Pose Estimation & Obj. Detection)	MAE: 0.50% SRCC: 0.732	MAE: 0.33% SRCC: 0.868



Keith G. Mills, Di Niu, Mohammad Salameh, Weichen Qiu, Fred X. Han, Puyuan Liu, Jialin Zhang, Wei Lu, Shangling Jui. “**AIO-P: Expanding Neural Performance Predictors Beyond Image Classification**,” in Proceedings of **AAAI 2023**.

AutoGO Experiment Results on ImageNet

AutoGO results on popular DNNs including ResNet-50, ResNet-101, VCG-16:

- ✓ Can improve ImageNet Top-1 accuracy by 1%, without using new operations or increasing FLOPs
- ✓ Can optimize network performance when it is used as backbone for semantic segmentation and Human Pose Estimation
- ✓ Allow customized optimization objectives (accuracy, FLOPs, latency, power)

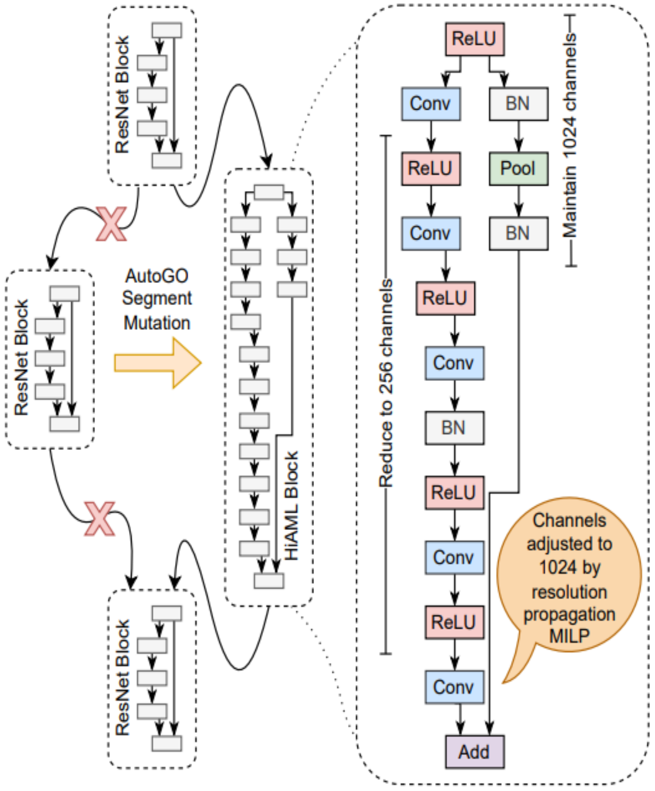


Figure 5: Example of a segment mutation that helped create ResNet-50 AutoGO Arch 2 from Table 3. A ResNet residual block is replaced by a HiAML block.

Improve performance with latency reduction(GPU) on Classification, Segmentation, Human Pose Estimation

Architecture	ImageNet Top-1/5	Cityscapes mIoU	MPII PCK	FLOPs [1e9]	Lat. [ms]
ResNet-50 Original	74.02%/91.22%	63.42%	82.36%	6.29	7.18
ResNet-50 AutoGO Arch 1	75.34%/92.16%	65.88%	**84.07%**	6.71	7.50
ResNet-50 AutoGO Arch 2	**75.66%/92.45%**	**66.65%**	82.70%	5.88	6.92
ResNet-101 Original	75.09%/91.94%	65.92%	82.77%	13.76	15.86
ResNet-101 AutoGO Arch 1	**76.56%/93.09%**	**67.12%**	83.59%	13.66	15.56
ResNet-101 AutoGO Arch 2	75.69%/92.15%	66.38%	**84.64%**	13.35	15.36
VGG-16 Original	74.18%/91.83%	65.36%	85.92%	30.81	4.65
VGG-16 AutoGO	**74.91%/93.23%**	**66.91%**	**85.99%**	24.34	4.20

Mohammad Salameh, Keith G. Mills, Negar Hassanpour, Fred X. Han, Shuting Zhang, Wei Lu, Shangling Jui, Chunhua Zhou, Fengyu Sun, Di Niu. "**AutoGO: Automated Computation Graph Optimization for Neural Network Evolution**," in Proceedings of **NeurIPS 2023.**

AutoGO Results on Other Tasks (e.g., Super Resolution)

Table 4: Peak Signal-to-Noise Ratio (PSNR) for EDSR on the DIV2K validation set and several SR benchmarks in the 2x upscaling setting. Higher is better. We measure latency on an RTX 2080 Ti.

SR Architecture	DIV2K	Set5	Set14	BSD100	Urban100	Manga109	FLOPs [1e9]	Lat. [ms]
EDSR Original	36.19	36.86	32.57	31.39	29.14	36.09	141	18.04
EDSR AutoGO Arch 1	**37.28**	**38.01**	**33.62**	**32.18**	**31.56**	**38.49**	118	15.38
EDSR AutoGO Arch 2	37.27	37.97	33.55	32.16	31.53	38.47	110	14.52
EDSR AutoGO Arch 3	37.25	**38.01**	33.58	32.16	31.46	38.44	105	13.81

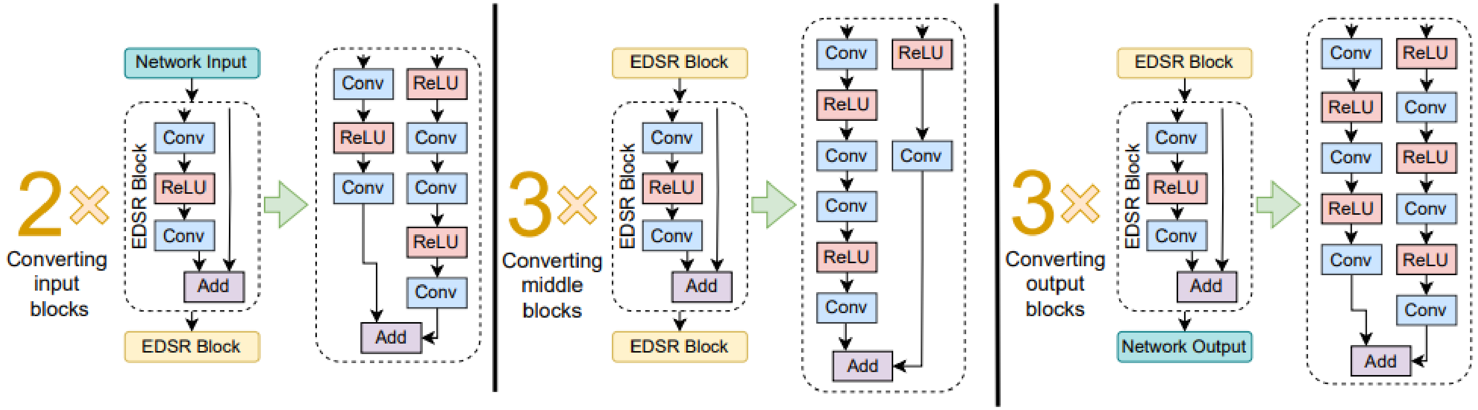


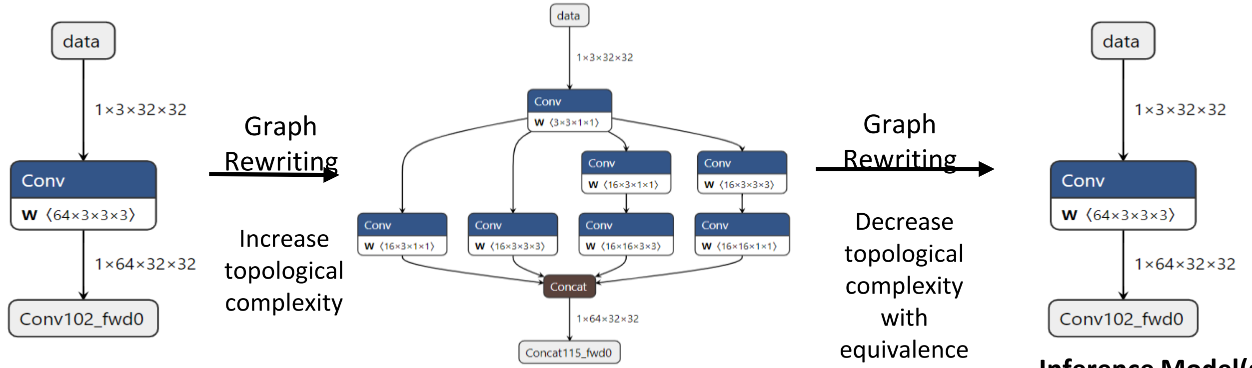
Figure 9: Example mutations performed by AutoGO to create EDSR Arch 2 in Table 4 by swapping out 8 EDSR blocks. Specifically, AutoGO will swap out multiple, simple 'Conv-ReLU-Conv' residual blocks for larger blocks that have operations on both branches.

Mohammad Salameh, Keith G. Mills, Negar Hassanpour, Fred X. Han, Shuting Zhang, Wei Lu, Shangling Jui, Chunhua Zhou, Fengyu Sun, Di Niu. "**AutoGO: Automated Computation Graph Optimization for Neural Network Evolution**," in Proceedings of **NeurIPS 2023.**

Spatial Gradient Scaling: Deep Learning Optimization

- **Reparameterization**: replacing a subgraph with a computationally equivalent subgraph by algebraic manipulation of the weights.
- **Goal:**
  - increase accuracy
  - help the model to learn and generalize better

We discovered an equivalence between Branched Reparametrization and Spatial Gradient Scaling



Original Model (simple)

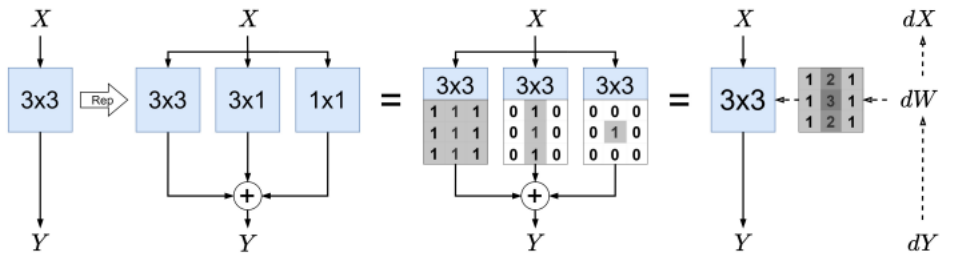
Training Model (complex)
more complex to train but has "higher capacity" than starting model.

Inference Model(simple)

Same inference result but faster

Model	Rep method	Cost (GPU days)	Avg. FLOPs (G)	Avg. params (M)	Acc (%)
ResNet-18	Origin	4.8	1.81	11.7	$71.13_{\pm 0.04}$
	DBB*	8.1	4.13	26.3	70.99
	DyRep*	6.3	2.42	16.9	71.58
	DyRep	9.1	2.92	22.1	$71.50_{\pm 0.03}$
	SGS (ours)	5.1	1.81	11.7	$71.65_{\pm 0.05}$
ResNet-34	Origin	5.3	3.66	21.8	$74.17_{\pm 0.05}$
	DBB*	12.8	8.44	49.9	74.33
	DyRep*	7.7	4.72	33.1	74.68
	DyRep	10.6	4.95	38.3	$74.40_{\pm 0.03}$
	SGS (ours)	5.8	3.66	21.8	$74.62_{\pm 0.05}$
ResNet-50	Origin	7.5	4.09	25.6	$76.95_{\pm 0.05}$
	DBB*	13.7	6.79	40.7	76.71
	DyRep*	8.5	5.05	31.5	77.08
	DyRep	11.0	5.84	38.3	$77.11_{\pm 0.03}$
	SGS (ours)	7.9	4.09	25.6	$77.10_{\pm 0.01}$

Table 2: Results on ImageNet dataset. We use the official implementation of DyRep (Huang et al., 2022) on 8 NVIDIA Tesla V100 GPUs. FLOPs and Parameters are averaged across DyRep runs. Results marked with * are taken from DyRep paper; the rest are our runs averaged over 3 seeds.



(a) Base (b) Structural Rep (c) Mathematical Rep (d) Spatial Gradient Scaling

Original Training

$$K'_{11} \Leftarrow K'_{11} - \alpha \frac{\partial L}{\partial K_{11}}$$

$$K'_{22} \Leftarrow K'_{22} - \alpha \frac{\partial L}{\partial K_{22}}$$

Spatial Gradient Scaling

$$K'_{11} \Leftarrow K'_{11} - \alpha \frac{\partial L}{\partial K_{11}}$$

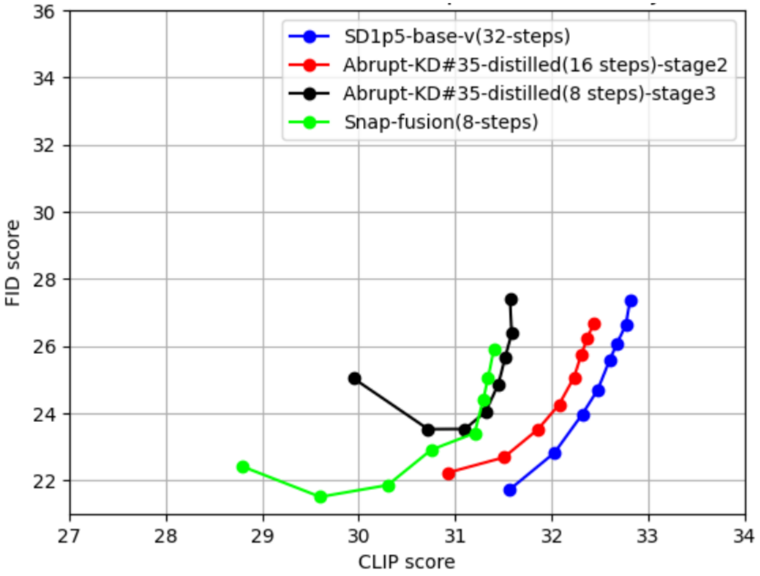
$$K'_{22} \Leftarrow K'_{22} - 2\alpha \frac{\partial L}{\partial K_{22}}$$

Alexander Detkov, Mohammad Salameh, Muhammad Fetrat, Jialin Zhang, Robin Luwei, Shangling Jui, Di Niu. "**Reparameterization through Spatial Gradient Scaling**," in Proceedings of the Eleventh International Conference on Learning Representations (**ICLR 2023**).

Lightweight Diffusion on Edge (Text to Image)

Method:

- Performed finetuning for velocity prediction on Midjourney dataset
- Neural Architecture Search with AutoGO
- Step Distillation on MidJourney 1.3M dataset to reduce inference steps
- Generation performance improvement with:
  - Adaptive Jump Step-Distillation and Semantic Alignment with Attention
  - Faithful and Realistic Text-to-Image Generation with Adaptive prompt-weighting



Achievements

- Reduced architecture latency by 78% with AutoGO
- We beat SnapFusion on 6k MS-COCO in terms of generation quality
  - CLIP score denoting better semantics (higher is better)
  - While maintaining comparable FID i.e generation quality (lower is better)

Prompt: "***Lion riding a bike in Paris***

Original (32steps, 55.9sec)

Original (8steps, 14sec)

Our reduced and optimized (8steps, <3sec)

SnapFusion

A photo of an astronaut riding a horse on mars

Our model

SnapFusion is able to generate a 512-by-512-pixel image, requires 8 steps

SD (Text-to-Image) for Edge

8-step model compared to SnapFusion (8 steps)

- Better Quality and Semantics

Ours – 8 steps

A photo of an astronaut riding a horse on mars

SnapFusion

Ours (8 steps)

3d render of voxel pink elephant

SnapFusion (8 steps)

A beautiful image of a cute animal surrounded by natural light, capturing their delicate beauty and charm

SD-based Inpainting for Edge

Optimize diffusion-based inpainting model for edge deployment

- Used **Progressive Step Distillation** to reduce UNet inference to 4 steps, while maintaining comparable output quality
- Used **UFOGen** to distill a 50-step model to a few-step generator following Generative Adversarial training
    - Reduce UNet inference steps to 1, with minimal drop in output quality
    - Estimated <1 second inference for 512 resolution

Metric Eval	Curation Score Two numbers closer = comparable PQ to baseline
UFOGen 1 Step	[159, 201]
StepDistill 4 Steps	[148, 167]

Human Eval	# Good PQ	# Acceptable, but has minor issues	# Bad PQ
Baseline 20 Steps	248	68	113
StepDistill 4 Steps	236	83	110

Original

inpainting

UFOGen
1 Step

StepDistill
4 Steps

Baseline
32 Steps

Software Mapping (Tensor Fusion)

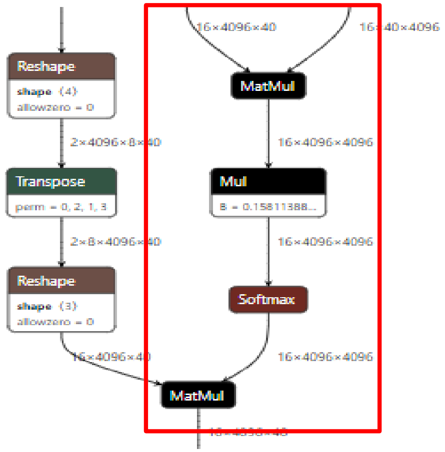
Tiling and Fusion: Software Mapping Techniques to Reduce Memory Access

Operational intensity is the number of compute operations an algorithm takes divided by the number of byte accesses it requires and is a hardware-agnostic measurement.

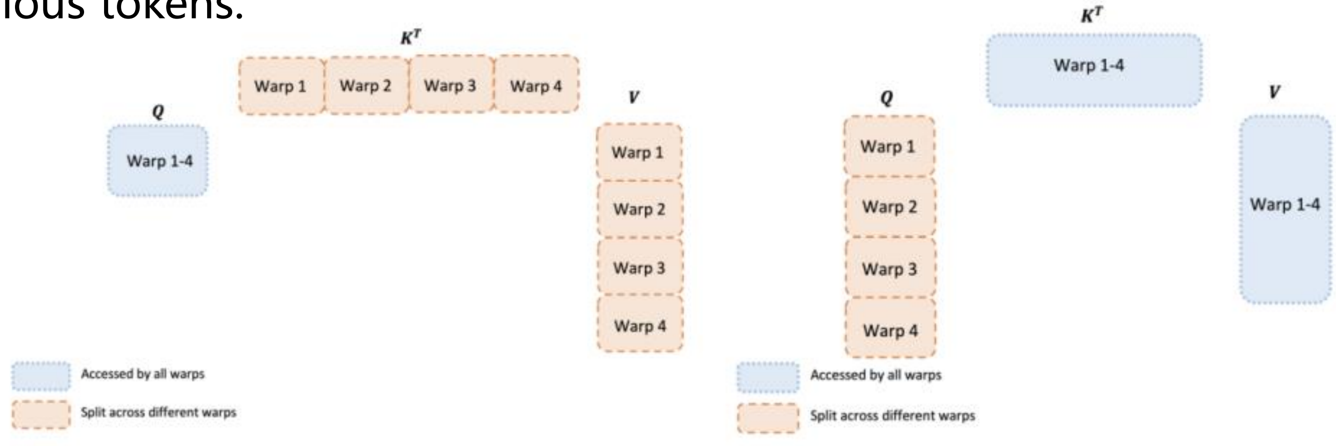
The most computationally expensive parts of a 7B parameter LLM are the **Attention layers**, which ensure next token predictions are weighted based on the relevance of previous tokens.

Attention Block

1. $Matmul(Q, K^T); Q, K \in R^{head_{size} \times Seq_{size} \times Emb_{size}}$
2. $Mul\left(QK^T, \frac{1}{\sqrt{d_{emb}}}\right); d_{emb} \in R$
3. $Softmax\left(\frac{QK^T}{\sqrt{d_{emb}}}\right)$
4. $Matmul(softmax, V)$



- ❖ $Latency = MatMul$ + Mul + Softmax + MatMul
- ❖ Issues: Memory bound, low compute utilization



(a) FlashAttention (b) FlashAttention-2

Figure 3: Work partitioning between different warps in the forward pass

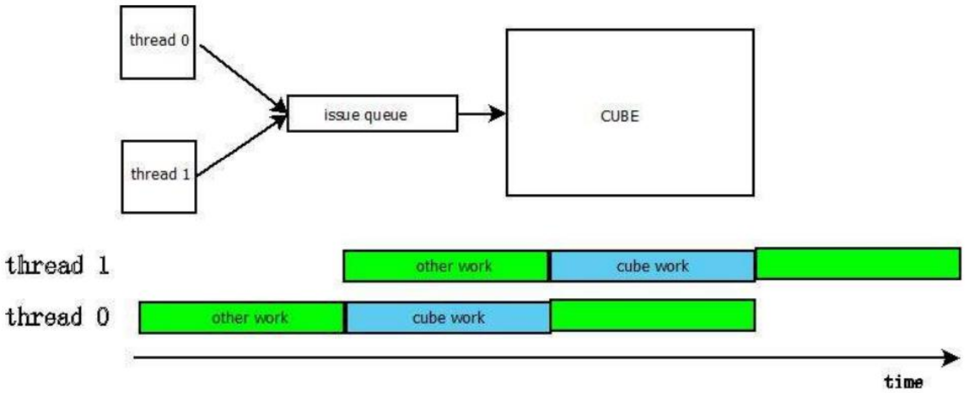
FlashAttention (& v2): For the GPU, perform tiling on K or Q, and allocate the fused tile computation to different Warps.

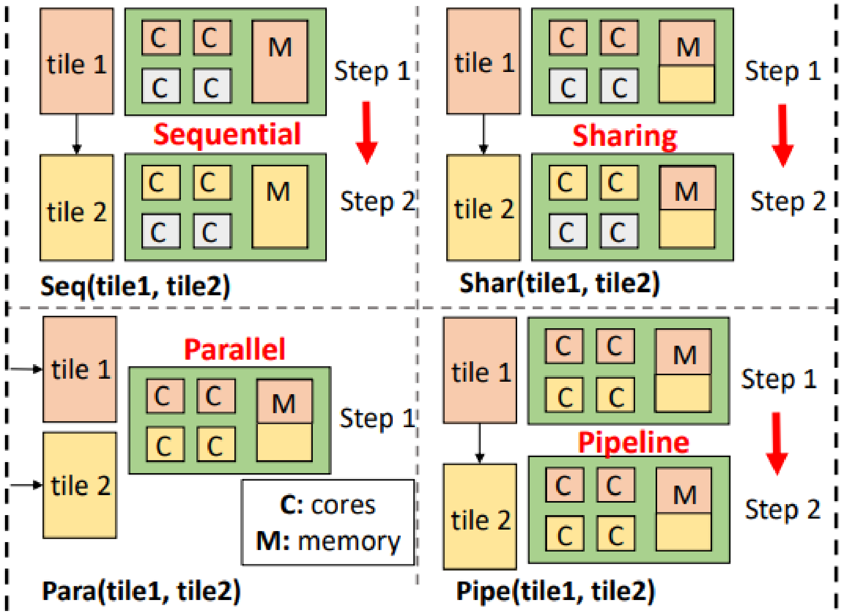
However, FlashAttention cannot directly benefit the edge devices.

- Different hardware architecture: resource binding is different but less explored in the literature
- Different parallelism: aim more aggressively at hiding memory operations

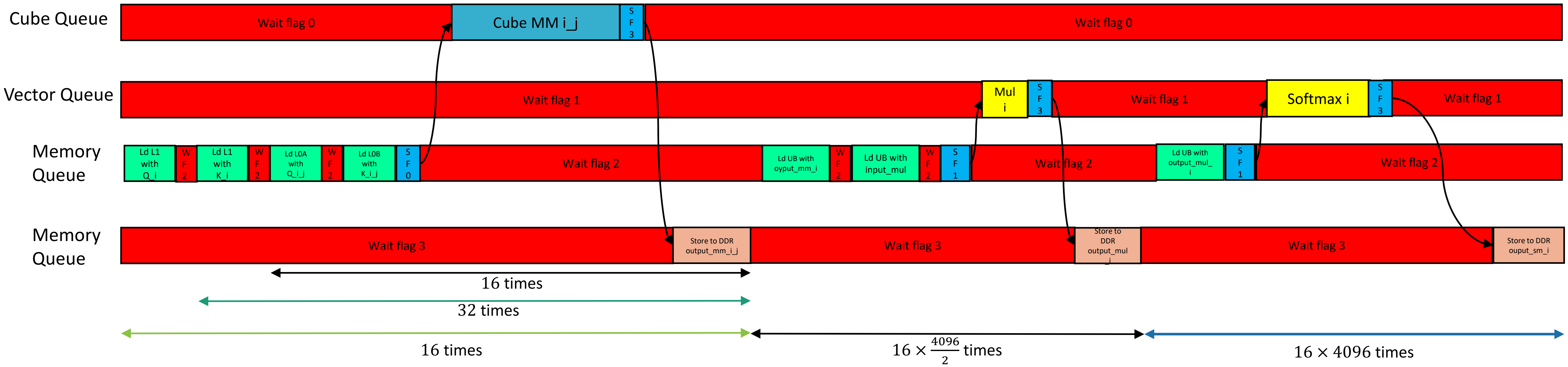
Tiling and Resource Binding as a Scheduling Problem

Given the accelerator HW configuration, resource binding optimization is to bind the flow of compute and mem operations for each tile onto hardware components, while reducing DDR memory access and parallelizing the operations.





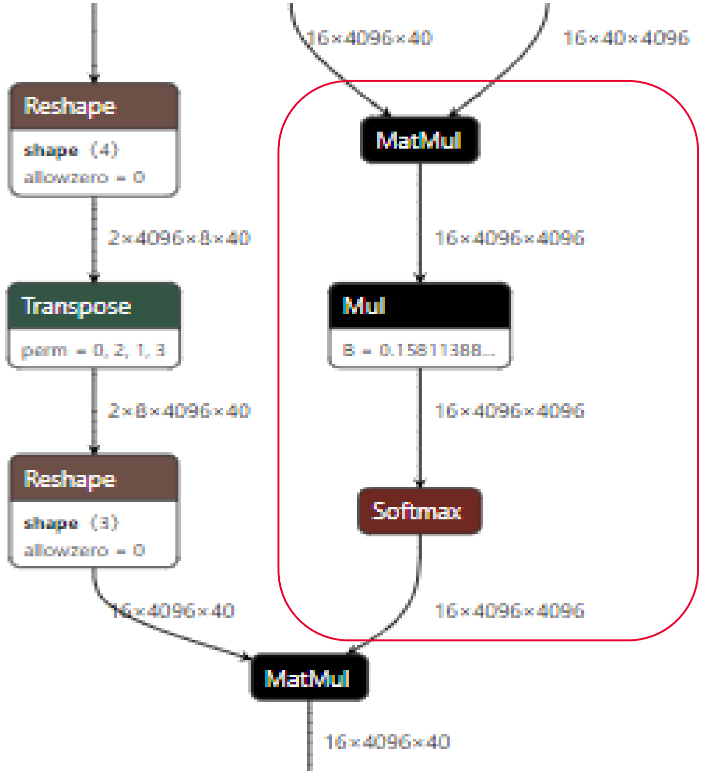
Figue credit: TileFlow (MICRO 2023)



Pipelined-Attention: Cube-Vector Pipelining for Attention

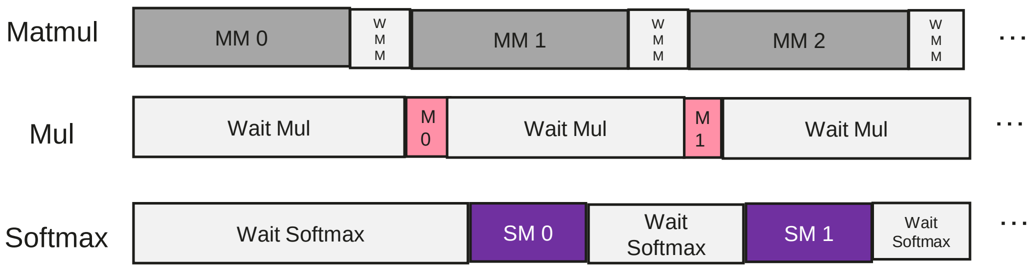
- Here the goal is to pipeline following operations:
  - $Matmul(Q, K^T); Q, K \in R^{head_{size} \times Seq_{size} \times Emb_{size}}$
  - $Mul\left(QK^T, \frac{1}{\sqrt{d_{emb}}}\right); d_{emb} \in R$
  - $Softmax\left(\frac{QK^T}{\sqrt{d_{emb}}}\right)$

- <u>Splitting sequence length of Q</u> with split factor (eg. 16)
- For each sub-sequence (tile) i, computing $Matmul_i$ and Mul_i and $Softmax_i$ using Cube-Vector pipelining



❖ Approximate Inference time for Baseline:
- $\#\text{Heads} \times (MM_{one\_head} + \frac{\text{Seq. Len}}{2} \times Mul_{i_{base}} + \text{Seq. Len.} \times SM_{i_{base}})$

❖ Approximate Inference time for Pipelining:
- $\#\text{Heads} \times \frac{\text{Seq. Len.}}{Split\ factor} \times \max\left(MM_1, (Mul_0 + SM_0)\right)$



❖ Empirical Inference time for 3 operations with a split factor of 16:

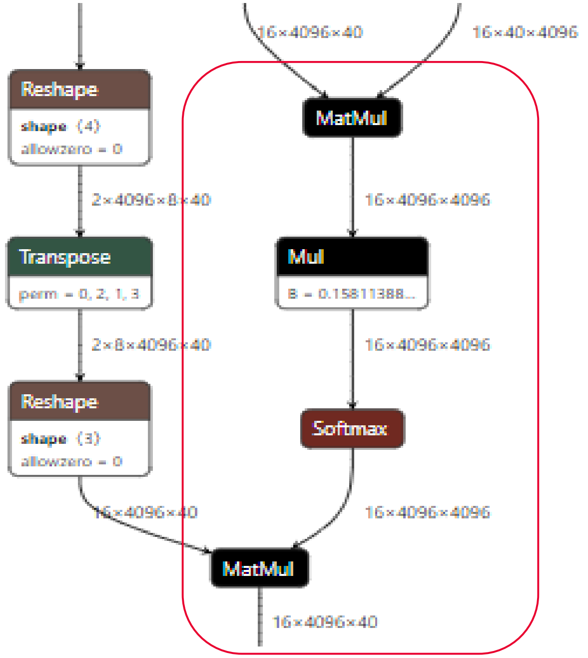
Method	Q_{shape}	K^T_{shape}	Inference Time
Baseline (no fusion)	$(16, 4096, 40)$	$(16, 40, 4096)$	$443.389ms$
PipelinedAttention-v1	$(16, 4096, 40)$	$(16, 40, 4096)$	$257.234ms$

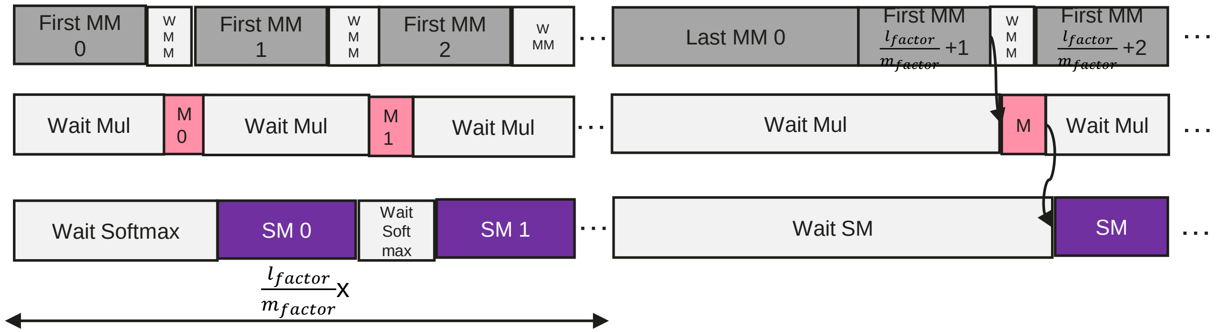
➢ **Latency Reduction for 3 operations: 41.99%**

Pipelined-Attention as an Optimization Problem

❖ Use optimization to solve the following problem:

1. Pipeline the same operations:
   - $Matmul(Q, K^T); Q, K \in R^{head_{size} \times Seq_{size} \times Emb_{size}}$
   - $Mul\left(QK^T, \frac{1}{\sqrt{d_{emb}}}\right); d_{emb} \in R$
   - $Softmax\left(\frac{QK^T}{\sqrt{d_{emb}}}\right)$
2. Find the optimal split factor "$\boldsymbol{m}$" to split Q
3. Find the optimal number of consecutive pipelined chunks of $Softmax\left(\frac{QK^T}{\sqrt{d_{emb}}}\right)$ to be given to the last *Matmul* operator.





❖ Example inference time for certain choices:
- The best parameters for Pipelined-Attention should be searched for

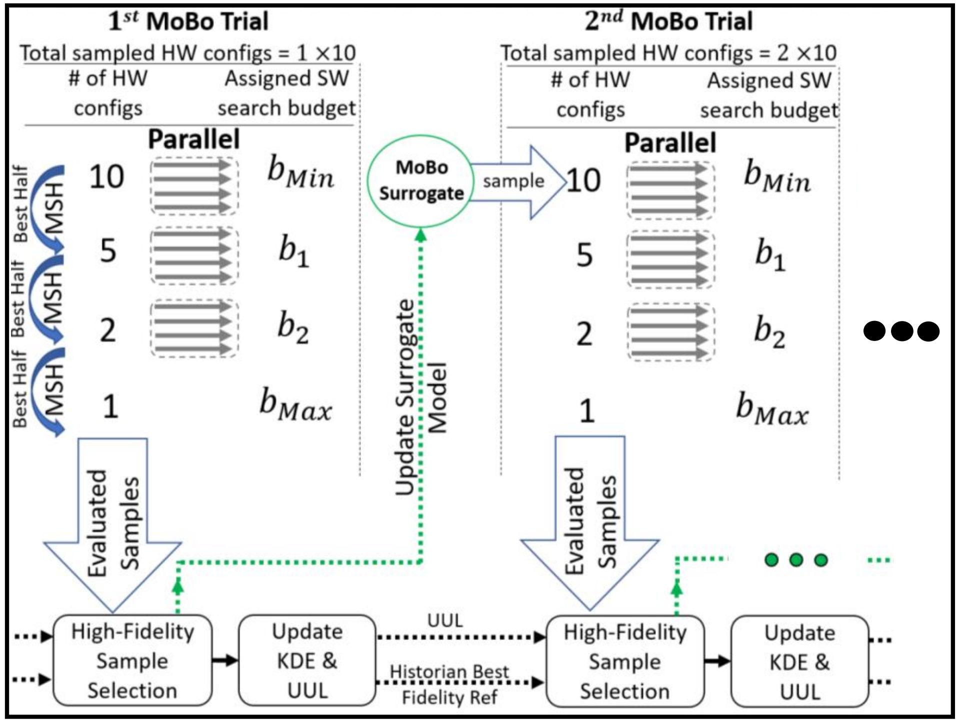
Method	Q, K, V_{shape}	l_{factor}	m_{factor}	Latency Reduction for 4 operations
Baseline (no fusion)	$(16, 4096, 40)$	-	-	-
PipelinedAttention-v1	$(16, 4096, 40)$	-	16	24.1%
PipelinedAttention-v2.1	$(16, 4096, 40)$	512	16	<u>**31.15%**</u>

Hardware Configuration

UNICO: Robust Accelerator Hardware Configuration Search

Innovation:

- Use Successive halving to reduce SW search budget for some "bad" HW parameters
- Proposed robust HW search, based on power/latency sensitivity to SW, to improve generalization of the found HW to different unseen DNNs



On open-source platform:

- Training NN: MobileNetV2, ResNet, SRGAN, VGG;
- Tested on 7 new NNs: UNICO vs HASCO, 44% improvement
- Both convergence speed and robustness to new DNN workloads outperform HASCO

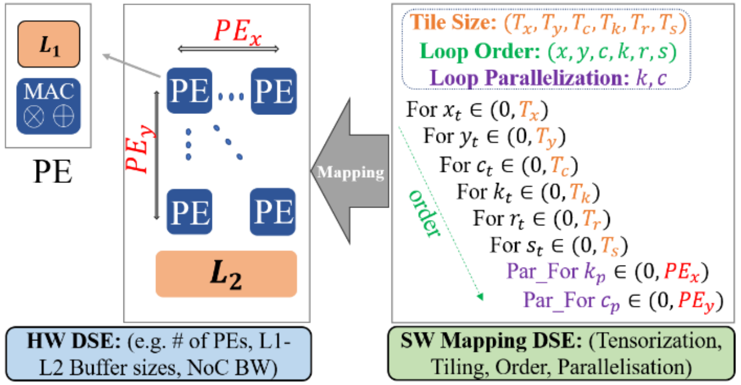


Figure 1: A typical 2D spatial accelerator HW design components (e.g. (PE_x, PE_y), L_1 and L_2 buffer sizes)

DaVinci NPU verification:

- Model: Denoise U-Net, 6 variants of FSRCNN, DLSS
- Search: Use Power as the main target
- By adjusting L0A/B/C buffer parameters, on DLSS can achieve 54% power reduction;
- On all other NNs, there is no power or lateny degradation

UNICO HW config. Vs. Original DaVinci HW Config

HW	L0A	L0B	L0C	…	Cube	Area
Original	32	32	64	unchanged	unchanged	unchanged
UNICO found	128	8	16	unchanged	unchanged	unchanged

Bahador Rashidi, Chao Gao, Shan Lu, Zhisheng Wang, Chunhua Zhou, Di Niu and Fengyu Sun. "**UNICO: Unified Hardware Software Co-Optimization for Robust Neural Network Acceleration**", in IEEE/ACM International Symposium on Microarchitecture (**MICRO 2023**)