Adversarial Neural Trees

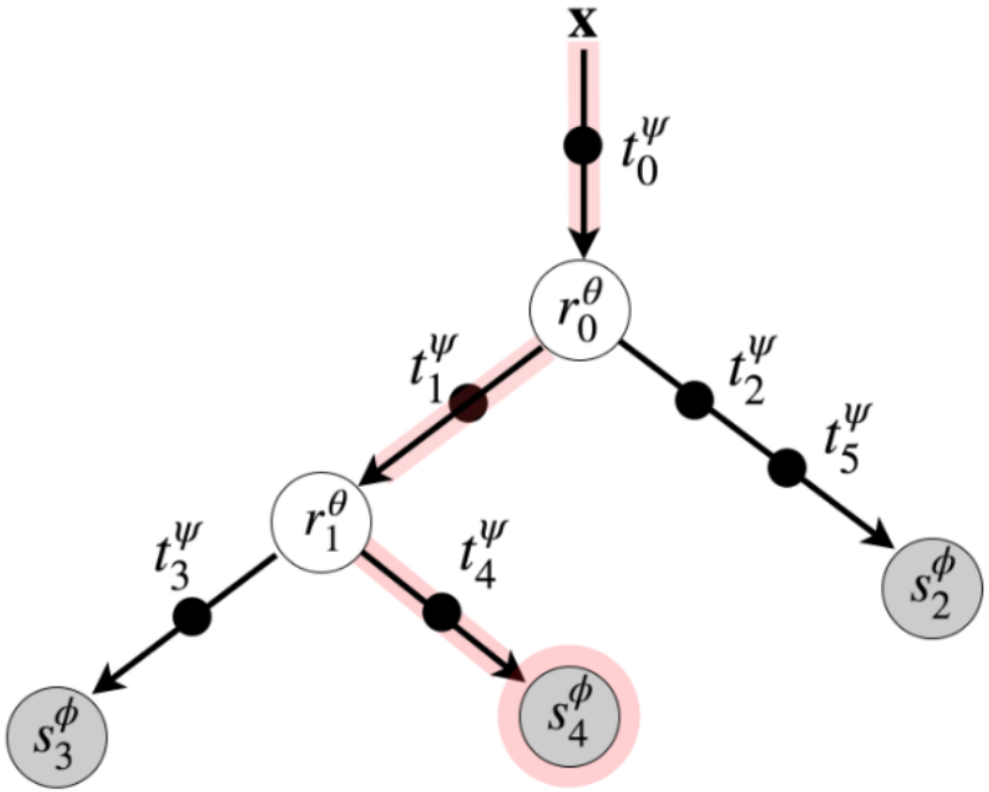
Adaptive Neural Trees

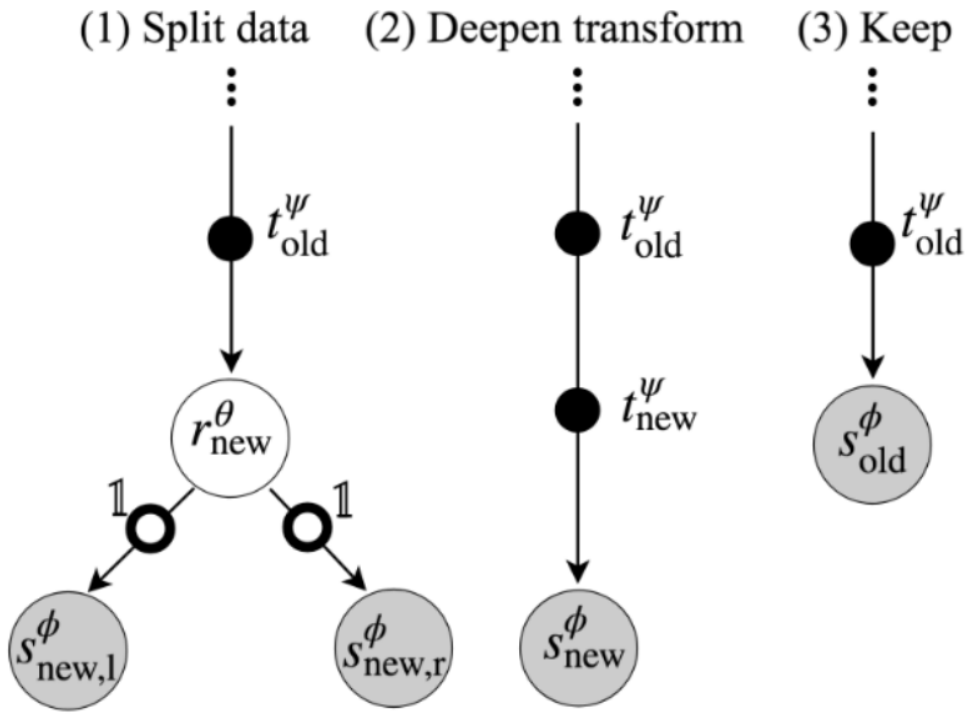
- https://arxiv.org/pdf/1807.06699.pdf
- Submit: Ryutaro Tanno, Kai Arulkumaran, Daniel C. Alexander, Antonio Criminisi, Aditya Nori. ICLR (2019)
- Arxiv에는 2018년 6월에 공개됨

Machine Learning의 두 갈래

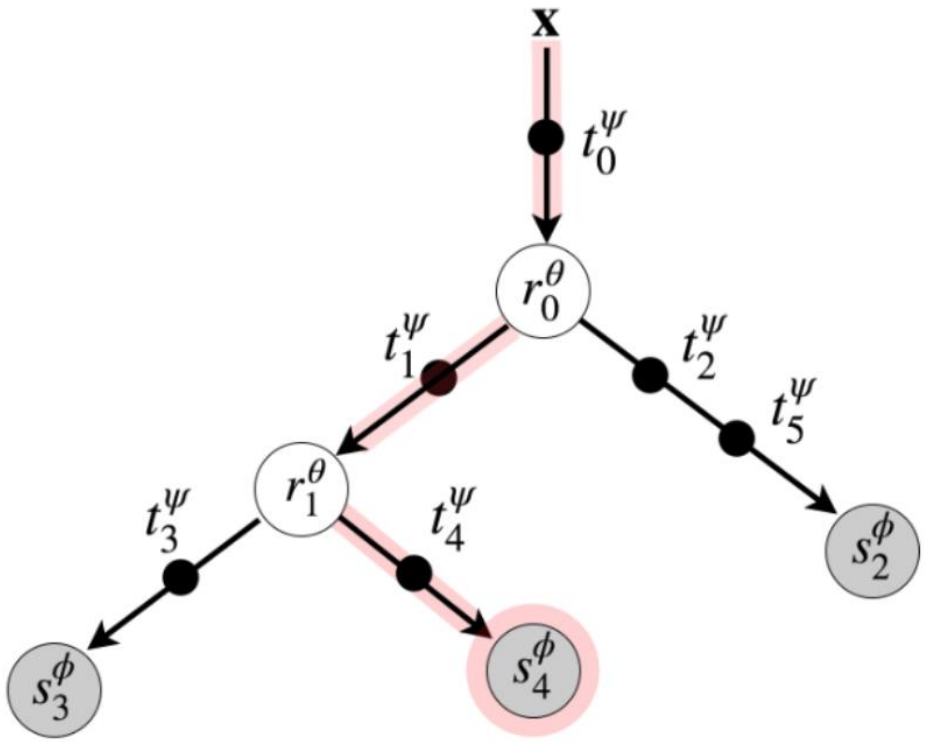
- Neural Network: 구조는 사람이 지정, feature는 자동으로 학습
  - 모델 구조는 사람이 넣어 줘야 하고 하이퍼파라미터 튜닝도 필요
  - Optimizing을 통해 자동으로 파라미터 학습, 이 과정에서 feature를 자동으로 학습
- Decision Tree: 구조는 자동으로 학습, feature는 사람이 지정
  - Tree 구조는 학습을 통해 학습 가능
  - 사람 손을 이용해 수동으로 feature 추출, 모델에 넣어 주어야 함
- 구조를 자동으로 생성하는 Decision Tree의 장점, feature를 자동으로 학습하는 Neural Network의 장점을 합쳐서 새로운 모델을 만들자

Adaptive Neural Trees



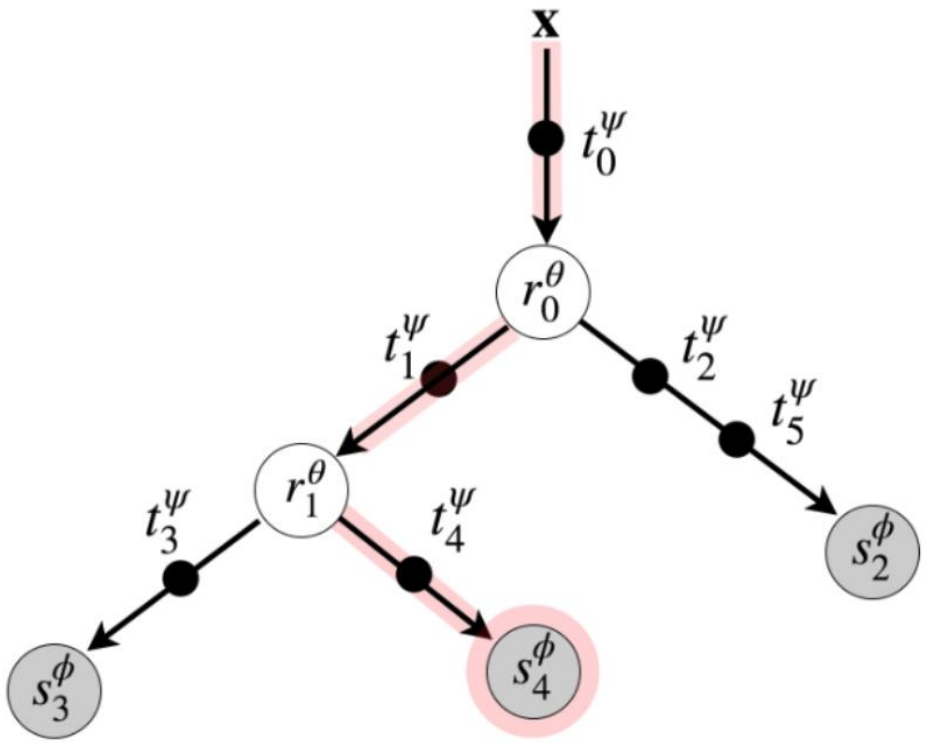


Adaptive Neural Trees (구조)



- Routers (R)
  - 들어온 edge(데이터)를 어느 쪽 child로 보낼 것이 결정
  - 작은 CNN 모델로 구현 가능
- Transformers (T)
  - 데이터를 변형
  - 1차원 CNN + Relu 등으로 구현
- Solvers (S)
  - 전체 class에 대한 Classification 을 수행하는 단계

Adaptive Neural Trees



1. 데이터를 Tree에 넣는다.
2. Router를 거치며, 더 적합한 child로 이동한다
3. Transformer를 만나면 데이터가 변형된다
4. Tree의 leaf node(Solver)에서 classification을 한다

→ Router가 정한 경로에 따라 다른 Solver를 사용하게 됨

Loss function

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

$$p(\mathbf{y}|\mathbf{x},\Theta) = \sum_{l=1}^{L} \underbrace{p(z_l = 1|\mathbf{x},\boldsymbol{\theta},\boldsymbol{\psi})}_{\text{Leaf-assignment prob. } \pi_l^{\boldsymbol{\theta},\boldsymbol{\psi}}} \underbrace{p(\mathbf{y}|\mathbf{x}, z_l = 1, \boldsymbol{\phi},\boldsymbol{\psi})}_{\text{Leaf-specific prediction. } p_l^{\boldsymbol{\phi},\boldsymbol{\psi}}}$$

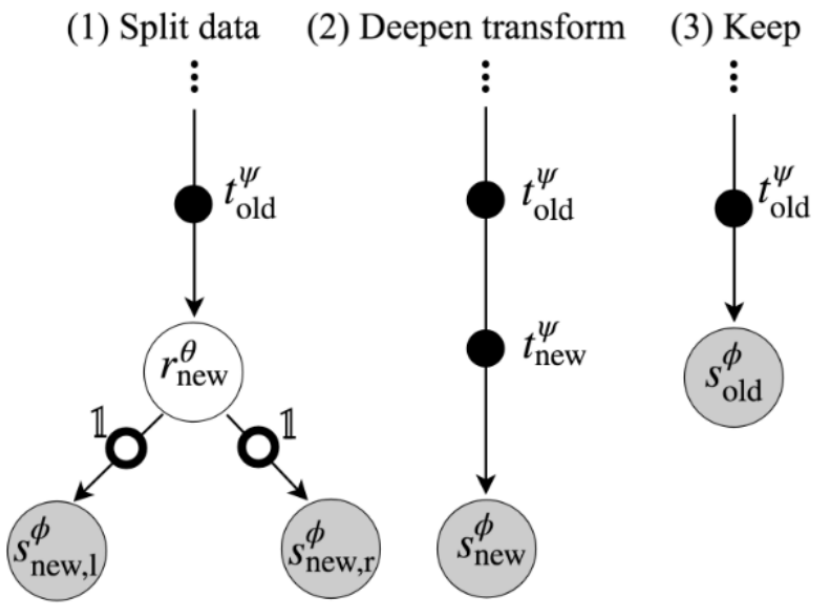
$$\pi_l^{\boldsymbol{\psi},\boldsymbol{\theta}}(\mathbf{x}) = \prod_{r_j^{\boldsymbol{\theta}} \in \mathcal{P}_l} r_j^{\boldsymbol{\theta}}(\mathbf{x}_j^{\boldsymbol{\psi}})^{\mathbb{1}[l \swarrow j]} \cdot \left(1 - r_j^{\boldsymbol{\theta}}(\mathbf{x}_j^{\boldsymbol{\psi}})\right)^{1-\mathbb{1}[l \swarrow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

$$\mathbf{x}_j^{\boldsymbol{\psi}} := \left(t_{e_n}^{\boldsymbol{\psi}} \circ \ldots \circ t_{e_2}^{\boldsymbol{\psi}} \circ t_{e_1}^{\boldsymbol{\psi}}\right)(\mathbf{x}).$$

- Negative log-likelihood
- $p(y|x,\Theta)$: 이 leaf를 선택했을 때 기대되는, 올바르게 classification을 할 확률
- 좌측과 같은 loss function을 이용해 모델의 구조를 학습한다

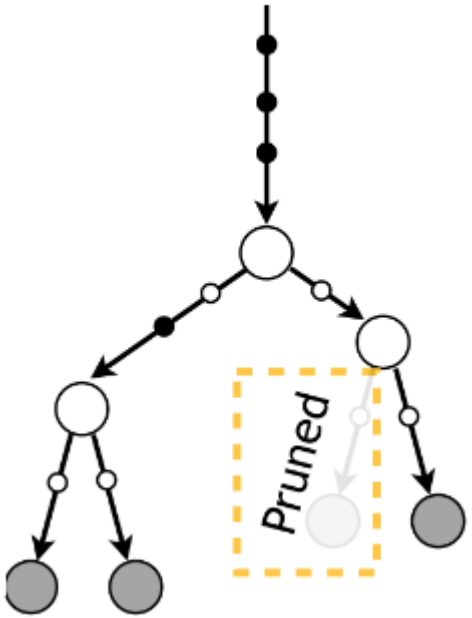
- Θ: 전체 parameter
- ψ: Transformer의 parameter
- θ: Router의 parameter

학습 방법 - Growth phase



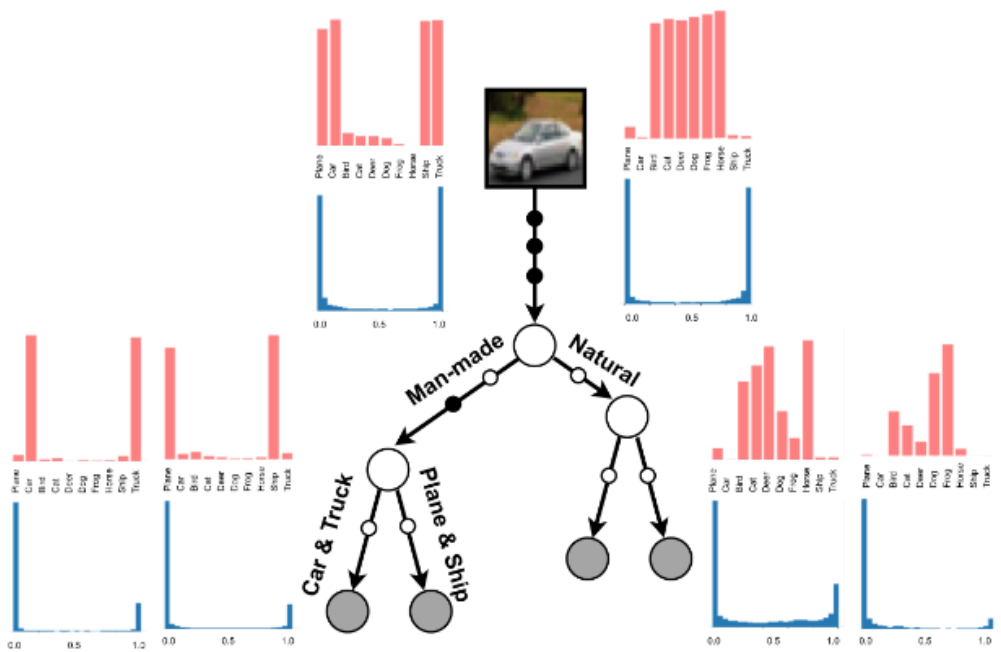
- 트리를 확장시켜 나가는 과정
- Split data: router를 새로 만들고 node를 split
- Deepen transform: incoming edge에 transformer 하나 더 추가
- Keep: 현재 모델 유지

학습 방법 – Refine Phase

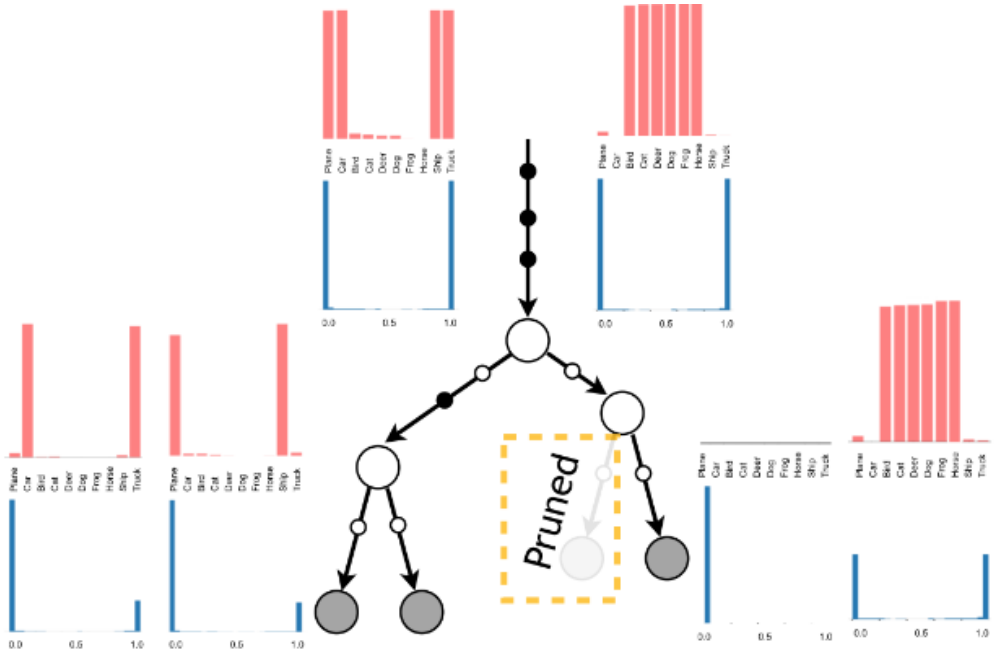


- Growth Phase에서 생성된 모델 구조 튜닝
- 학습 초반에는 Growth Phase만 하다가, 일정 epoch가 지나면 Refine 시작

Refinement의 의의



(a) Before refinement



(b) After refinement

Refinement를 하면 같은 class의 이미지는 같은 edge로 이동하여,
Solver가 한정된 class에 최적화될 여지를 줄 수 있음

결과

Model	Router, $\mathcal{R}$	Transformer, $\mathcal{T}$	Solver, $\mathcal{S}$	Downsample Freq.
ANT-SARCOS	1 × FC +Sigmoid	1×FC+ tanh	LR	0
ANT-MNIST-A	1 × conv5-40 + GAP + 2×FC +Sigmoid	1 × conv5-40 + ReLU	LC	1
ANT-MNIST-B	1 × conv3-40 + GAP + 2×FC +Sigmoid	1 × conv3-40 +ReLU	LC	2
ANT-MNIST-C	1 × conv5-5 + GAP + 2×FC +Sigmoid	1 × conv5-5+ReLU	LC	2
ANT-CIFAR10-A	2 × conv3-128 + GAP + 1×FC +Sigmoid	2 × conv3-128 +ReLU	GAP + LC	1
ANT-CIFAR10-B	2 × conv3-96 + GAP + 1×FC +Sigmoid	2 × conv3-96 +ReLU	LC	1
ANT-CIFAR10-C	2 × conv3-48 + GAP + 1×FC +Sigmoid	2 × conv3-96 +ReLU	GAP + LC	1

결과

	Method	Error (multi-path)	Error (single-path)	Params. (multi-path)	Params. (single-path)	Ensemble Size
SARCOS	Linear regression	10.693	N/A	154	N/A	1
	MLP with 2 hidden layers (Zhao et al., 2017)	5.111	N/A	31,804	N/A	1
	Decision tree	3.708	3.708	319,591	25	1
	MLP with 1 hidden layer	2.835	N/A	7,431	N/A	1
	Gradient boosted trees	2.661	2.661	391,324	2,083	7 × 30
	MLP with 5 hidden layers	2.657	N/A	270,599	N/A	1
	Random forest	2.426	2.426	40,436,840	4,791	200
	Random forest	2.394	2.394	141,540,436	16,771	700
	MLP with 3 hidden layers	2.129	N/A	139,015	N/A	1
	SDT (with MLP routers)	2.118	2.246	28,045	10,167	1
	Gradient boosted trees	1.444	1.444	988,256	6,808	7 × 100
	ANT-SARCOS	1.384	1.542	103,823	61,640	1
	ANT-SARCOS (ensemble)	1.226	1.372	598,280	360,766	8
MNIST	Linear classifier	7.91	N/A	7,840	N/A	1
	RDT (Léon & Denoyer, 2015)	5.41	–	–	–	1
	Random Forests (Breiman, 2001)	3.21	3.21	–	–	200
	Compact Multi-Class Boosted Trees (Ponomareva et al., 2017)	2.88	–	–	–	100
	Alternating Decision Forest (Schulter et al., 2013)	2.71	2.71	–	–	20
	Neural Decision Tree (Xiao, 2017)	2.10	–	1,773,130	502,170	1
	ANT-MNIST-C	1.62	1.68	39,670	7,956	1
	MLP with 2 hidden layers (Simard et al., 2003)	1.40	N/A	1,275,200	N/A	1
	LeNet-5† (LeCun et al., 1998)	0.82	N/A	431,000	N/A	1
	gcForest (Zhou & Feng, 2017)	0.74	0.74	–	–	500
	ANT-MNIST-B	0.72	0.73	76,703	50,653	1
	Neural Decision Forest (Kontschieder et al., 2015)	0.70	–	544,600	463,180	10
	ANT-MNIST-A	0.64	0.69	100,596	84,935	1
	ANT-MNIST-A (ensemble)	0.29	0.30	850,775	655,449	8
	CapsNet (Sabour et al., 2017)	0.25	–	8.2M	N/A	1
CIFAR-10	Compact Multi-Class Boosted Trees (Ponomareva et al., 2017)	52.31	–	–	–	100
	Random Forests (Breiman, 2001)	50.17	50.17	–	–	2000
	gcForest (Zhou & Feng, 2017)	38.22	38.22	–	–	500
	MaxOut (Goodfellow et al., 2013)	9.38	N/A	6M	N/A	1
	ANT-CIFAR10-C	9.31	9.34	0.7M	0.5M	1
	ANT-CIFAR10-B	9.15	9.18	0.9M	0.6M	1
	Network in Network (Lin et al., 2014)	8.81	N/A	1M	N/A	1
	All-CNN† (Springenberg et al., 2015)	8.71	N/A	1.4M	N/A	1
	ANT-CIFAR10-A	8.31	8.32	1.4M	1.0M	1
	ANT-CIFAR10-A (ensemble)	7.71	7.79	8.7M	7.4M	8
	ANT-CIFAR10-A*	6.72	6.74	1.3M	0.8M	1
	ResNet-110 (He et al., 2016)	6.43	N/A	1.7M	N/A	1
	DenseNet-BC (k=24) (Huang et al., 2017)	3.74	N/A	27.2M	N/A	1

결과

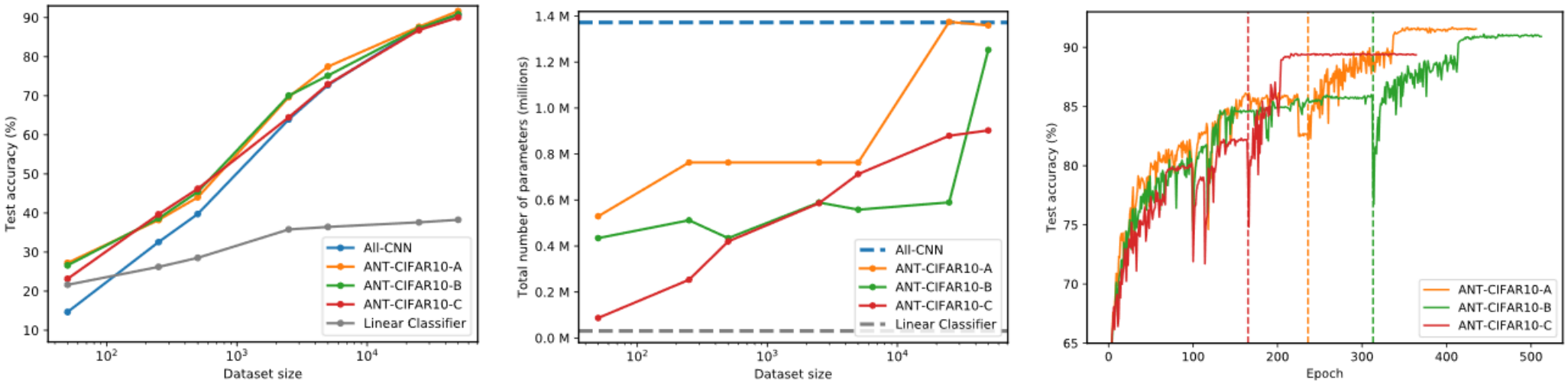


Figure 3: (Left). Test accuracy on CIFAR-10 of ANTs for varying amounts of training data. (Middle) The complexity of the grown ANTs increases with dataset size. (Right) Refinement improves generalisation; the dotted lines show where the refinement phase starts.

1. 데이터 셋 크기가 커질수록, 모델이 복잡할수록 Accuracy가 커진다
2. 데이터 셋 크기가 커질수록, 모델이 복잡할수록 파라미터 수도 많아진다
3. Pruning을 하면 accuracy가 높아진다

결론

- Adaptive Neural Tree를 이용하면 Neural Net과 Decision Tree의 장점을 모두 가질 수 있다.
- Router, Transformer, Solver를 간단한 모델로 사용하기 때문에, 필요한 파라미터 수를 줄일 수 있으며, 기존 모델과 유사한 성능을 보인다.
- Router, Transformer, Solver를 간단/복잡하게 해서 파라미터수와 성능의 trade-off가 가능하다.

Reference

- https://arxiv.org/pdf/1807.06699.pdf