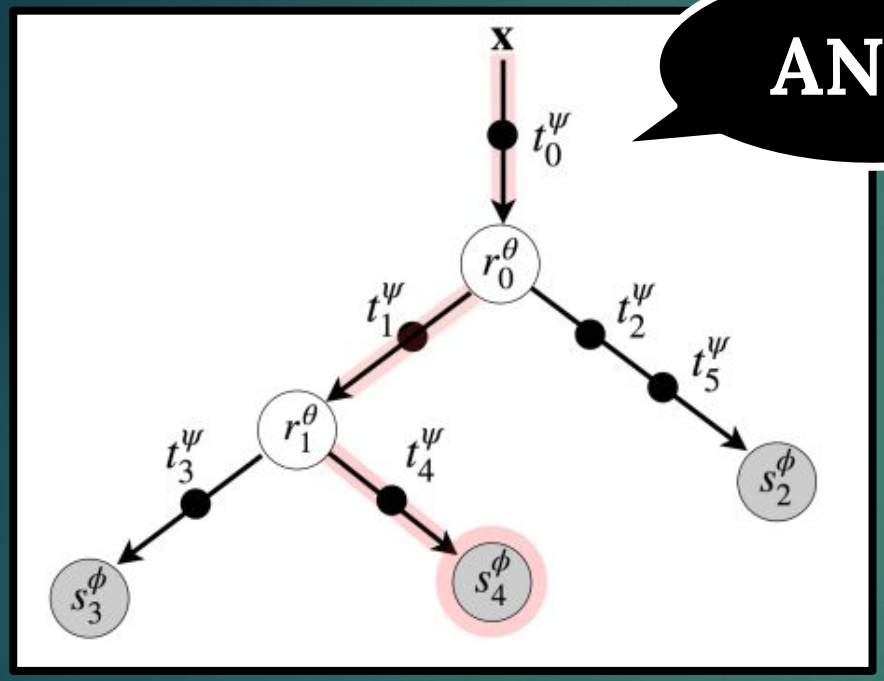
ADAPTIVE NEURAL TREES



"Adaptive Neural Trees", by R. Tanno et al. ,
from arxiv.org/pdf/1807.06699.pdf

2019 ICLR
International Conference on
Learning Representations

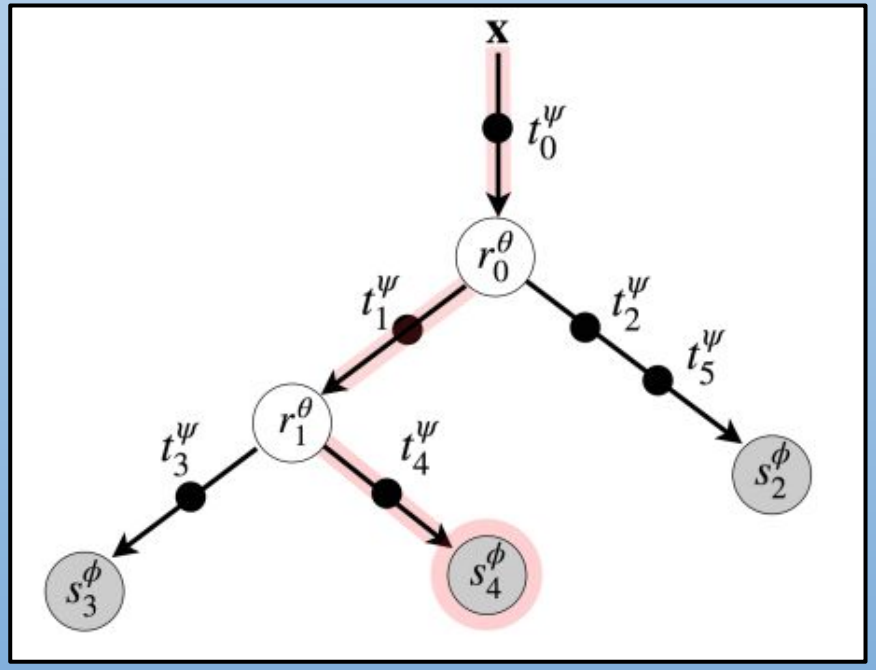
by R. Tanno et al.

Presented by

61605041 Lada Phonrungwong

ADAPTIVE NEURAL TREES

2019 ICLR
International Conference on Learning Representations



"Adaptive Neural Trees", by R. Tanno et al. ,
from arxiv.org/pdf/1807.06699.pdf

by
Ryutaro Tanno
Kai Arulkumaran
Daniel C. Alexander
Antonio Criminisi
Aditya Nori

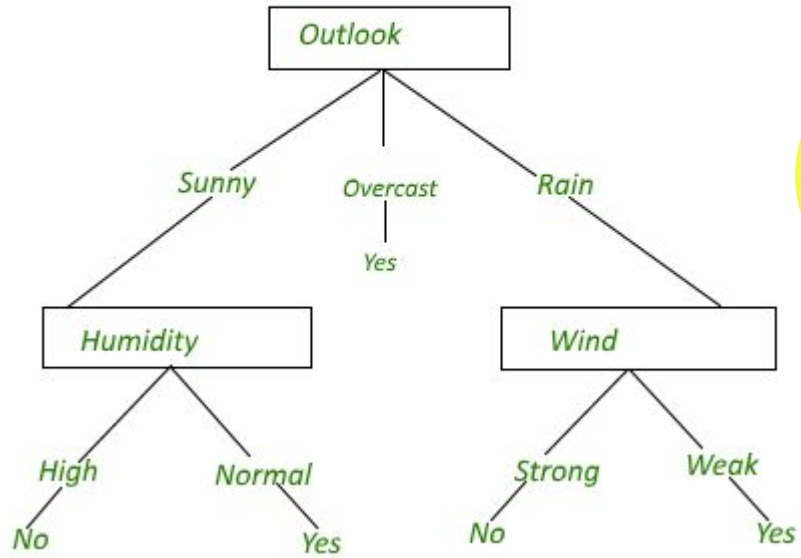
OUTLINE

1. ANTs?
2. Model Topology
3. Model Operations
4. Loss Function
5. Growth Phase
6. Refinement Phase
7. Algorithm
8. Compare
9. Training Time
10. Discovered Architectures

1.
ANTs?

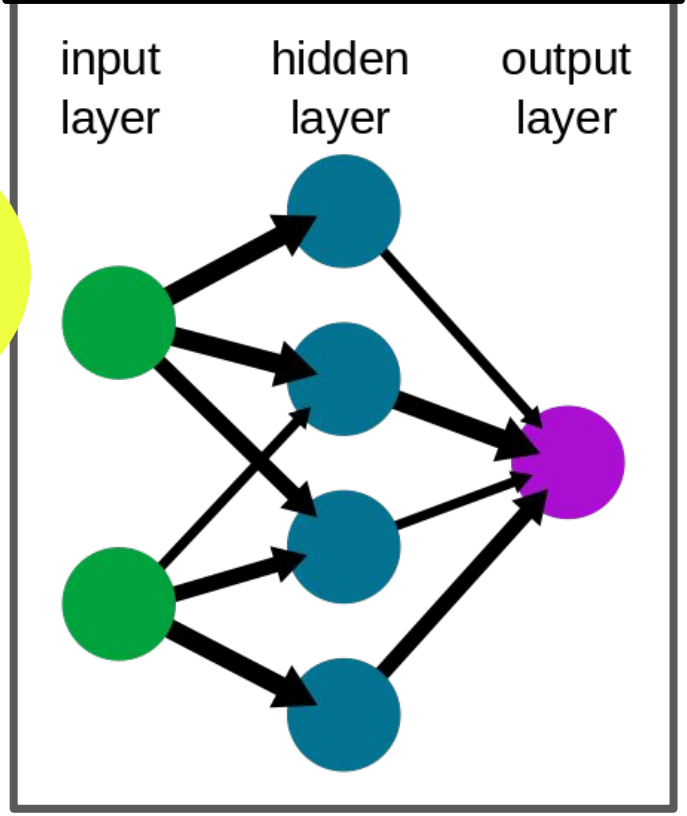
ADAPTIVE NEURAL TREES

DECISION TREE



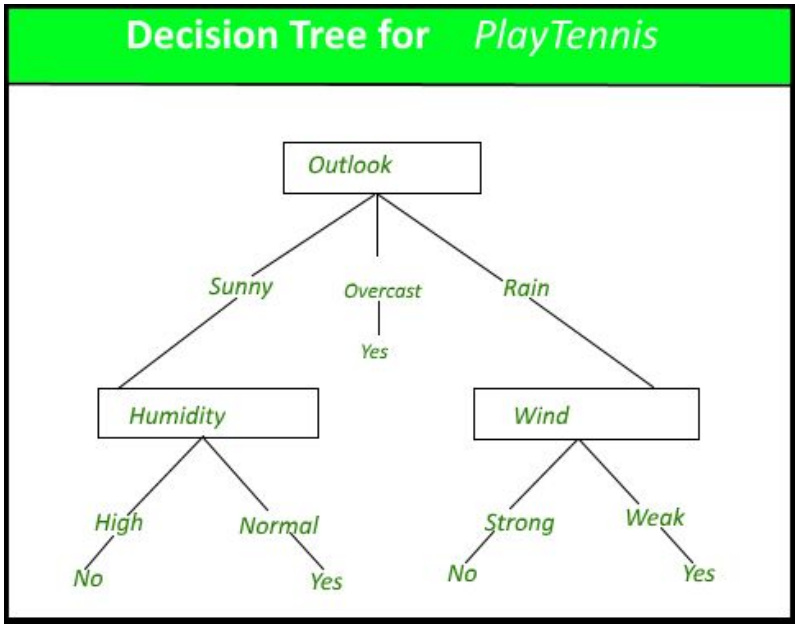
source : www.geeksforgeeks.org/decision-tree

NEURAL NETWORK



source : en.m.wikipedia.org/wiki/Neural_network

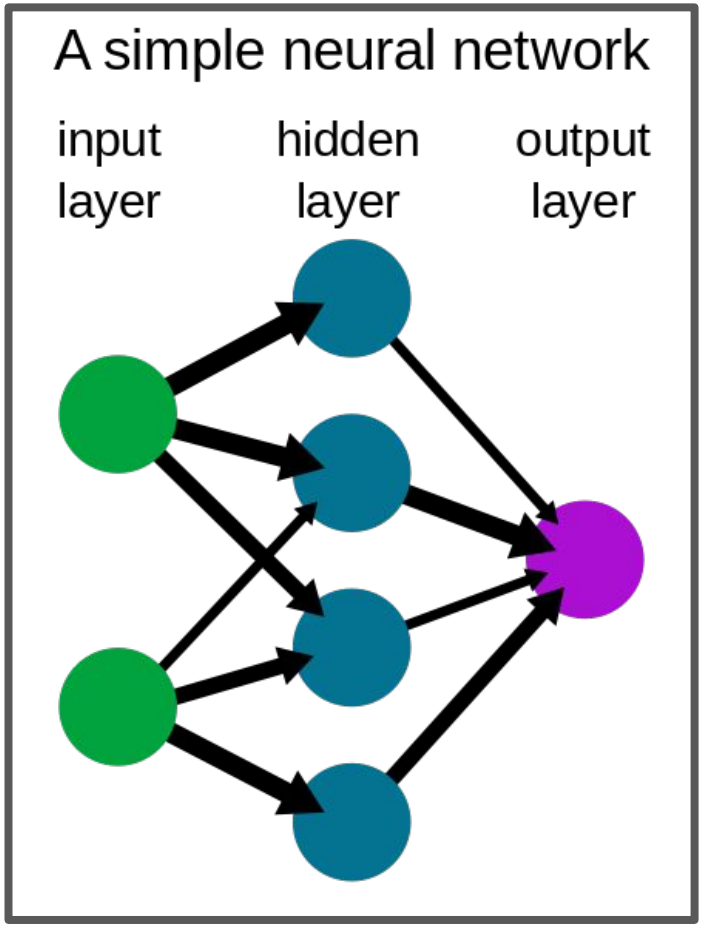
DECISION TREE



source : www.geeksforgeeks.org/decision-tree

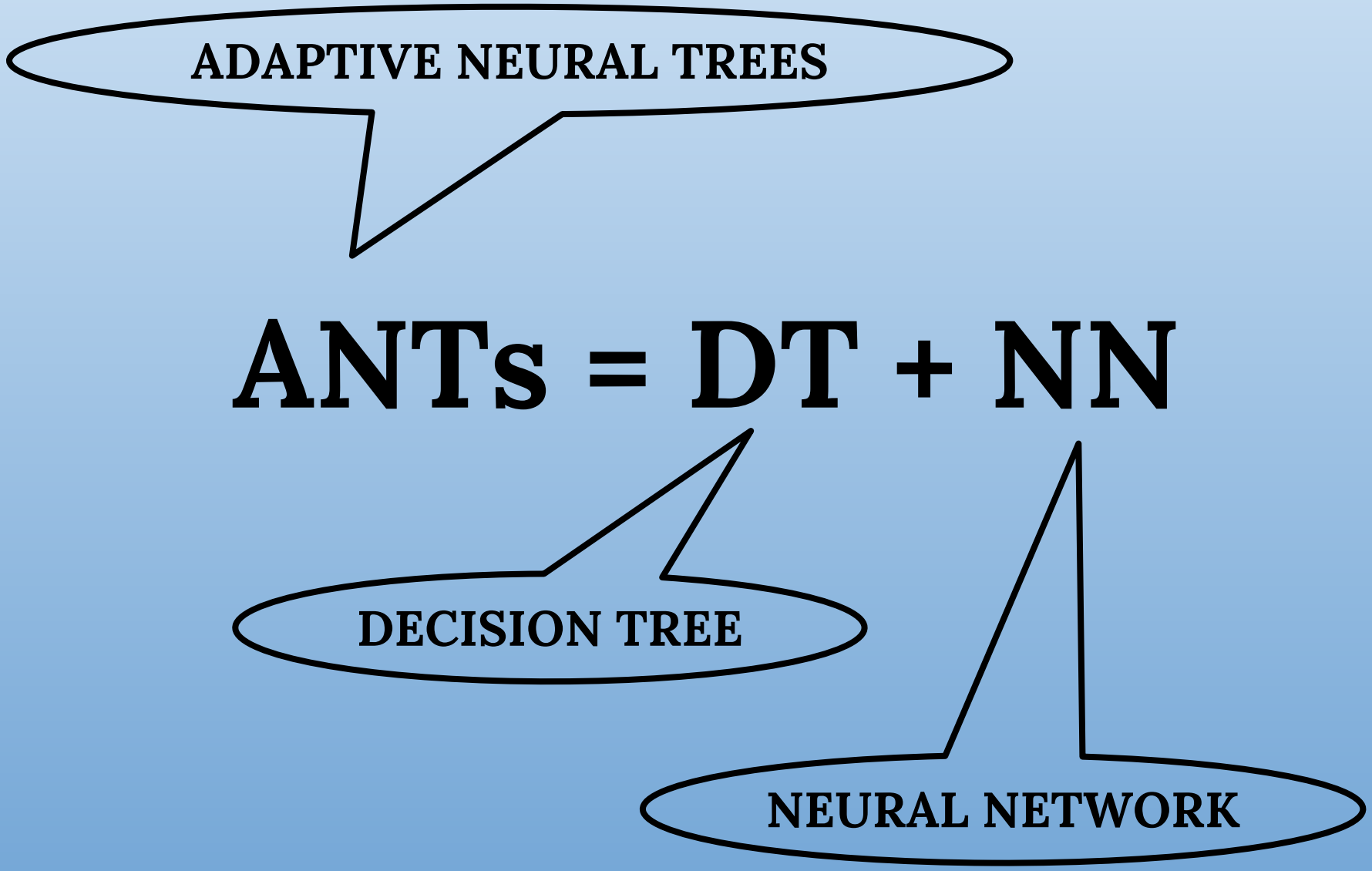
- โครงสร้าง ตาม ข้อมูลฝึกสอน
- พึ่ง feature engineering มาก
- ตอนใช้ คำนวณแค่บางส่วน
- คน สามารถ เข้าใจ ที่มาผลลัพธ์

NEURAL NETWORK



- โครงสร้าง ตาม ที่กำหนดไว้ล่วงหน้า
- ไม่พึ่ง feature engineering มาก
- ตอนใช้ คำนวณทำหมด
- คน เข้าใจ ที่มาผลลัพธ์ ยาก

source : en.m.wikipedia.org/wiki/Neural_network

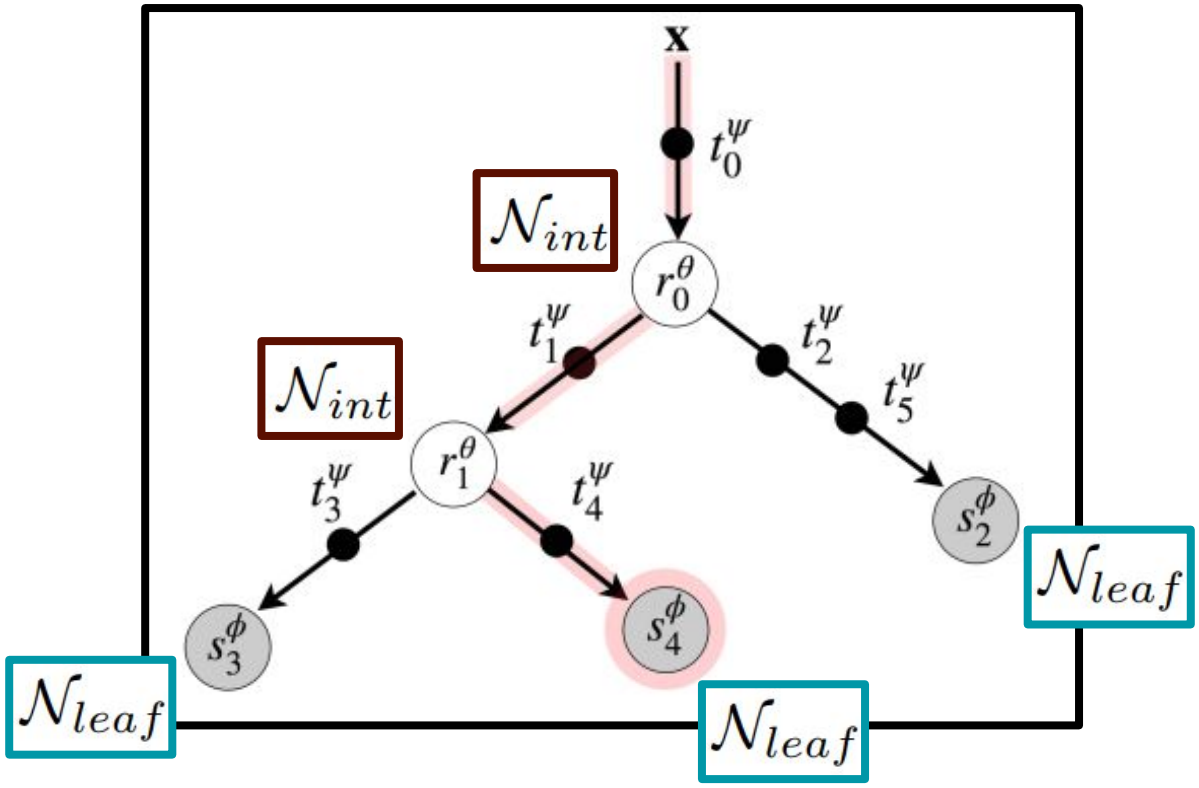

ADAPTIVE NEURAL TREES
ANTs = DT + NN
DECISION TREE
NEURAL NETWORK

2.
Model Topology

Model Topology

Binary Trees

$$\mathbb{T} := \{\mathcal{N}, \mathcal{E}\}$$



"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

3.
Model Operations

Model Operations

$$\mathbb{O} = (\mathcal{R}, \mathcal{T}, \mathcal{S})$$

$$\Theta = (\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi})$$

Routers

$r_j^{\boldsymbol{\theta}}$

Transformers

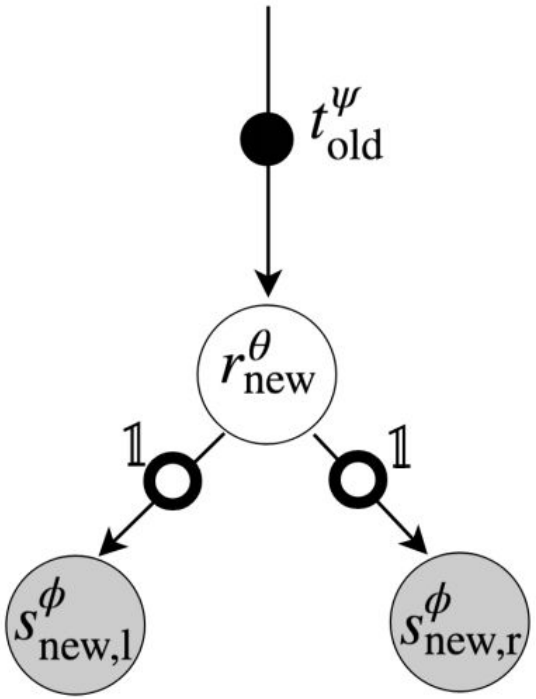
$t_e^{\boldsymbol{\psi}}$

Solvers

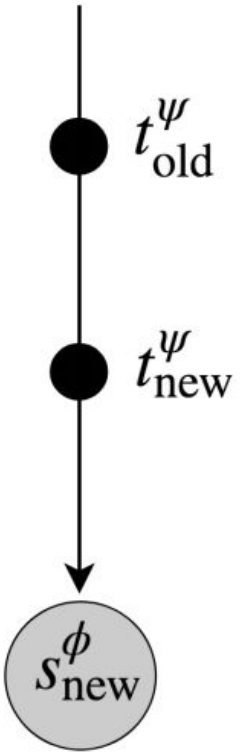
$s_l^{\boldsymbol{\phi}}$

Model Operations

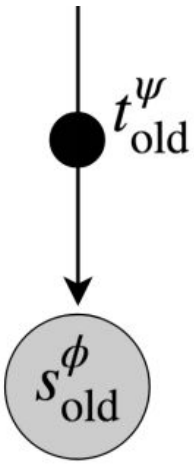
Routers



Transformers



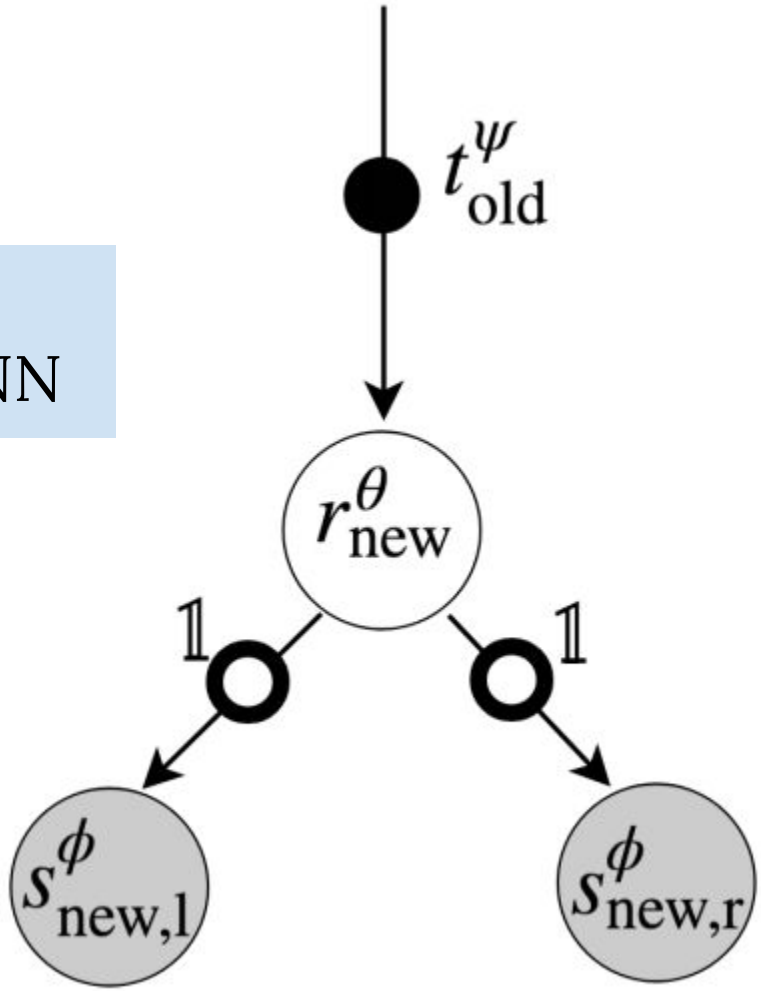
Solvers



"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

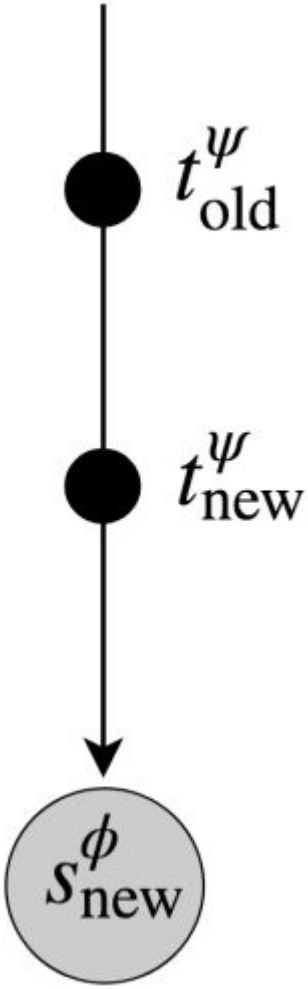
Model Operations : Routers

Routers
Eg. A small CNN



"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

Model Operations : Transformers



Transformers

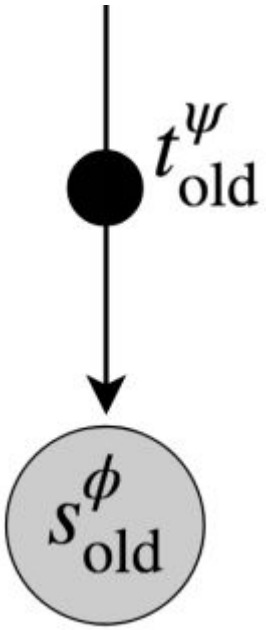
E.g. a single convolutional layer followed by ReLU

"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

Model Operations : Solvers

$$\mathcal{X}_l \rightarrow \mathcal{Y}$$

Solvers
E.g. A linear classifier



"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

4.
Loss Function

Loss Function

1) growth phase
Learn model architecture -> local optimisation

2) refinement phase
-> global optimisation

For both phases,
use the negative log-likelihood (NLL) to minimise:

$$-\log p(\mathbf{Y}|\mathbf{X},\Theta) = -\sum_{n=1}^{N} \log \left(\sum_{l=1}^{L} \pi_l^{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{x}^{(n)})\, p_l^{\boldsymbol{\phi},\boldsymbol{\psi}}(\mathbf{y}^{(n)})\right)$$

Loss Function

$$\mathbf{X} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\},\ \mathbf{Y} = \{\mathbf{y}^{(1)}, ..., \mathbf{y}^{(N)}\}$$

$$-\log p(\mathbf{Y}|\mathbf{X}, \Theta) = -\sum_{n=1}^{N} \log \left(\sum_{l=1}^{L} \pi_l^{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{x}^{(n)})\, p_l^{\boldsymbol{\phi},\boldsymbol{\psi}}(\mathbf{y}^{(n)})\right)$$

$$\pi_l^{\boldsymbol{\psi},\boldsymbol{\theta}}(\mathbf{x}) = \prod_{r_j^{\boldsymbol{\theta}} \in \mathcal{P}_l} r_j^{\boldsymbol{\theta}}(\mathbf{x}_j^{\boldsymbol{\psi}})^{\mathbb{1}_{l \swarrow j}} \cdot \left(1 - r_j^{\boldsymbol{\theta}}(\mathbf{x}_j^{\boldsymbol{\psi}})\right)^{1 - \mathbb{1}_{l \swarrow j}}$$

$$\underbrace{p(\mathbf{y}|\mathbf{x}, z_l = 1, \boldsymbol{\phi}, \boldsymbol{\psi})}$$

Leaf-specific prediction. $p_l^{\boldsymbol{\phi},\boldsymbol{\psi}}$

5.
Growth Phase

Growth Phase

Evaluate 3 choices at each leaf node

(1) split data
by splitting the node with a new router R

(2) deepen transform
increases depth edge by adding a new transformer T

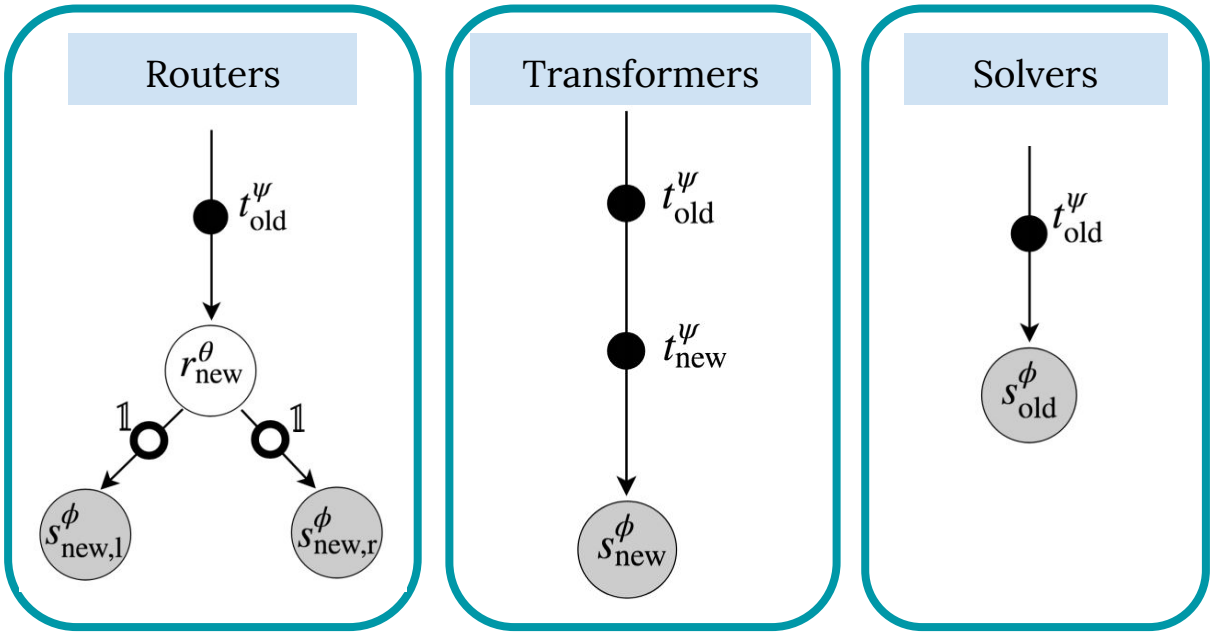
(3) keep
retains the current model

And minimising NLL via gradient descent

Growth Phase

select the model with the lowest NLL

This process is repeated until no more "split data" or "deepen transform" operations pass the validation test



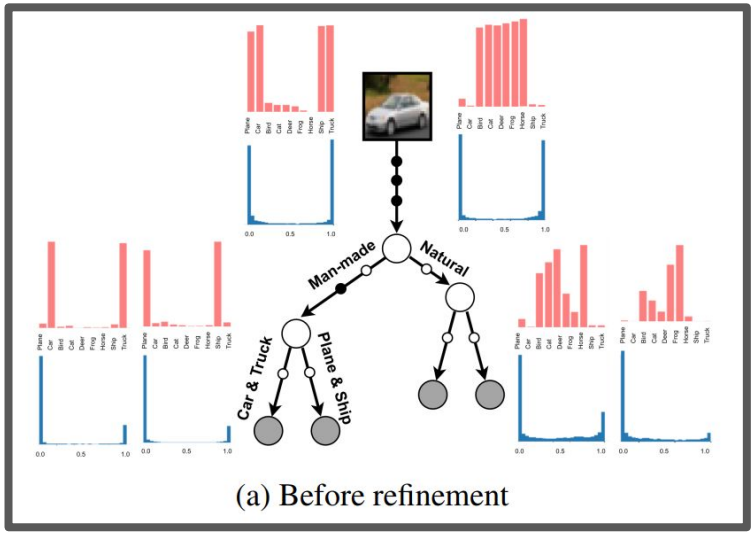
"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

6. Refinement Phase

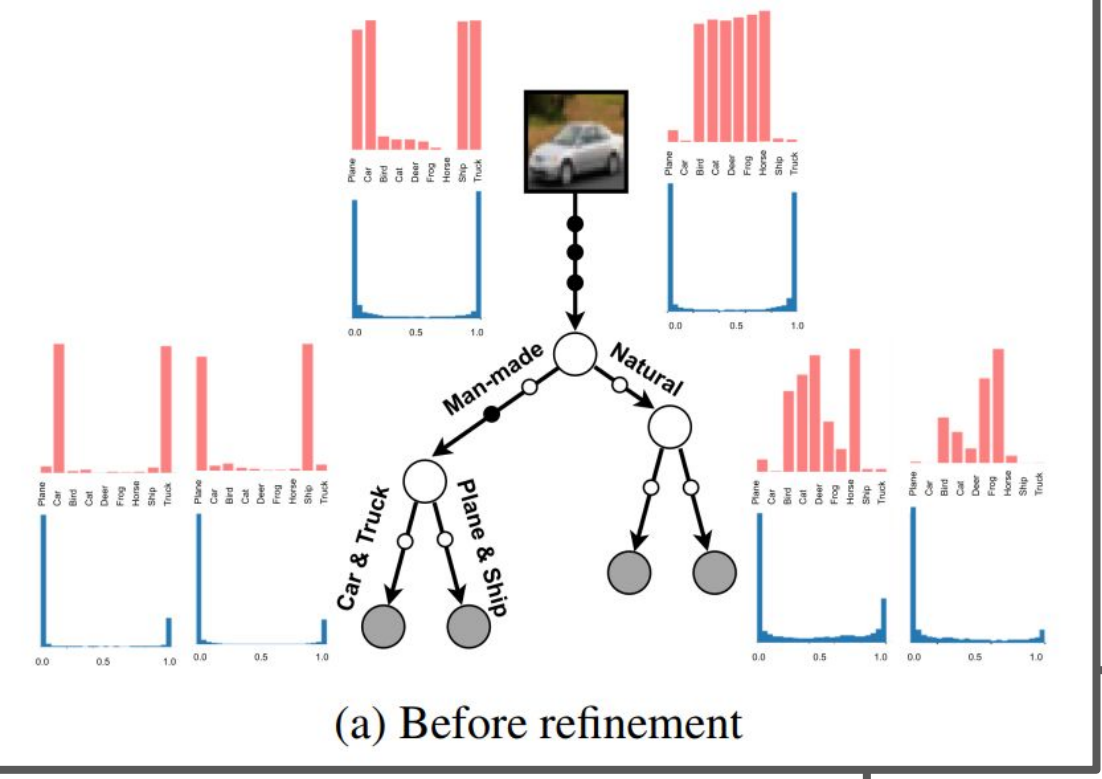
Refinement Phase

After growth phase,
performing global optimisation to
refine the parameters of the model

Gradient descent on the NLL with
respect to the parameters of all modules in the graph



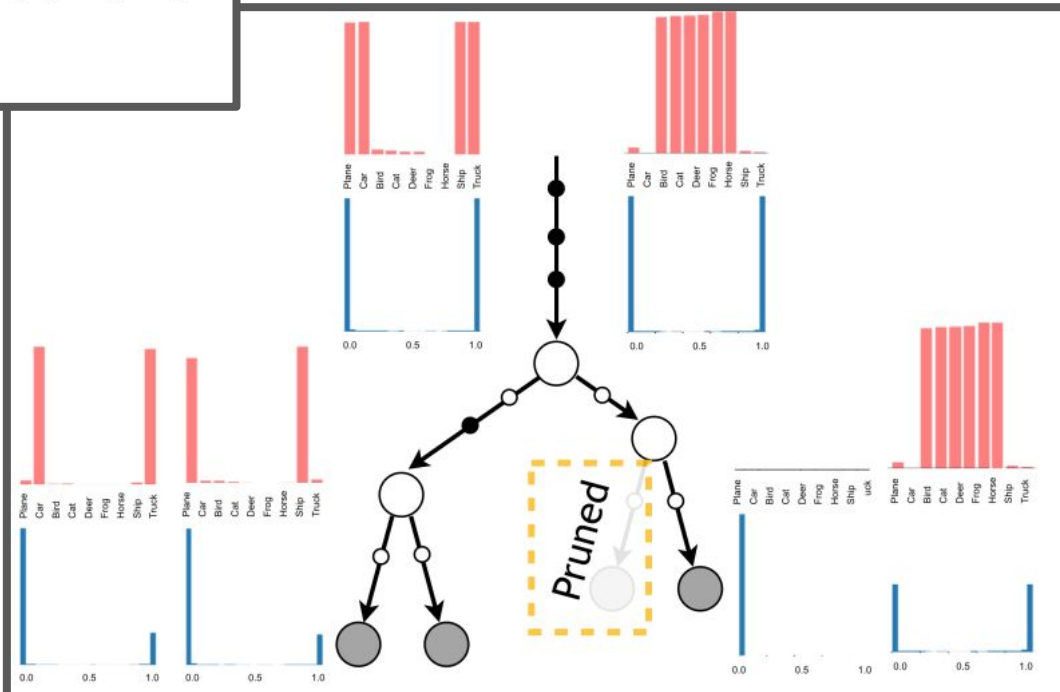
"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf



(a) Before refinement

class distributions (red)

path probabilities (blue)



(b) After refinement

"Adaptive Neural Trees", by R. Tanno et al. ,
from arxiv.org/pdf/1807.06699.pdf

7.
Algorithm

Algorithm

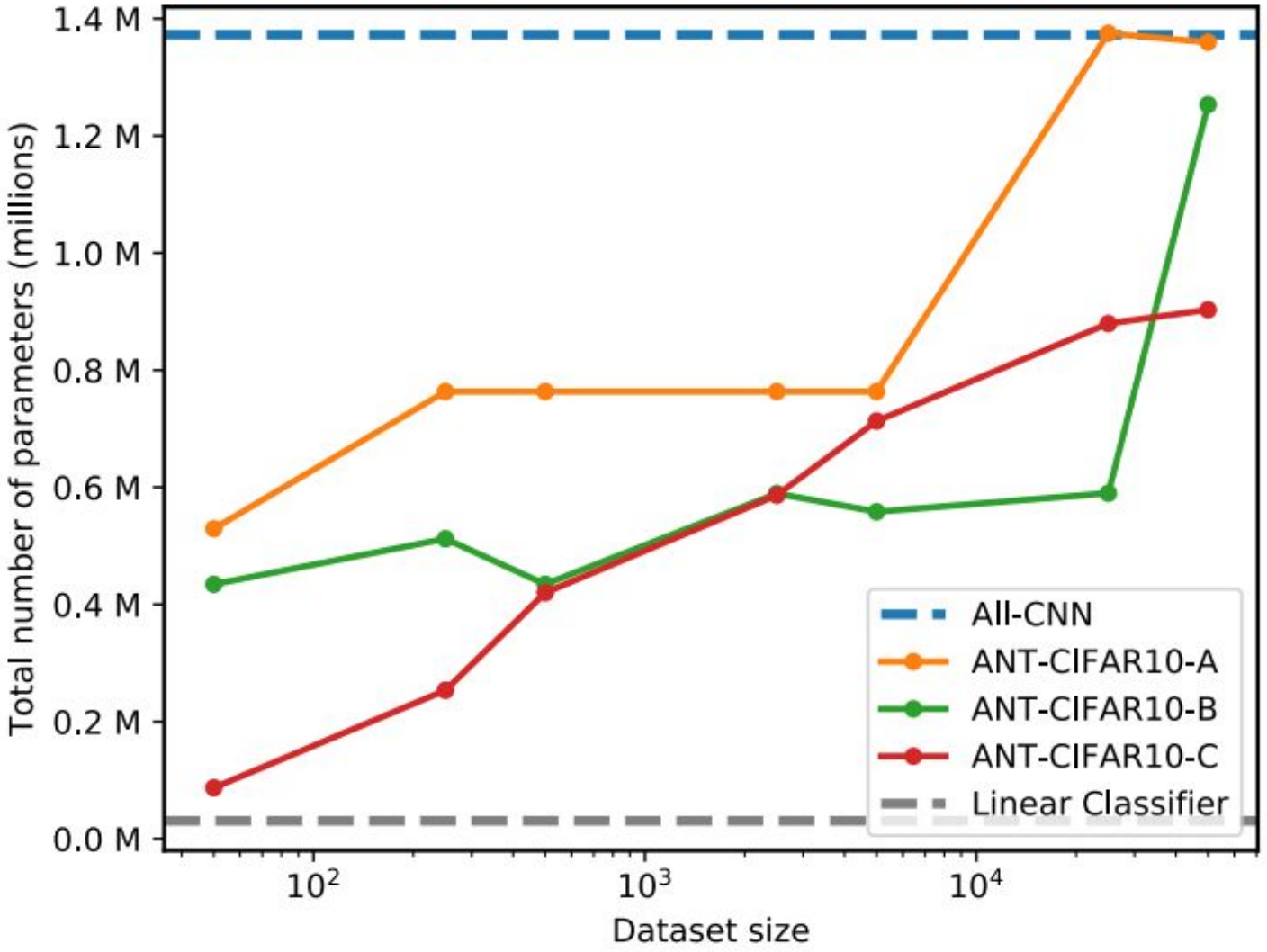
Algorithm 1 ANT Optimisation

Initialise topology $\mathbb{T}$ and parameters $\mathbb{O}$ ▷ $\mathbb{T}$ is set to a root node with one solver and one transformer
Optimise parameters in $\mathbb{O}$ via gradient descent on NLL ▷ Learning root classifier
Set the root node "suboptimal"
while true **do** ▷ Growth of $\mathbb{T}$ begins
    Freeze all parameters $\mathbb{O}$
    Pick next "suboptimal" leaf node $l \in \mathcal{N}_{leaf}$ in the breadth-first order
    Add (1) router to l and train new parameters ▷ Split data
    Add (2) transformer to l and train new parameters ▷ Deepen transform
    Add (1) or (2) to $\mathbb{T}$ if validation error decreases, otherwise set l to "optimal"
    Add any new modules to $\mathbb{O}$
    if no "suboptimal" leaves remain **then**
        Break
Unfreeze and train all parameters in $\mathbb{O}$ ▷ Global refinement with fixed $\mathbb{T}$

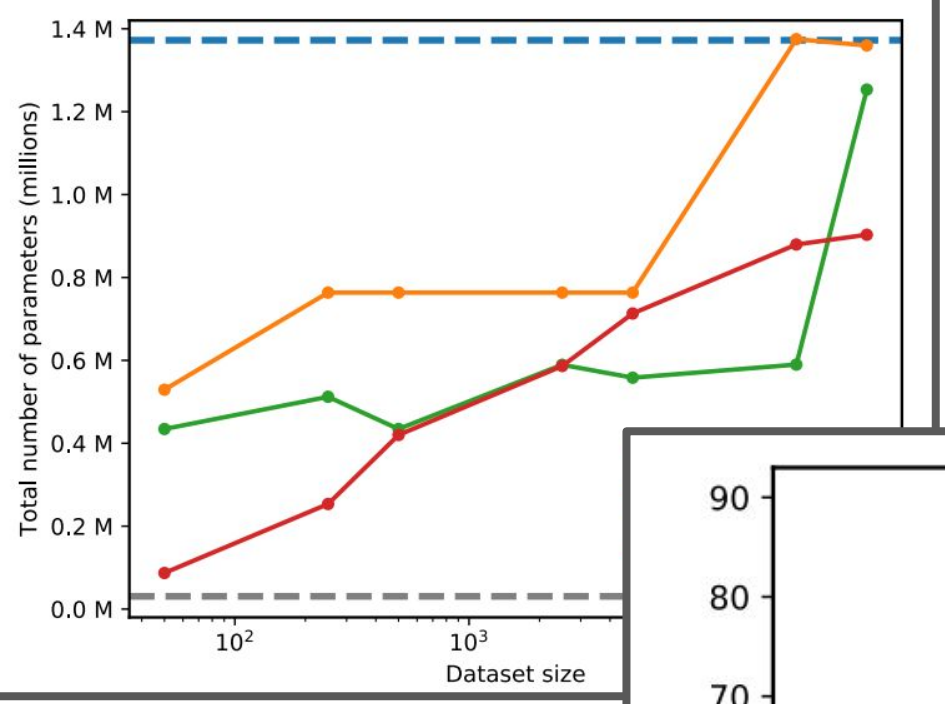
"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

8.
Compare

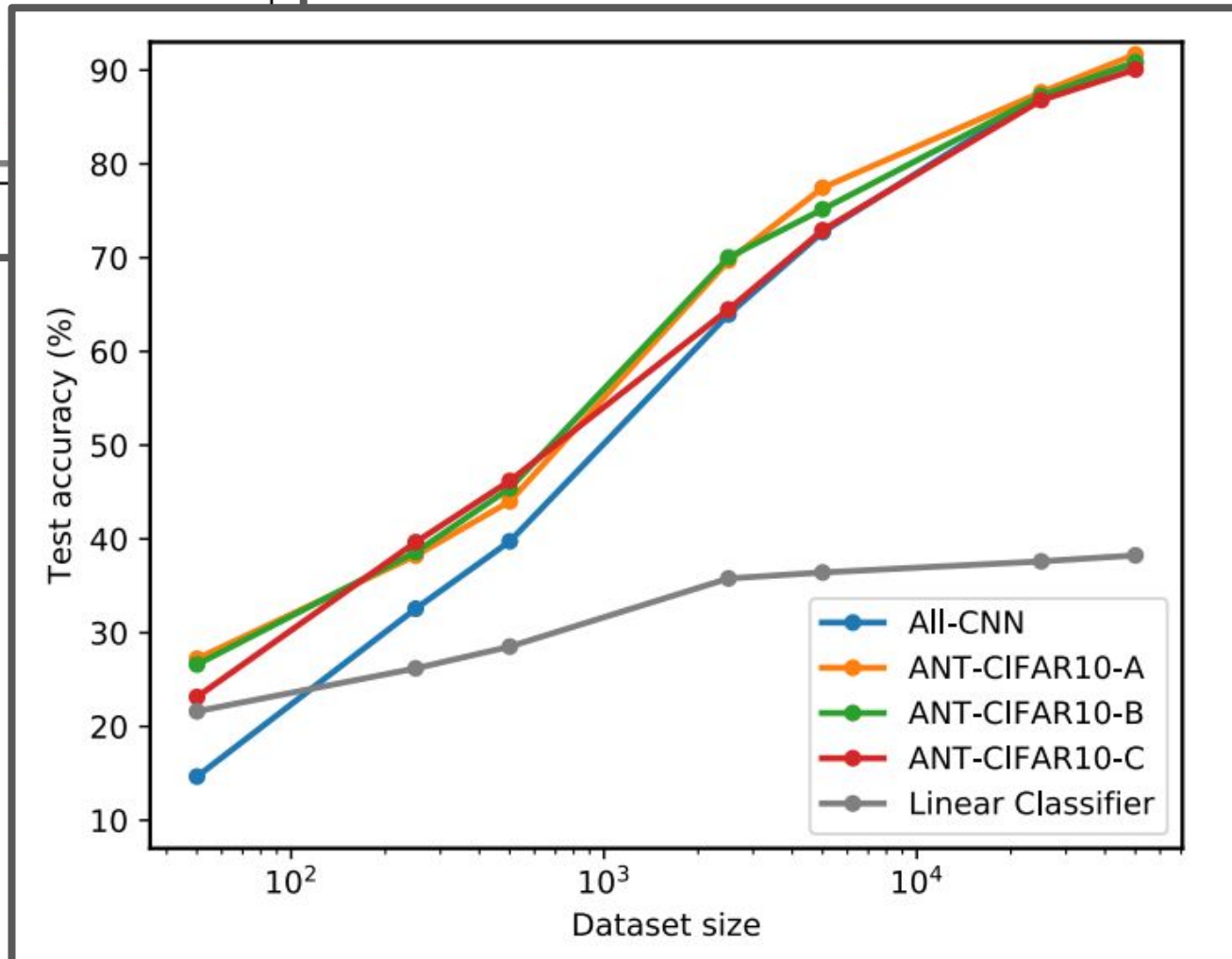
Compare



"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf



"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf



9. Training Time

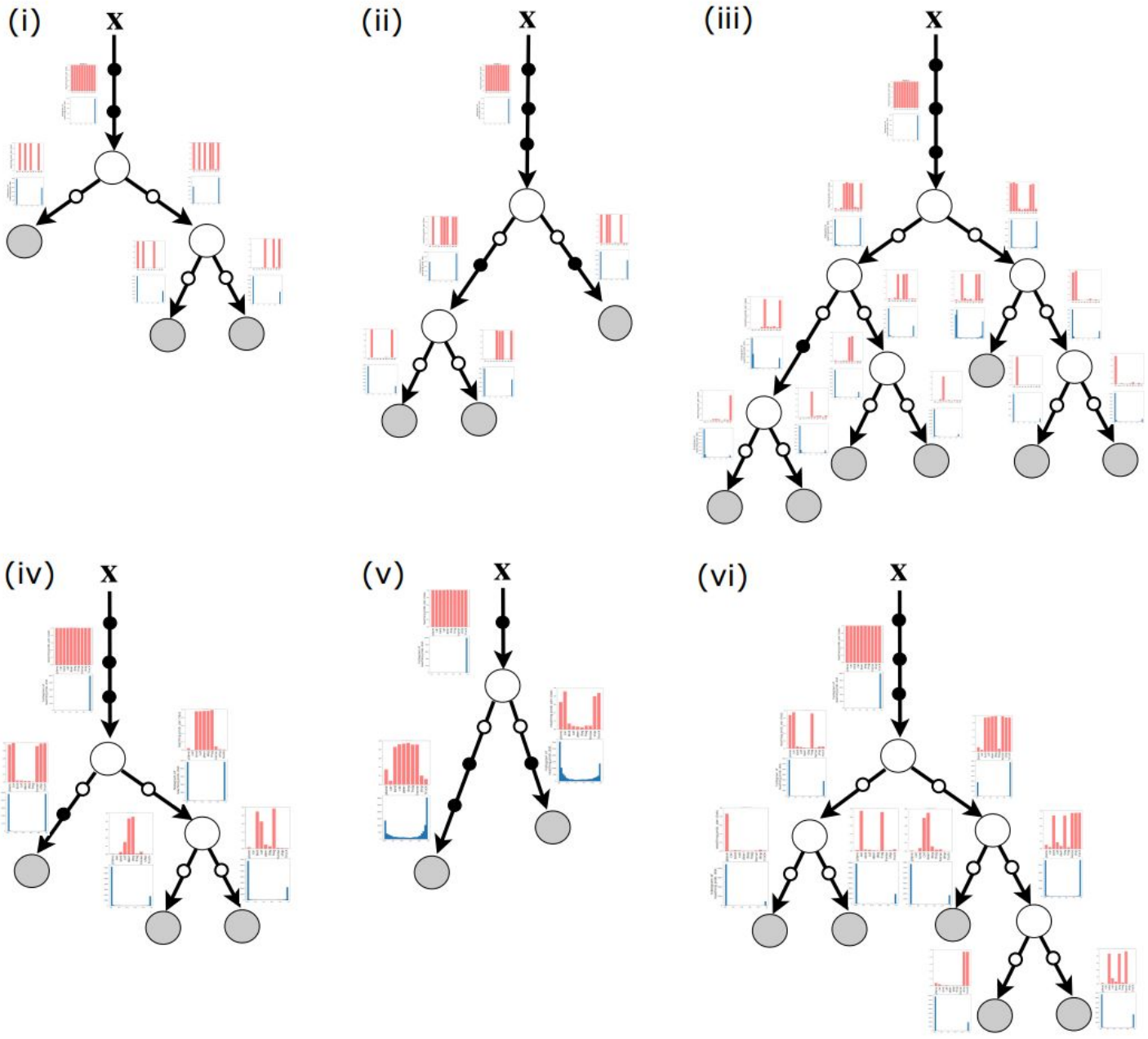
Training Time

A single Titan X GPU

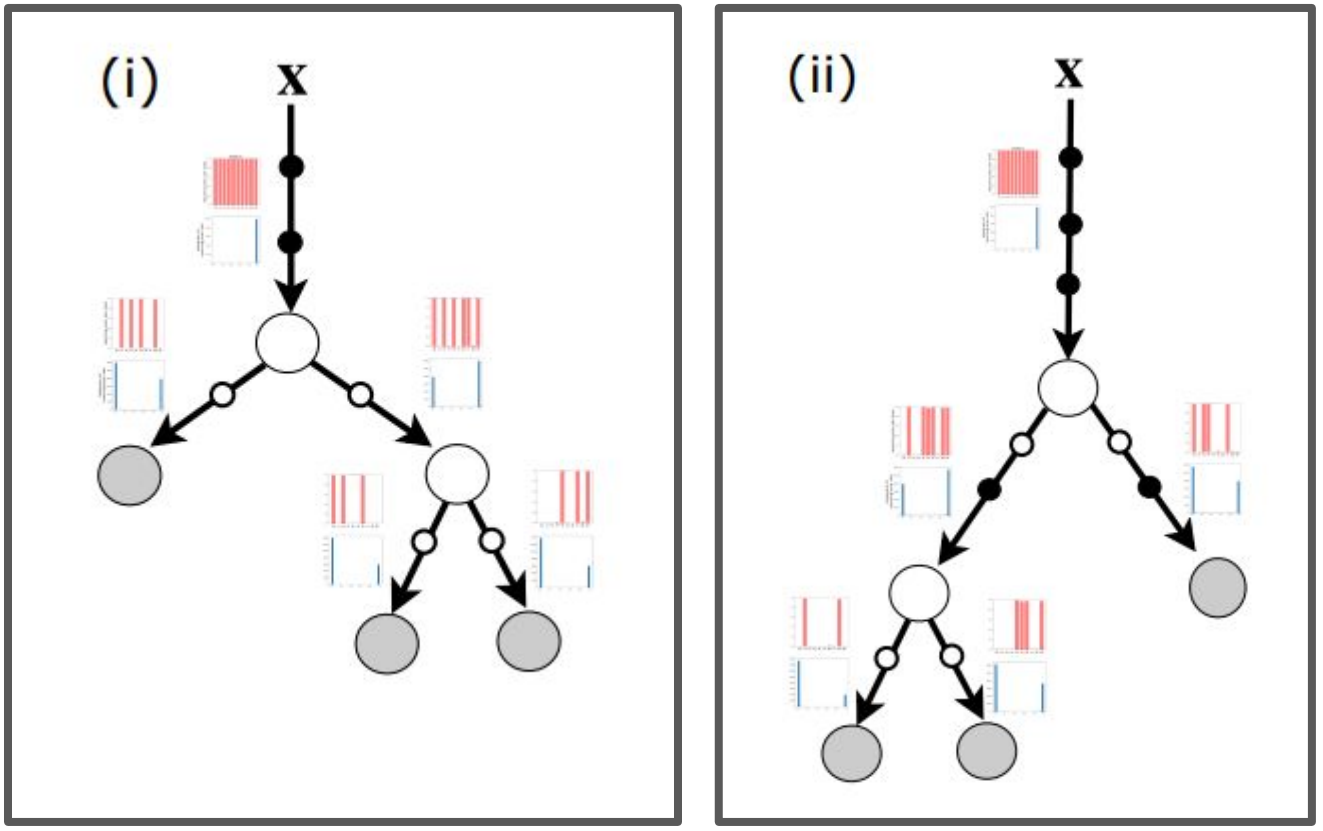
	Growth		Fine-tune	
Model	Time	Epochs	Time	Epochs
All-CNN (baseline)	–	–	1.1 (hr)	200
ANT-CIFAR10-A	1.3 (hr)	236	1.5 (hr)	200
ANT-CIFAR10-B	0.8 (hr)	313	0.9 (hr)	200
ANT-CIFAR10-C	0.7 (hr)	285	0.8 (hr)	200

"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

10.
Discovered Architectures



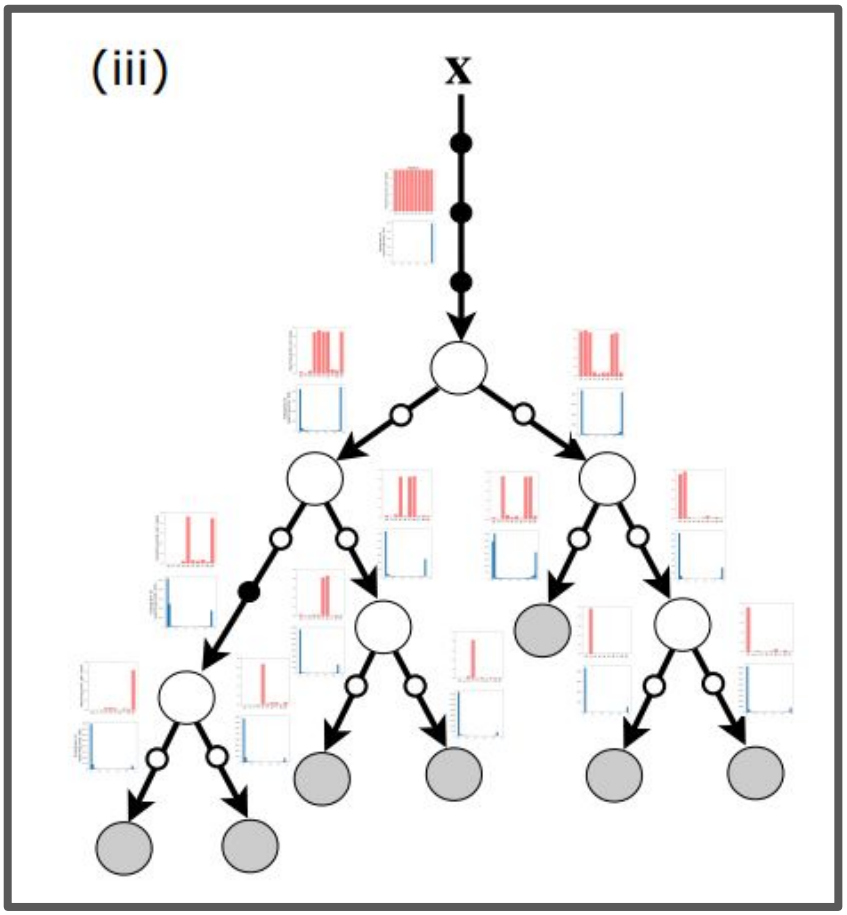
"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

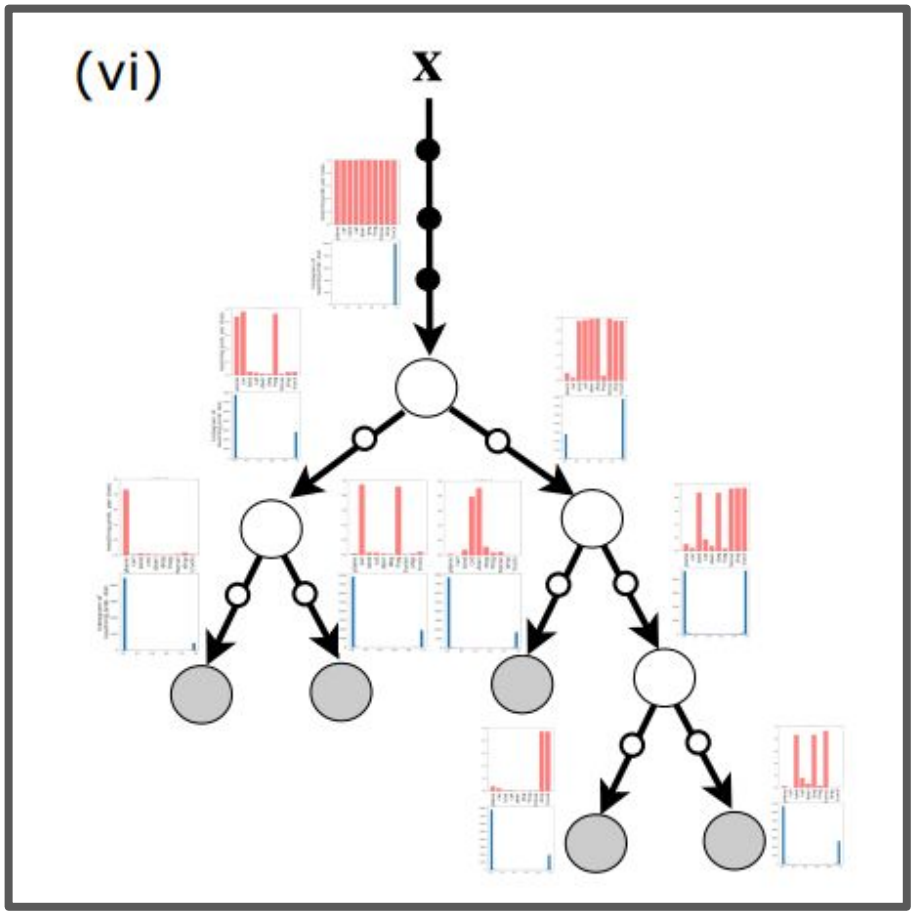


"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

Routers separate examples based on their classes (red histograms)

Most architectures learn a few levels of features before first splits



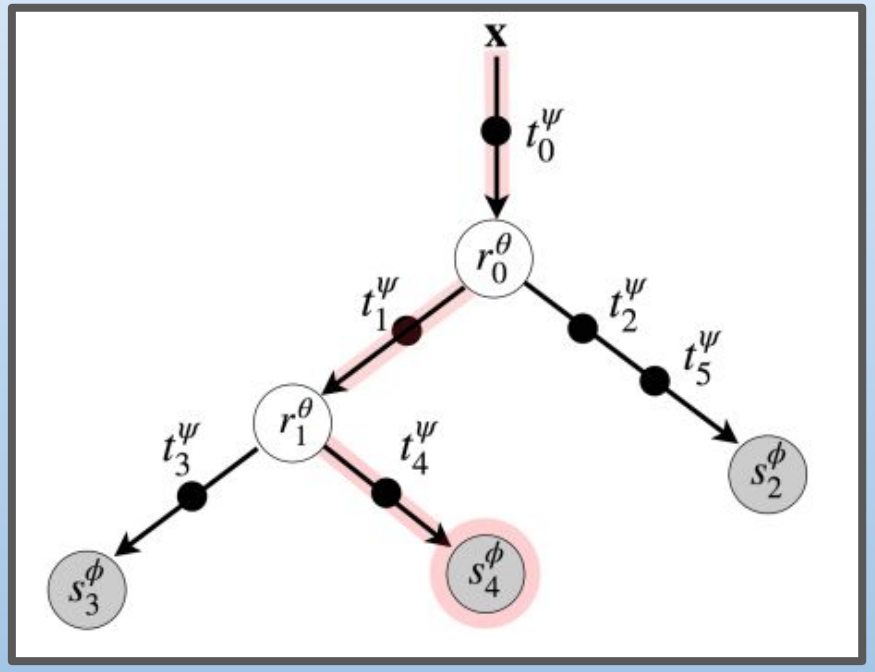


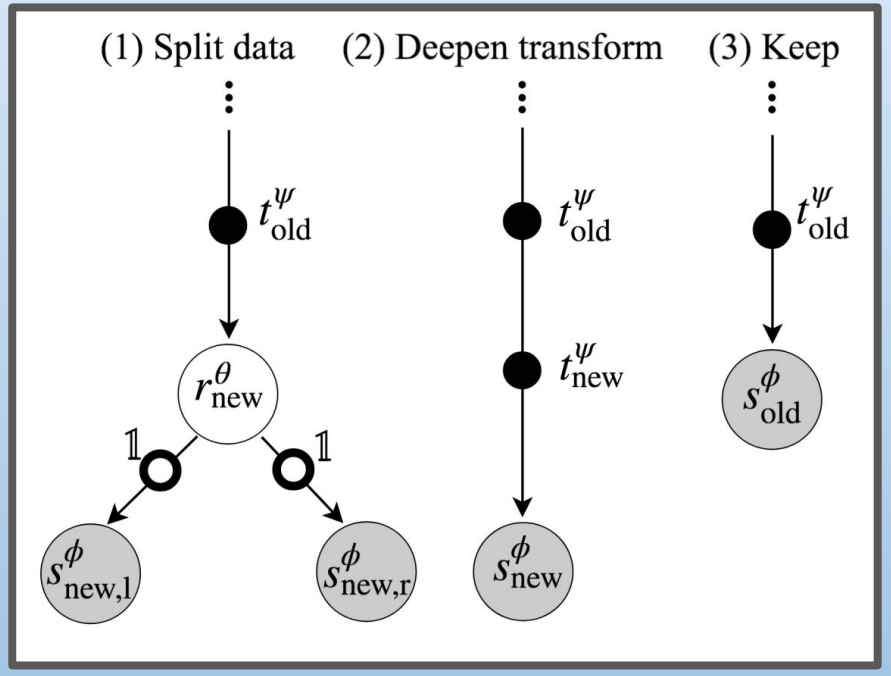
"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

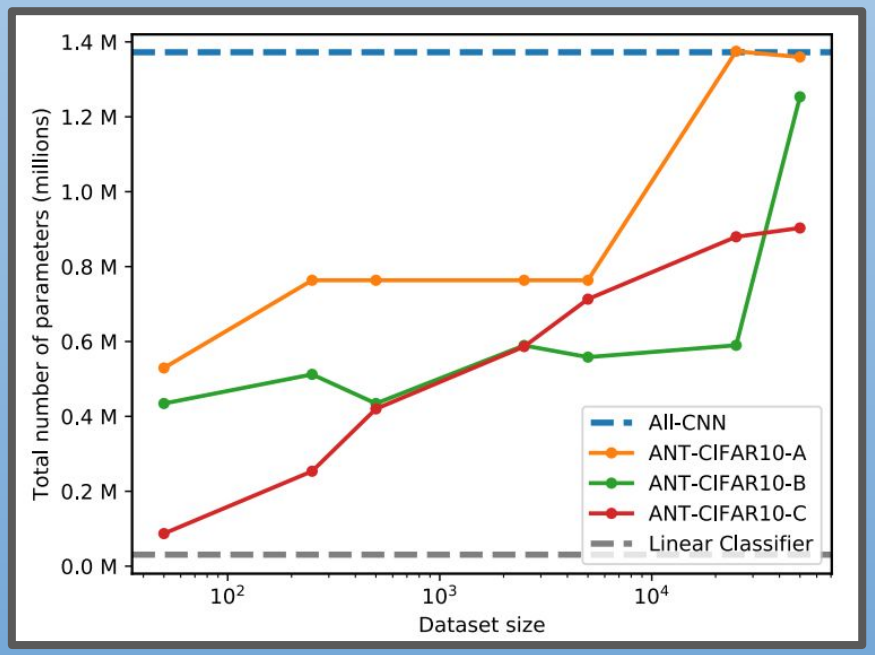
All architectures are unbalanced
->Some samples be easier to classify than others

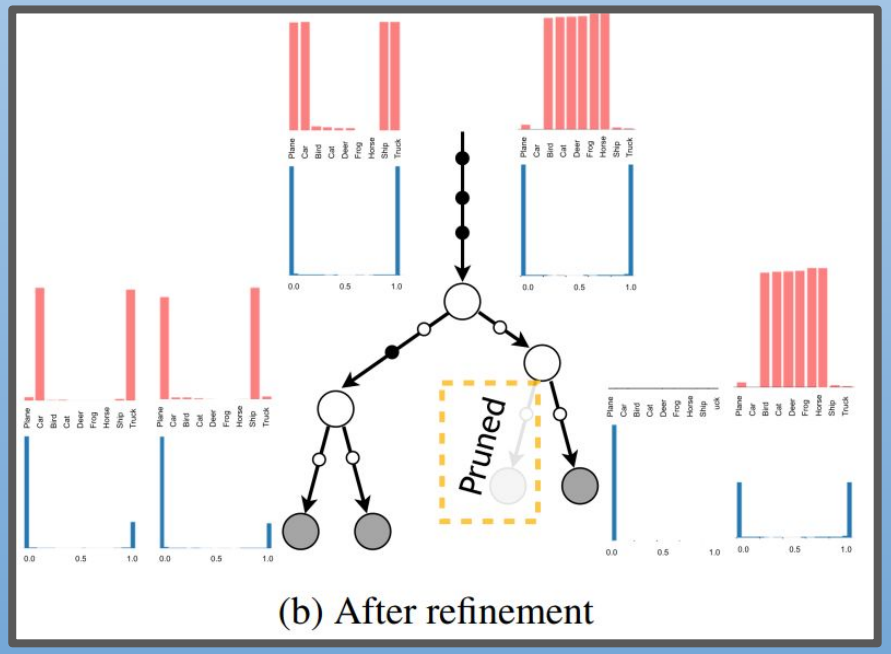
->Traditional DT algorithms,
but not NN models that prespecified architectures

Summary









"Adaptive Neural Trees", by R. Tanno et al. , from arxiv.org/pdf/1807.06699.pdf

Thank You
&
Discussion