DNNFusion: Accelerating Deep Neural Networks Execution with Advanced Operator Fusion

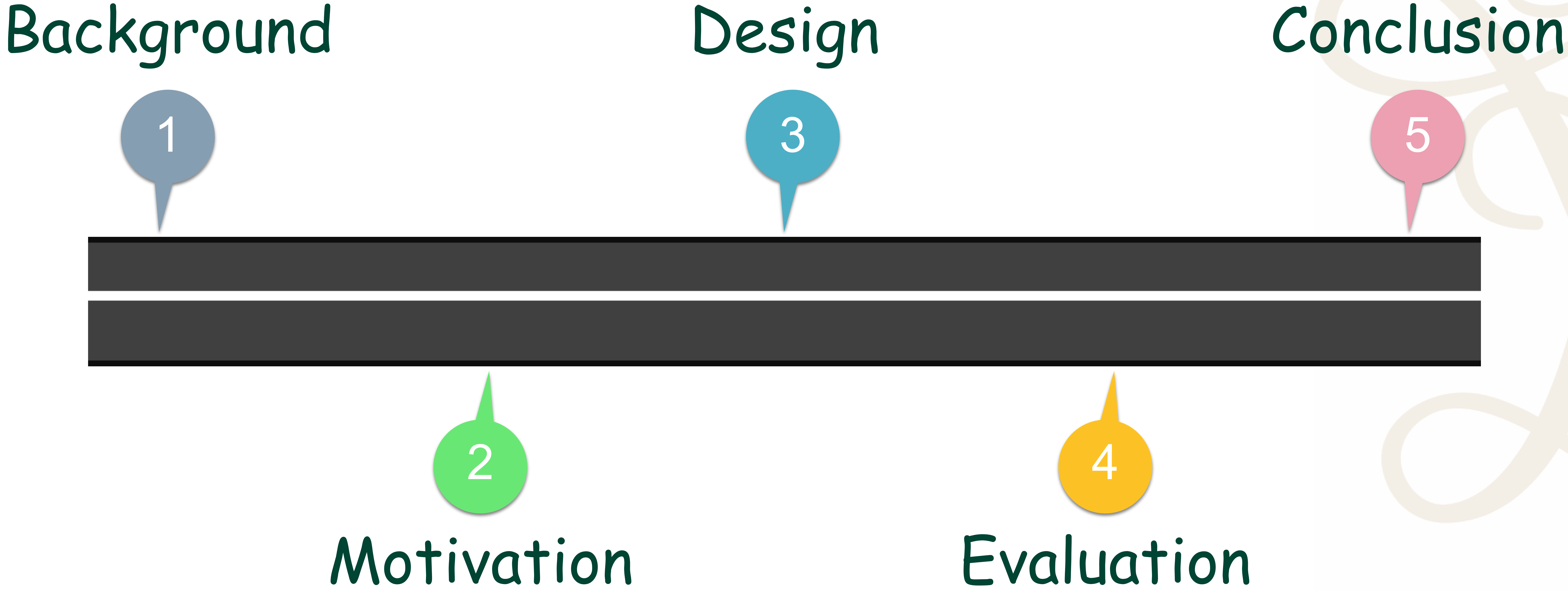
Wei Niu, Jiexiong Guan, Yanzhi Wang, Gagan Agrawal, Bin Ren

William & Mary
CHARTERED 1693

Augusta
University

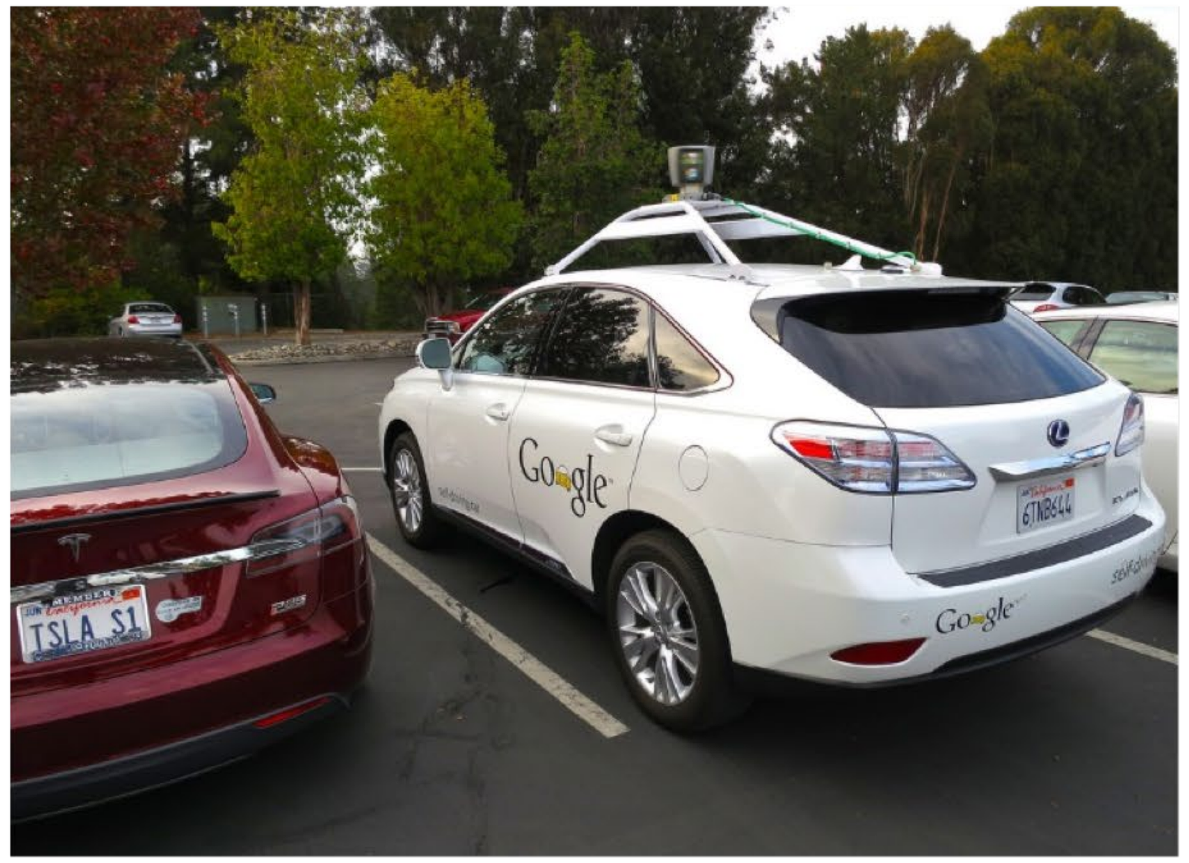
Northeastern
University

Roadmap


Background
1
Motivation
2
Design
3
Evaluation
4
Conclusion
5

Deep learning is everywhere

Self-Driving



Speech Recognition

Object Detection

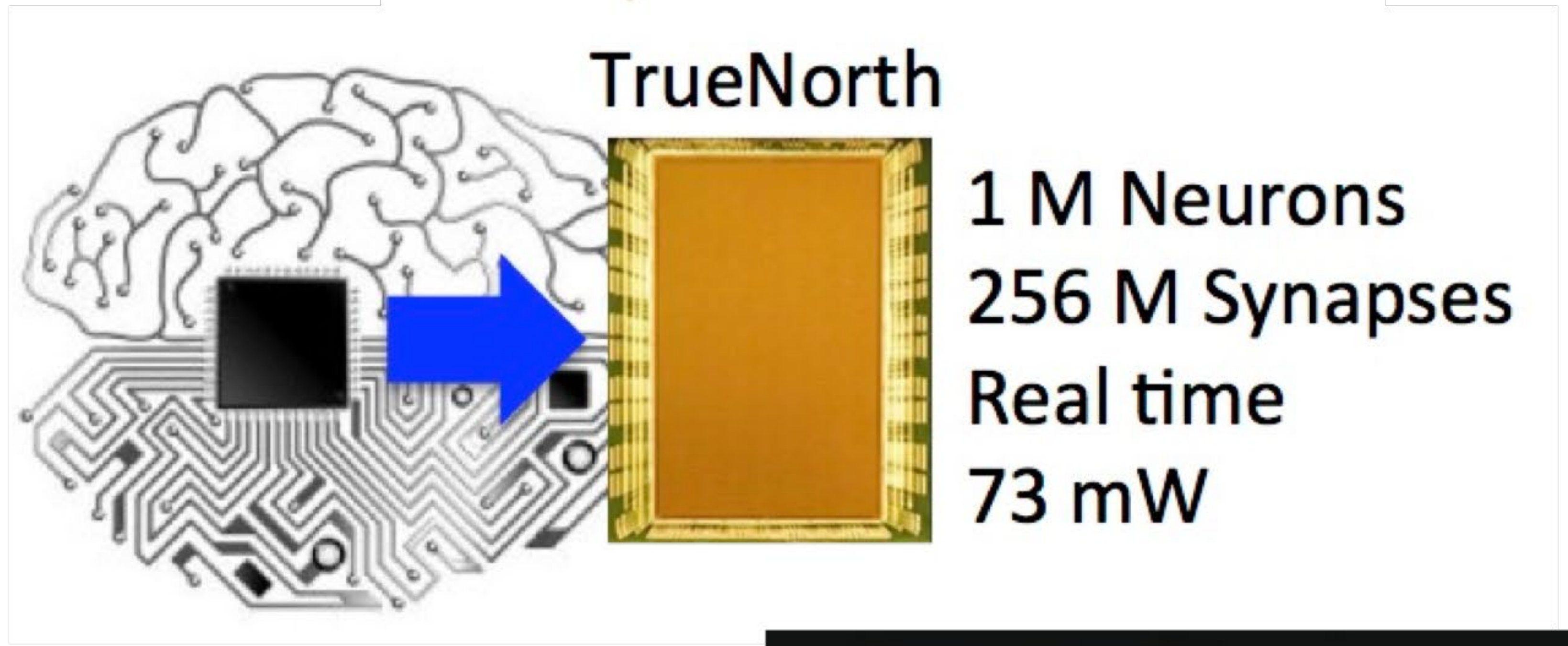
OpenAI Five playing Dota 2
Source: OpenAI

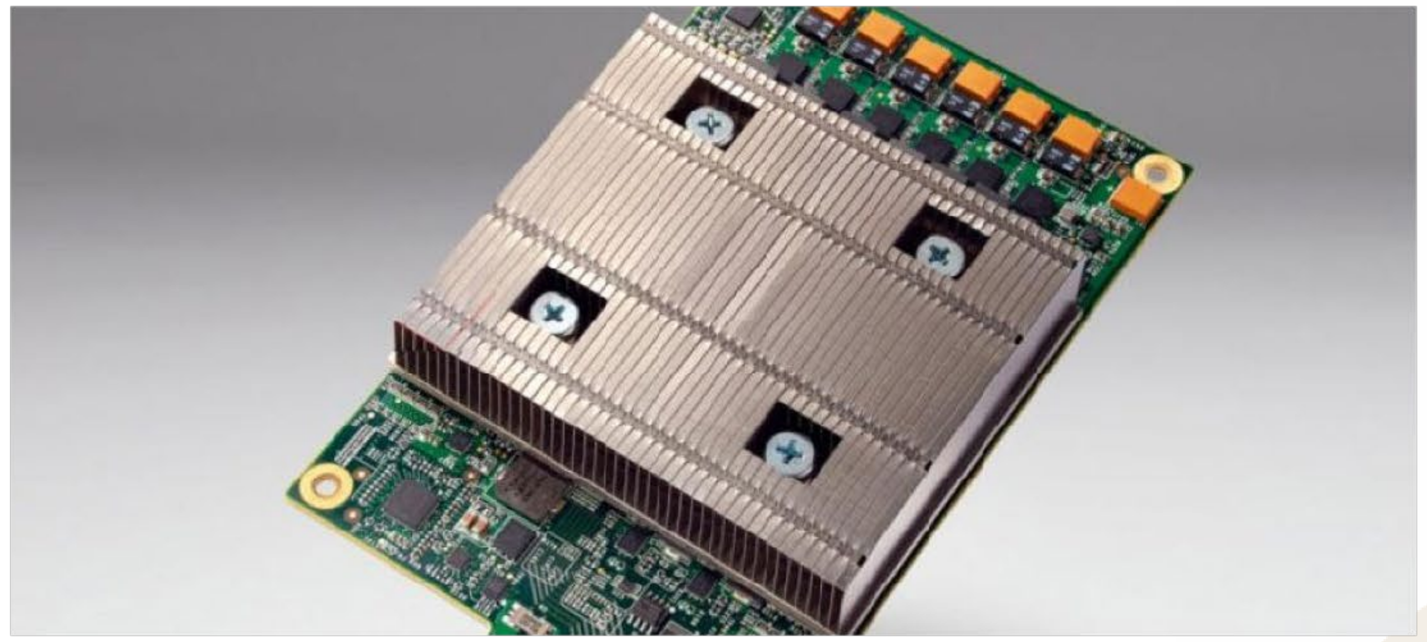
Healthcare
Image credits to Brother UK.

Deep learning computation devices

GPU

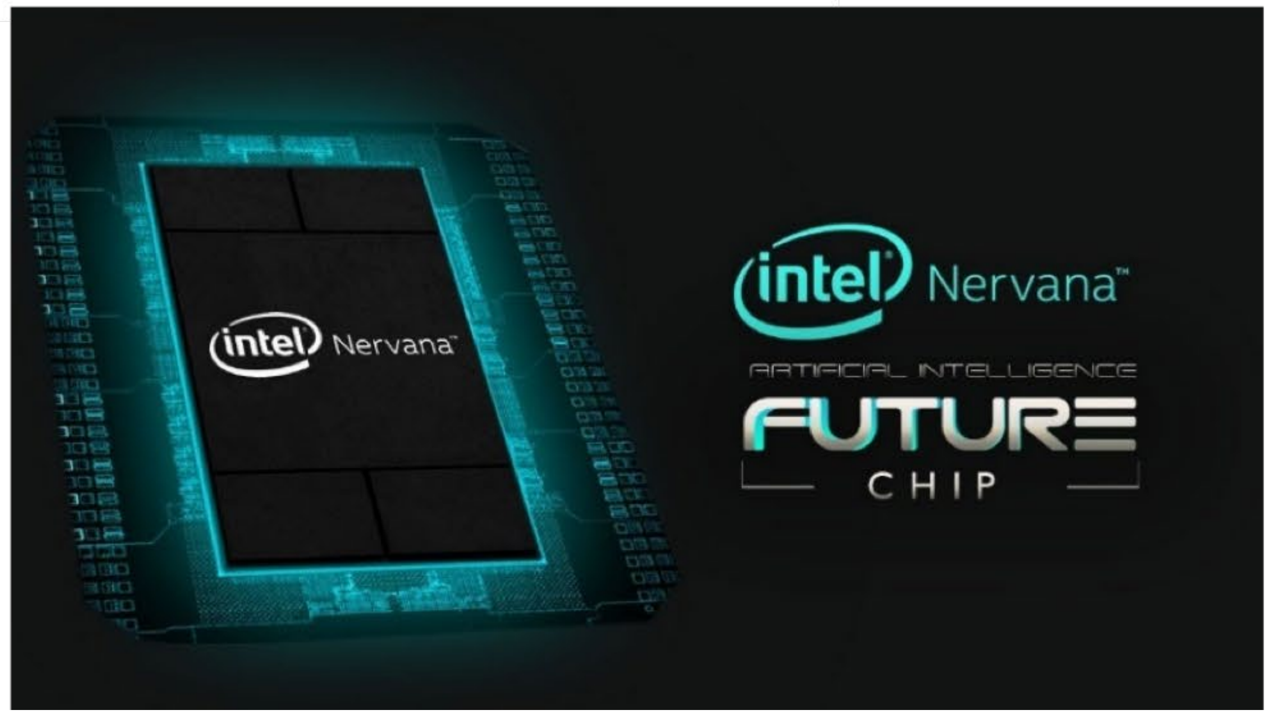
FPGA





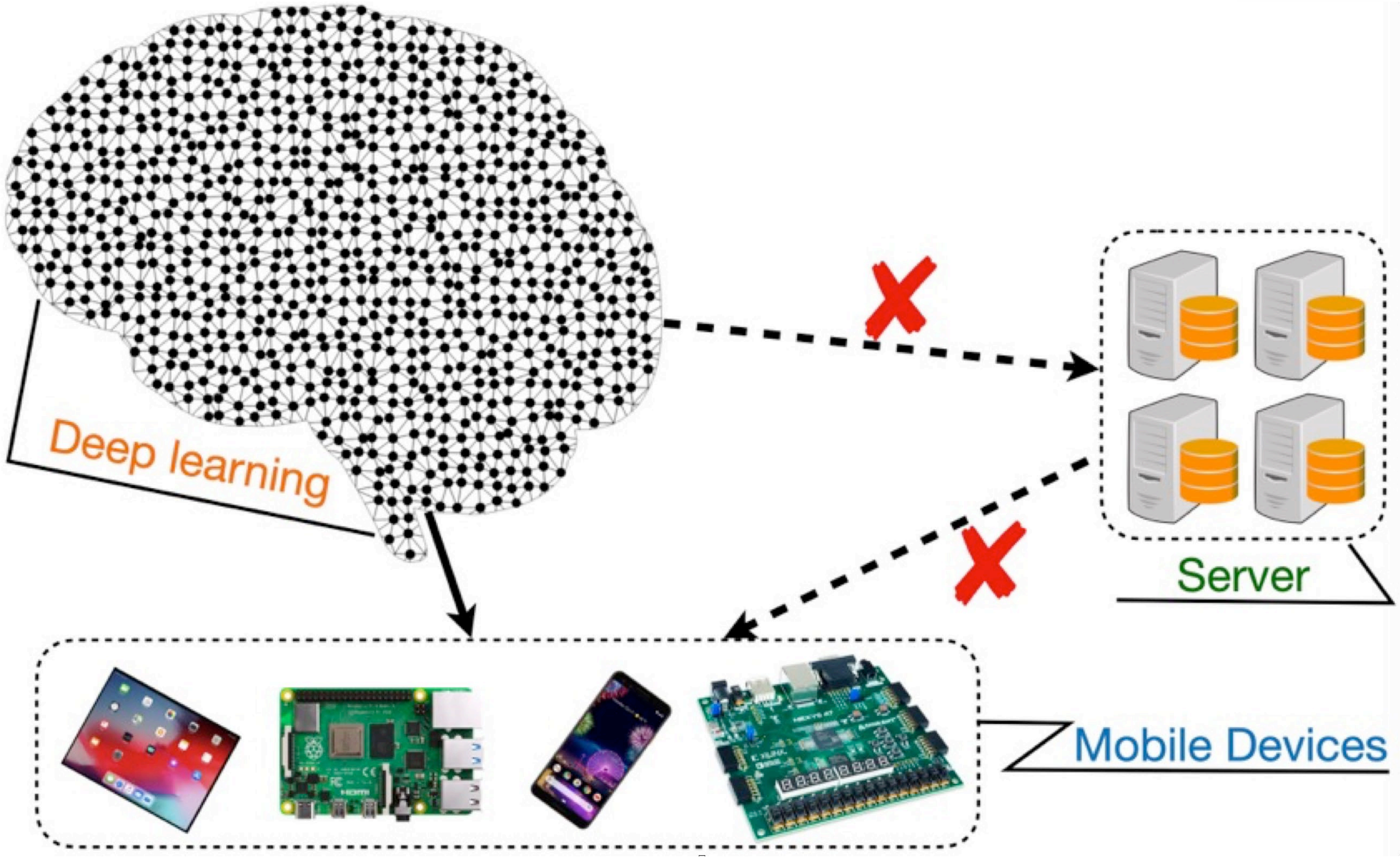
IBM TrueNorth

Google TPU

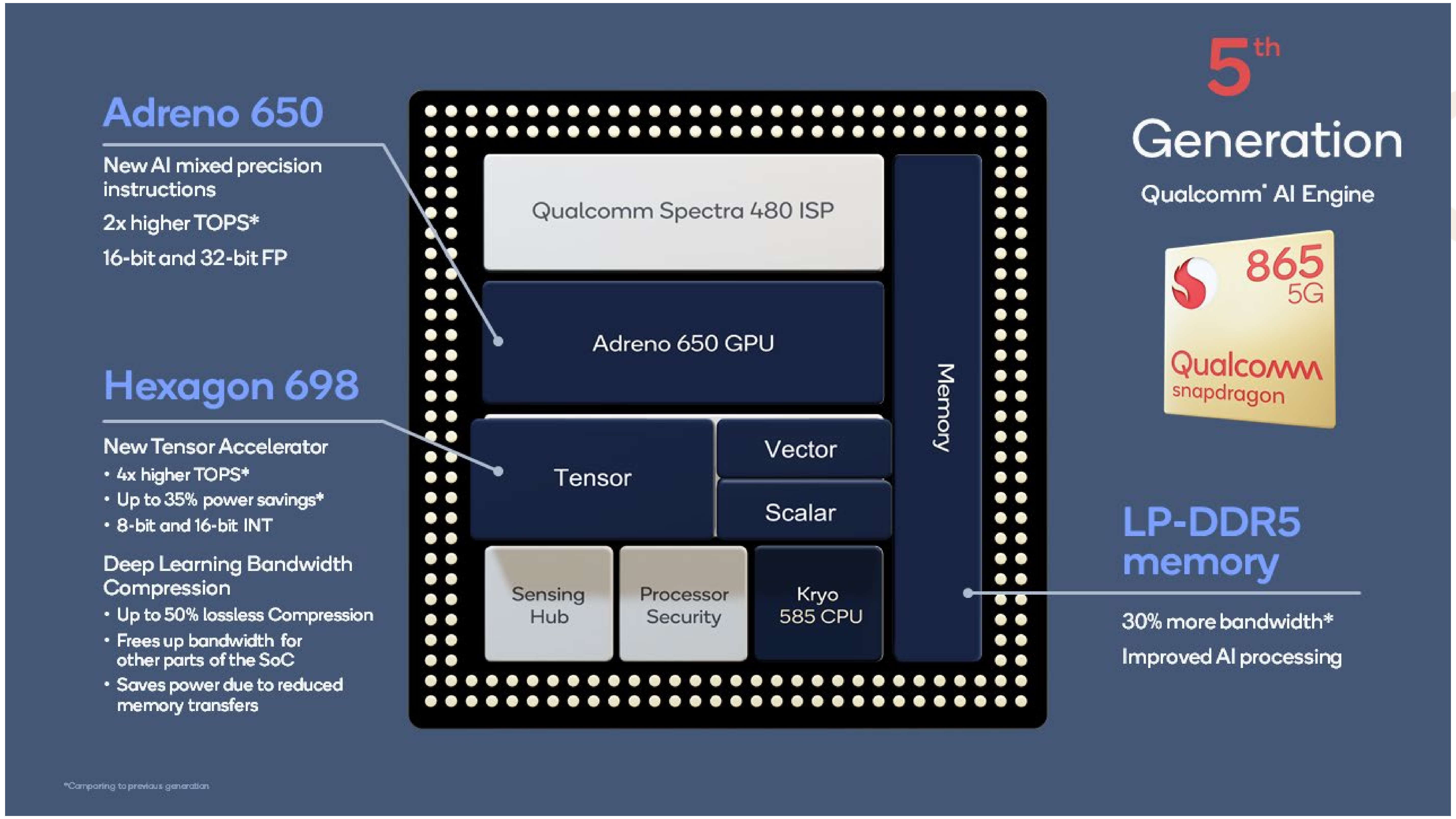


Intel Nervana

Powerful & efficient mobile devices



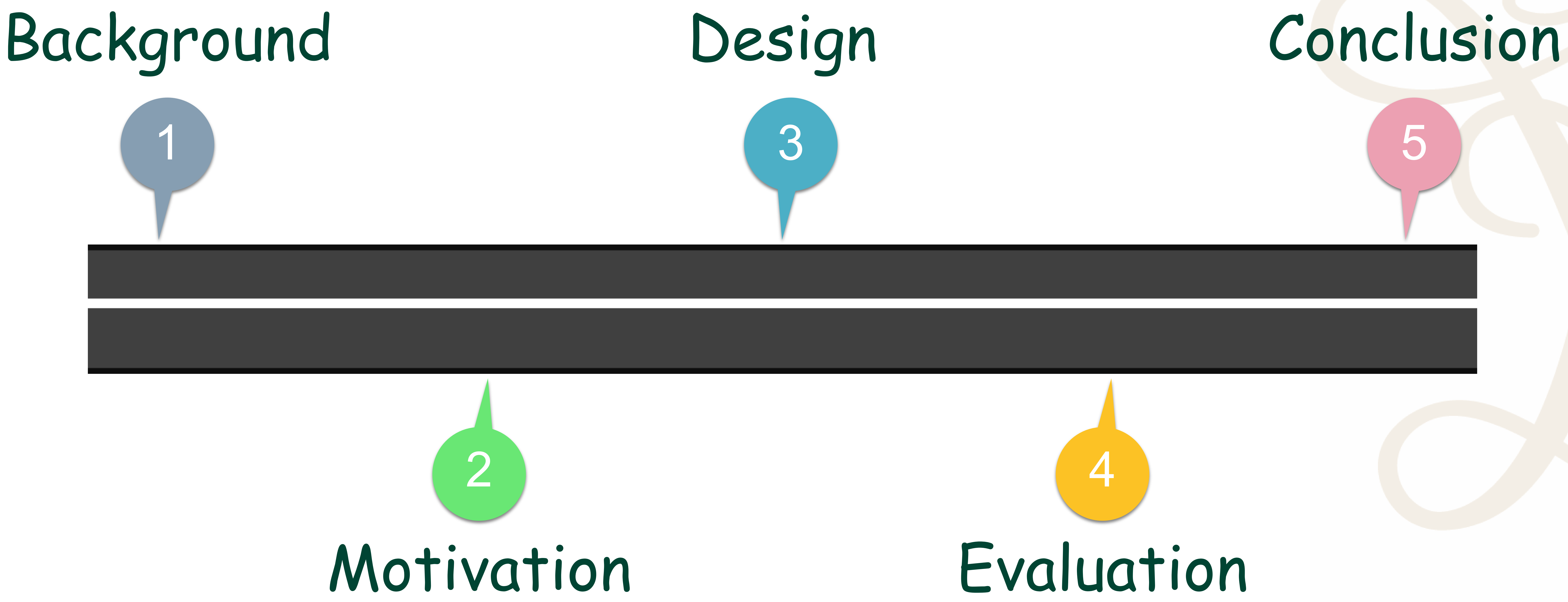
Powerful & efficient mobile devices



Adreno 650 [1] can even achieve over 1T flops peak performance with only 5W power consumption [2]

[1]. https://developer.qualcomm.com/blog/tvm-open-source-compiler-now-includes-initial-support-qualcomm-hexagon-dsp
[2]. List of Qualcomm Snapdragon processors: https://www.wikiwand.com/en/List_of_Qualcomm_Snapdragon_processors

Roadmap

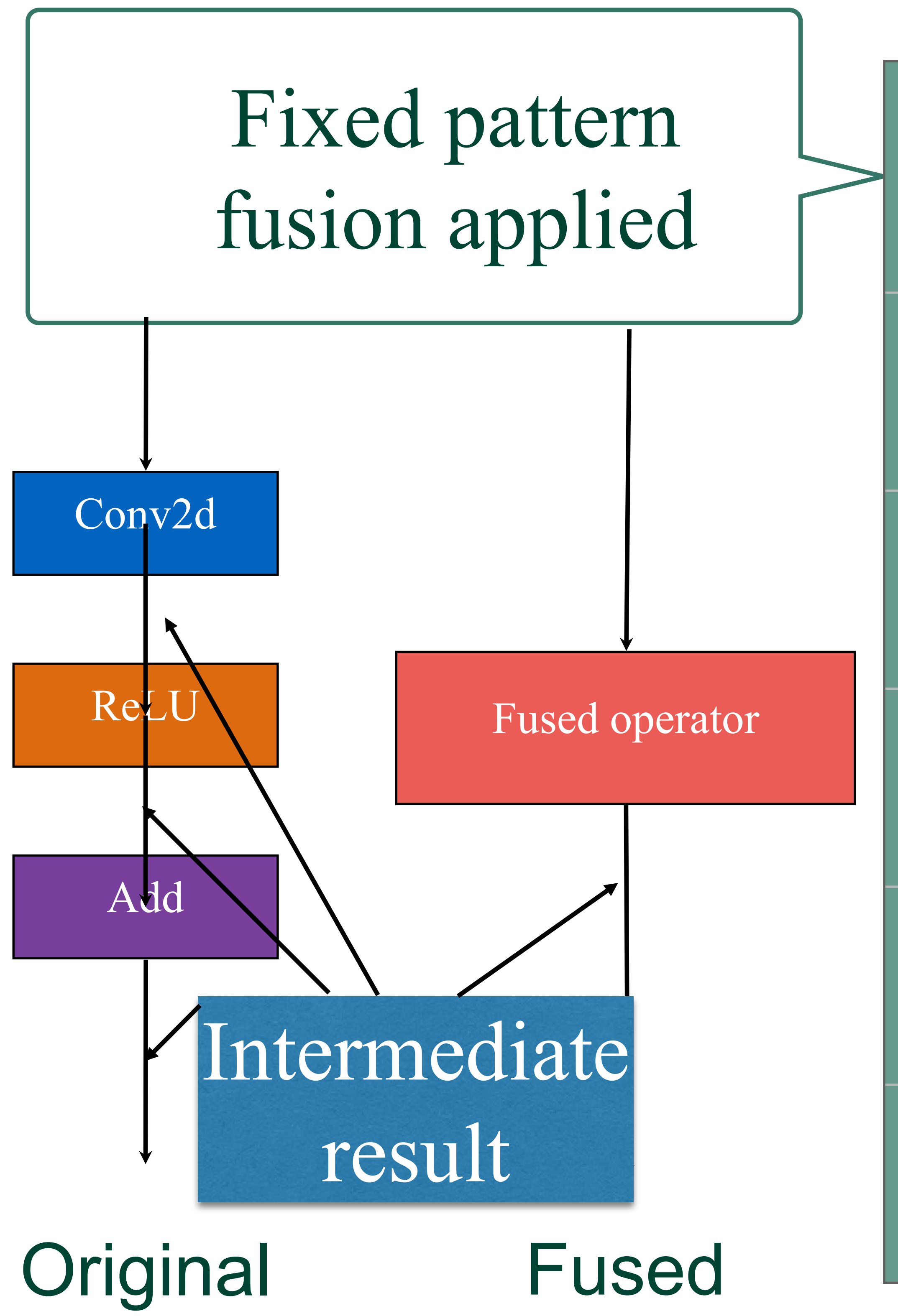

Background
1
Design
3
Conclusion
5
2
Motivation
4
Evaluation

Deeper neural networks

- There has been a trend towards deeper models
  1. E.g., MobileBERT, GPT-2, Conformer

Over 1000 operators

Deep VS shallow



Model	Number of layers	Number of FLOPS	Speed (FLOPs/S)
VGG-16	51	31.0B	320G
YOLO-V4	389	34.6B	135G
DistilBERT	457	35.3B	78G
MobileBERT	2387	17.6B	44G
GPT-2	2533	69.1B	62G

Correlation between Speed and #FLOPS and #layers

Deeper neural networks

- Depth of the model is the critical impediment to efficient execution
  1. More intermediate results, thus increasing the memory/cache pressure
  2. Insufficient amount of computations in each layer, thus degrading the processor's utilization

Limitation of state-of-the-art frameworks

End to end mobile frameworks

- MNN
- TVM
- Pytorch-Mobile
- TFLite

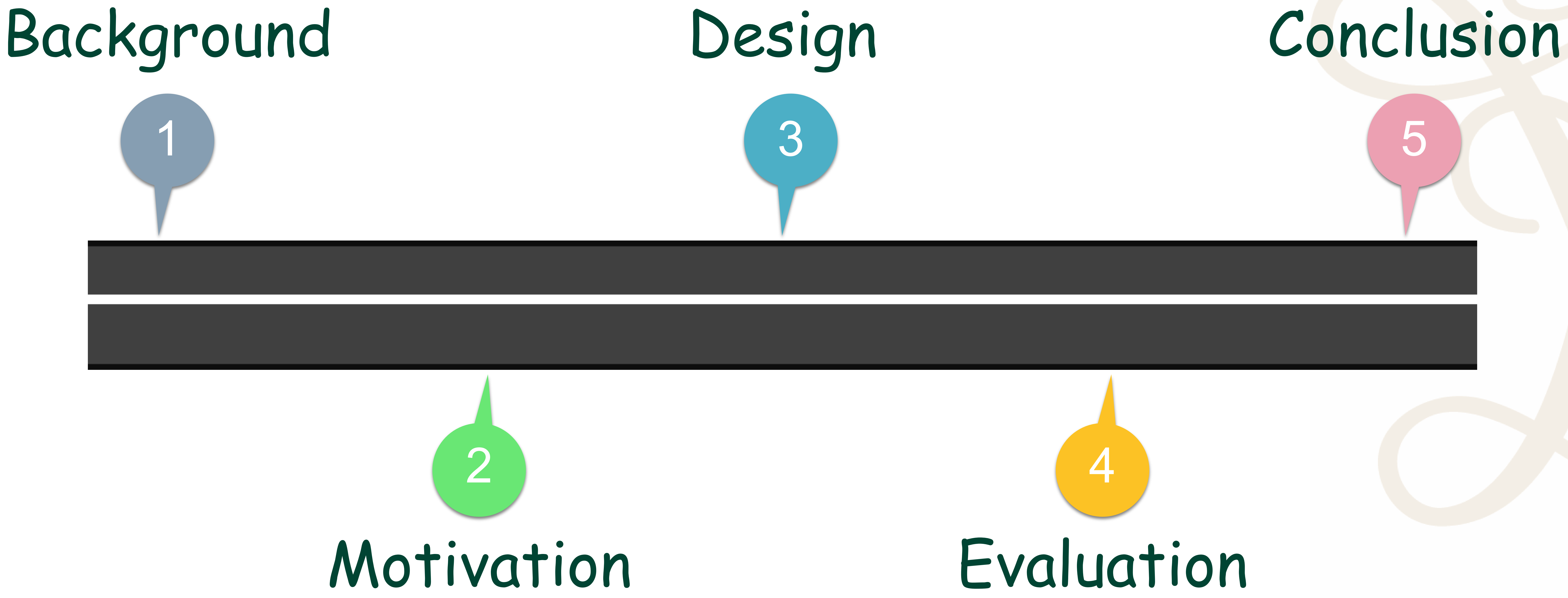
Already have over 100 different operators

Our contribution

Operator mapping analysis	Mathematical-property-based graph rewriting
Fusion plan generation	Fusion code generation

Niu W, Ma X, Lin S, et al. Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning[C], ASPLOS 2020

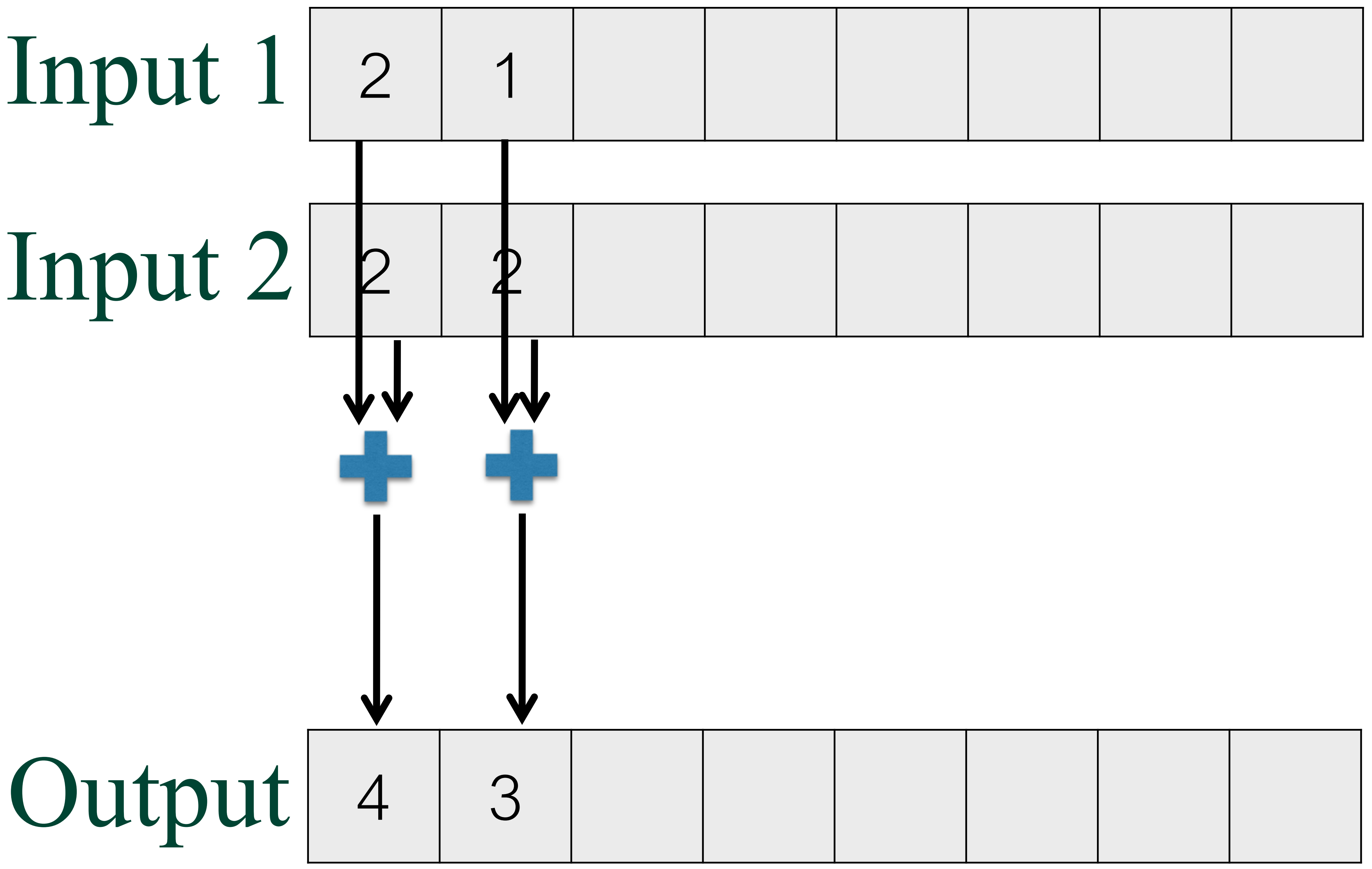
Roadmap


Background
1
Design
3
Conclusion
5
2
Motivation
4
Evaluation

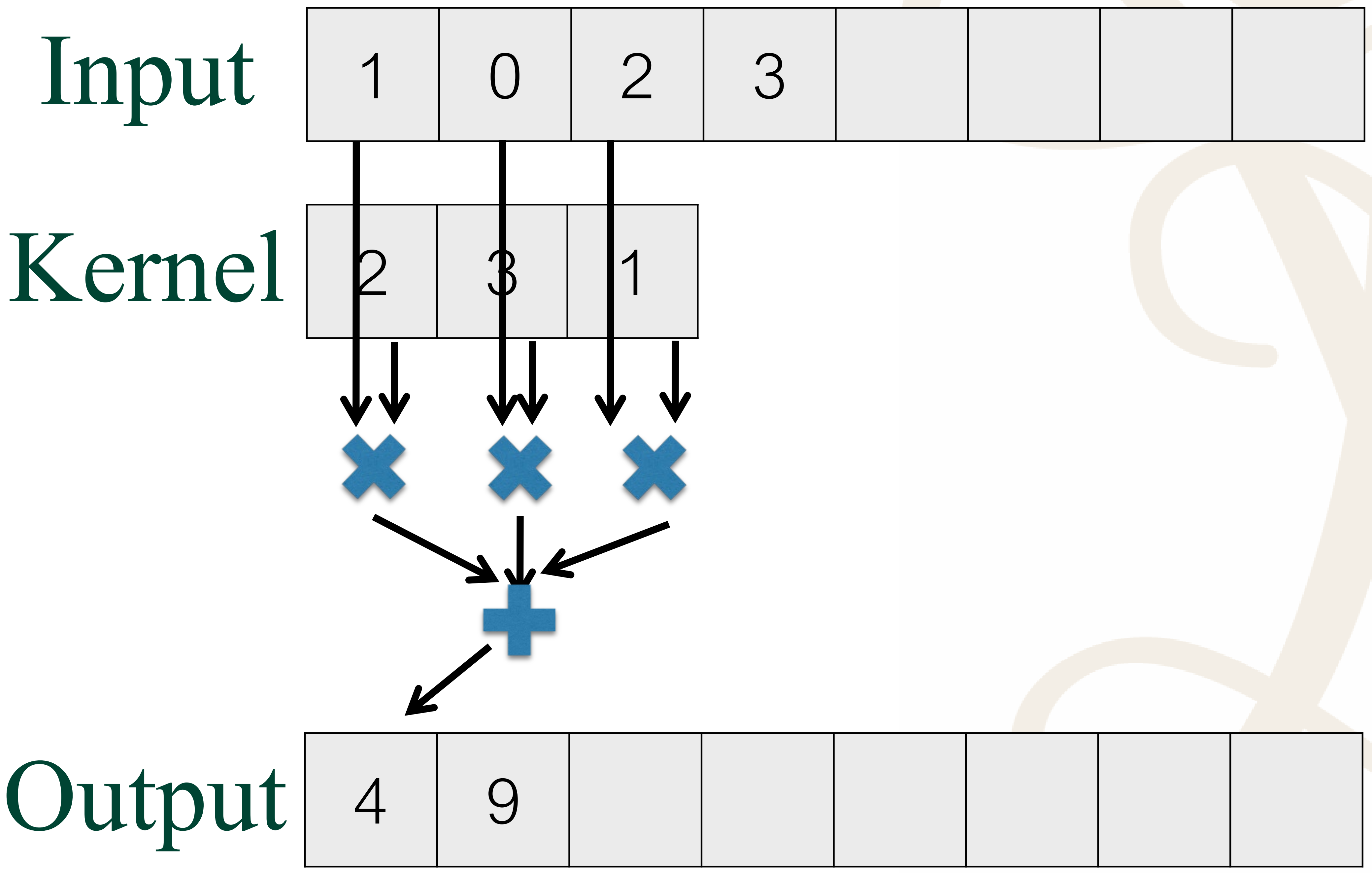
Operator mapping types

Mapping type	Representative operators
One-to-One	Add, Relu
One-to-Many	Gather, Upsample
Many-to-Many	Convolution, GEMM
Reorganize	Reshape
Shuffle	Transpose

Operator mapping types

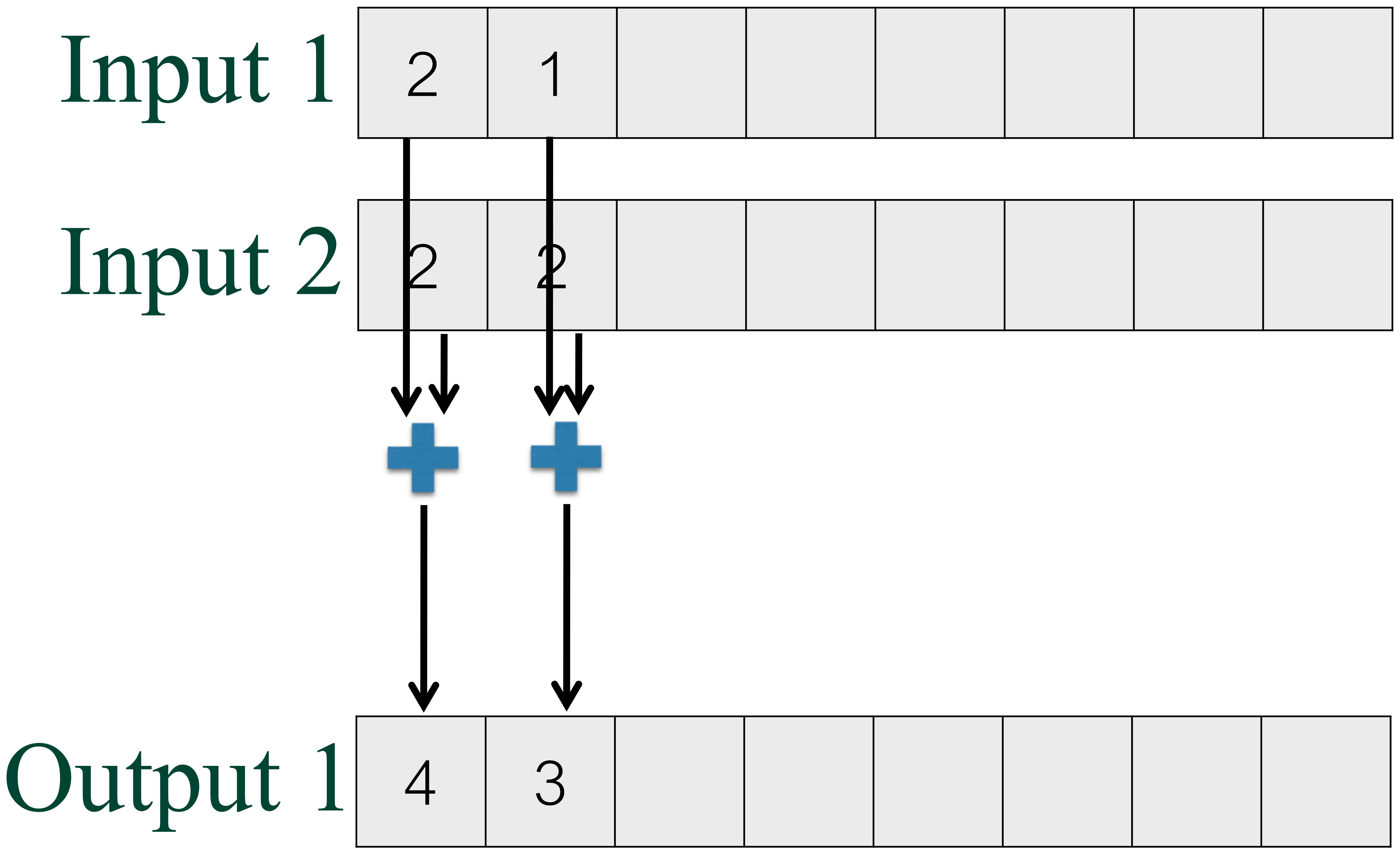


One-to-One (Add)

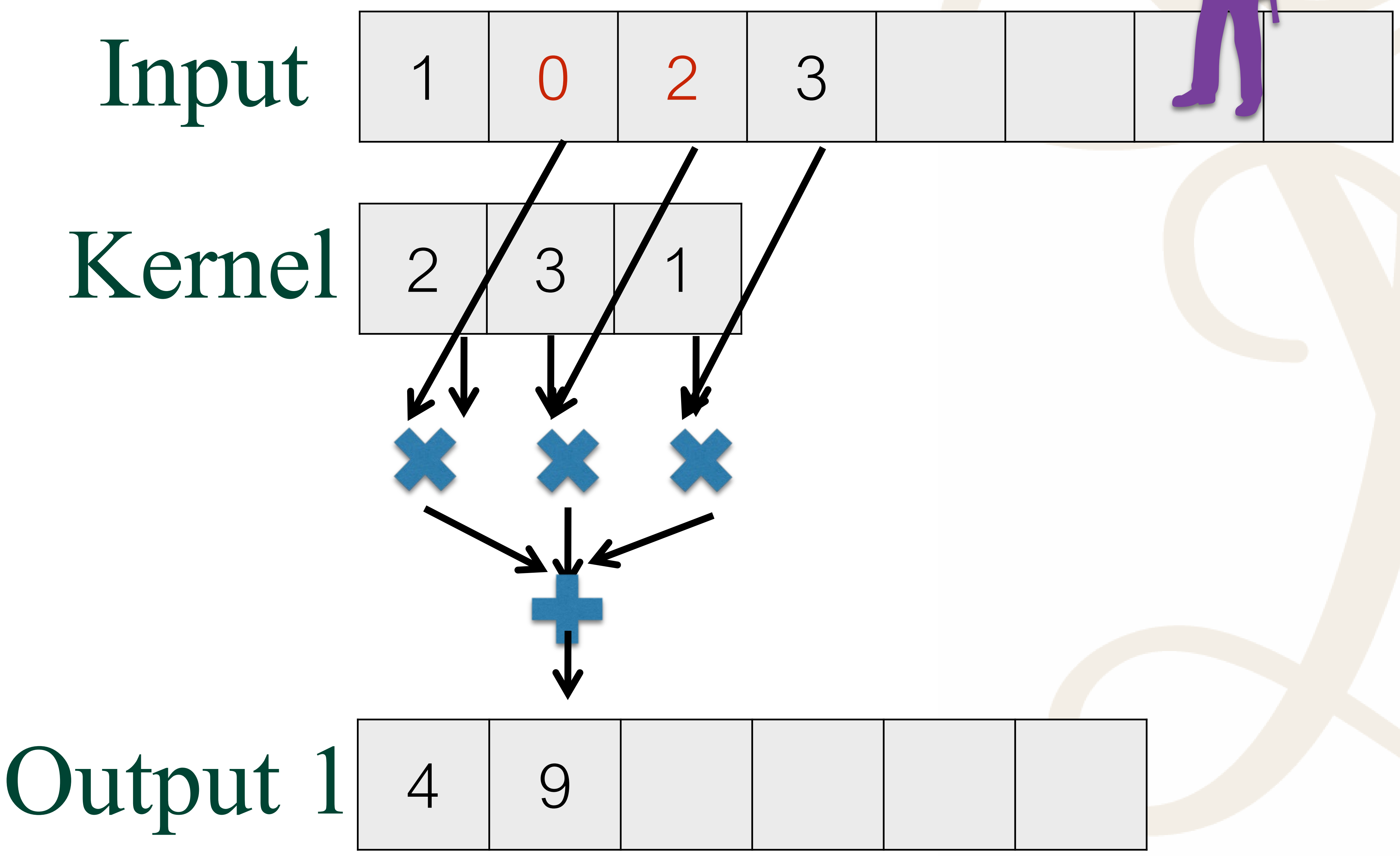


Many-to-Many (Conv)

Operator mapping types



One-to-One (Add)

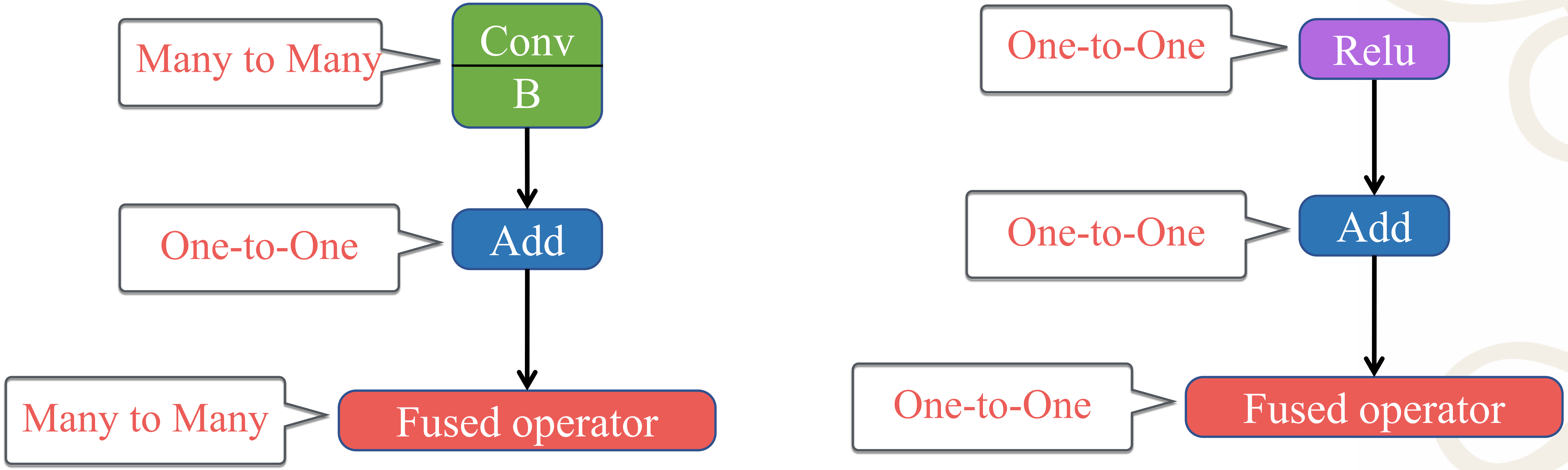


Many-to-Many (Conv)

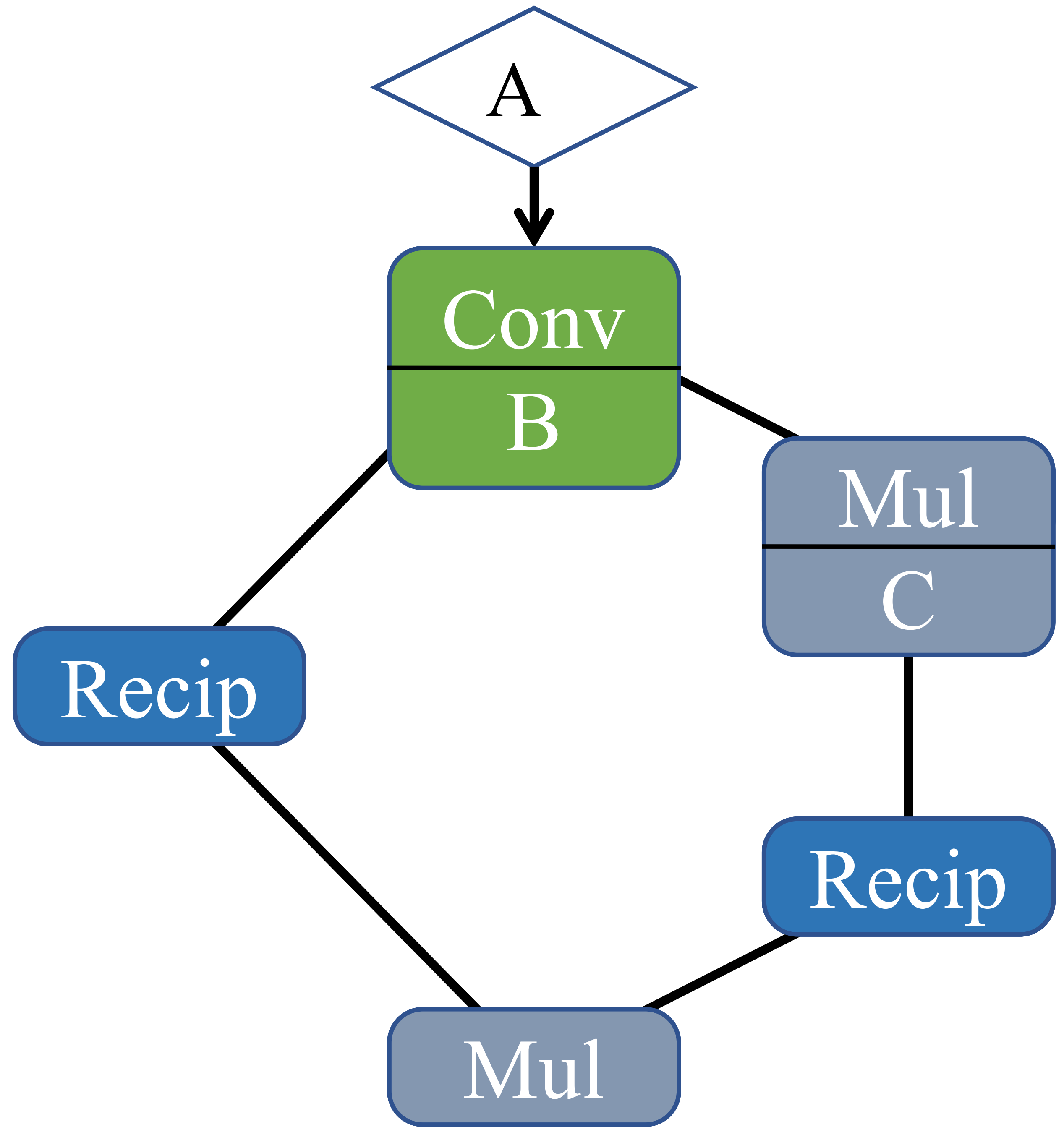
Mapping types combination analysis

	One-to-One	One-to-Many	Many-to-Many	Reorganize	Shuffle
One-to-One	One-to-One	One-to-Many	Many-to-Many	Reorganize	Shuffle
One-to-Many	One-to-Many	One-to-Many	X	One-to-Many	One-to-Many
Many-to-Many	Many-to-Many	Many-to-Many	X	Many-to-Many	Many-to-Many
Reorganize	Reorganize	One-to-Many	Many-to-Many	Reorganize	Reorganize
Shuffle	Shuffle	One-to-Many	Many-to-Many	Reorganize	Shuffle

Operator mapping types

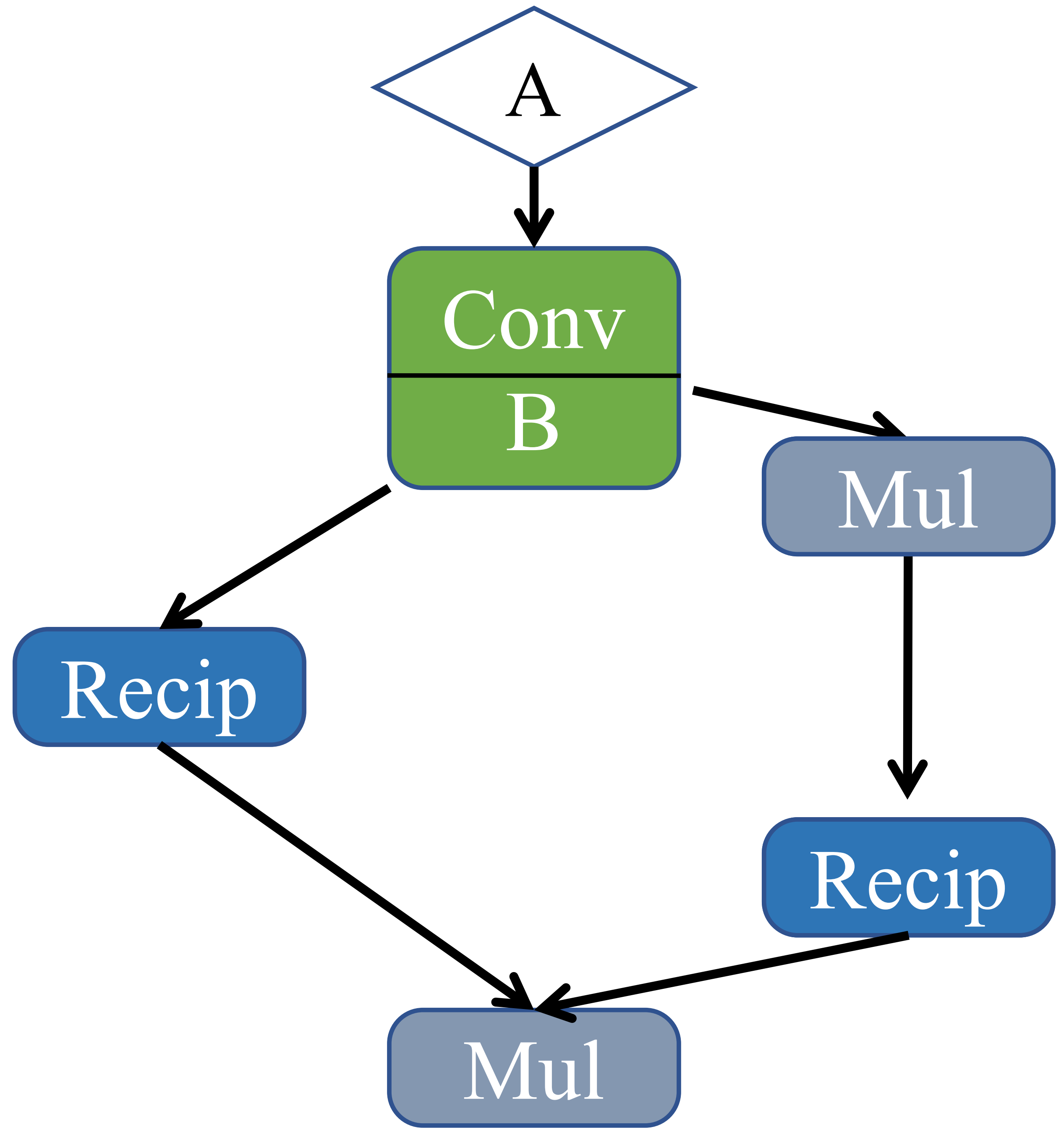


Mathematical-property-based graph rewriting

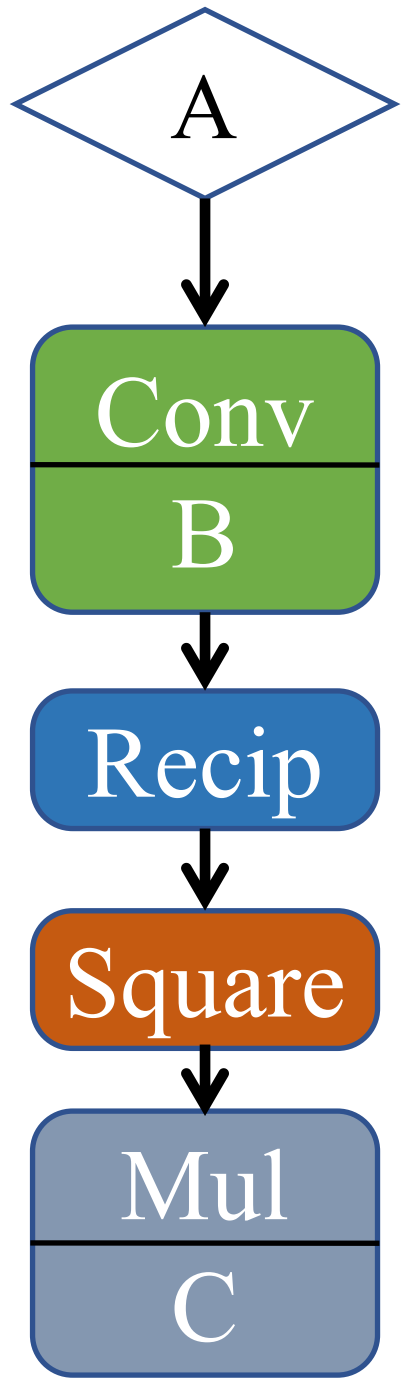


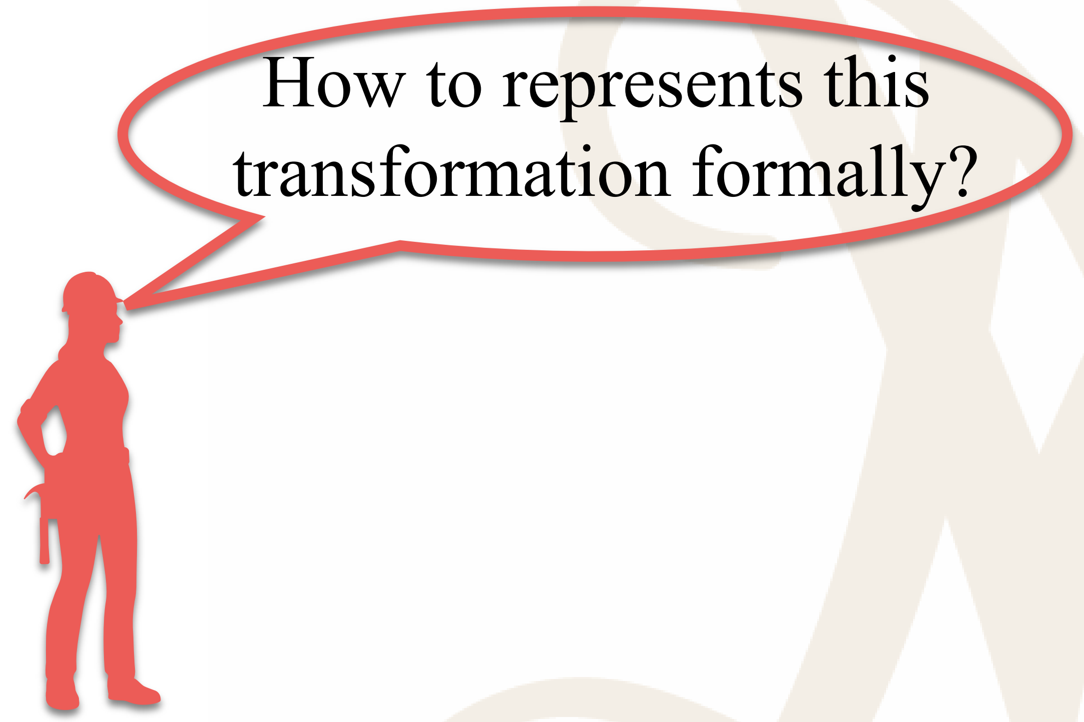
Before: $(A \circledast B)^{-1} \odot ((A \circledast B) \odot C)^{-1}$

Mathematical-property-based graph rewriting



Before: $(A \circledast B)^{-1} \odot ((A \circledast B) \odot C)^{-1}$



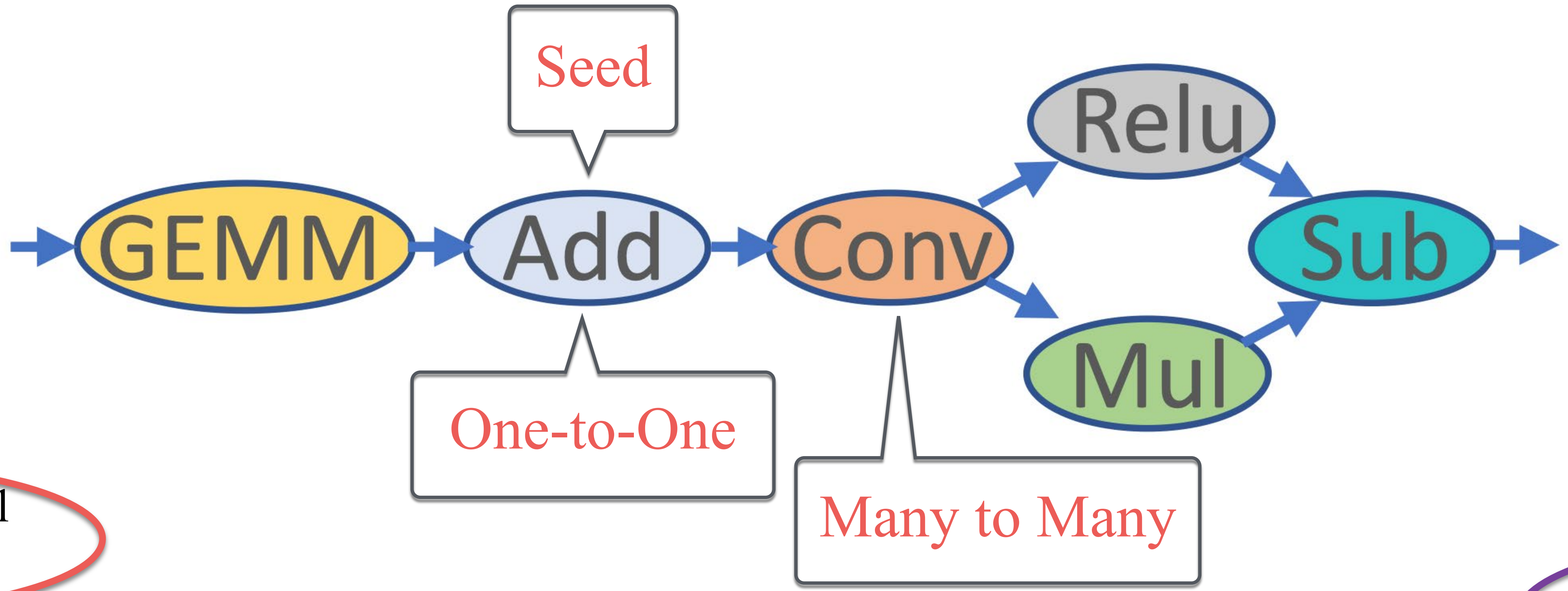


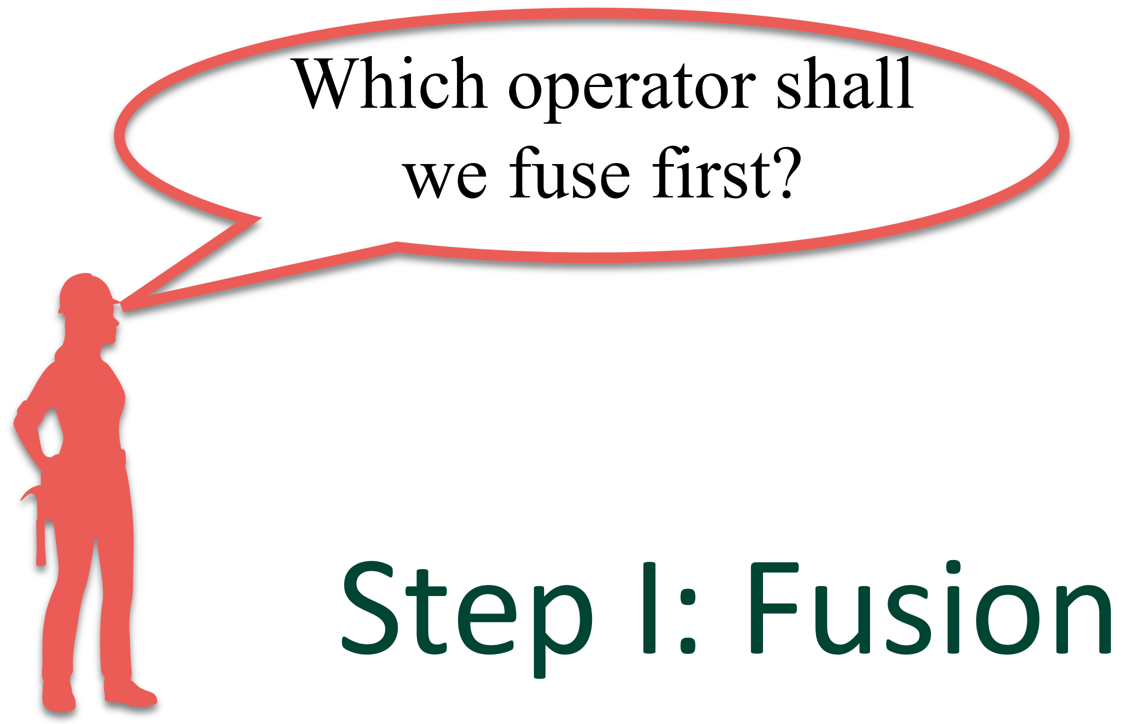
After: $(A \circledast B)^{-2} \odot C$

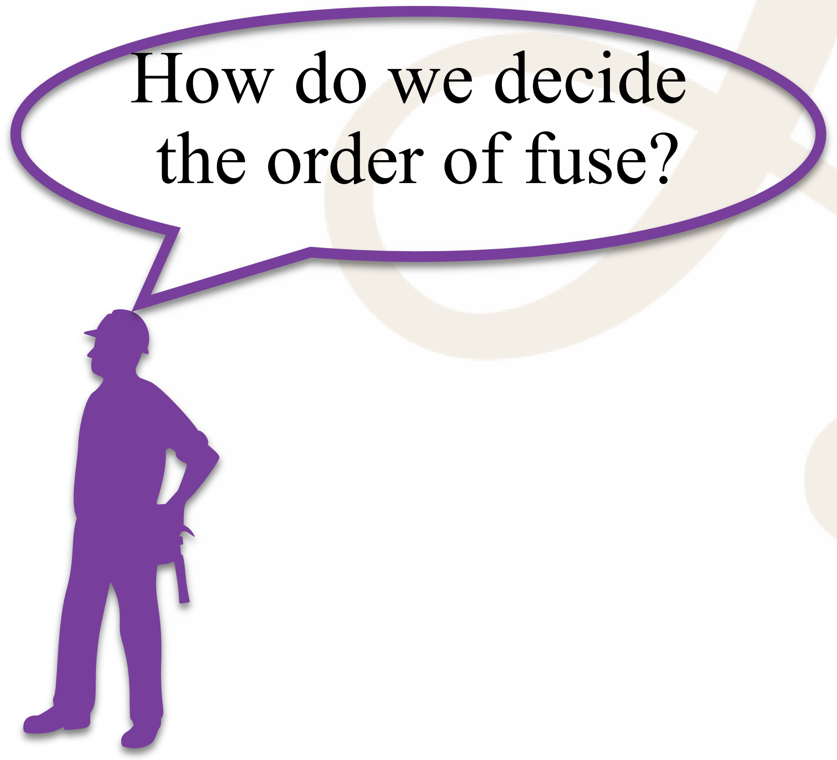
Mathematical-property-based graph rewriting

Property	Without graph rewriting		With graph rewriting	
	Graph structure in equation	Number of FLOPS	Graph structure in equation	Number of FLOPS
Associative	$(A \odot \sqrt{B}) \odot (\sqrt{B} \odot C)$	$5 * m * n$	$A \odot B \odot C$	$2 * m * n$
Distributive	$A \odot C + A \odot B$	$3 * m * n$	$(A + B) \odot C$	$2 * m * n$
Commutative	$ReduceProd(Exp(A))$	$2 * m * n$	$Exp(ReduceSum(A))$	$m * n + \mathrm{m}$

Heuristic fusion plan exploration

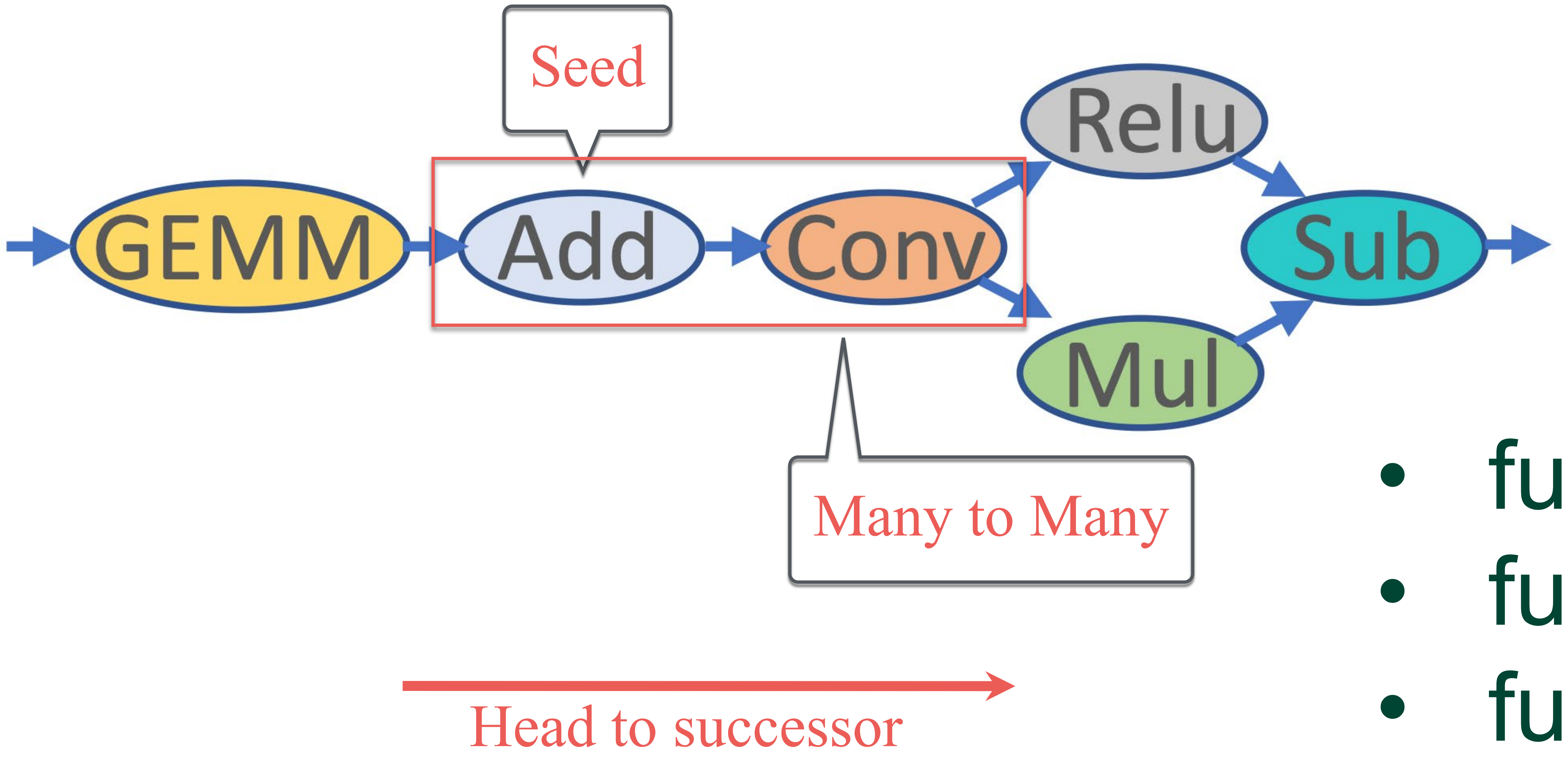






Step I: Fusion seed operator(s) selection.

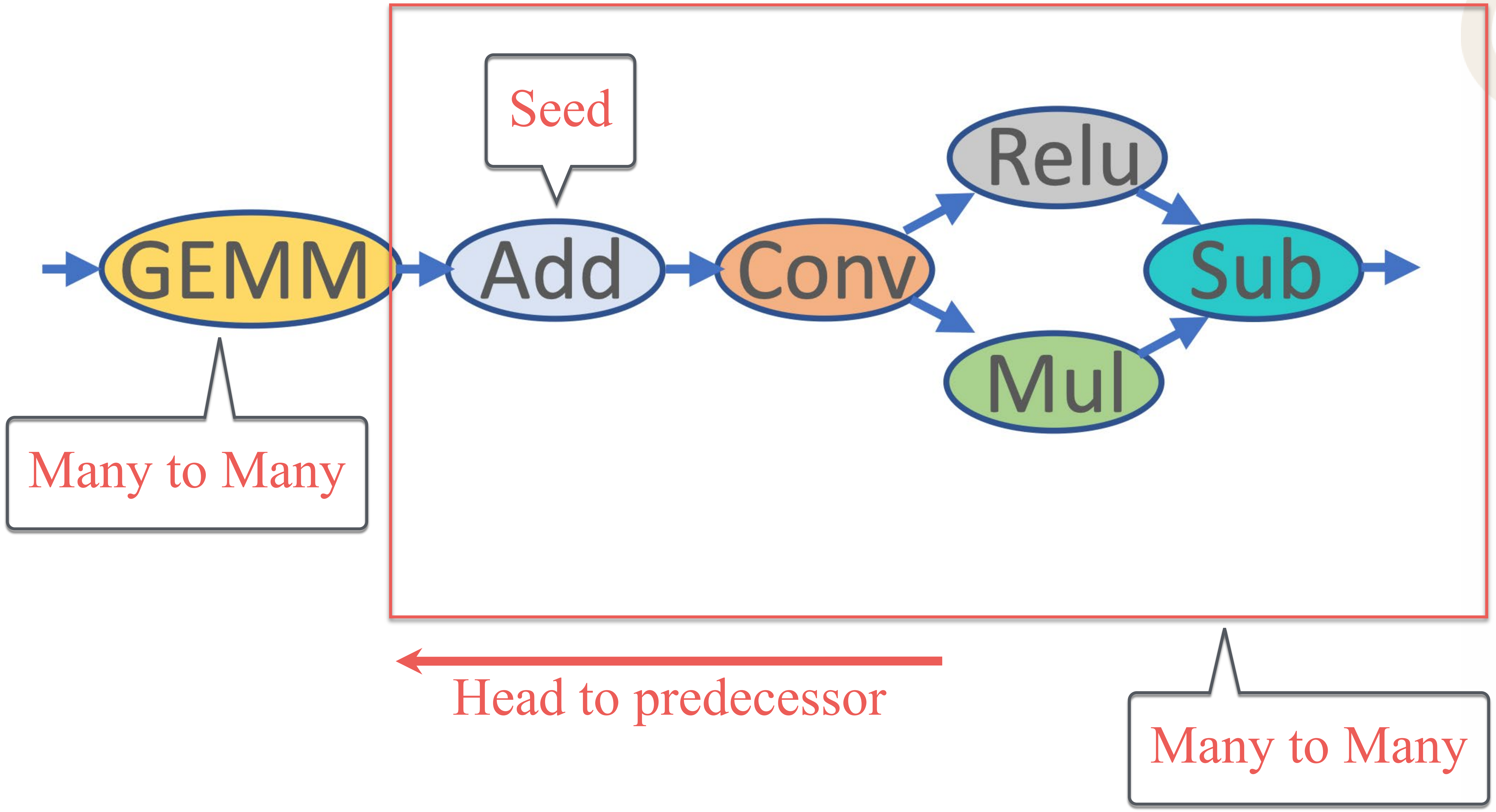
Heuristic fusion plan exploration



- fuse_break
- fuse_through
- fuse_depend

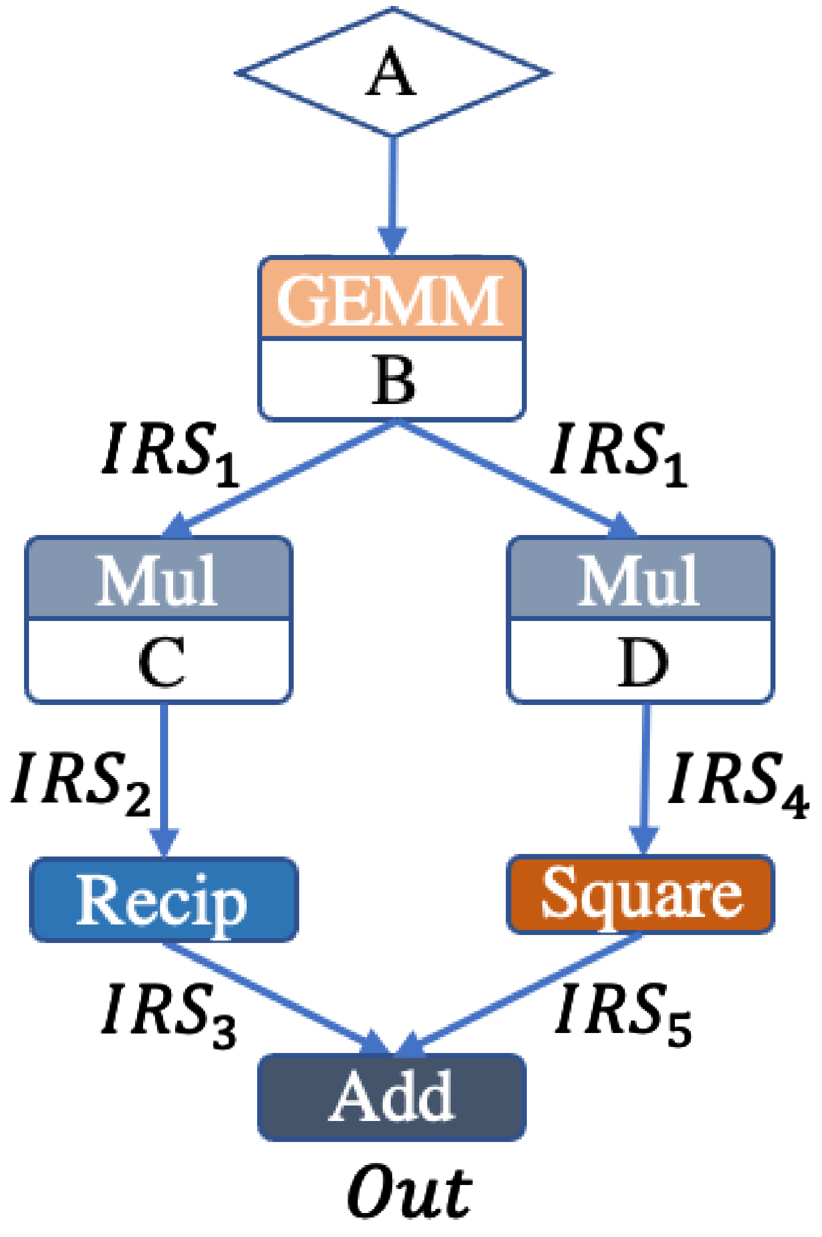
Step II: Propagated exploration along seed’s successors.

Heuristic fusion plan exploration



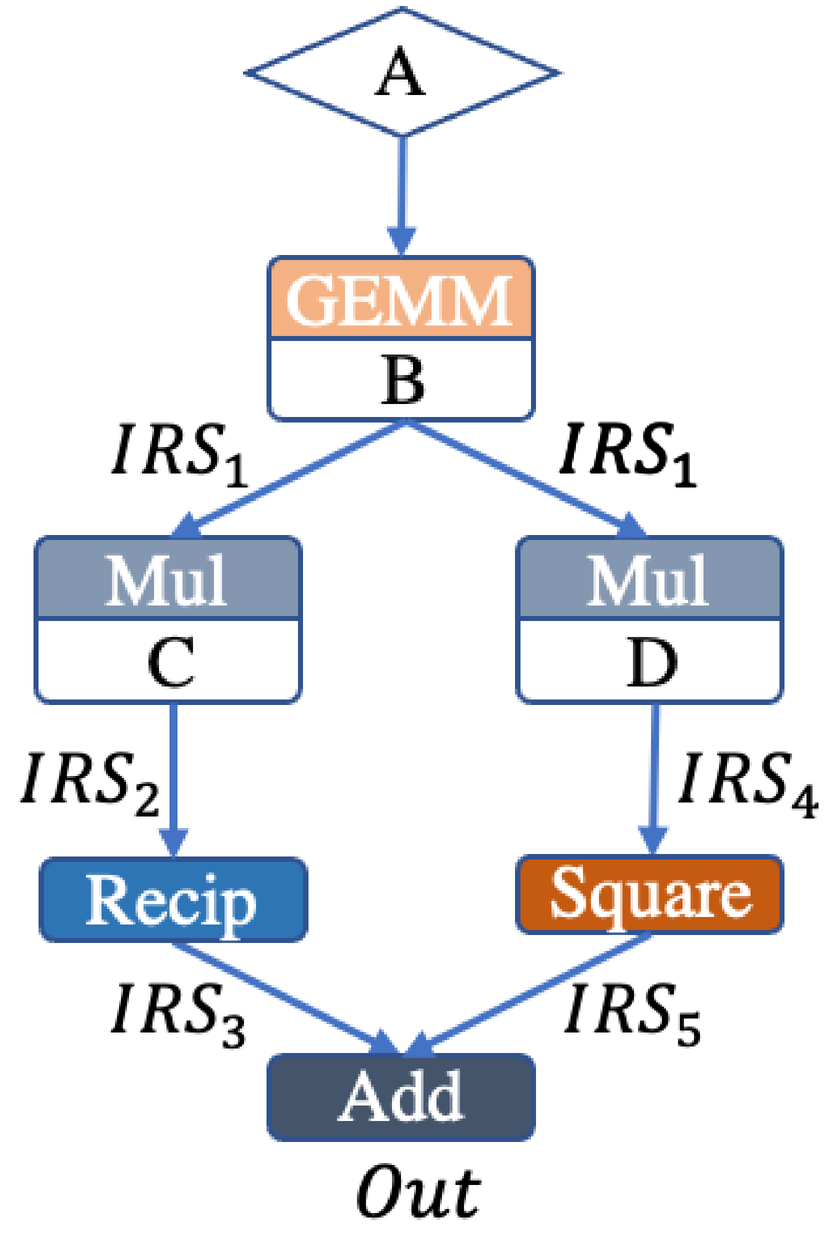
Step III: Propagated exploration along seed's predecessors.

Code generation

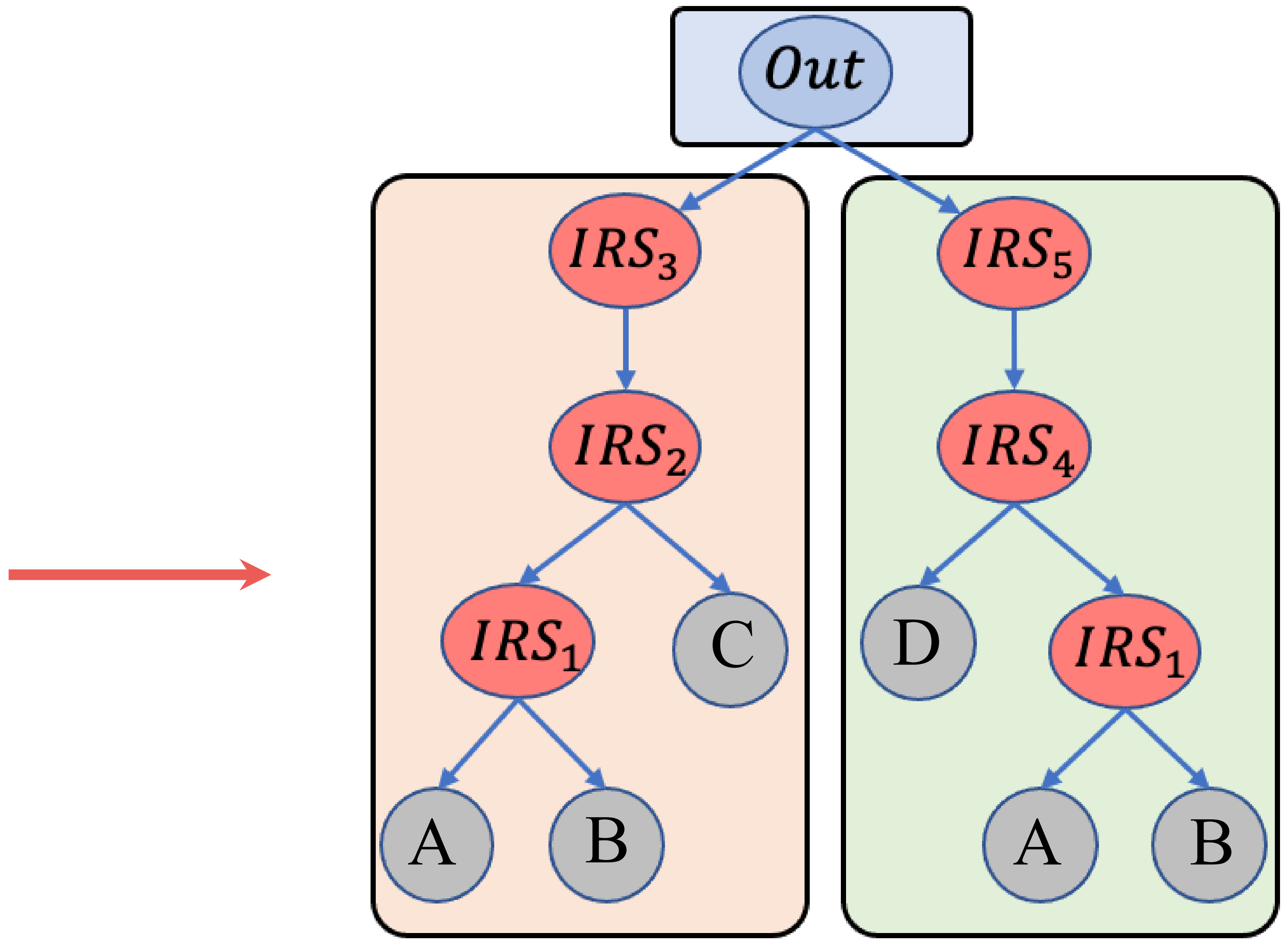


Extended computational graph

Code generation

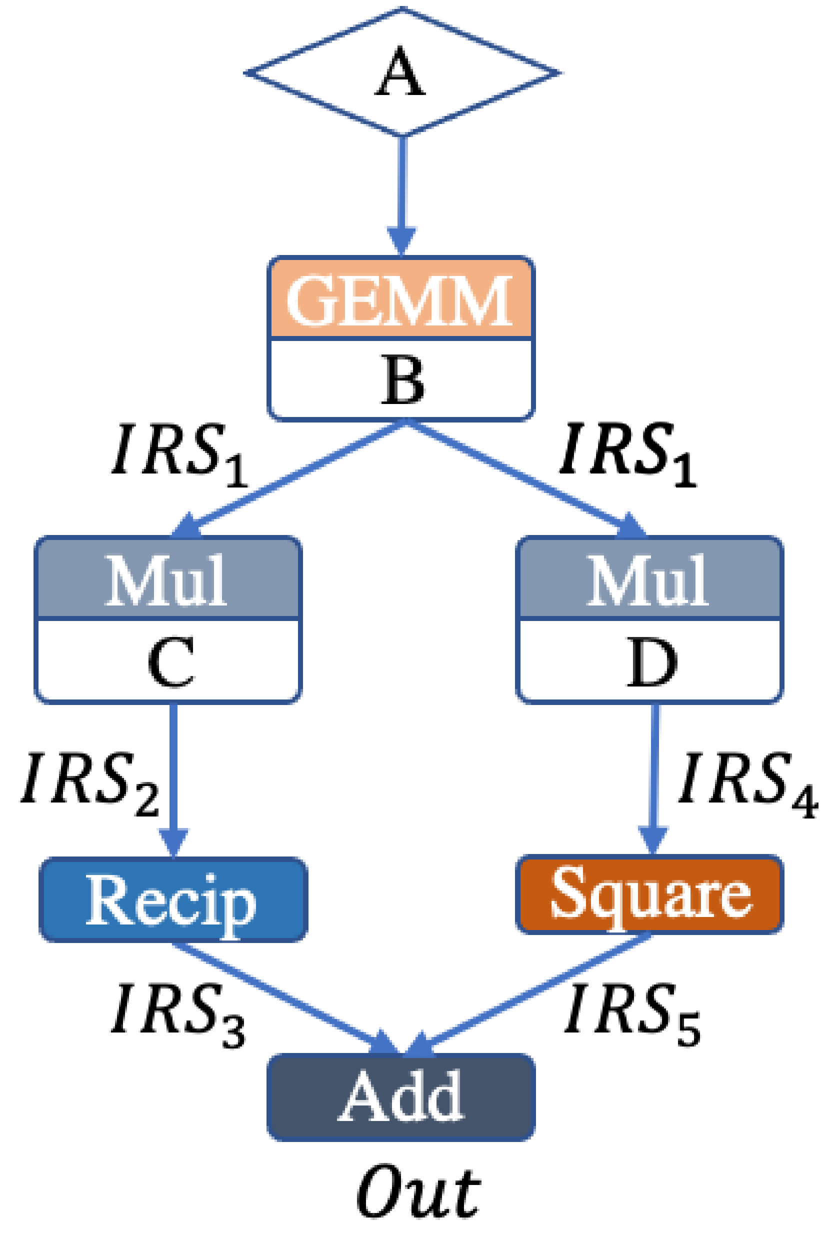


Extended computational graph

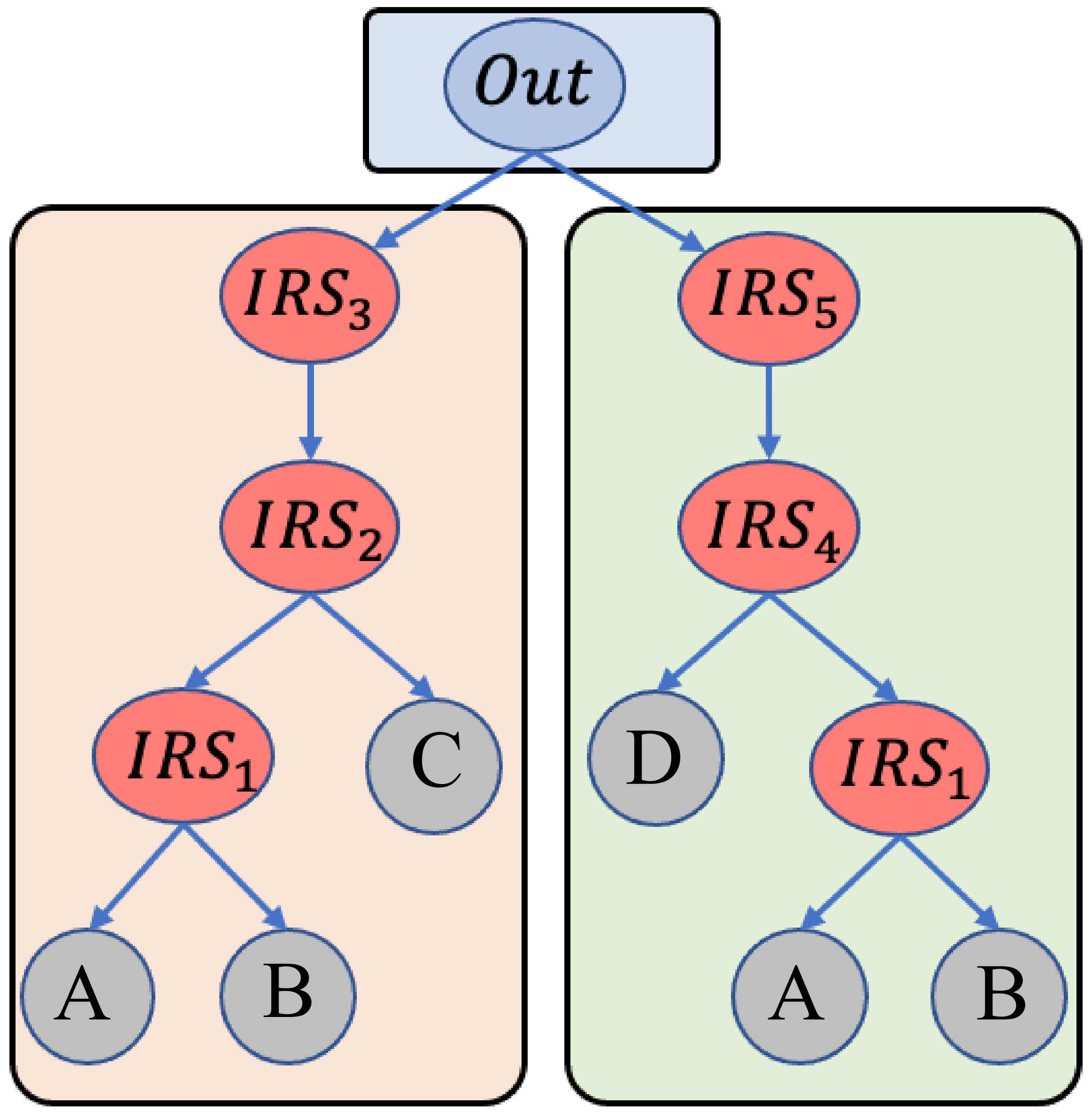


Data-flow tree

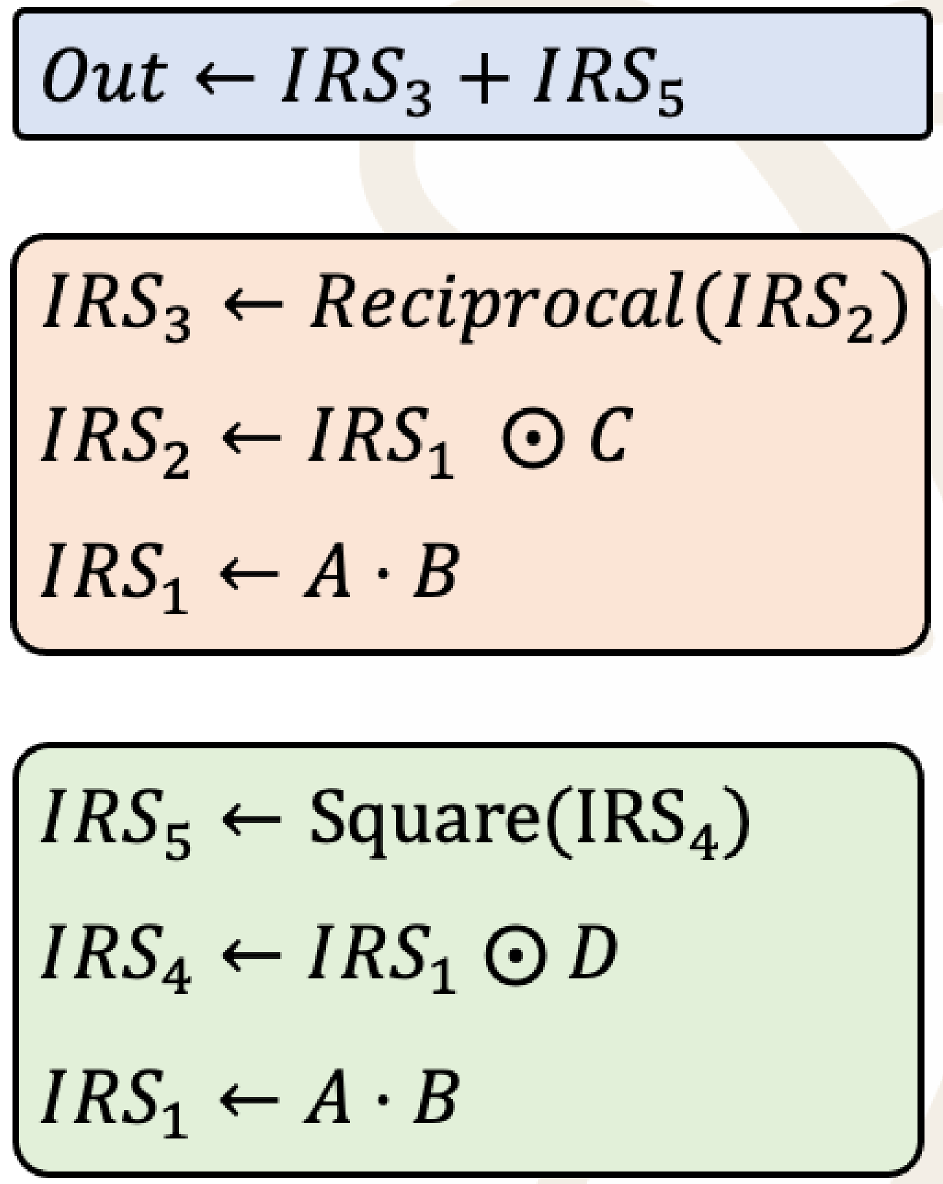
Code generation



Extended computational graph

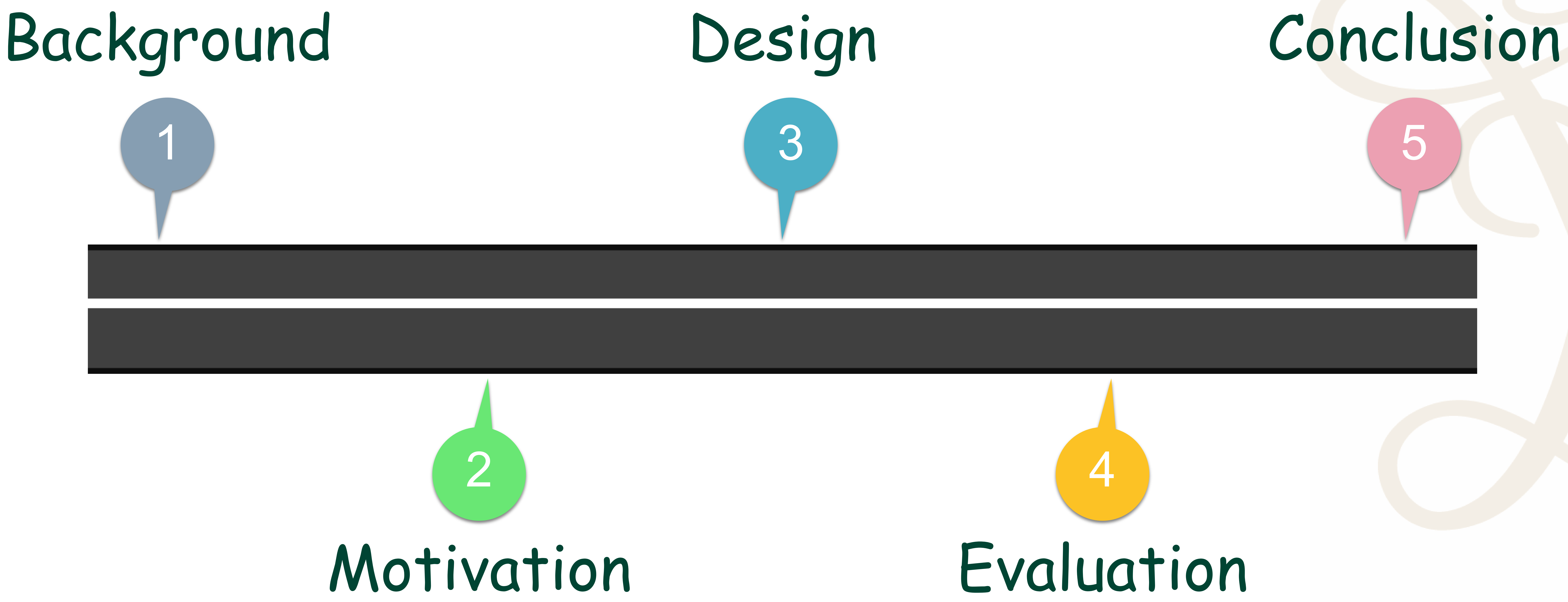


Data-flow tree



Code generation

Roadmap


Background
1
Design
3
Conclusion
5
2
Motivation
4
Evaluation

Evaluation setup

- Comparison frameworks
  1. MNN, TVM, TFLite, Pytorch-Mobile, TASO, Our baseline version
- Inference testing device
  1. Samsung Galaxy S20 (with Qualcomm Snapdragon 865 platform)
- Models
  1. 2D CNN: EfficientNet-B0, VGG-16, MobileNetV1-SSD, YOLO-V4, U-Net
  2. 3D CNN: C3D, S3D
  3. R-CNN: Faster R-CNN, Mask R-CNN,
  4. Transformer: TinyBERT, DistilBERT, ALBERT, BERT-Base, MobileBERT, GPT-2

Fusion rate evaluation

Up-to 8.8x fusion rate

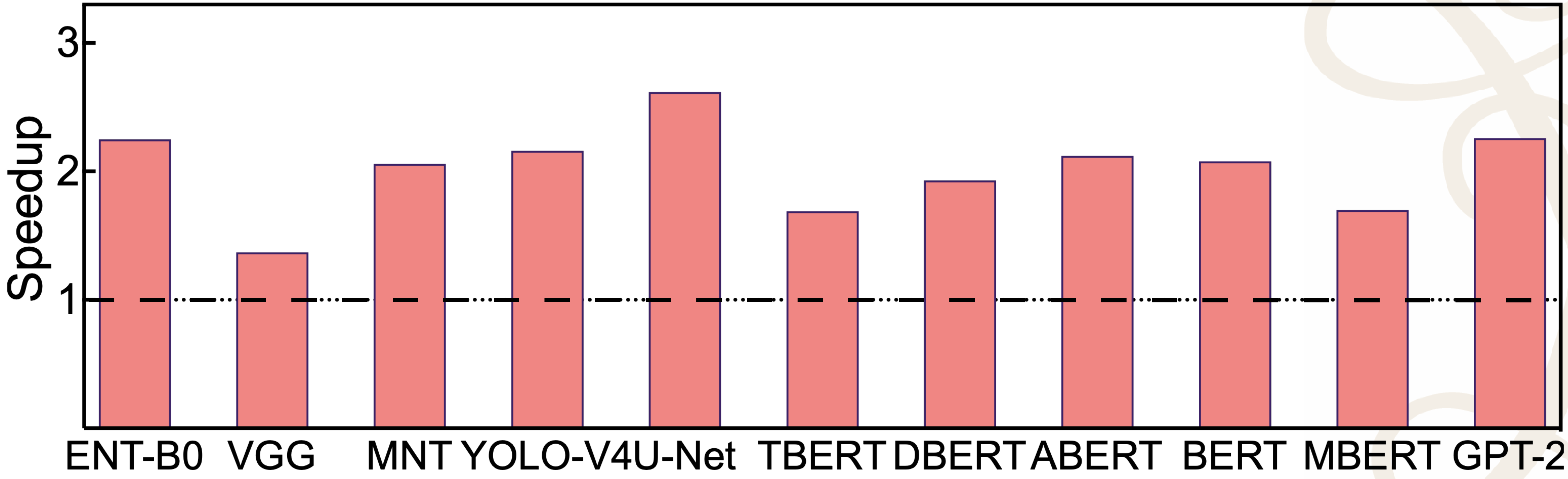
Models	Layer count before fusion	Layer count After fusion				
	Number of total layers	MNN	TVM	TFLite	Pytorch	DNNFusion
EfficientNet-B0	309	199	195	201	210	97
VGG-16	51	22	22	22	22	17
MobileNetV1-SSD	202	138	124	138	148	71
YOLO-V4	398	198	192	198	232	135
C3D	27	27	27	-	27	16
S3D	272	-	-	-	272	98
U-Net	292	241	232	234	-	82
FasterR-CNN	3,640	-	-	-	-	942
MaskR-CNN	3,999	-	-	-	-	981
TinyBERT	366	-	304	322	-	74
DistilBERT	457	-	416	431	-	109
ALBERT	936	-	746	855	-	225
BERT-BASE	976	-	760	873	-	216
MobileBERT	2,387	-	1,678	2,128	-	510
GPT-2	2,533	-	2,047	2,223	-	254

Inference latency comparison

Up-to 9.3x speedup

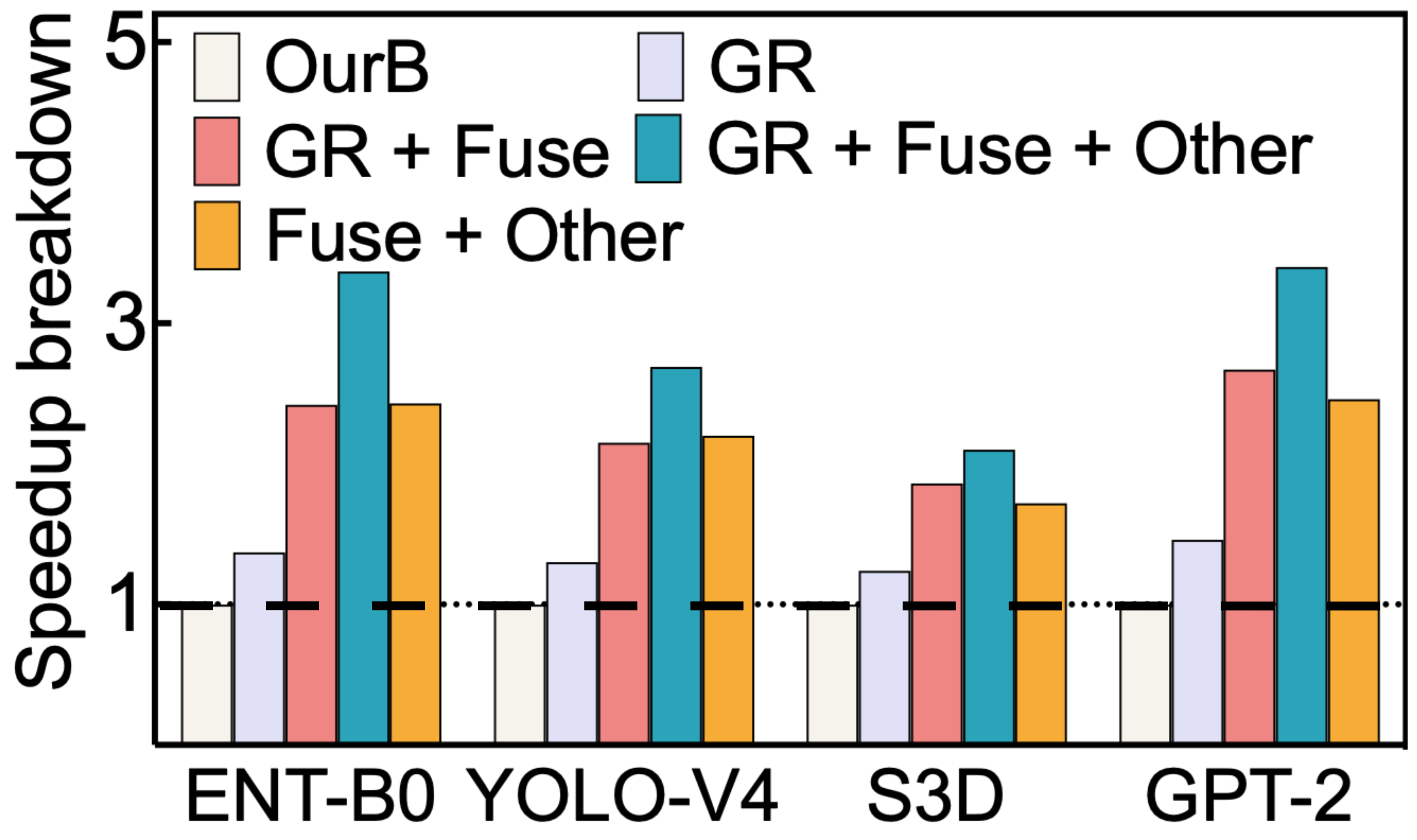
Models	MNN (ms)		TVM (ms)		TFLite (ms)		Pytorch (ms)		OurB		OurB+		DNNFusion (ms)	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
EfficientNet-B0	41	26	56	27	52	30	76	-	54	35	38	24	16	10
VGG-16	242	109	260	127	245	102	273	-	251	121	231	97	171	65
MobileNetV1-SSD	67	43	74	52	87	68	92	-	79	56	61	39	33	17
YOLO-V4	501	290	549	350	560	288	872	-	633	390	426	257	235	117
C3D	867	-	1,487	-	-	-	2,541	-	880	551	802	488	582	301
S3D	-	-	-	-	-	-	6,612	-	1,409	972	1,279	705	710	324
U-Net	181	106	210	120	302	117	271	-	227	142	168	92	99	52
FasterR-CNN	-	-	-	-	-	-	-	-	2,325	3,054	1,462	1,974	862	531
MaskR-CNN	-	-	-	-	-	-	-	-	5,539	6,483	3,907	4,768	2,471	1,680
TinyBERT	-	-	-	-	97	-	-	-	114	89	92	65	51	30
DistilBERT	-	-	-	-	510	-	-	-	573	504	467	457	224	148
ALBERT	-	-	-	-	974	-	-	-	1,033	1,178	923	973	386	312
BERT-BASE	-	-	-	-	985	-	-	-	1,086	1,204	948	1,012	394	293
MobileBERT	-	-	-	-	342	-	-	-	448	563	326	397	170	102
GPT-2	-	-	-	-	1,102	-	-	-	1,350	1,467	990	1,106	394	292

DNNFusion VS TASO

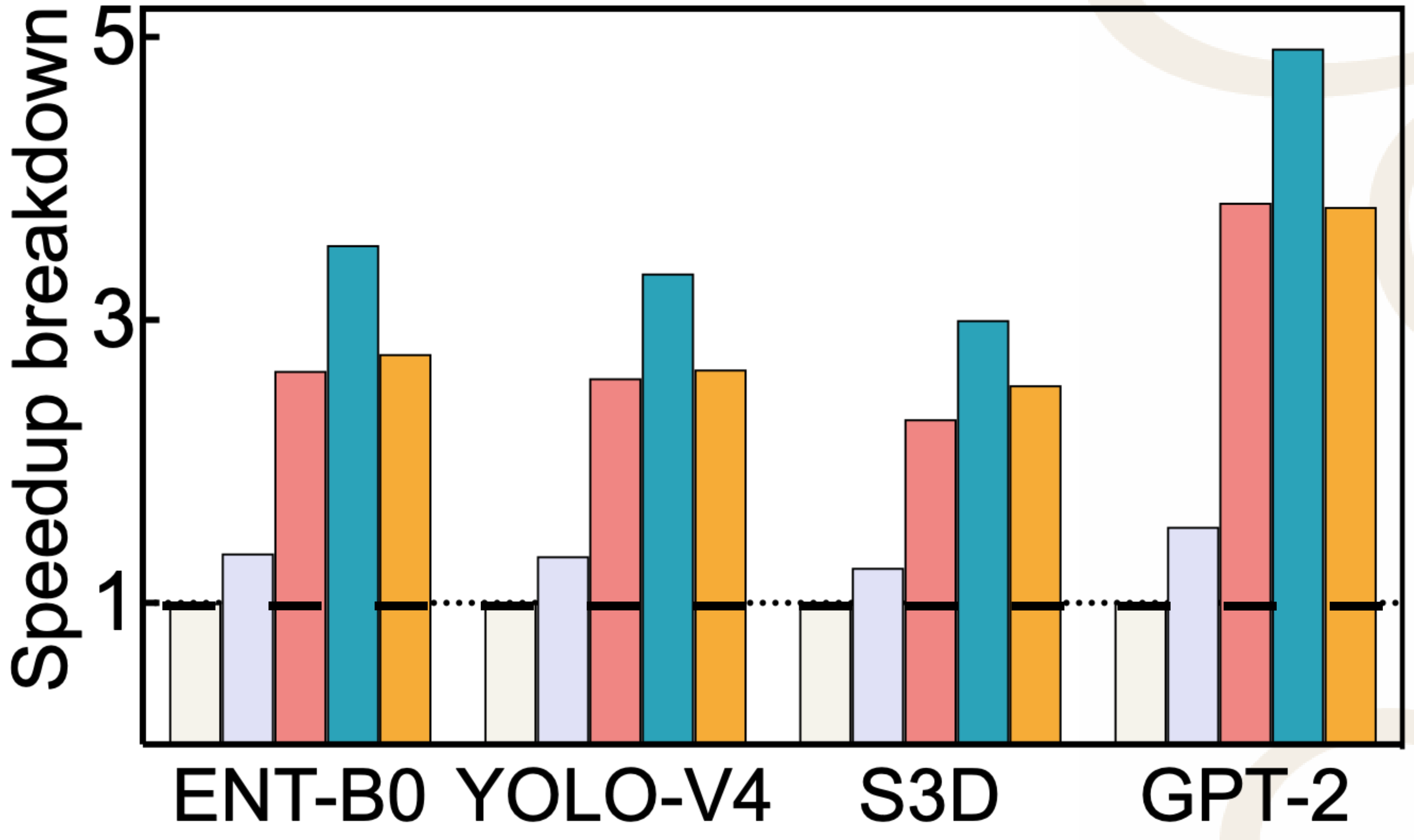


Baseline: TFLite
Devices: CPU

DNNFusion optimizations breakdown



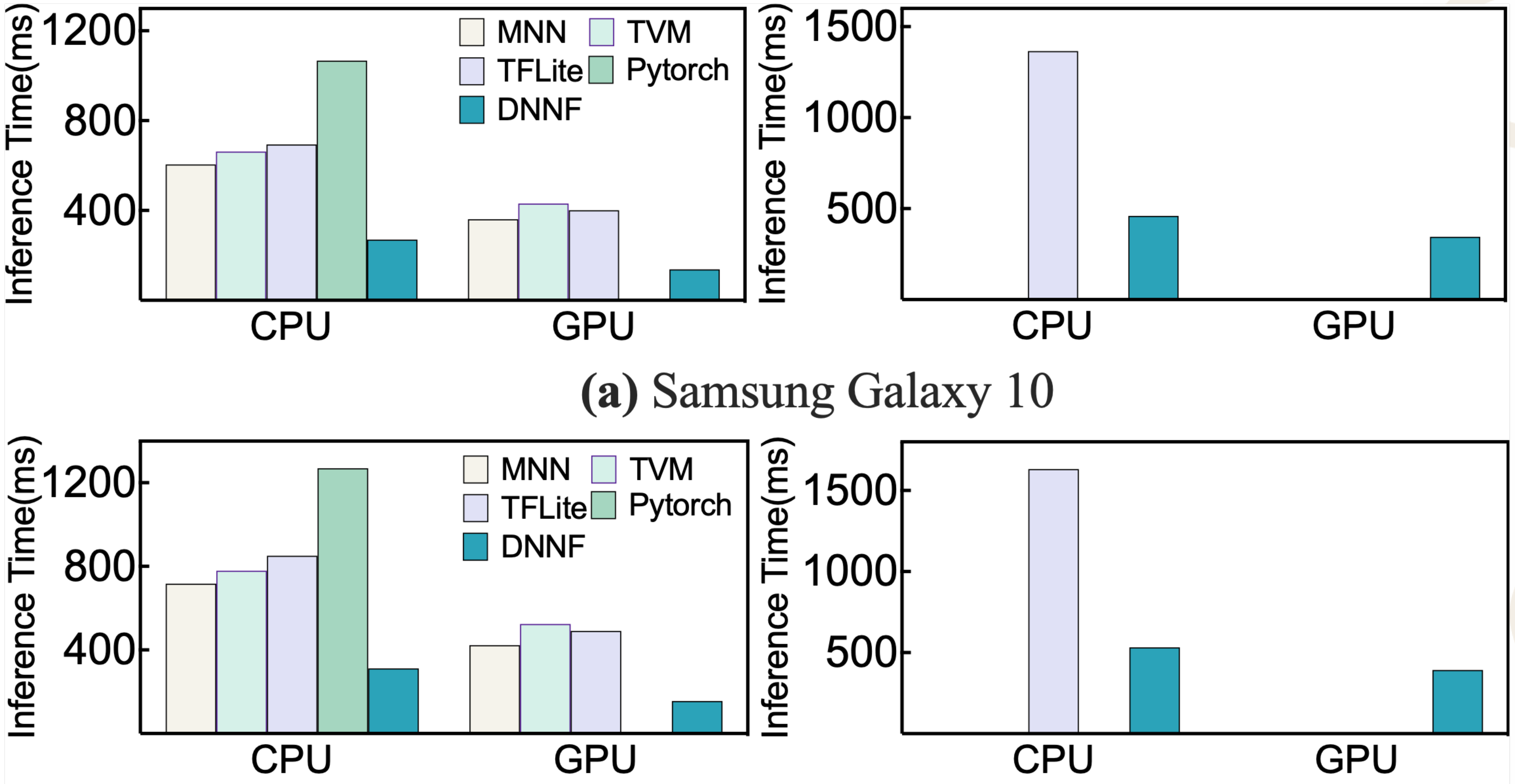
(a) CPU.



(b) GPU.

Fusion contributes most

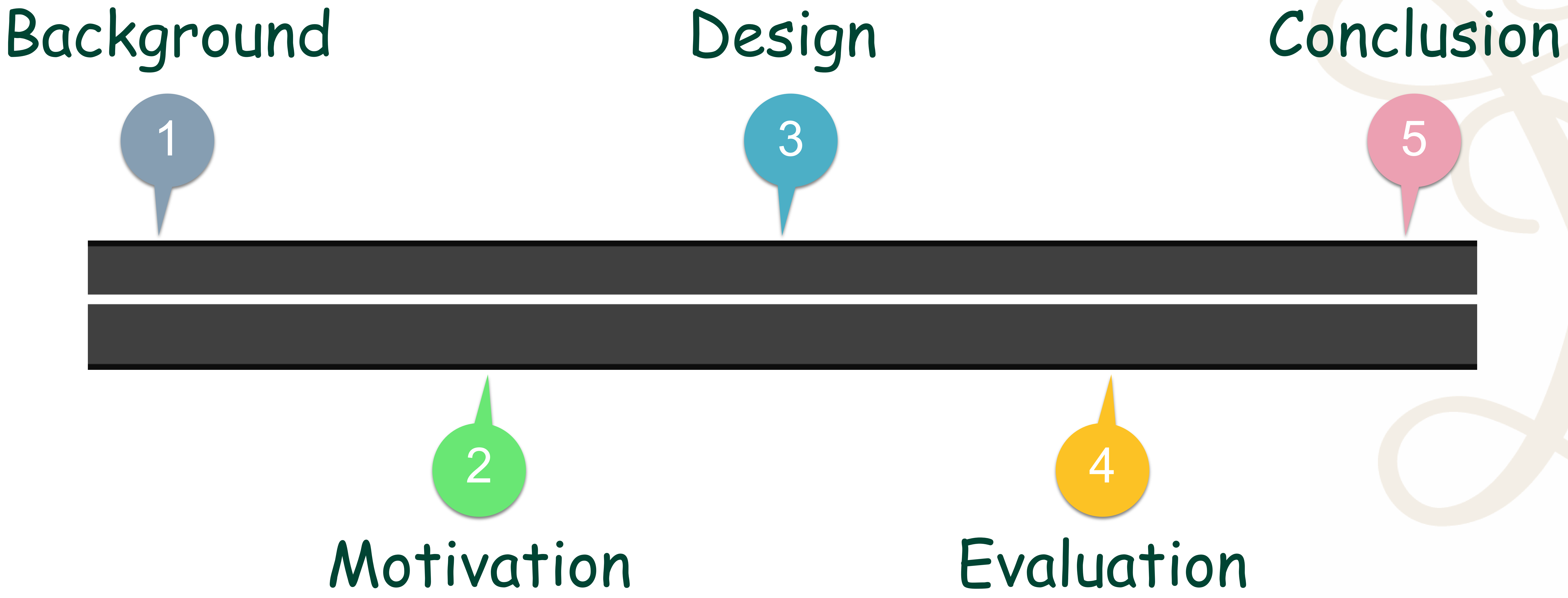
Portability evaluation



(a) Samsung Galaxy 10

(b) Honor Magic 2

Roadmap



Background
1
Design
3
Conclusion
5
2
Motivation
4
Evaluation

Conclusion

- Designs high-level abstractions for operator fusion by leveraging high-level DNN operator information
- Proposes a novel mathematical-property-based graph rewriting to simplify ECG structure, and enable more fusion plan
- Evaluates 15 cutting-edge DNN models with varied types of tasks, model size, and layer counts
- Outperforms four state-of-the-art DNN execution frameworks with up to 9.3x speedup, and allows many latest DNN models that are not supported by any existing end-to-end frameworks to run on mobile devices efficiently, even in real-time.

Thanks

Wei Niu: wniu@email.wm.edu

WILLIAM & MARY
CHARTERED 1693

AUGUSTA
UNIVERSITY

Northeastern
University