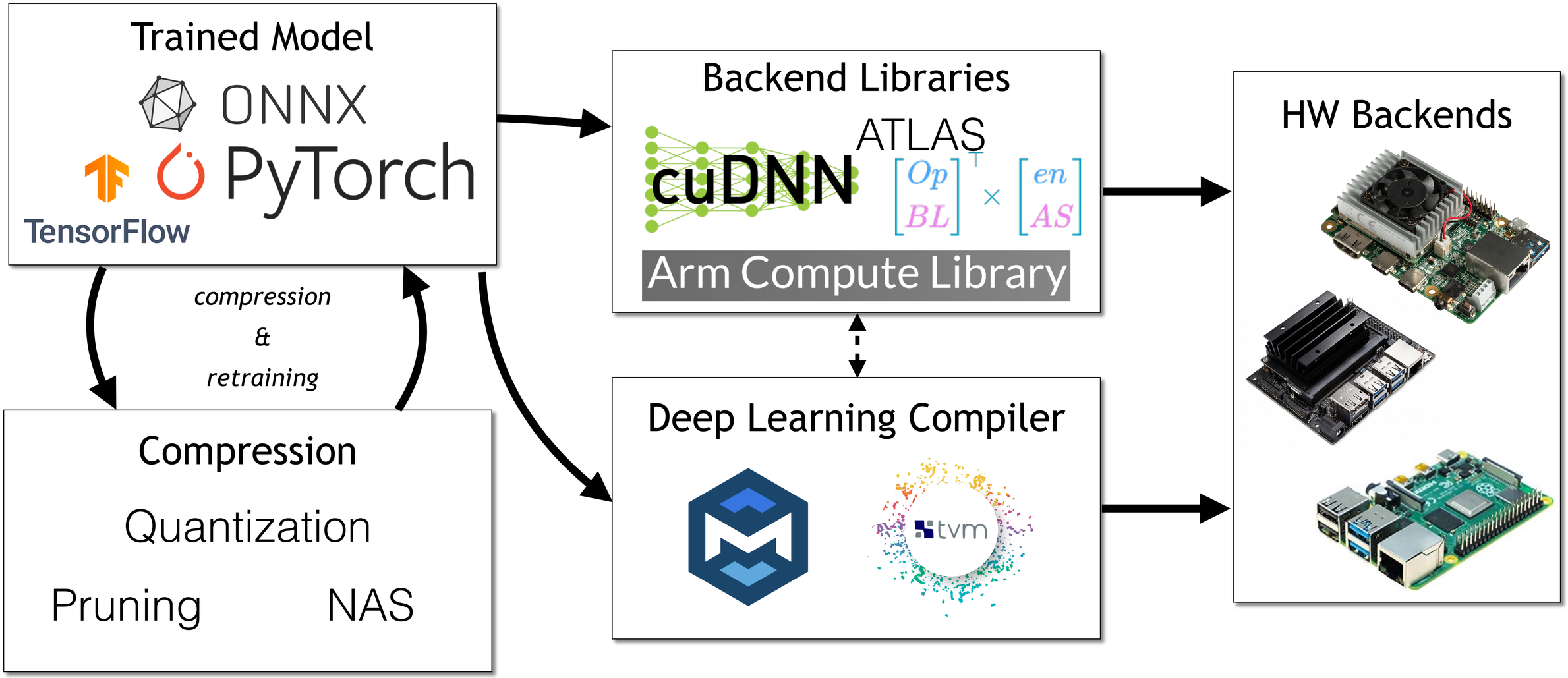
UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HARDWARE-AWARE DEPLOYMENT OF (PROBABILISTIC) NEURAL ARCHITECTURES

Bernhard Klein
Computing Systems Group
Institute of Computer Engineering
Heidelberg University

DEPLOYMENT CHAIN: DNNS & EMBEDDED SYSTEMS

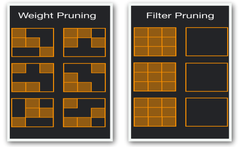


GALEN

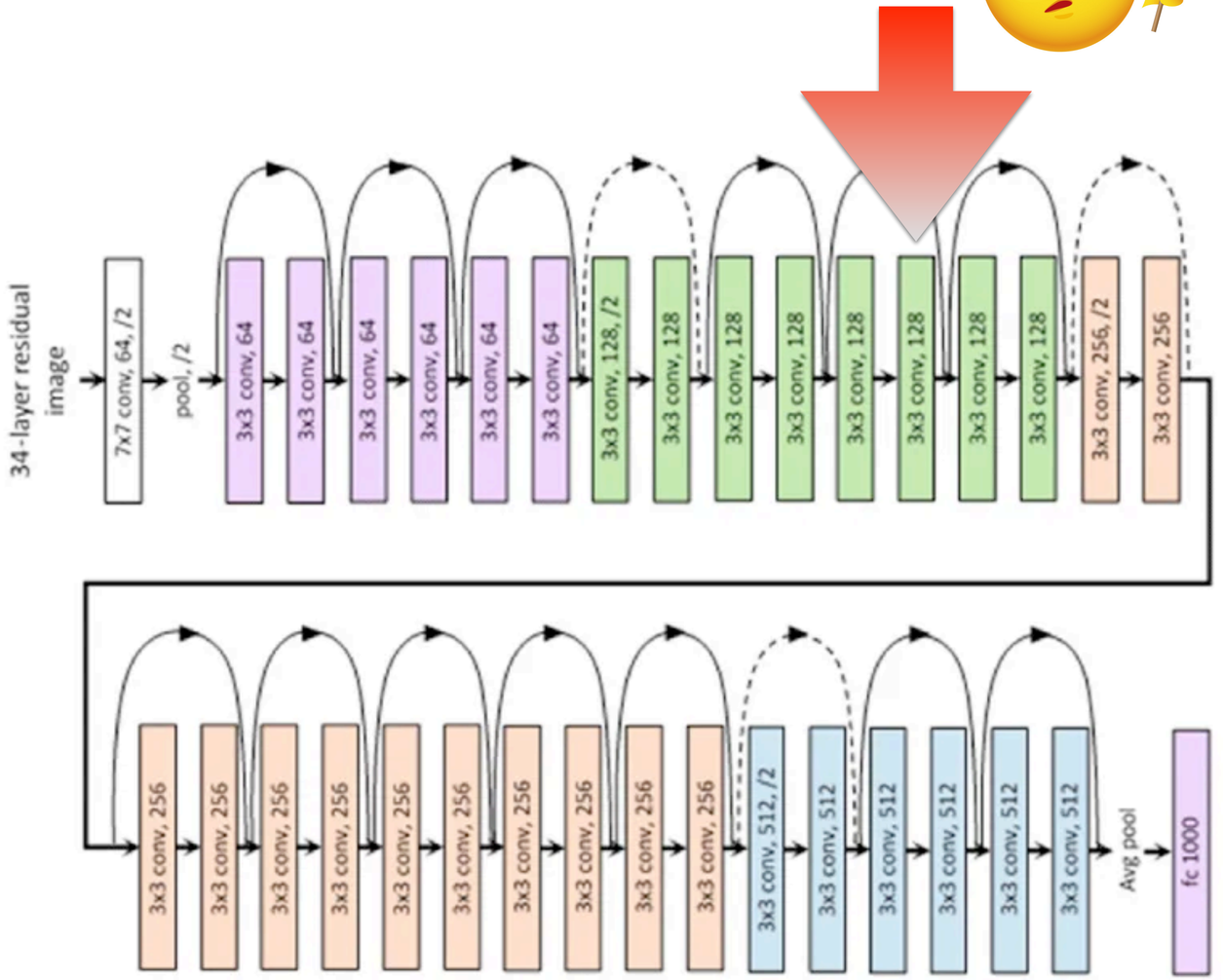
ARCHITECTURE-SPECIFIC AUTOMATIC COMPRESSION

COMPRESSION POLICY

- Quantization
  - Low bit width data types
- Pruning
  - Remove structures



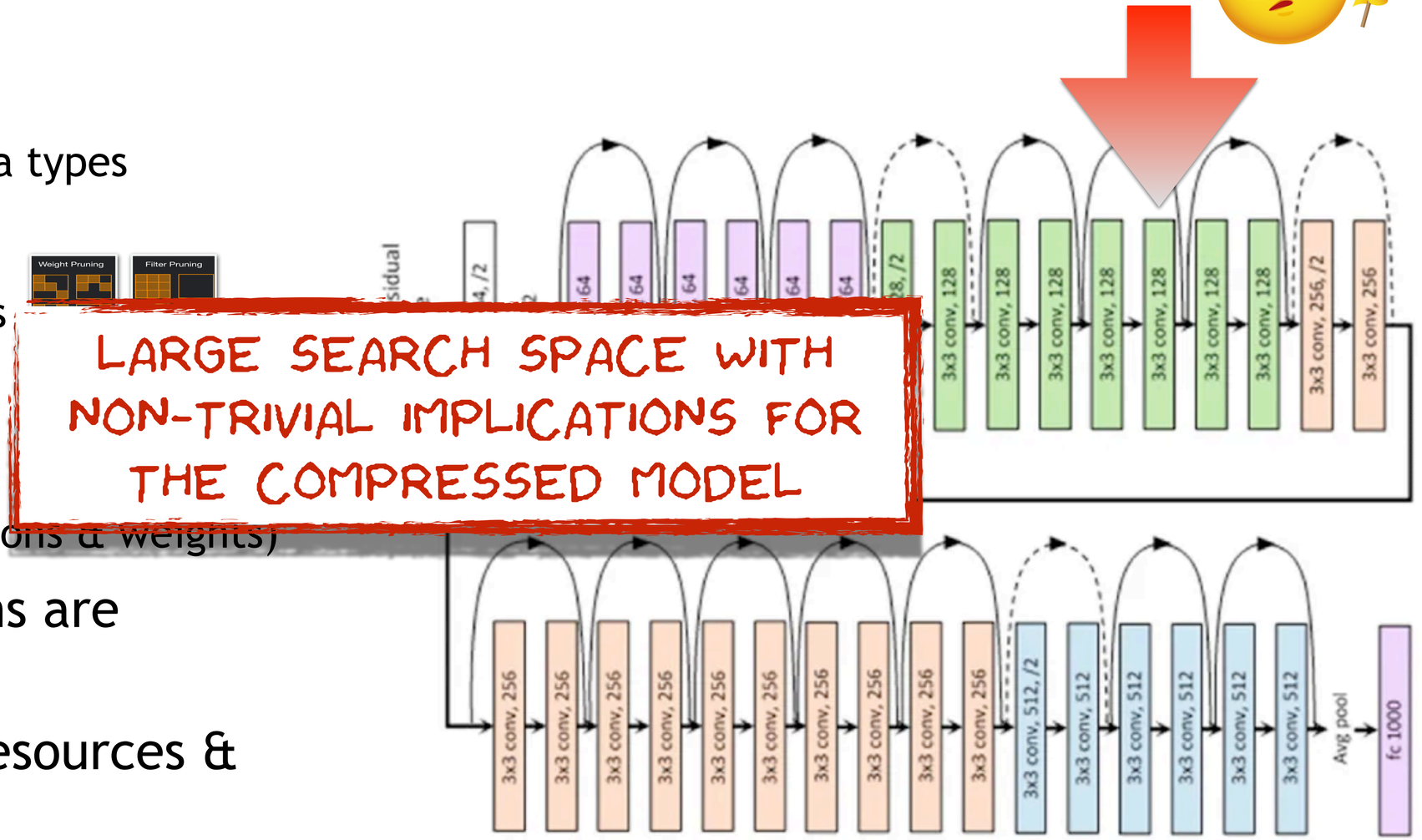
- Layer-specific
  - Sparsity
  - Bit width (activations & weights)
- Not all combinations are possible
- Affects hardware resources & model accuracy
  - Model & hardware expertise



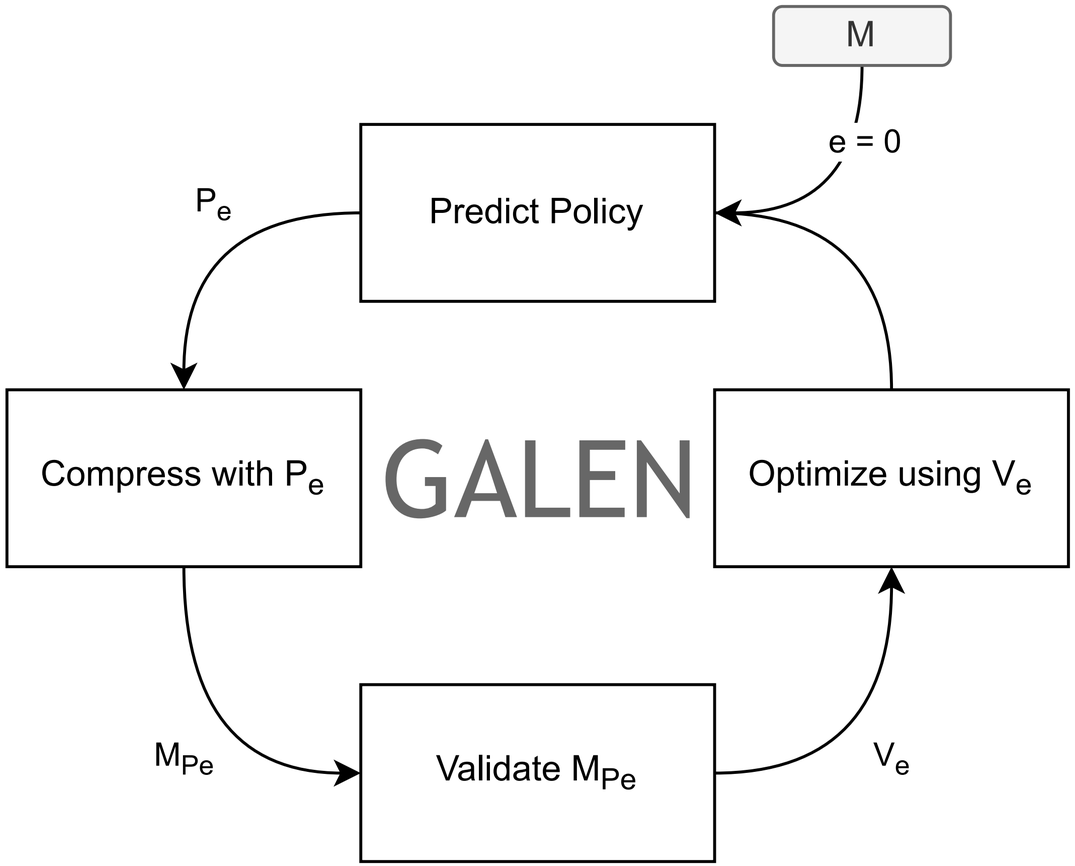
Siddhesh Bangar, Resnet Architecture Explained

COMPRESSION POLICY

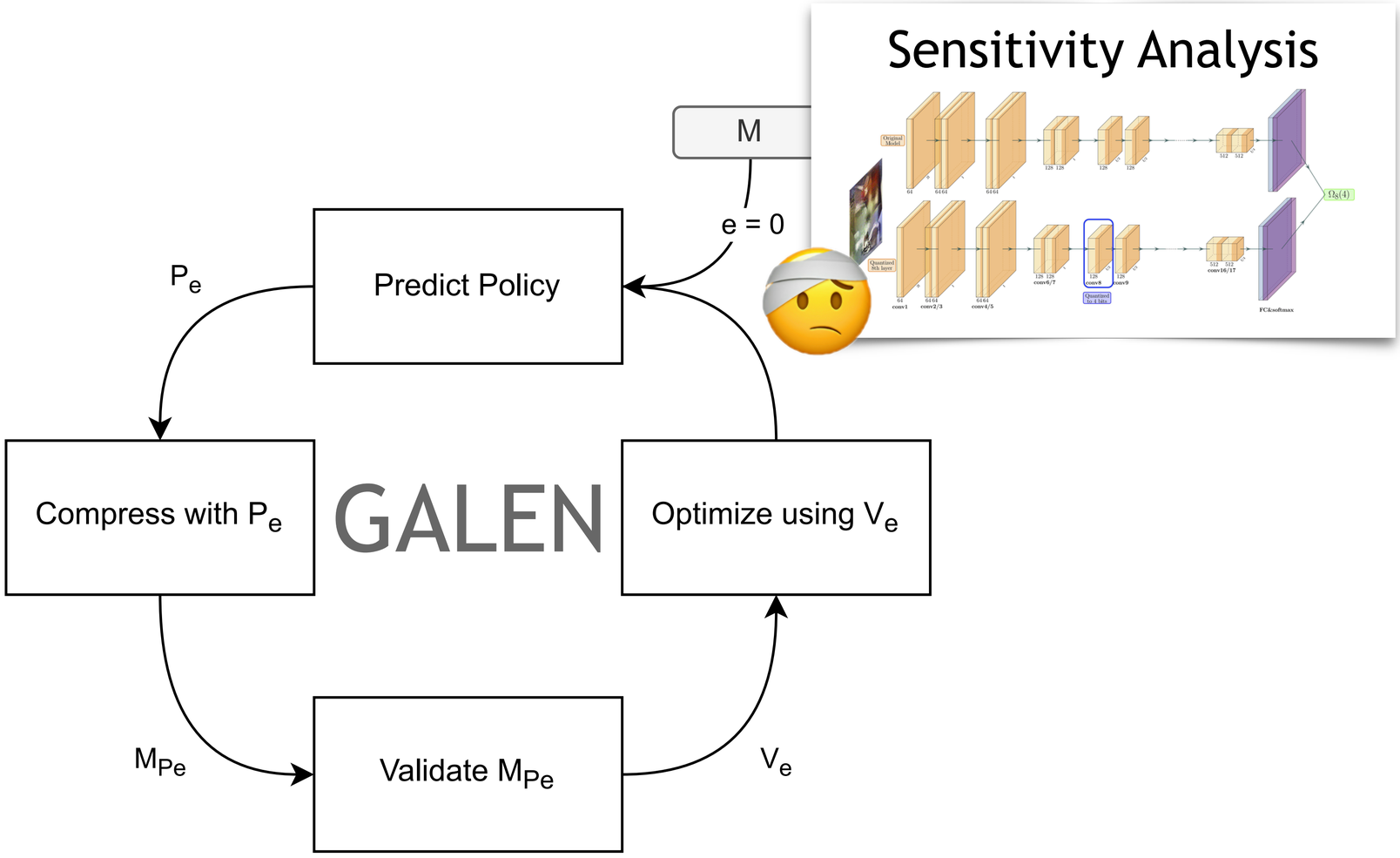
- Quantization
  - Low bit width data types
- Pruning
  - Remove structures
- Layer-specific
  - Sparsity
  - Bit width (activations & weights)
- Not all combinations are possible
- Affects hardware resources & model accuracy
  - Model & hardware expertise



Siddhesh Bangar, Resnet Architecture Explained



https://github.com/UniHD-CEG/galen

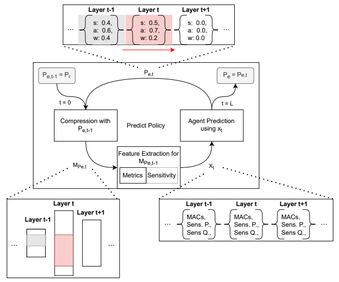


https://github.com/UniHD-CEG/galen

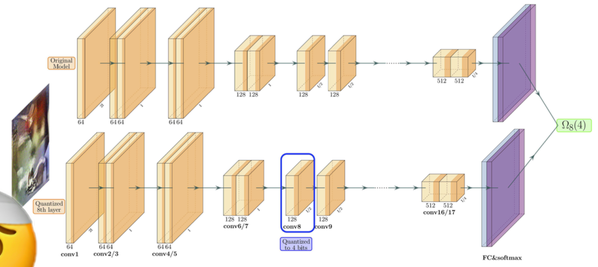
Predict Comp. Policies

Iterative

Guided
- Sensitivity
- MACs,BOPs



Sensitivity Analysis



M

e = 0

Predict Policy

Compress with P_e

GALEN

Optimize using V_e

M_{Pe}

Validate M_{Pe}

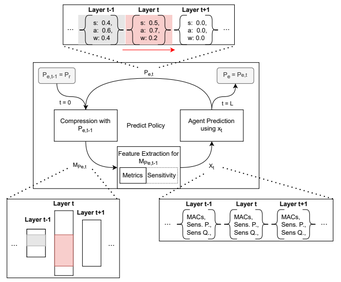
V_e

https://github.com/UniHD-CEG/galen

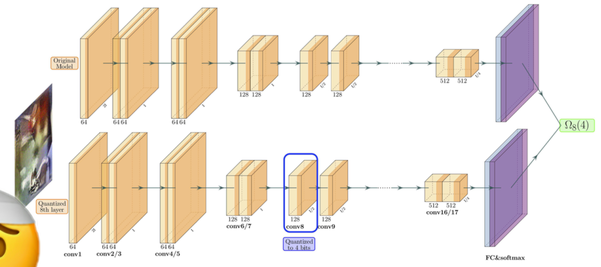
Predict Comp. Policies

Iterative

Guided
- Sensitivity
- MACs,BOPs



Sensitivity Analysis



M

e = 0

Predict Policy

Compress with P_e

GALEN

Optimize using V_e

M_{Pe}

Validate M_{Pe}

V_e

Pruning

Sparsity → Channel Count
Select channels with ℓ_1 strategy

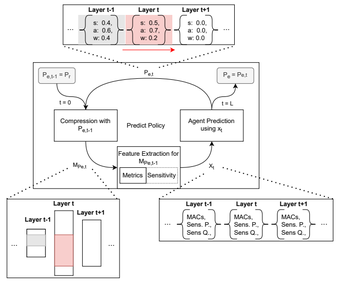
Track dependencies between layers with Torch-Pruning*

https://github.com/UniHD-CEG/galen

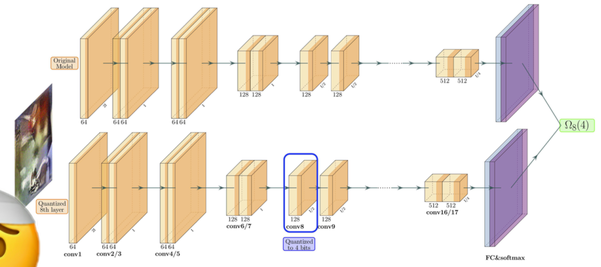
Predict Comp. Policies

Iterative

Guided
- Sensitivity
- MACs,BOPs



Sensitivity Analysis



M

e = 0

Predict Policy

Compress with P_e

GALEN

Optimize using V_e

M_{Pe}

Validate M_{Pe}

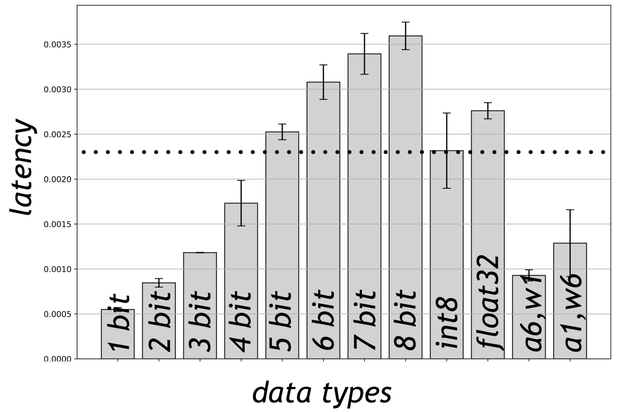
V_e

Pruning

Sparsity → Channel Count
Select channels with ℓ_1 strategy

Track dependencies between layers with Torch-Pruning*

Quantization

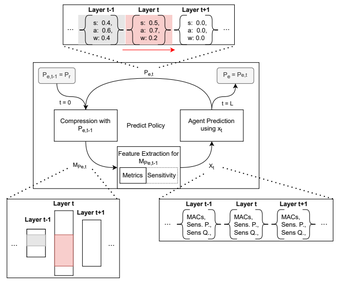


https://github.com/UniHD-CEG/galen

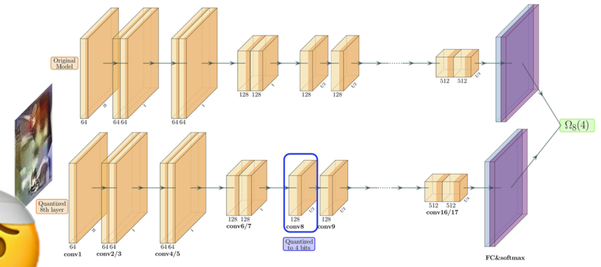
Predict Comp. Policies

Iterative

Guided
- Sensitivity
- MACs,BOPs



Sensitivity Analysis



M

e = 0

Predict Policy

Compress with P_e

GALEN

Optimize using V_e

M_{Pe}

Validate M_{Pe}

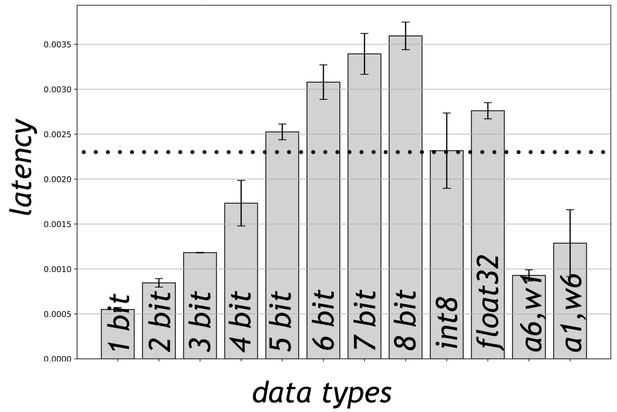
V_e

Pruning

Sparsity → Channel Count
Select channels with ℓ_1 strategy

Track dependencies between layers with Torch-Pruning*

Quantization



Evaluation

Accuracy
(1 ep. retraining)

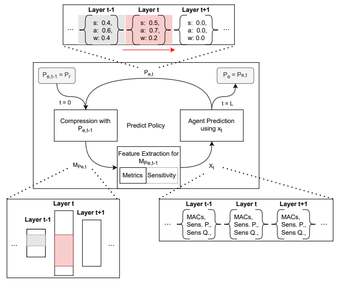
Latency Benchmark
(quantized operators)

https://github.com/UniHD-CEG/galen

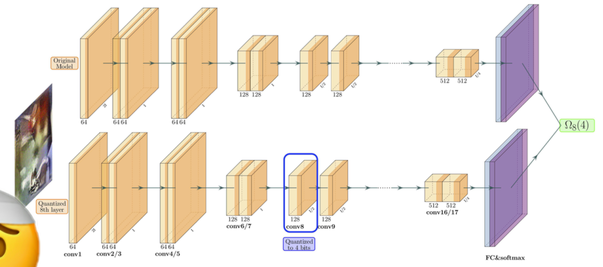
Predict Comp. Policies

Iterative

Guided
- Sensitivity
- MACs,BOPs



Sensitivity Analysis



M

e = 0

Predict Policy

Compress with P_e

GALEN

Optimize using V_e

M_{Pe}

Validate M_{Pe}

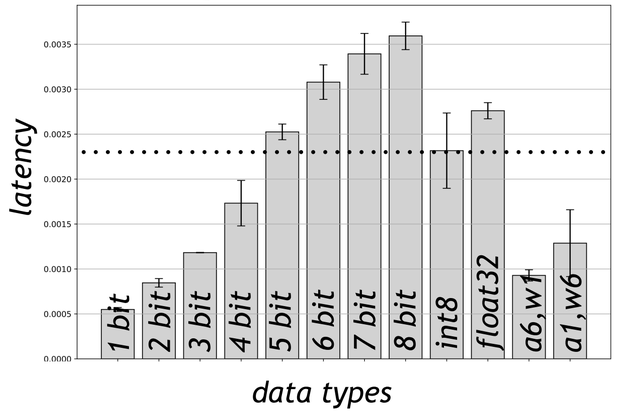
V_e

Pruning

Sparsity → Channel Count
Select channels with ℓ_1 strategy

Track dependencies between layers with Torch-Pruning*

Quantization



Absolute Reward Function

Bender, *et al.* 2020. Can Weight Sharing Outperform Random Architecture Search? An Investigation With TuNAS. In *IEEE/CVF CVPR.*

$$r(P) = acc_{M_P} + \beta \left| \frac{T_{\mathcal{M}_P}}{c \cdot T_{\mathcal{M}}} - 1 \right|$$

negative cost exponent

target latency

Evaluation

Accuracy
(1 ep. retraining)

Latency Benchmark
(quantized operators)

https://github.com/UniHD-CEG/galen

PERFORMANCE OF THE AGENTS

Three different RL agents

- Pruning minimizes MACs
- Quantization minimizes Bit Operations (BOPs)
- Joint approach most balanced

Performance for Various Target Latencies

Method	c	MACs	BOPs	Latency	Accuracy
Uncompressed		$4.75 \cdot 10^{10}$	$4.86 \cdot 10^{13}$	330 ms	93.0 %
Pruning Agent	0.3	$1.42 \cdot 10^{10}$	$1.45 \cdot 10^{13}$	98 ms	93.0 %
Quantization A.		$4.75 \cdot 10^{10}$	$8.23 \cdot 10^{11}$	98 ms	92.5 %
Joint Agent		$4.35 \cdot 10^{10}$	$9.42 \cdot 10^{11}$	99 ms	93.2 %
Pruning Agent	0.2	$9.24 \cdot 10^{9}$	$9.45 \cdot 10^{12}$	66 ms	92.4 %
Quantization A.		$4.75 \cdot 10^{10}$	$4.01 \cdot 10^{11}$	57 ms	45.0 %
Joint Agent		$2.82 \cdot 10^{10}$	$6.74 \cdot 10^{11}$	64 ms	92.8 %

PERFORMANCE OF THE AGENTS

Three different RL agents

- Pruning minimizes MACs
- Quantization minimizes Bit Operations (BOPs)
- Joint approach most balanced

Performance for Various Target Latencies

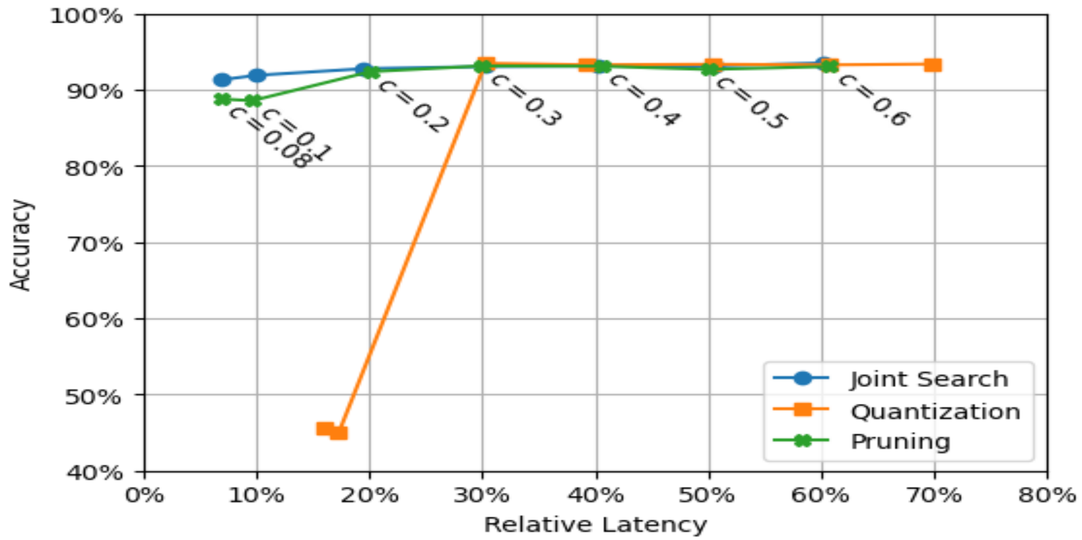
Method	c	MACs	BOPs	Latency	Accuracy
Uncompressed		$4.75 \cdot 10^{10}$	$4.86 \cdot 10^{13}$	330 ms	93.0 %
Pruning Agent	0.3	$1.42 \cdot 10^{10}$	$1.45 \cdot 10^{13}$	98 ms	93.0 %
Quantization A.		$4.75 \cdot 10^{10}$	$8.23 \cdot 10^{11}$	98 ms	92.5 %
Joint Agent		$4.35 \cdot 10^{10}$	$9.42 \cdot 10^{11}$	99 ms	93.2 %
Pruning Agent	0.2	$9.24 \cdot 10^{9}$	$9.45 \cdot 10^{12}$	66 ms	92.4 %
Quantization A.		$4.75 \cdot 10^{10}$	$4.01 \cdot 10^{11}$	57 ms	45.0 %
Joint Agent		$2.82 \cdot 10^{10}$	$6.74 \cdot 10^{11}$	64 ms	92.8 %

PERFORMANCE OF THE AGENTS

Three different RL agents

- Pruning minimizes MACs
- Quantization minimizes Bit Operations (BOPs)
- Joint approach most balanced

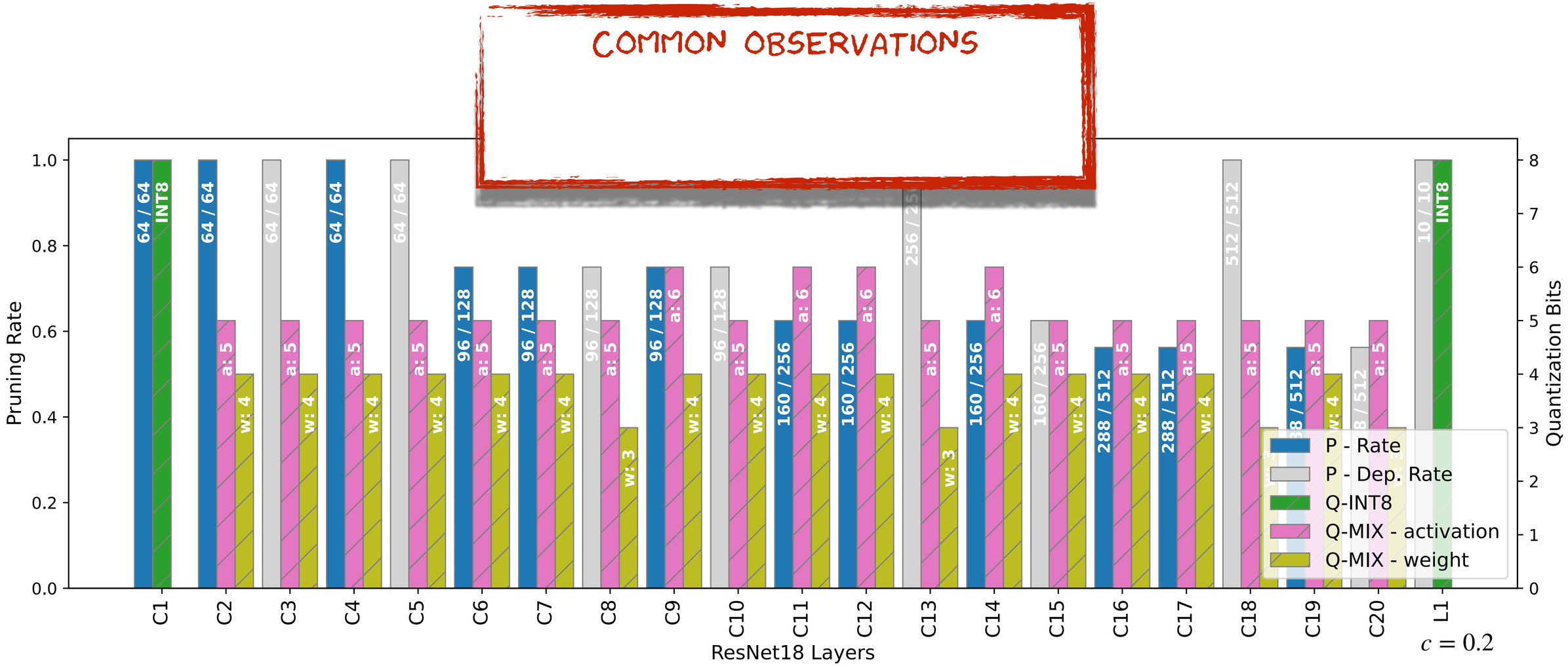
Method	c	MACs	BOPs	Latency	Accuracy
Uncompressed		$4.75 \cdot 10^{10}$	$4.86 \cdot 10^{13}$	330 ms	93.0 %
Pruning Agent	0.3	$1.42 \cdot 10^{10}$	$1.45 \cdot 10^{13}$	98 ms	93.0 %
Quantization A.		$4.75 \cdot 10^{10}$	$8.23 \cdot 10^{11}$	98 ms	92.5 %
Joint Agent		$4.35 \cdot 10^{10}$	$9.42 \cdot 10^{11}$	99 ms	93.2 %
Pruning Agent	0.2	$9.24 \cdot 10^{9}$	$9.45 \cdot 10^{12}$	66 ms	92.4 %
Quantization A.		$4.75 \cdot 10^{10}$	$4.01 \cdot 10^{11}$	57 ms	45.0 %
Joint Agent		$2.82 \cdot 10^{10}$	$6.74 \cdot 10^{11}$	64 ms	92.8 %



Performance for Various Target Latencies

CHALLENGING TARGET LATENCIES FORCE THE AGENT TOWARDS EXTREME COMPRESSION RESULTING IN ACCURACY DROPS

CAN WE UNDERSTAND A PREDICTED POLICY?

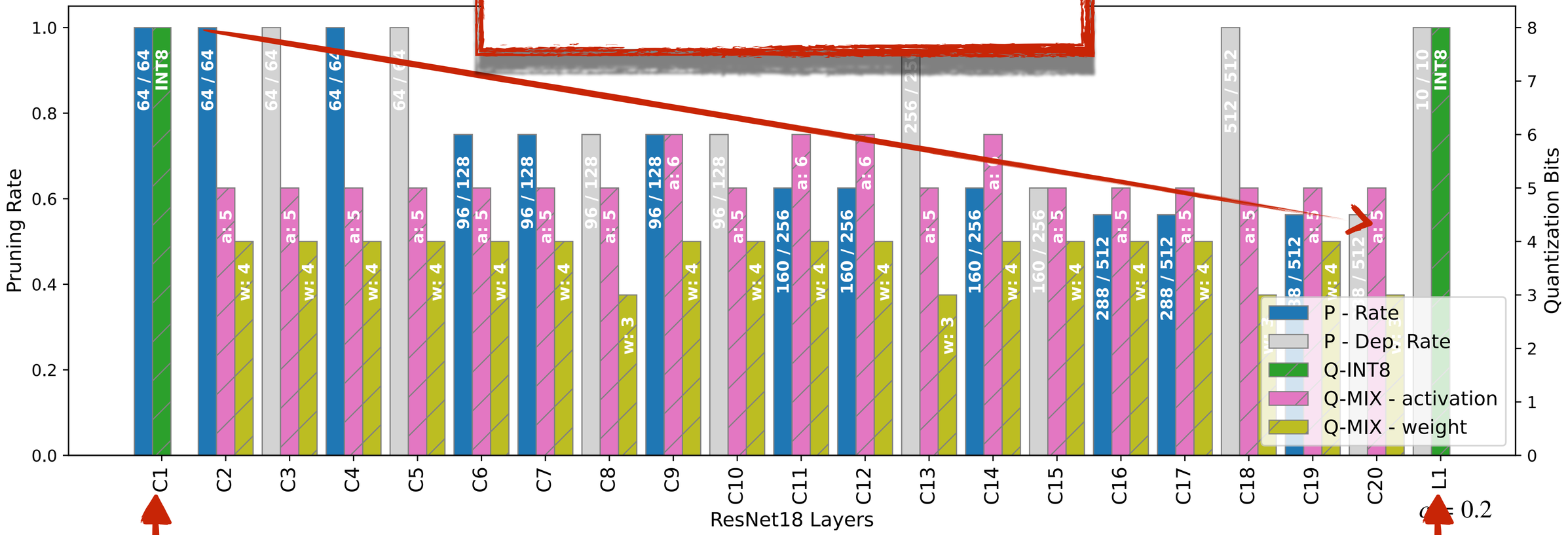


Policy of the joint agent

CAN WE UNDERSTAND A PREDICTED POLICY?

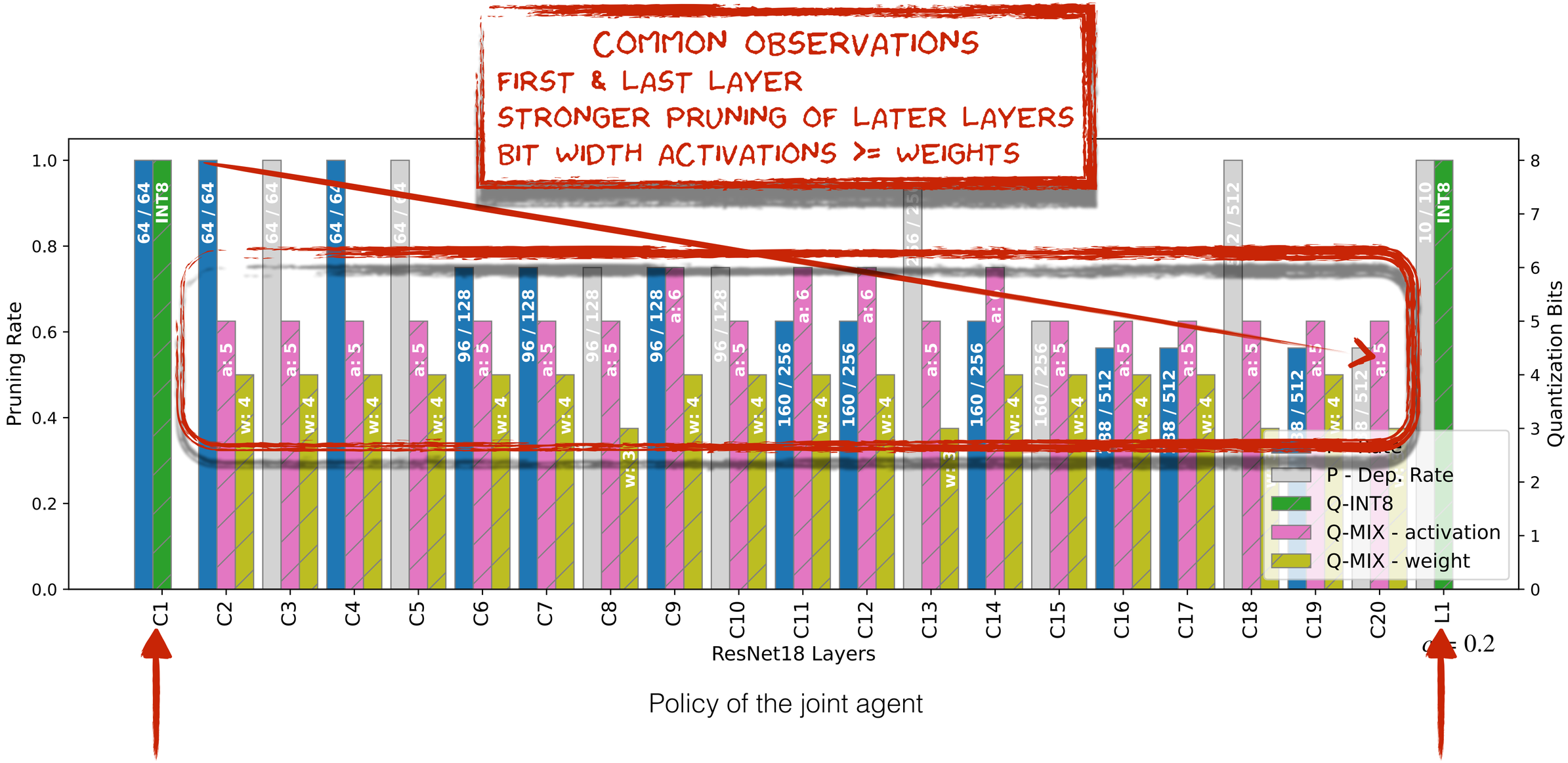
CAN WE UNDERSTAND A PREDICTED POLICY?

COMMON OBSERVATIONS
FIRST & LAST LAYER
STRONGER PRUNING OF LATER LAYERS



Policy of the joint agent

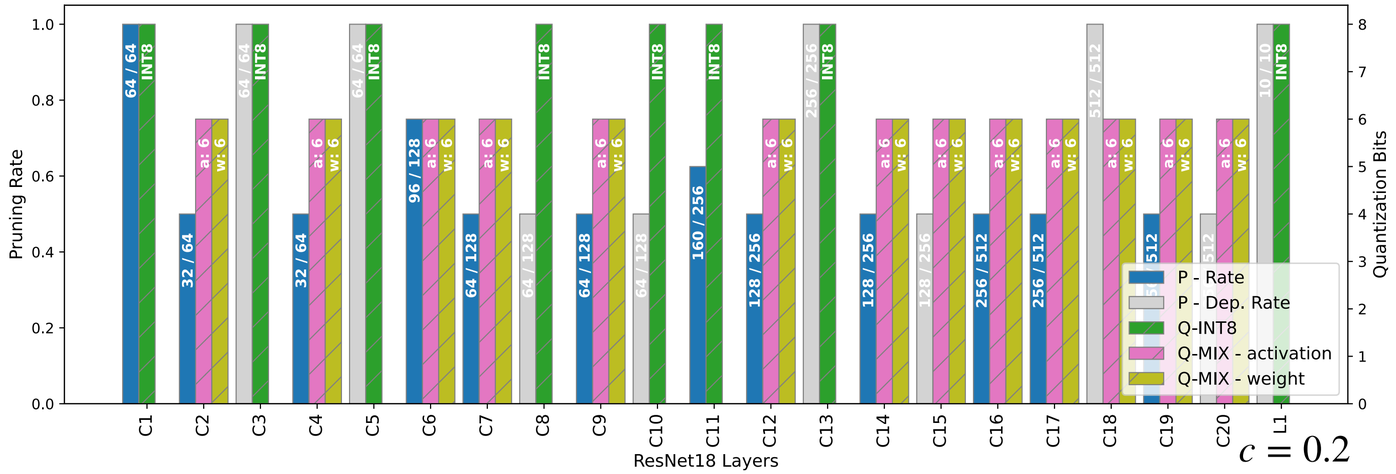
CAN WE UNDERSTAND A PREDICTED POLICY?



IS THE SENSITIVITY ANALYSIS NECESSARY?

Without sensitivity analysis

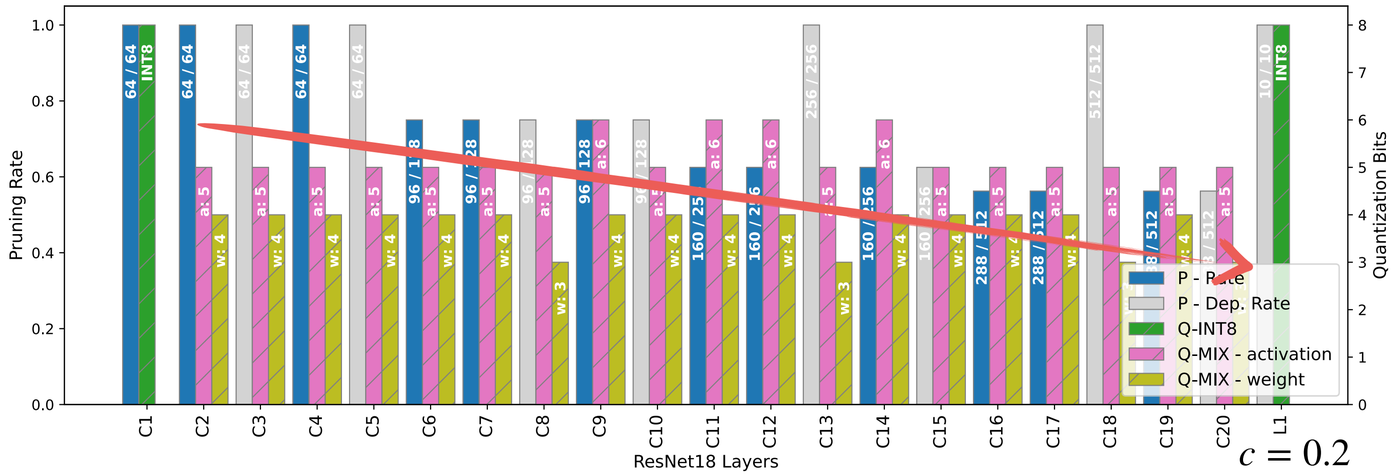
- Homogeneous pruning
- Minimal quantization



Policy **without sensitivity** feature

With sensitivity information

- **Heterogeneous compression**
- Activations ≠ weight precision
- Stronger compression in the last layers

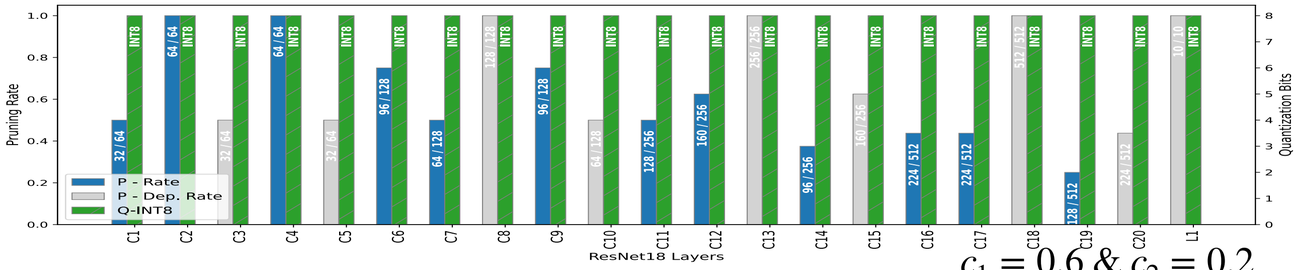


Policy with **sensitivity** feature

IS A CONCURRENT JOINT SEARCH BENEFICIAL?

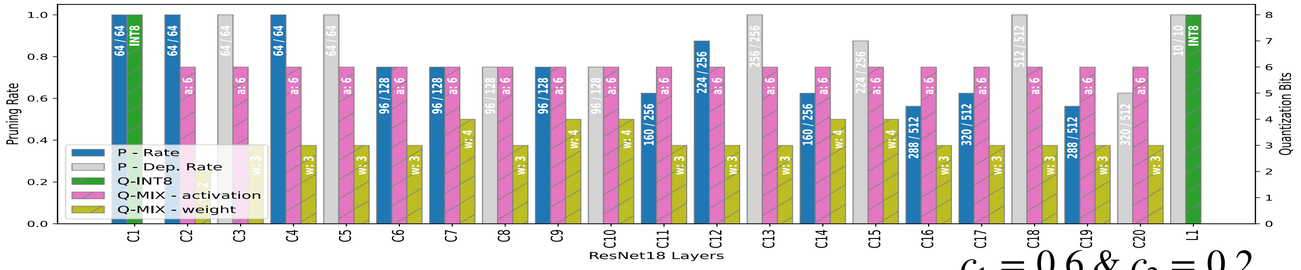
Subsequent searches use latter compression more aggressively

Joint search is more balanced



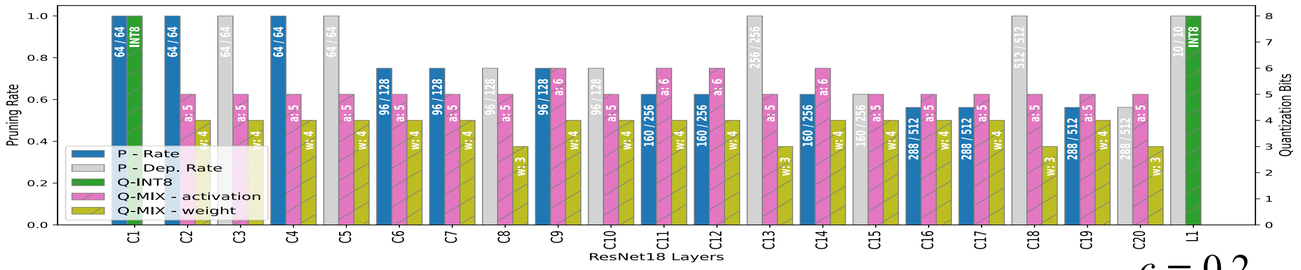
$c_1 = 0.6 \;\&\; c_2 = 0.2$

First quantization, then pruning search



$c_1 = 0.6 \;\&\; c_2 = 0.2$

First pruning, then quantization search

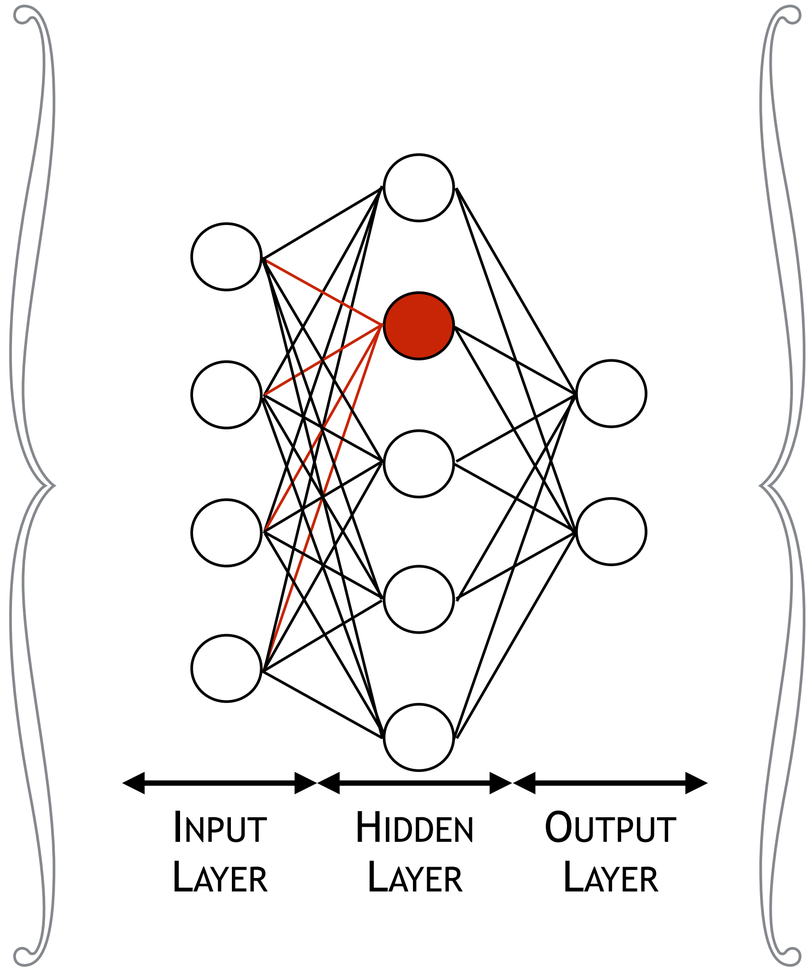


$c = 0.2$

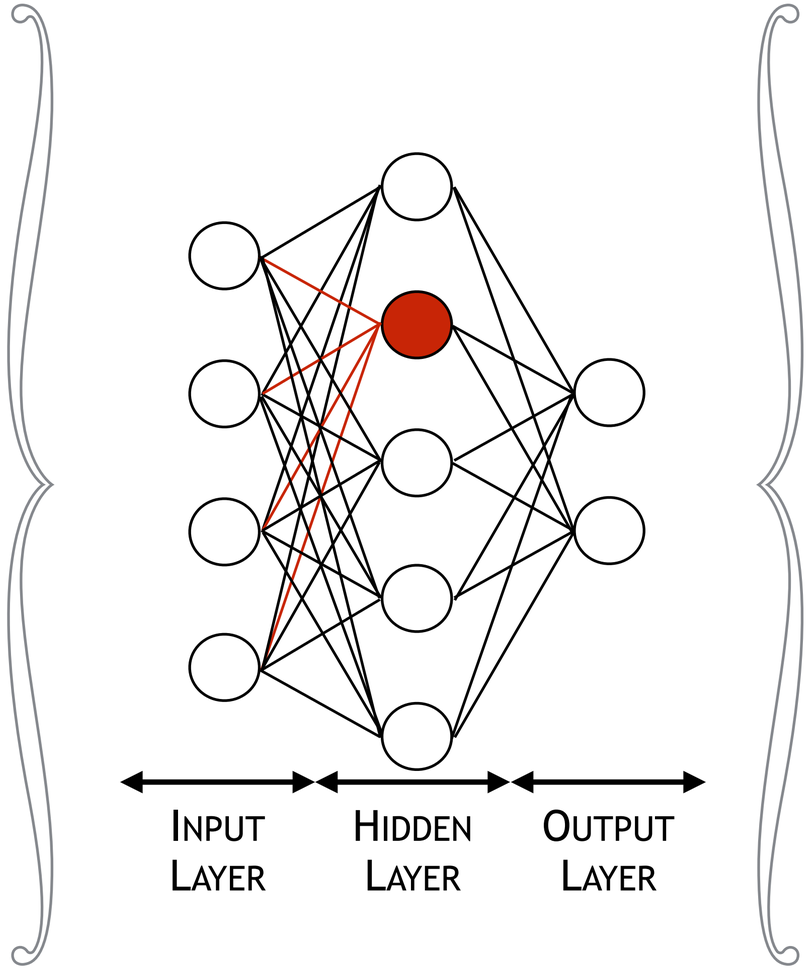
Joint search

BAYESIAN NEURAL NETWORKS

OUT-OF-DOMAIN DATA

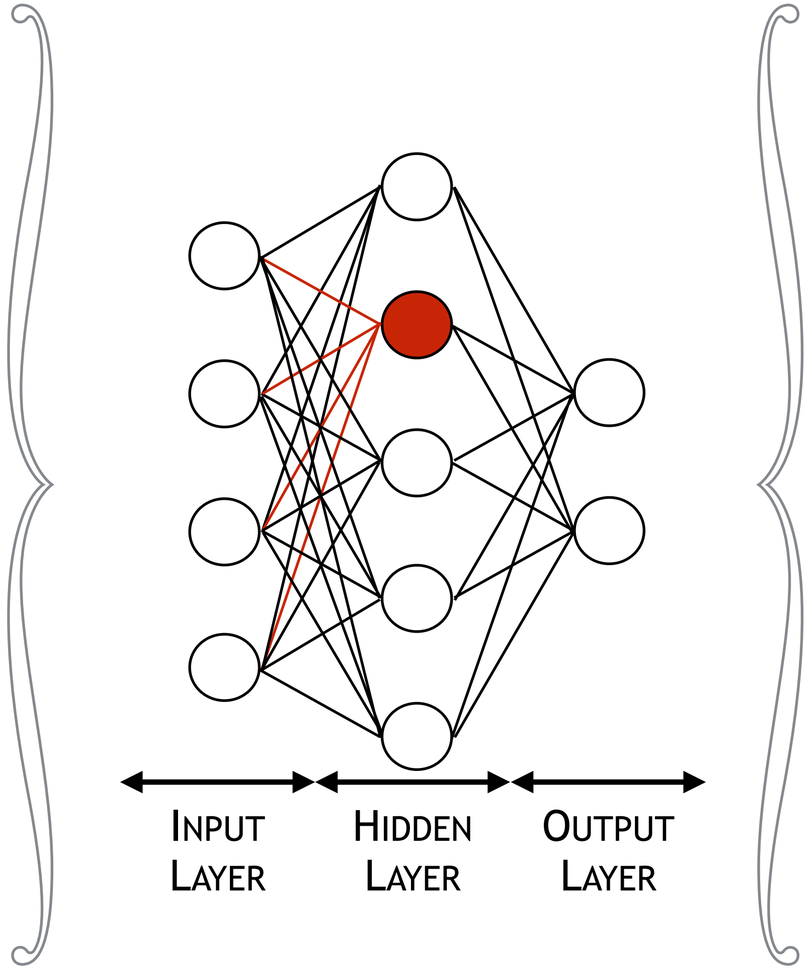


OUT-OF-DOMAIN DATA



3

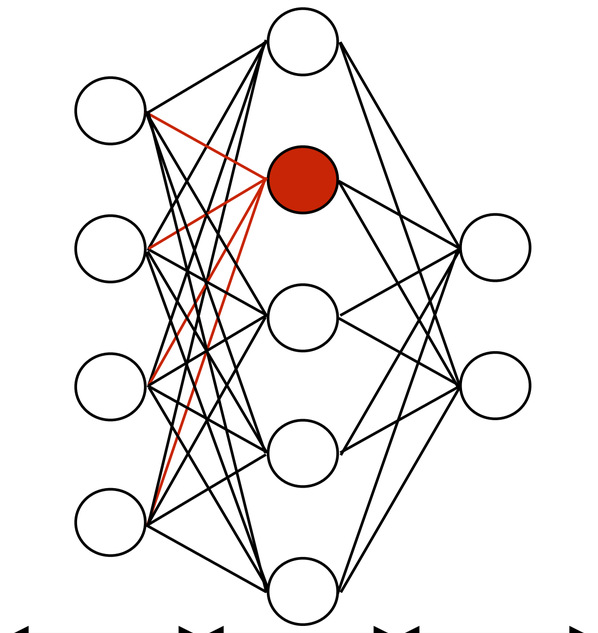
OUT-OF-DOMAIN DATA



3

5

OUT-OF-DOMAIN DATA

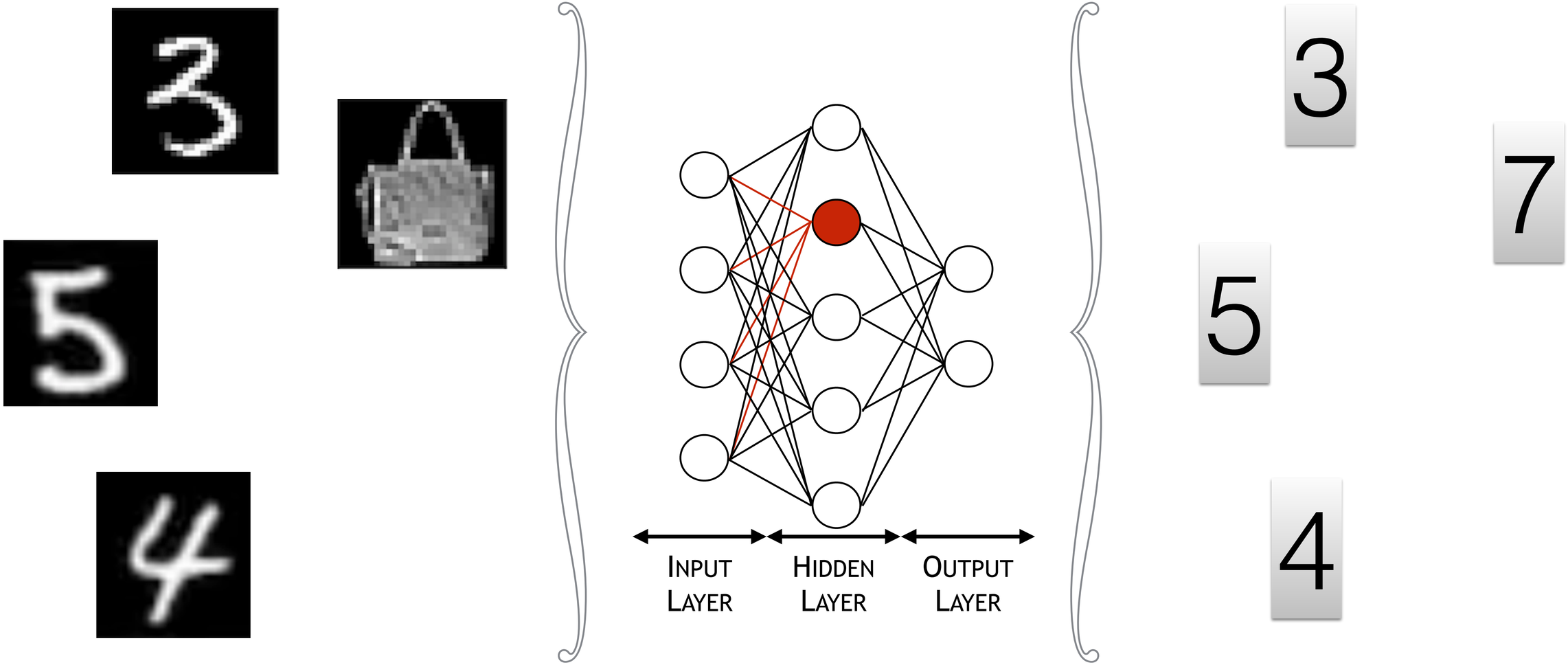


3

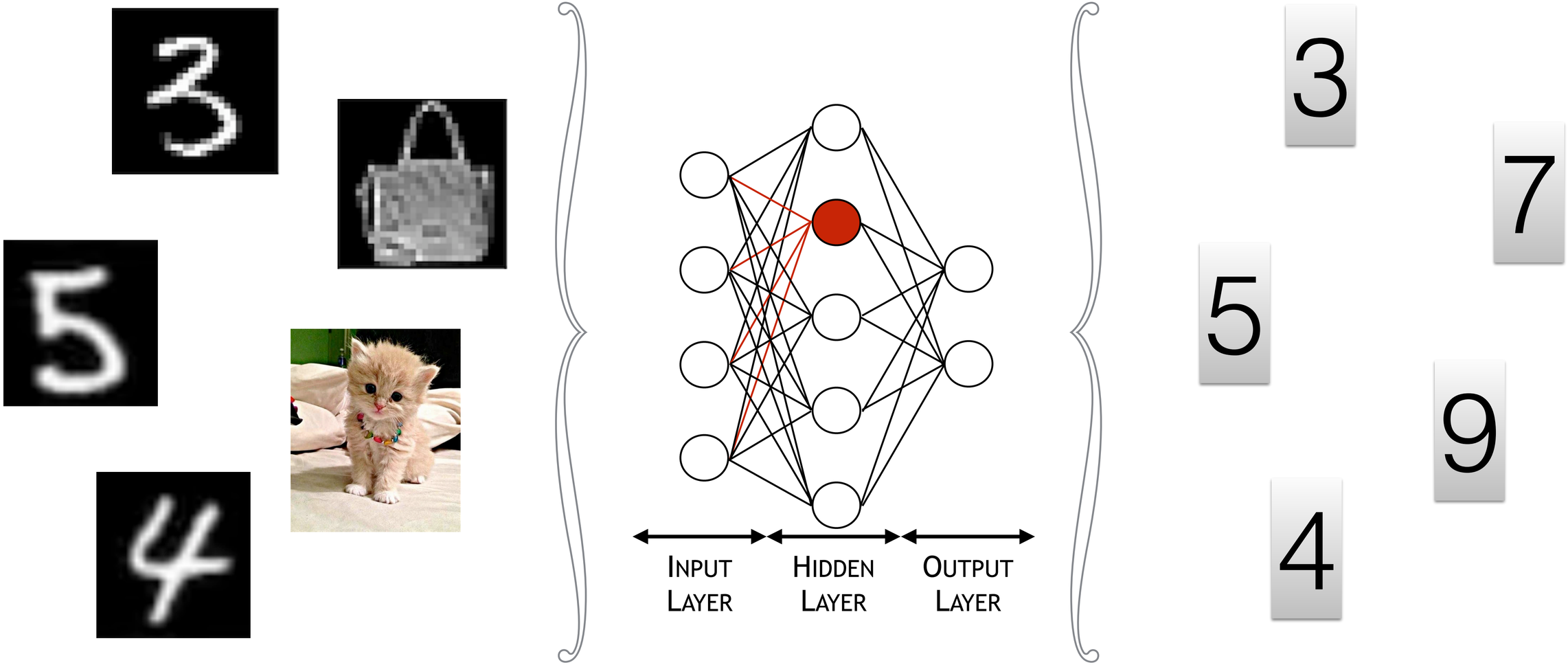
5

4

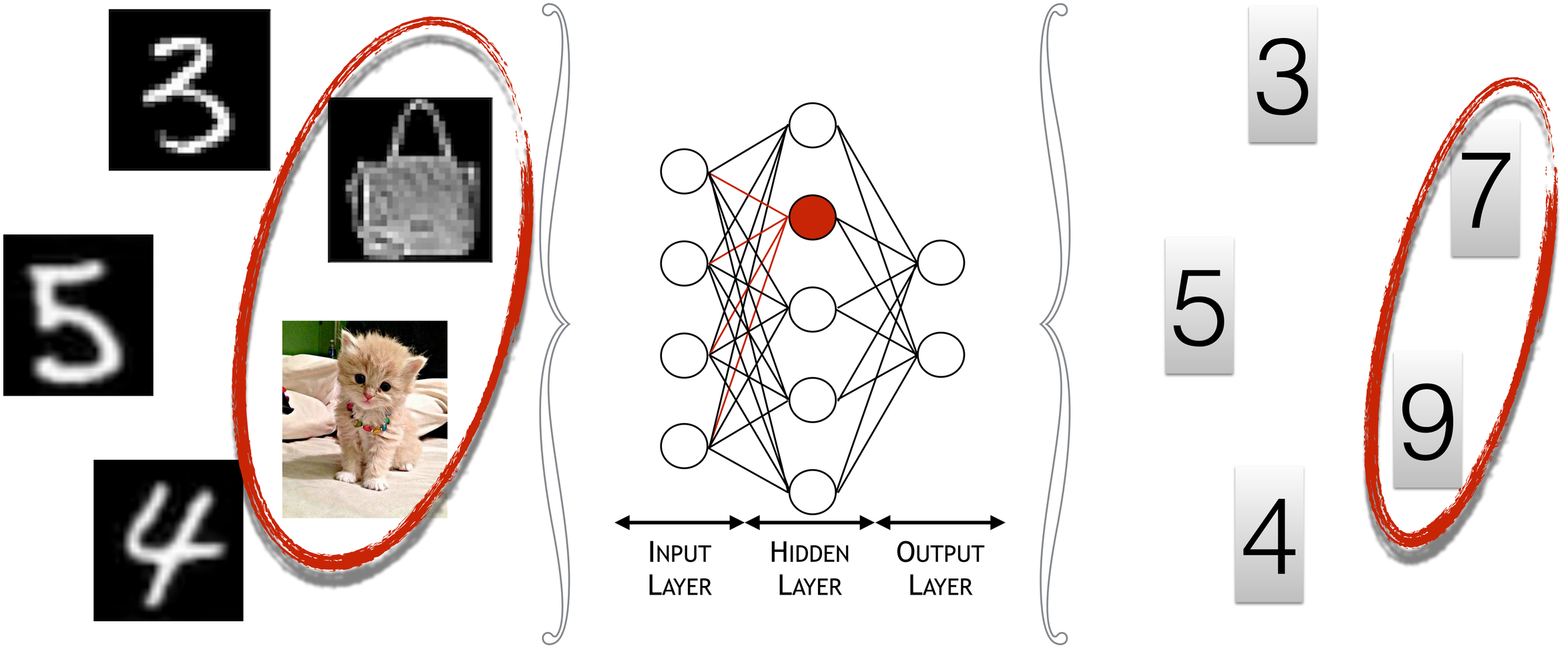
OUT-OF-DOMAIN DATA



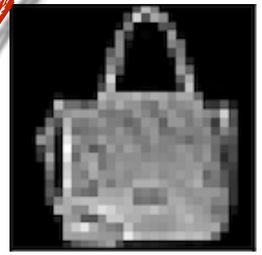
OUT-OF-DOMAIN DATA

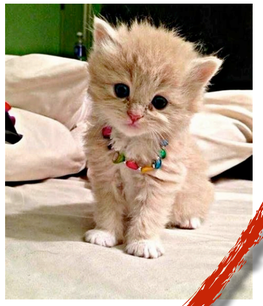


OUT-OF-DOMAIN DATA



OUT-OF-DOMAIN DATA

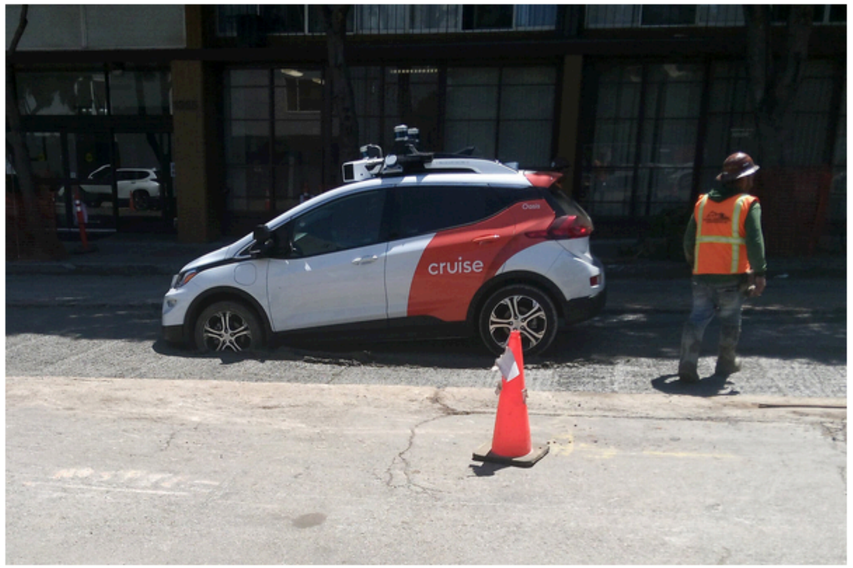


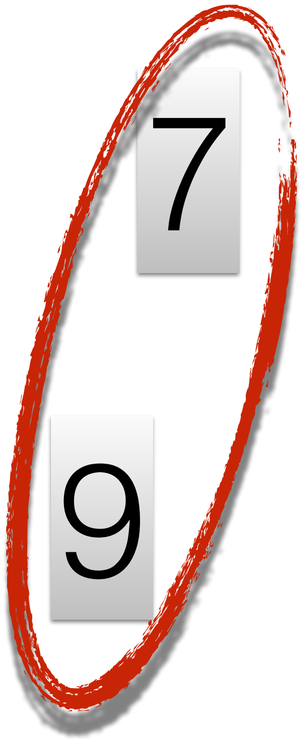


Driverless Car Gets Stuck in Wet Concrete in San Francisco

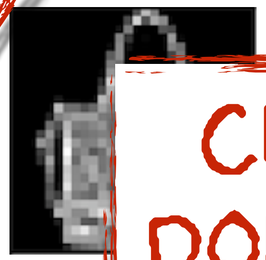
Though driverless cars have not been blamed for any serious injuries or crashes in the city, they have been involved in several jarring episodes.

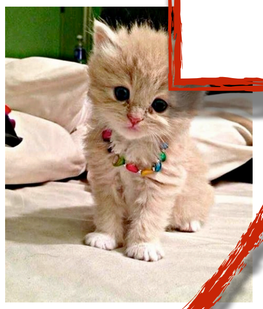
Share full article 376





OUT-OF-DOMAIN DATA





Driverless Car Gets Stuck in Wet Concrete in San Francisco

Though driverless cars have not been blamed for any serious injuries or crashes in the city, they have been involved in several

CLASSICAL NEURAL NETWORKS DON'T HAVE THE ABILITY TO SAY - I DON'T KNOW -

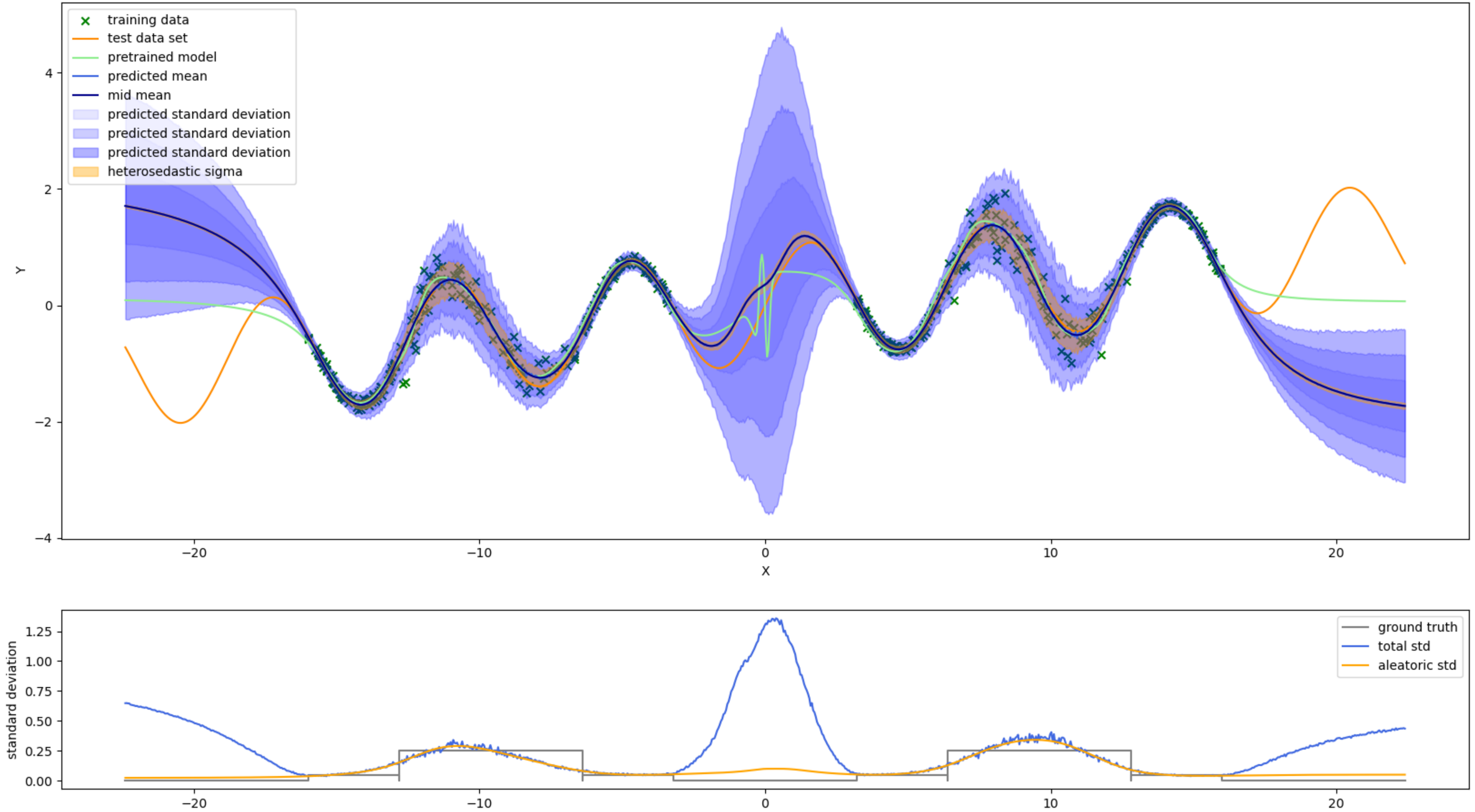
3

7

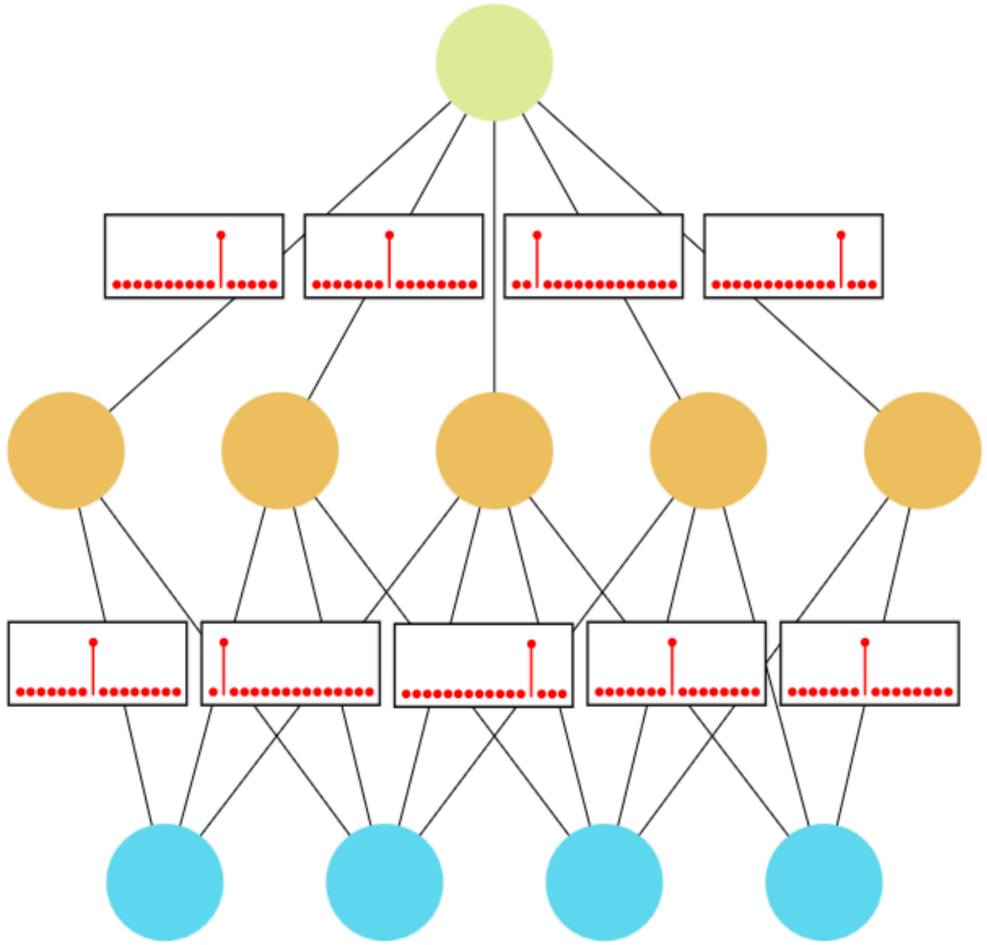
9

4

UNCERTAINTY PREDICTION

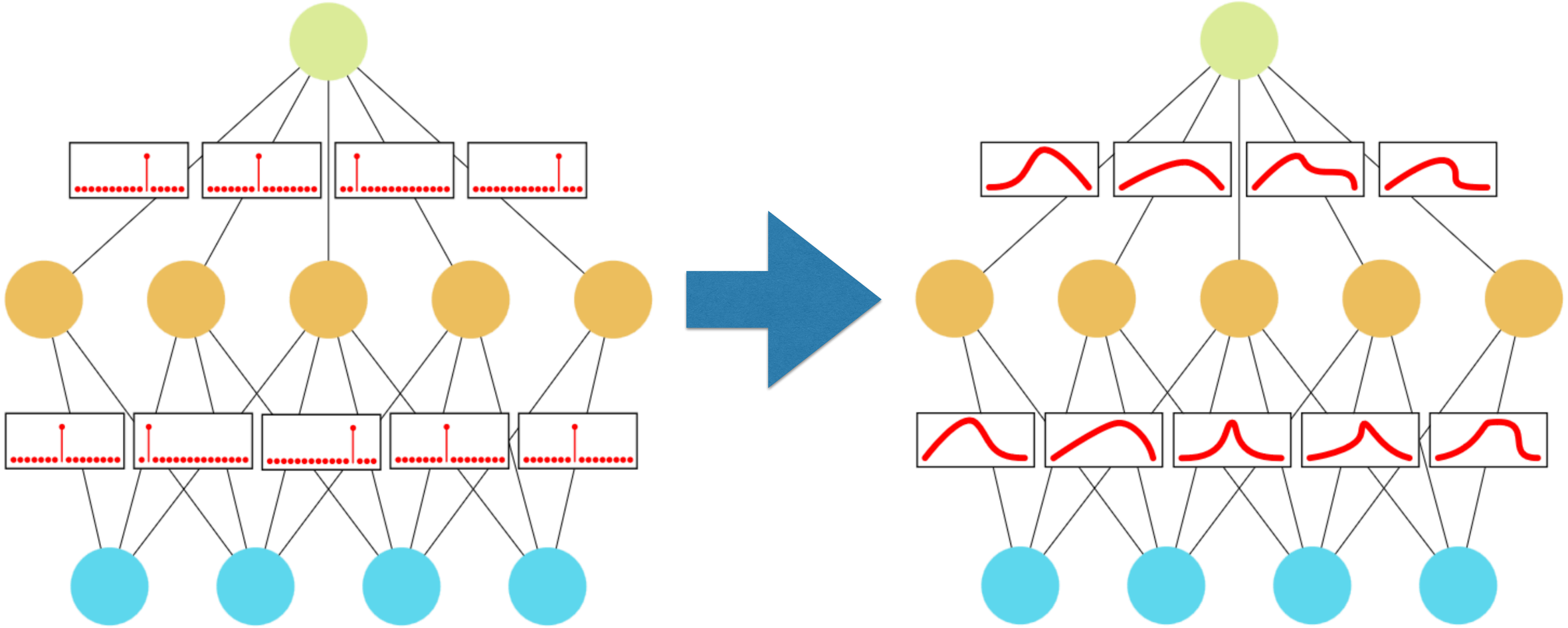


BAYESIAN NEURAL NETWORKS



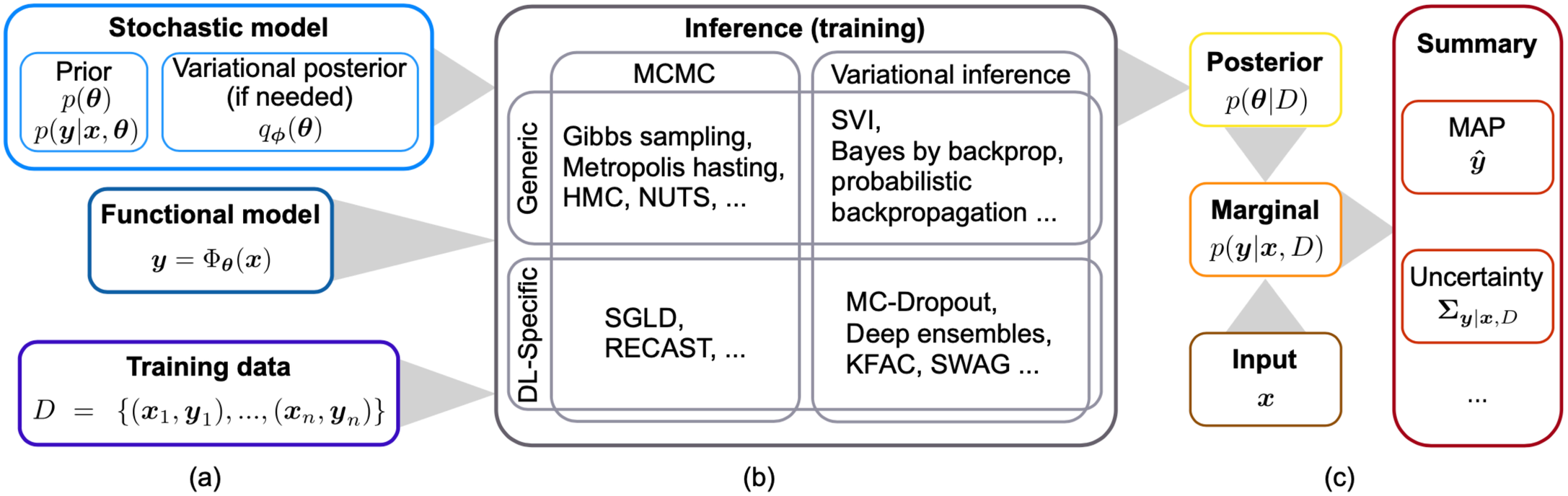
Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

BAYESIAN NEURAL NETWORKS



Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

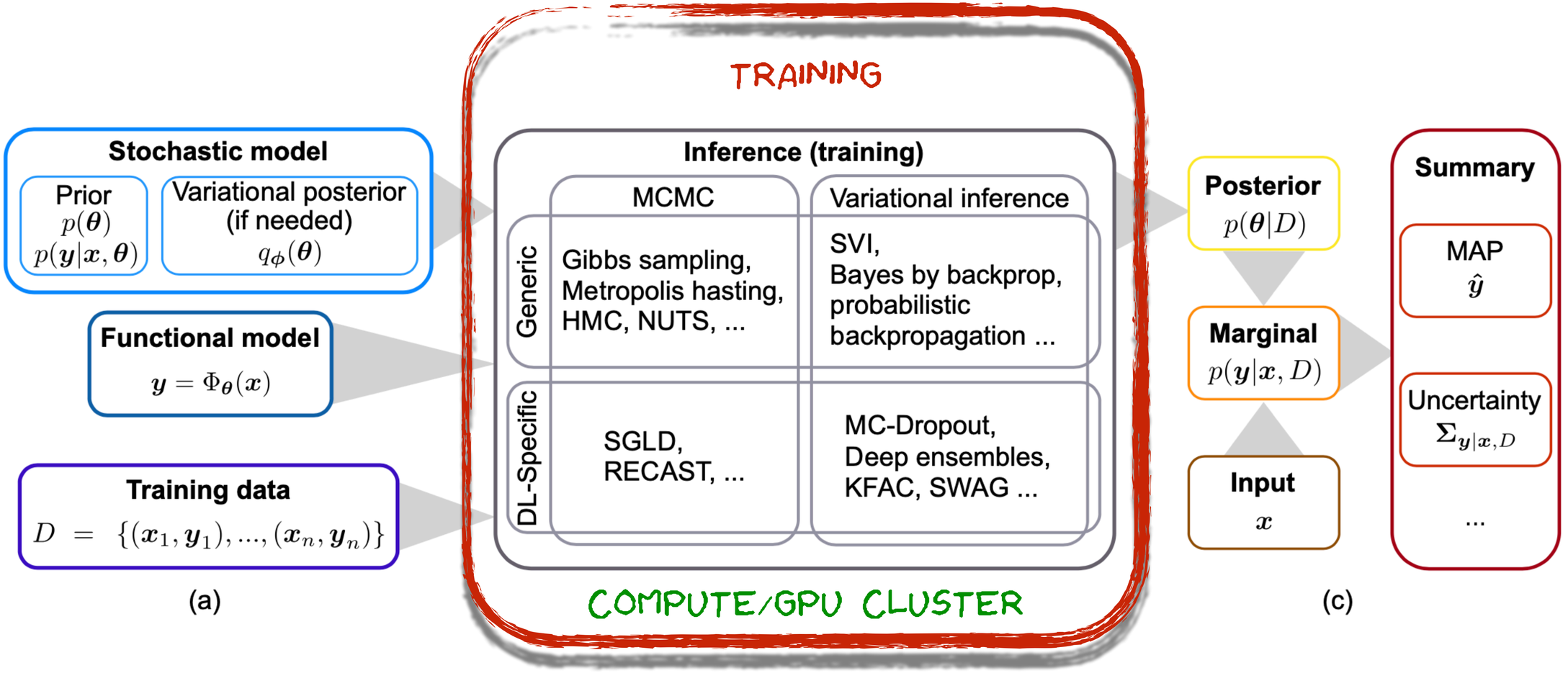
BNN WORKFLOW



Workflow to design (a), train (b) and use a BNN for predictions (c)

Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

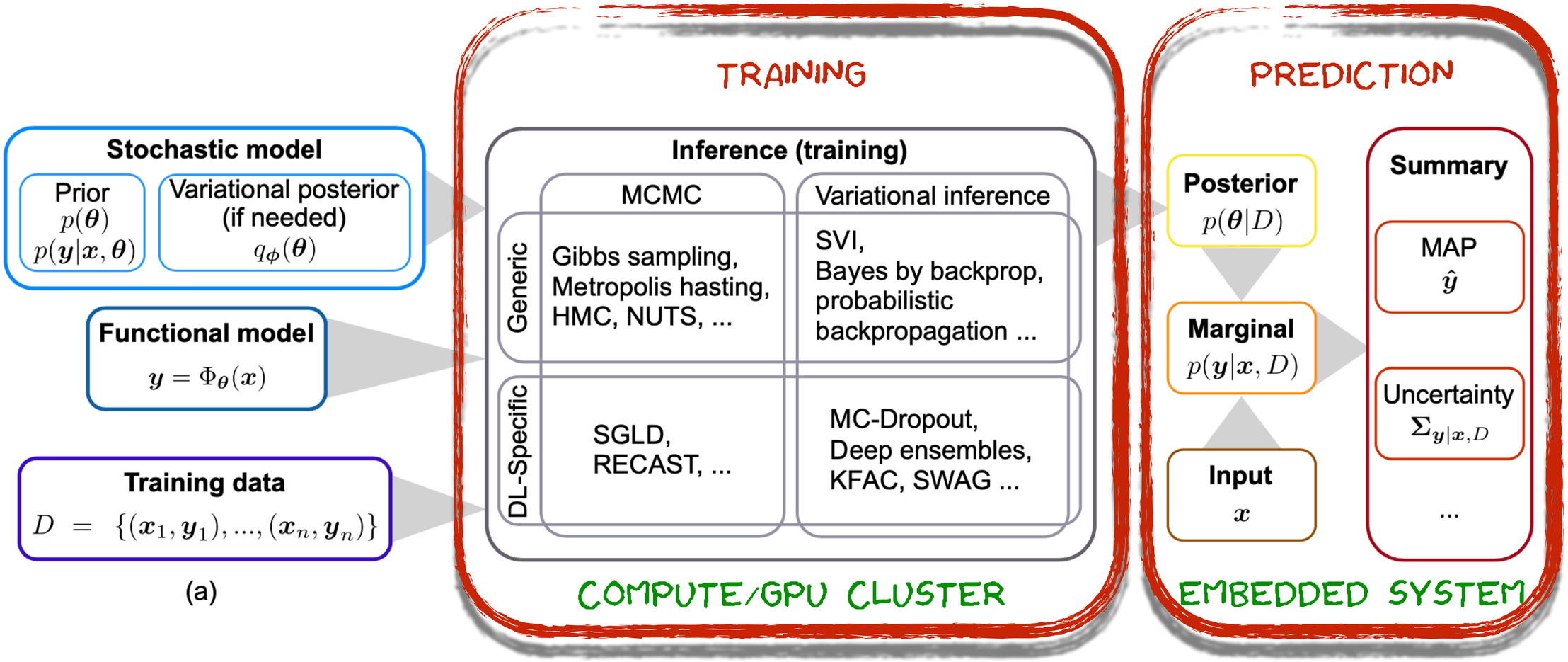
BNN WORKFLOW



Workflow to design (a), train (b) and use a BNN for predictions (c)

Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

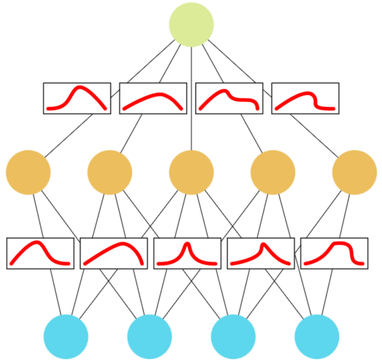
BNN WORKFLOW



Workflow to design (a), train (b) and use a BNN for predictions (c)

Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

BAYESIAN INFERENCE FOR BNNS



Finding the weight distributions

Markov Chain Monte Carlo (MCMC)

Approximate complex distributions of arbitrary shape by drawing many samples

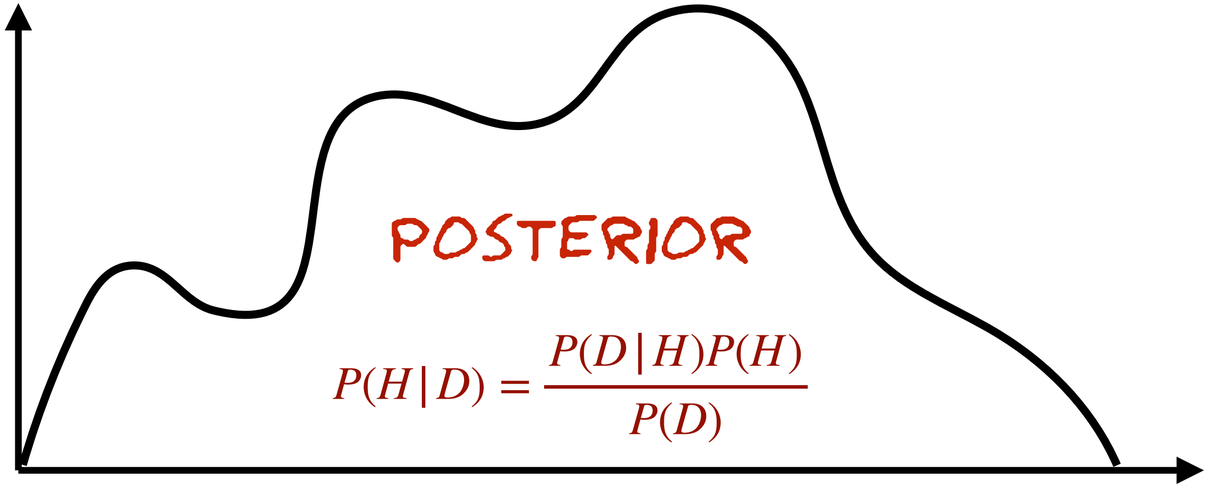
Variational Inference (VI)

Approximate all distributions with simple parametrized (Gaussian) distributions

NN-based approximations

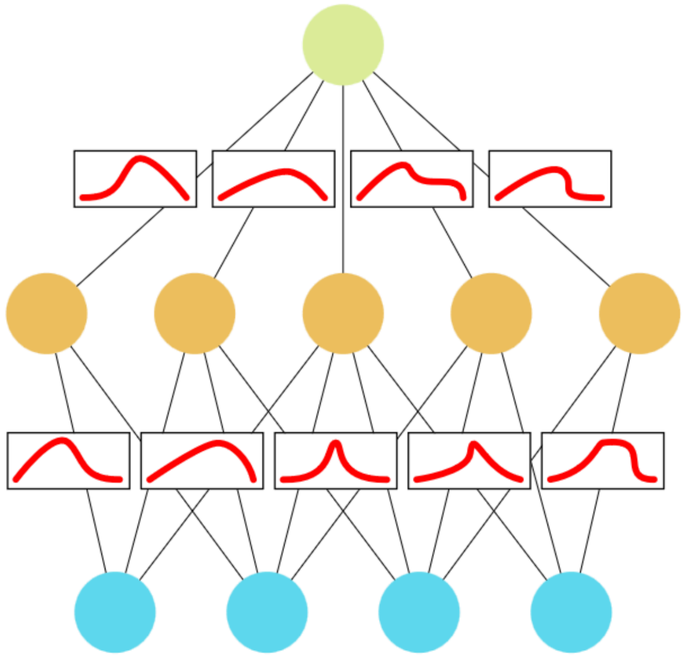
DeepEnsembles (DE)

Monte Carlo Dropout (MCDO)



BAYESIAN NEURAL NETWORKS

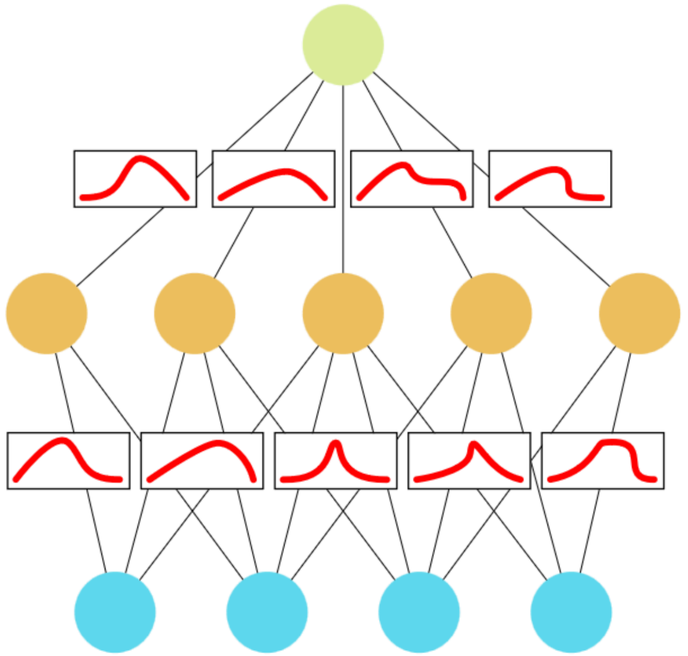
1. Predict weight distributions with **Bayesian Inference**



Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

BAYESIAN NEURAL NETWORKS

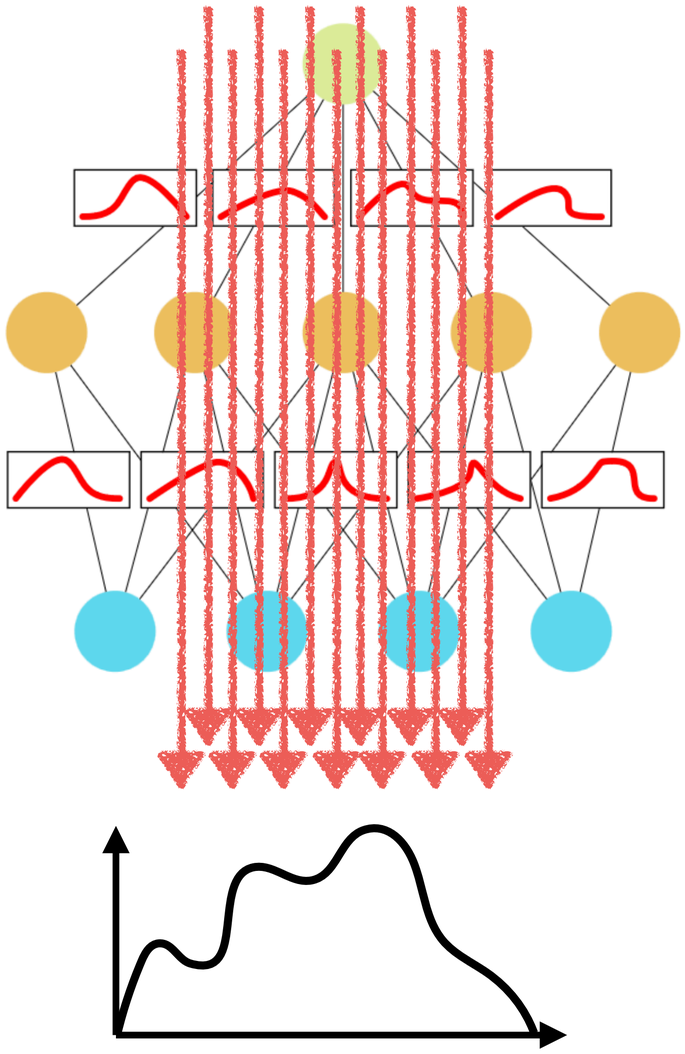
1. Predict weight distributions with **Bayesian Inference**
2. **Use the BNN** for predictions
   - Use multiple forward passes
   - Each time sample the weights from the weight distributions
   - Analyse the distribution of predictions



Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

BAYESIAN NEURAL NETWORKS

1. Predict weight distributions with **Bayesian Inference**
2. **Use the BNN** for predictions
   - Use multiple forward passes
   - Each time sample the weights from the weight distributions
   - Analyse the distribution of predictions



Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." IEEE Computational Intelligence Magazine 17.2 (2022)

VARIATIONAL INFERENCE WITH RADICAL APPROXIMATIONS

Deployment on Resource-constrained Embedded Devices

PROBABILISTIC FORWARD PASS

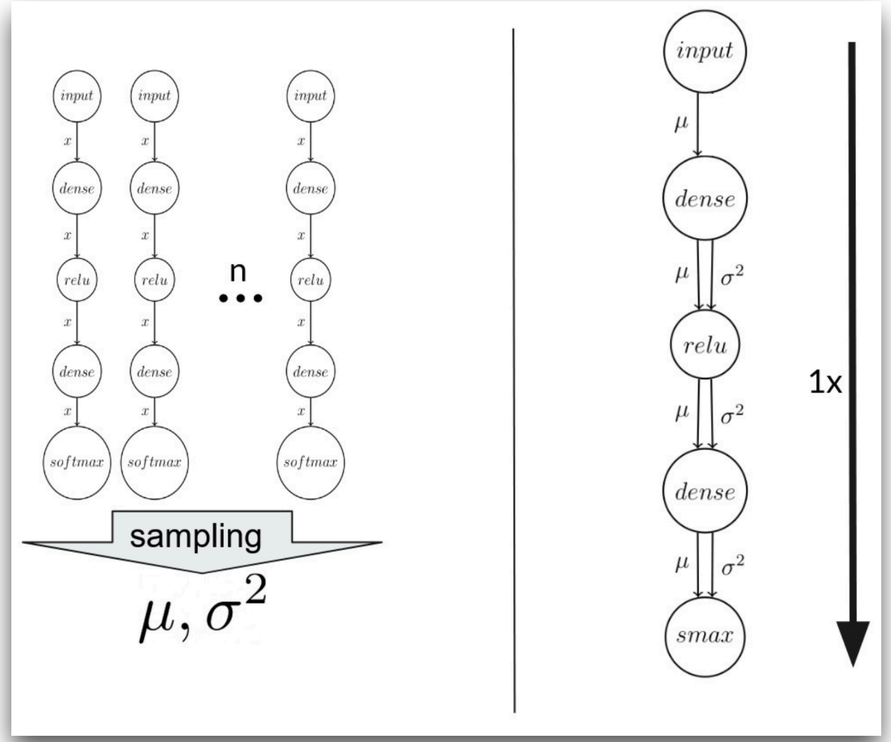
The PFP forces the output activations to be Gaussian distributed.

Reducing the multiple forward passes of VI sampling to **a single, but more complex, forward pass**.

$$E[x] = \mu_x = \frac{\mu_a}{2}\left(1 + erf\left(\frac{\mu_a}{\sqrt{2\sigma_a^2}}\right)\right) + \sqrt{\frac{\sigma_a}{2\pi}} exp\left(-\frac{{\mu_a}^2}{2\sigma_a}\right)$$

$$E[x^2] = \mu_{x^2} = \frac{\sigma_a^2 + {\mu_a}^2}{2}\left(1 + erf\left(\frac{\mu_a}{\sqrt{2\sigma_a^2}}\right)\right) + \mu_a\sqrt{\frac{\sigma_a^2}{2\pi}} exp\left(-\frac{{\mu_a}^2}{2\sigma_a}\right)$$

PFP ReLU activation function



Sampling-based inference VI (left) vs. Probabilistic Forward Path (Right) [2]

[1] W. Roth, Variational Inference in Neural Networks using an Approximate Closed-Form Objective, Workshop on BayesianDL, NIPS 2016.

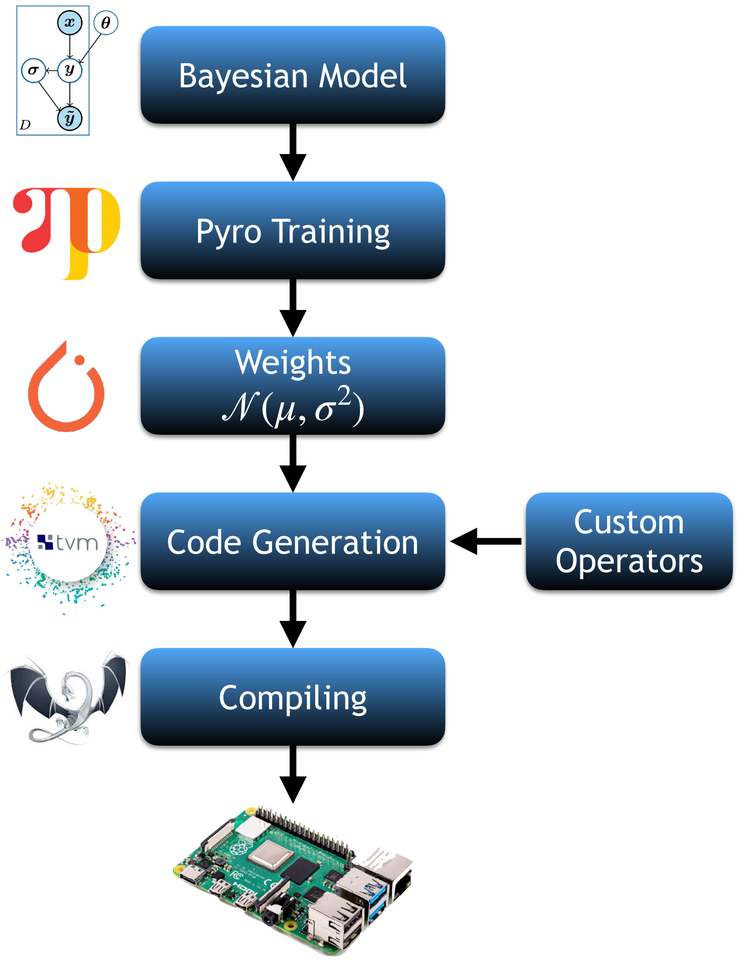
[2] F. Selker, Optimization of an Approximation Based Approach to Bayesian Neural Networks with TVM on Embedded Hardware, Master's thesis, Heidelberg University, 2023.

DEPLOYMENT ON EMBEDDED HARDWARE

Exotic operators[1] are not supported by standard libraries

ML Compilers like TVM[2] allow implementation and tuning of custom operators

All operators (convolutions, fully connected layers, activation functions, pool,…) need to be reimplemented and optimized



Workflow

[1] W. Roth, Variational Inference in Neural Networks using an Approximate Closed-Form Objective, Workshop on BayesianDL, NIPS 2016.

[2] https://github.com/apache/tvm

PERFORMANCE ON A RASPBERRY PI

Processor	Architecture	Deterministic NN		VI with Pyro	PFP	
		not tuned	tuned		not tuned	tuned
Cortex-A53	MLP	14.0 *ms*	0.933 *ms*	-	15.3 *ms*	4.99 *ms*
Cortex-A72	MLP	4.59 *ms*	0.186 *ms*	738 *ms*	3.75 *ms*	0.742 *ms*
Cortex-A76	MLP	1.64 *ms*	0.071 *ms*	309 *ms*	1.89 *ms*	0.341 *ms*
Cortex-A53	LeNet-5	21.1 *ms*	4.72 *ms*	-	76.1 *ms*	37.5 *ms*
Cortex-A72	LeNet-5	6.89 *ms*	0.754 *ms*	1200 *ms*	21.2 *ms*	10.1 *ms*
Cortex-A76	LeNet-5	3.16 *ms*	0.347 *ms*	797 *ms*	9.63 *ms*	4.02 *ms*

PERFORMANCE ON A RASPBERRY PI

Processor	Architecture	Deterministic NN		VI with Pyro	PFP	
		not tuned	tuned		not tuned	tuned
Cortex-A53	MLP	14.0 *ms*	0.933 *ms*	-	15.3 *ms*	4.99 *ms*
Cortex-A72	MLP	4.59 *ms*	0.186 *ms*	738 *ms*	3.75 *ms*	0.742 *ms*
Cortex-A76	MLP	1.64 *ms*	0.071 *ms*	309 *ms*	1.89 *ms*	0.341 *ms*
Cortex-A53	LeNet-5	21.1 *ms*	4.72 *ms*	-	76.1 *ms*	37.5 *ms*
Cortex-A72	LeNet-5	6.89 *ms*	0.754 *ms*	1200 *ms*	21.2 *ms*	10.1 *ms*
Cortex-A76	LeNet-5	3.16 *ms*	0.347 *ms*	797 *ms*	9.63 *ms*	4.02 *ms*

AUTO TUNING
10X

AUTO TUNING
5X

PERFORMANCE ON A RASPBERRY PI

Processor	Architecture	Deterministic NN		VI with Pyro	PFP	
		not tuned	tuned		not tuned	tuned
Cortex-A53	MLP	14.0 *ms*	0.933 *ms*	-	15.3 *ms*	4.99 *ms*
Cortex-A72	MLP	4.59 *ms*	0.186 *ms*	738 *ms*	3.75 *ms*	0.742 *ms*
Cortex-A76	MLP	1.64 *ms*	0.071 *ms*	309 *ms*	1.89 *ms*	0.341 *ms*
Cortex-A53	LeNet-5	21.1 *ms*	4.72 *ms*	-	76.1 *ms*	37.5 *ms*
Cortex-A72	LeNet-5	6.89 *ms*	0.754 *ms*	1200 *ms*	21.2 *ms*	10.1 *ms*
Cortex-A76	LeNet-5	3.16 *ms*	0.347 *ms*	797 *ms*	9.63 *ms*	4.02 *ms*

AUTO TUNING
10X

AUTO TUNING
5X

OVERHEAD PFP 5X/10X

PERFORMANCE ON A RASPBERRY PI

Processor	Architecture	Deterministic NN		VI with Pyro	PFP	
		not tuned	tuned		not tuned	tuned
Cortex-A53	MLP	14.0 *ms*	0.933 *ms*	-		4.99 *ms*
Cortex-A72	MLP	4.59 *ms*	0.186 *ms*	738 *ms*	3.75 *ms*	0.742 *ms*
Cortex-A76	MLP	1.64 *ms*	0.071 *ms*	309 *ms*		0.341 *ms*
Cortex-A53	LeNet-5	21.1 *ms*	4.72 *ms*	-	76.1 *ms*	37.5 *ms*
Cortex-A72	LeNet-5	6.89 *ms*	0.754 *ms*	1200 *ms*	21.2 *ms*	10.1 *ms*
Cortex-A76	LeNet-5	3.16 *ms*	0.347 *ms*	797 *ms*		4.02 *ms*

COST REDUCTION

900X

100X

AUTO TUNING
10X

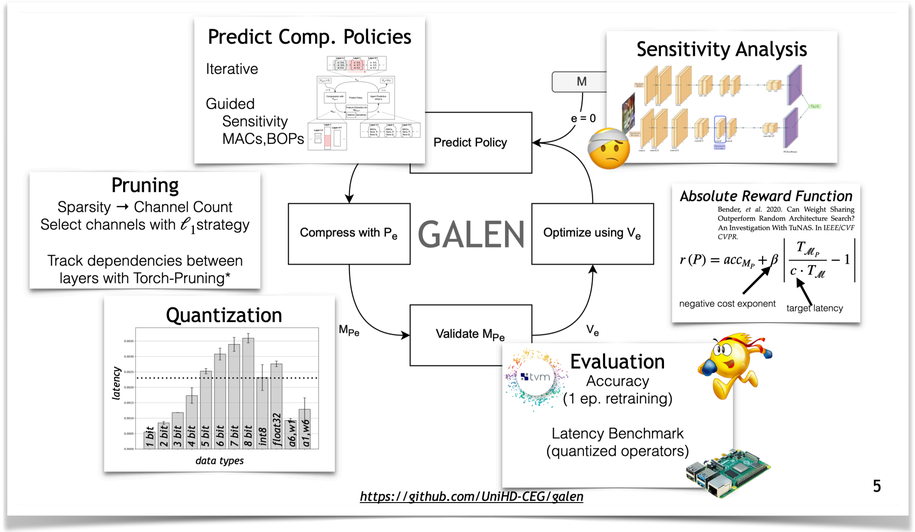
AUTO TUNING
5X

OVERHEAD PFP 5X/10X

SUMMARY

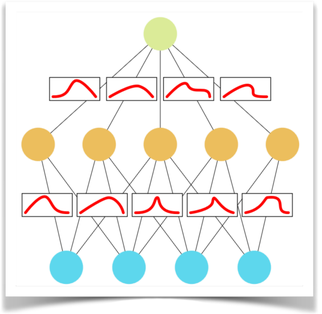
Automatic Compression (Galen)

- Guide search with
  - Sensitivity Information
  - Latency measurements
- Compression methods can balance each other.



Bayesian Neural Networks

- Report probabilities and can say “I don’t know”.
- Computational cost is significantly higher than their classical counterparts.
- Approximations can be used to reduce this cost drastically.



Deep Learning Compilers (TVM)

- Custom Operators
- Auto-tuning