Ptolemy: Architecture Support for Robust Deep Learning

Yiming Gan
Department of Computer Science,
University of Rochester

with
Yuxian Qiu, Shanghai Jiao Tong University
Jingwen Leng, Shanghai Jiao Tong University
Minyi Guo, Shanghai Jiao Tong University
Yuhao Zhu University of Rochester

https://github.com/Ptolemy-dl/Ptolemy

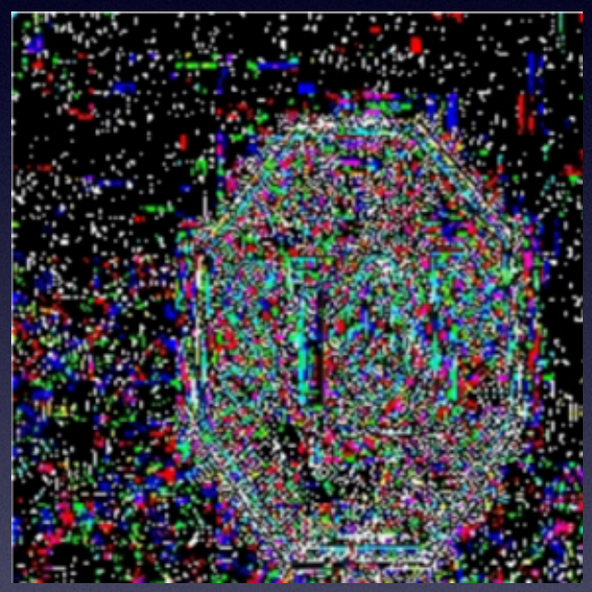
Deep Learning: Not Robust

Deep Learning: Not Robust

Legitimate Example

+

Perturbation

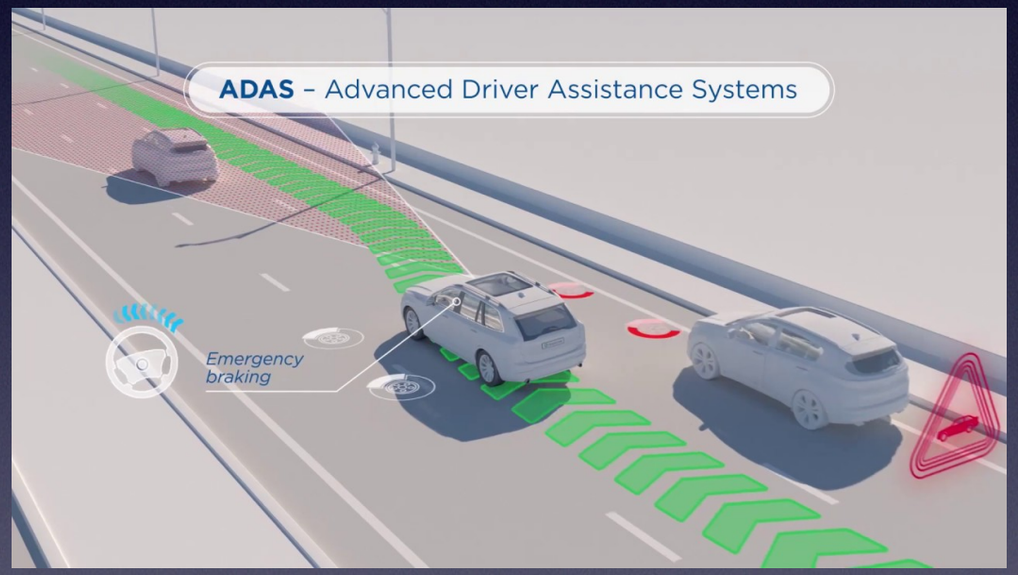


=

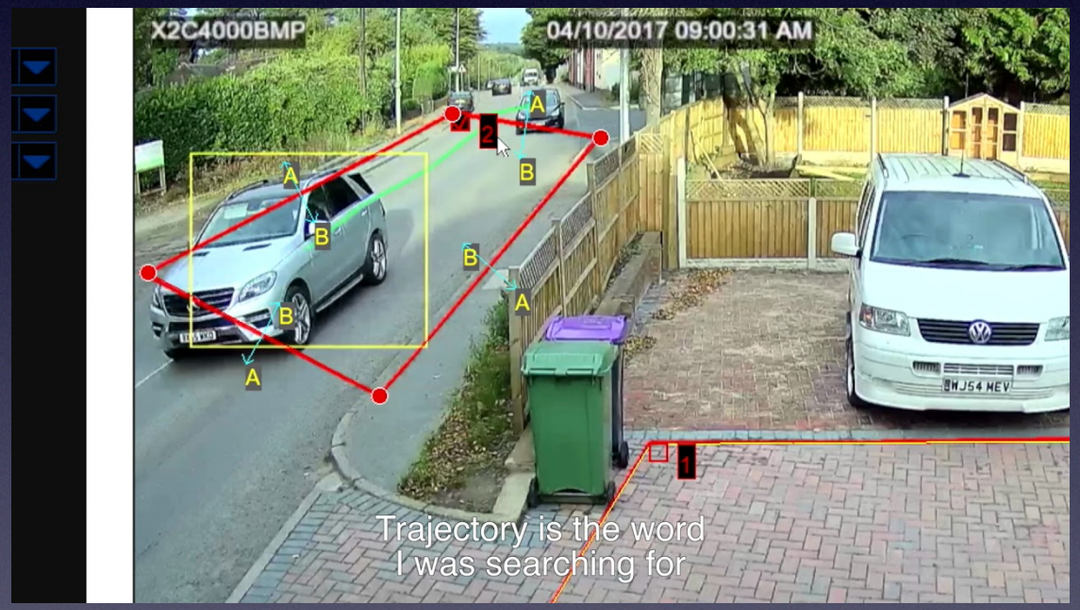
Adversarial Example

Mission Critical System

ADAS



Security Cameras



Robust Deep Learning Requirements

- Accurately detect adversarial examples

Robust Deep Learning Requirements

- Accurately detect adversarial examples

+

- Do not bring large overhead on system performance

Robust Deep Learning Requirements

- Accurately detect adversarial examples

+

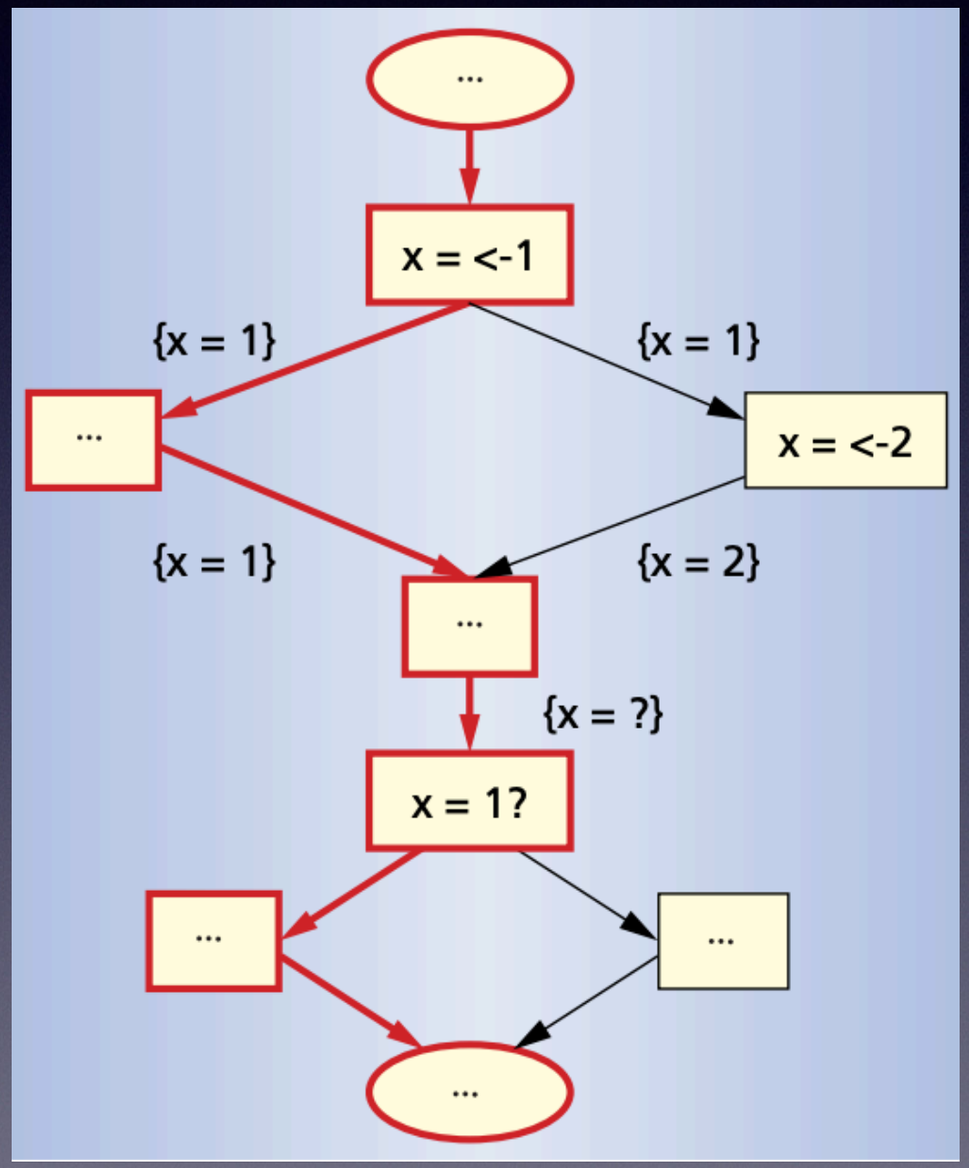
- Do not bring large overhead on system performance

=

Ptolemy

Hot Path

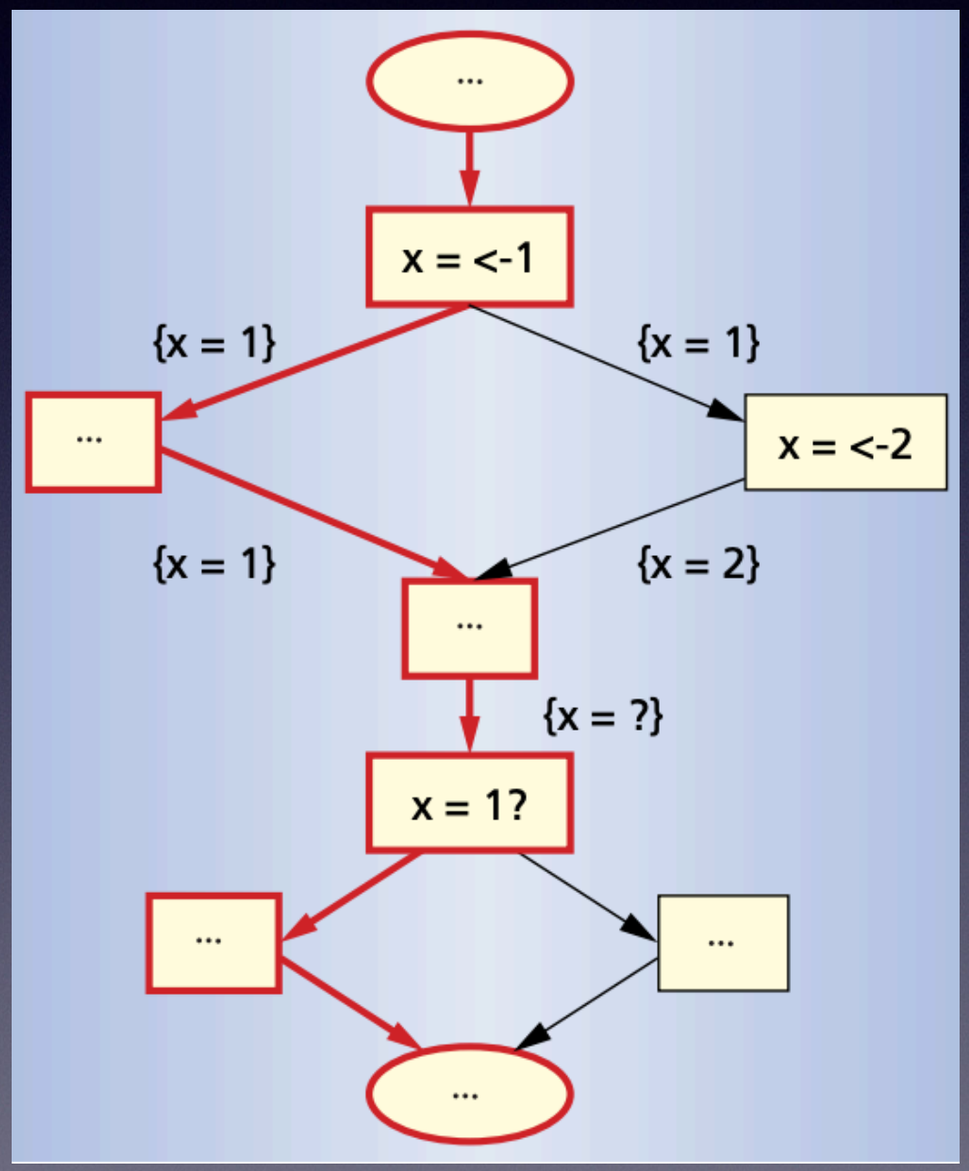
Traditional Software



[1]Thomas Ball, James R. Larus, Using Paths to Measure, Explain, and Enhance Program Behavior

Hot Path

Traditional Software

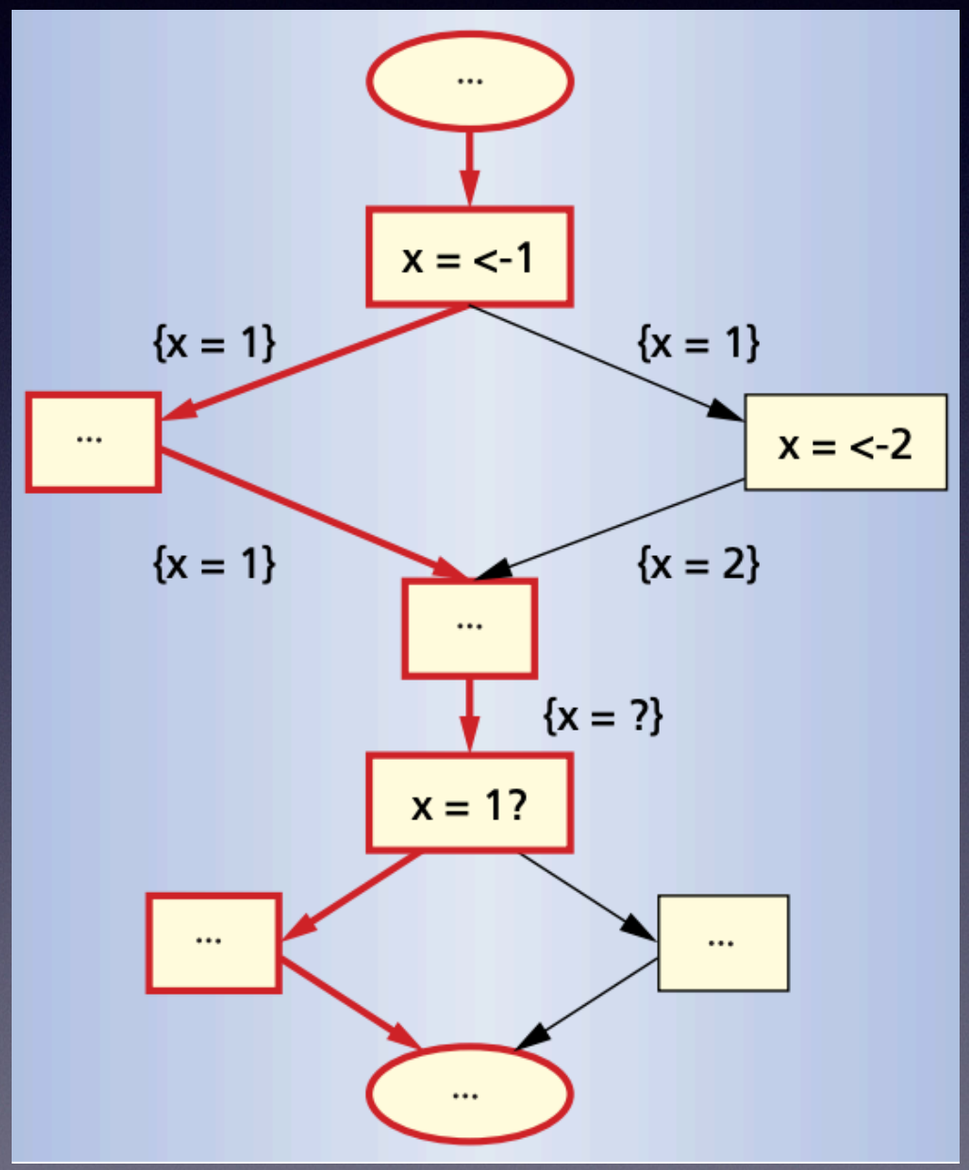


- Measure Program Behavior
- Optimizing Program
- Debugging

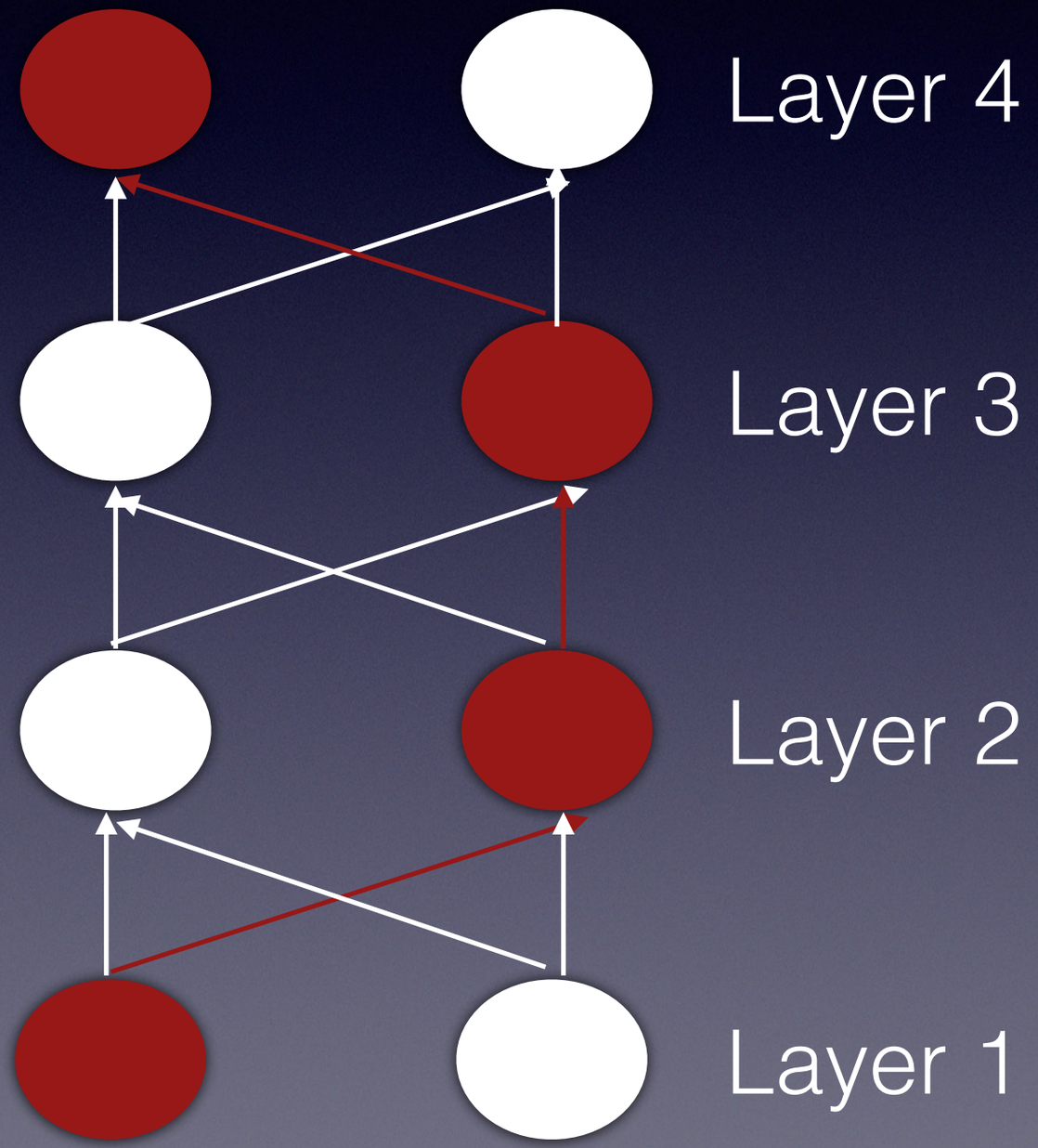
[1]Thomas Ball, James R. Larus, Using Paths to Measure, Explain, and Enhance Program Behavior

Hot Path

Traditional Software



Deep Learning



Defining Important Neuron

Input Feature Map

0.3	0.4	0.2	1.0	0.1

x

Weights

0.2	0.2	0.3
0.3	0.2	0.4
0.4	0.1	0.2
-0.1	0.09	0.1
-1.0	2.1	0.5

=

Output Feature Map

0.06
0.46
0.44

Defining Important Neuron

Input Feature Map

0.3	0.4	0.2	1.0	0.1

x

Weights

0.2	0.2	0.3
0.3	0.2	0.4
0.4	0.1	0.2
-0.1	0.09	0.1
-1.0	2.1	0.5

=

Output Feature Map

0.06
0.46
0.44

Defining Important Neuron

Input Feature Map

0.3	0.4	0.2	1.0	0.1

x

Weights

0.2	0.2	0.3
0.3	0.2	0.4
0.4	0.1	0.2
-0.1	0.09	0.1
-1.0	2.1	0.5

=

Output Feature Map

0.06
0.46
0.44

Defining Important Neuron

Input Feature Map

0.3	0.4	0.2	**1.0**	**0.1**

x

Weights

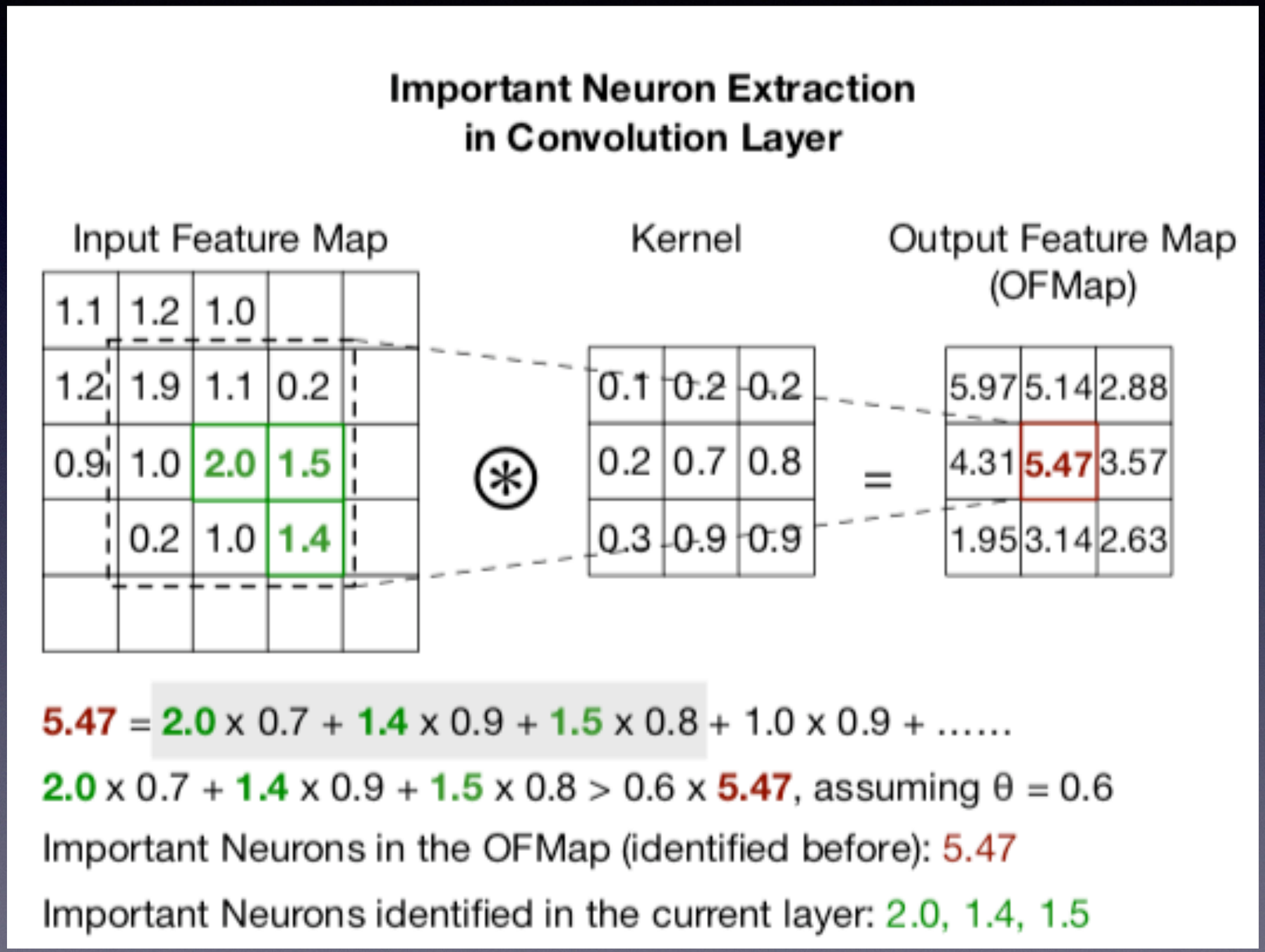
0.2	0.2	0.3
0.3	0.2	0.4
0.4	0.1	0.2
-0.1	**0.09**	0.1
-1.0	**2.1**	0.5

=

Output Feature Map

0.06
0.46
0.44

Defining Important Neuron

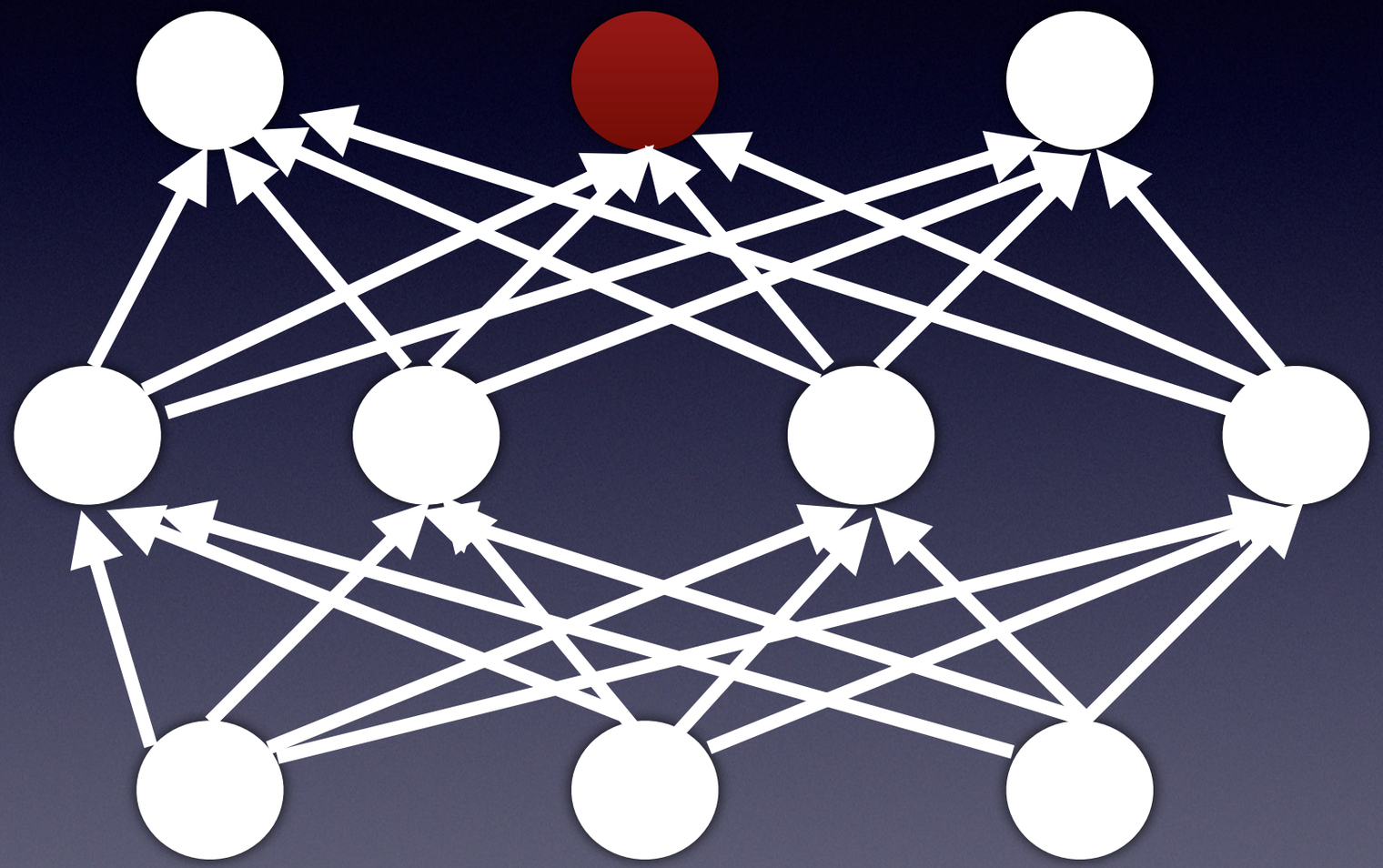


From Neuron to Path

Output Layer

Hidden Layer

Input Layer

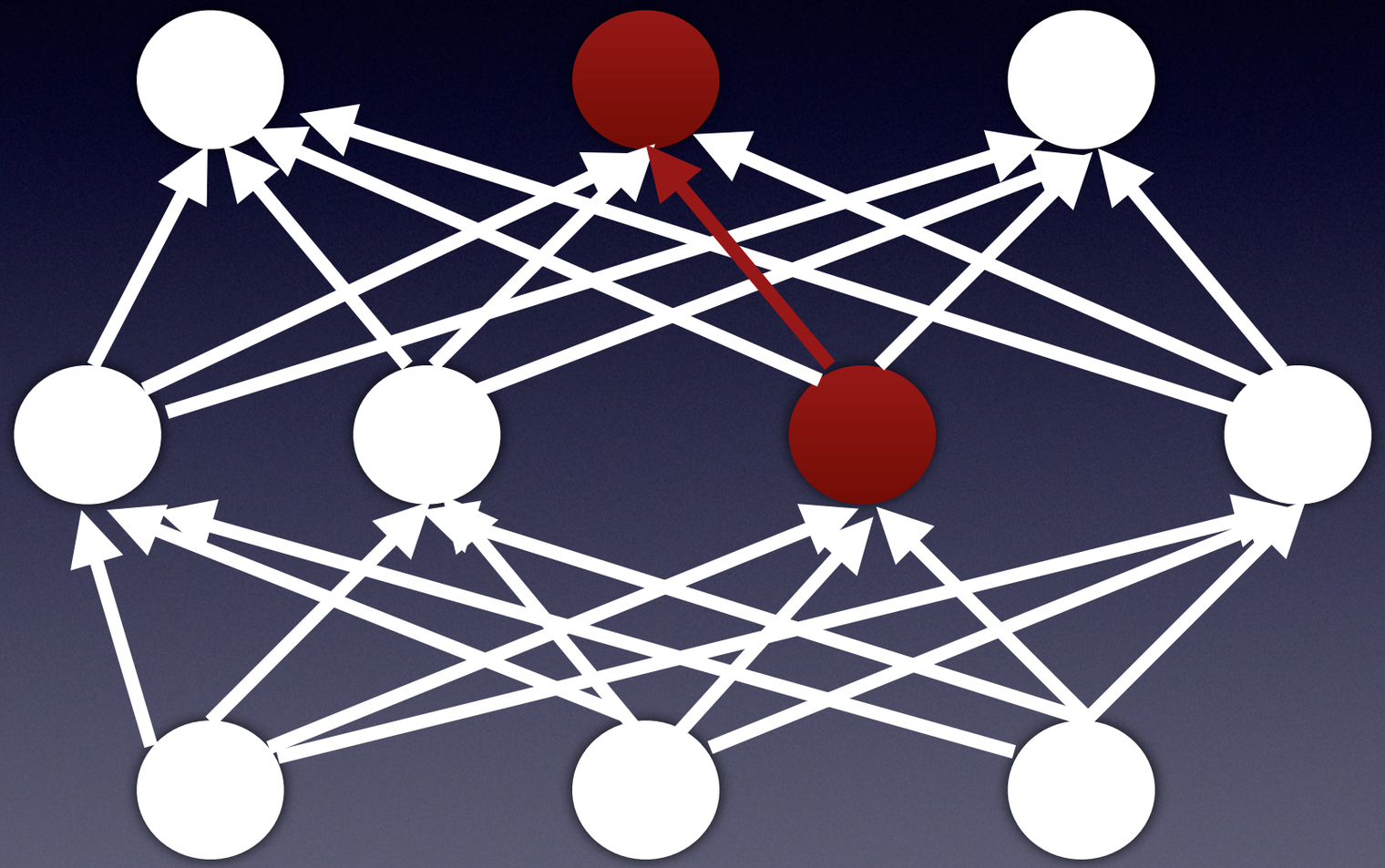


From Neuron to Path

Output Layer

Hidden Layer

Input Layer

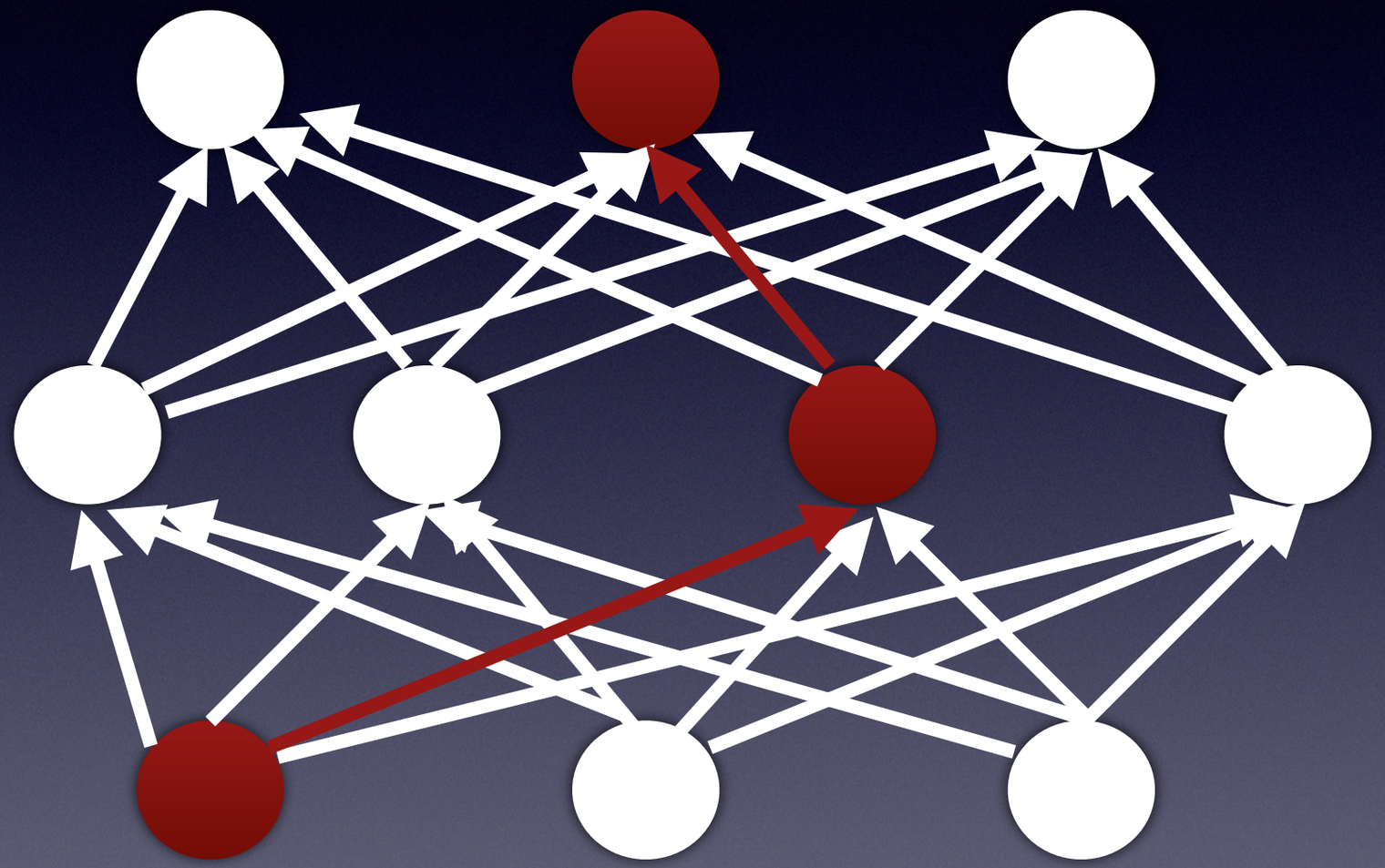


From Neuron to Path

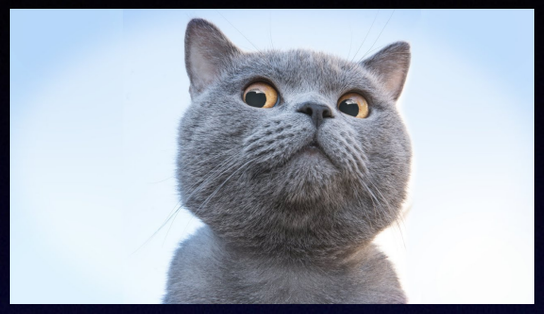
Output Layer

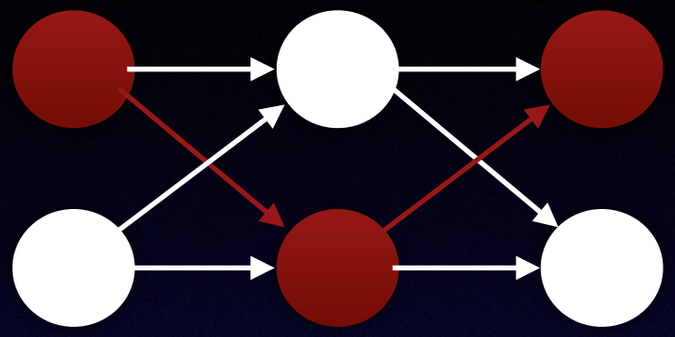
Hidden Layer

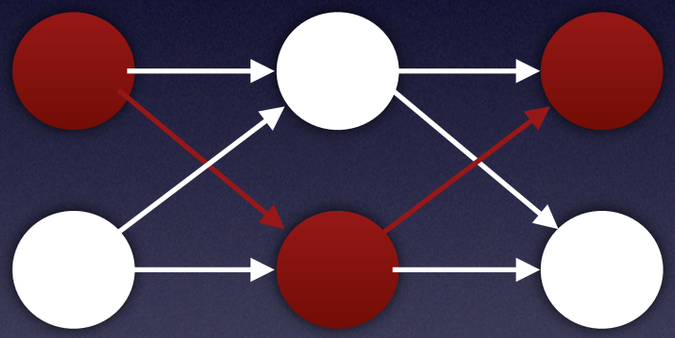
Input Layer

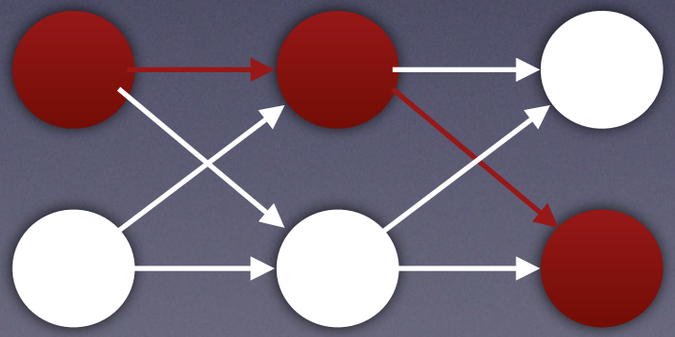


Class Path

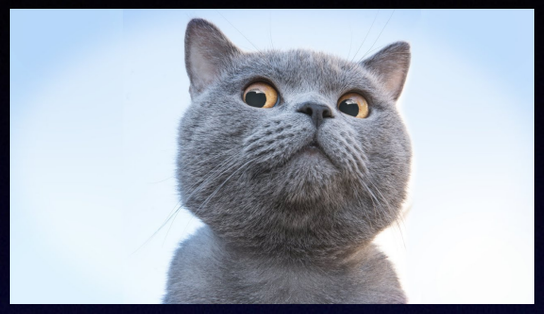


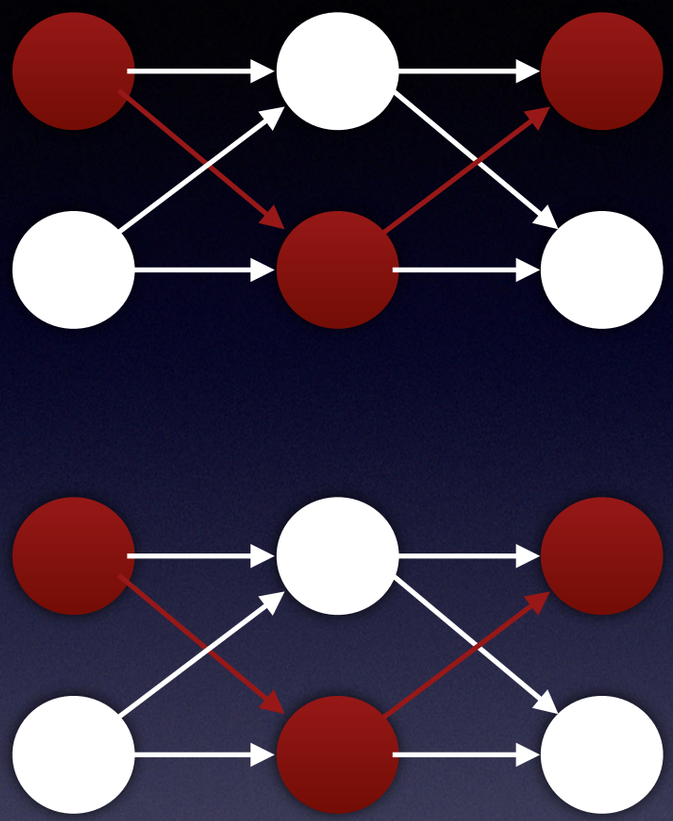


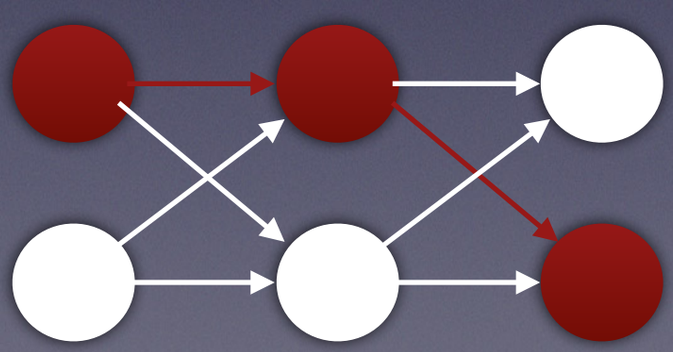




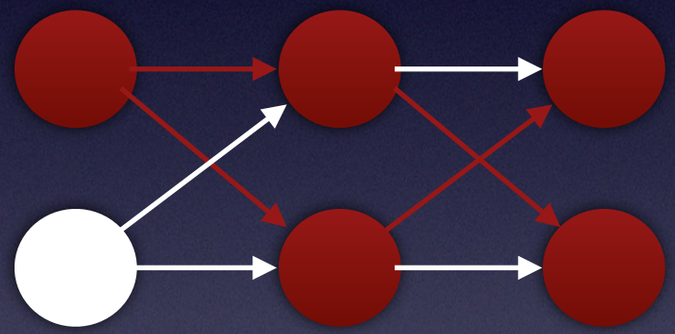
Class Path



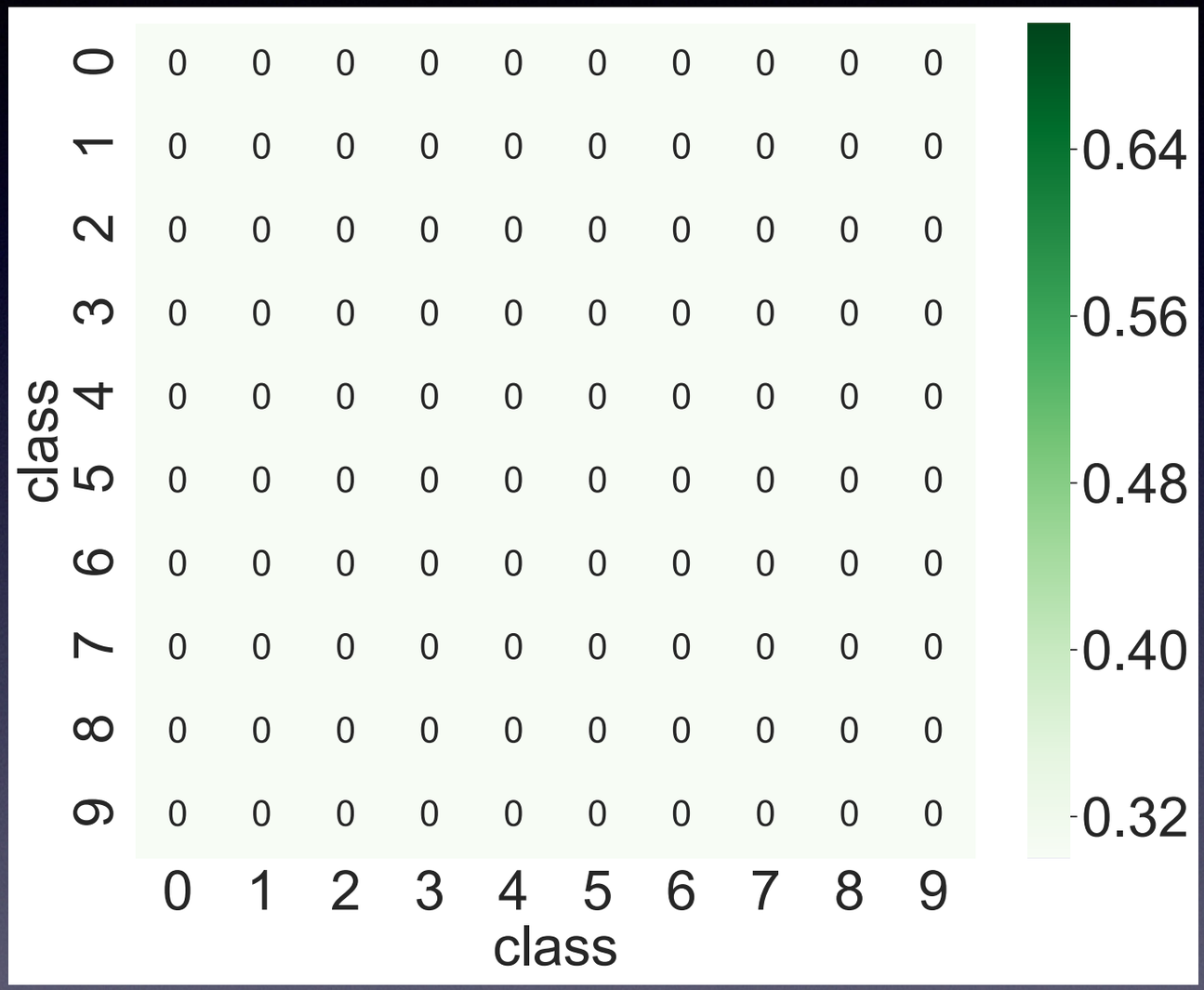




Union

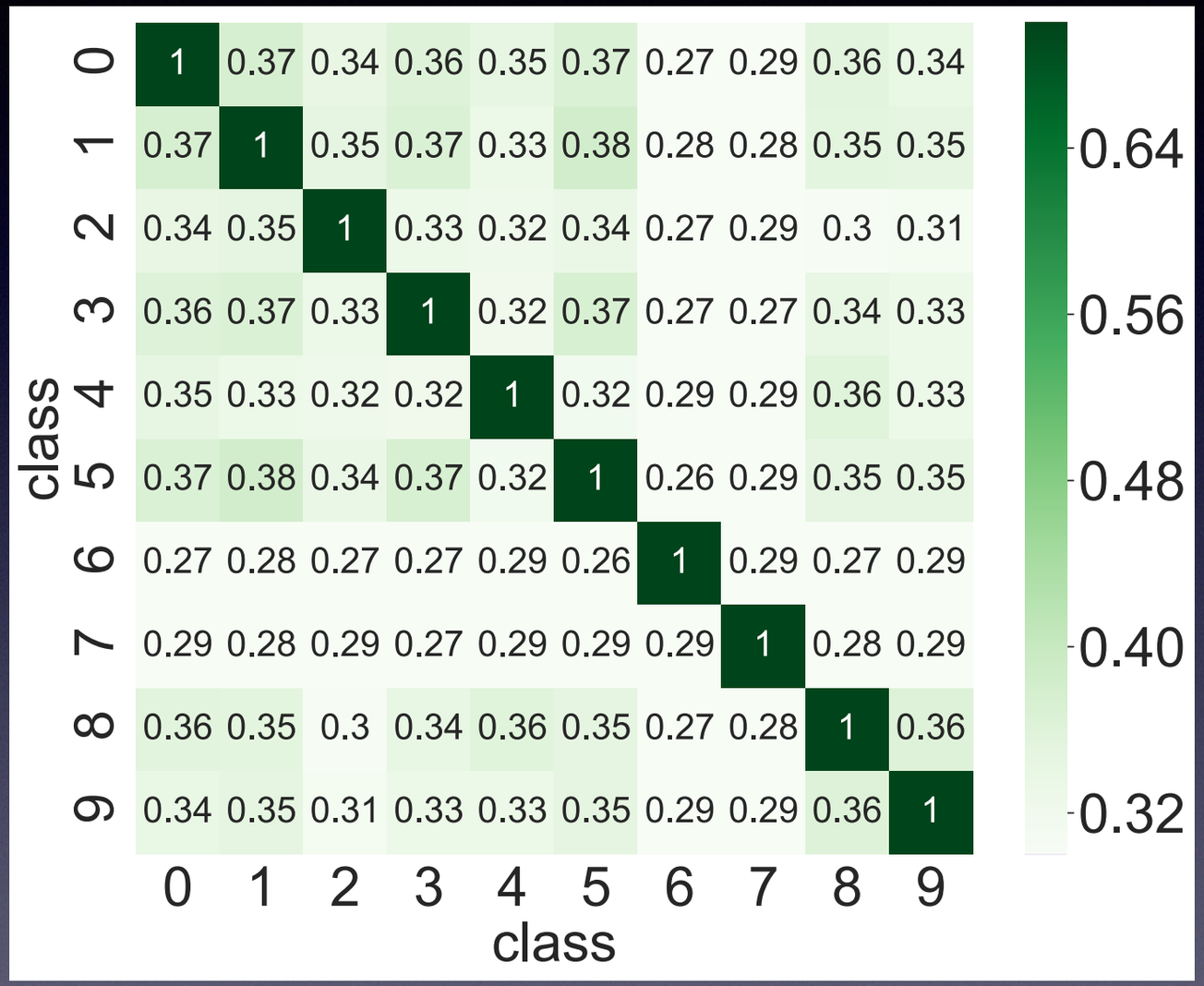


Class Path Similarity



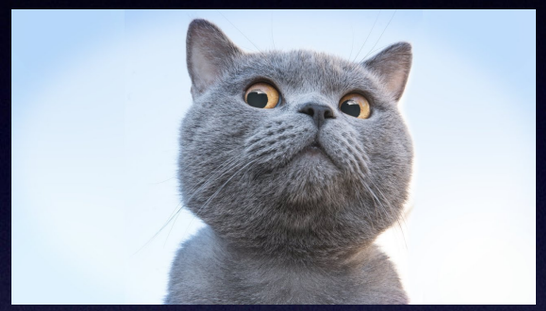
AlexNet @ ImageNet

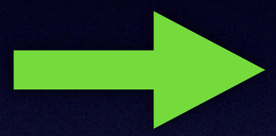
Class Path Similarity



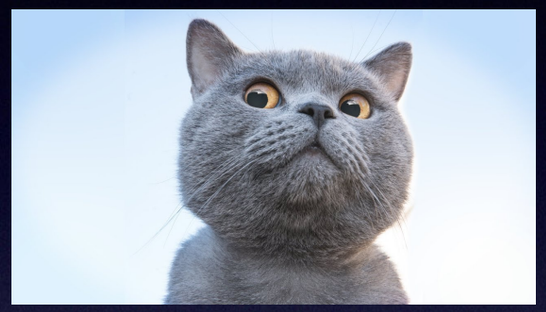
AlexNet @ ImageNet

Ptolemy Overview





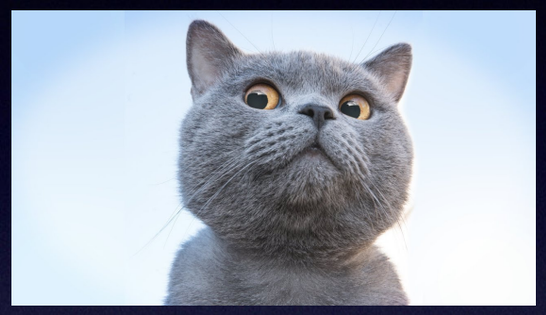
Ptolemy Overview

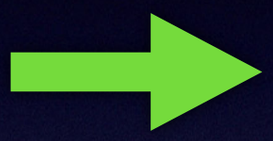


Neural Networks

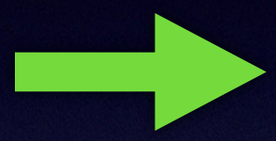
“Cat”

Ptolemy Overview



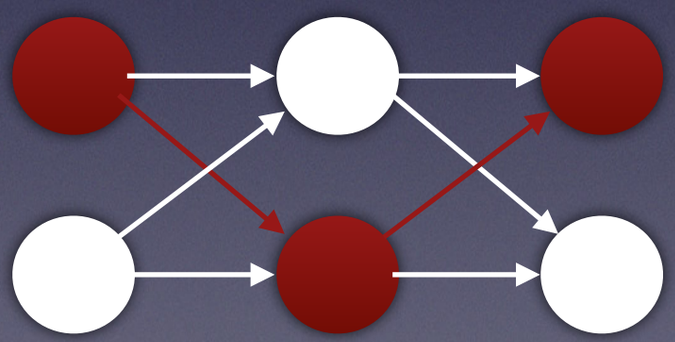


Neural Networks

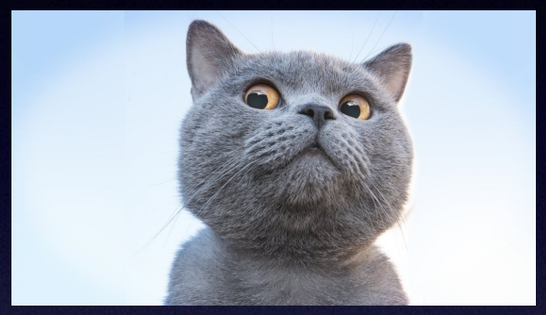


“Cat”

Extract



Ptolemy Overview

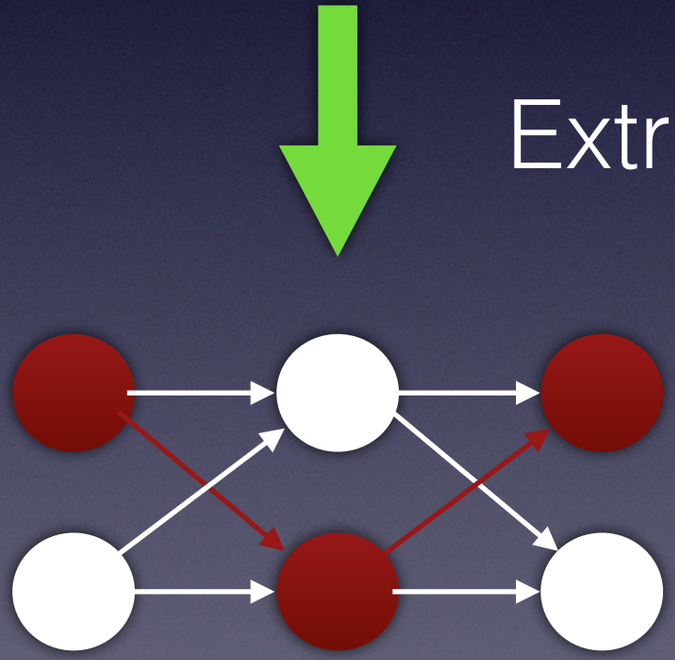


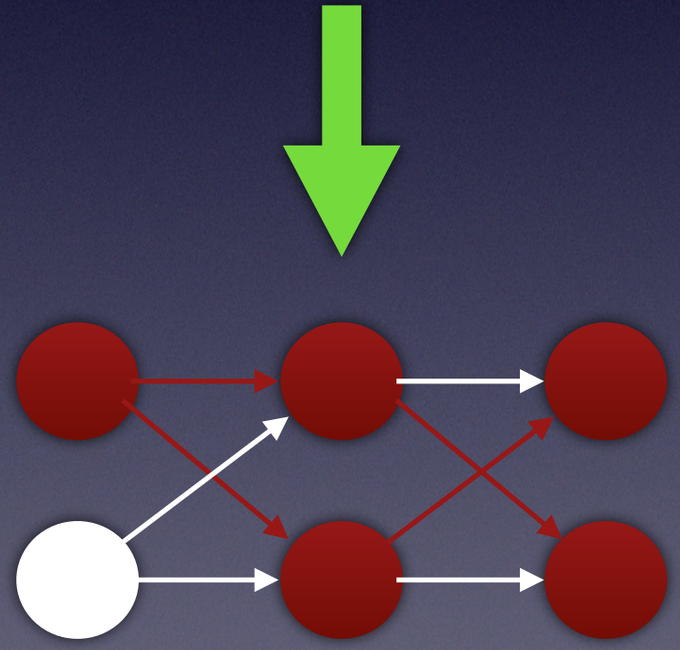
Neural Networks

“Cat”

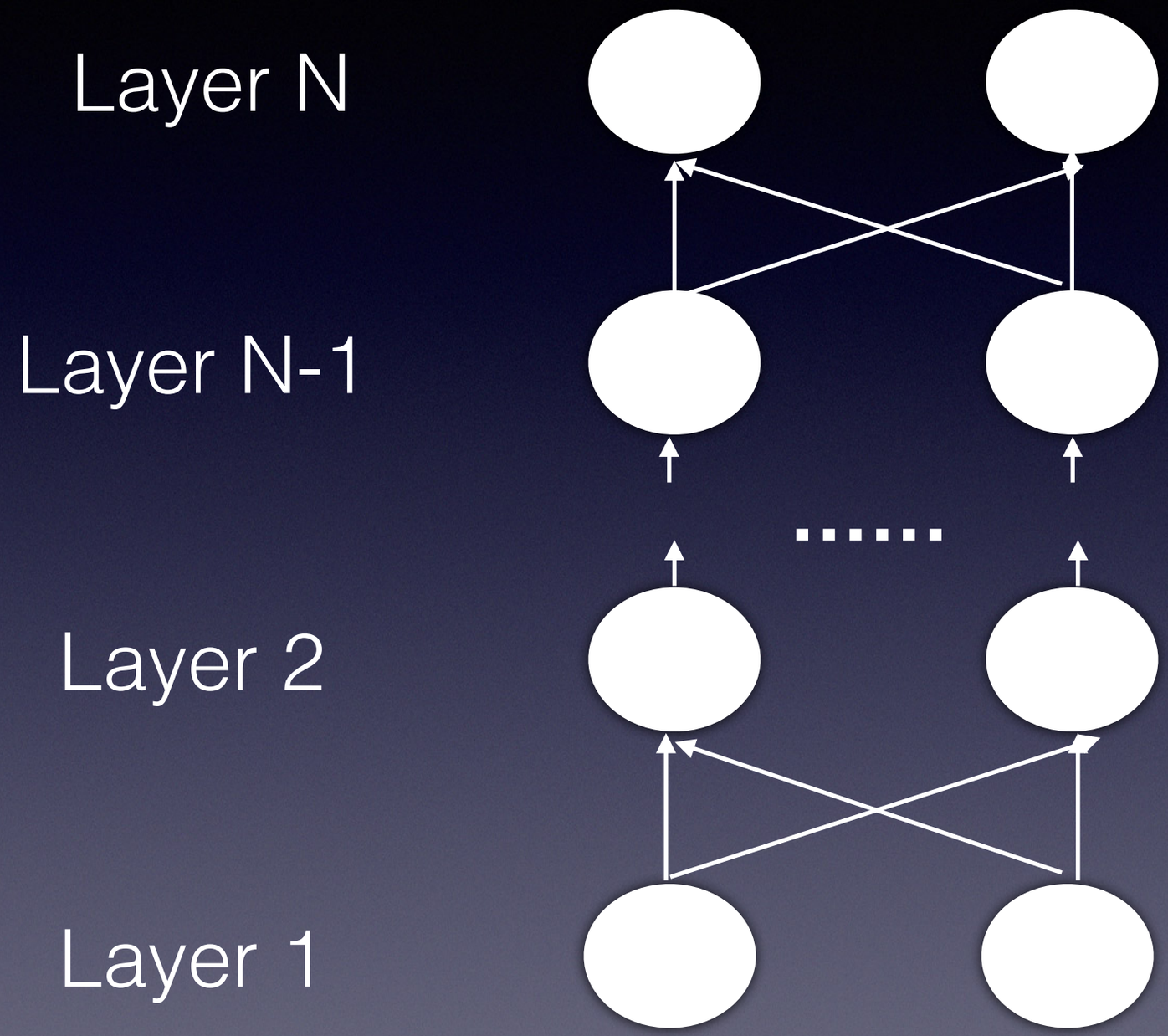
Extract

Compare





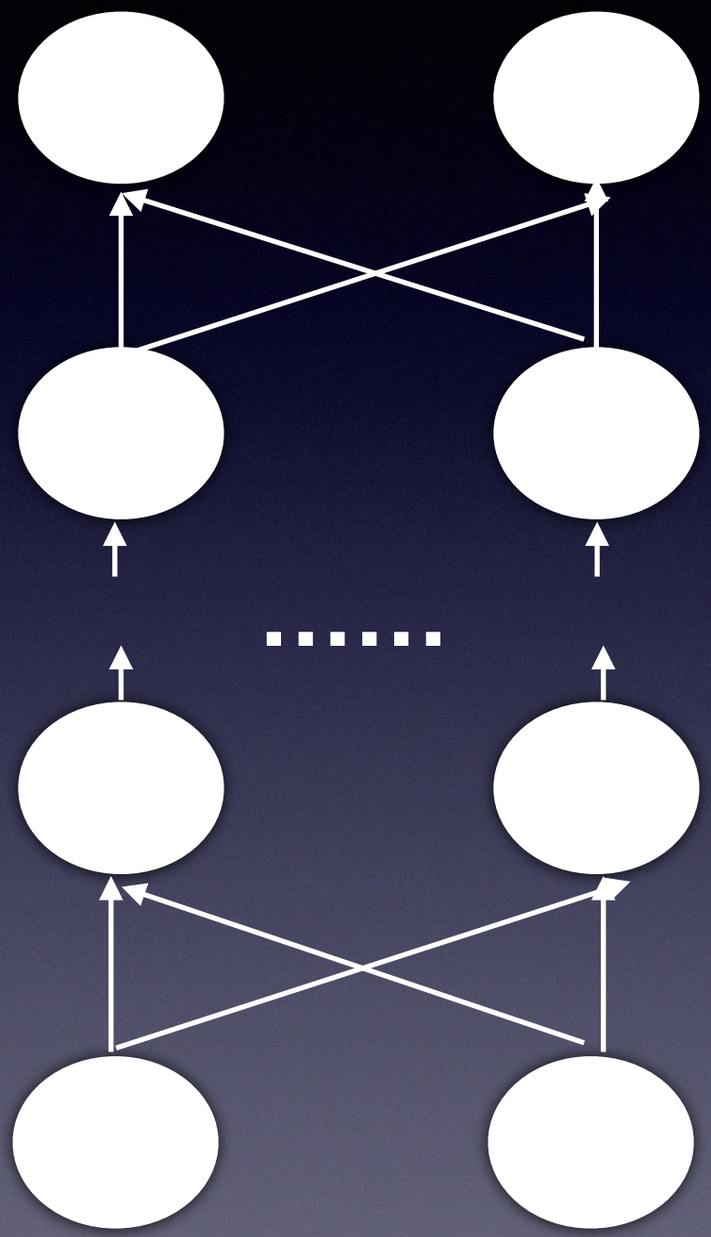
Ptolemy Pipeline



Ptolemy Pipeline

Layer N

Layer N-1



Layer 2

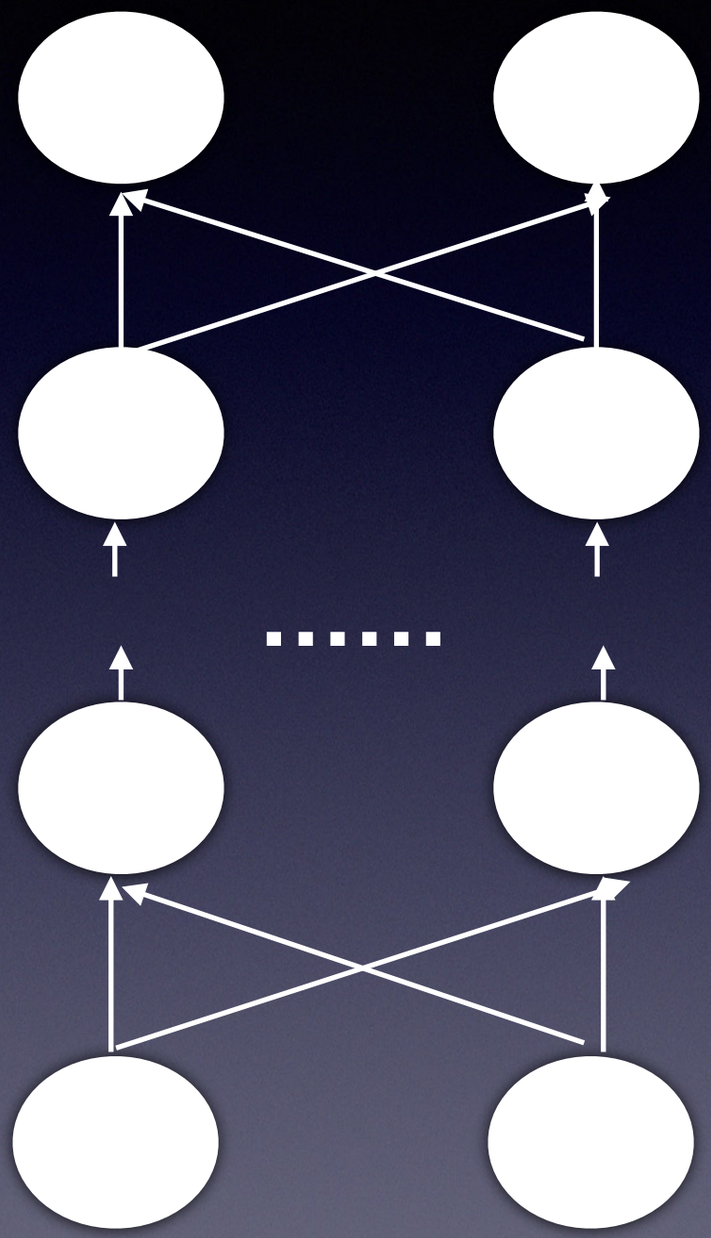
Layer 1

Inference

Ptolemy Pipeline

Layer N

Layer N-1



Layer 2

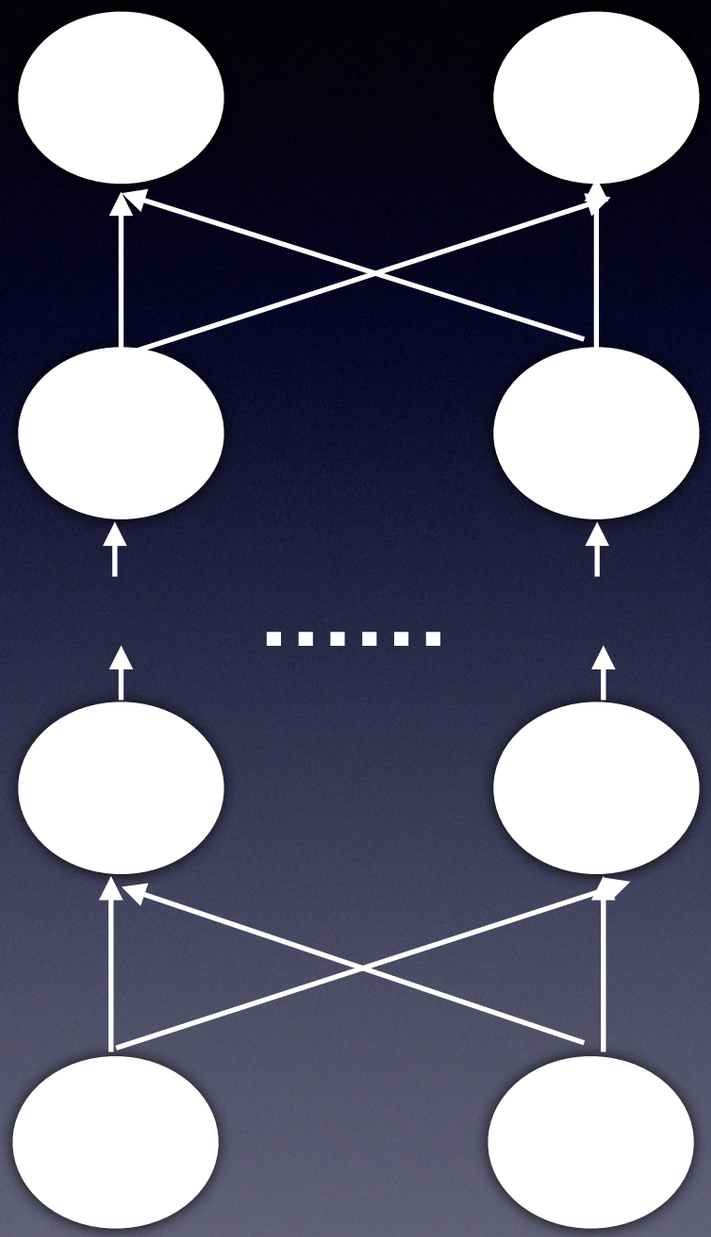
Layer 1

Inference

Ptolemy Pipeline

Layer N

Layer N-1



Inference

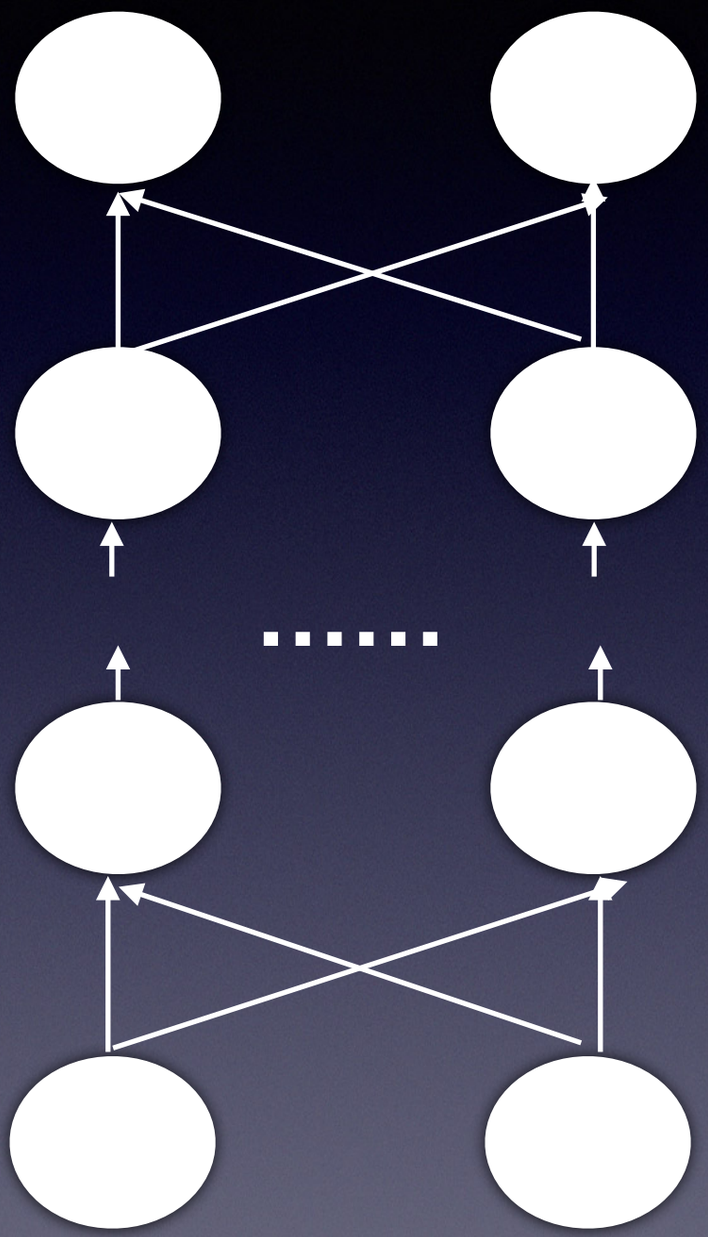
Layer 2

Layer 1

Ptolemy Pipeline

Layer N

Layer N-1



Inference

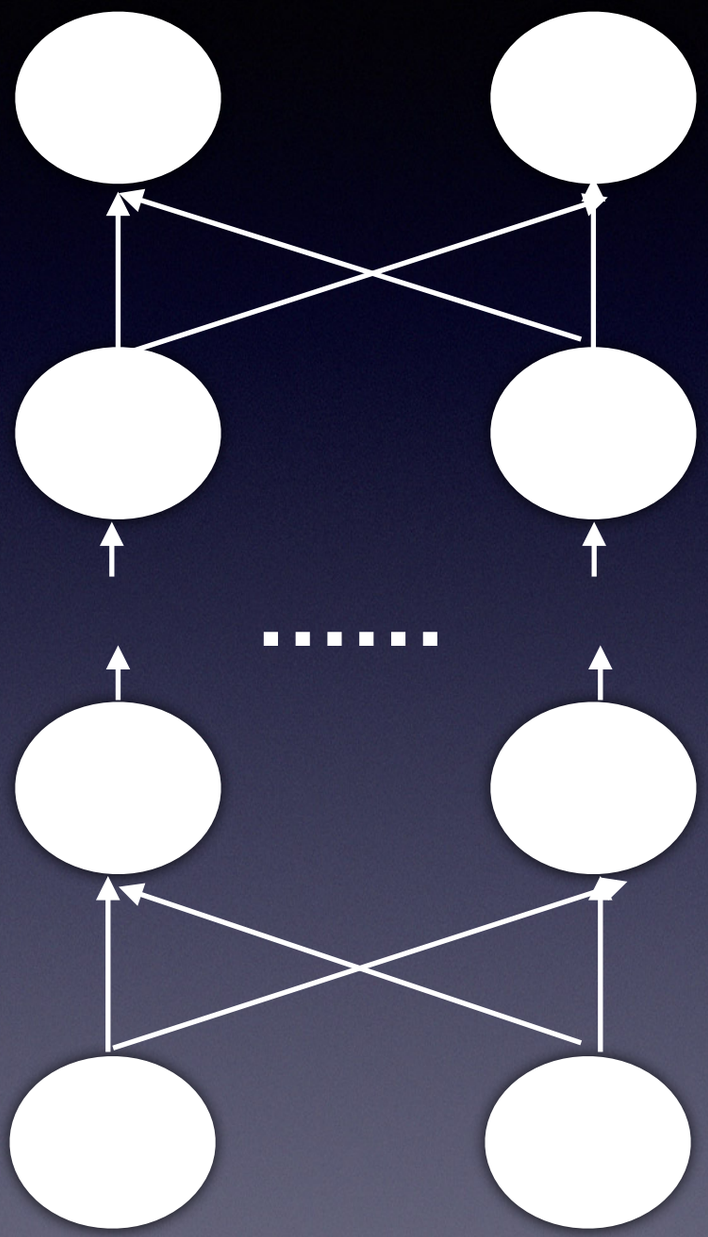
Layer 2

Layer 1

Ptolemy Pipeline

Layer N

Layer N-1



Extraction

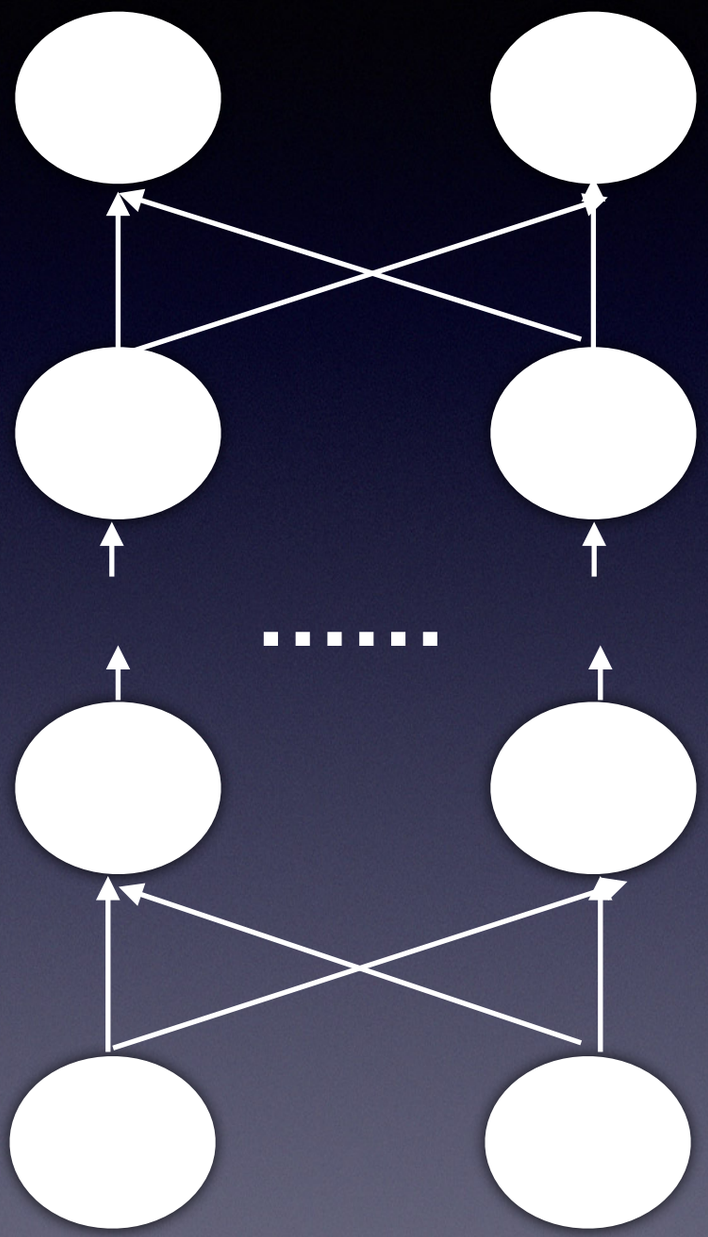
Layer 2

Layer 1

Ptolemy Pipeline

Layer N

Layer N-1



Extraction

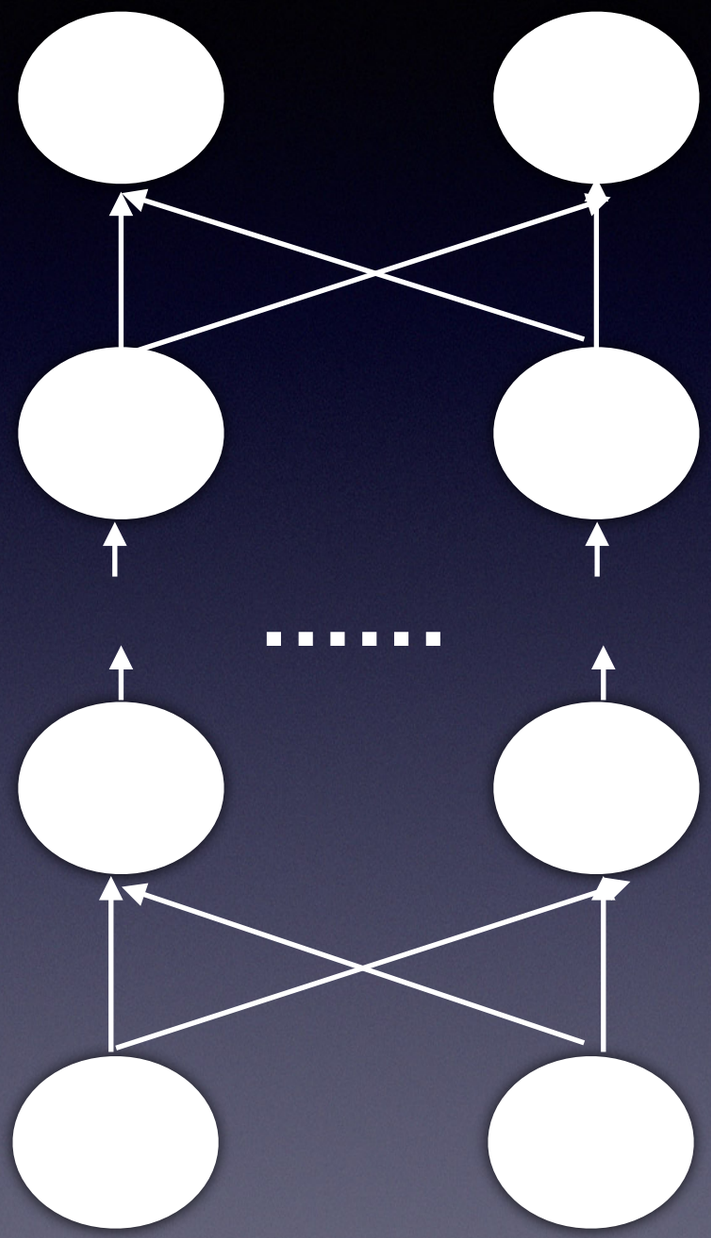
Layer 2

Layer 1

Ptolemy Pipeline

Layer N

Layer N-1



Layer 2

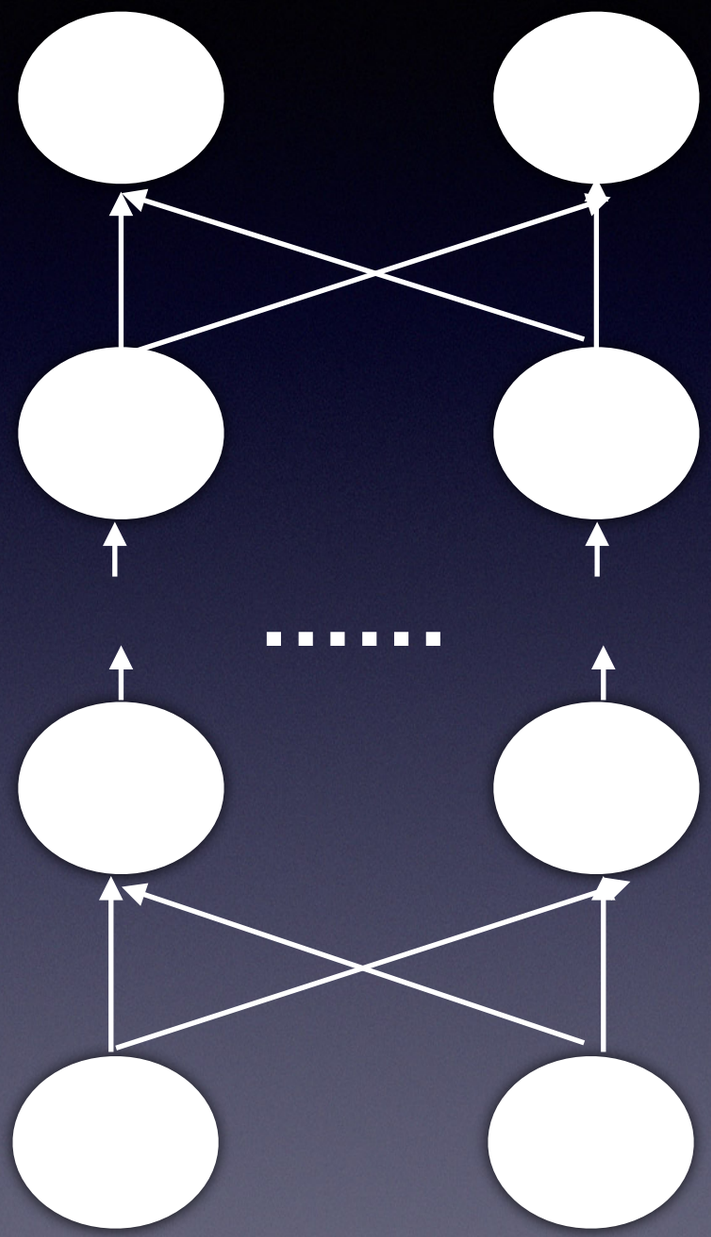
Layer 1

Extraction

Ptolemy Pipeline

Layer N

Layer N-1

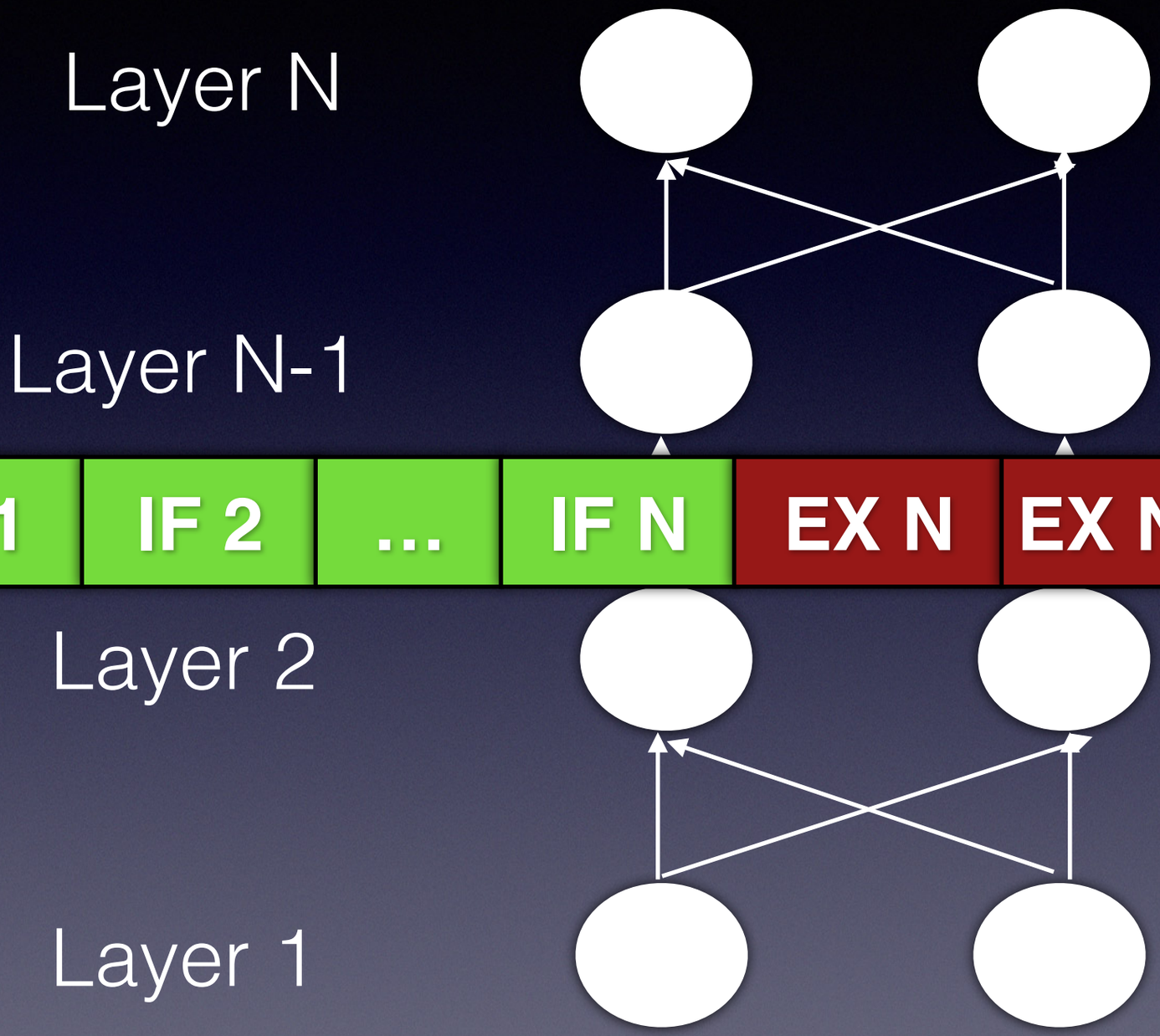


Layer 2

Layer 1

Extraction

Ptolemy Pipeline



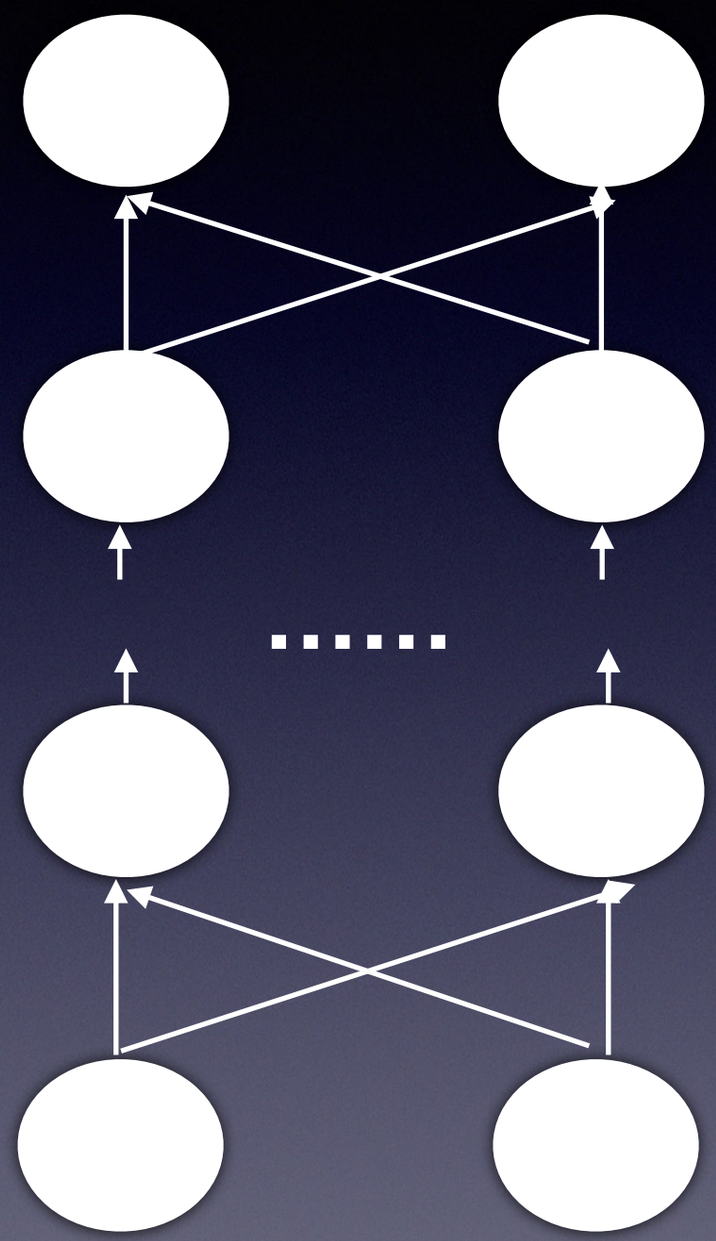
IF 1	IF 2	…	IF N	EX N	EX N-1	…	EX 1	Det

Extraction

Algorithmic Variation

Layer N

Layer N-1



Layer 2

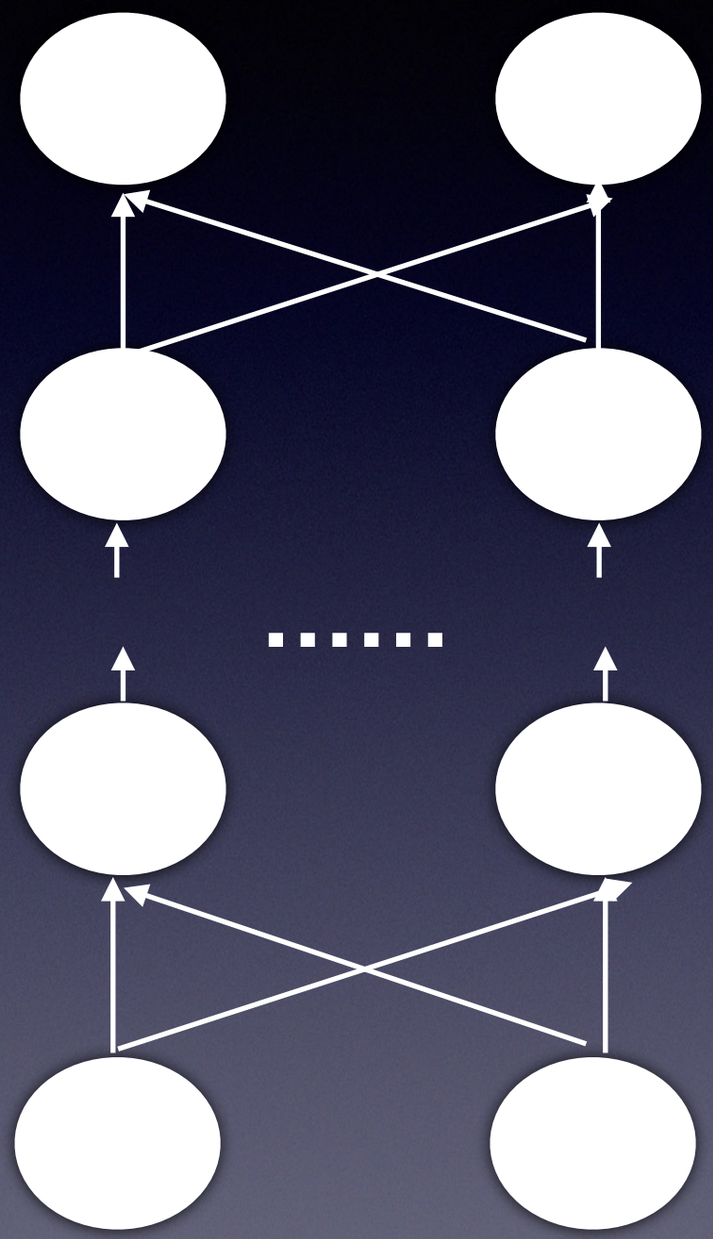
Layer 1

Inference

Algorithmic Variation

Layer N

Layer N-1



Layer 2

Layer 1

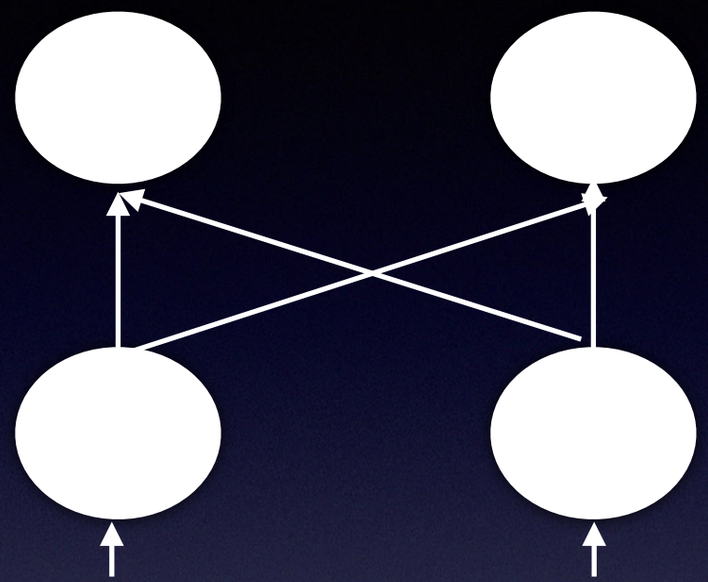
Inference

Extraction

Algorithmic Variation

Layer N

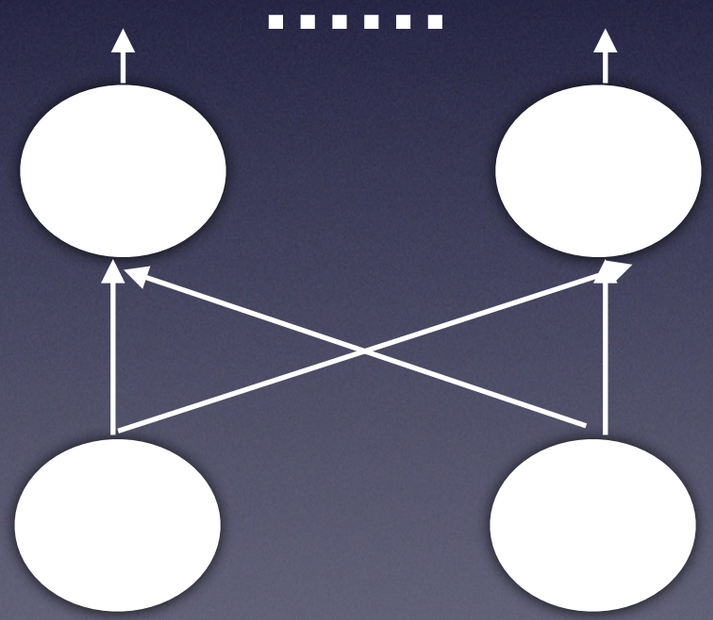
Layer N-1



Inference

Layer 2

Layer 1

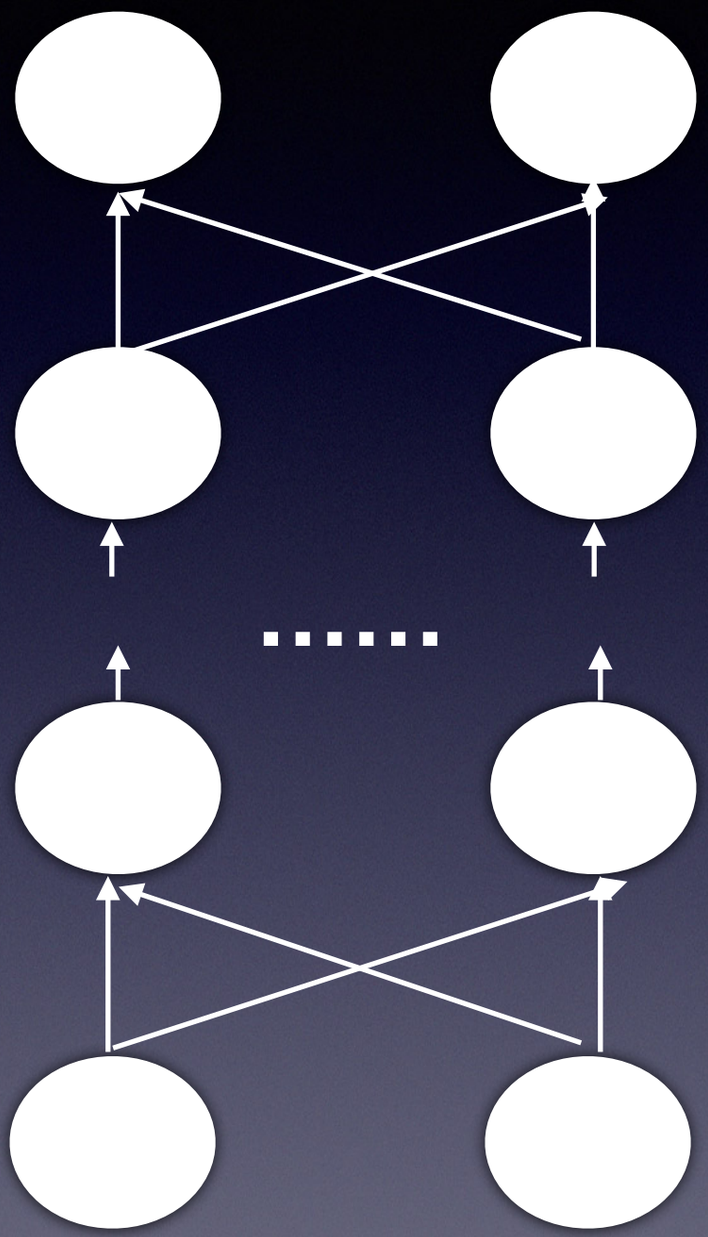


Extraction

Algorithmic Variation

Layer N

Layer N-1



Inference

Extraction

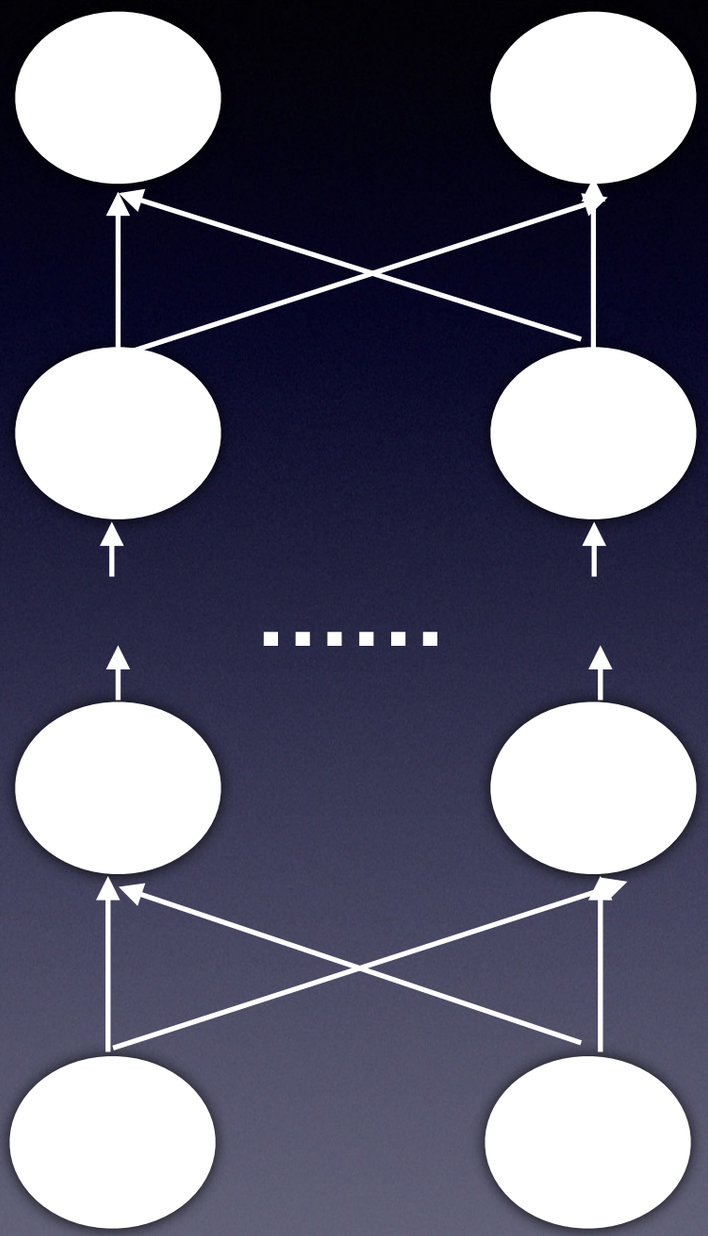
Layer 2

Layer 1

Algorithmic Variation

Layer N

Layer N-1



Extraction

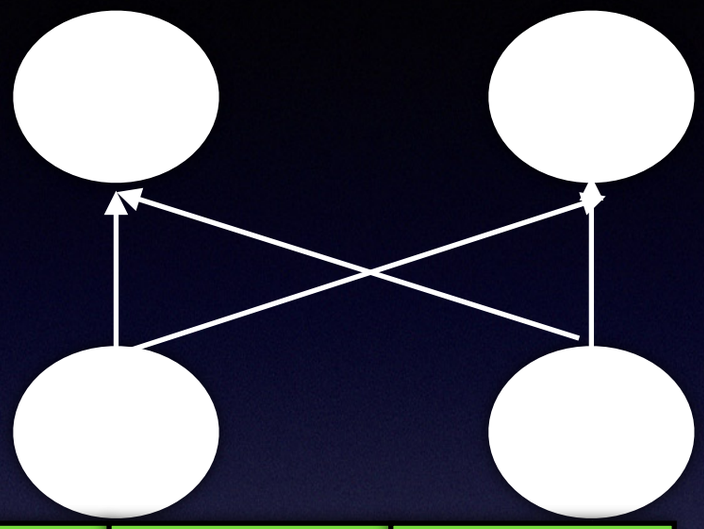
Layer 2

Layer 1

Algorithmic Variation

Layer N

Layer N-1



Extraction

IF 1 | **IF 2** | **…** | **IF N**

Layer 2

EX 1 | **EX 2** | **EX N** | **Det**

Layer 1

Algorithmic Variation

Sorting

Threshold

IF: Inference, EX: Extraction, Det: Detection

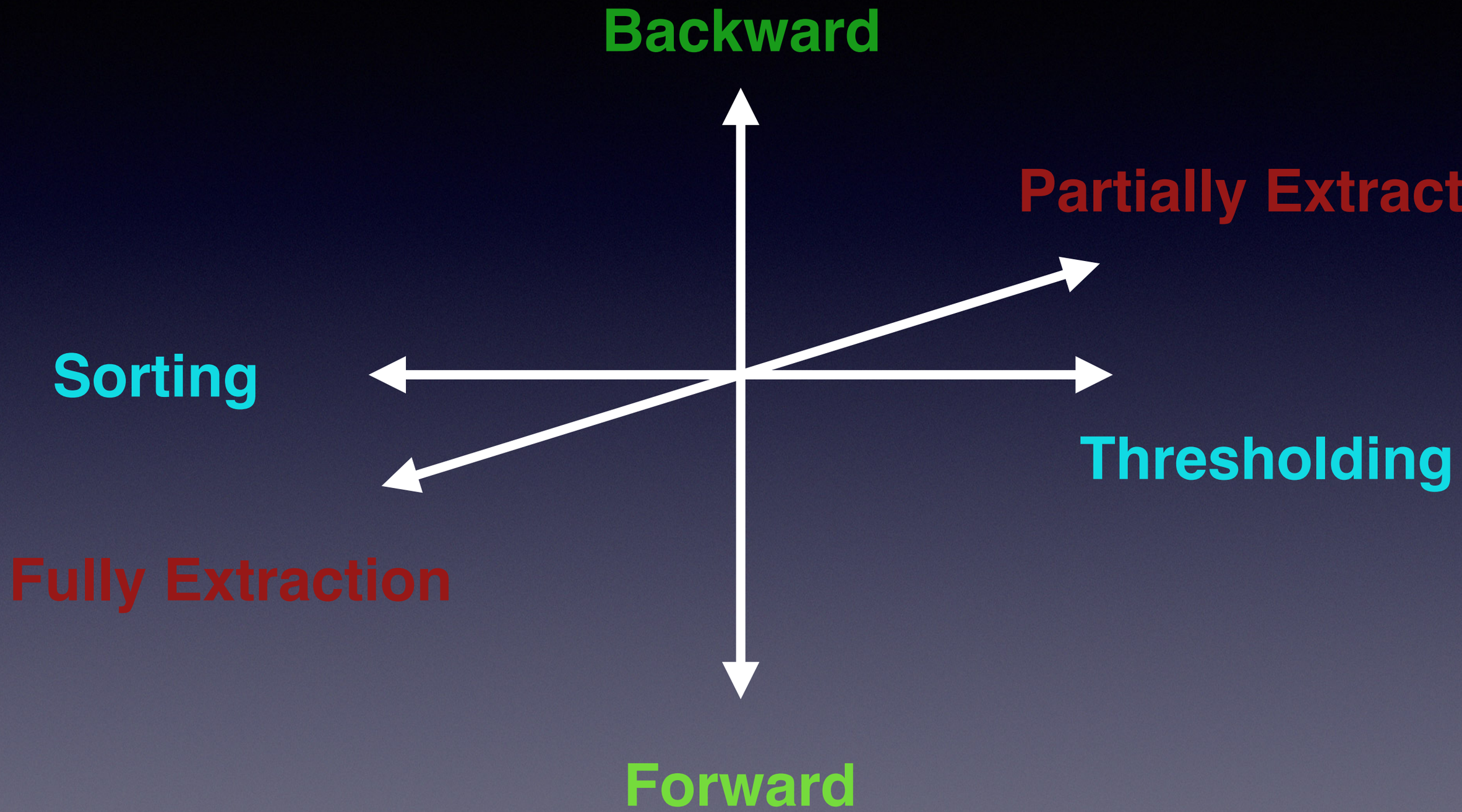
Algorithmic Variation

Full Extraction

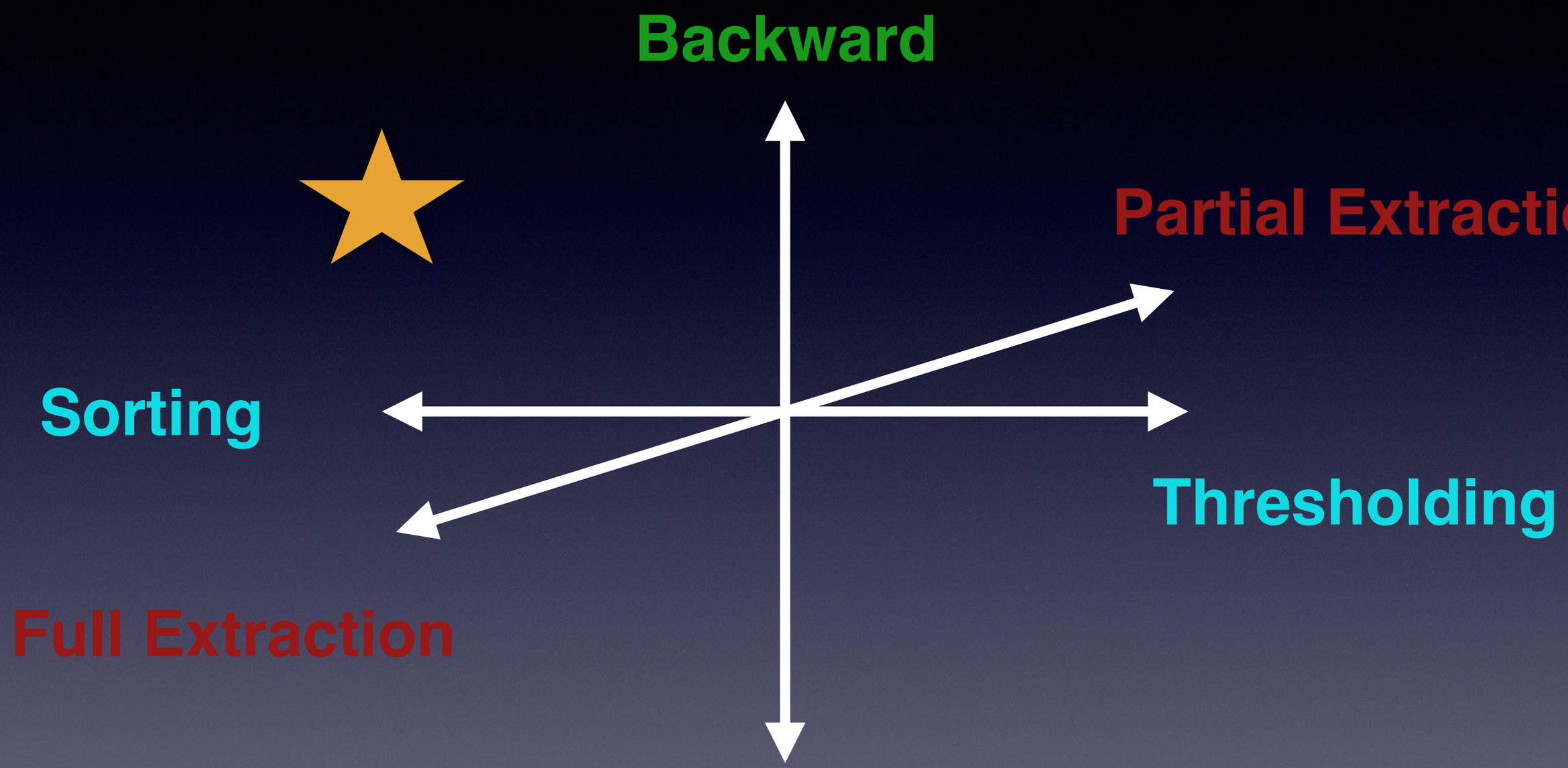
Partially Extraction

IF: Inference, EX: Extraction, Det: Detection

Framework


Backward
Partially Extraction
Sorting
Thresholding
Fully Extraction
Forward

Framework



= Backward + Fully Extraction + Sorting

Interface

- High-level: Python-based, user define input

Interface

- High-level: Python-based, user define input

Compiler

- Low-level: Customized ISA

Compiler Optimization: Layer Level

```
for j = 1 to L {
      inf(j)
      <extraction on layer j>
}
```

Compiler Optimization: Layer Level

```
for j = 1 to L {
      inf(j)
      <extraction on layer j>
}
```

↓

```
inf(1)
for j = 1 to L {
      inf (j+1)
      <extraction on layer j>
}
<extraction on layer L>
```

Compiler Optimization: Neuron Level

```
for j = 1 to N {
    sort(i)
    acum(i)
}
```

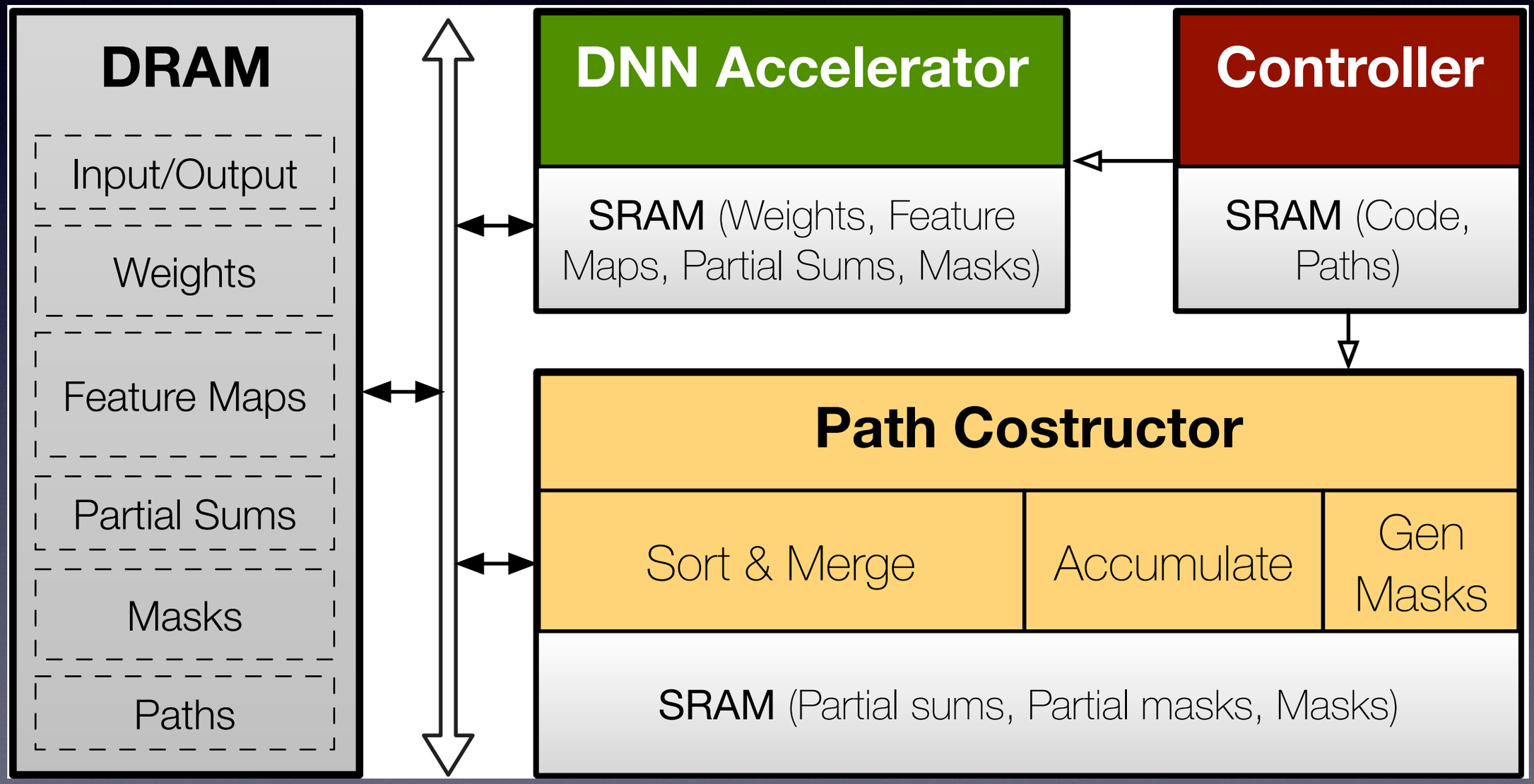
Compiler Optimization: Neuron Level

```
for j = 1 to N {
    sort(i)
    acum(i)
}
```

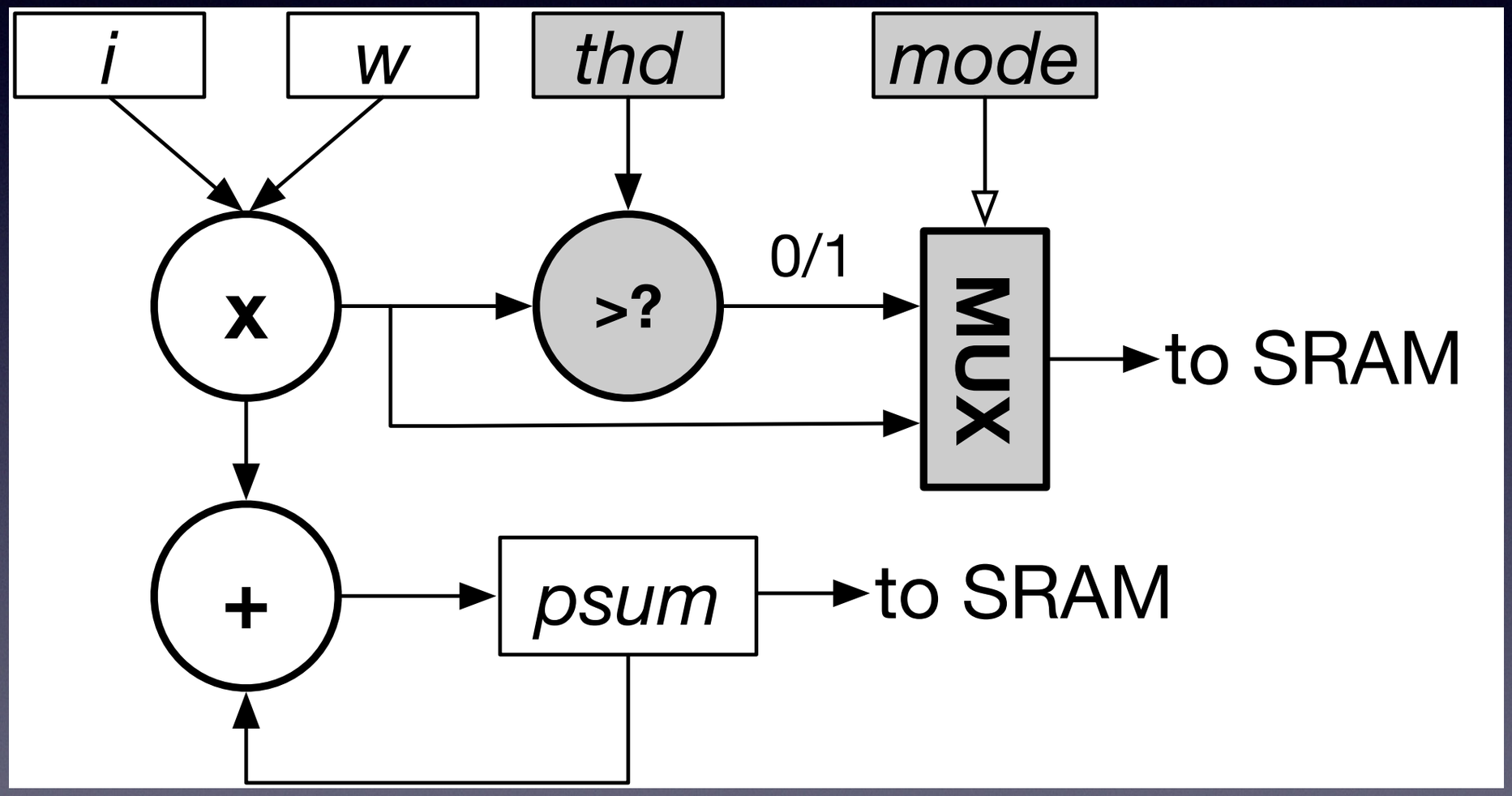
↓

```
sort(1)
for i = 1 to N-1{
    sort(i+1)
    acum(i)
}
acum(N)
```

Architecture Overview



Enhanced MAC unit

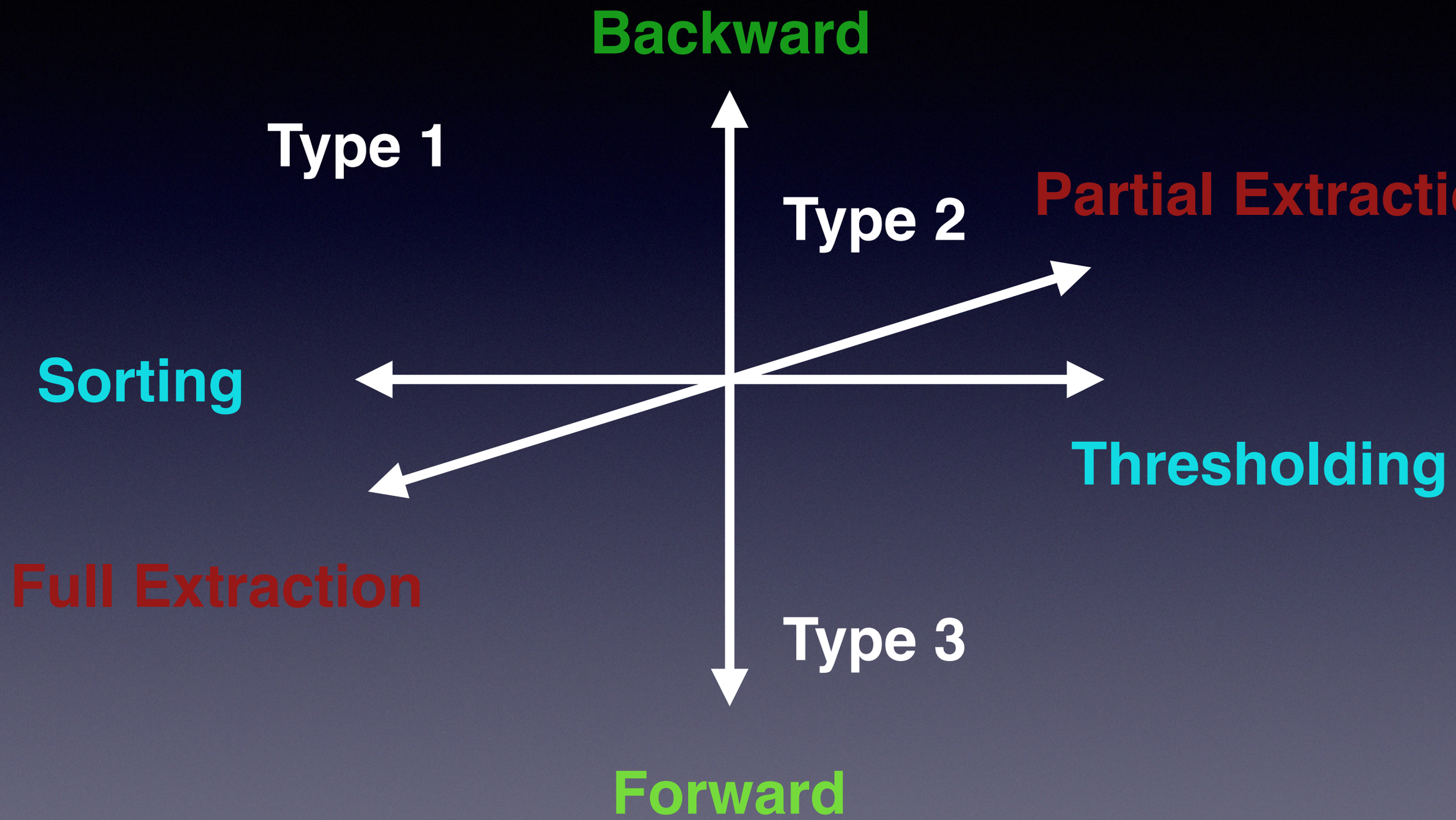


Evaluation

Network	AlexNet, ResNet
Dataset	Cifar10, Cifar100,ImageNet
Attacks	BIM, CWL2, DeepFool, FGSM,JSMA
Adaptive Attacks	Self constructed
Baselines	EP[1], CDRP[2]

[1]Y. Qiu, J. Leng, C. Guo, et.al, Adversarial Defense Through Network Profiling Based Path Extraction
[2]Y. Wang, H. Su, B. Zhang, X. Hu, Interpret neural networks by identifying critical data routing paths.

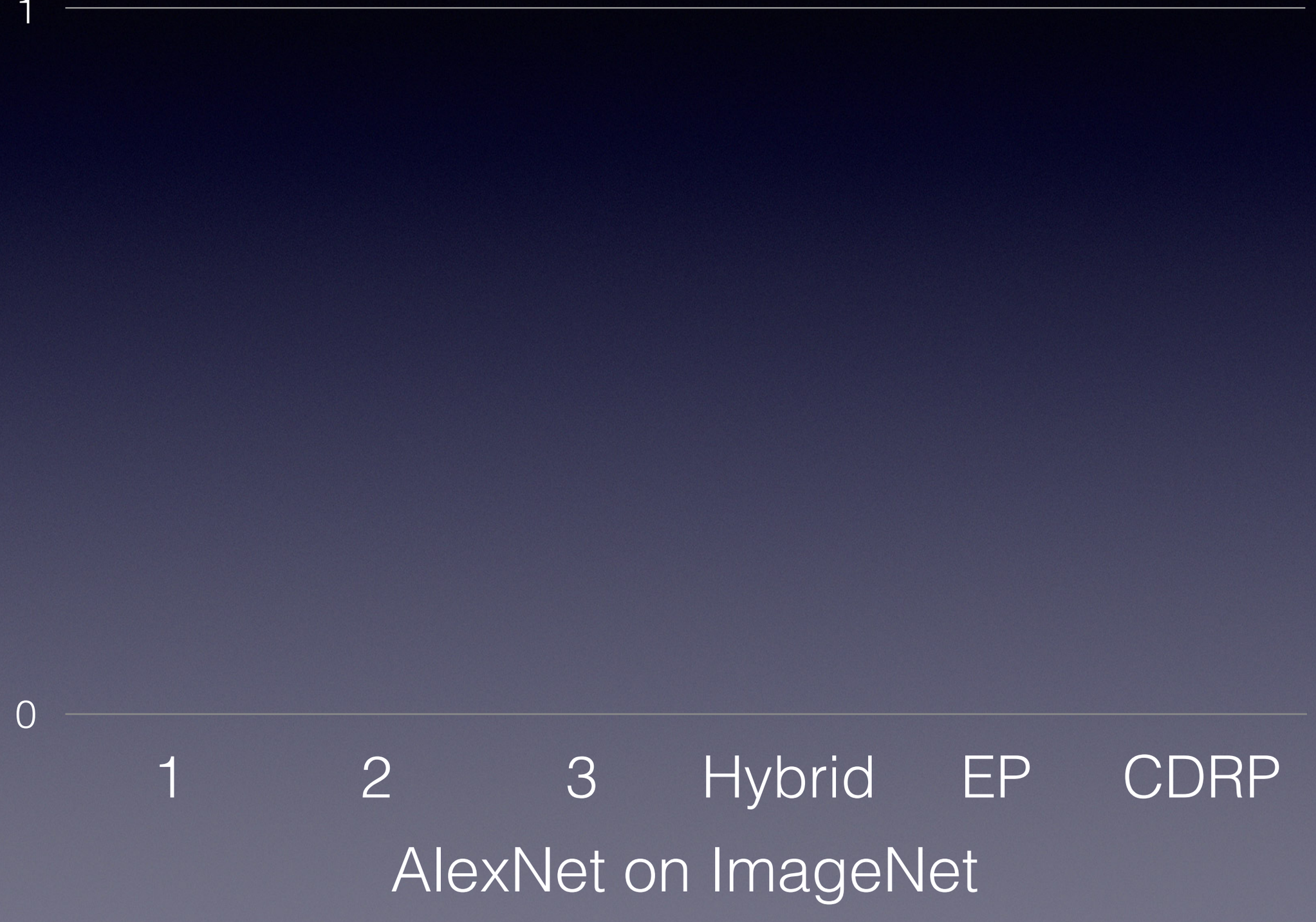
Evaluation


Backward
Type 1
Type 2
Partial Extraction
Sorting
Thresholding
Full Extraction
Type 3
Forward

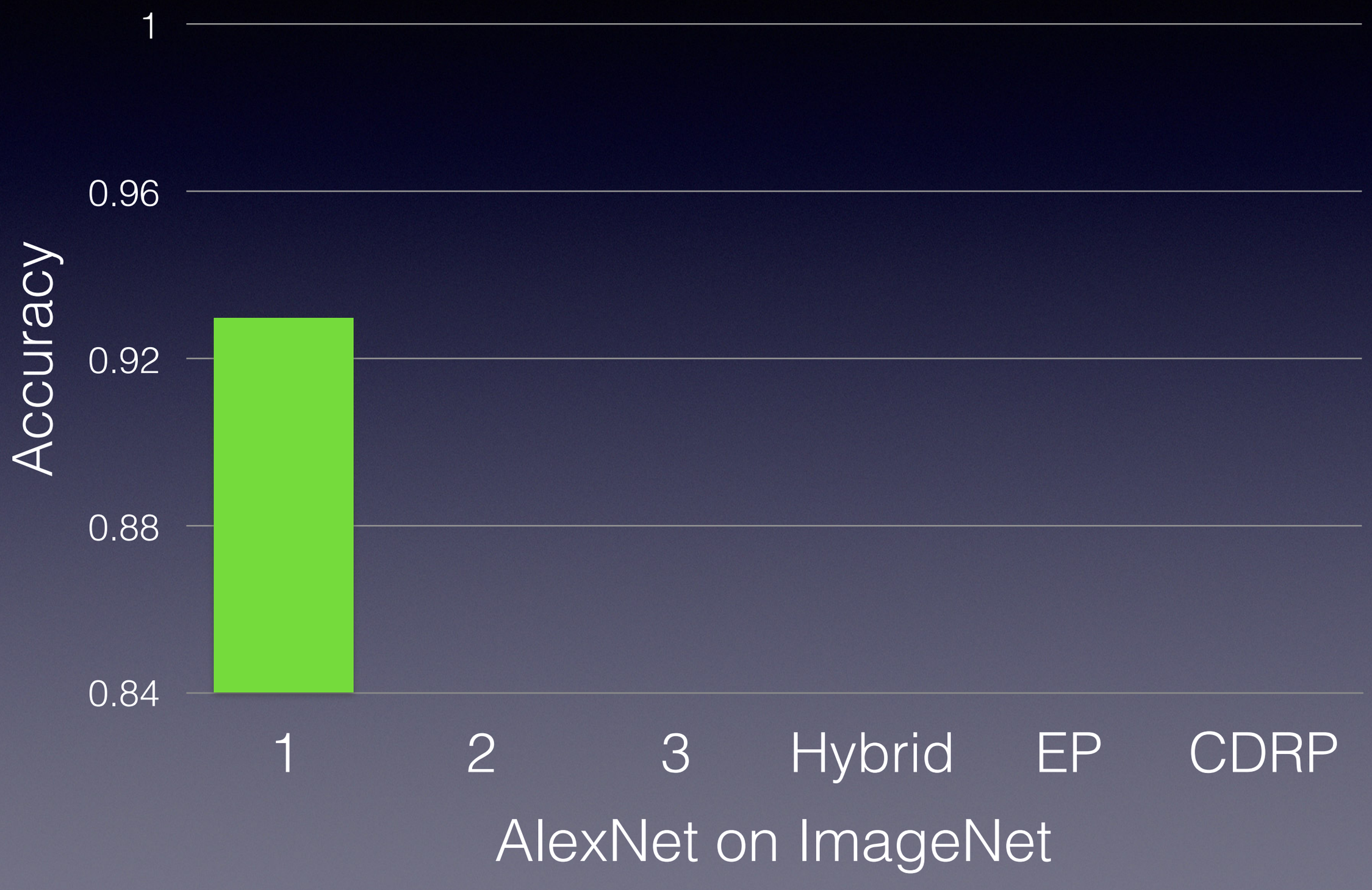
Hardware Setup

DNN Accelerator	20 x 20
Technology	Silvaco 15nm
On-chip SRAM	1.5MB

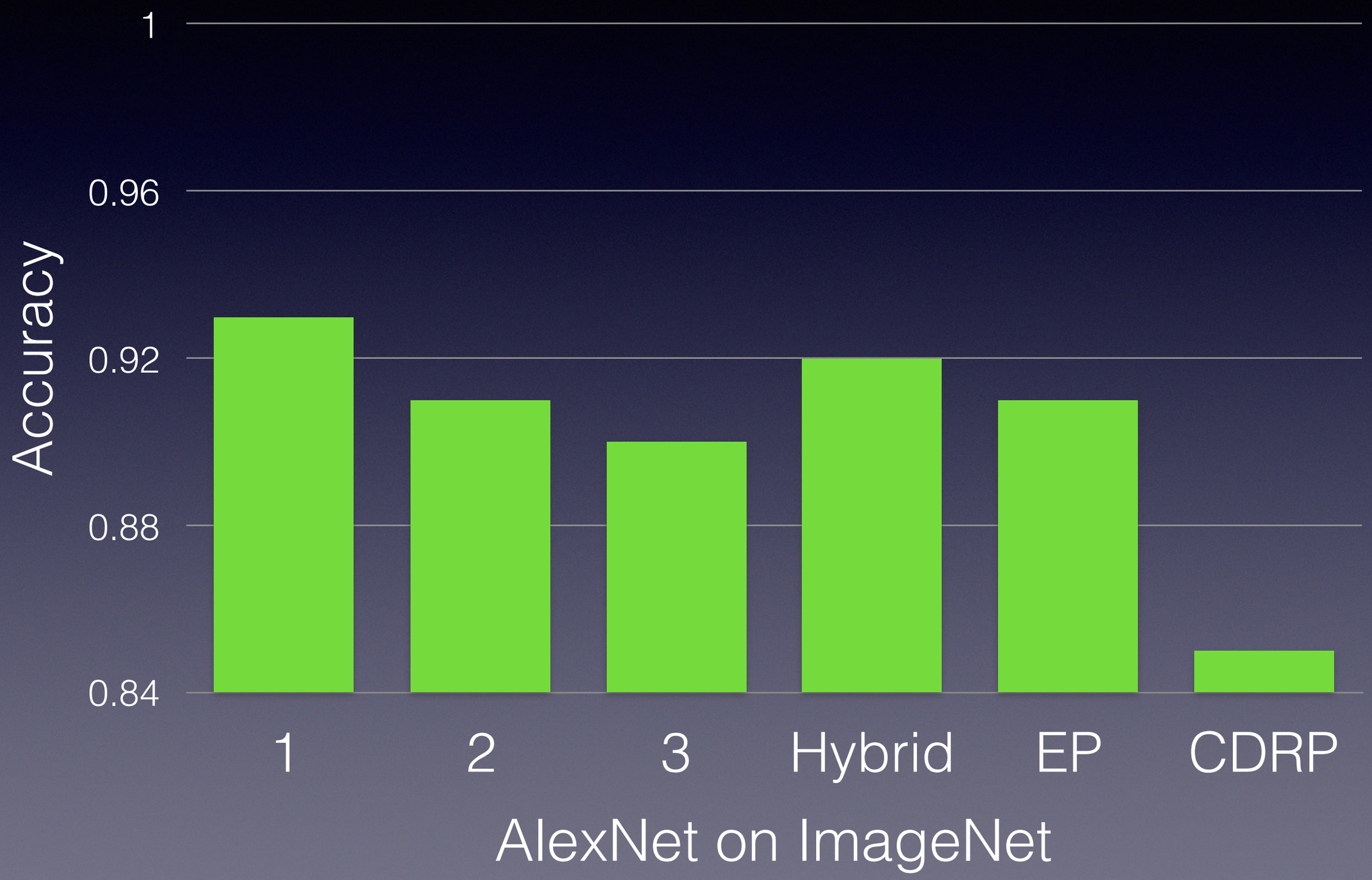
Evaluation



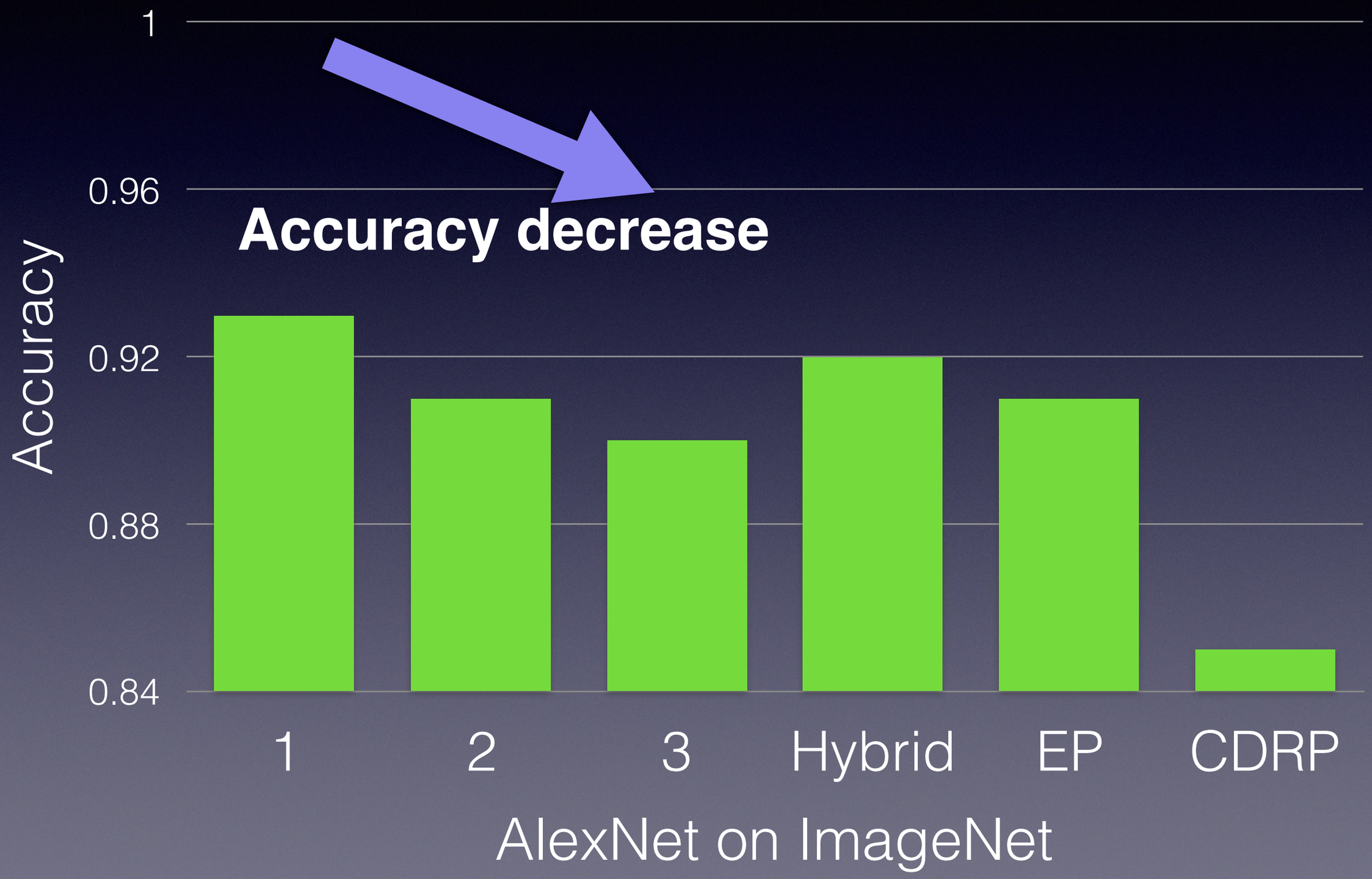
Evaluation


1
0.96
0.92
0.88
0.84
Accuracy
1
2
3
Hybrid
EP
CDRP
AlexNet on ImageNet

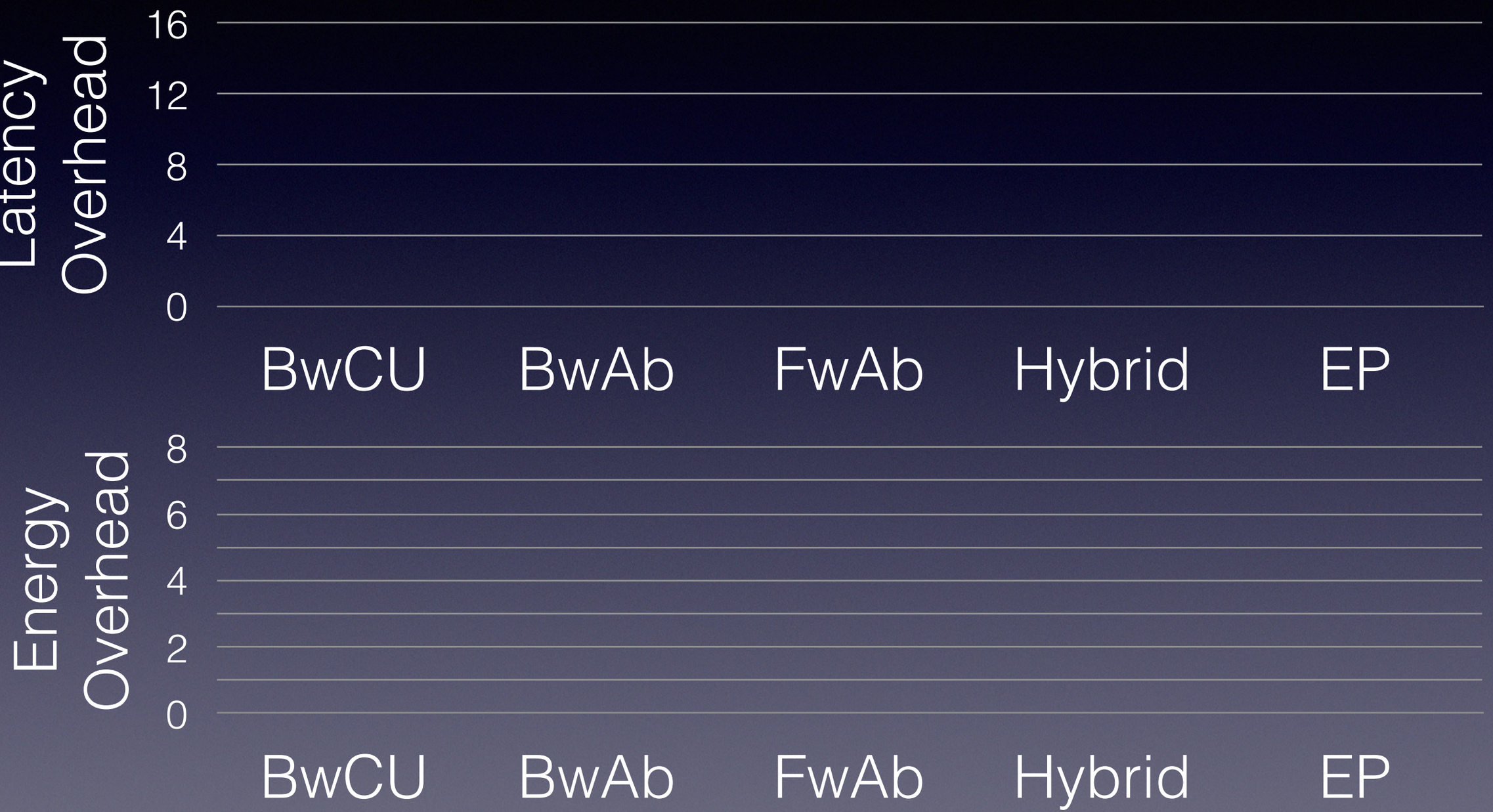
Evaluation


1
0.96
0.92
0.88
0.84
Accuracy
1
2
3
Hybrid
EP
CDRP
AlexNet on ImageNet

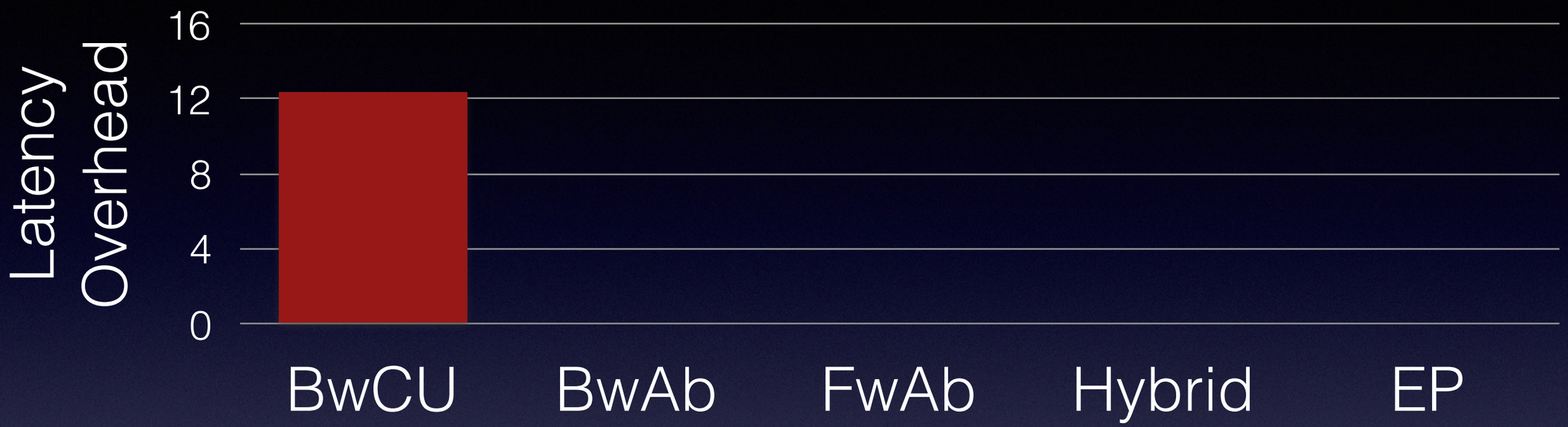
Evaluation


1
0.96
0.92
0.88
0.84
Accuracy
Accuracy decrease
1
2
3
Hybrid
EP
CDRP
AlexNet on ImageNet

Evaluation

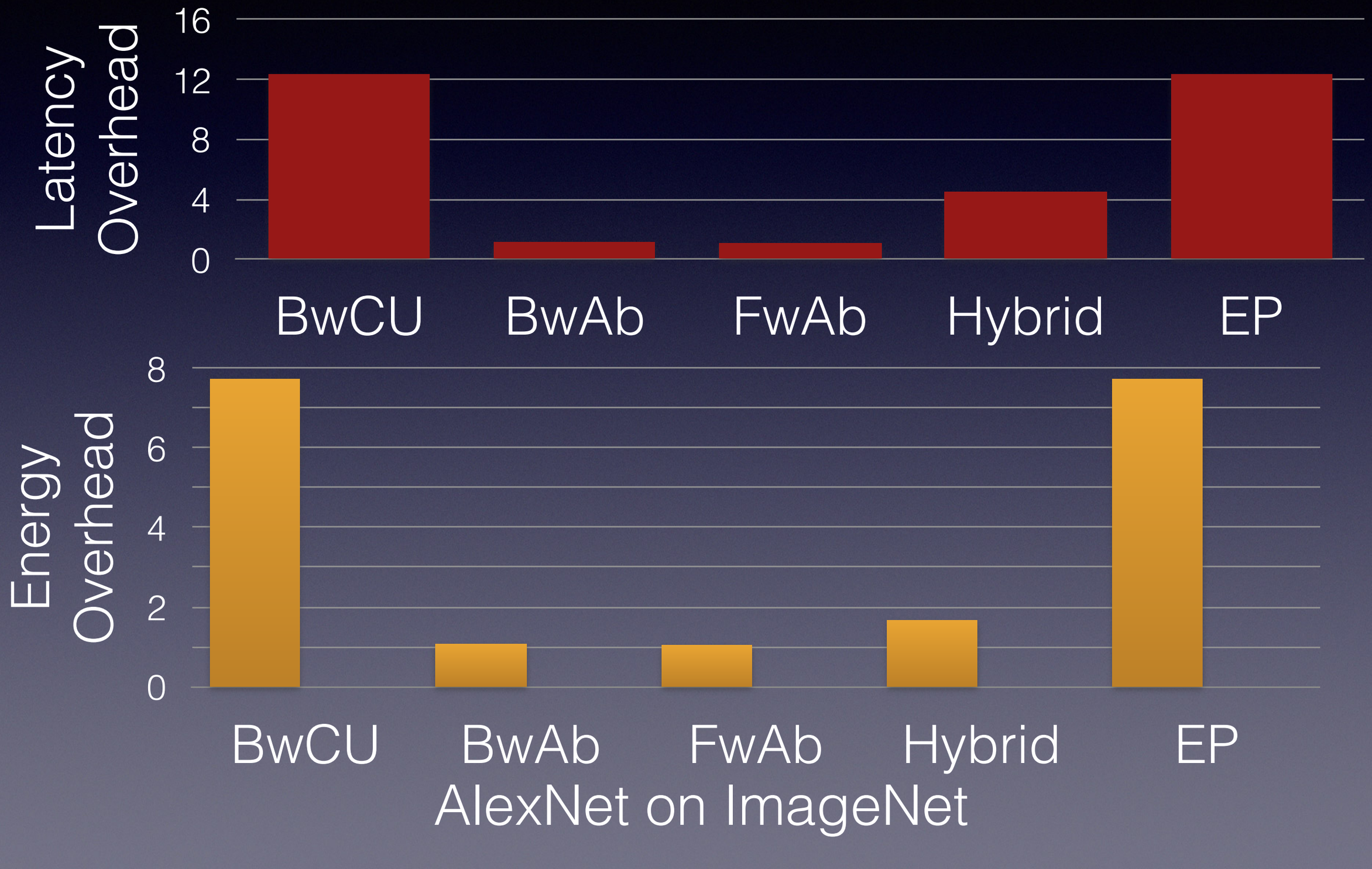

Latency Overhead
16
12
8
4
0
BwCU
BwAb
FwAb
Hybrid
EP
Energy Overhead
8
6
4
2
0
BwCU
BwAb
FwAb
Hybrid
EP
AlexNet on ImageNet

Evaluation

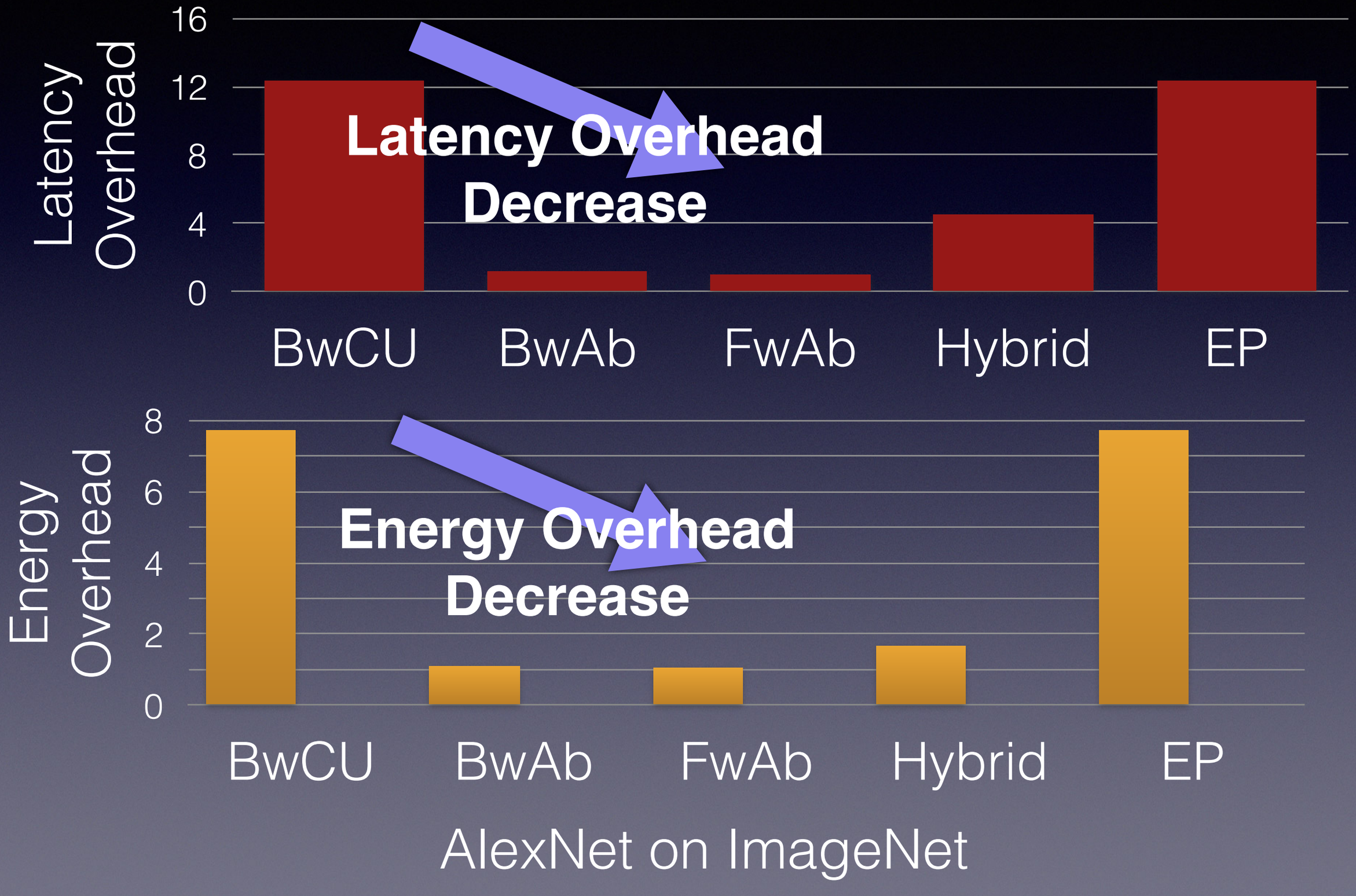


AlexNet on ImageNet

Evaluation


Latency Overhead
16
12
8
4
0
BwCU
BwAb
FwAb
Hybrid
EP
Energy Overhead
8
6
4
2
0
BwCU
BwAb
FwAb
Hybrid
EP
AlexNet on ImageNet

Evaluation


Latency Overhead
16
12
8
4
0
Latency Overhead Decrease
BwCU
BwAb
FwAb
Hybrid
EP
Energy Overhead
8
6
4
2
0
Energy Overhead Decrease
BwCU
BwAb
FwAb
Hybrid
EP
AlexNet on ImageNet

Conclusion

Ptolemy: Accurate, low overhead, adversarial attack detection

- Algorithm Framework
- Compiler Optimization
- Architecture Support

Collaborators

Yuxian Qiu

Jingwen Leng

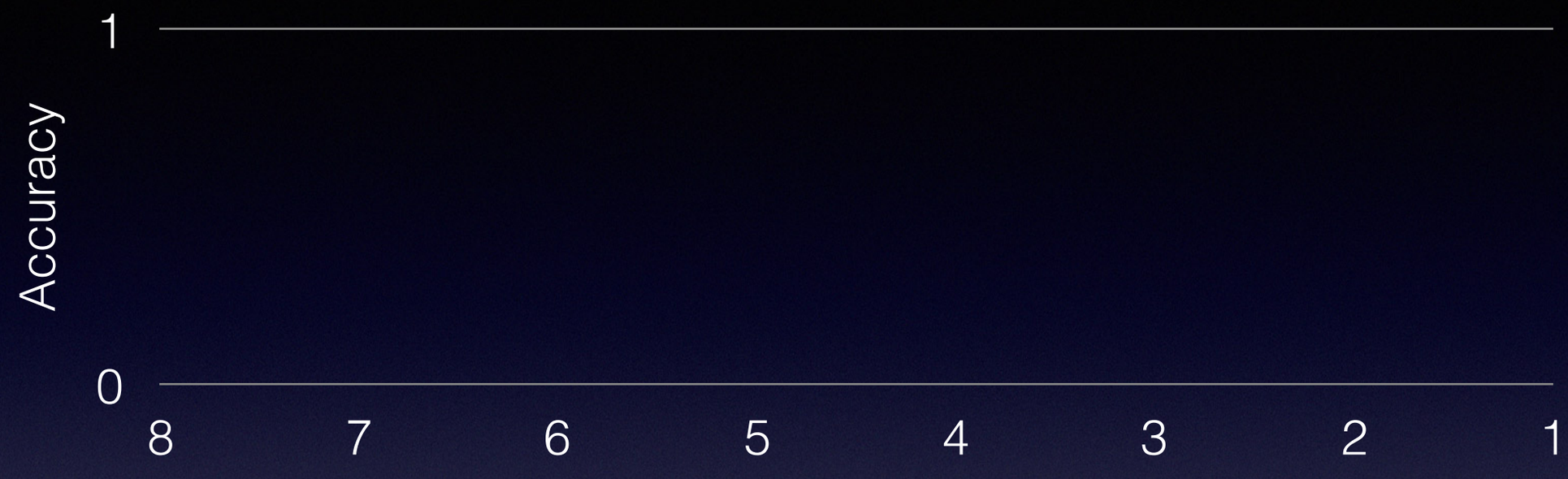
Minyi Guo

Yuhao Zhu

Questions

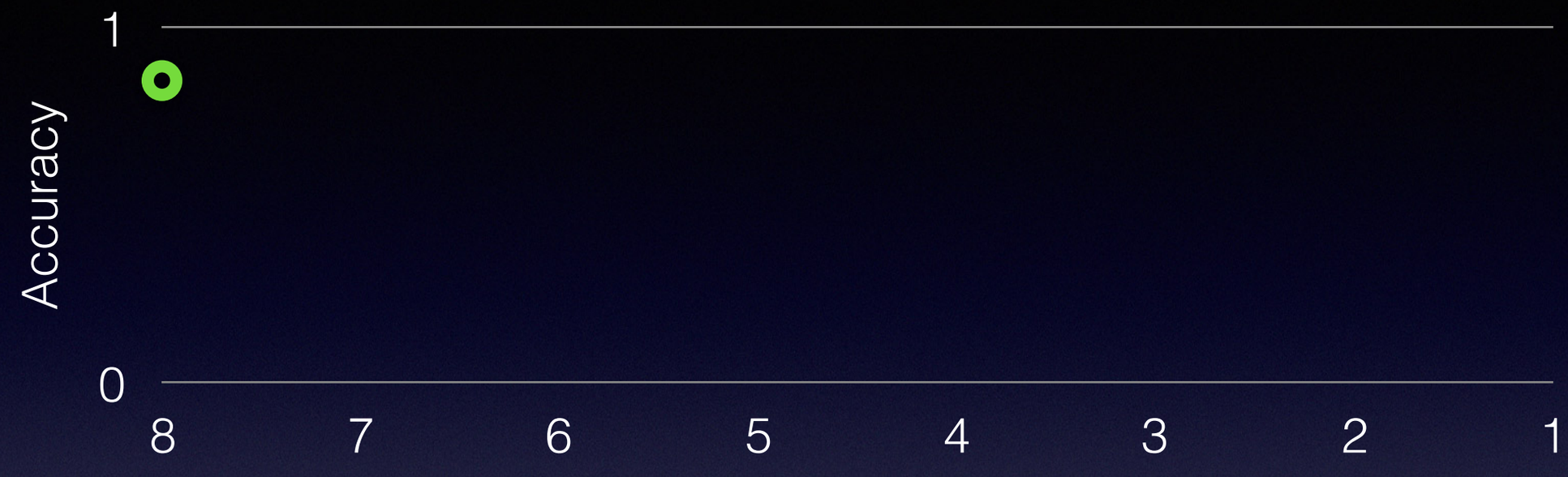
https://github.com/Ptolemy-dl/Ptolemy

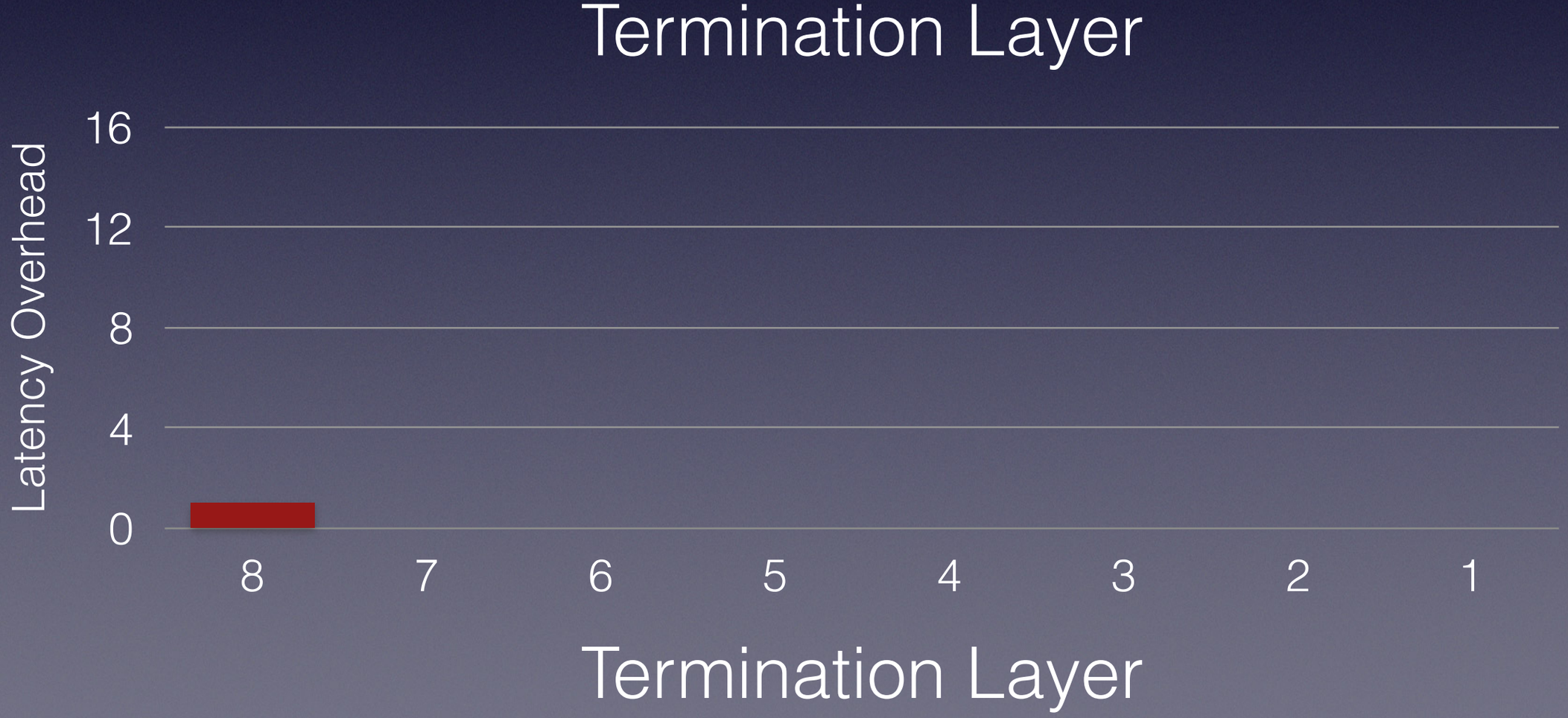
Evaluation


1
0
Accuracy
8
7
6
5
4
3
2
1

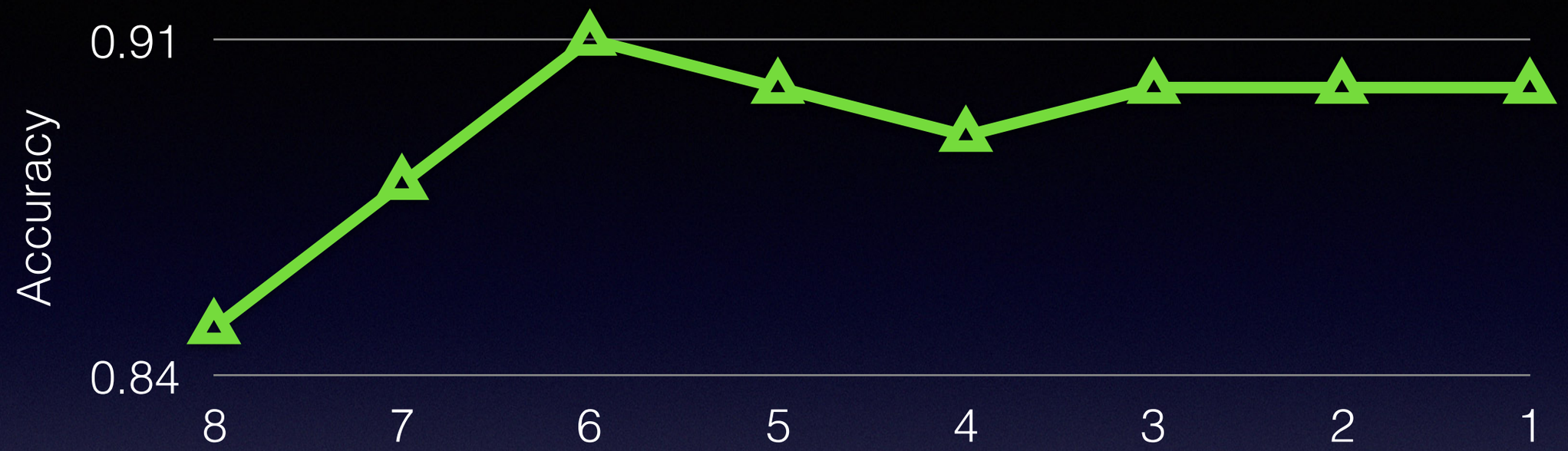

Termination Layer
4
3
2
1
0
Latency Overhead
8
7
6
5
4
3
2
1
Termination Layer

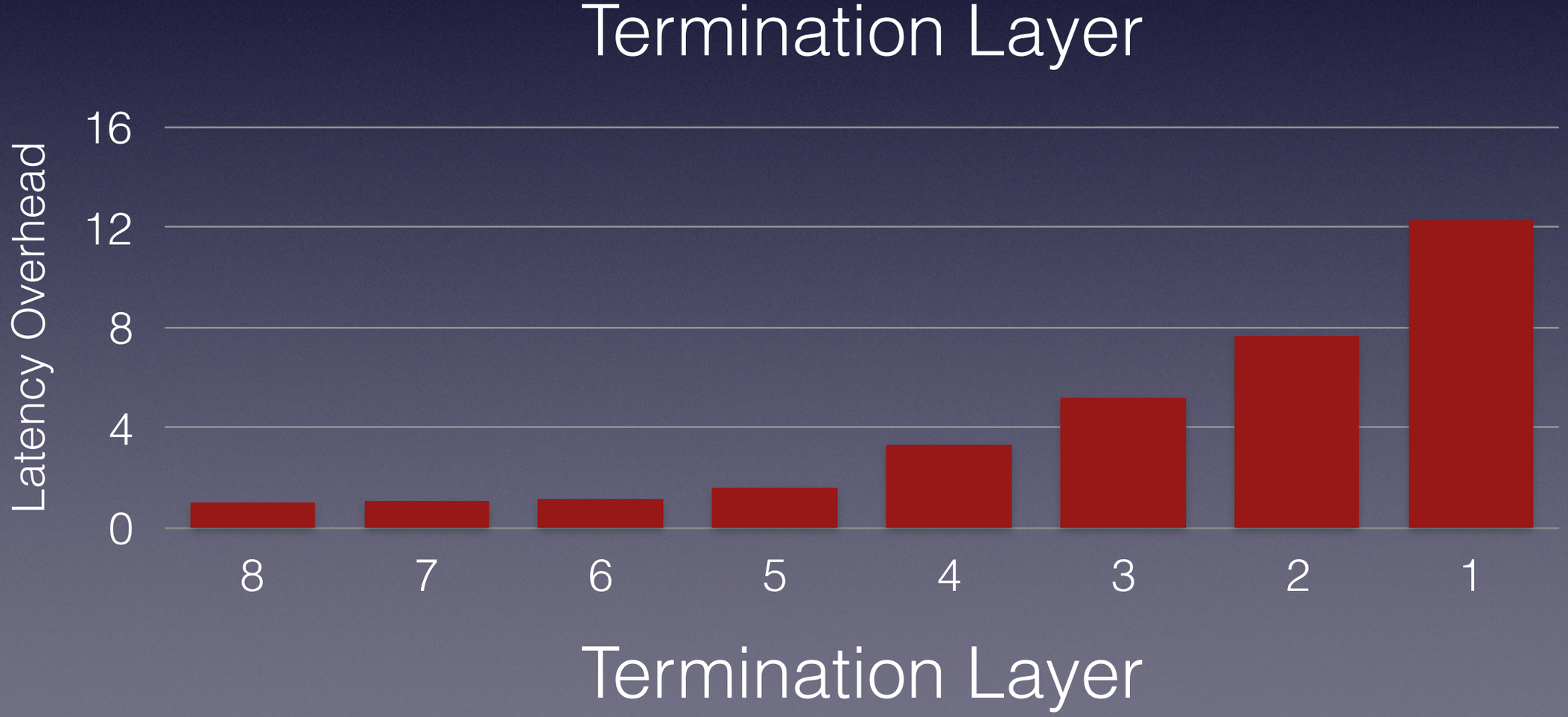
Evaluation


Accuracy
1
0
8
7
6
5
4
3
2
1


Termination Layer
Latency Overhead
16
12
8
4
0
8
7
6
5
4
3
2
1
Termination Layer

Evaluation


0.91
0.84
Accuracy
8
7
6
5
4
3
2
1


Termination Layer
16
12
8
4
0
Latency Overhead
8
7
6
5
4
3
2
1
Termination Layer

Backup

```
def AdversaryDetection(model, input, θ, ϕ):
  output = Inference(model, input)
  N = model.num_layers
  // Selective extraction only in the last three layers
  for L in range(N-3, N):
    if L != N-1:
      // Forward extraction using absolute thresholds
      ImptN[L] = ExtractImptNeurons(1, 1, ϕ, L)
    else:
      // Forward extraction using cumulative thresholds
      ImptN[L] = ExtractImptNeurons(1, 0, θ, L)
    dynPath.concat(GenMask(ImptN[L]))
  classPath = LoadClassPath(argmax(output))
  is_adversary = Classify(classPath, dynPath)
  return is_adversary
```

Backup

Class	Name	23-20	19-16	15-12	11-8	7-4	3-0
Inference	**inf**	0000	Input addr.	Weight addr.	Output addr.	Unused	
	infsp	0001	Input addr.	Weight addr.	Output addr.	First partial sum addr.	Unused
	csps	0010	Output neuron ID	Layer ID	First partial sum addr.	Unused	
Path Construction	**sort**	0011	Unsorted seq. start addr.	Seq. length	Sorted seq. start addr.	Unused	
	acum	0100	Input addr.	Output addr.	Cumulative threshold	Unused	
	genmasks	0101	Input addr.	Output addr.	Unused		
	findneuron	0110	Layer ID	Neuron position	Target neuron addr.	Unused	
	findrf	0111	Neuron addr.	Receptive field addr.	Unused		
Classification	**cls**	1000	Class path addr.	Activation path addr.	Result	Unused	
Others	Omitted for simplicity (**mov**, **dec**, **jne**, etc.)						

Backup

; inference & extraction of a model

```
for j = 1 to L {
  inf(j)
  <extraction for j>
}
```

↓

```
inf(1)
for j = 1 to L-1 {
  inf(j+1)
  <extraction for j>
}
<extraction for L>
```

(a) Overlapping inference with extraction across layers in forward extraction. L is the total number of DNN layers.

; extraction within a layer

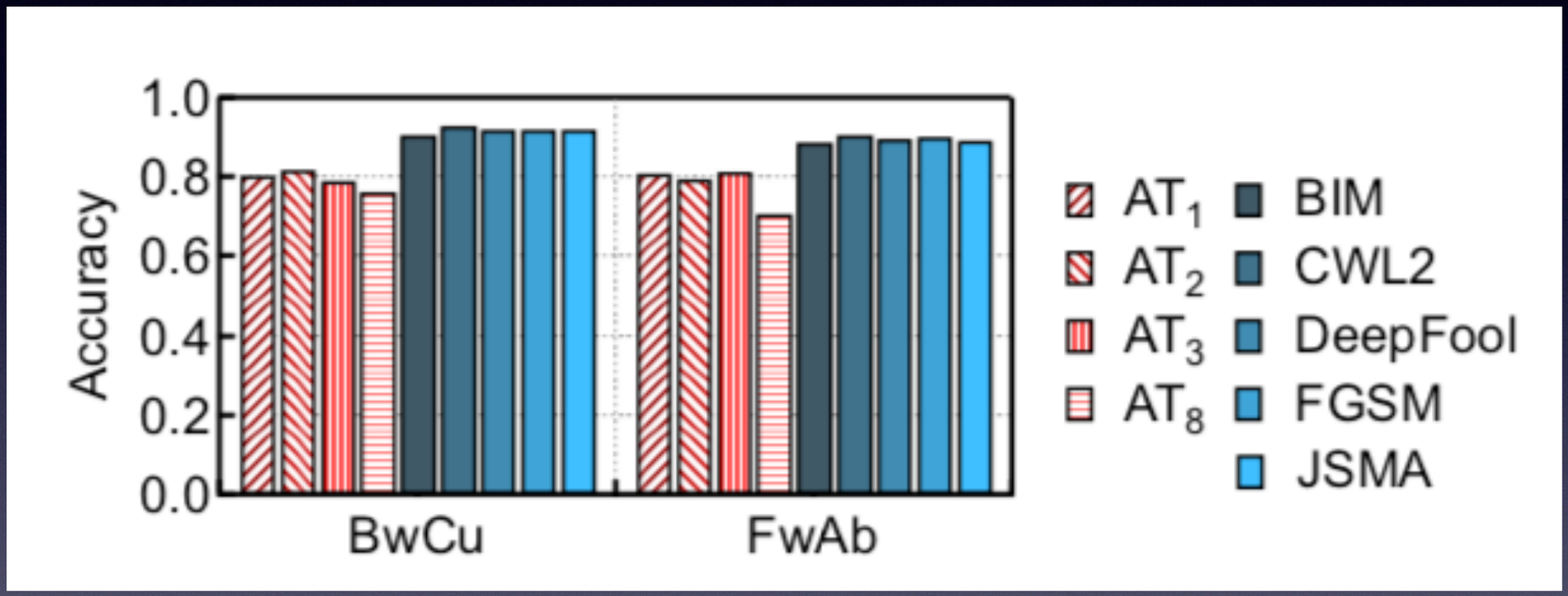
```
for i = 1 to N {
  sort(i)
  acum(i)
}
```

↓

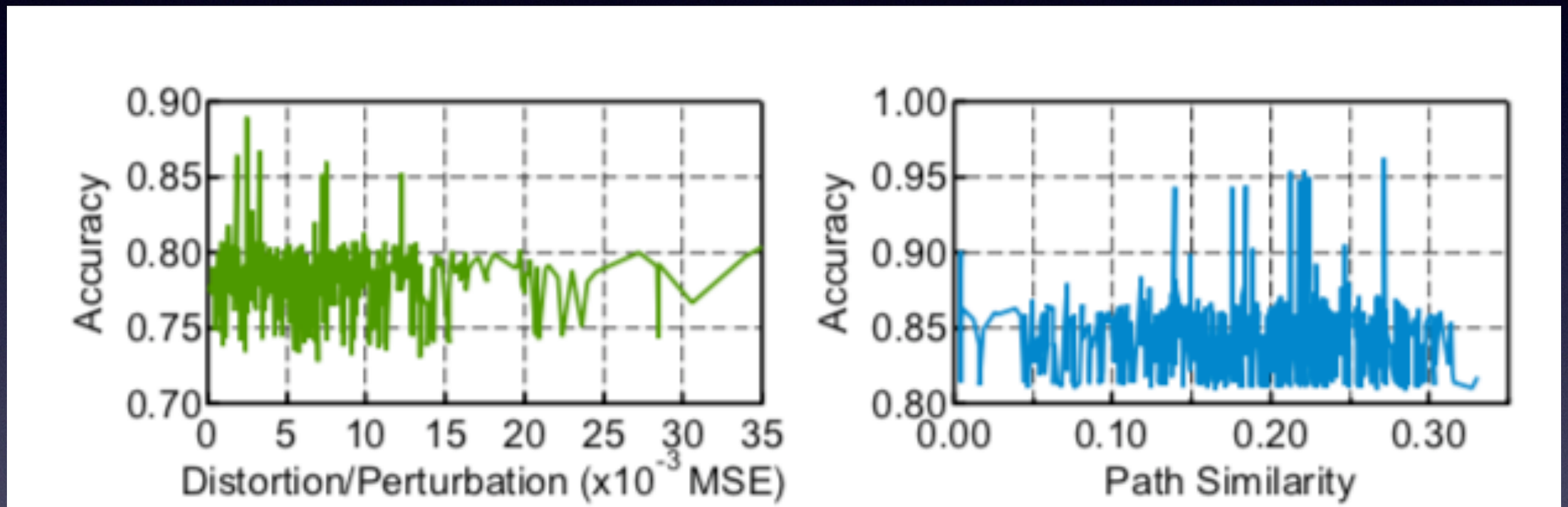
```
sort(1)
for i = 1 to N-1 {
  sort(i+1)
  acum(i)
}
acum(N)
```

(b) Neuron-level pipelining in important neuron extraction. N denotes the number of important neurons in the current layer's output.

Backup


1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
BwCu
FwAb
AT1
AT2
AT3
AT8
BIM
CWL2
DeepFool
FGSM
JSMA

Backup


0.90
0.85
0.80
0.75
0.70
Accuracy
0
5
10
15
20
25
30
35
Distortion/Perturbation (x10^-3 MSE)
1.00
0.95
0.90
0.85
0.80
Accuracy
0.00
0.10
0.20
0.30
Path Similarity

Backup