Resource Efficient Artificial Intelligence

Dr Varun Ojha
School of Computing, Newcastle University
varun.ojha@newcastle.ac.uk
ojhavk.github.io

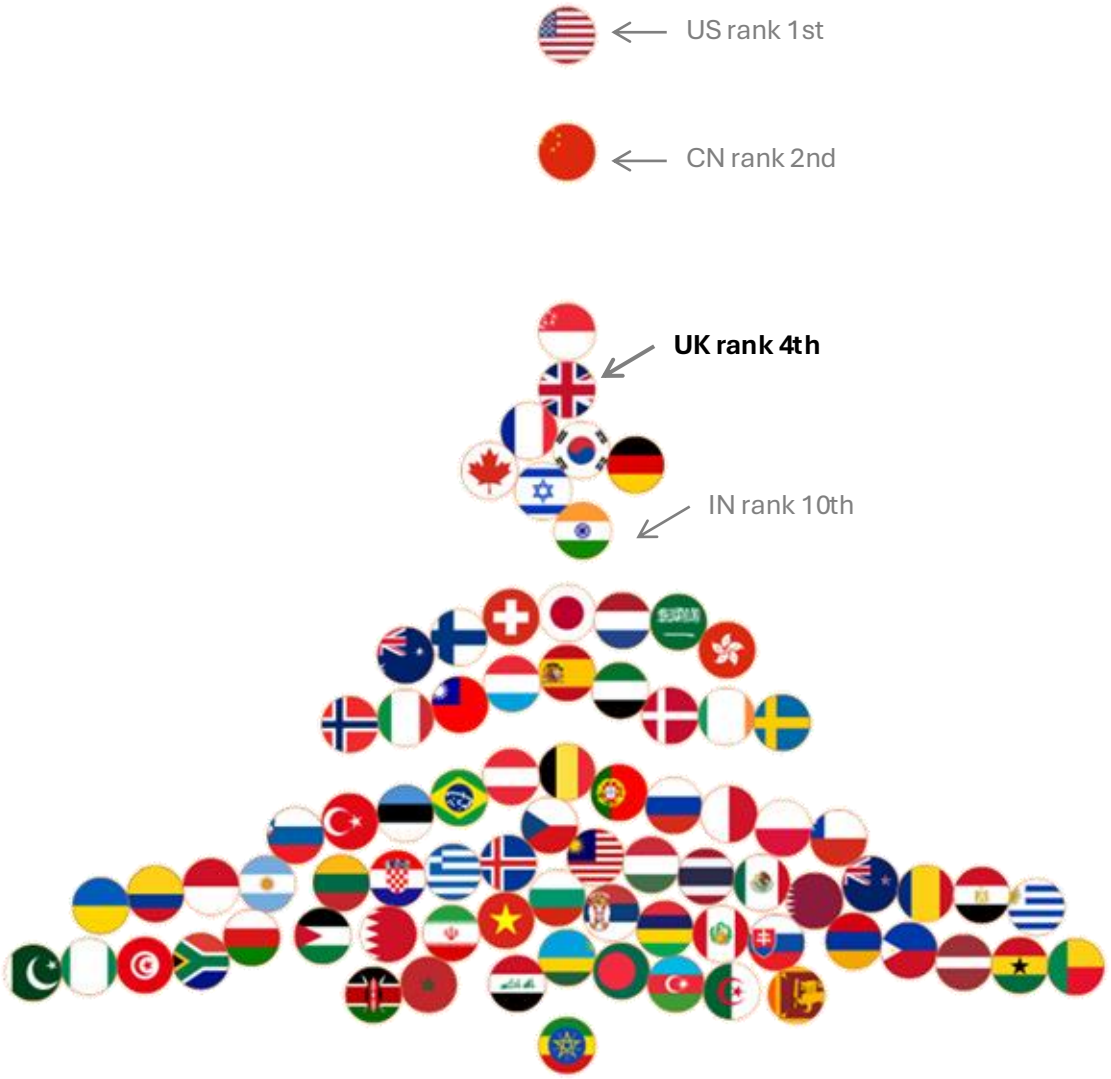
Resource Efficient Artificial Intelligence

Agenda

- Why Resource Efficient AI
- Understanding AI Model Performance
- AI Model Compression Methods and Results
- Search for Efficient AI Models
- Tricks to make AI models resource efficient

Part 1
Why Resource Efficient AI

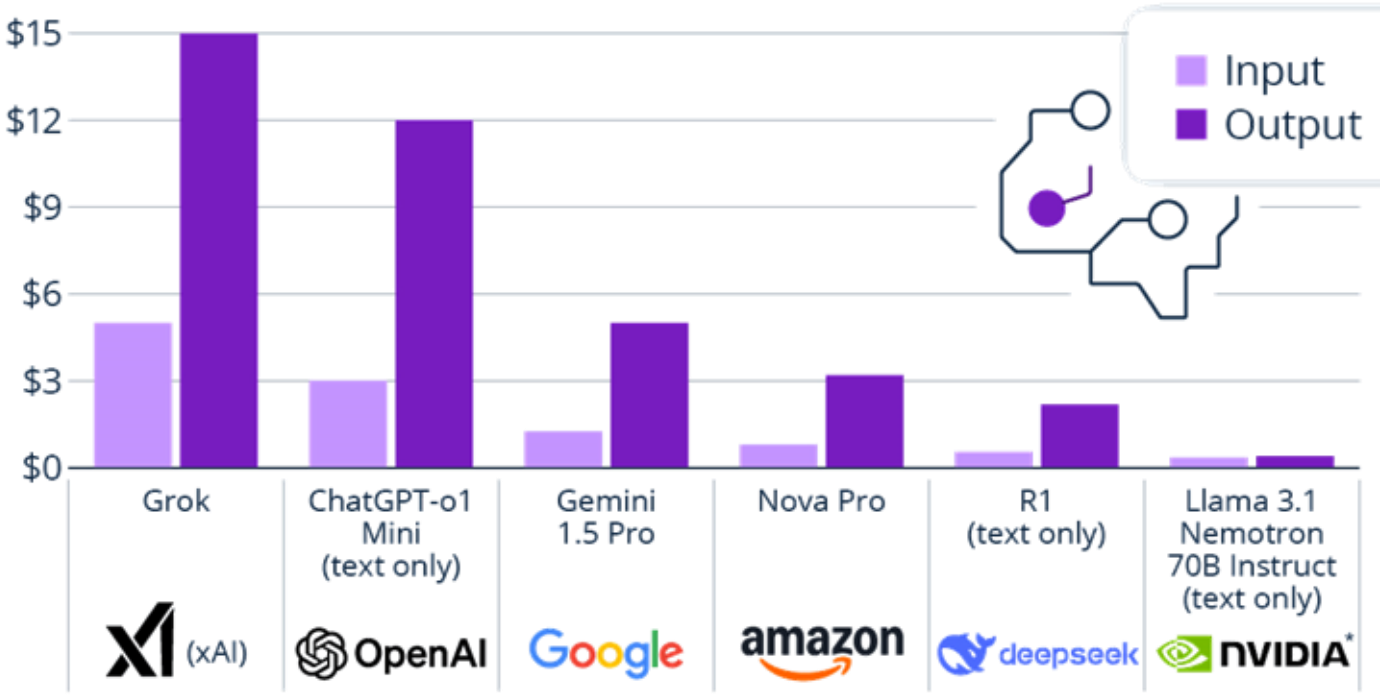
AI Arms Race



https://www.tortoisemedia.com/intelligence/global-ai

DeepSeek-R1 Upsets AI Market With Low Prices

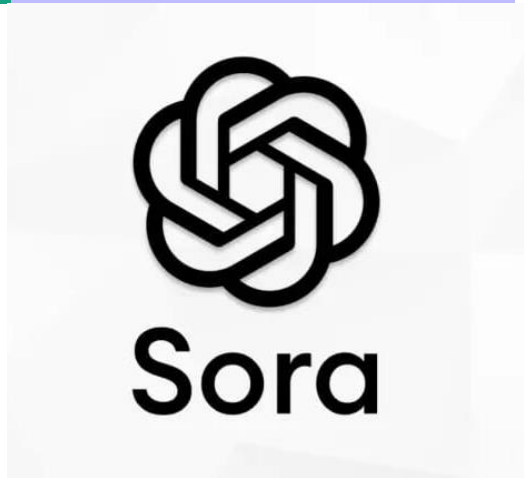
Estimated price for processing one million input/output tokens on different AI models



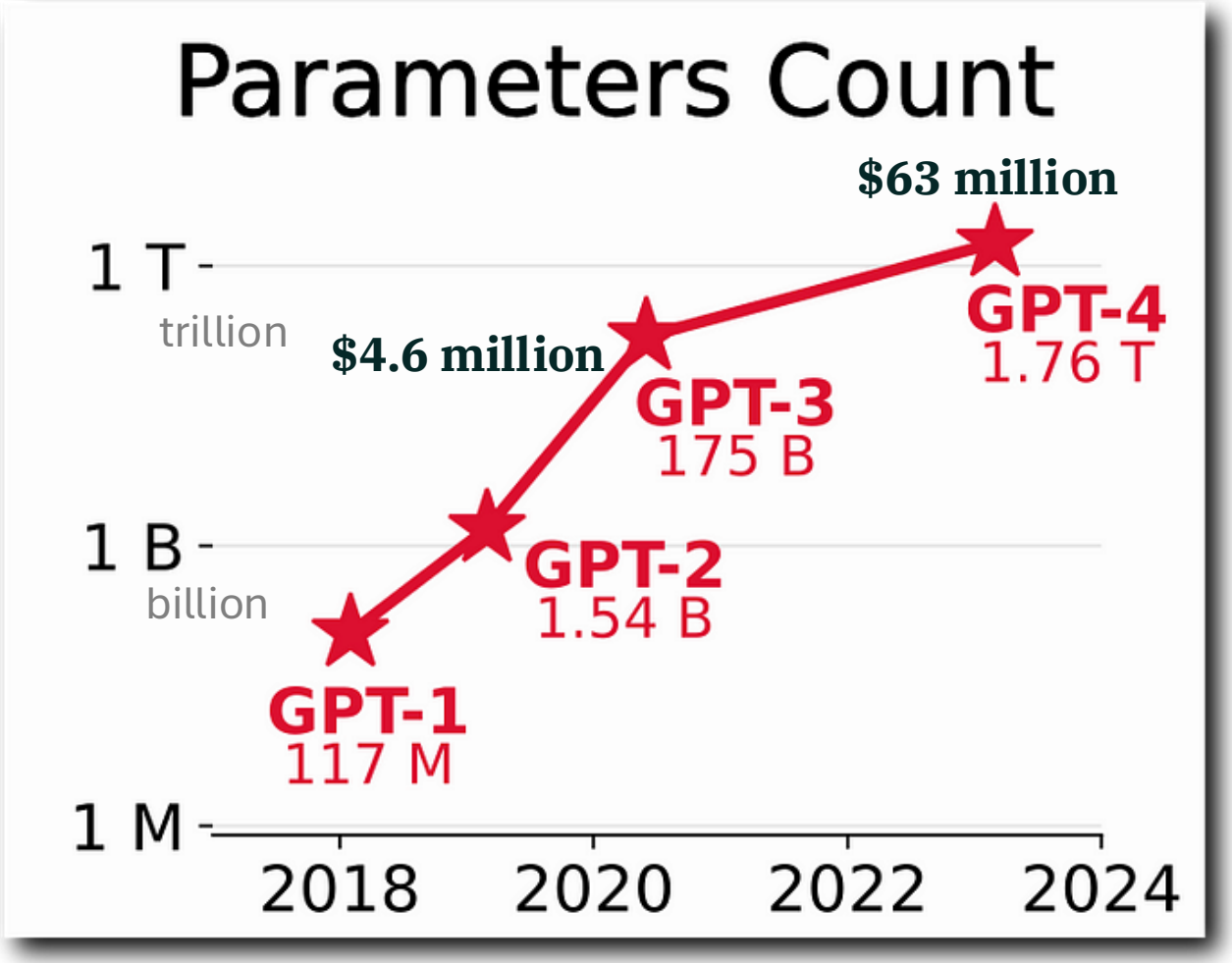
A token is the smallest unit of AI model processing (~4 characters).
o1 is ChatGPT's latest model. List includes most comparable model per company
* Uses Meta's open-source Llama AI

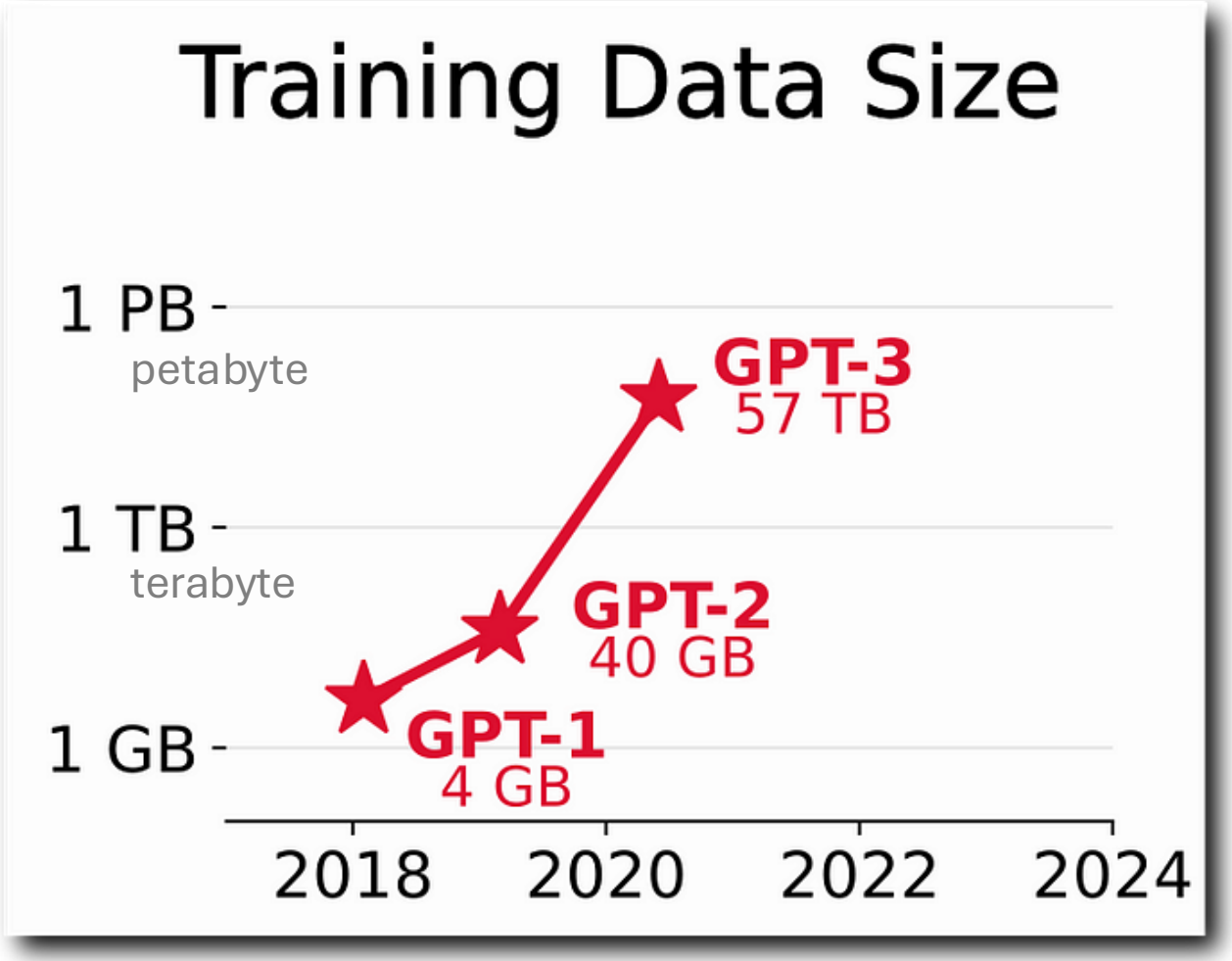
Source: DocsBot

Advanced AI Models



Massive AI Models



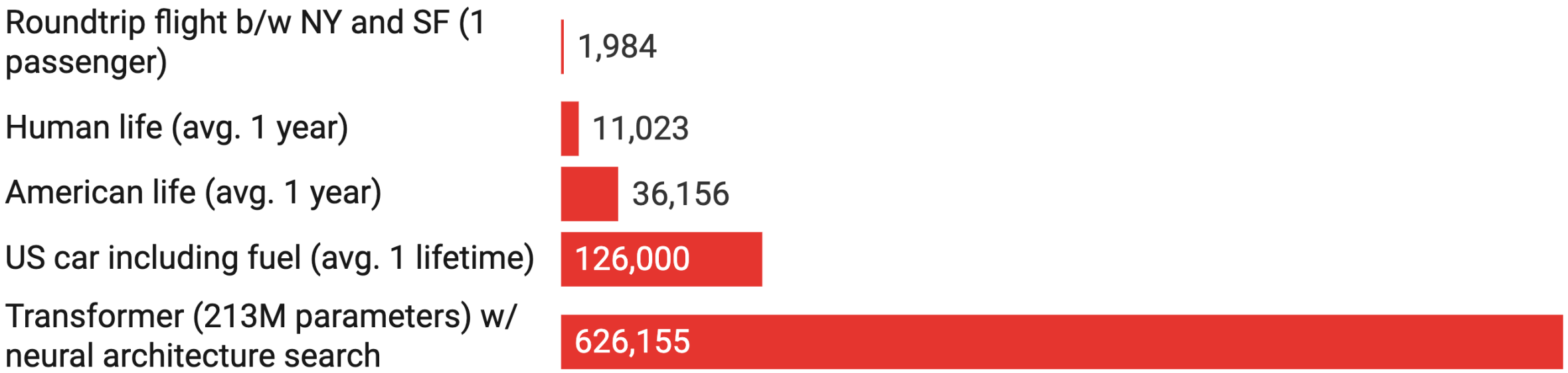


Source: Francesco Casalegno, ChatGPT Unveiled: What's the ML Model Inside it, from GPT-1 to GPT-4

AI Model Training Energy Consumption

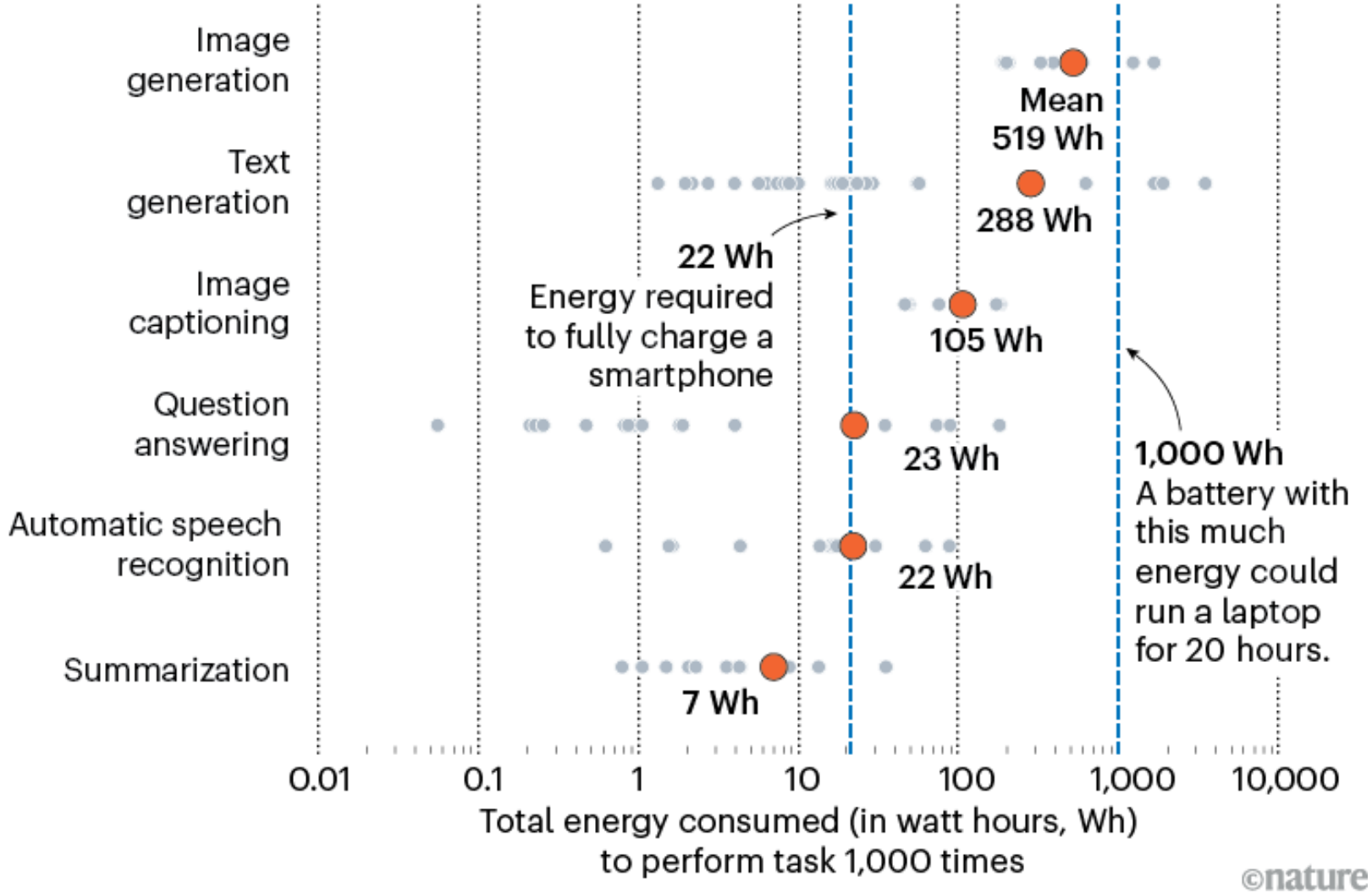
Common carbon footprint benchmarks

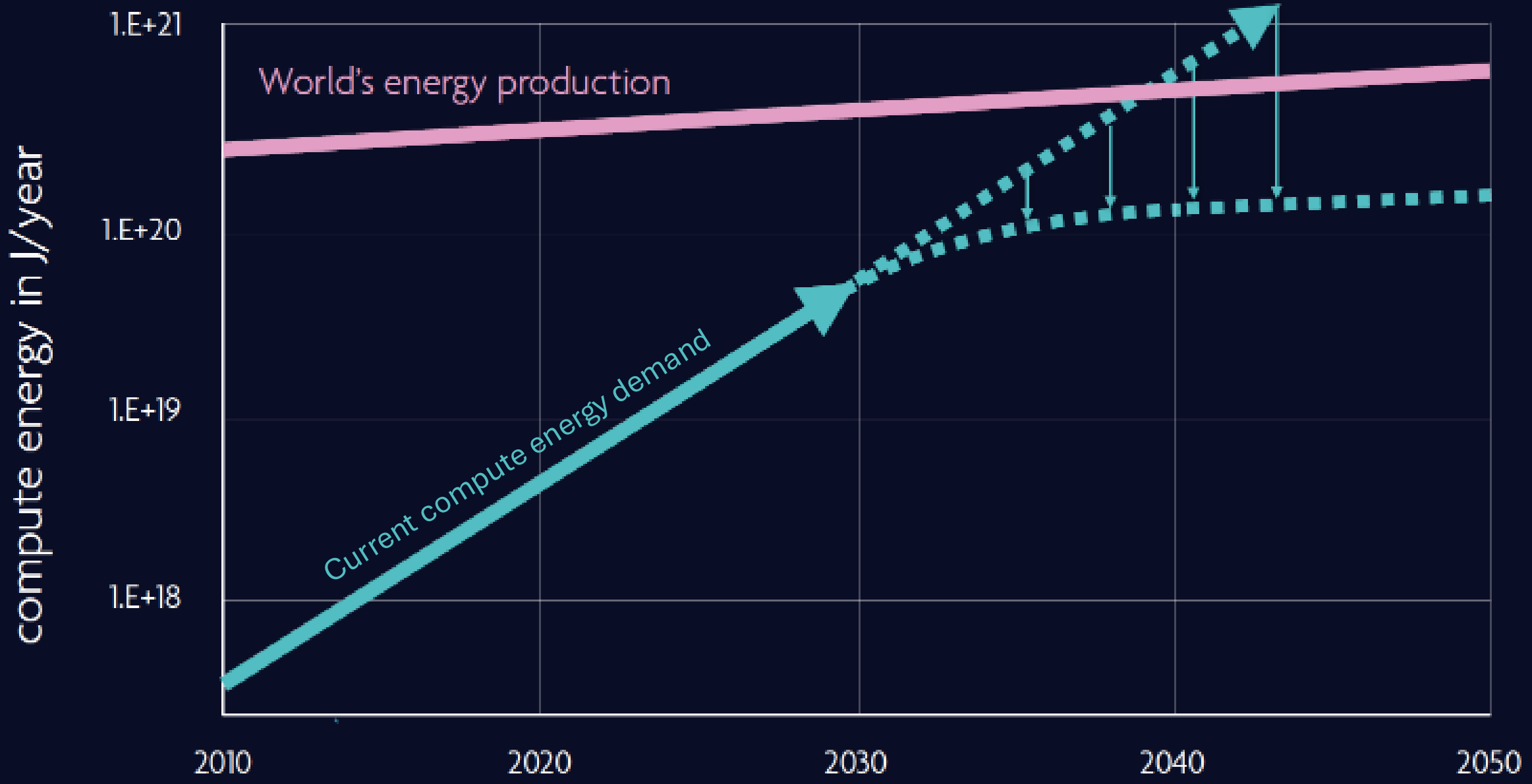
in lbs of CO_2 equivalent



MIT Technology Review

AI Model Inference Energy Consumption





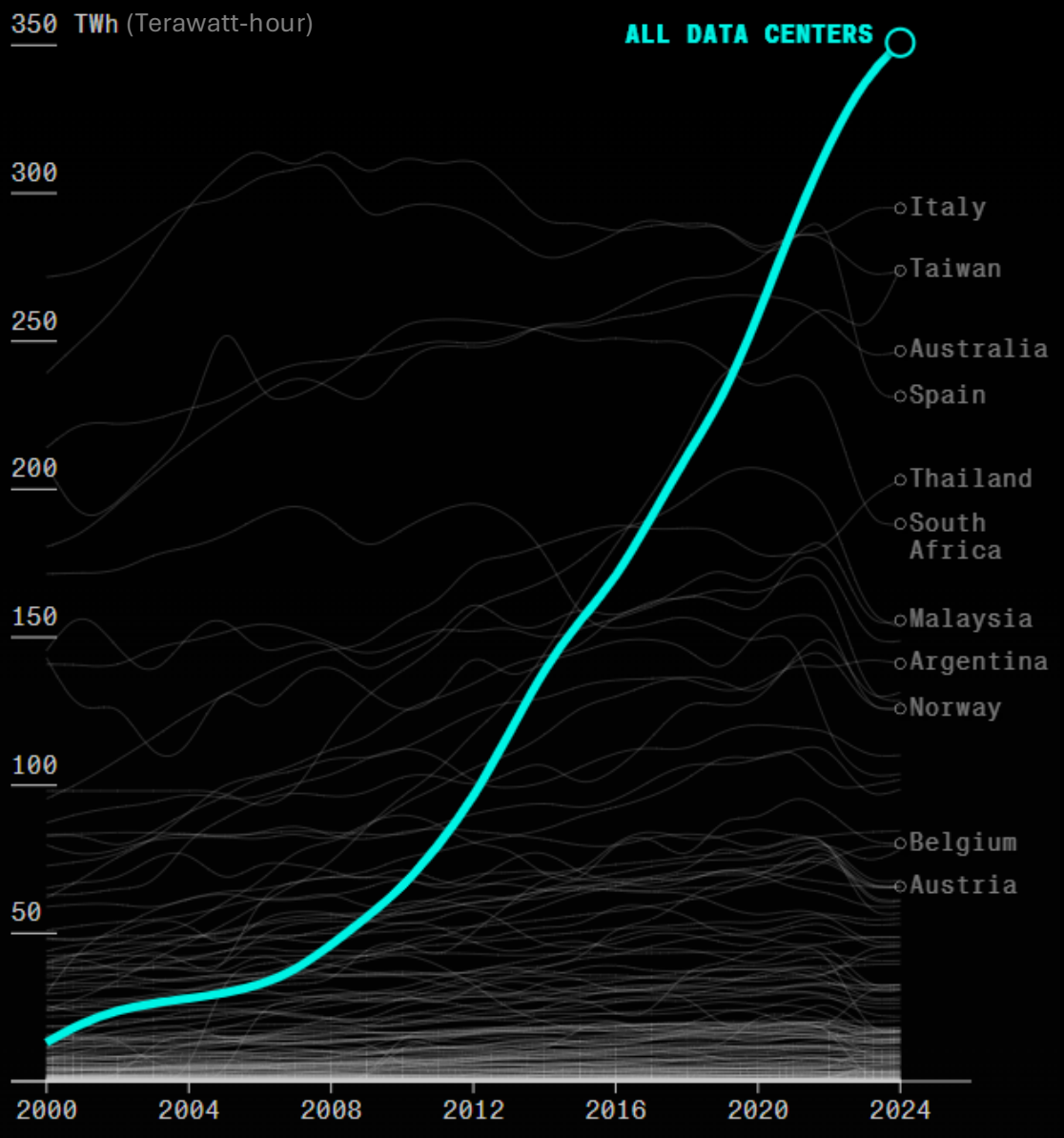
Source: SRC decadal plan 2020

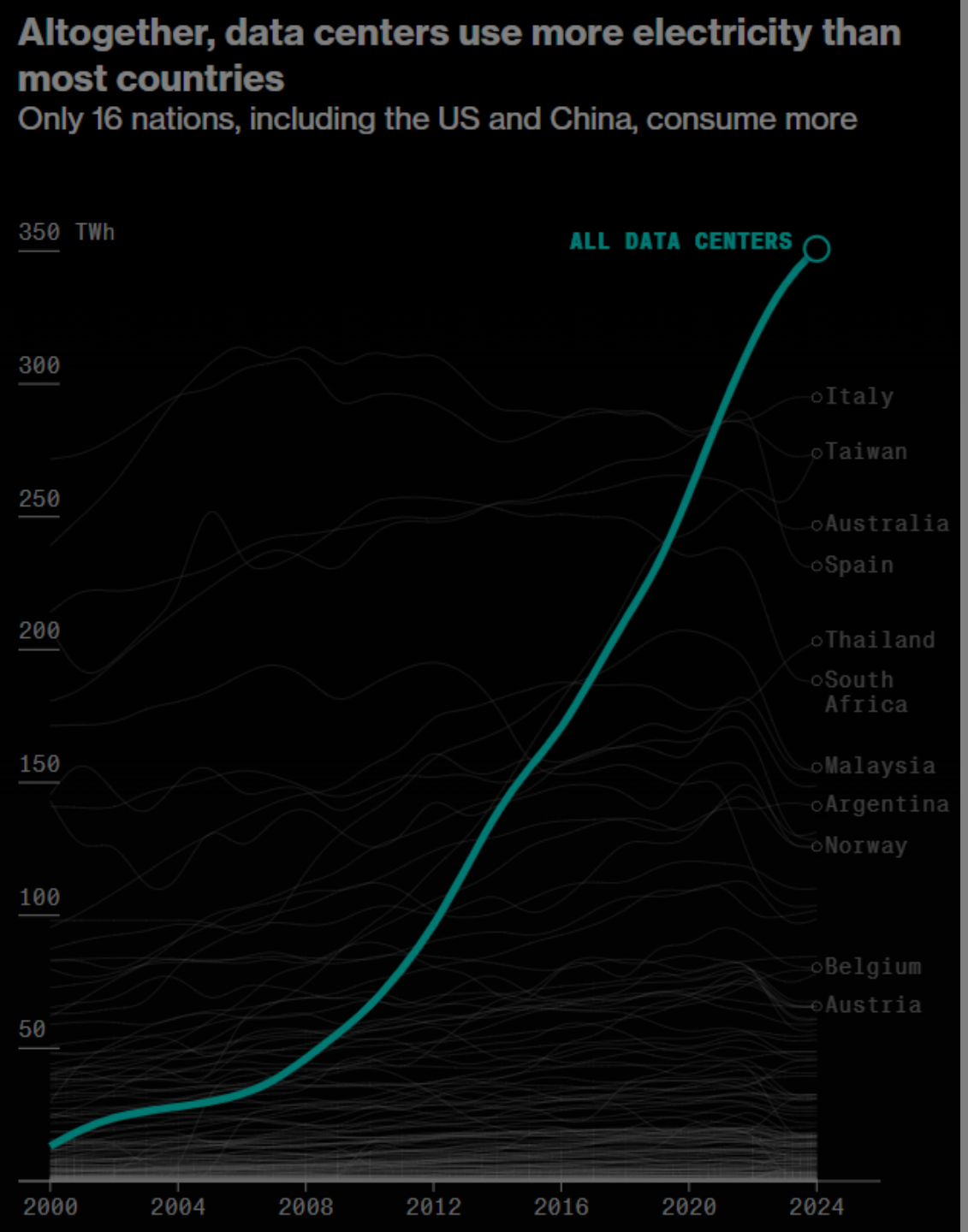
Data Centres

Altogether, data centers use more electricity than most countries

Only 16 nations, including the US and China, consume more

Source: Bloomberg





Potential solutions to energy problem: Cloud AI → Edge AI

- Moving AI applications from the cloud to the edge
- AI model simplification techniques to reduce power consumption
- AI Model quantization, etc.
- On device (Edge) AI model training
- Federated Learning

Edge AI

- Unlike Cloud AI (e.g., ChatGPT that runs in data centers), edge AI runs at the edge computing devices such smartphones, cameras, cars, medical devices, ensuring quality of data for inference
- Reduces latency, cost, and power consumption
- Protects data privacy and reduce improve data security and cybersecurity
- Reduces risk of inference failure in critical systems (e.g., autonomous vehicles, healthcare devices) that may endanger lives

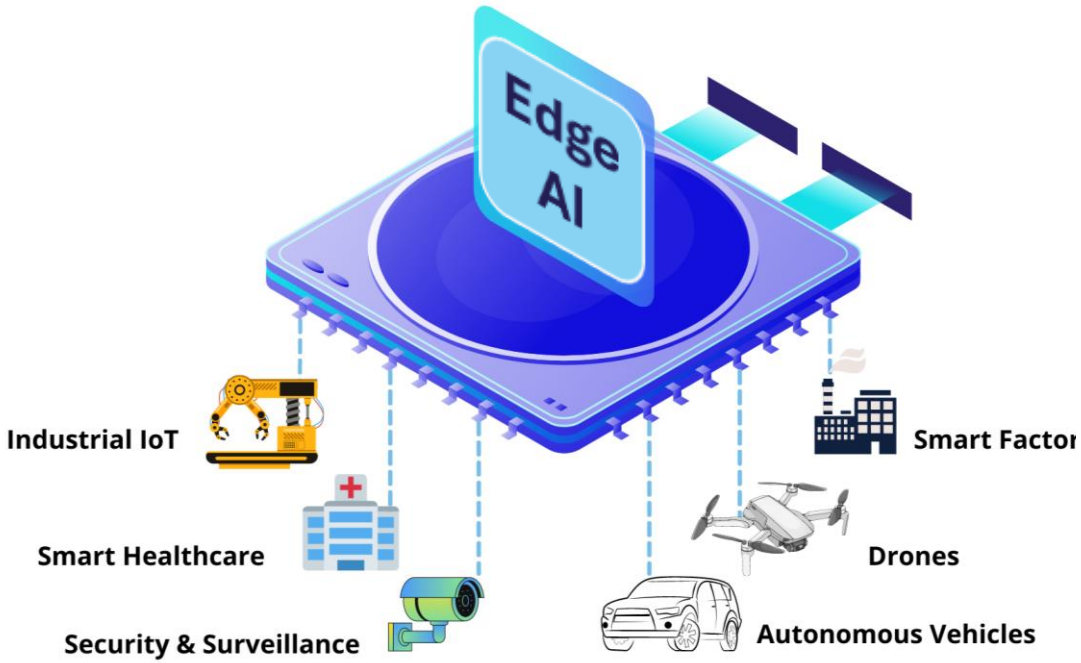
Everyone is a walking AI computer

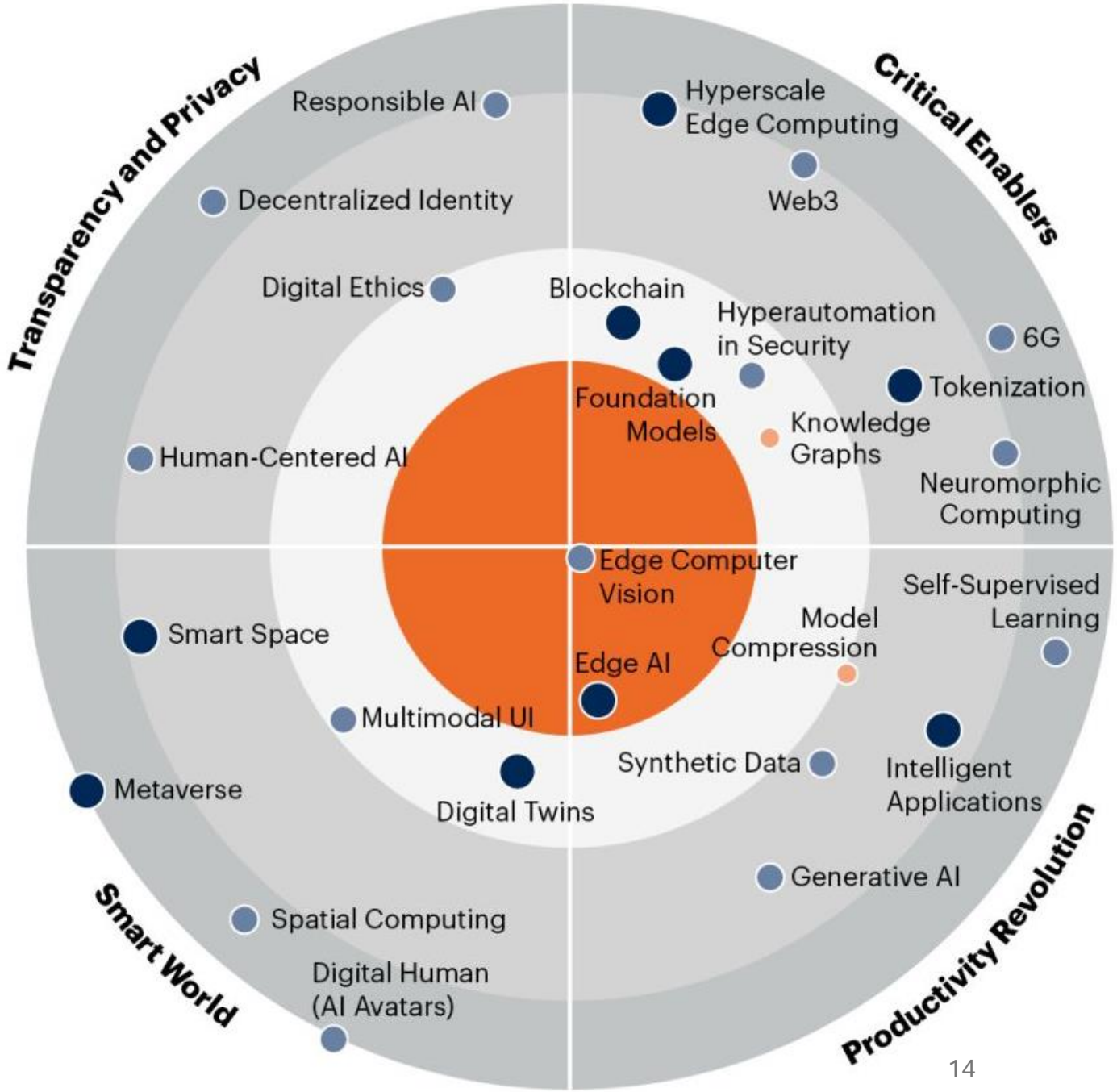
Any random mobile configuration/specification these days

- **Chip**
- A18 Bionic chip
- 6-core CPU with 2 performance
- 4 efficiency cores
- 5-core GPU
- 16-core Neural Engine
- **Capacity**
- 512GB
- **Multiple Sensors**
- 48MP camera
- Satellite and GPS

The Emergence of Edge AI: a game changer for industries

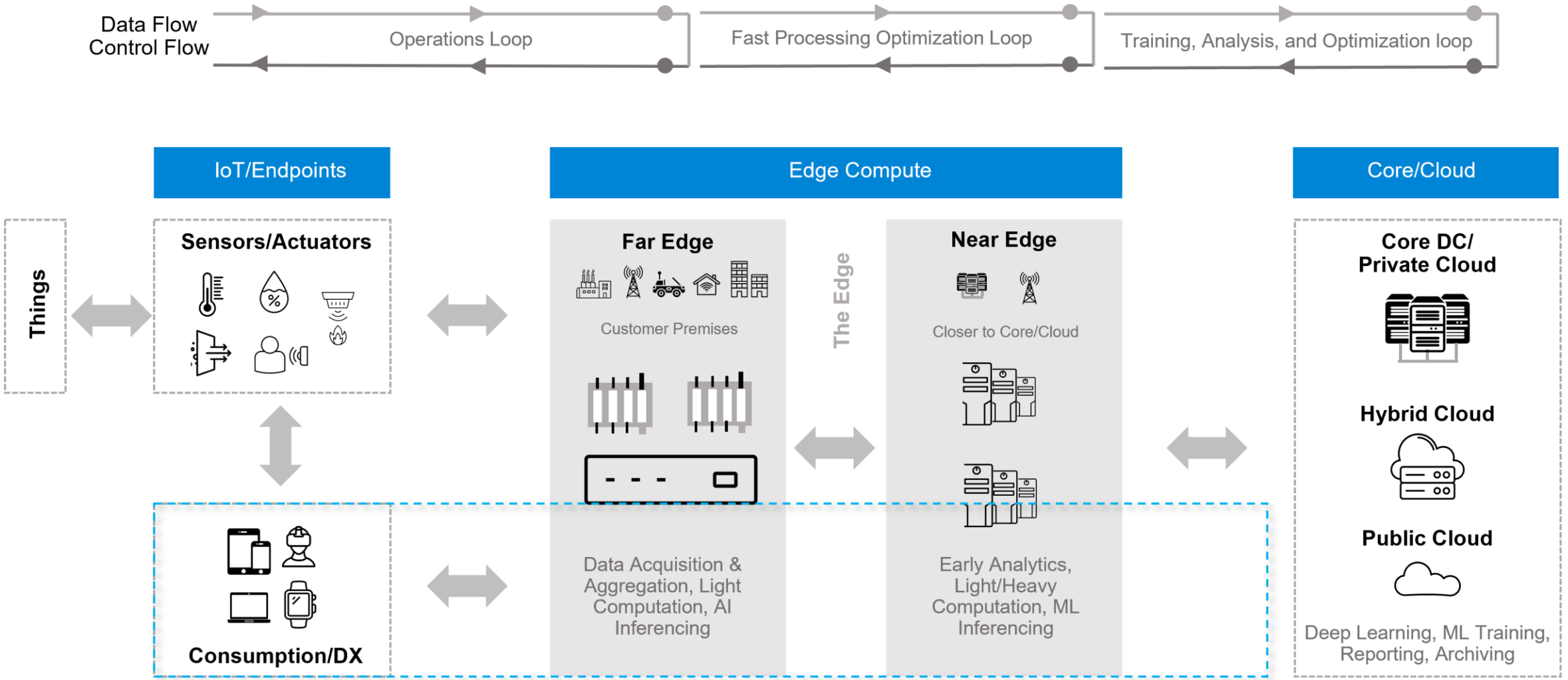
(Gartner 2023)





The Rise of Generative AI at the Edge

image source: Arrow Intelligent Solutions



Small Language Models Could Redefine The AI Race,

Forbes

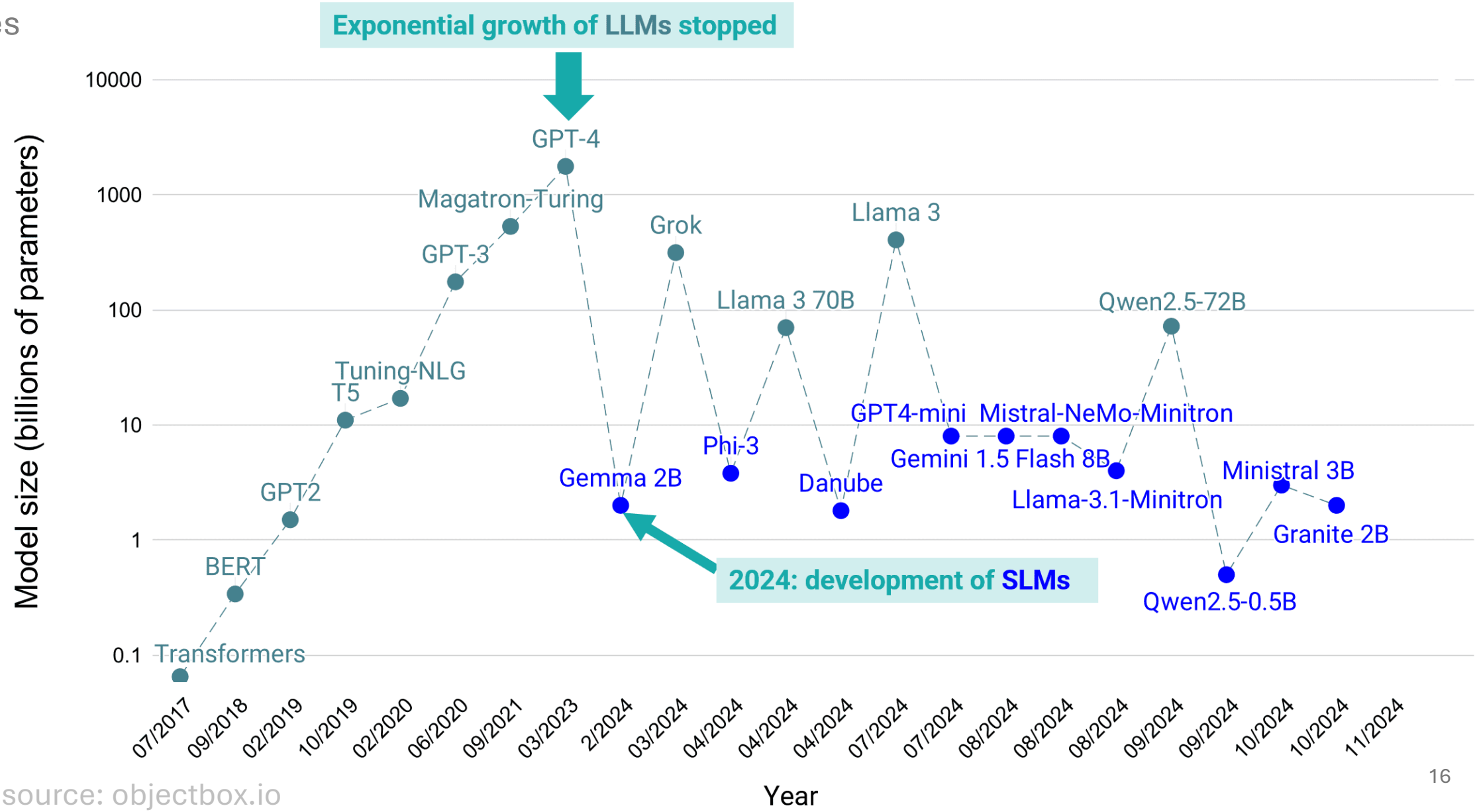


Image source: objectbox.io

AI Model Size

Performance Trade-offs

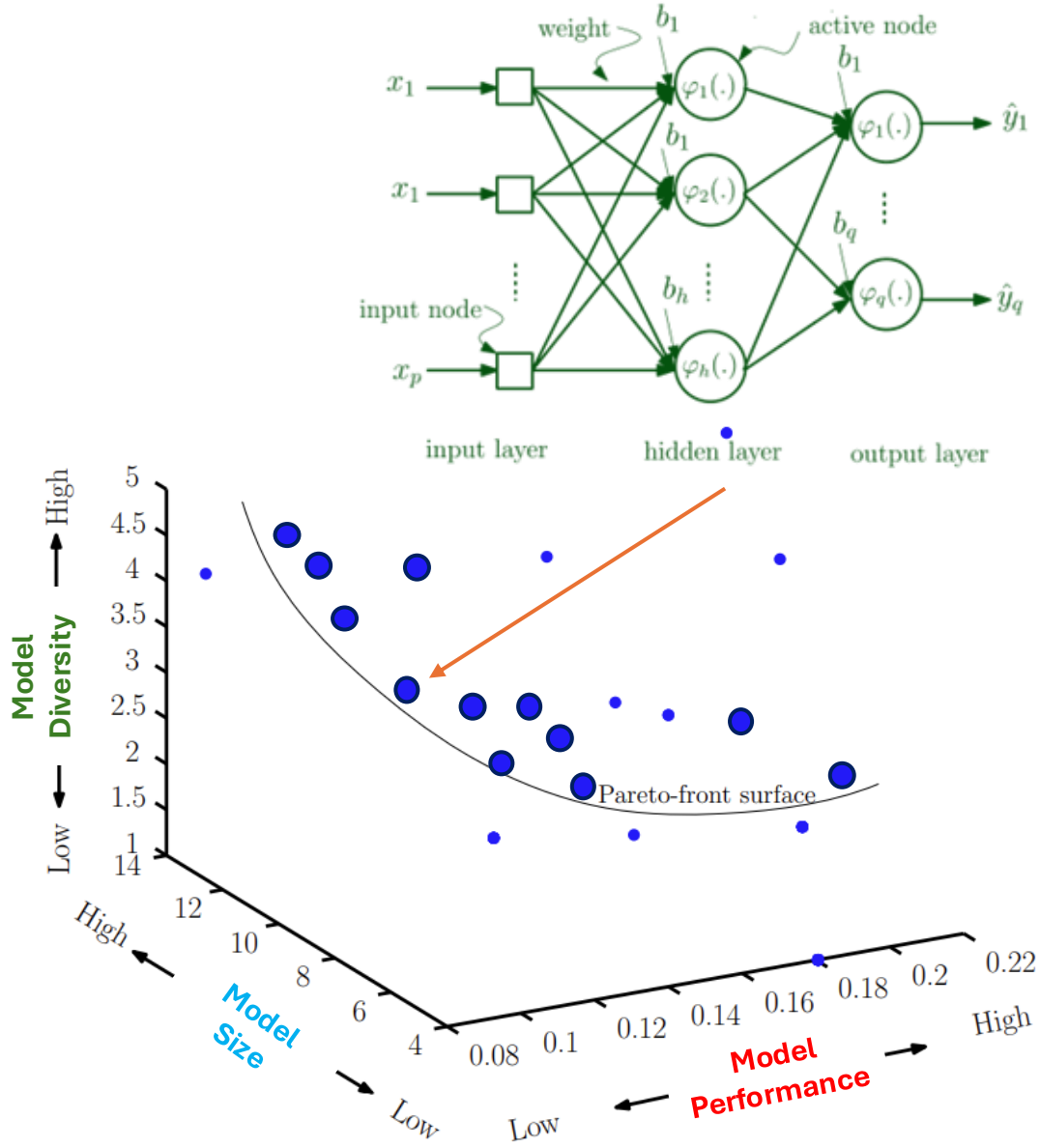
Larger models, such as GPT-3 with 175 billion parameters, can handle complex tasks but require significant computational resources and incur higher operational costs.

Efficiency of Smaller Models

Smaller models offer faster performance and lower costs, making them suitable for real-time applications, though they may compromise on accuracy.

Architectural Impact

The structure of a model significantly affects how effectively its parameters are utilized, with optimized architectures enhancing performance metrics.

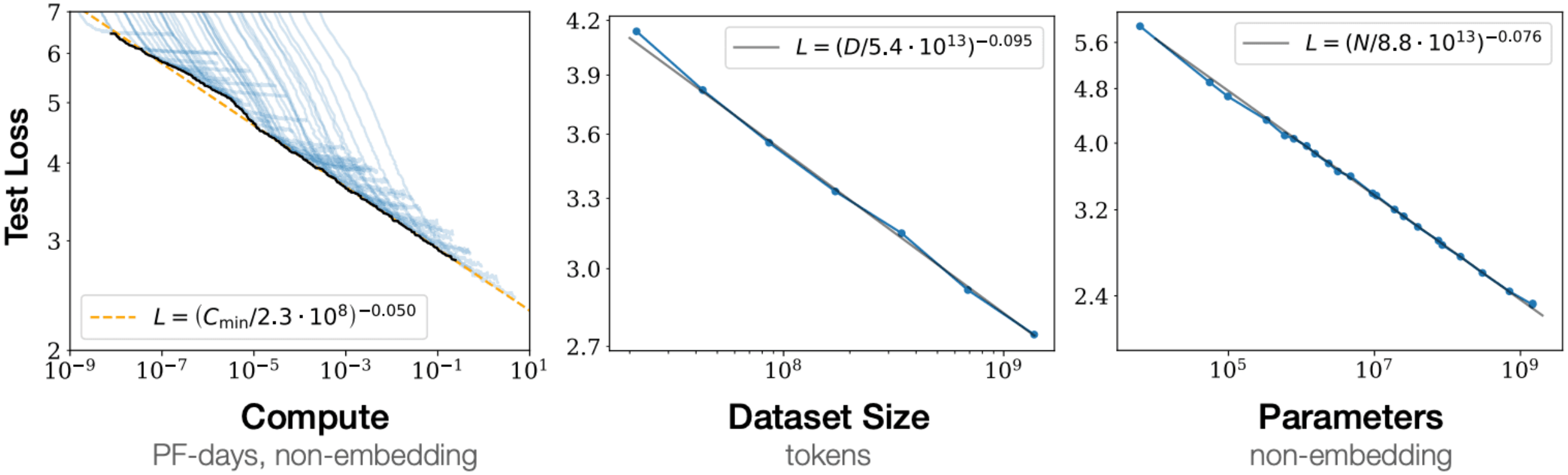


Ojha et al, *Applied Soft Compt.*

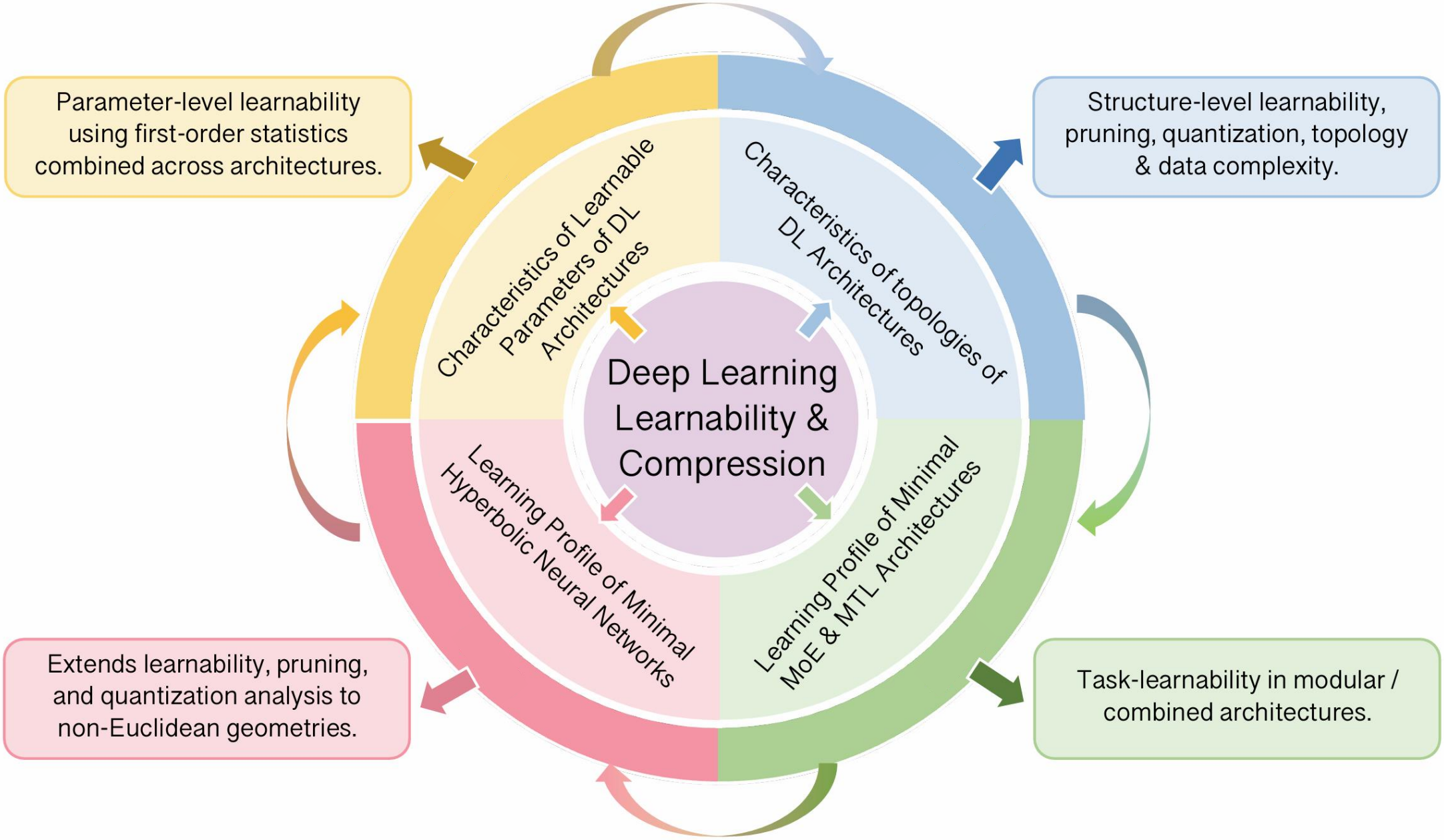
Part 2
Understanding AI Model Performance

AI Model Performance

AI model's performance predictably improves as you increase resources like model size (parameters), dataset size, and compute power, often following power-law relationships where gains diminish but remain consistent.

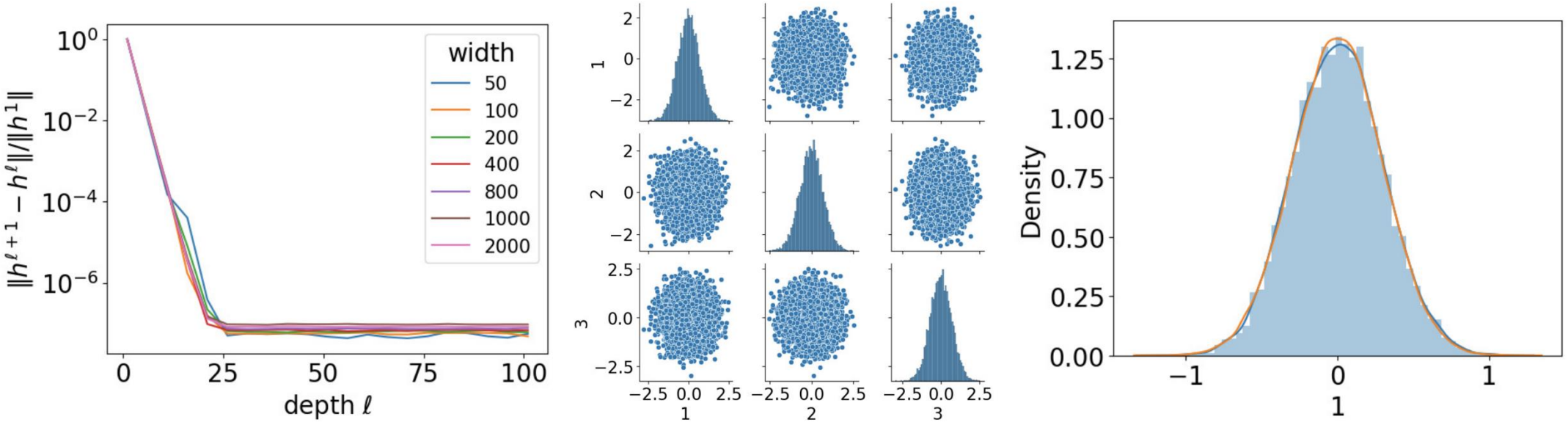


Kaplan et al. (2020)


Parameter-level learnability using first-order statistics combined across architectures.
Structure-level learnability, pruning, quantization, topology & data complexity.
Characteristics of Learnable Parameters of DL Architectures
Characteristics of topologies of DL Architectures
Deep Learning Learnability & Compression
Learning Profile of Minimal Hyperbolic Neural Networks
Learning Profile of Minimal MoE & MTL Architectures
Extends learnability, pruning, and quantization analysis to non-Euclidean geometries.
Task-learnability in modular / combined architectures.

Trained Network Exabits Gaussian distribution

Convergence to the fixed point (left); joint distributions for the first neuron for three outputs for 10, 000 neuron networks, with orange curve denotes the Gaussian distribution (right)

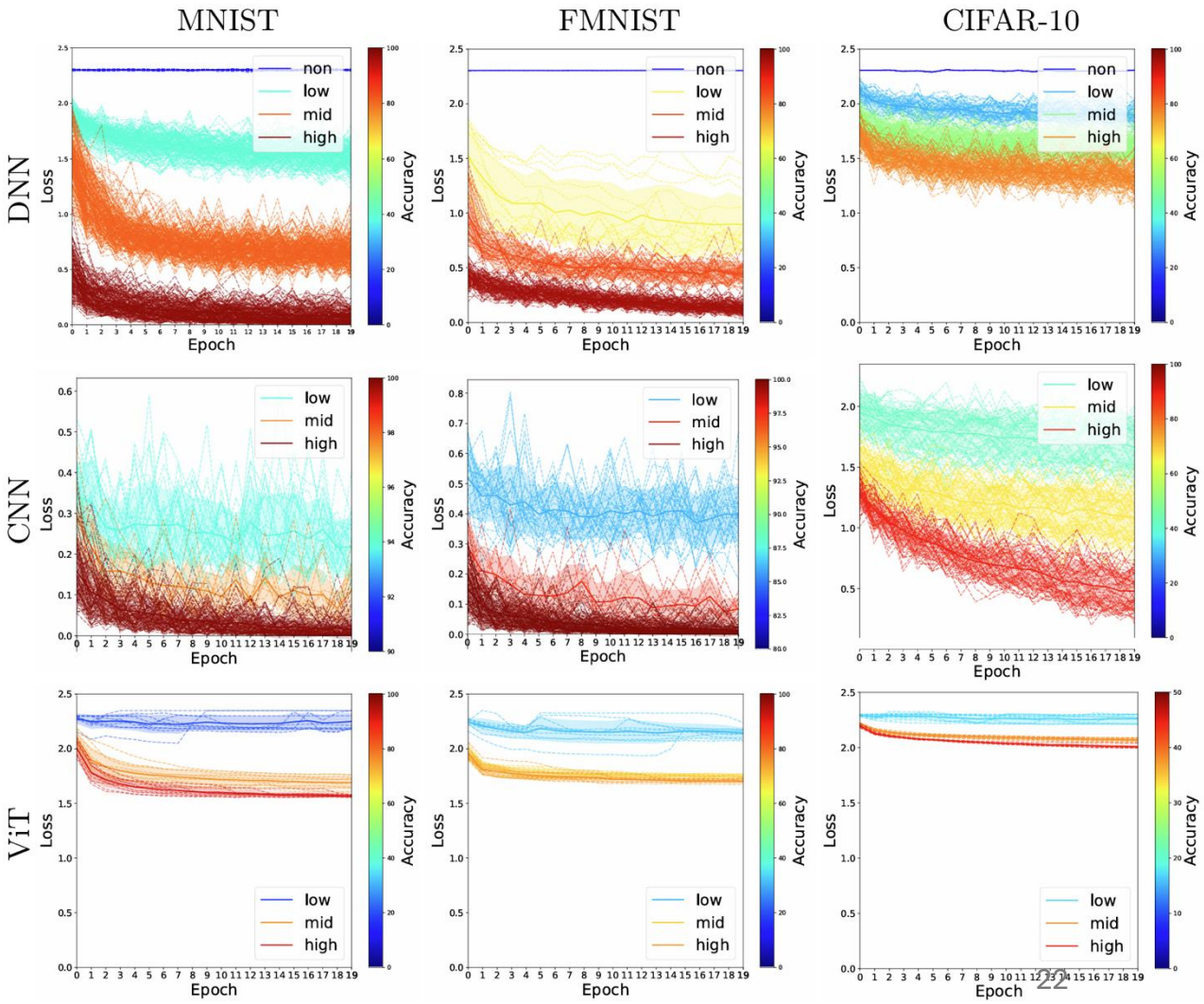


Gao et al· 2023, NeuriIPS

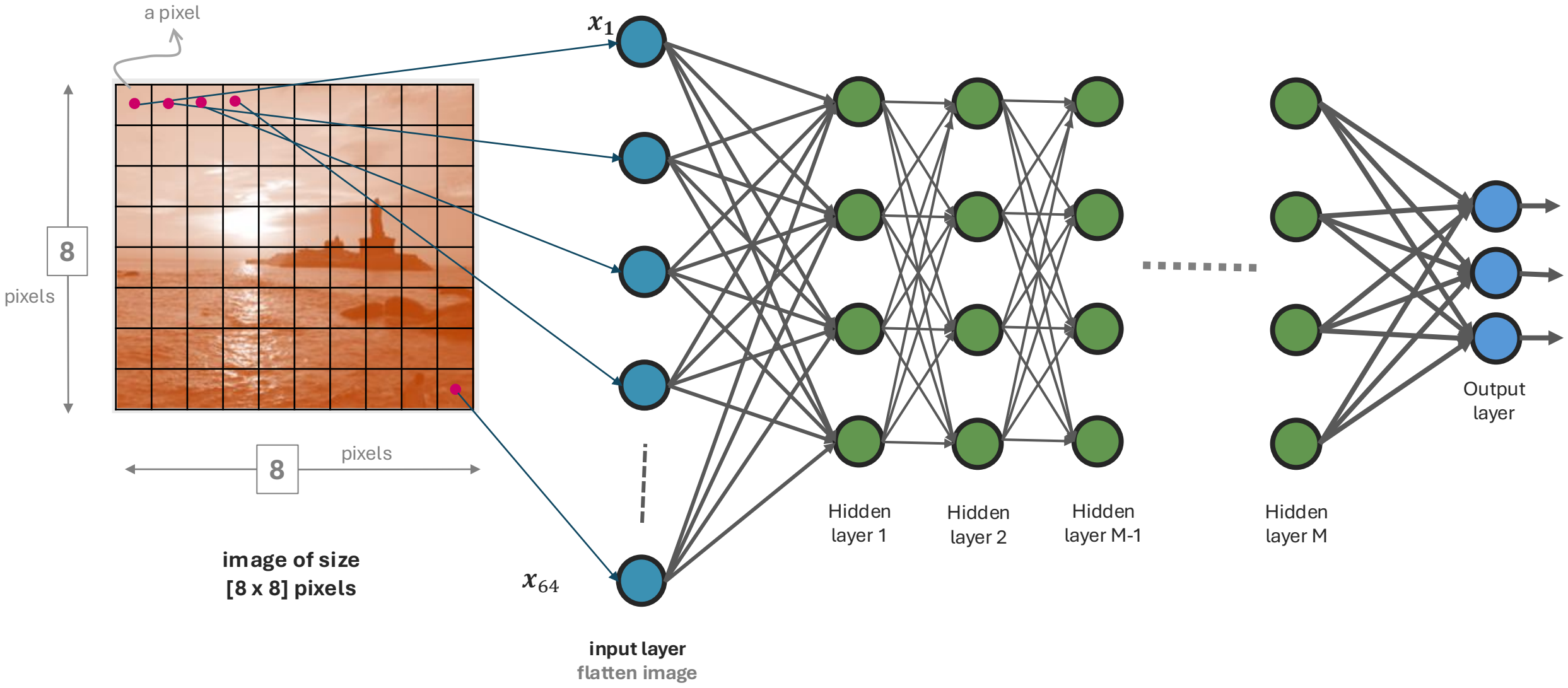
Performance Characterization of Models

Irrespective of model parameter size the model's convergence cluster into optimal and suboptimal patterns

Network	Data	Layer	Input	Output
DNN	MNIST/ F-MNIST	FC1	28×28	5-200
		FC2	5-200	5-200
		FC3	5-200	10
	CIFAR-10	FC1	3×32×32	5-1000
		FC2	5-1000	5-1000
		FC3	5-1000	10
CNN	MNIST F-MNIST	Conv1	28×28×1	26×26×C
		FC	26×26×C	10
	CIFAR-10	Conv1	3×32×32	30×30×C
		FC	30×30×C	10
ViT	MNIST F-MNIST CIFAR-10	Encoder	d_model=784 nhead=2-16	-
		FC	784 (input)	10
		-	-	-

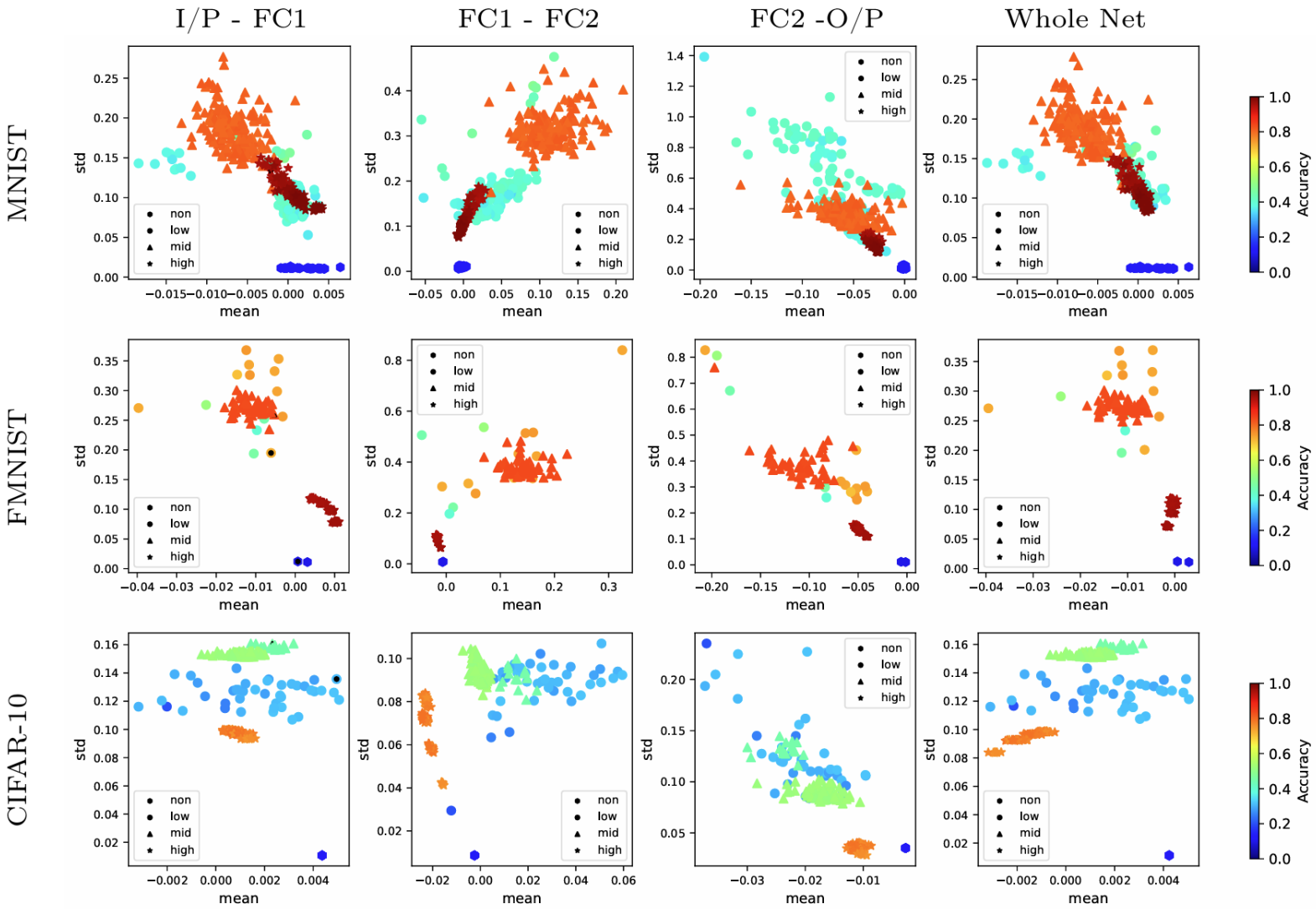


Deep Neural Networks



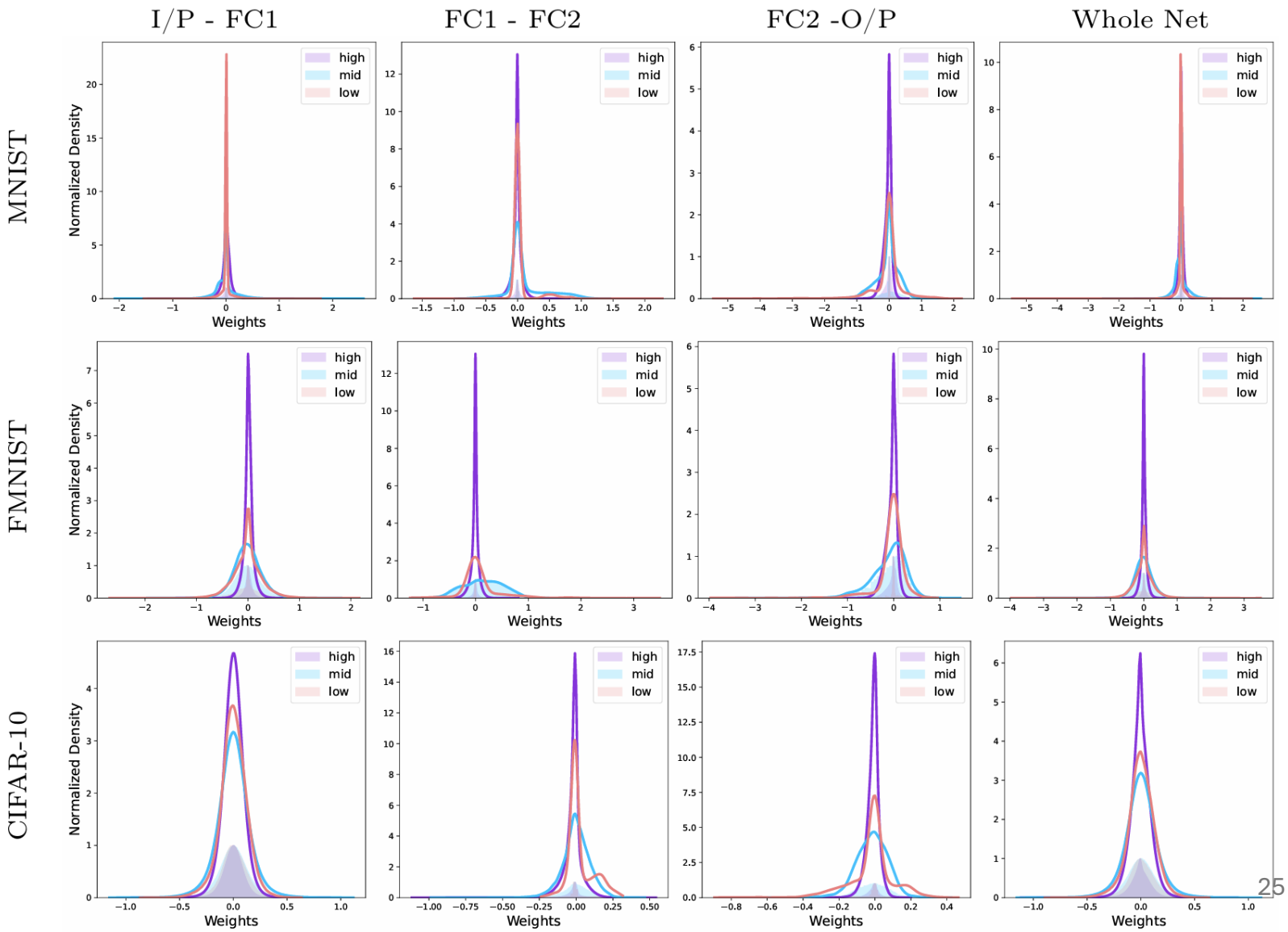
DNN's Learnable Parameters

Learned parameter of all **failed models**, **suboptimal models** and **successes full models** clearly show similarity of their statistical properties and weight distribution irrespective of their model size. i.e. models of any size converge to a similar weight distribution when trained to a fixed number of epochs on a dataset

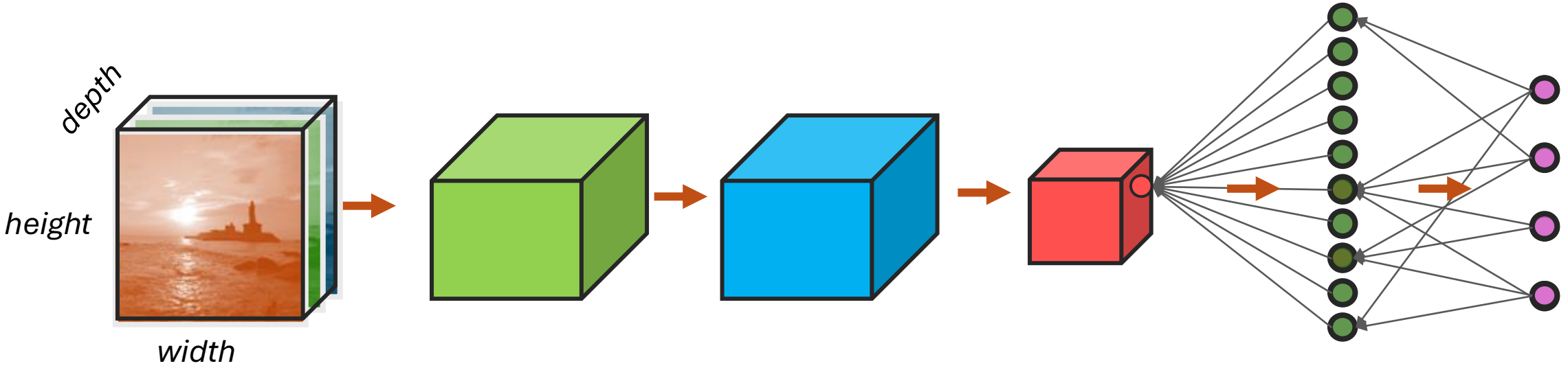


DNN's Learnable Parameters Distribution

Learned parameter of all failed models, suboptimal models and successes full models clearly show similarity of their statistical properties and weight distribution irrespective of their model size. i.e. models of any size converge to a similar weight distribution when trained to a fixed number of epochs on a dataset



Convolutional Neural Nets



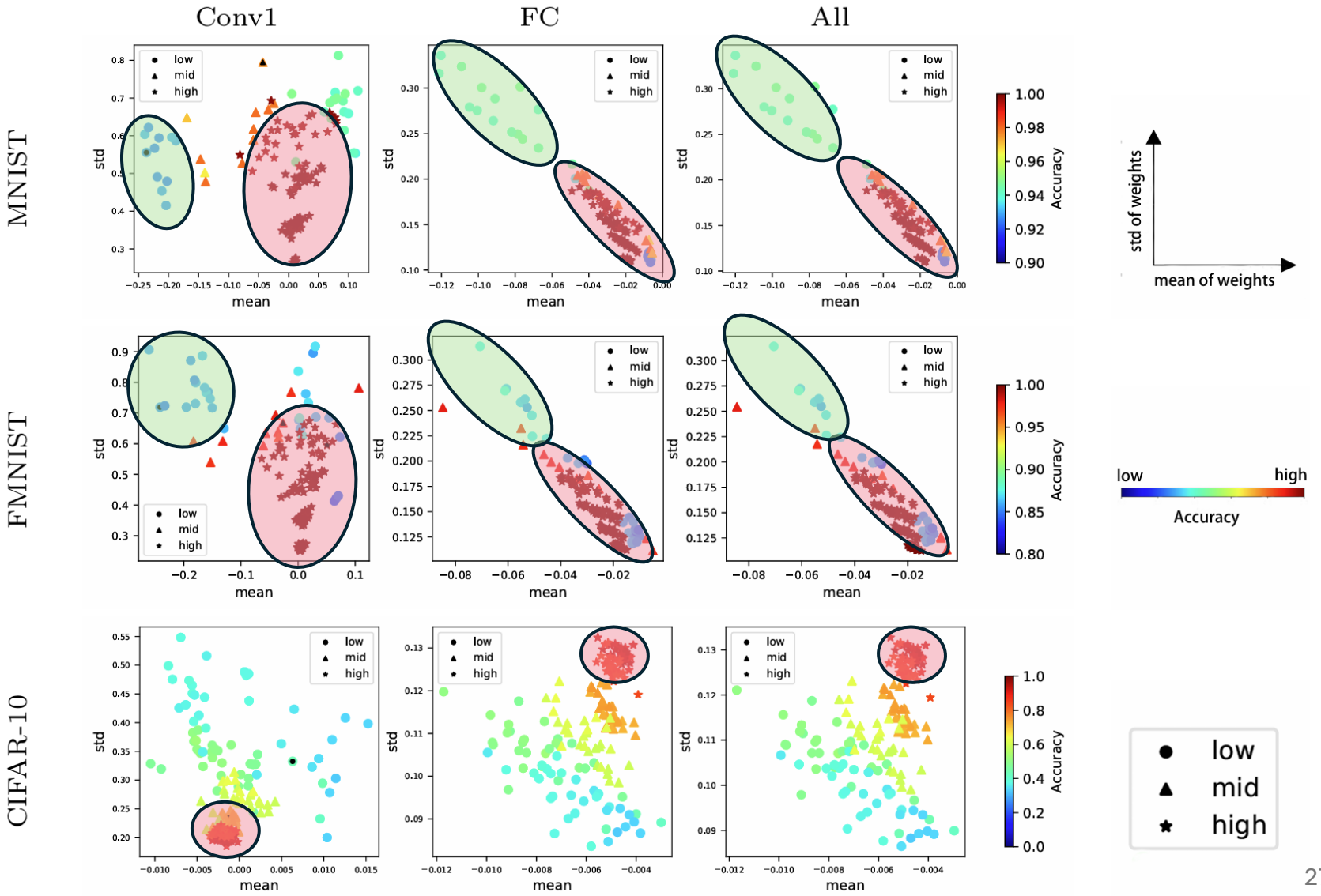
[Input - Convolution - RELU - POOLING - Fully Connected - Output]

CNN's Learnable Parameters

Similar complexity data shows similar trainable parameter **statistical property pattern** (weight distribution), irrespective of model size

High performing model weights show that models of any size converge to parameter with **similar statistics** (weight distribution)

High complexity data shows that statistical property of successful models has very **specific statistical property** or specific weight distribution

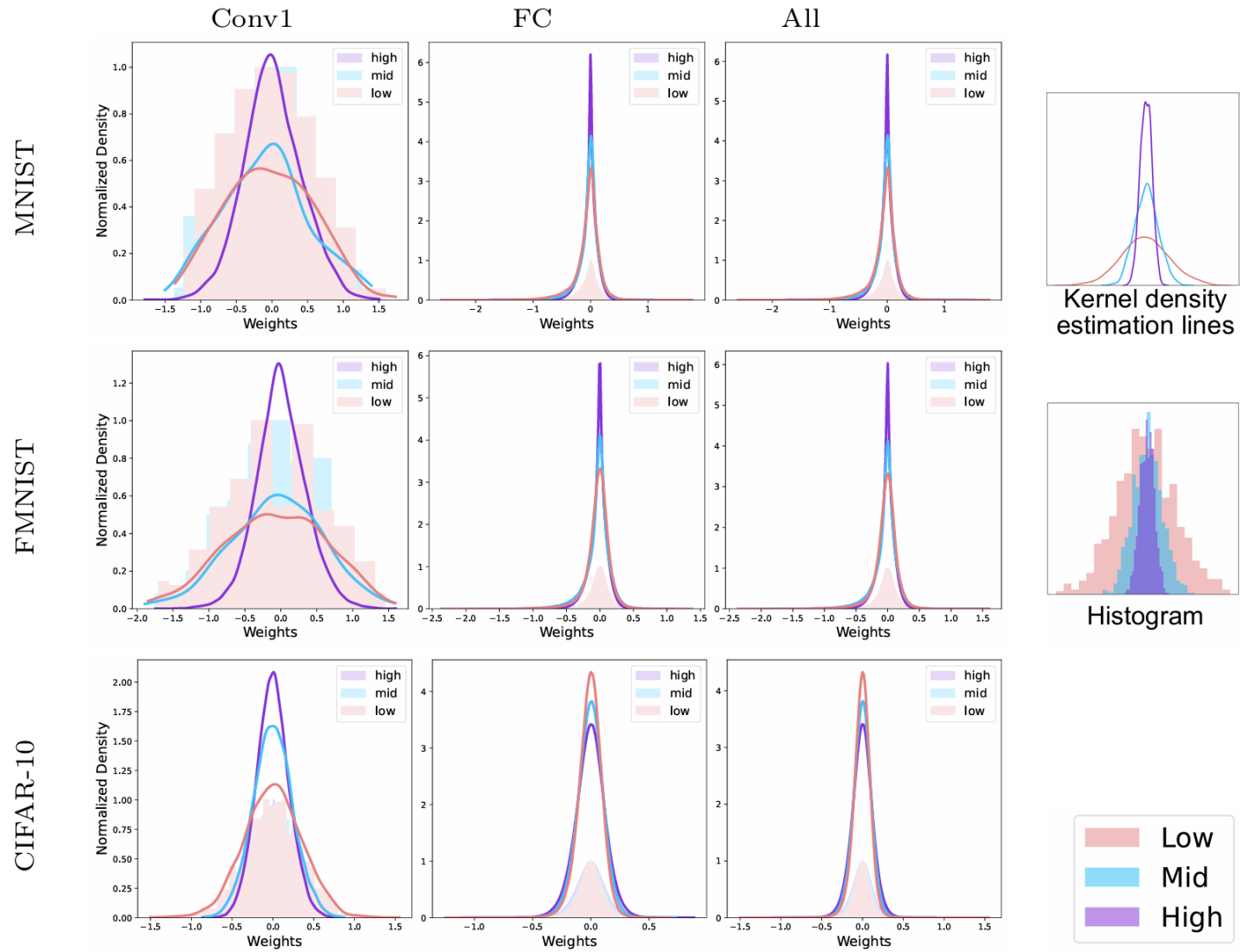


CNN's Learnable Parameters Distribution

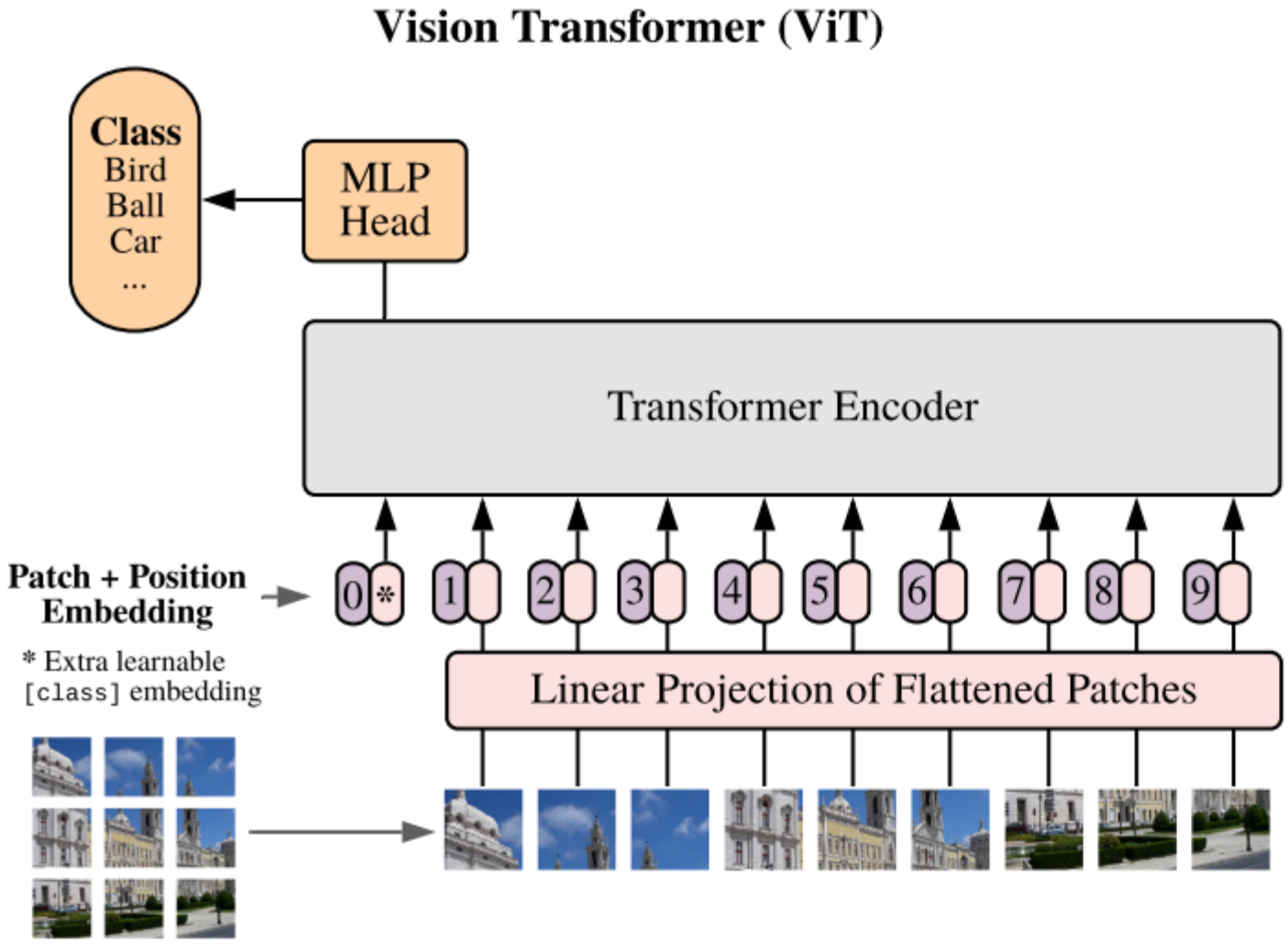
Similar complexity data shows similar trainable parameter statistical property pattern (**weight distribution**), irrespective of model size

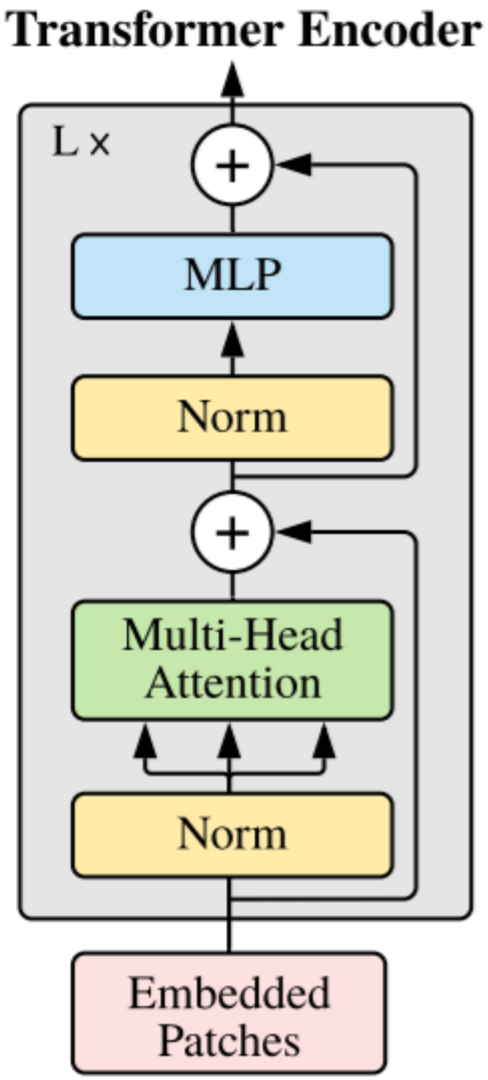
High performing model weights show that models of any size converge to parameter with similar statistics (**weight distribution**)

High complexity data shows that statistical property of successful models has very specific statistical property or **specific weight distribution**



Vision Transformer Models

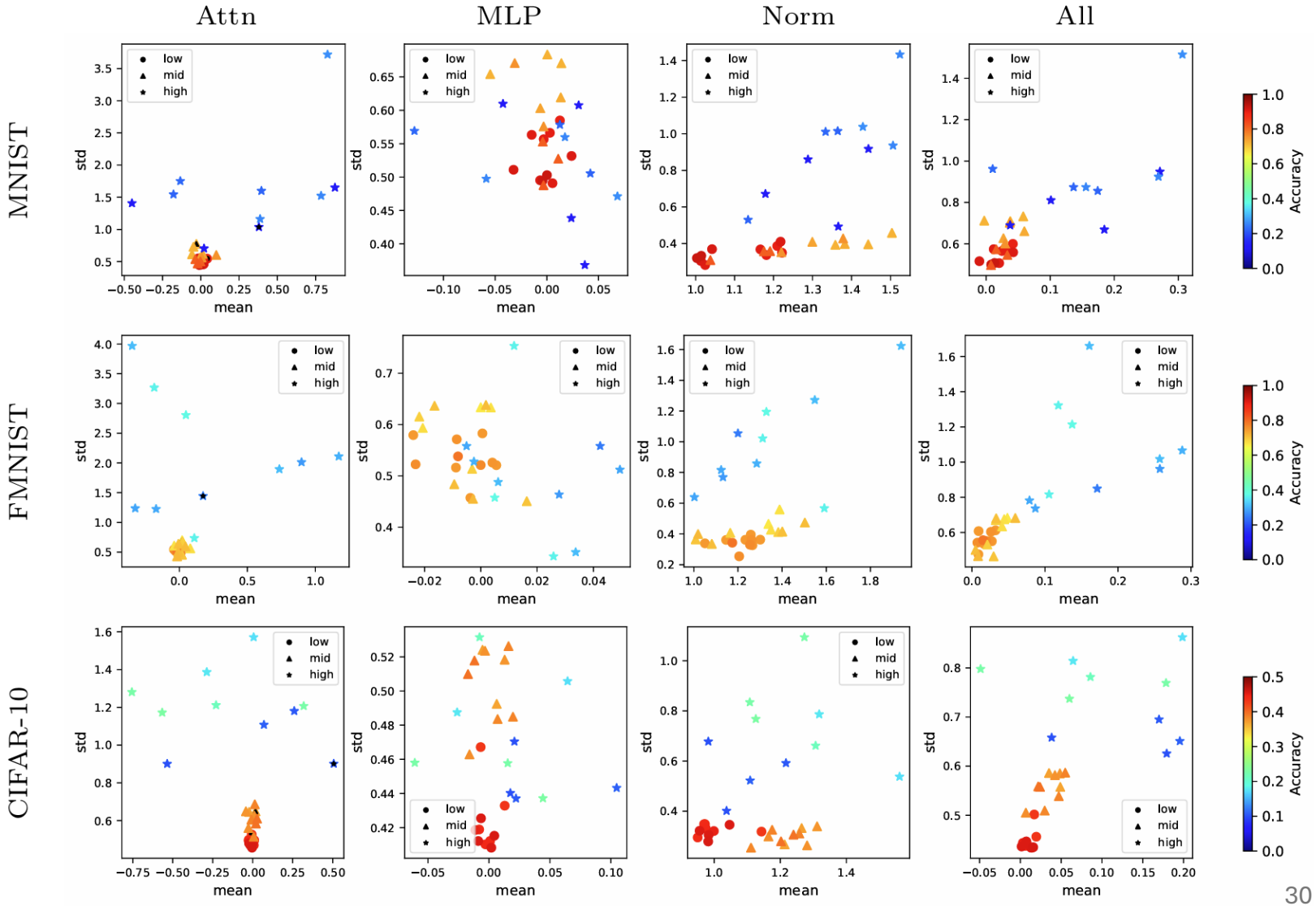




Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2017

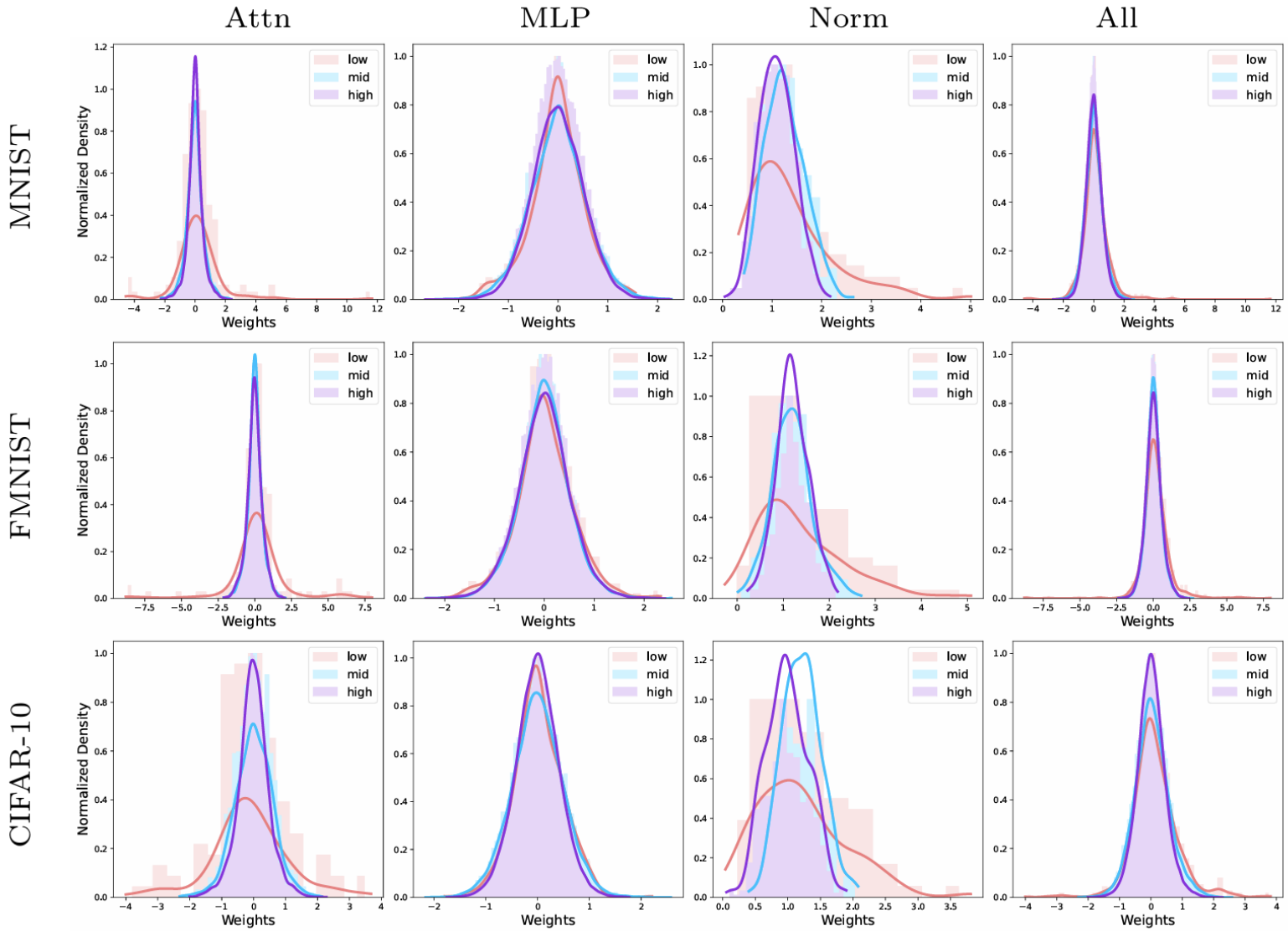
ViT’s Learnable Parameters

All failed models, suboptimal models and successes full model's learned parameter clearly show similarity of their statistical properties (weight distribution) **irrespective of Model Architecture**



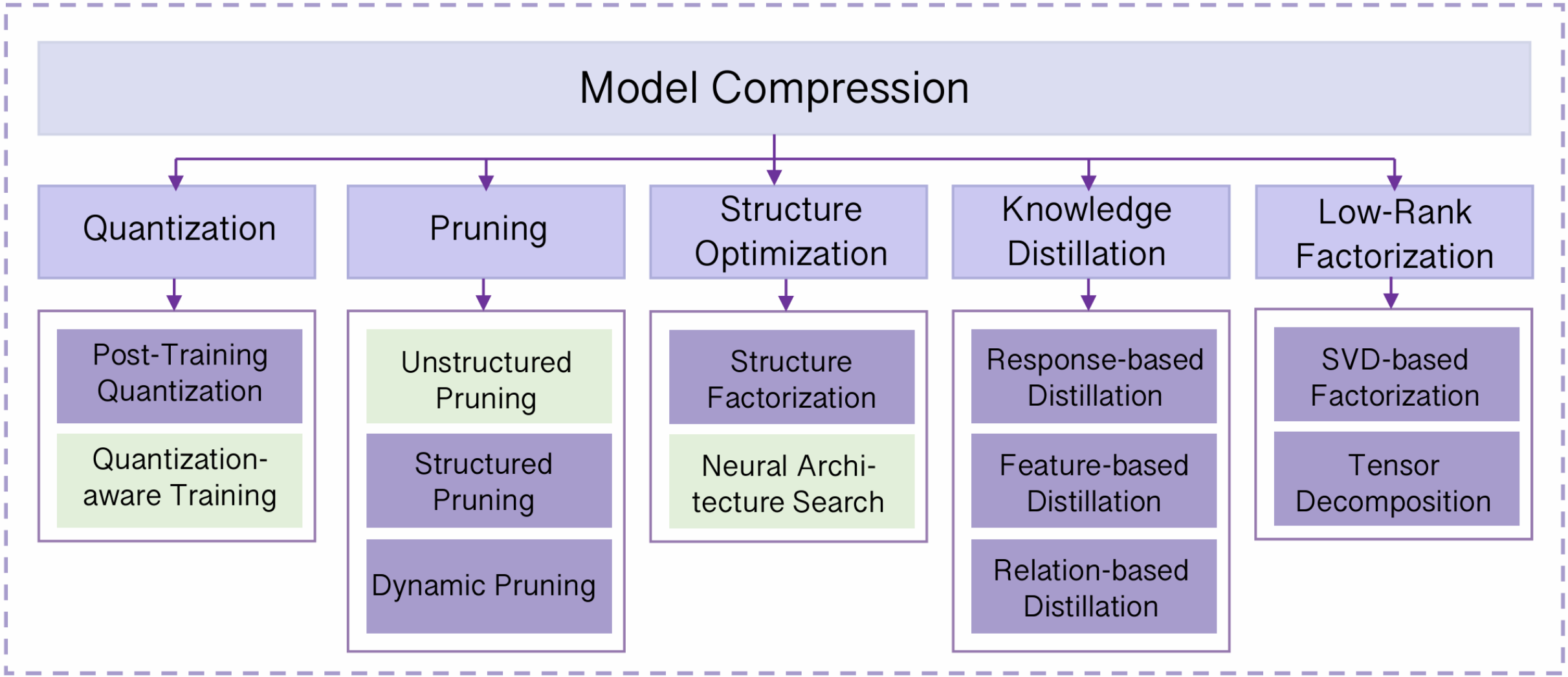
ViT’s Learnable Parameters Distribution

All failed models, suboptimal models and successes full model's learned parameter clearly show similarity of their statistical properties (weight distribution) **irrespective of Model Architecture**



Part 3
AI Model Compression Methods and Results

AI Model "Slimming" Techniques



AI Model "Slimming" Techniques

Quantization

Reduces model precision (32-bit to 8-bit), cutting memory by ~75% with minimal accuracy loss.

Pruning

Eliminates less important weights to create smaller, sparser models while maintaining accuracy.

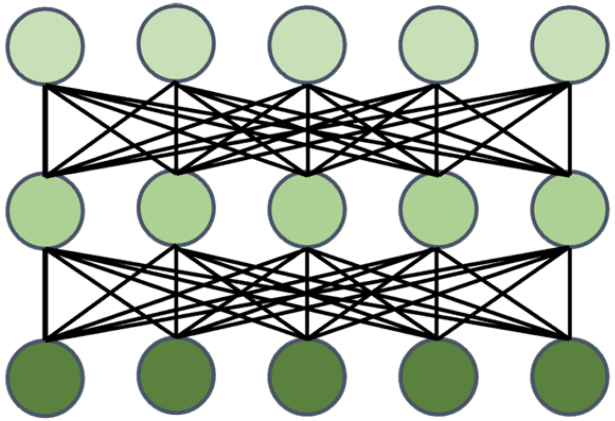
Knowledge Distillation

Trains a compact student model to replicate behaviors of a large teacher model.

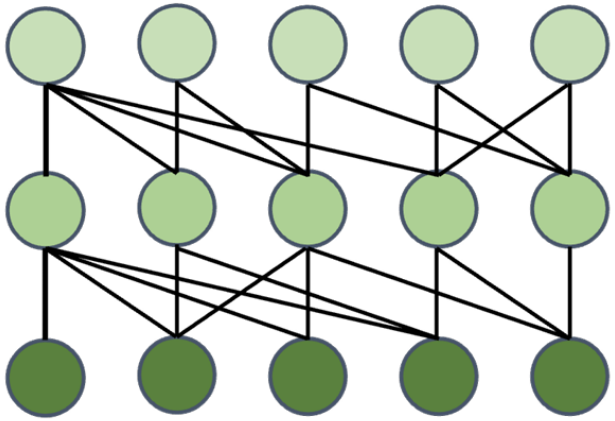
Model Splitting

Partitions processing between edge device and cloud for co-inference.

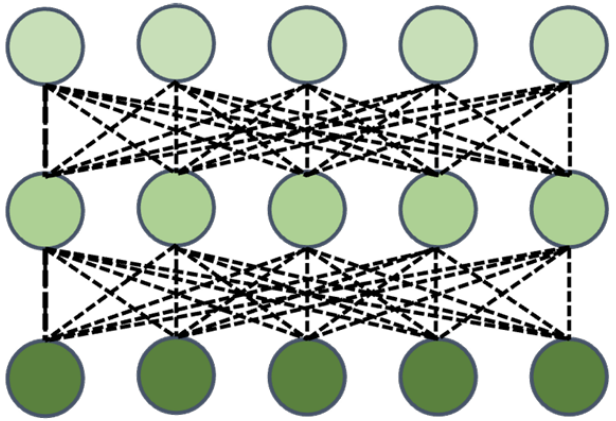
Pruning and Quantization Aware Neural Network Training



32-bit network



pruned network



8-bit network

Ziwie, Ojha, et al (2025)

Pruning

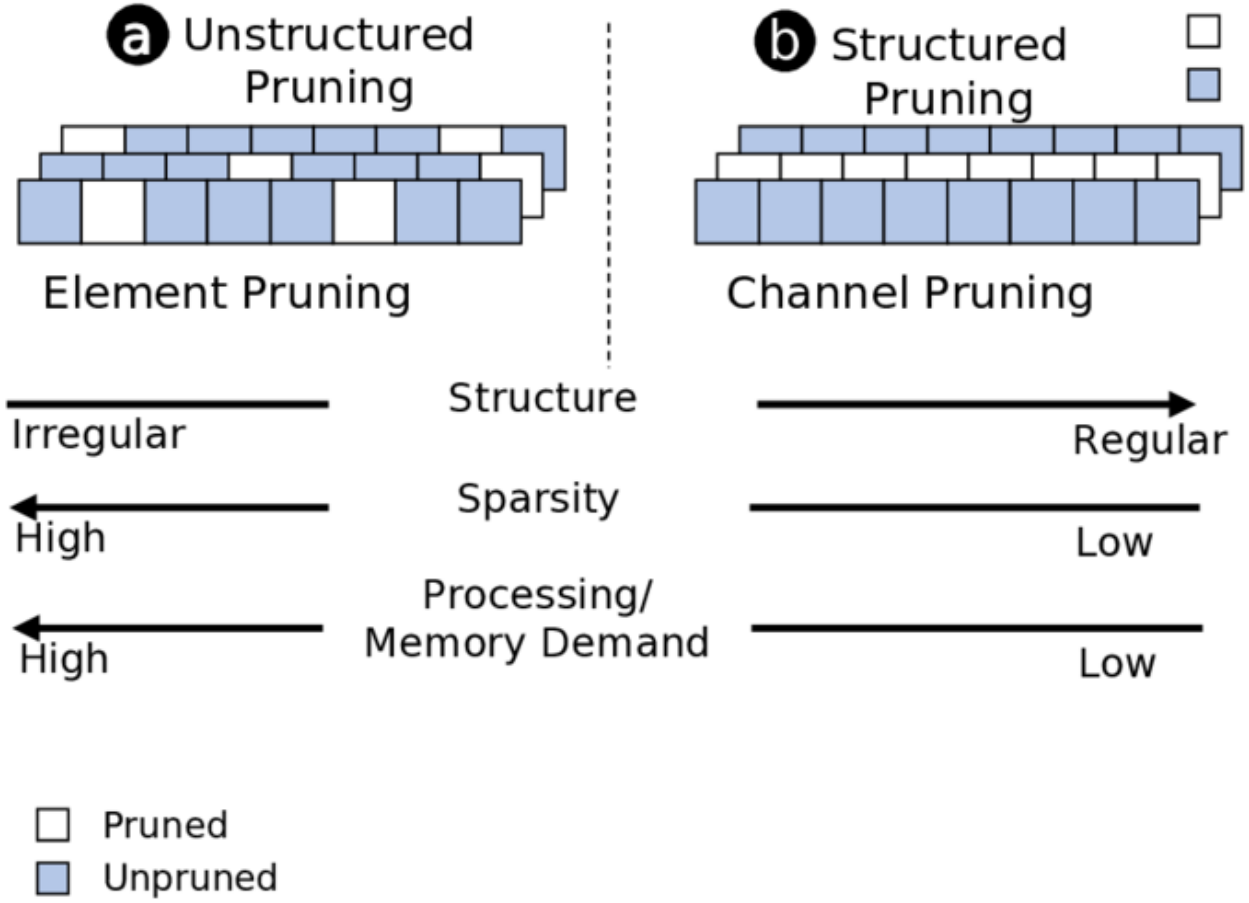
- **Train-time (dynamic) pruning** involves integrating the pruning process directly into the training phase of the neural network. During training, the model is trained in a way that encourages sparsity or removes less important connections or neurons as part of the optimization process

- **Unstructured pruning** involves zeroing individual weights within the weight matrices

- **Structured pruning** removing entire structured groups of weights, the method reduces the scale of calculations that would have to be made in the forward pass through the model's weights graph

Pruning

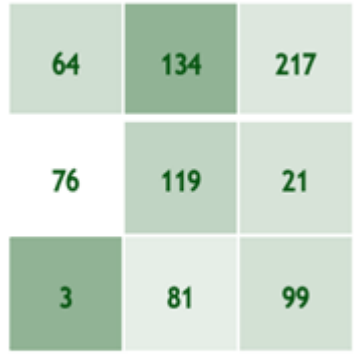
- **Magnitude-based Pruning**: Simple, small weights/channels are pruned
- **Gradient-based Pruning:** method prunes weights that show smaller gradients over time.
- **Importance-based Pruning:** Weights with lower “importance scores” are pruned first
- **Training-time (dynamic) pruning**: dynamic pruning adjusts the network structure during training based on real-time performance metrics



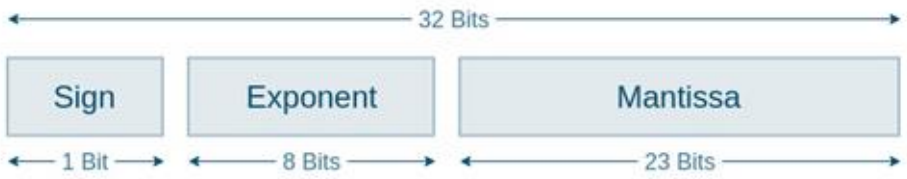
Quantization

The main task of model quantization in deep learning is to convert high-precision floating-point numbers in neural networks into low-precision numbers

FP32

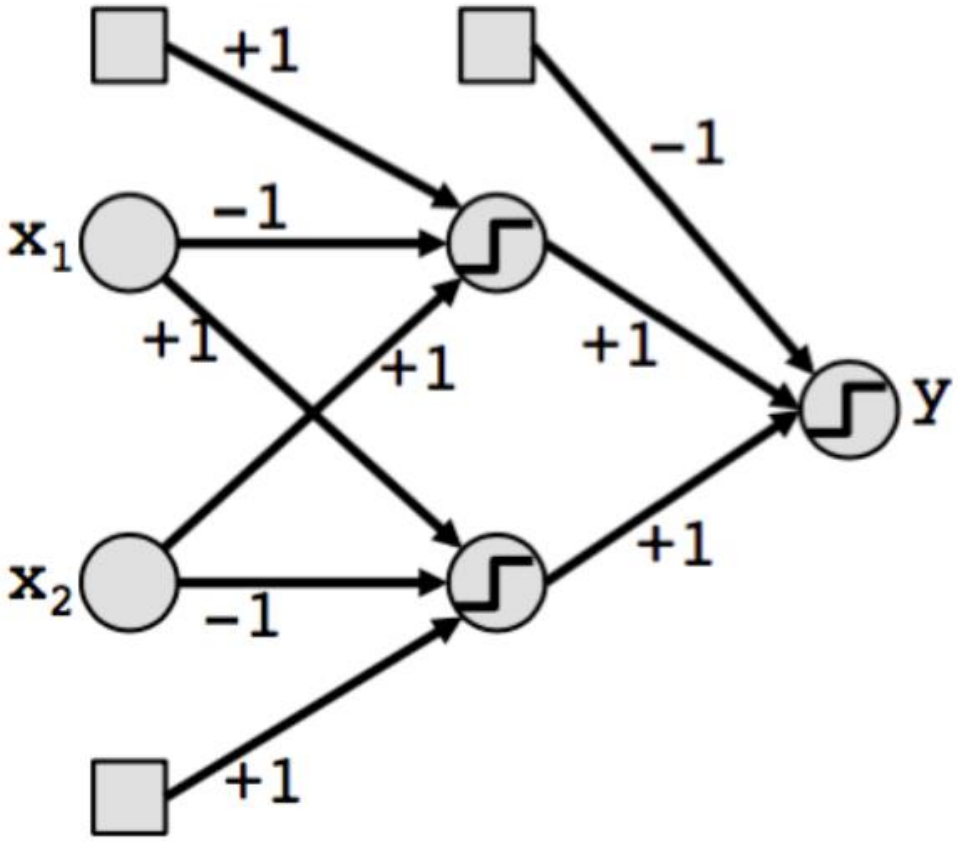


INT8



Single Precision
IEEE 754 Floating-Point Standard

Binarized Neural Networks



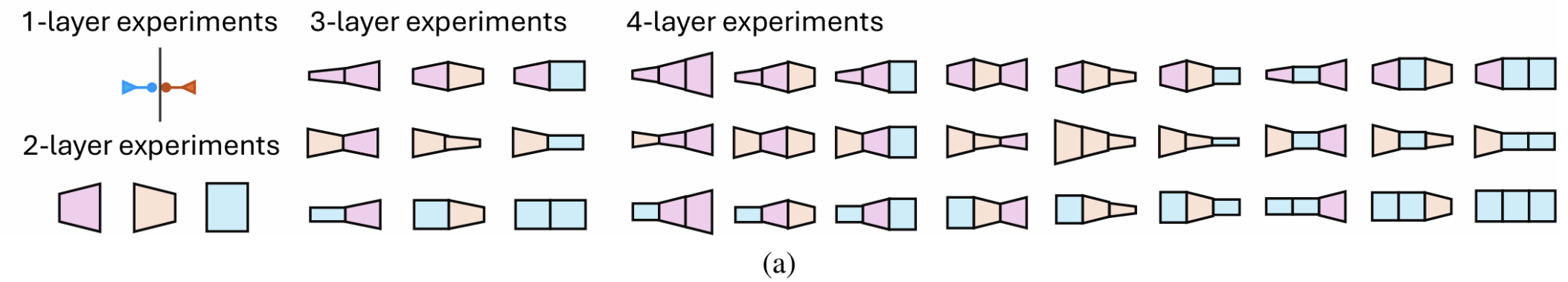
Weights and activation of Neural Networks:

$$x^b = \text{Sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise} \end{cases}$$

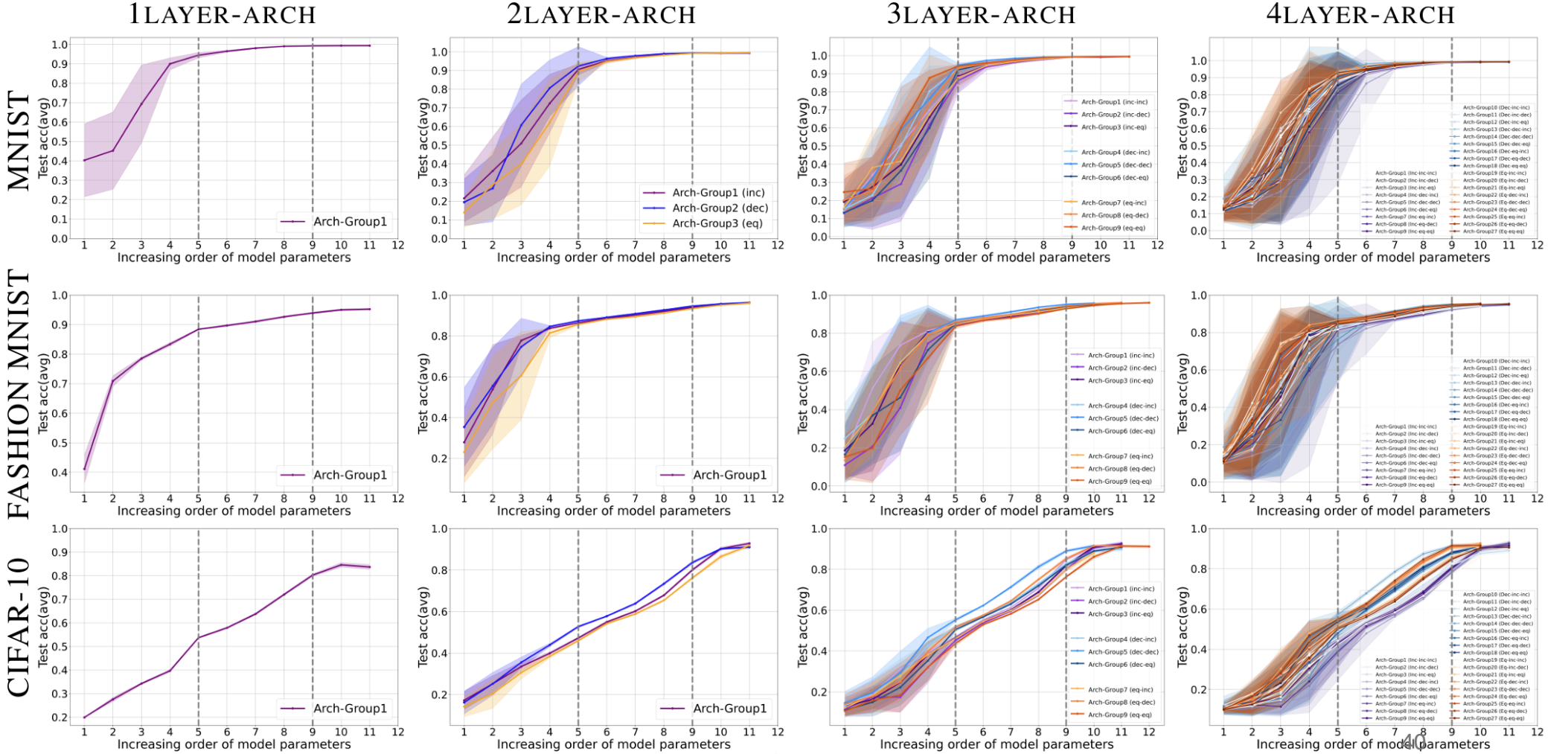
$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x), \\ -1 & \text{with probability } 1 - p, \end{cases}$$

Binarized Neural Nets: (Courbariaux et al. 2016) https://arxiv.org/pdf/1602.02830

Architectural Invariance

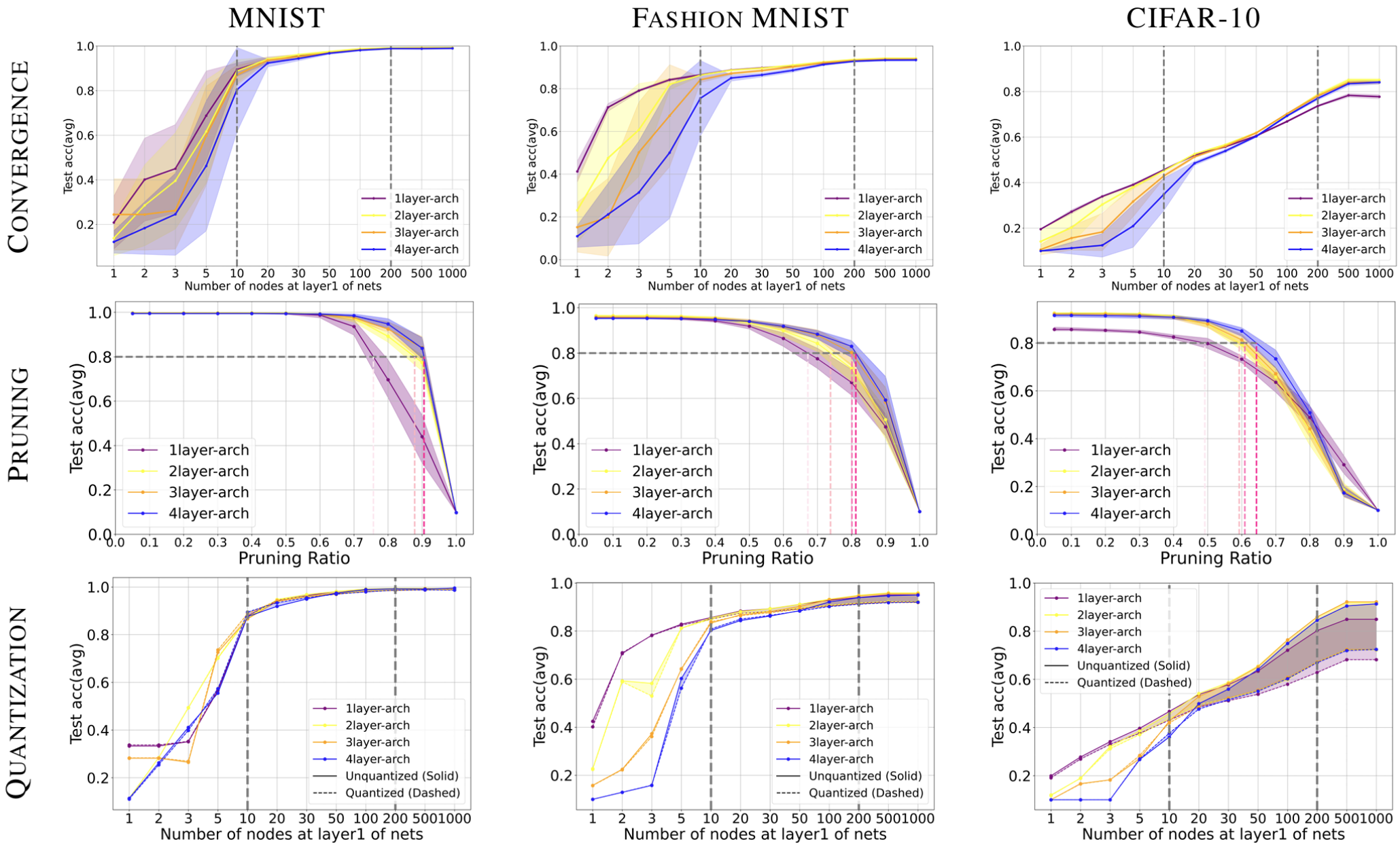


(a)

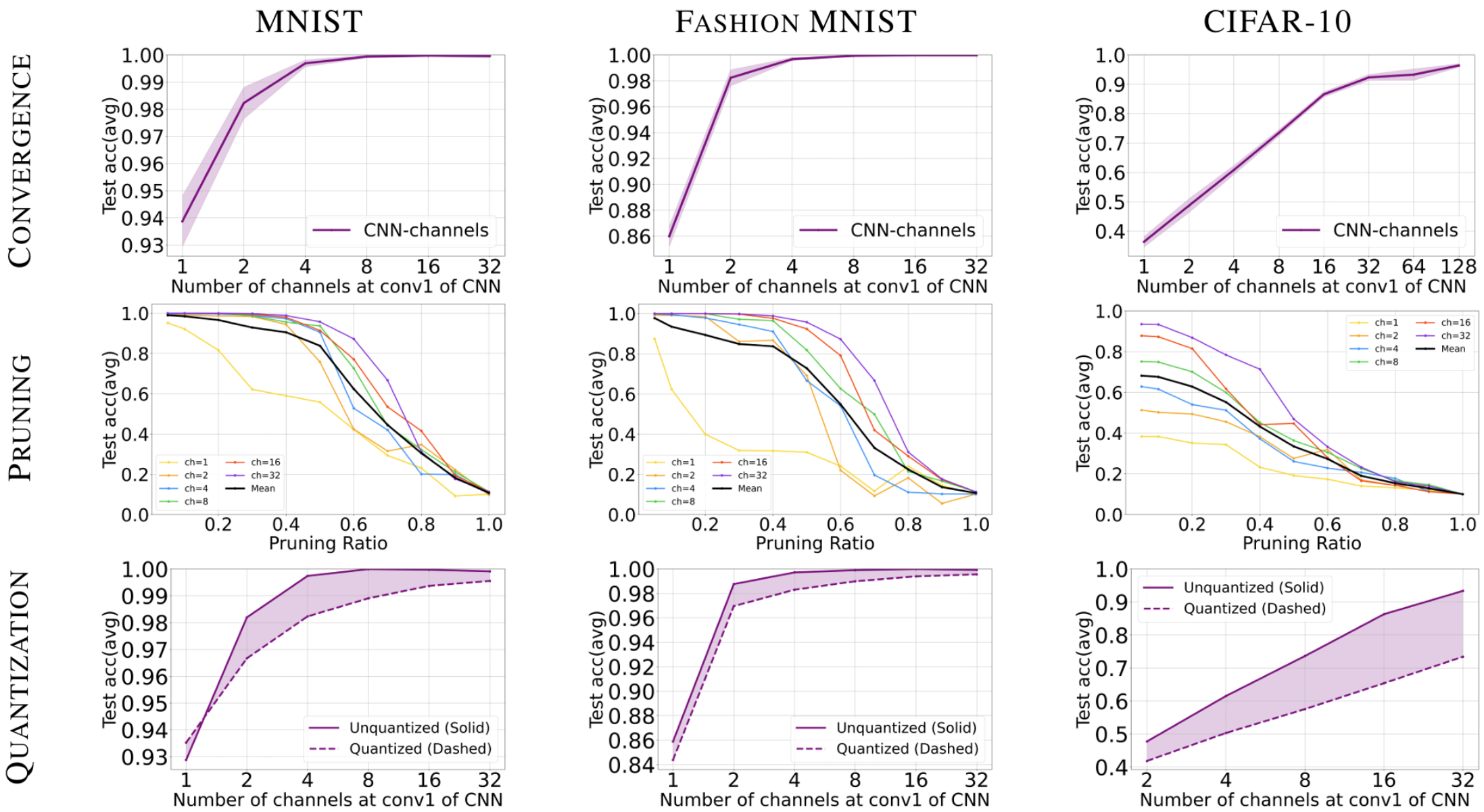


(b)

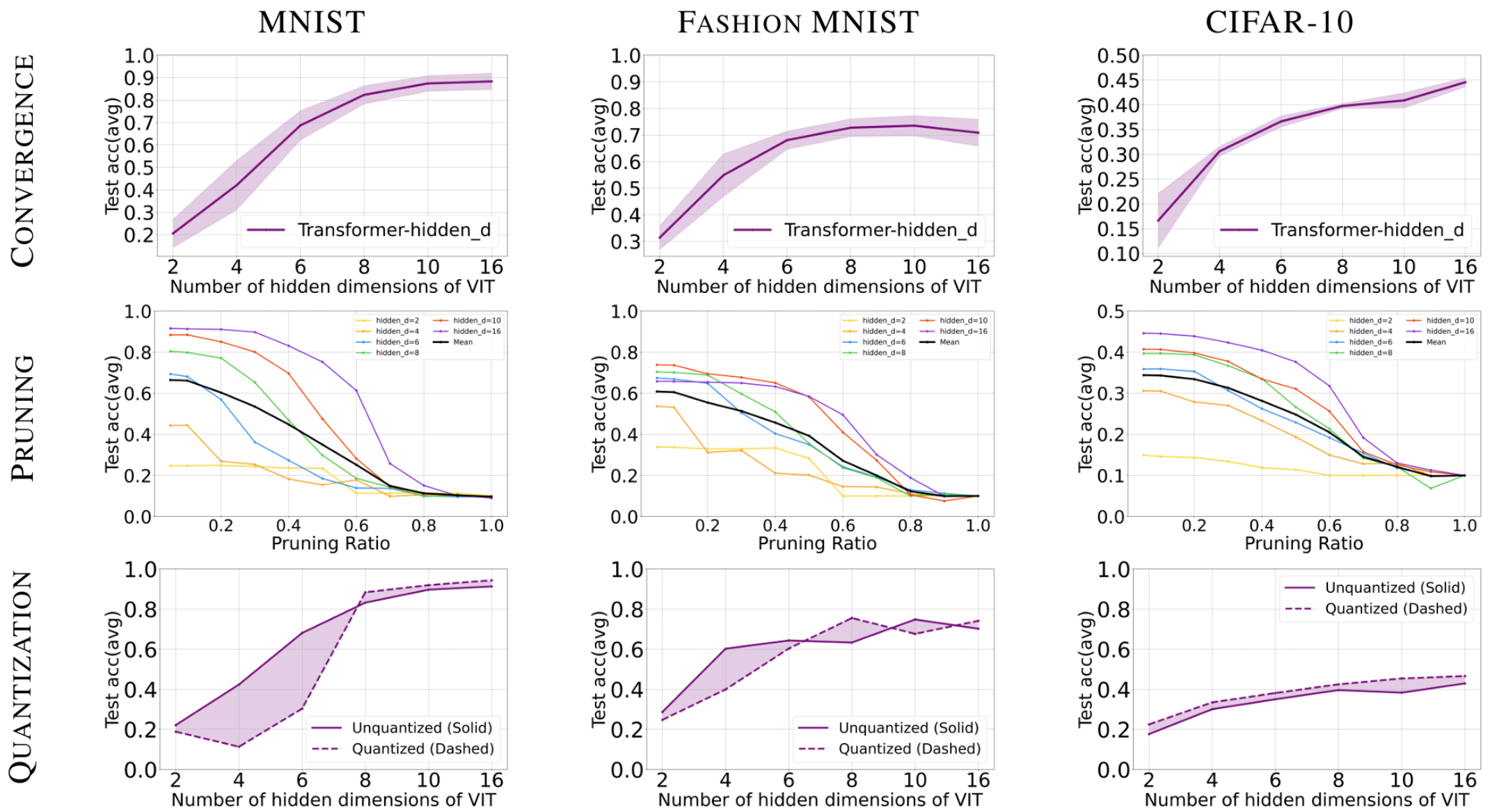
DNN Architecture



CNN Architecture



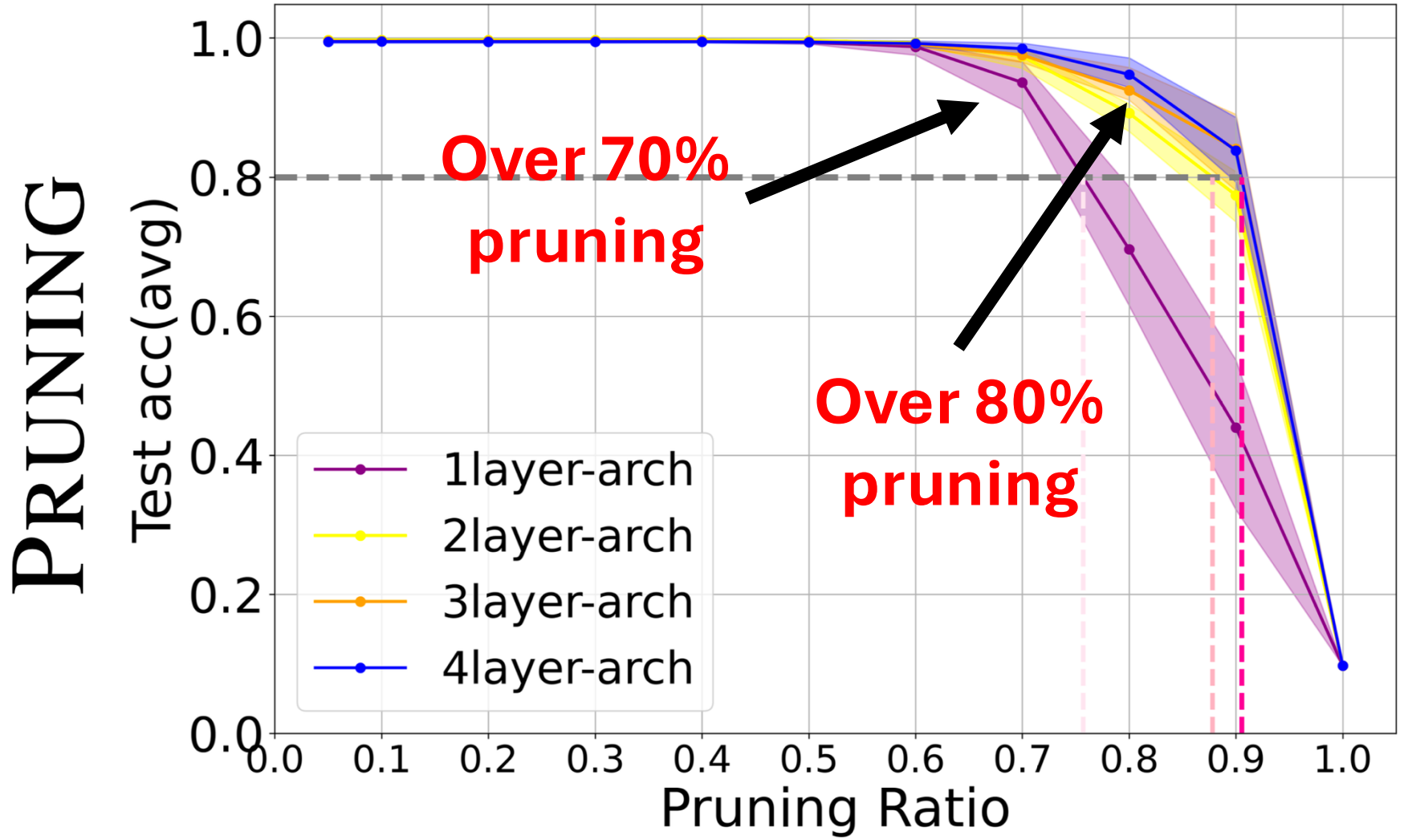
ViT Architecture



Maximal Pruning for Minimal Network

test results on MNIST

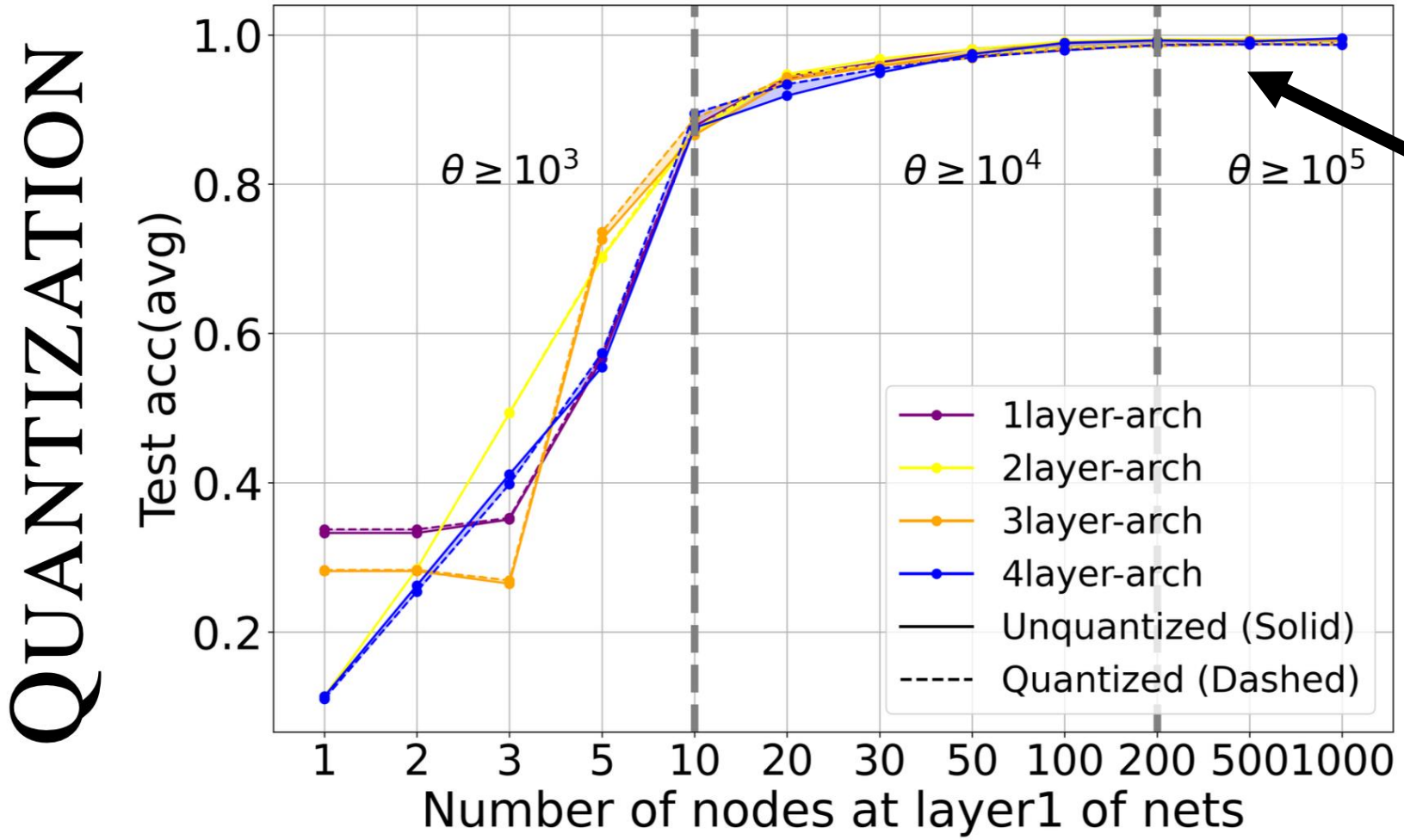
similar results of CIFAR 100 and other 8 datasets



Maximal Quntizatian for Minimal Network

test results on MNIST

similar results of CIFAR 100 and other 8 datasets

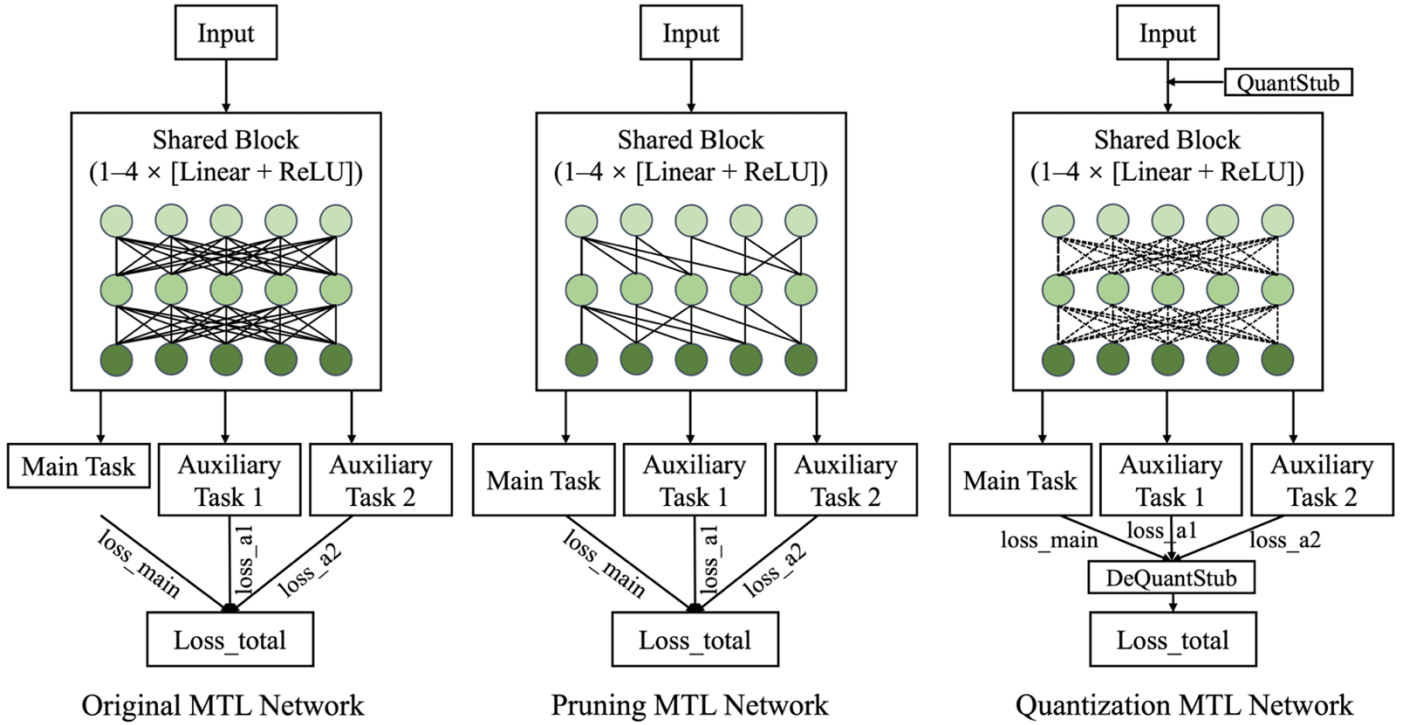


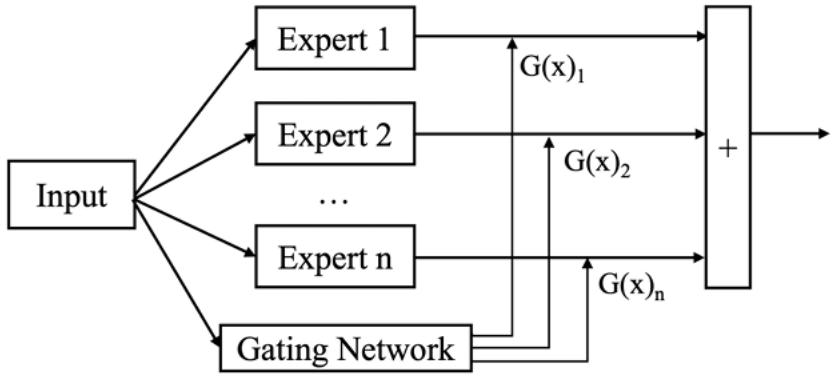
Minimal degradation on 8-bit Nets

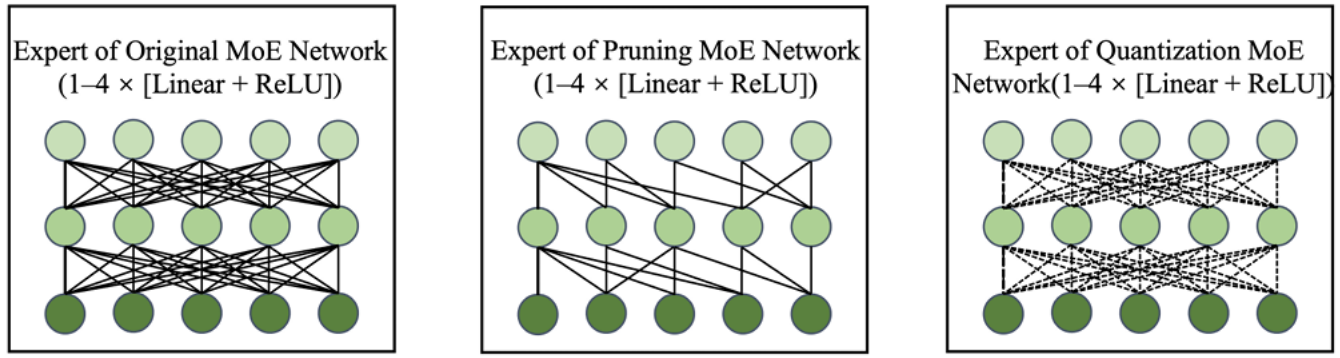
Summary of Pruning and Quantization on Models

Dataset	Architecture	Min. Params (Stability)	Safe Pruning %	8-bit Gap %
MNIST	DNN	2×10^4	60	0.15
	CNN	3×10^4	20	0.74
	ViT	5×10^3	20	10.66
F-MNIST	DNN	2×10^4	40	1.42
	CNN	3×10^4	20	1.10
	ViT	5×10^3	20	3.22
CIFAR-10	DNN	2×10^6	40	7.50
	CNN	3×10^6	5	14.80
	ViT	6×10^3	10	4.17

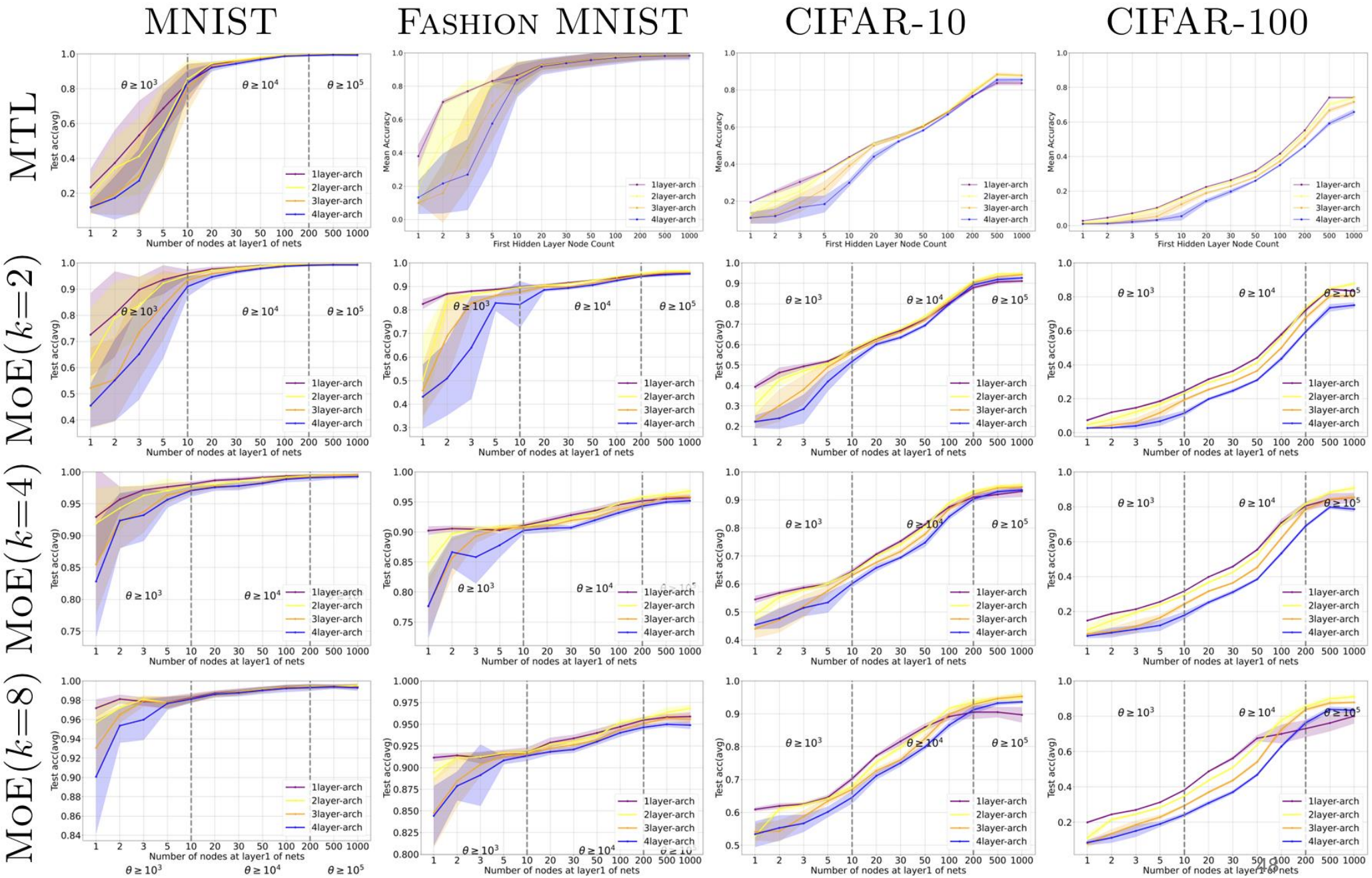
Multitask Learning and Mixture of Experts



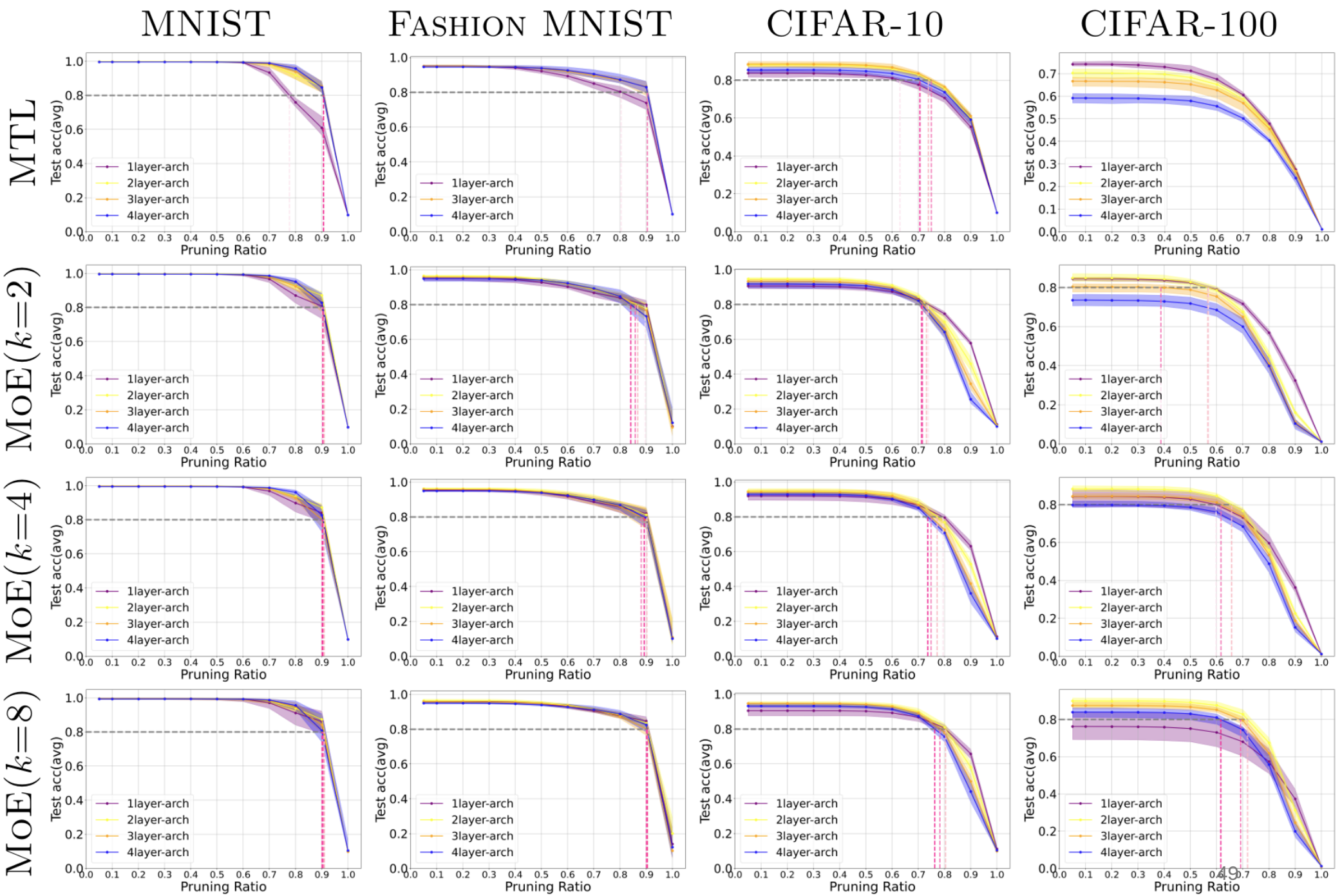




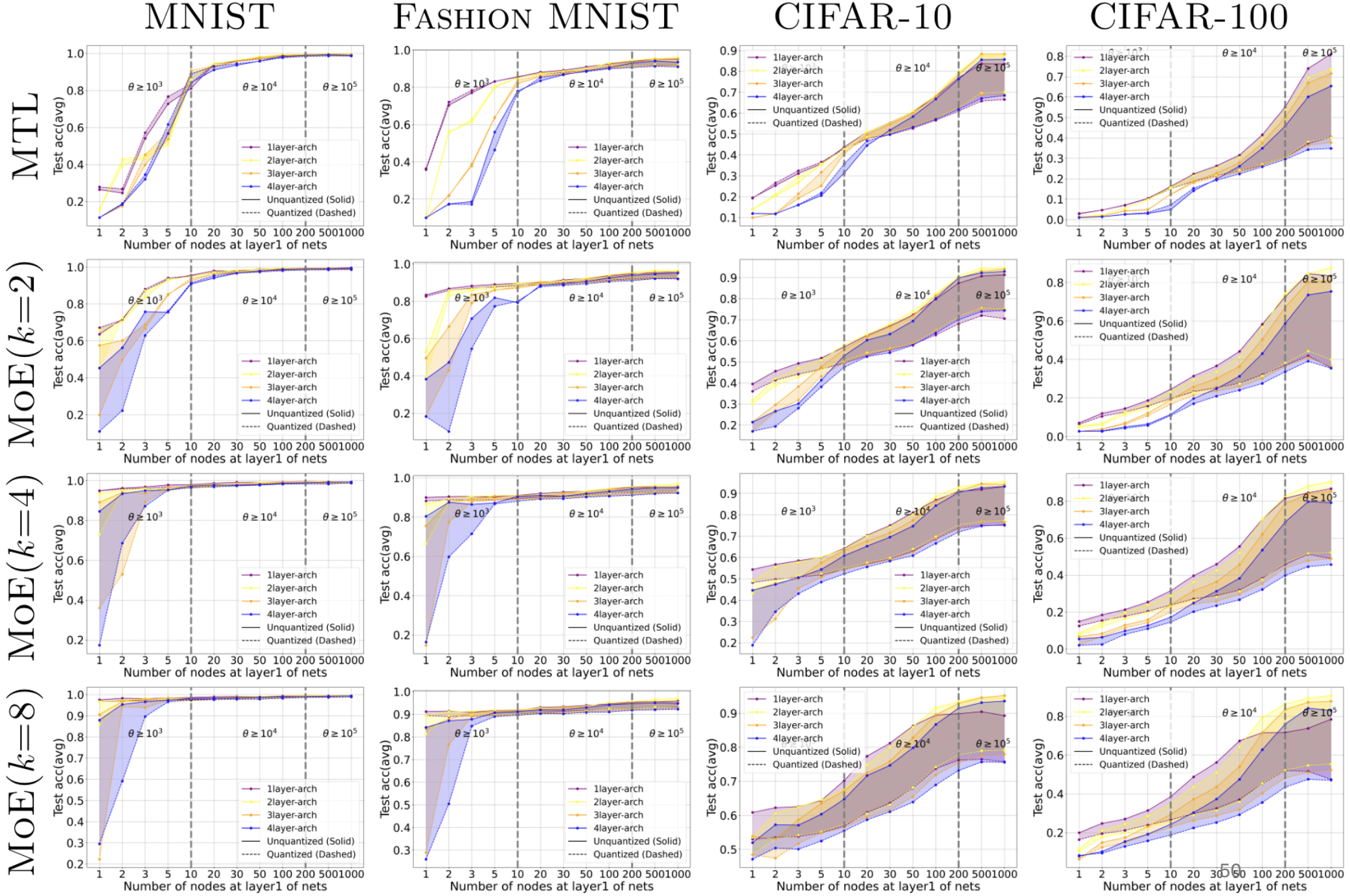
MTL and MoE Convergence



MTL and MoE Pruning

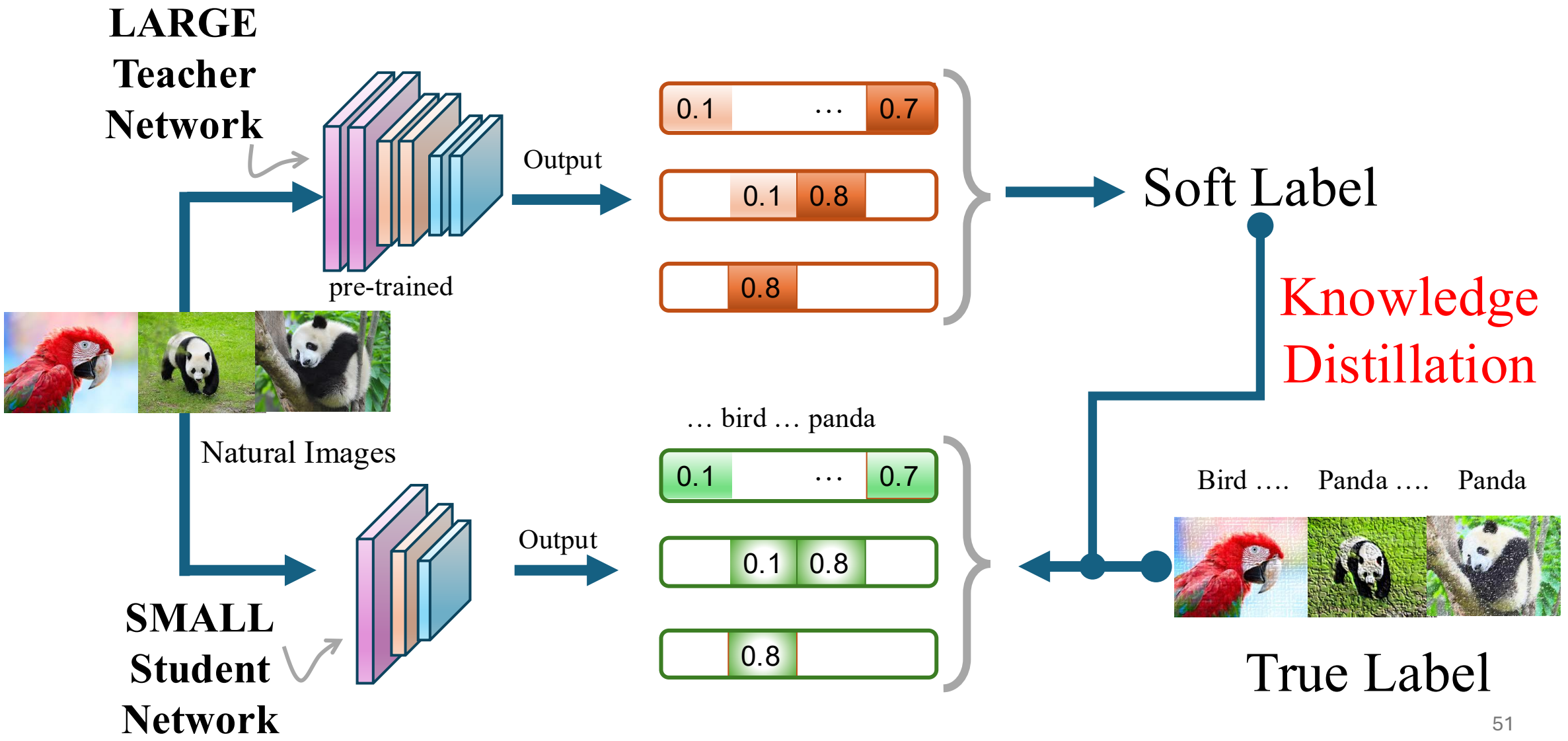


MTL and MoE Quantization



Knowledge Distillation and Model Robustness

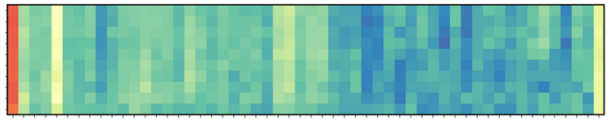
Zhenyu, Ojha et al (2024), ICONIP



Compression Using “Parameter Score” -> “ “Importance Score”

ResNet18 model on ImageNet before filtering ~45 MB

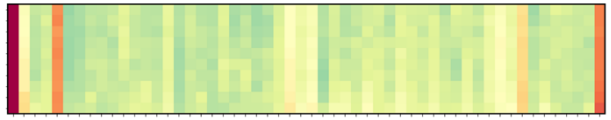
Model layers and parameters

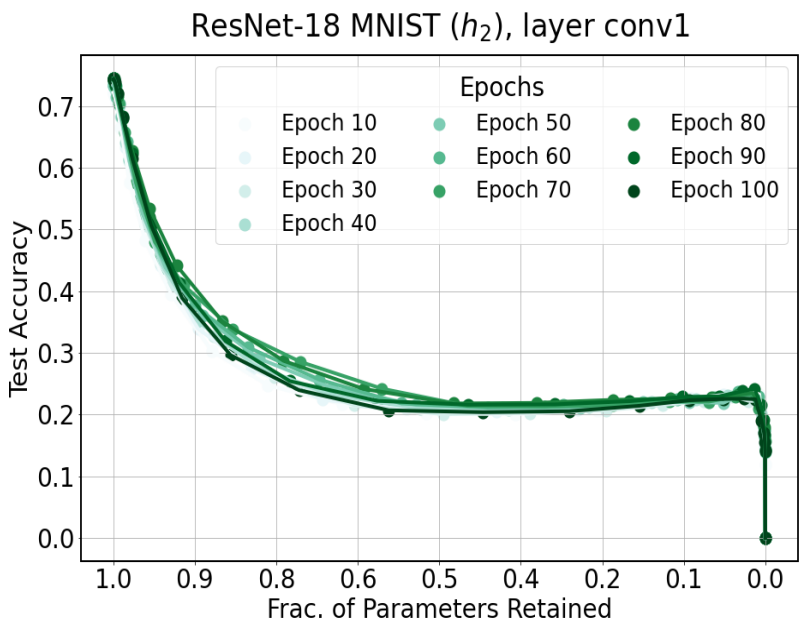


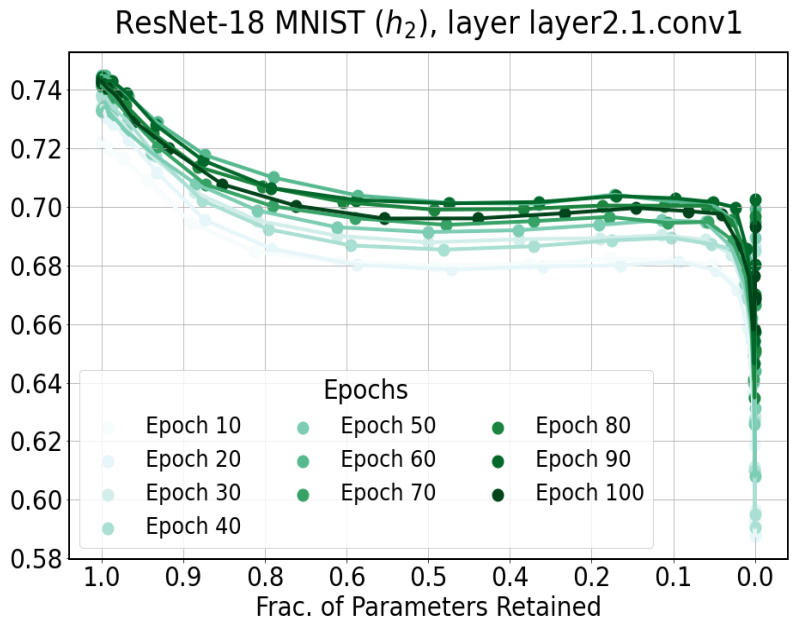
Singla processing funnels (filters)

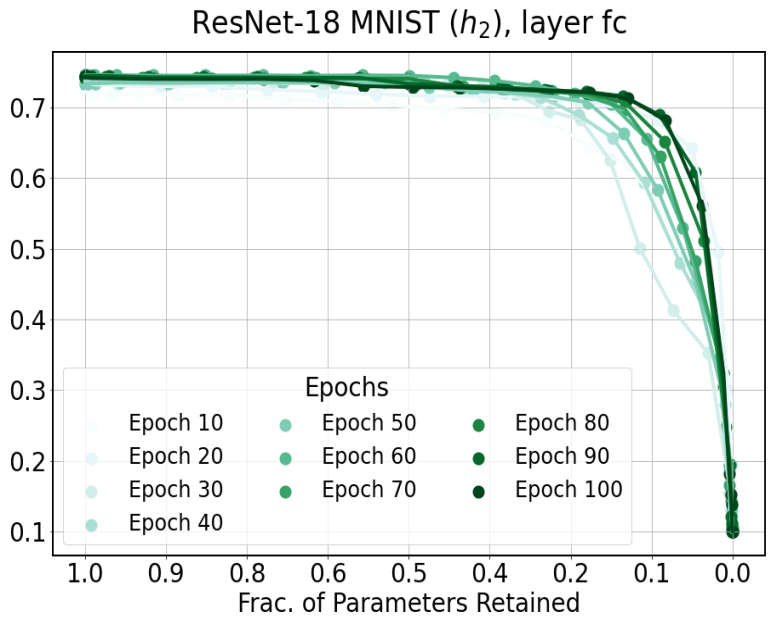
ResNet18 model on ImageNet after filtering

Model layers and parameters









Pravin, Ojha et al (2024), AIJ

Optimization Techniques for Edge Deployment

Beyond quantization, pruning, and distillation, several other methods can enhance the deployment of large models on edge and IoT devices:

Neural Architecture Search (NAS)

Automates the design of efficient neural networks tailored for specific hardware constraints.

Hardware-Aware Training (HAT)

Incorporates hardware characteristics during training to optimize models for particular devices.

Sparsity-Inducing Regularization

Encourages models to develop sparse weight matrices during training, leading to smaller and faster models.

Winograd Transformations

Optimizes convolutional operations to reduce the number of multiplications, improving inference speed.

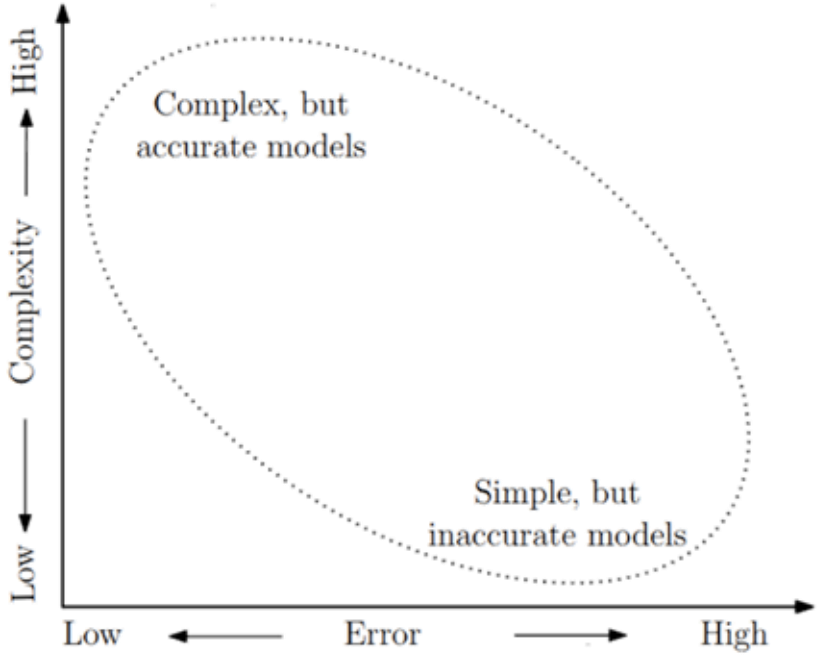
Fast Fourier Transforms (FFTs)

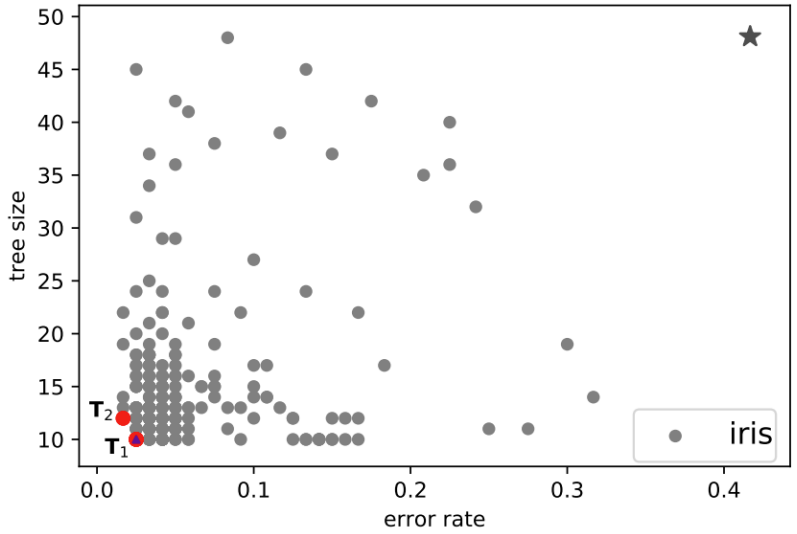
Efficiently performs convolutions in the frequency domain, particularly useful for large kernel sizes.

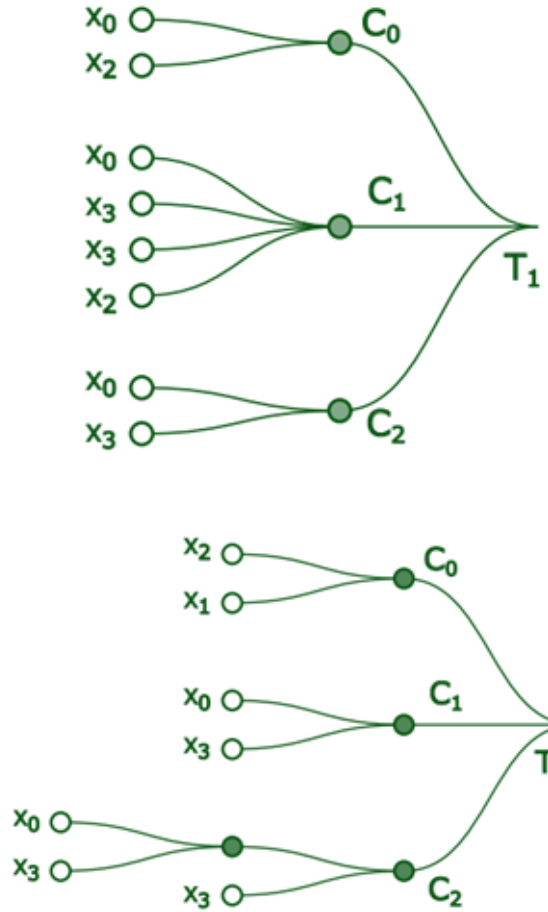
Part 4
Search for Efficient AI Models

Neural Architecture Search (NAS)

Catering the need of specialized problem and hardware





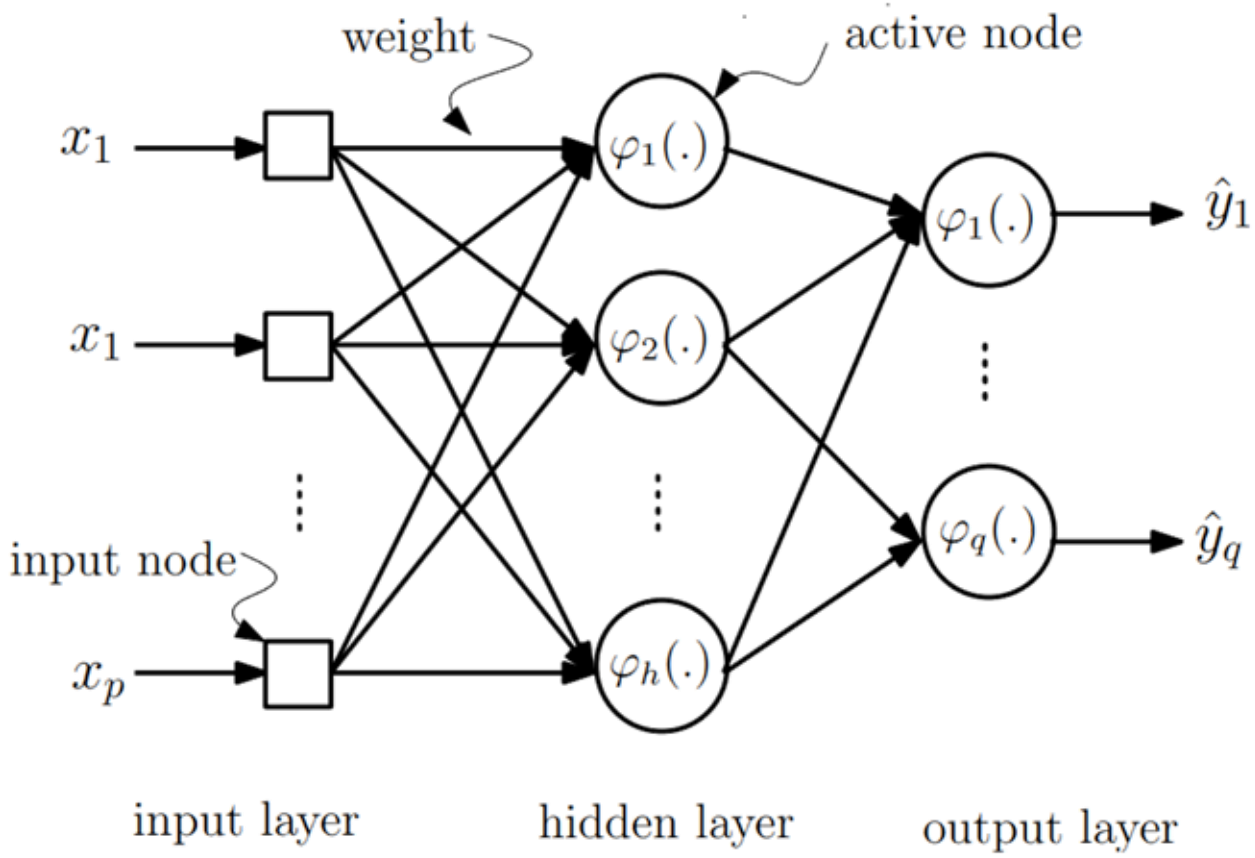


Ojha et al (2020), CEC

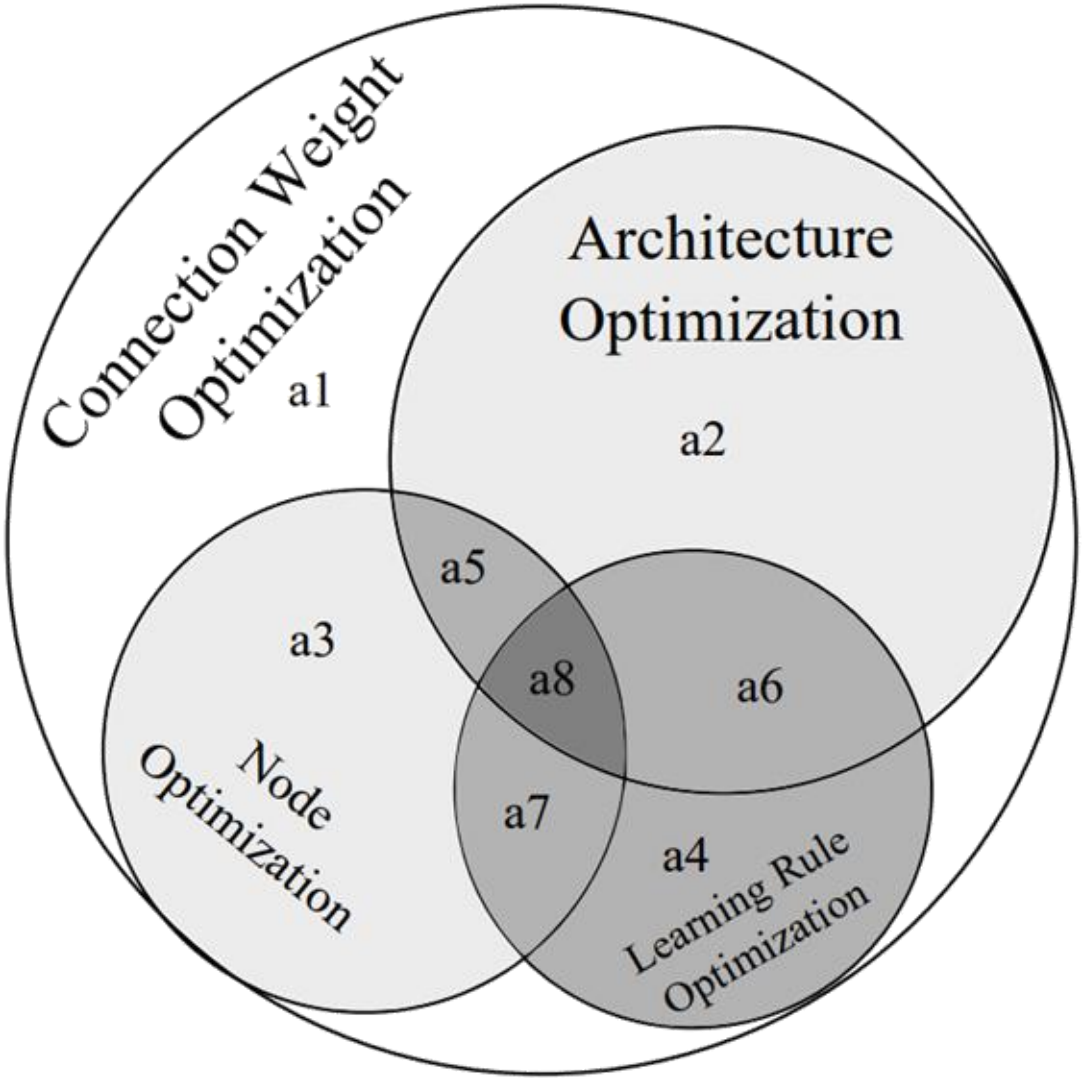
Neural Networks

NN components:

- Inputs
- Weights
- Architecture
- Activation functions
- Learning algorithms

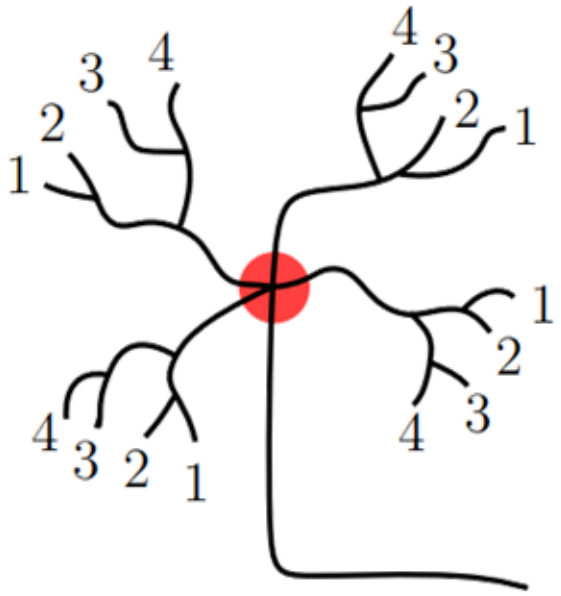


What Could be optimized?

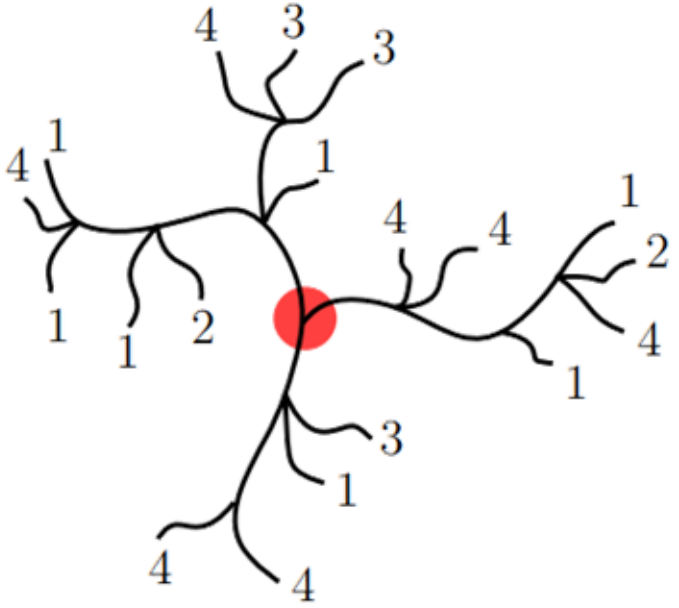


Plausible Biological Inspiration

Travis et al. (2005)



Jones and Kording (2021)

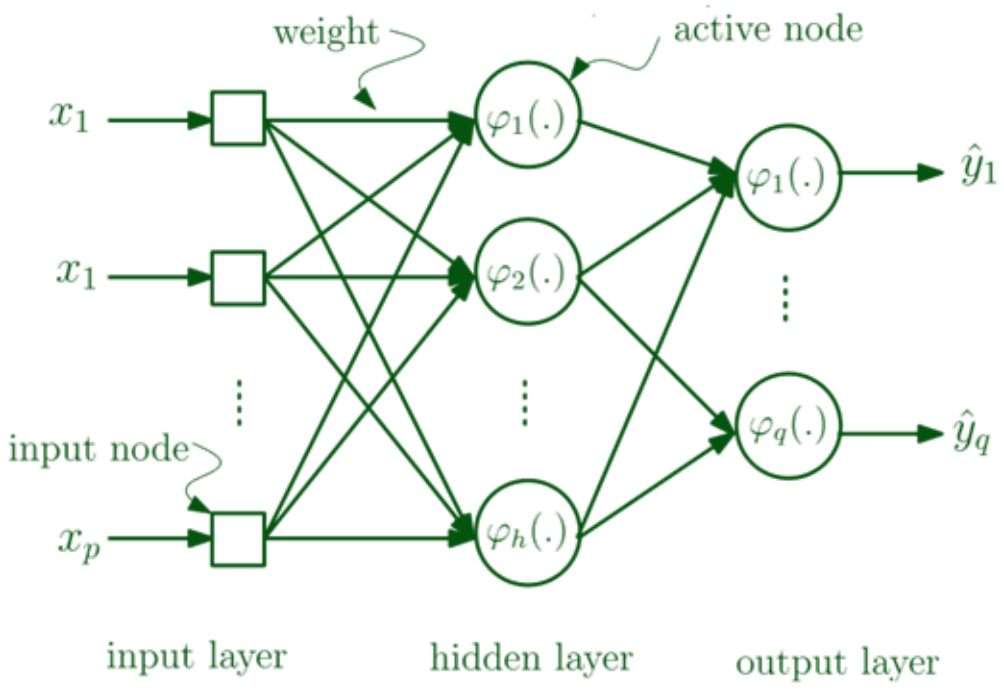


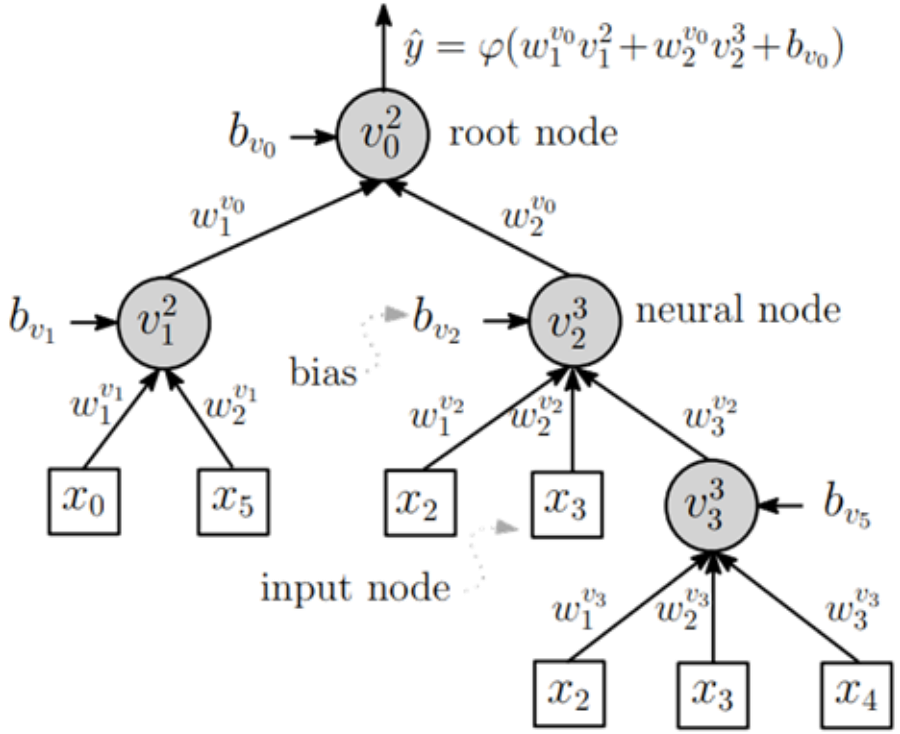
Ojha and Nicosia (2022)

Ojha et al (2022), Neural Networks

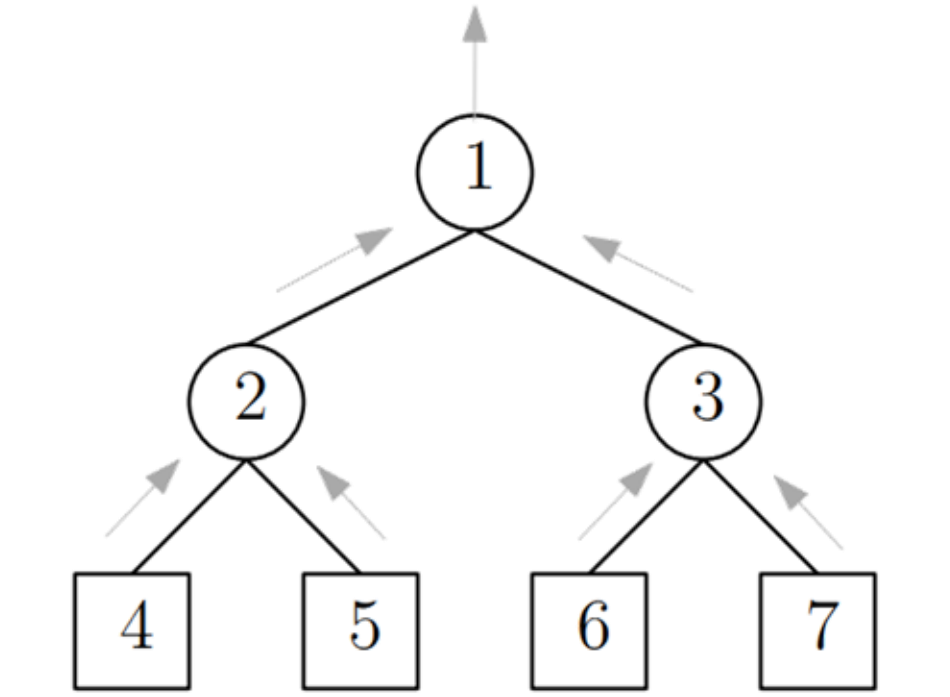
Neural Tree

Neural Networks Architecture Search





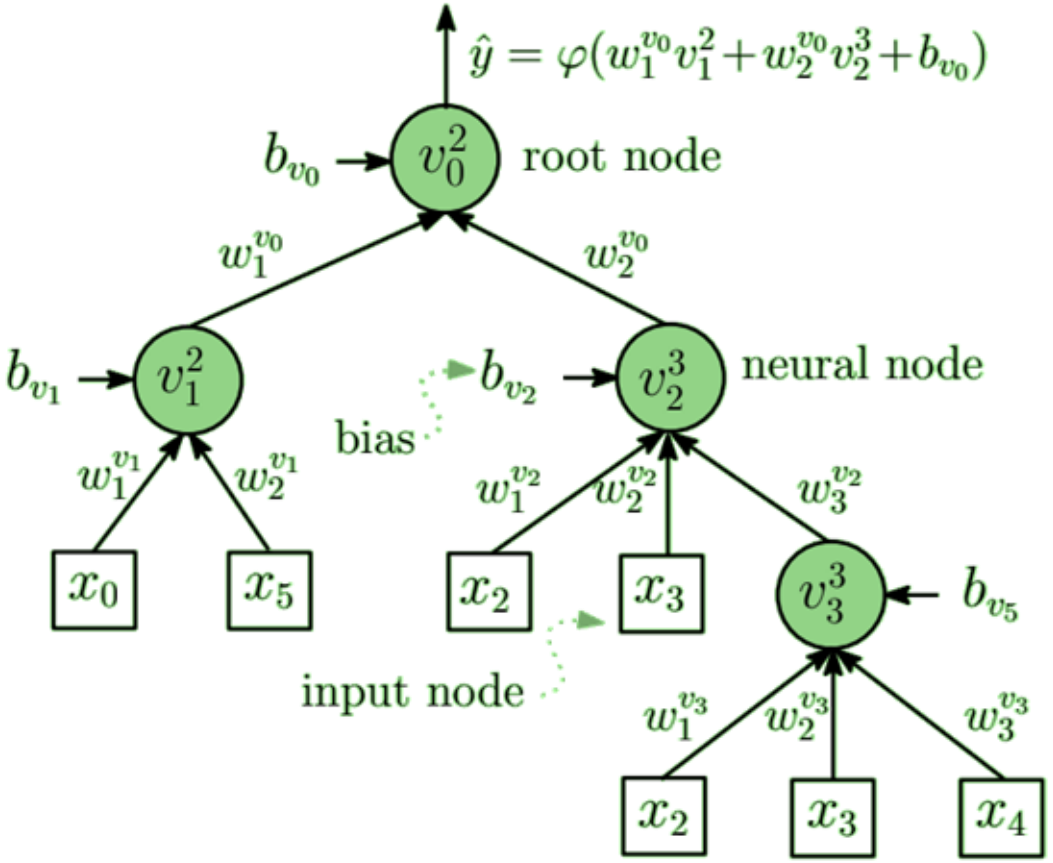
Neural Computation



Ojha et al (2022), Neural Networks

Types of Neural Tree

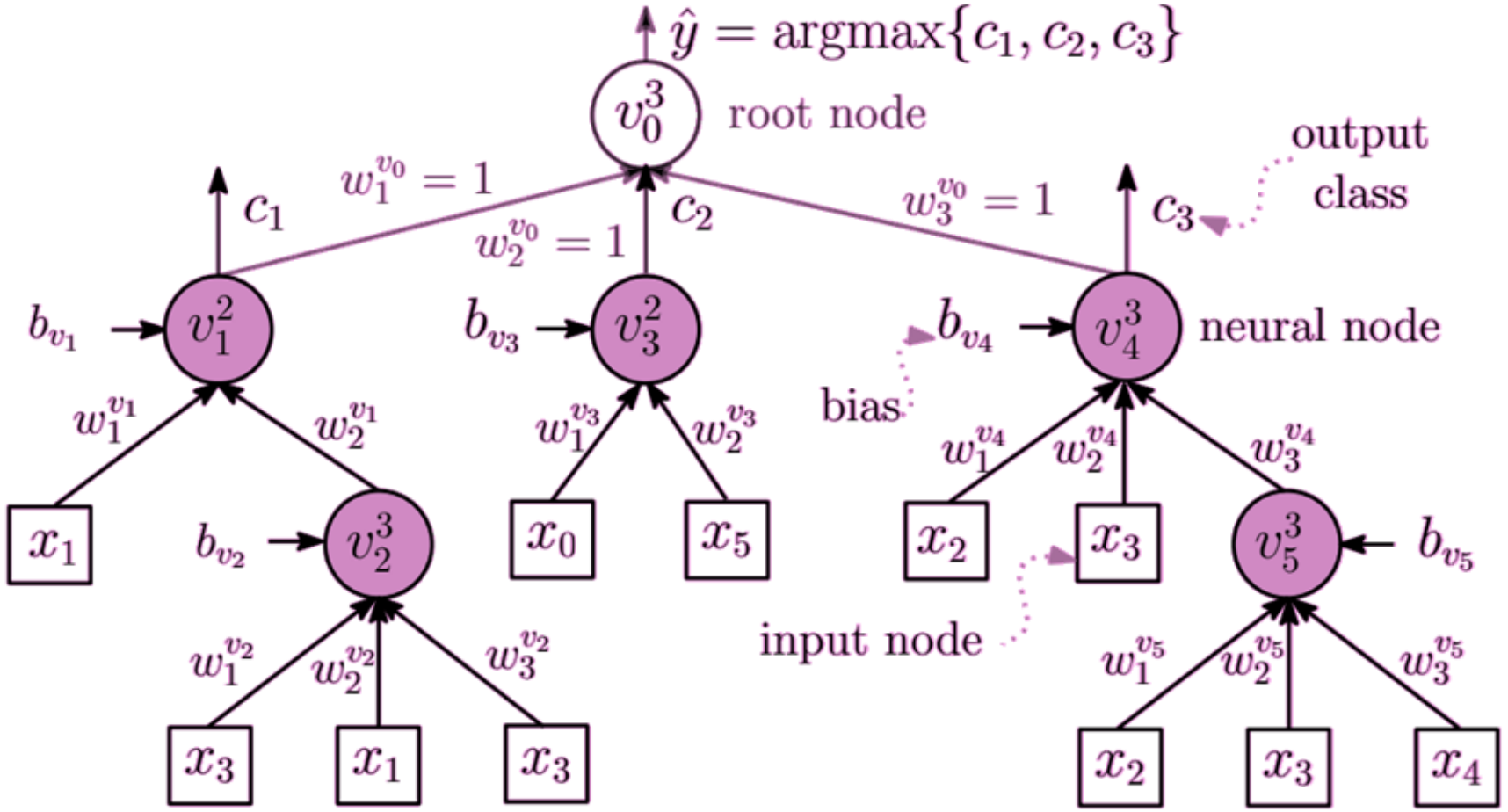
Regression Tree



Ojha et al (2022), Neural Networks

Types of Neural Tree

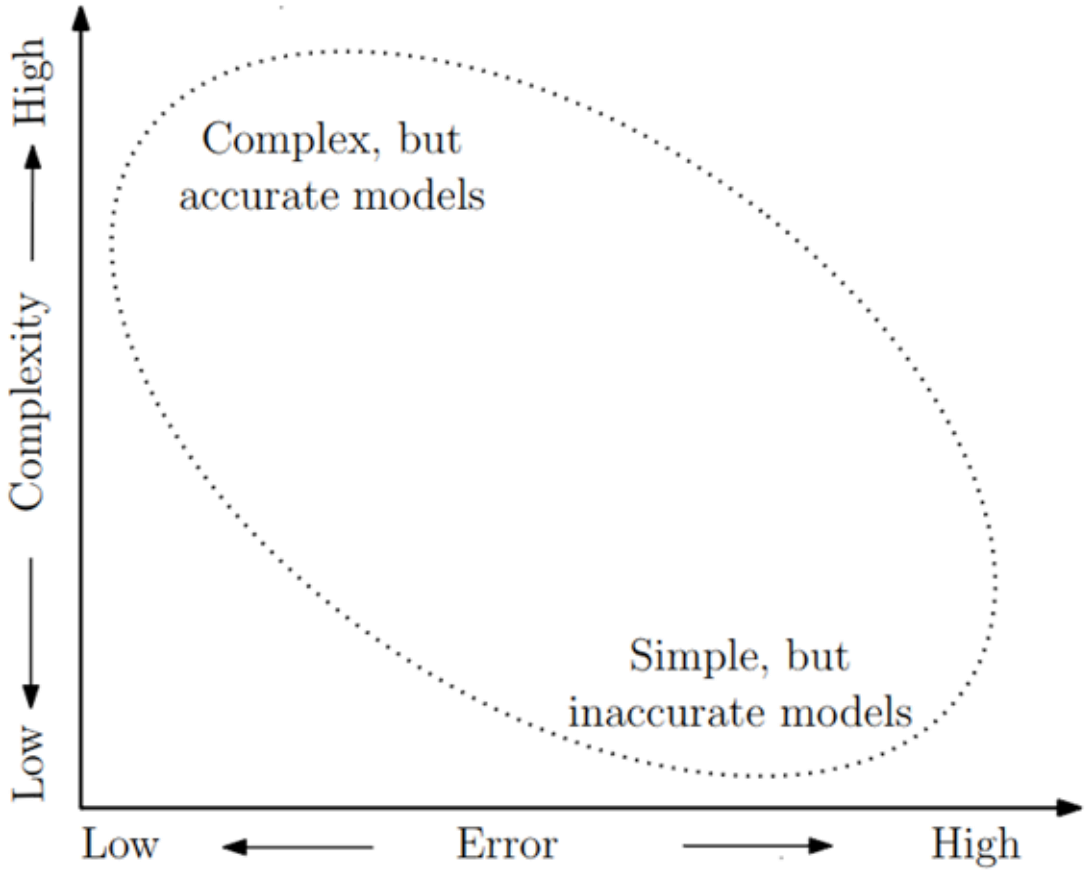
Classification Tree



Ojha et al (2020), CEC

Neural Architecture Search

Trade-offs



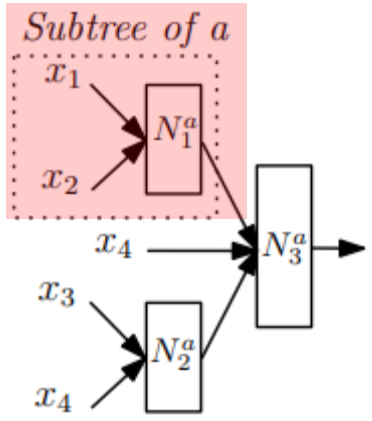
Ojha et al (2017), *Applied Soft Computing*

Neural Architecture Search

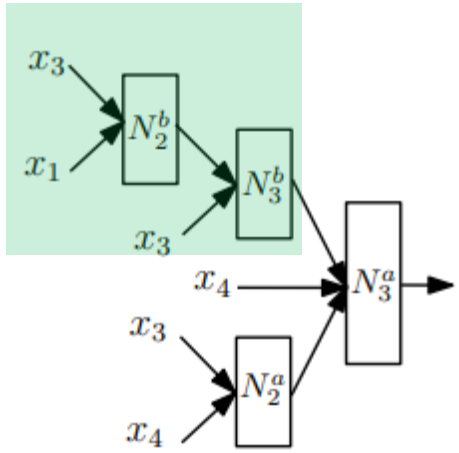
Trade-offs

Multiobjective
Genetic Programming
Crossover

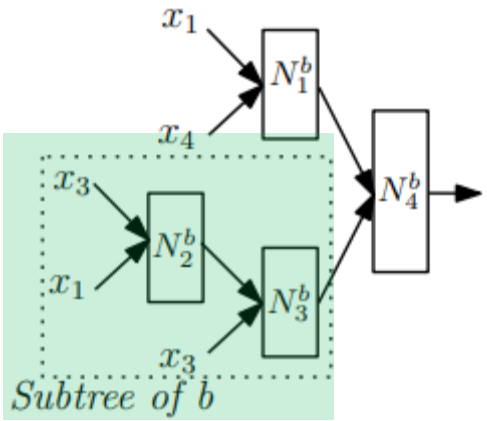
Ojha et al (2017), *IEEE Trans. Fuzzy Systems*



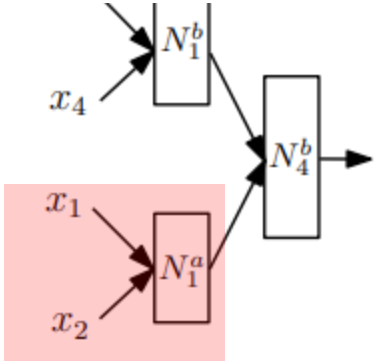
Parent tree: a



Child tree: c



Parent tree: b



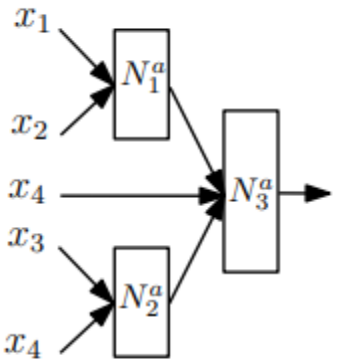
Child tree: d

Neural Architecture Search

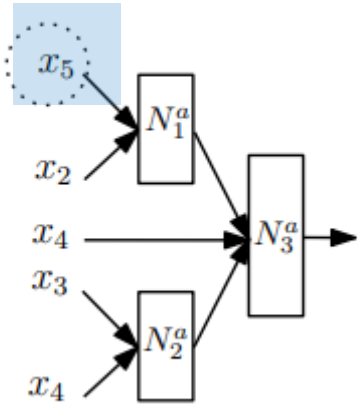
Trade-offs

Multiobjective
Genetic Programming
Mutation

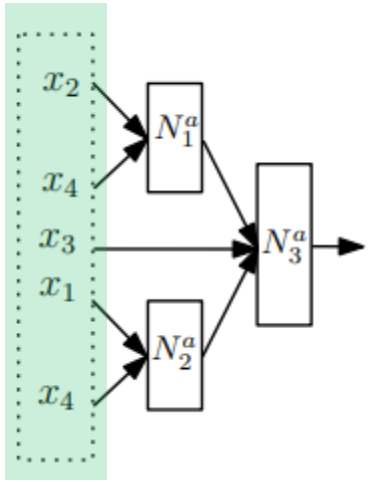
Ojha et al (2017), *IEEE Trans. Fuzzy Systems*



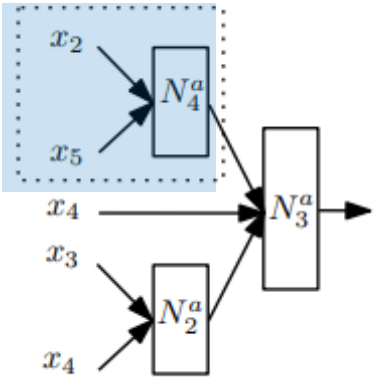
Parent tree



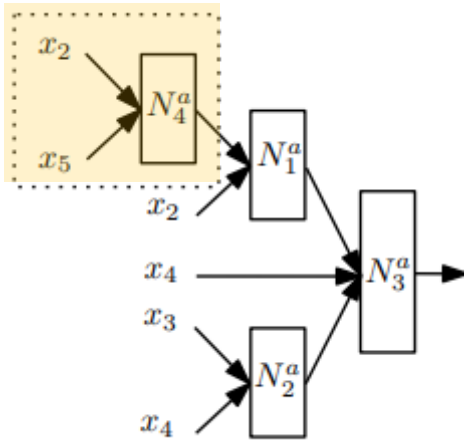
Single leaf mutation



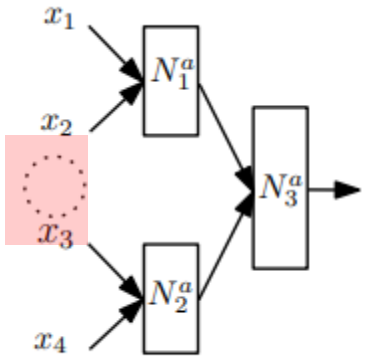
All leaves mutation



A subtree replacement



A subtree insertion

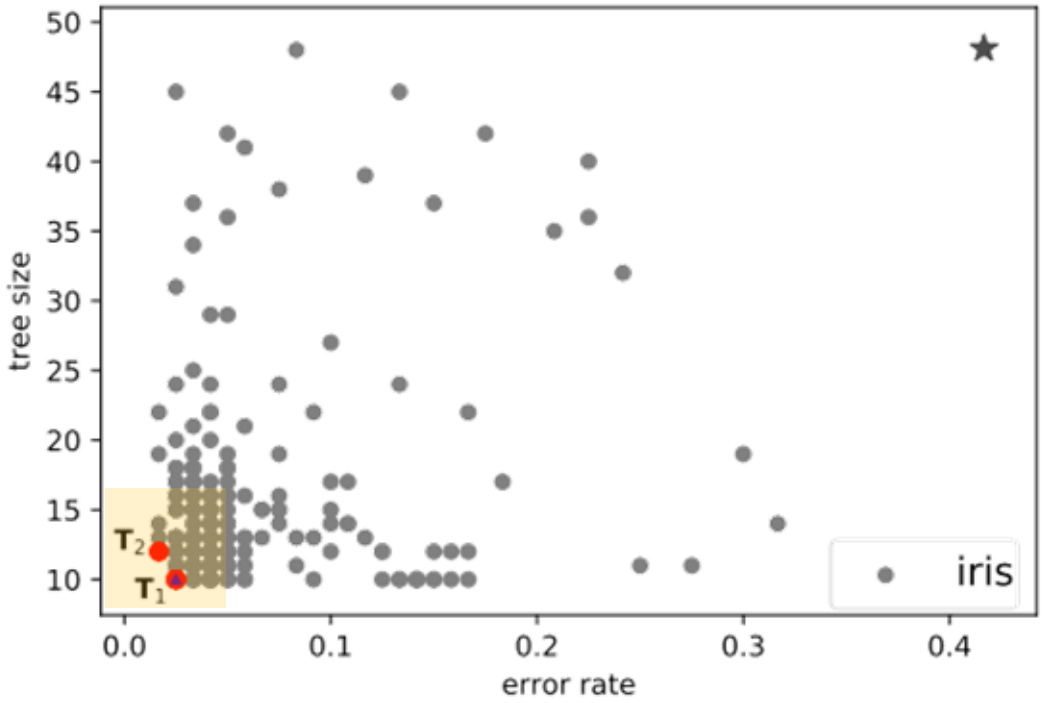


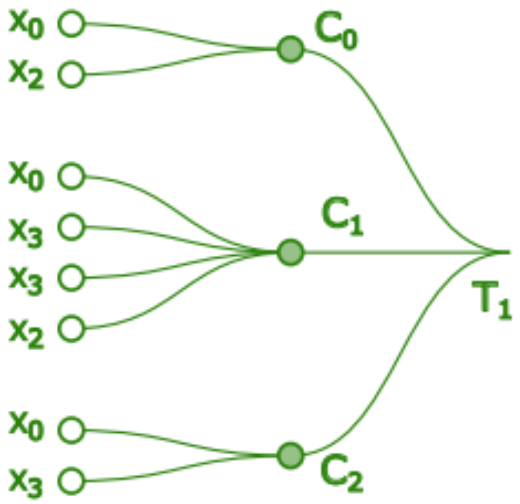
A subtree deletion

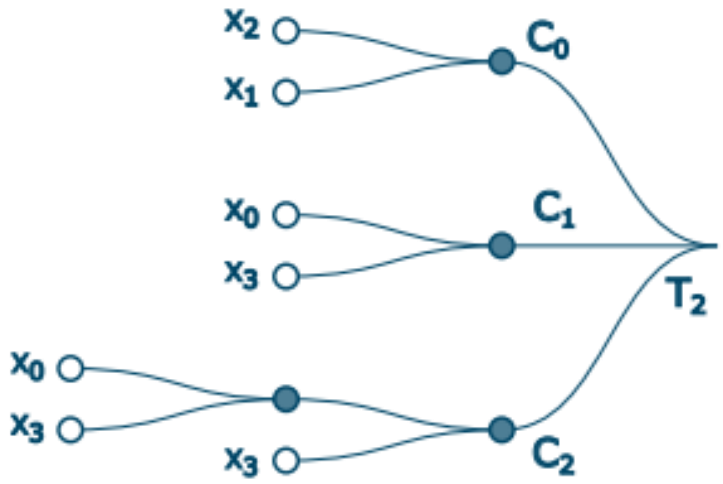
Architecture Search Trade-offs

Multiobjective Genetic Programming

Selection of trees using Hypervolume indicator from a Pareto Front



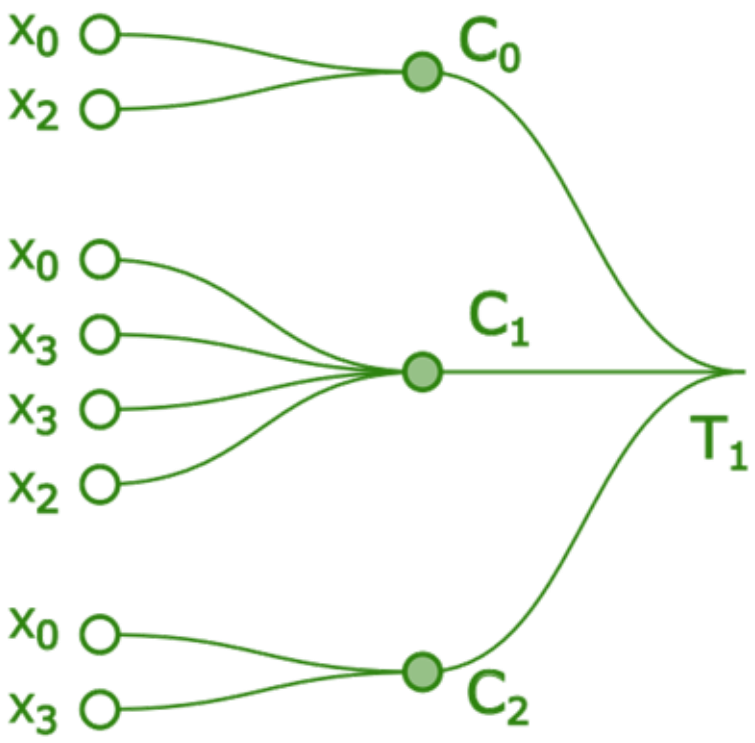


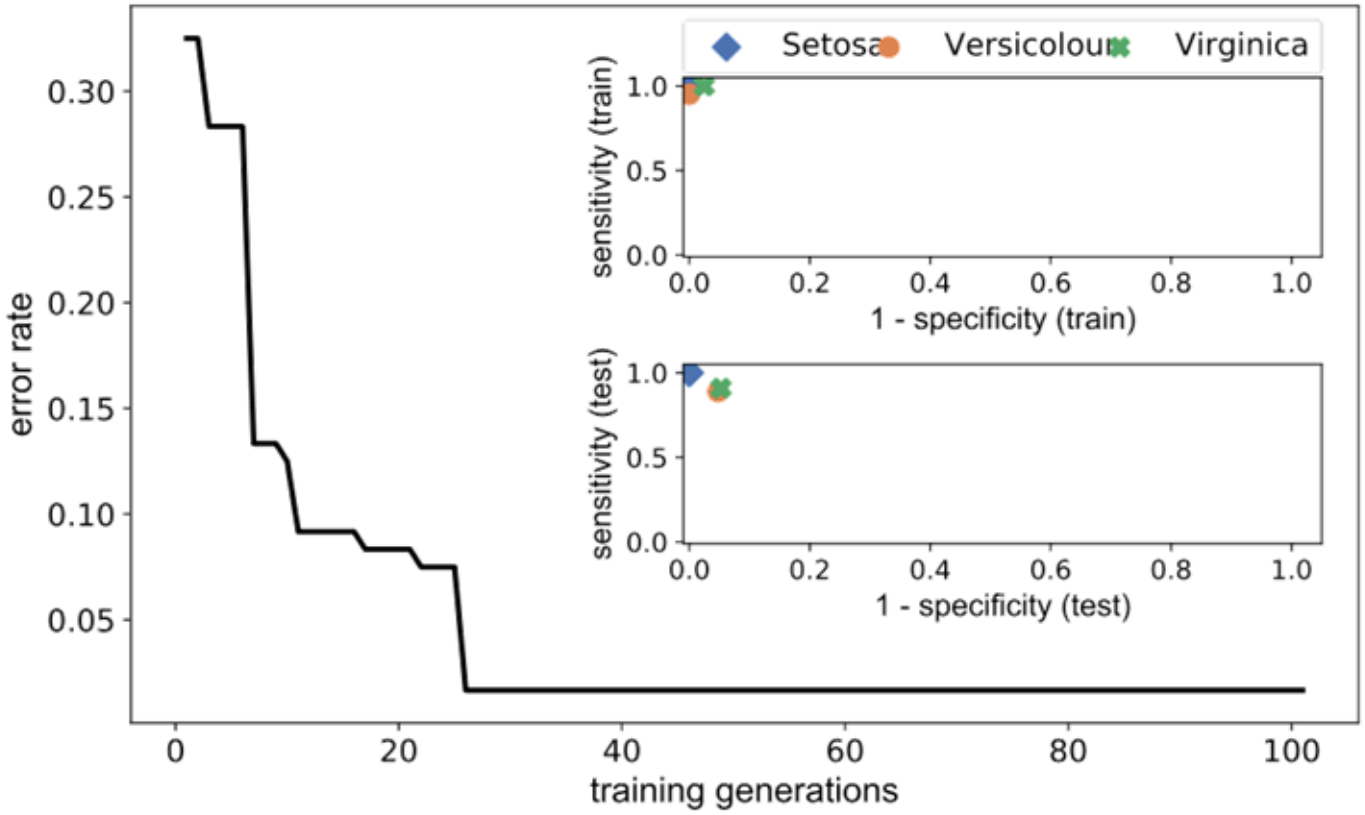


Ojha and Nicosia (2020), *CEC*

Learnability of Classes

Competition between classes: TPR/Recall/Sensitivity vs FPR/(1 - Specificity)





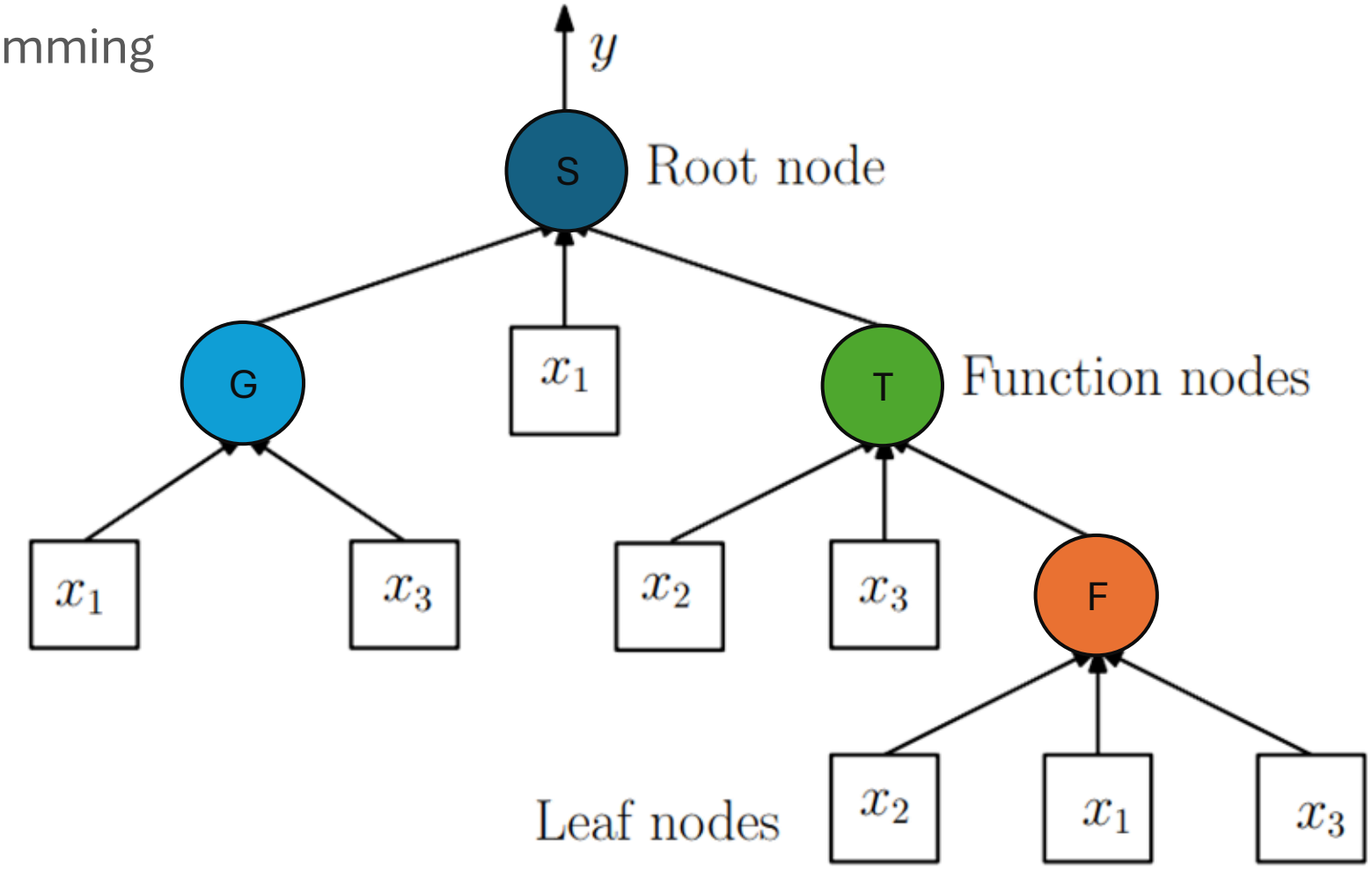
Ojha and Nicosia (2020), CEC

Heterogeneous Neural Tree

Multiobjective Genetic Programming

Activation Function Search

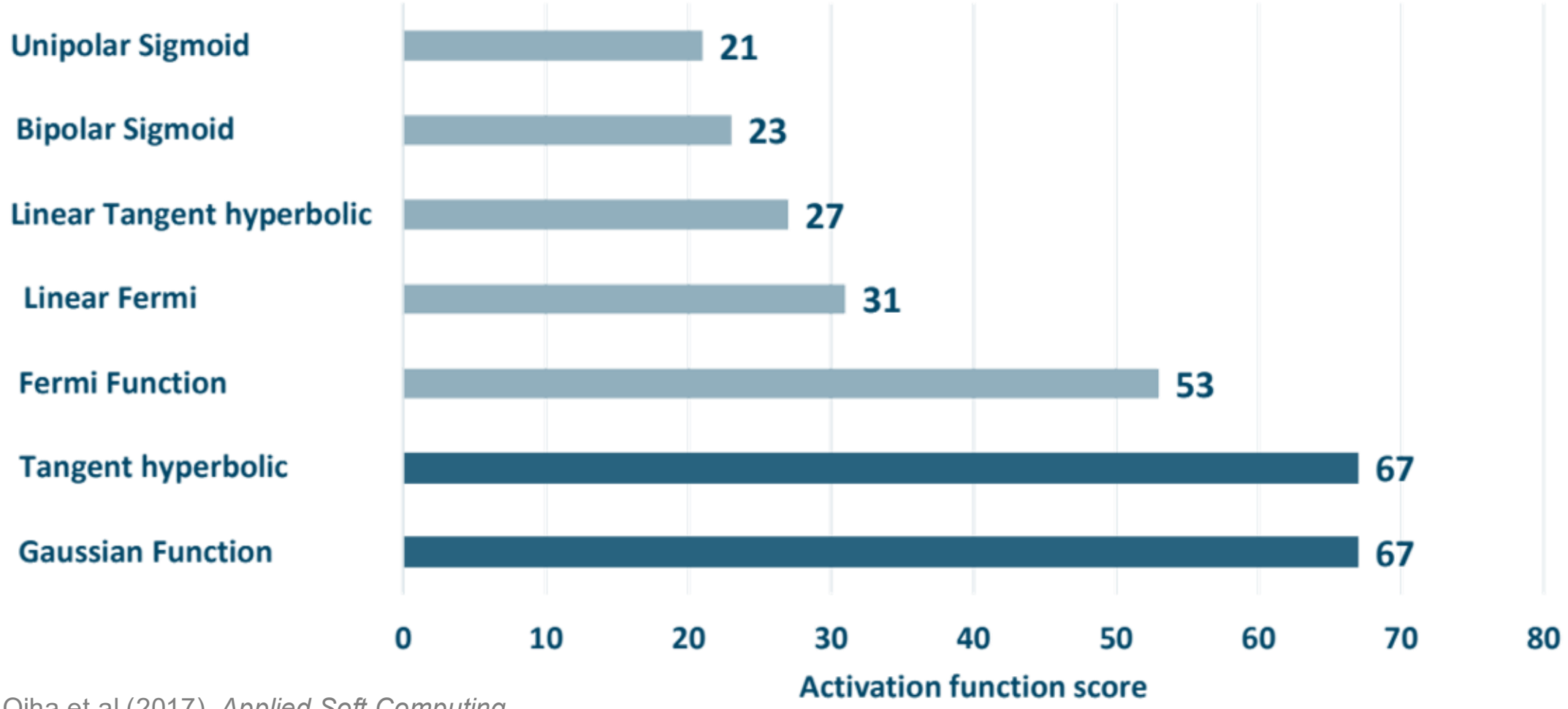
- S – Sigmoid
- G – Gaussian
- T – Tanh
- F – Fermi



Ojha et al (2017), *Applied Soft Computing*

Activation Function Performance

Higher values are better



Ojha et al (2017), *Applied Soft Computing*

Neural Modelling
Sparse Neural Tree

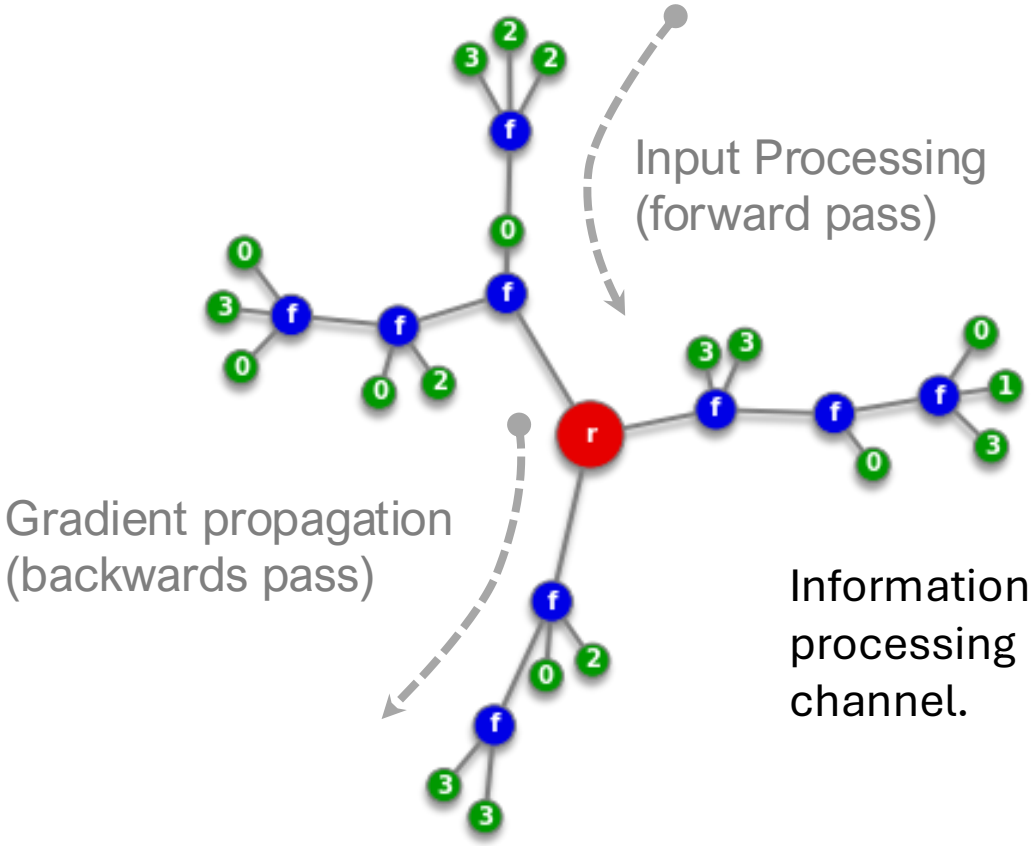
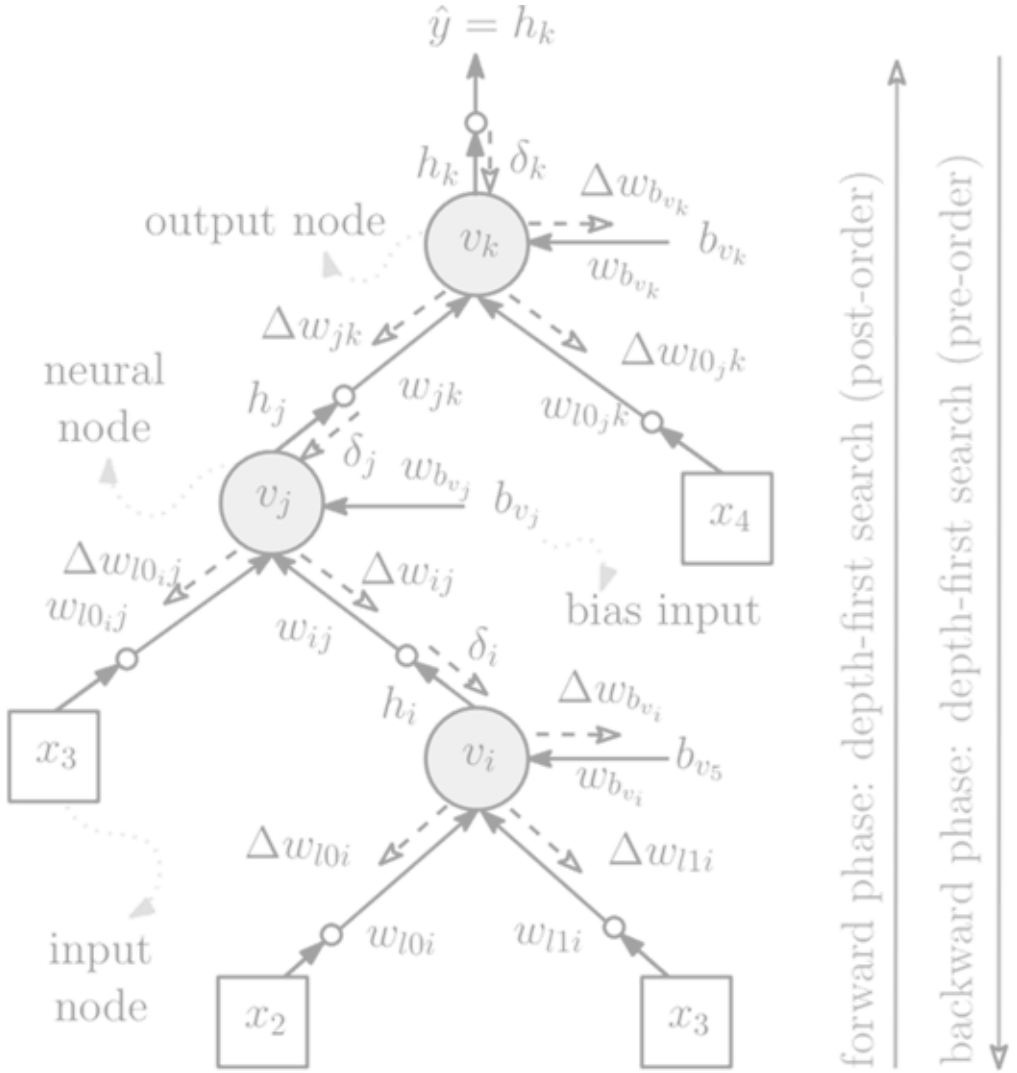
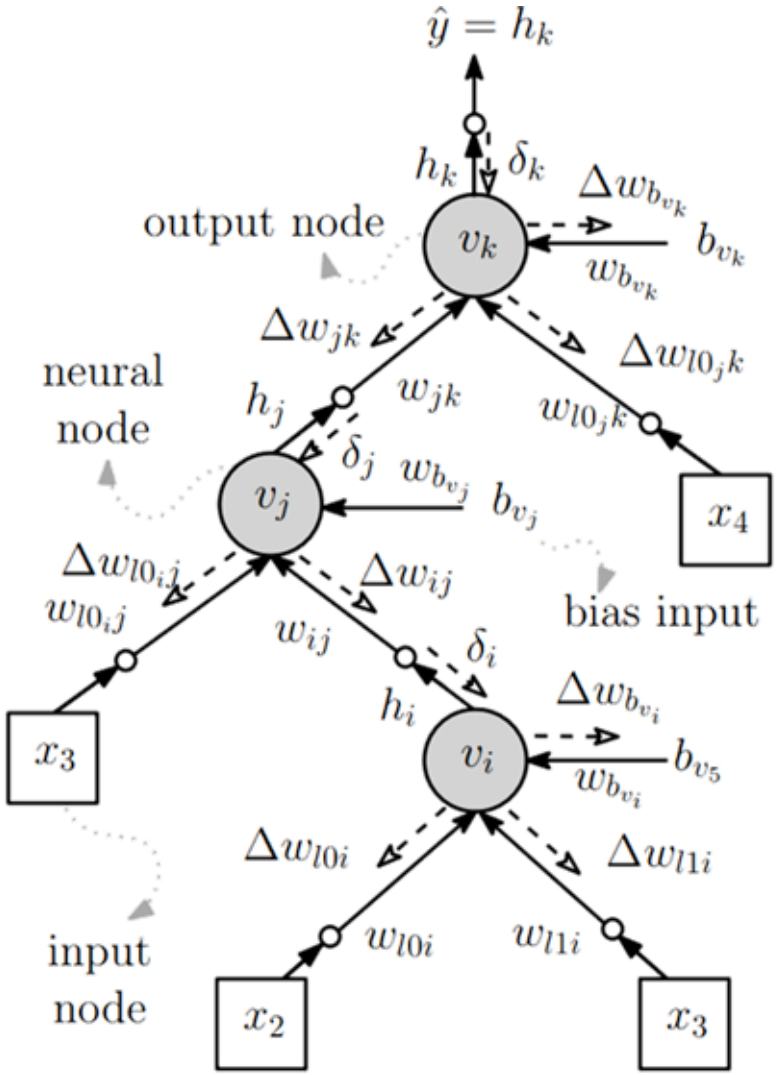


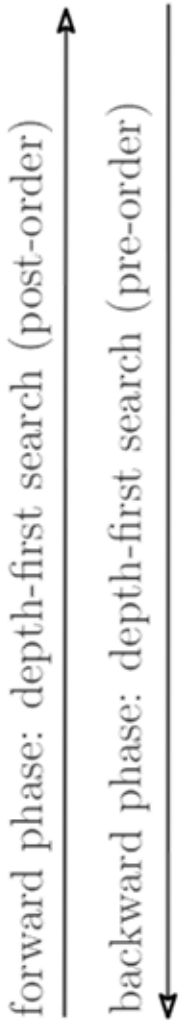
Fig A. Forward pass and gradient backpropagation

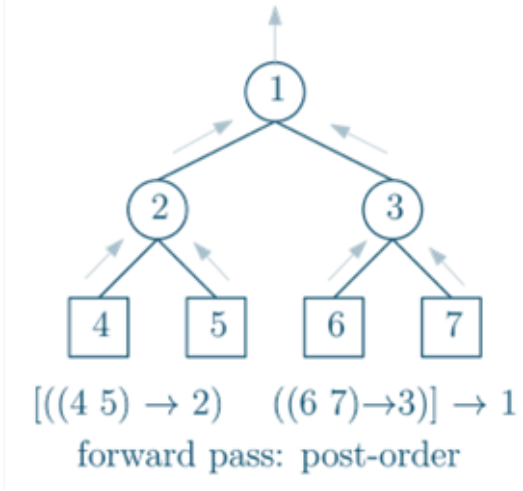


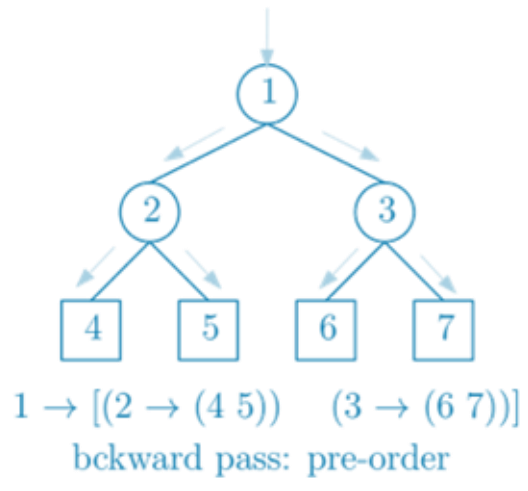
Ojha and Nicosia (2022), *Neural Networks*

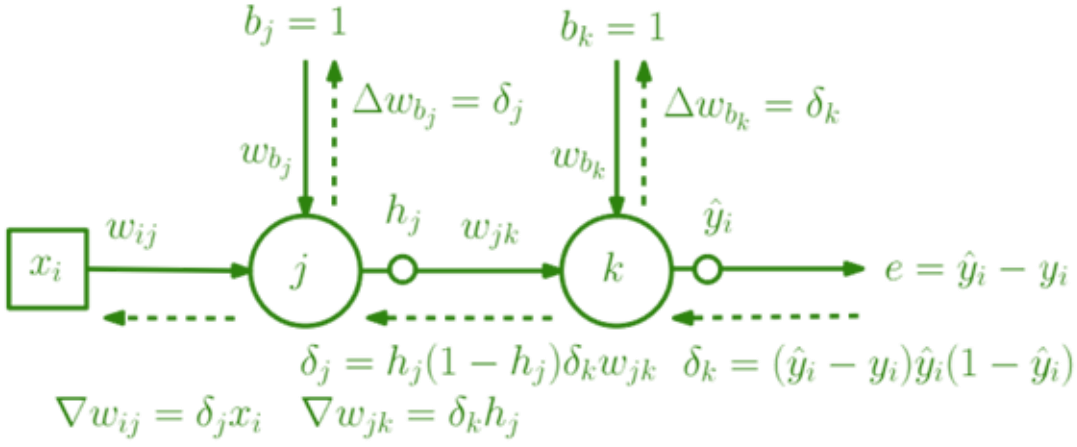
Backpropagation Neural Tree







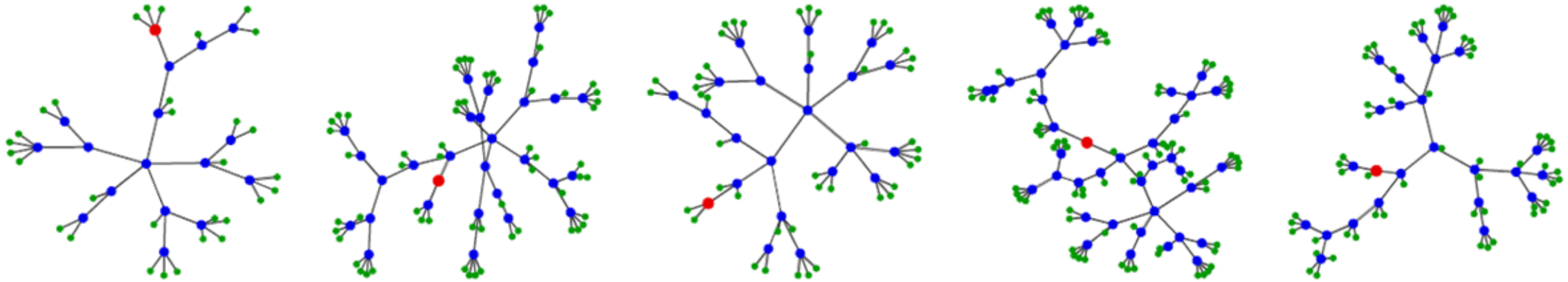




Ojha and Nicosia (2022), *Neural Networks*

Backpropagation Neural Tree

Regression results



(a) baseball (.85, 48) (b) dee (.89, 89) (c) diabetese (.63, 67) (d) friedman (.95, 116) (e) mpg6 (.9, 82)

Algorithm	Bas	Dee	Dia	Frd	Mpg	Avg Acc	Avg Weights
BNeuralT	0.665	0.837	0.492	0.776	0.867	**0.727**	**152**
MLP	0.721	0.829	0.49	0.943	0.874	**0.772**	**1041**

Ojha and Nicosia (2022), *Neural Networks*

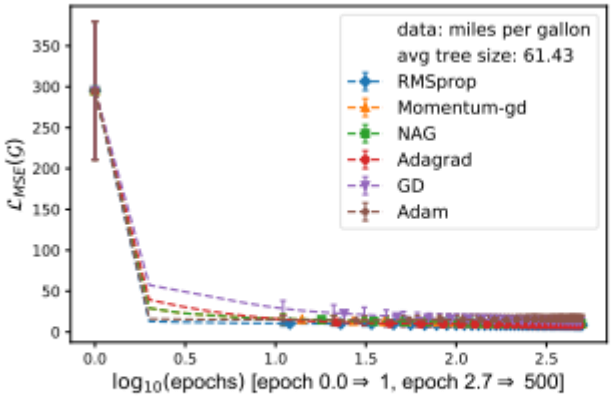
Backpropagation Neural Tree

Regression results

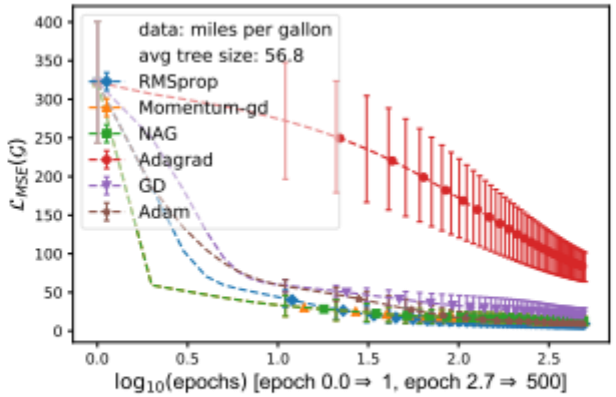
- BNeuralT used only 14.6% of MLP
- Accuracy differs only 5.8% lower than the best MLP result

Neural Tree vs Neural Networks

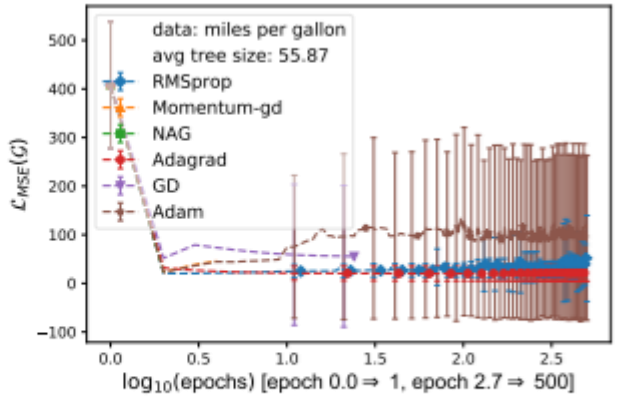
Regression Problems



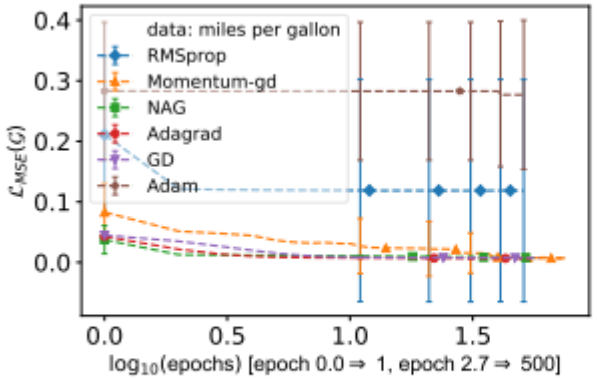
(g) BNeuralT: Sigmod, $\eta = 0.1$



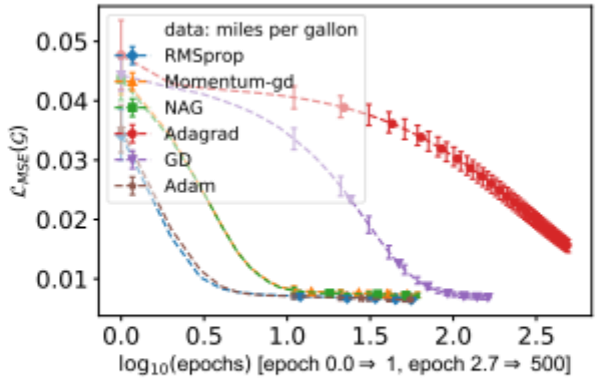
(h) BNeuralT: Sigmod, $\eta = default$



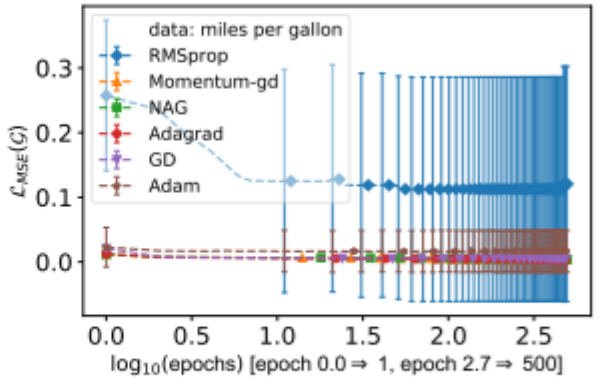
(i) BNeuralT: ReLU, $\eta = 0.1$



(j) MLP: Sigmod, $\eta = 0.1$



(k) MLP: Sigmod, $\eta = default$

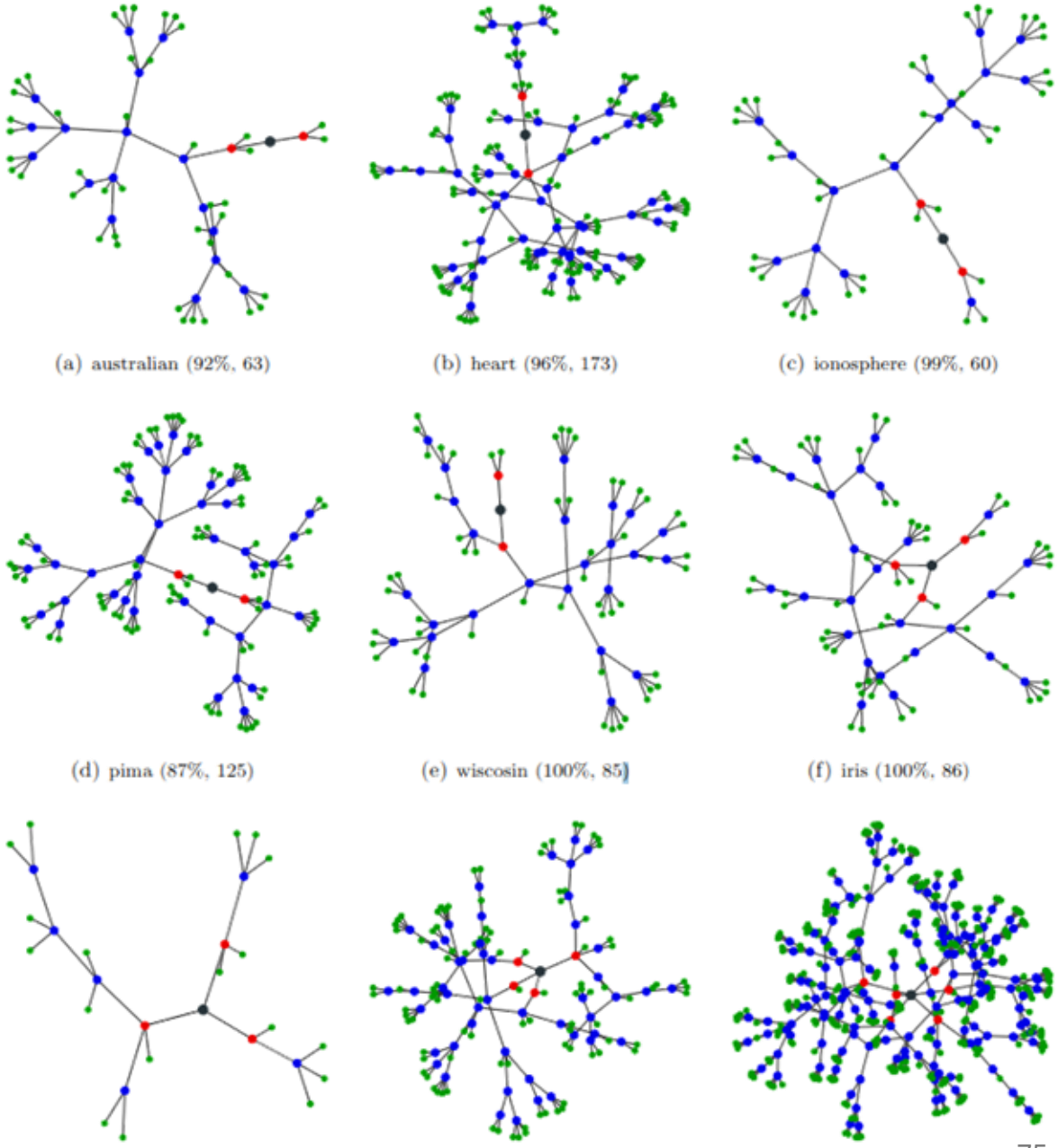


(l) MLP: ReLU, $\eta = 0.1$

Backpropagation Neural Tree

Classification results.

Data	BNeuralT	MLP
Aus	0.895	0.876
Hrt	0.897	0.833
Ion	0.952	0.882
Pma	0.822	0.774
Wis	0.986	0.984
Irs	0.992	0.972
Win	0.991	0.991
Vhl	0.75	0.826
Gls	0.732	0.635
Avg. Accuracy	**0.891**	**0.863**
Avg. Weights	**261**	1969



Ojha et al (2022), Neural Networks

Novel type of Neural Modelling
Sparse Neural Tree

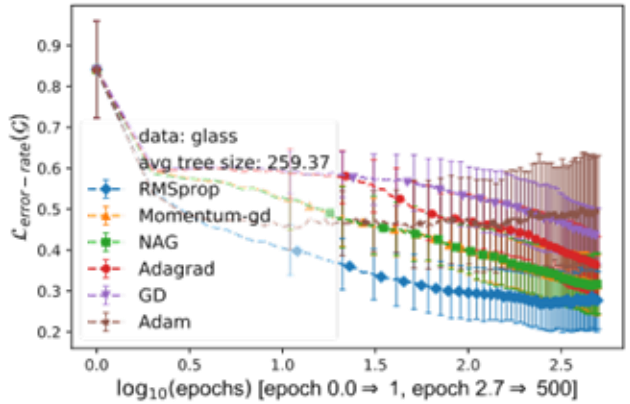
- Neural Tree used **only 13.25% parameters** of standard MLP

- Accuracy is **2.65% better than the best MLP** result

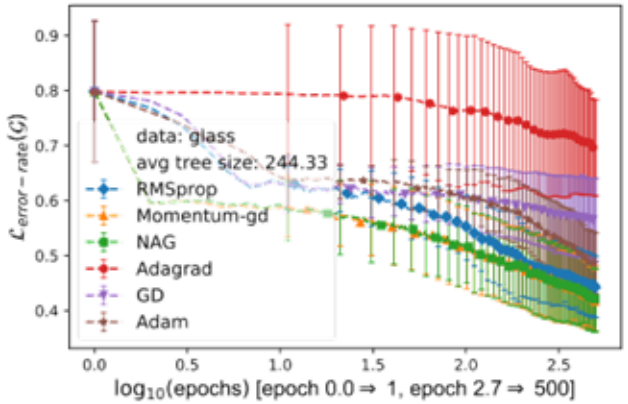
Ojha et al (2022), Neural Networks

Neural Tree vs Neural Networks

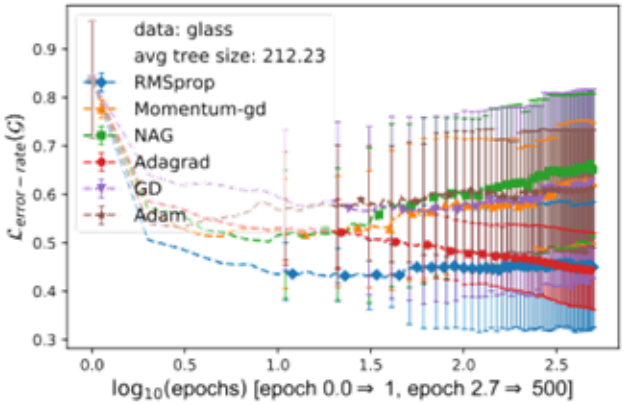
Classification Problems



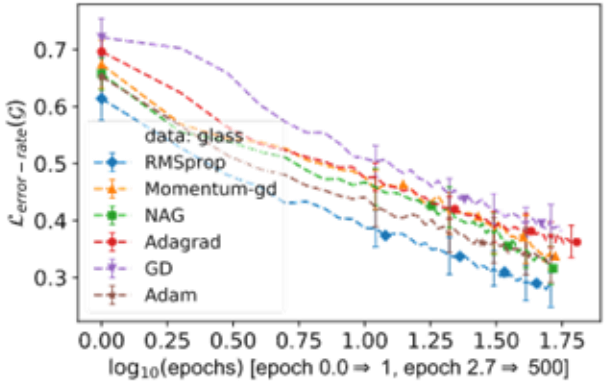
(a) BNeuralT: Sigmod, $\eta = 0.1$



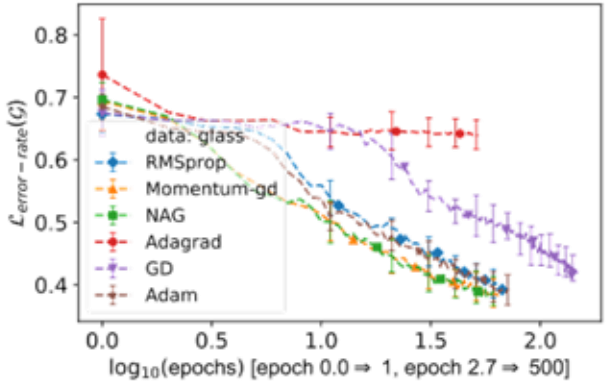
(b) BNeuralT: Sigmod, $\eta = default$



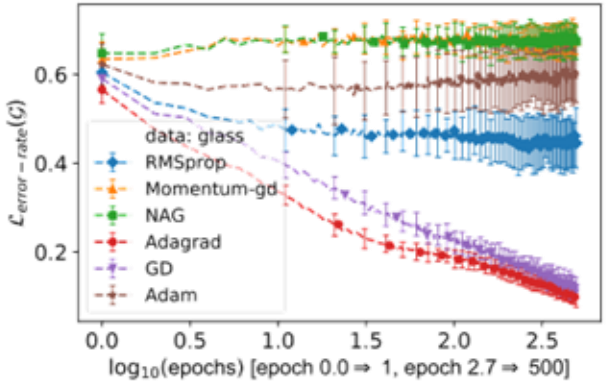
(c) BNeuralT: ReLU, $\eta = 0.1$



(d) MLP: Sigmod, $\eta = 0.1$

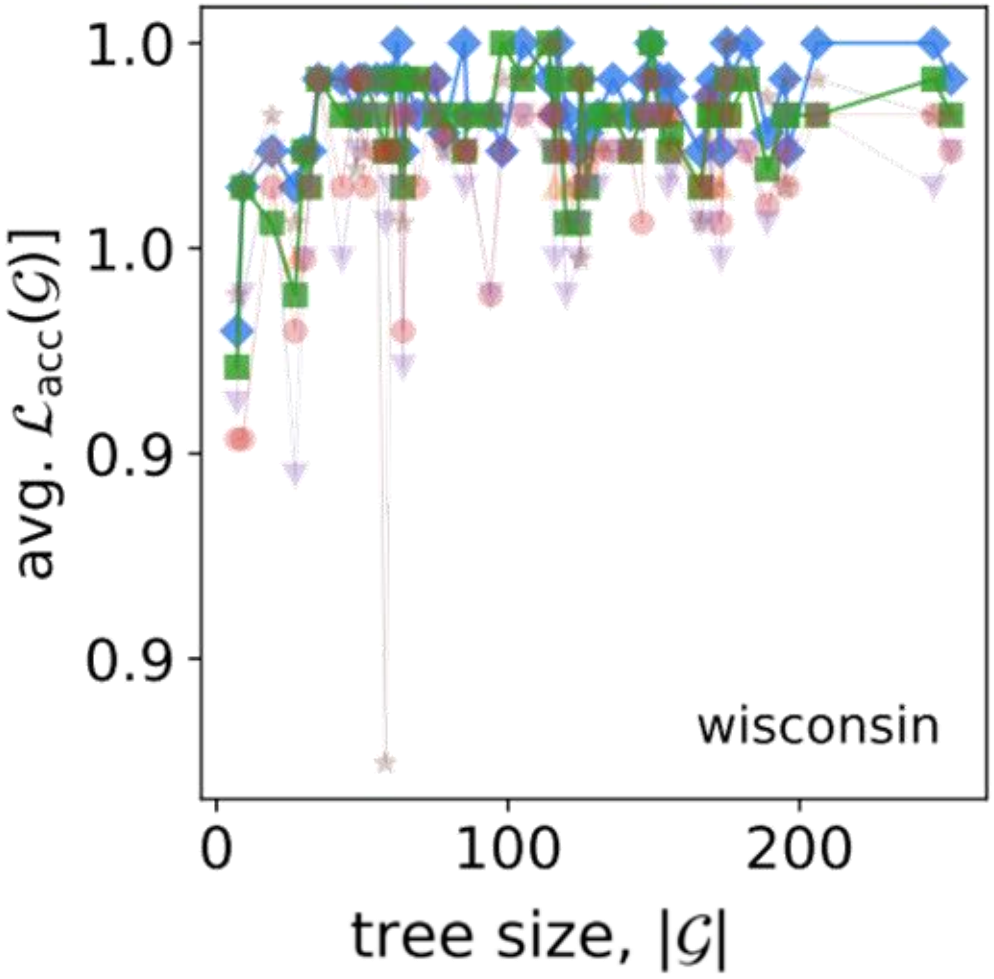


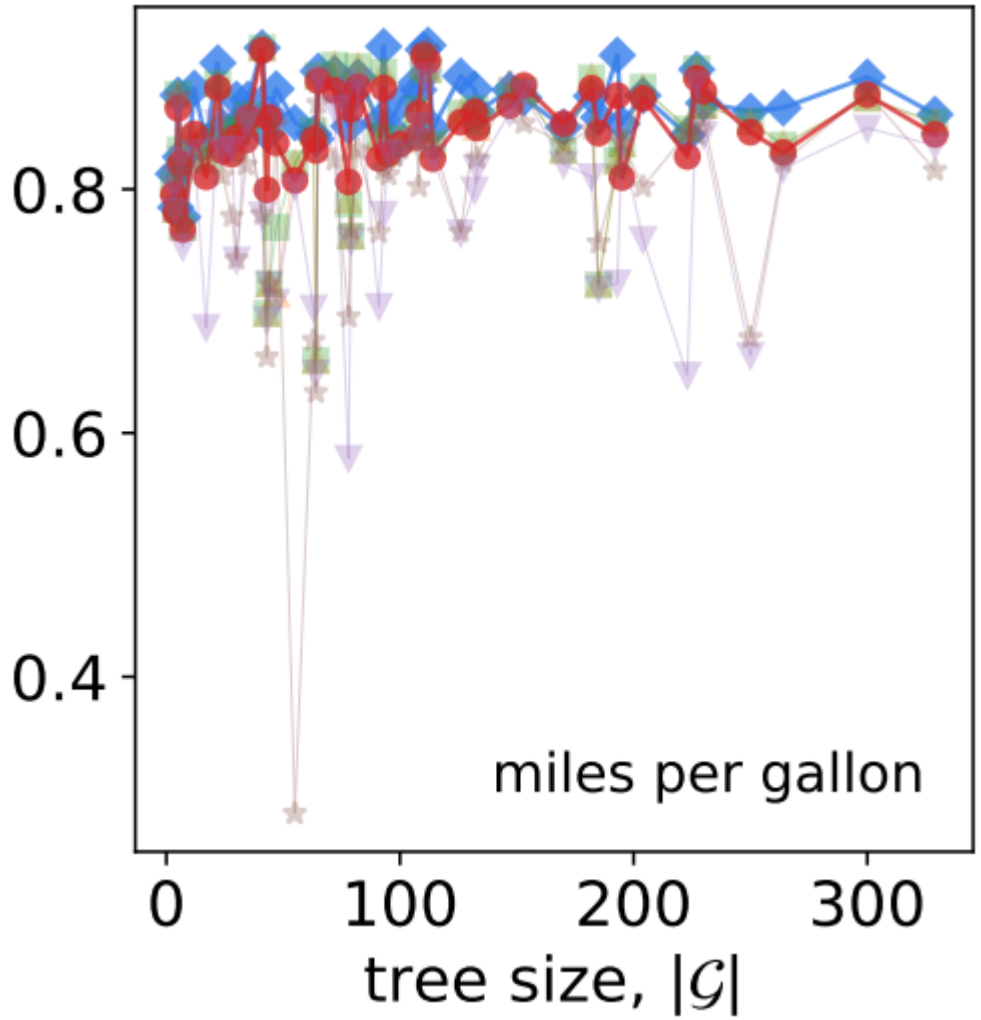
(e) MLP: Sigmod, $\eta = default$



(f) MLP: ReLU, $\eta = 0.1$

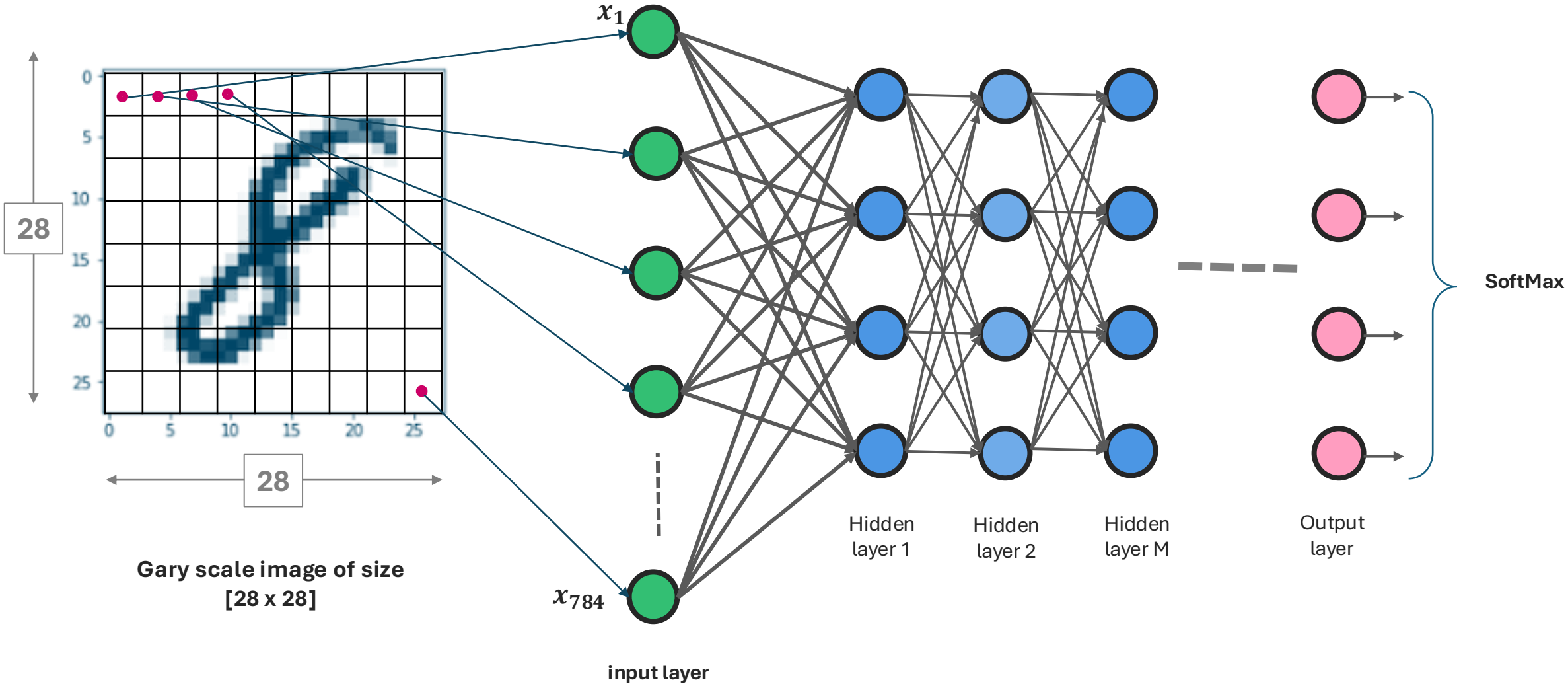
Architectural Stochasticity





Ojha et al (2022), Neural Networks

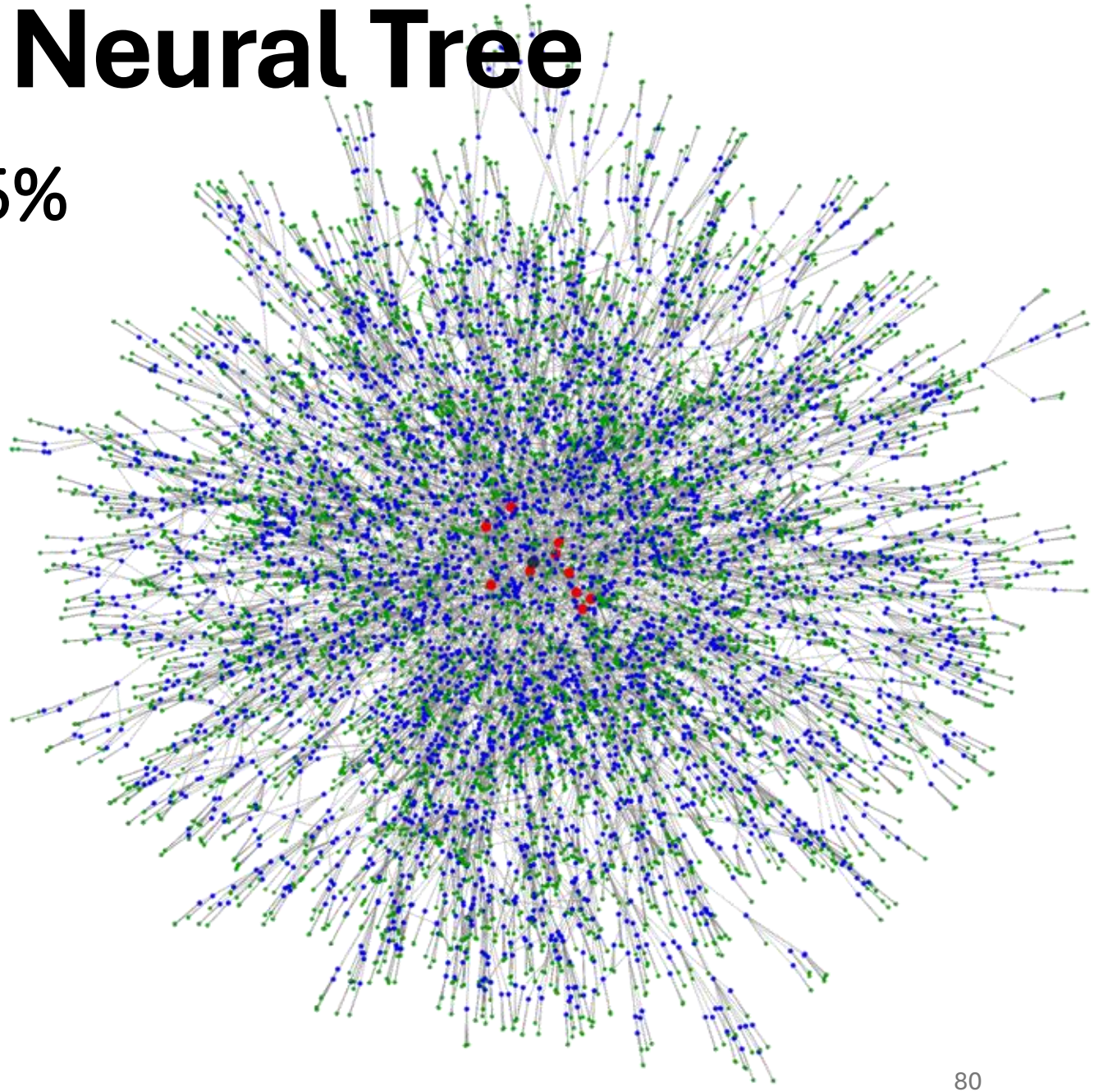
Deep Neural Networks



Backpropagation Neural Tree

MNIST Model Accuracy ~95%

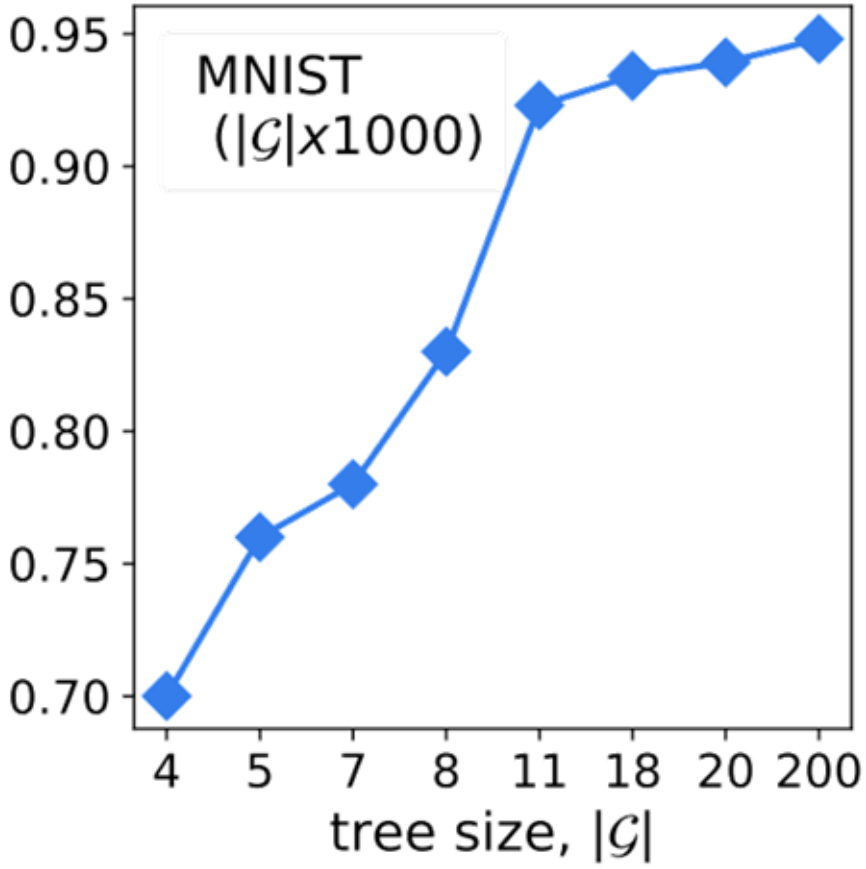
	Algorithms	Error(%)
BNeuralTs	BNeuralT-10K (pixels)	7.74
	BNeuralT-18K (pixels)	6.58
	BNeuralT-20K (pixels)	6.08
	BNeuralT-200K† (pixels)	**5.19**
Classification Trees	GUIDE (pixels, oblique split)	26.21
	OC1 (pixels, oblique split)	25.66
	GUIDE (pixels)	21.48
	CART-R (pixels)	11.97
	CART-P (pixels)	11.95
	C5.0 (pixels)	11.69
	TAO (pixels)	11.48
	TAO (pixels, oblique split)	5.26



Ojha and Nicosia (2022), *Neural Networks*

Model Size vs Accuracy

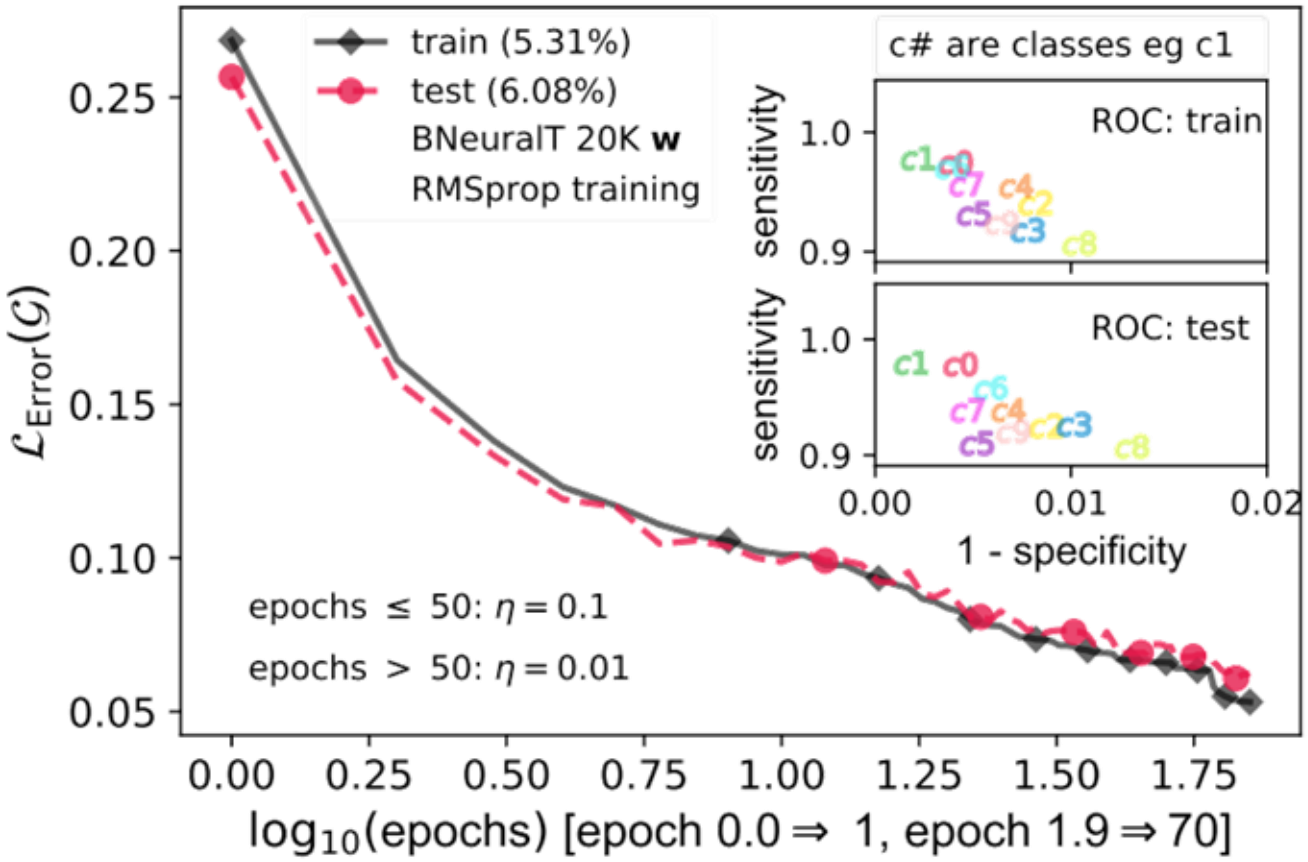
	Algorithms	Error(%)
BNeuralTs	BNeuralT-10K (pixels)	7.74
	BNeuralT-18K (pixels)	6.58
	BNeuralT-20K (pixels)	6.08
	BNeuralT-200K† (pixels)	**5.19**
Classification Trees	GUIDE (pixels, oblique split)	26.21
	OC1 (pixels, oblique split)	25.66
	GUIDE (pixels)	21.48
	CART-R (pixels)	11.97
	CART-P (pixels)	11.95
	C5.0 (pixels)	11.69
	TAO (pixels)	11.48
	TAO (pixels, oblique split)	5.26



Ojha and Nicosia (2022), *Neural Networks*

Learnability of Different Classes

Competition between classes: TPR/Recall/Sensitivity vs FPR/(1 - Specificity)

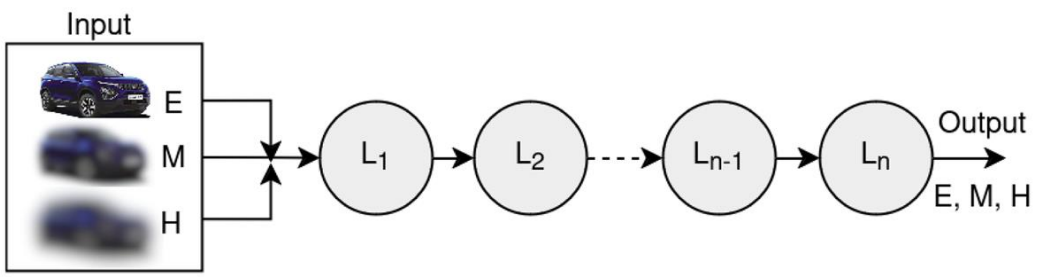


Ojha et al (2022), Neural Networks

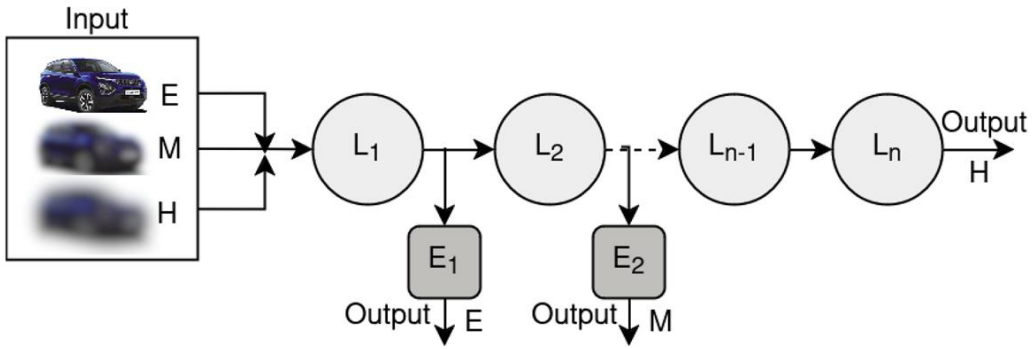
Part 5
Clever tricks to Make AI Models Resource Efficient

Early Exit Neural Nets

(E, M, and H represent samples with easy, medium, and hard complexity data inputs)

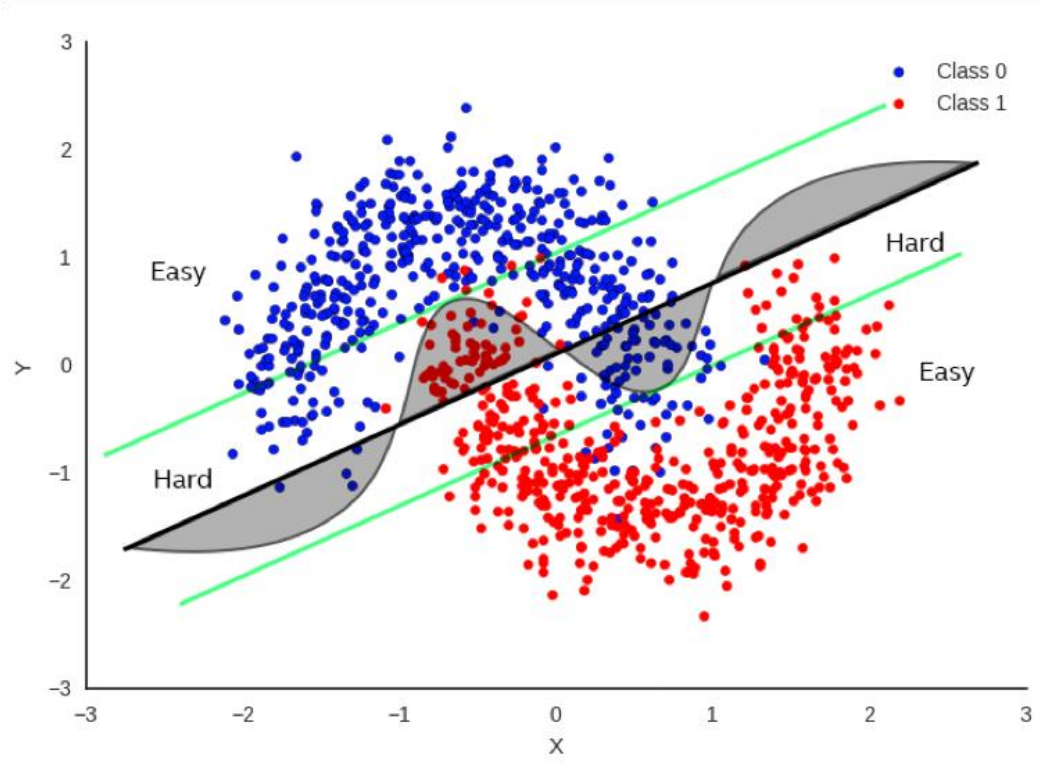


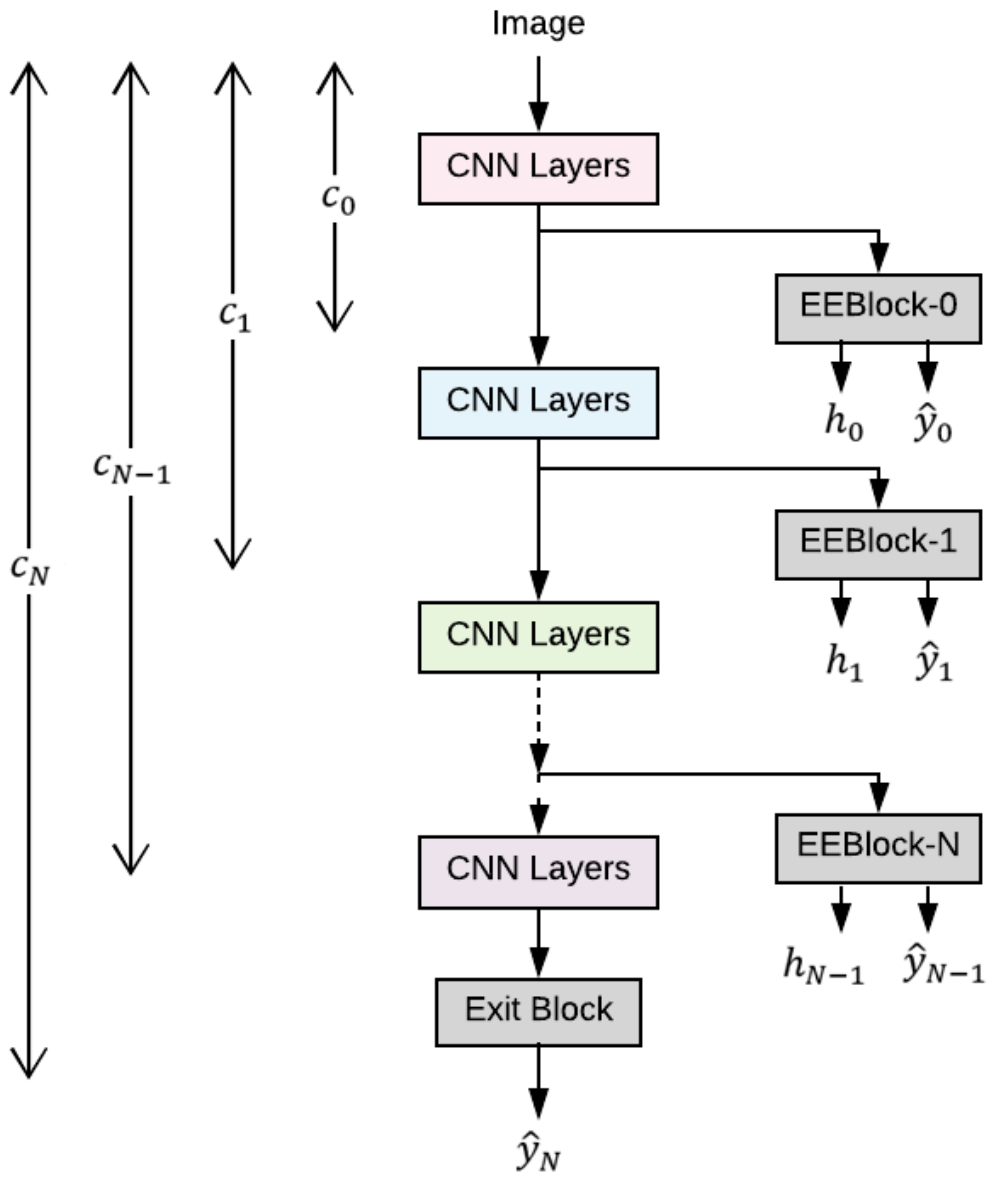
(a) Conventional DNN



(b) Early-exit DNN

Early Exit Neural Nets



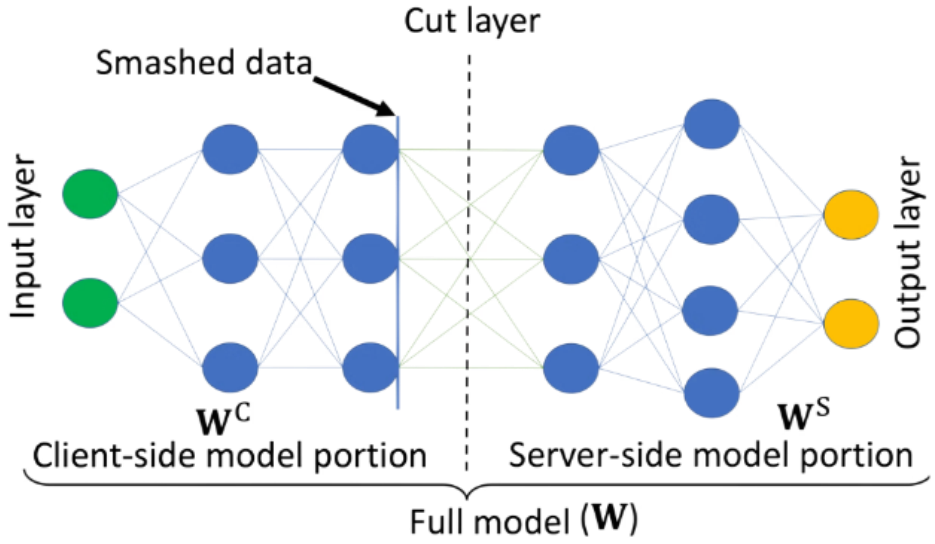


Source: https://intellabs.github.io/distiller/algo_earlyexit.html

Model Splitting

- Model splitting is a deployment technique used to divide a neural network into two or more parts, which are then executed on different devices or platforms.
- This is particularly useful when a model is too large or computationally intensive to run on a single edge or IoT device.

- There are several strategies for model splitting:
  - Layer-wise splitting, where different layers of the network are executed on different devices
  - Feature-wise splitting, where the model is split based on feature maps
- The choice of splitting strategy depends on the specific architecture of the model and the capabilities of the devices involved.
- Challenges associated with model splitting include:
  - Minimizing communication overhead between devices
  - Ensuring that the split model maintains acceptable accuracy and latency
- Techniques such as knowledge distillation and quantization can be used to mitigate these challenges.



References

- Backpropagation Neural Tree
  Neural Networks, Elsevier. (2022)
  Ojha V, Nicosia G
- Fragility, Robustness and Antifragility in Deep Learning
  Artificial Intelligence, Elsevier. (2024)
  Pravin C, Martino I, Nicosia G, Ojha V
- Dynamic Label Adversarial Training for Deep Learning Robustness Against Adversarial Attacks
  31st International Conference on Neural Information Processing (ICONIP). (2024)
  Liu Z, Duan H, Liang H, Long Y, Snasel V, Nicosia G, Ranjan R and Ojha V
- On Learnable Parameters of Optimal and Suboptimal Deep Learning Models
  31st International Conference on Neural Information Processing (ICONIP). (2024)
  Zheng Z, Liang H, Snasel V, Latora V, Pardalos P, Nicosia G, and Ojha V
- Sensitivity analysis for deep learning: Ranking hyper-parameter influence
  33rd IEEE Int. Conf. on Tools with Artificial Intelligence, ICTAI. IEEE (2021)
  Taylor R, Martino I, Nicosia G, Ojha V
- Multi-objective optimization of multi-output neural tree
  IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK (pp 1–8). IEEE. (2020)
  Ojha V, Nicosia G
- Multiobjective programming for type-2 hierarchical fuzzy trees (Code)
  IEEE Transaction on Fuzzy Systems, 26(2), 915–936. (2018)
  Ojha VK, Snášel V, Abraham A
- Metaheuristic design of feedforward neural networks: a review of two decades of research
  Engineering Applications in Artificial Intelligence, 60, 97-116. Elsevier (2017)
  Ojha VK, Snášel V, Abraham A
- Ensemble of heterogeneous flexible neural trees using multiobjective genetic programming
  Applied Soft Computing, 52, 909-924. Elsevier (2017)
  Ojha VK, Abraham A, Snášel V

Resource Efficient Artificial Intelligence

Varun Ojha

Senior Lecturer in Artificial Intelligence
AI Theme Lead National Edge AI Hub
School of Computing, Newcastle University
varun.ojha@newcastle.ac.uk
https://ojhavk.github.io/