AlphaNet: Improved Training of Supernets with Alpha-divergence

Dilin Wang [1], Chengyue Gong [2], Meng Li [1], Qiang Liu [2], Vikas Chandra [1]

[1] Facebook [2] UT Austin

Code and pretrained models:
https://github.com/facebookresearch/AlphaNet

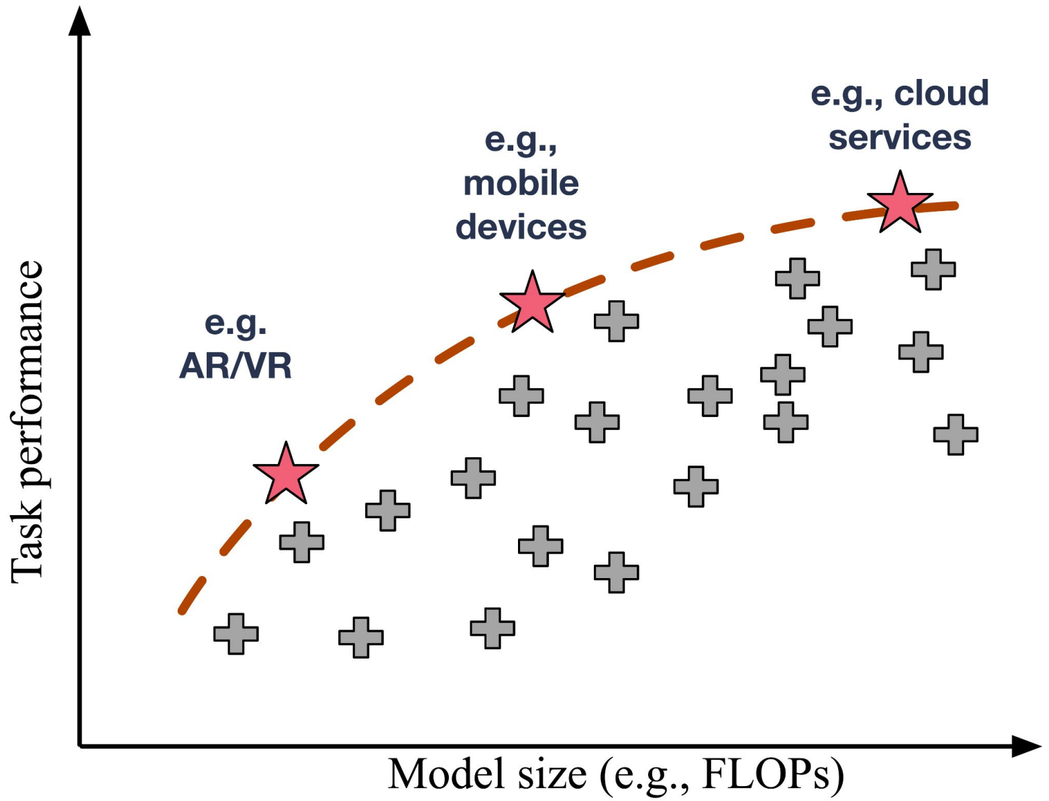
SCAN ME

facebook

TEXAS
The University of Texas at Austin

Neural architecture search

- Neural architecture search (NAS) automatically optimizes network for best accuracy given various constraints, e.g., FLOPs, latency, etc



NAS: a brief overview

Black-box optimization based NAS

- NasNet (Zoph et al., 2017)
- MnasNet (Tan et al., 2019)
- FBNetV3 (Dai et al., 2021)

	Low search cost	Simultaneously deliver a set of Pareto models	No retraining or finetuning
Black-box optimization based NAS	X	X	X

NAS: a brief overview

Black-box optimization based NAS

- NasNet (Zoph et al., 2017)
- MnasNet (Tan et al., 2019)
- FBNetV3 (Dai et al., 2021)

Continuous relaxation based NAS

- DARTS (Liu et al., 2019)
- ProxylessNAS (Cai et al., 2019)
- FBNetV2 (Wan et al., 2020)

	Low search cost	Simultaneously deliver a set of Pareto models	No retraining or finetuning
Black-box optimization based NAS	✗	✗	✗
Continuous relaxation based NAS	✓	✗	✗

Supernet based NAS (one-shot)

- A supernet assembles all the architectures as its sub-networks via weight-sharing
- Decouple NAS into two separate steps:
    1) training the supernet such that all sub-networks simultaneously reach good accuracy
    2) searching the best model given various resource constraints

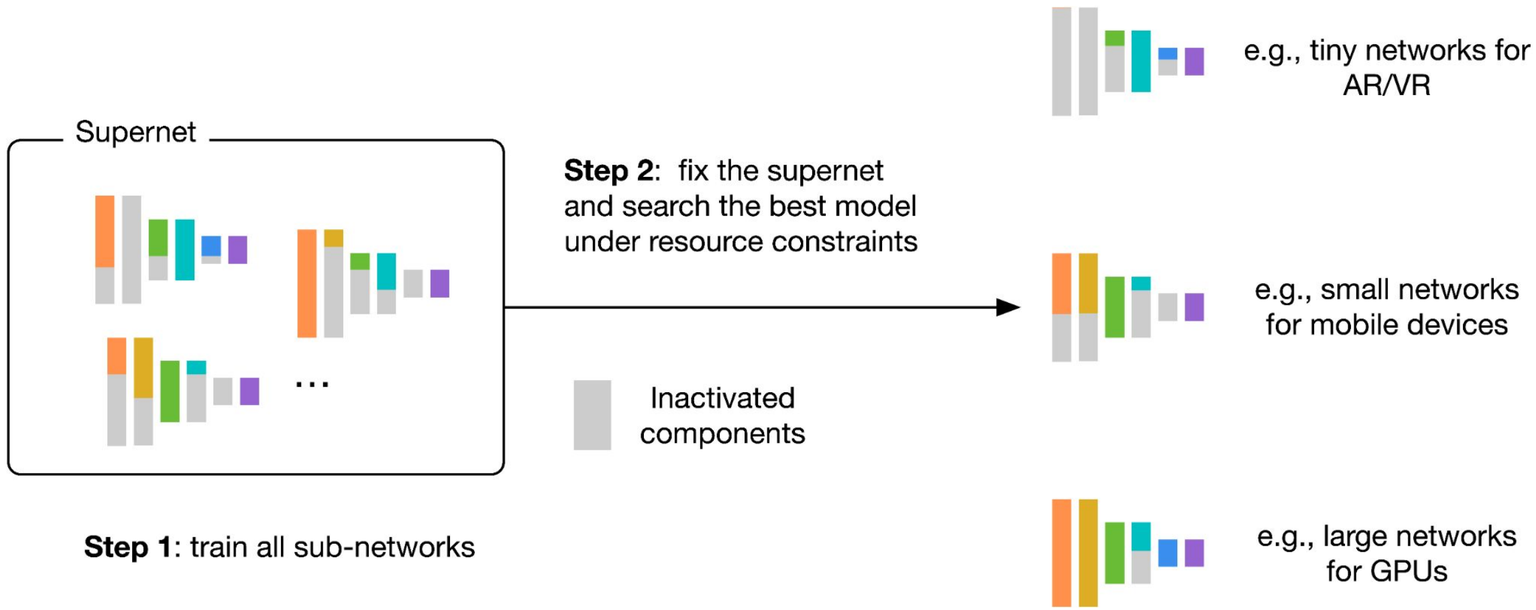


Figure: An overview of supernet based NAS.

NAS: a brief overview

Black-box optimization based NAS

- NasNet (Zoph et al., 2017)
- MnasNet (Tan et al., 2019)
- FBNetV3 (Dai et al., 2021)

Continuous relaxation based NAS

- DARTS (Liu et al., 2019)
- ProxylessNAS (Cai et al., 2019)
- FBNetV2 (Wan et al., 2020)

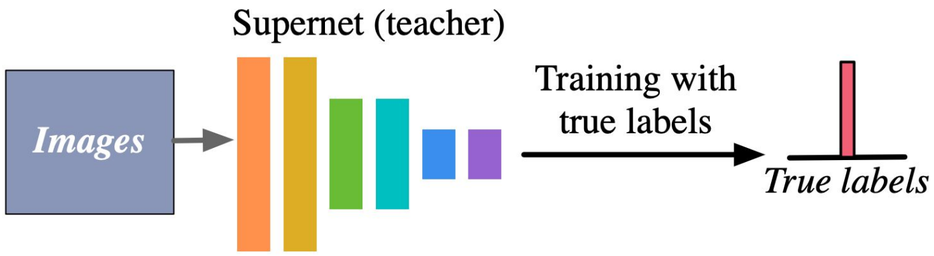
Supernet based NAS (one-shot)

- BigNAS (Yu et al., 2020)
- Once-for-all (Cai et al., 2020)
- AttentiveNAS (Wang et al., 2021)

	Low search cost	Simultaneously deliver a set of Pareto models	No retraining or finetuning
Black-box optimization based NAS	✗	✗	✗
Continuous relaxation based NAS	✓	✗	✗
One-shot supernet based NAS	✓	✓	If supernet is well trained

Supernet training with knowledge distillation

- Optimization goal: all sub-networks simultaneously reach good accuracy
- Optimization steps: for each mini-batch, 1) train the largest sub-network; 2) train k sub-networks with KD



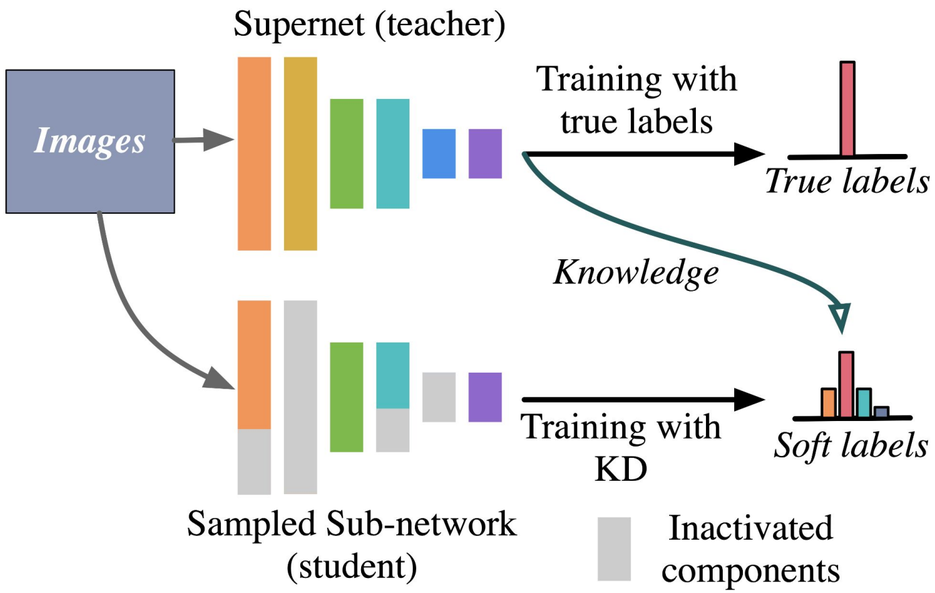
For each mini-batch

- Train the largest sub-network with ground truth labels

Figure: An illustration of training supernet with KD. Subnetworks are part of the supernet with weight-sharing. The colored part highlights the selected parameters.

Supernet training with knowledge distillation

- Optimization goal: all sub-networks simultaneously reach good accuracy
- Optimization steps: for each mini-batch, 1) train the largest sub-network; 2) train k sub-networks with KD



For each mini-batch

- Train the largest sub-network with ground truth labels
- Train `k` sub-network samples with KD

Figure: An illustration of training supernet with KD. Subnetworks are part of the supernet with weight-sharing. The colored part highlights the selected parameters.

KD is the key for good performance

- The success of supernet training heavily relies on KD.

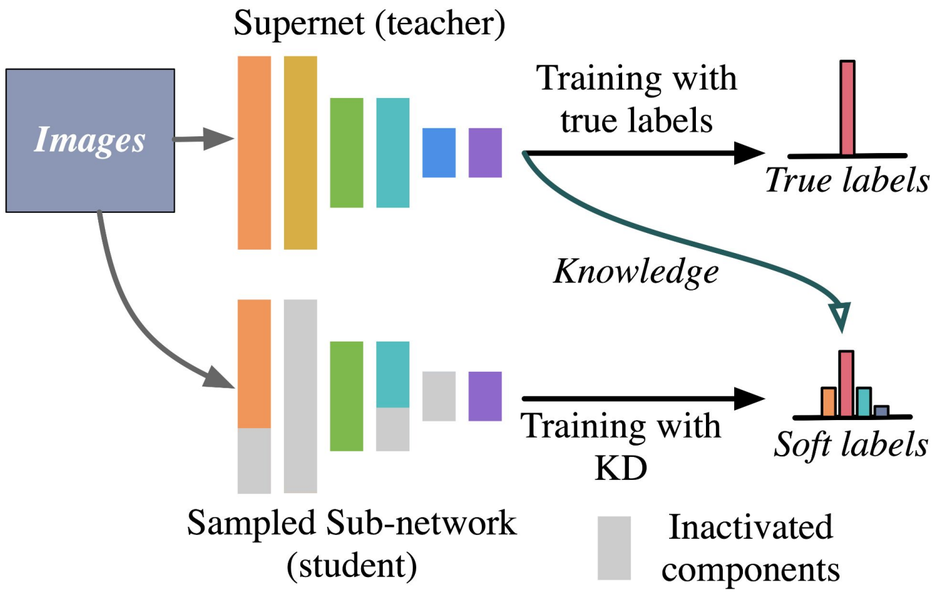
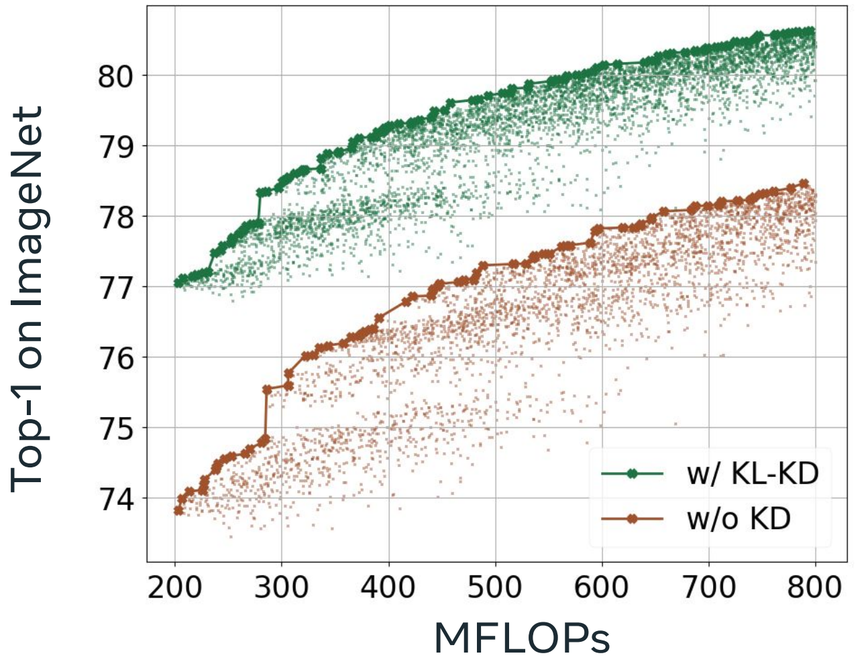


Figure: An illustration of training supernet with KD. Sub-networks are part of the supernet with weight-sharing. The colored part highlights the selected parameters.

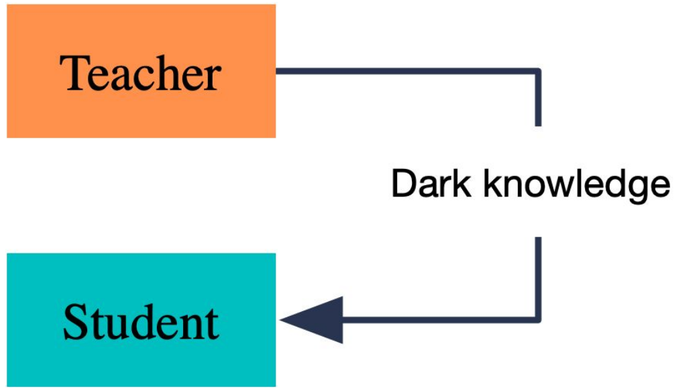


KD leads to significant improvements!

Distillation by KL minimization

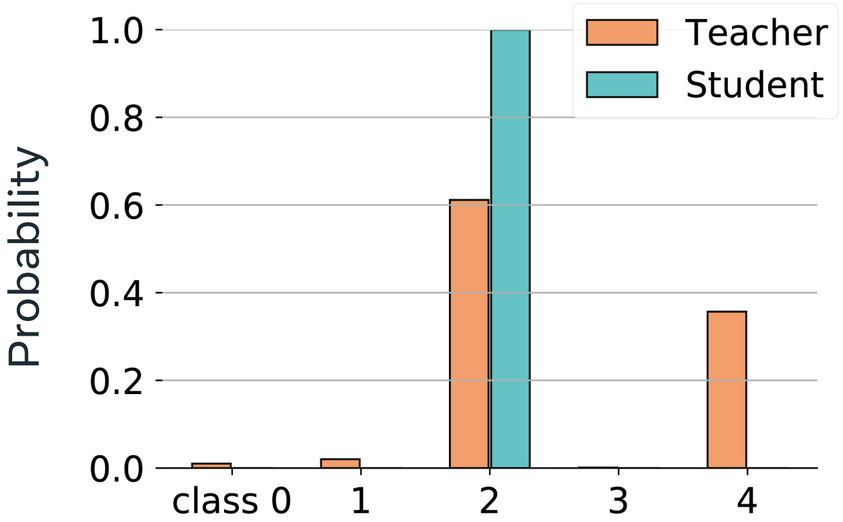
- Let $\mathtt{p}$ denote the teacher model and $\mathtt{q}$ denote the student model
- Traditional KD (e.g., Hinton et al., 2015) trains $\mathtt{q}$ by distilling knowledge from $\mathtt{p}$ via minimizing

$$\min_q \mathrm{KL}(p \parallel q) = \mathbb{E}_p \left[\log \frac{p}{q} \right]$$



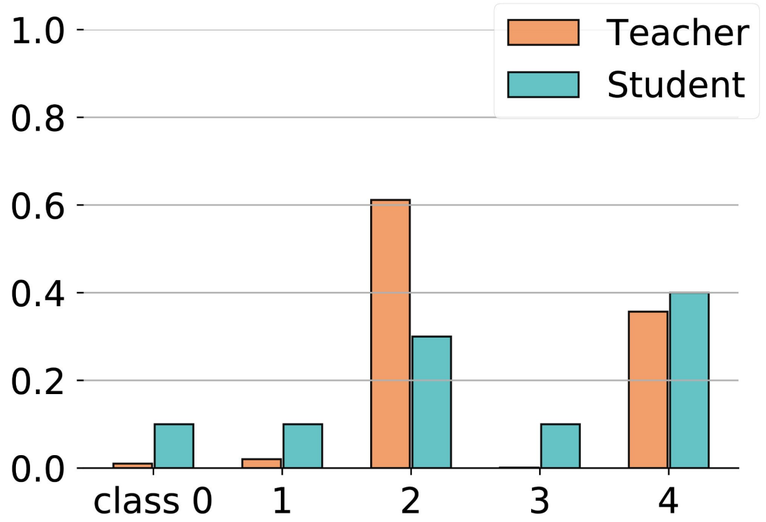
Potential failure cases of KD

Case 1: Uncertainty under-estimation



The student network under-estimates the uncertainty of the teacher model and misses important local modes of the teacher model.

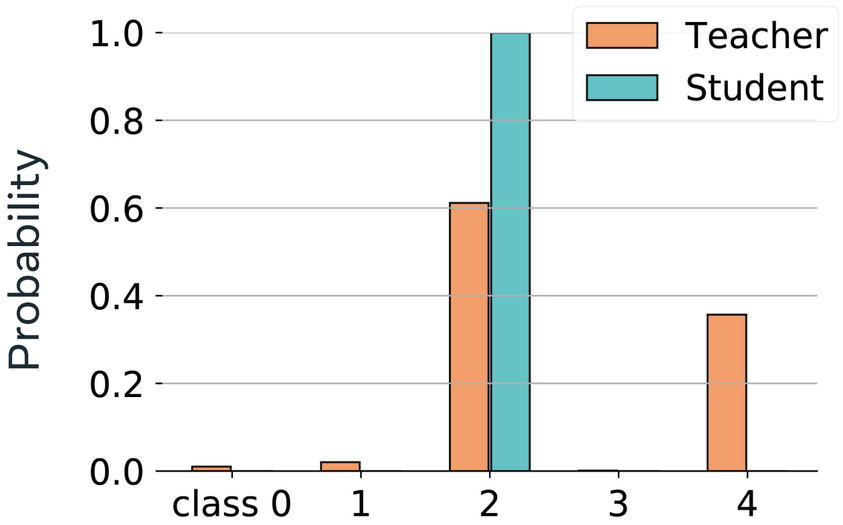
Case 2: Uncertainty over-estimation



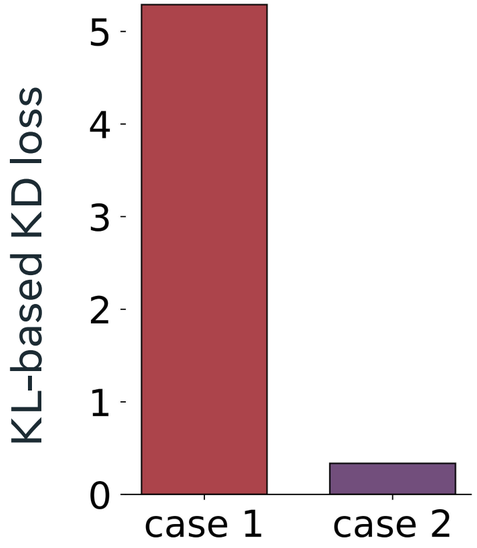
The student network over-estimates the uncertainty of the teacher model and mis-classifies the most dominant mode of the teacher model.

Limitations of KL-based KD

Case 1: Uncertainty under-estimation

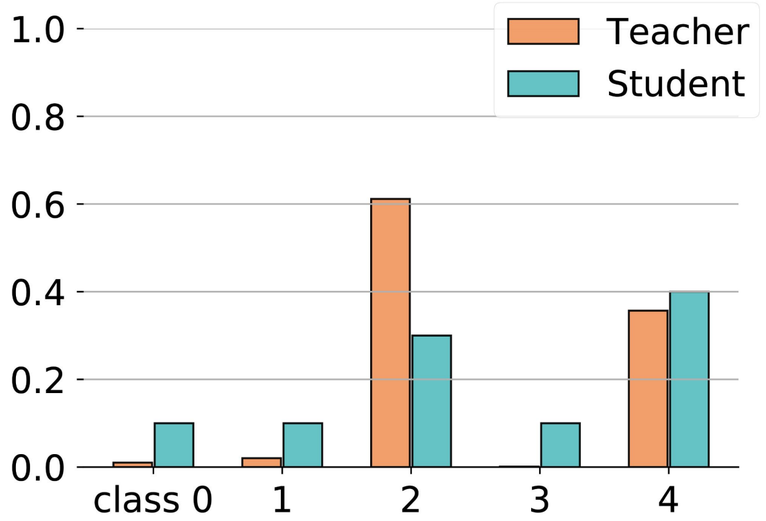


The student network under-estimates the uncertainty of the teacher model and misses important local modes of the teacher model.



KL loss penalizes less for over-estimation

Case 2: Uncertainty over-estimation



The student network over-estimates the uncertainty of the teacher model and mis-classifies the most dominant mode of the teacher model.

Generalizing KL with alpha-divergence

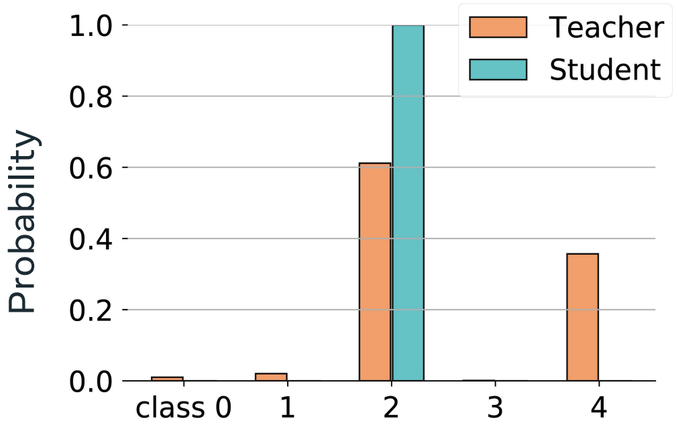
- Alpha-divergence $\alpha \in \mathbb{R}\backslash\{0,1\}$:

$$\min_q \mathrm{D}_\alpha(p \parallel q) = \frac{1}{\alpha(\alpha-1)} \mathbb{E}_q \left[\left(\frac{p}{q} \right)^\alpha - 1 \right]$$

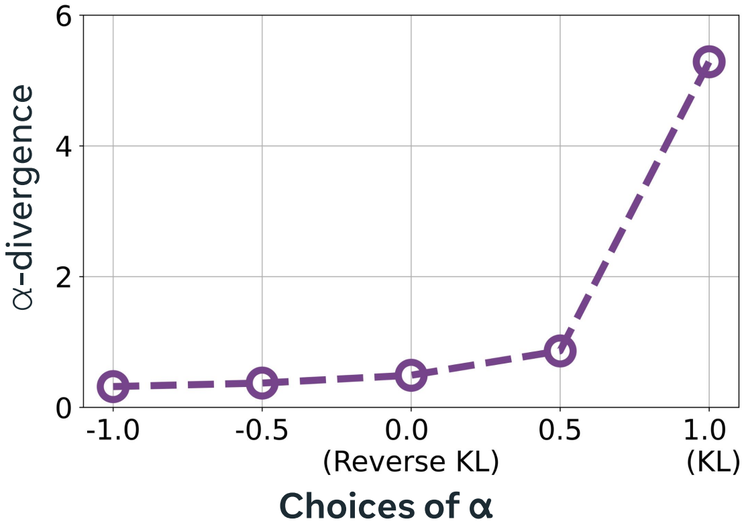
- Alpha-divergence generalizes KL divergence:

- Alpha-divergence has been widely explored in the literature [e.g., Amari, 1985; Minka et al., 2005; Hernandez-Lobato et al., 2016; Li & Turner 2016; Opper & Winther 2005; Dieng et al., 2016]

Why alpha-divergence?

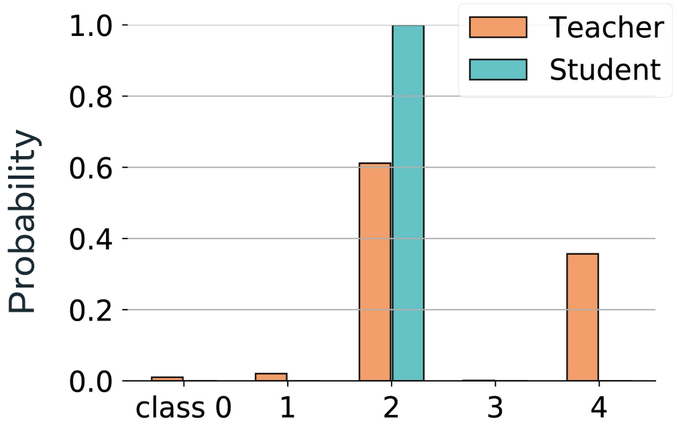


Case 1: Uncertainty under-estimation.

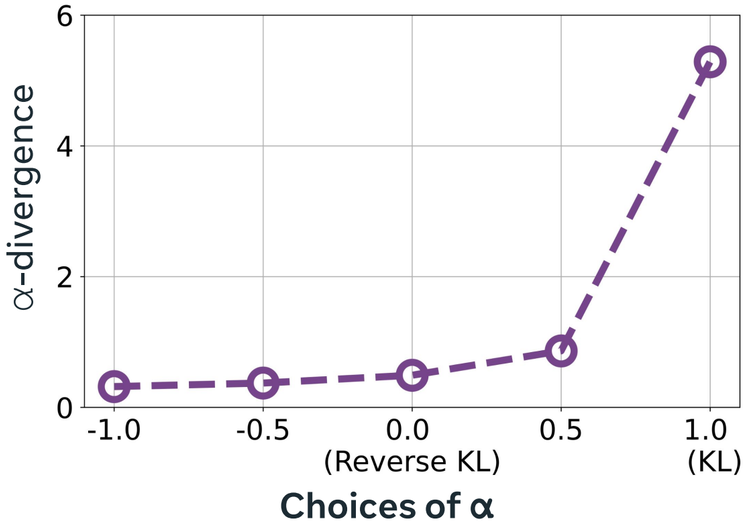


Heavily penalized by an alpha-divergence with a large and positive alpha value (e.g., $\alpha=1$)

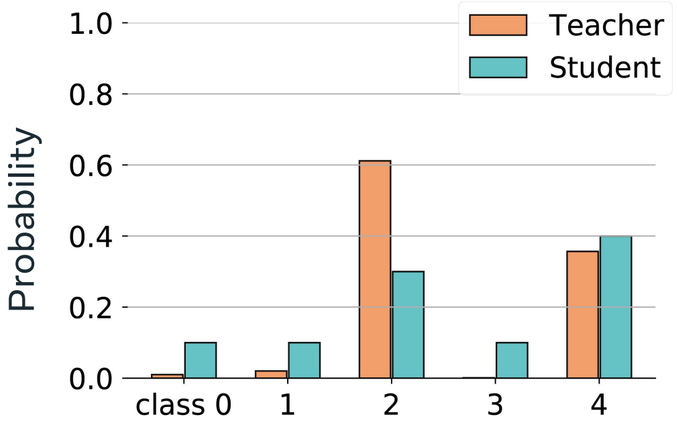
Why alpha-divergence?



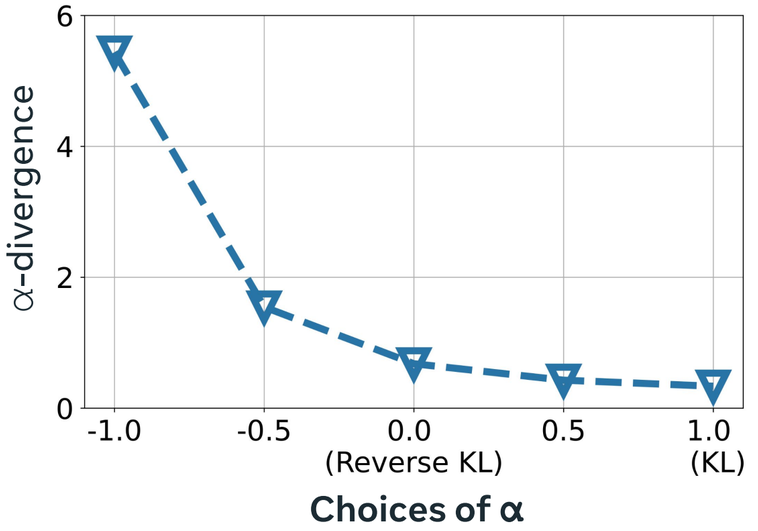
Case 1: Uncertainty under-estimation.



Heavily penalized by an alpha-divergence with a large and positive alpha value (e.g., $\alpha=1$)



Case 2: Uncertainty over-estimation.



Heavily penalized by an alpha-divergence with a small and negative alpha value (e.g., $\alpha=-1$)

Algorithm: KD with adaptive alpha-divergence

- Minimizing an adaptive alpha-divergence

$$D_{\alpha_+,\alpha_-}(p \parallel q) = \max\left\{ \underbrace{D_{\alpha_-}(p \parallel q)}_{\substack{\text{penalizing} \\ \text{over-estimation}}} , \quad \underbrace{D_{\alpha_+}(p \parallel q)}_{\substack{\text{penalizing} \\ \text{under-estimation}}} \right\}.$$

Algorithm: KD with adaptive alpha-divergence

- Minimizing an adaptive alpha-divergence

$$D_{\alpha_+,\alpha_-}(p \parallel q) = \max\left\{ \underbrace{D_{\alpha_-}(p \parallel q)}_{\substack{\text{penalizing} \\ \text{over-estimation}}}, \quad \underbrace{D_{\alpha_+}(p \parallel q)}_{\substack{\text{penalizing} \\ \text{under-estimation}}} \right\}.$$

- Gradients of alpha-divergence: hard to optimize (e.g., with $\alpha_+ = 1$ and $\alpha_- = -1$)

$$\nabla_\theta D_\alpha(p \parallel q) = -\frac{1}{\alpha}\mathbb{E}_q\left[\left(\frac{p}{q}\right)^\alpha \nabla \log q\right]$$

Algorithm: KD with adaptive alpha-divergence

- Minimizing an adaptive alpha-divergence

$$D_{\alpha_+,\alpha_-}(p \parallel q) = \max\left\{ \underbrace{D_{\alpha_-}(p \parallel q)}_{\substack{\text{penalizing} \\ \text{over-estimation}}}, \quad \underbrace{D_{\alpha_+}(p \parallel q)}_{\substack{\text{penalizing} \\ \text{under-estimation}}} \right\}.$$

- Gradients of alpha-divergence: hard to optimize (e.g., with $\alpha_+ = 1$ and $\alpha_- = -1$)

$$\nabla_\theta D_\alpha(p \parallel q) = -\frac{1}{\alpha}\mathbb{E}_q\left[\left(\frac{p}{q}\right)^\alpha \nabla \log q\right]$$

- Approximating gradients (equivalent to minimizing a f-divergence)

$$\tilde{\nabla}_\theta D_\alpha(p \parallel q) = -\frac{1}{\alpha}\mathbb{E}_q\left[\mathrm{Clip}_\beta\left(\left(\frac{p}{q}\right)^\alpha\right)\nabla \log q\right],$$

$$\text{with } \mathrm{Clip}_\beta(t) = \min(t, \beta)$$

Training supernets with adaptive KD

For each mini-batch

- Train the largest sub-network **p** with ground truth labels
- Sample k sub-networks
- **FOR** each sub-network **q**:
  - **If baseline (KL-based KD)**, minimizing:

$$\mathrm{KL}(p \parallel q)$$

  - **If adaptive KD (ours)**, minimizing:

$$D_{\alpha_+,\alpha_-}(p \parallel q)$$

Application: Slimmable neural networks

- A single model can run at different widths, allowing adaptive inference efficiency vs. accuracy tradeoffs (Yu et al., ICLR'17)

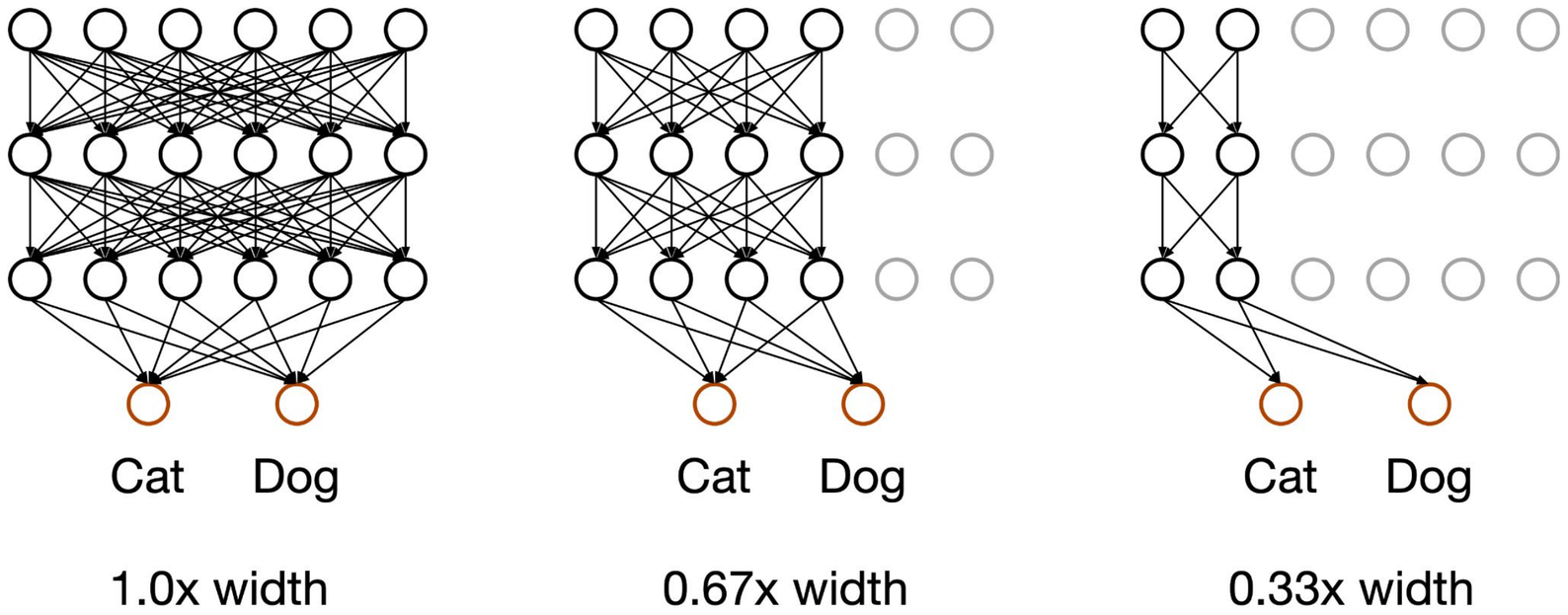


Figure: An illustration of Slimmable neural networks. This figure is adapted from Yu et al., ICLR'17.

Slimmable neural networks results

- Training with sandwich rule sampling and inplace KD (Yu et al., 2019)

Model	Method	0.25×	0.3×	0.35×	0.4×	0.45×	0.5×	0.55×	0.6×	0.65×	0.7×	0.75×
MbV1	w/o KD	53.9	55.3	57.1	59.1	61.1	62.9	64.0	65.8	66.9	67.9	68.8
	w/ KL-KD	**56.4**	57.8	59.5	61.0	63.0	64.4	65.5	67.1	68.3	69.1	69.8
	w/ Adaptive-KD (ours)	**56.4**	**57.9**	**59.7**	**61.7**	**63.4**	**65.0**	**66.2**	**67.7**	**68.8**	**69.5**	**70.1**
MbV2	w/o KD	-	-	61.9	62.8	63.7	64.5	65.1	67.2	67.7	68.3	69.0
	w/ KL-KD	-	-	63.2	64.4	65.1	66.0	66.5	68.4	69.2	69.5	70.1
	w/ Adaptive-KD (ours)	-	-	**63.7**	**64.6**	**65.6**	**66.3**	**66.9**	**68.7**	**69.3**	**69.9**	**70.5**

Table: Top-1 validation accuracy on ImageNet for Slimmable MobileNetV1 networks (denoted by MbV1) and Slimmable MobileNetV2 networks (denoted by MbV2) trained with different KD strategies.

Application: Weight-sharing search space

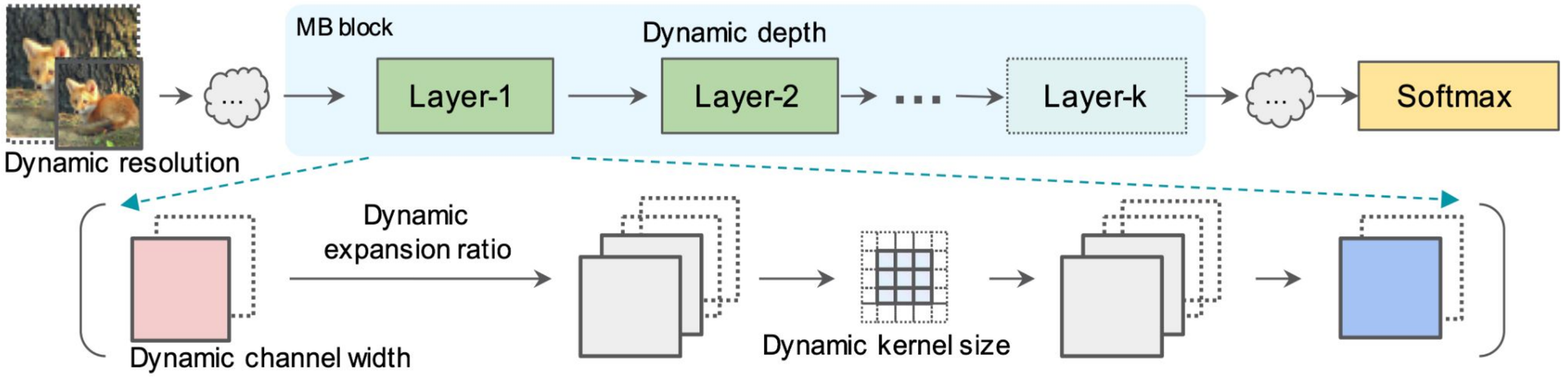
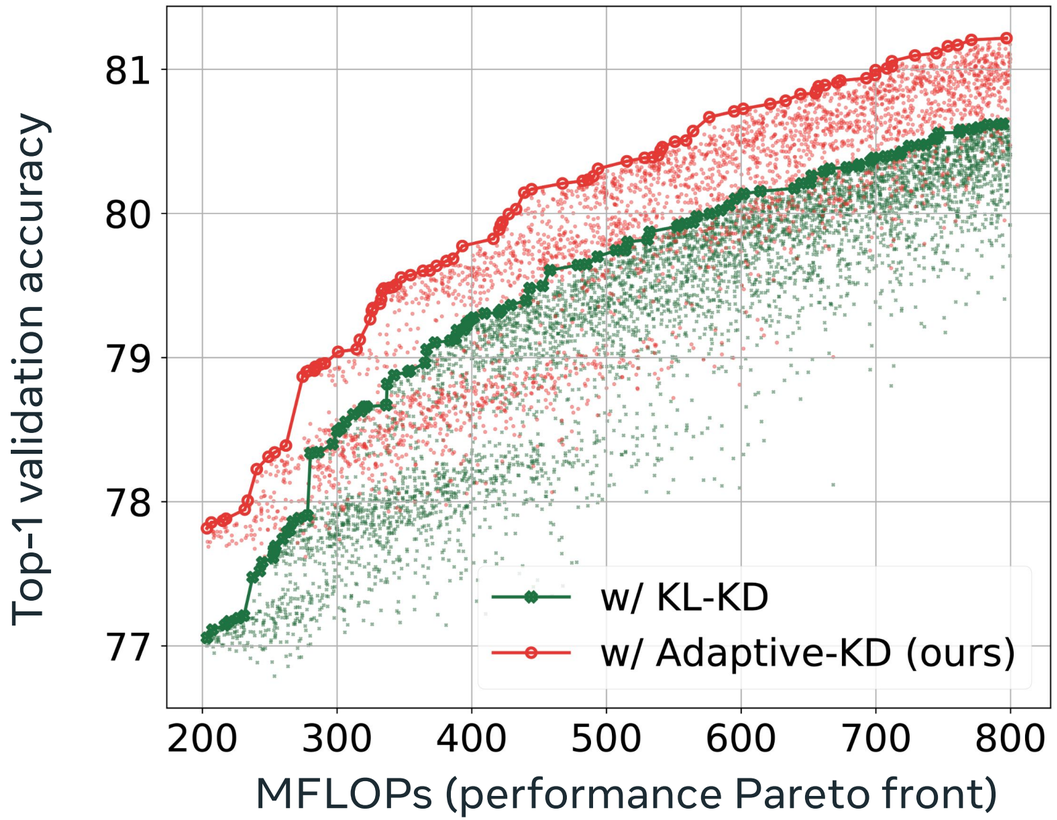


Figure: An illustration of supernets. The supernet in this figure provides a set of choices of the input resolution, channel widths, number of layers, kernel sizes, and expansion ratios. This figure is from AttentiveNAS (Wang et al., CVPR'21)

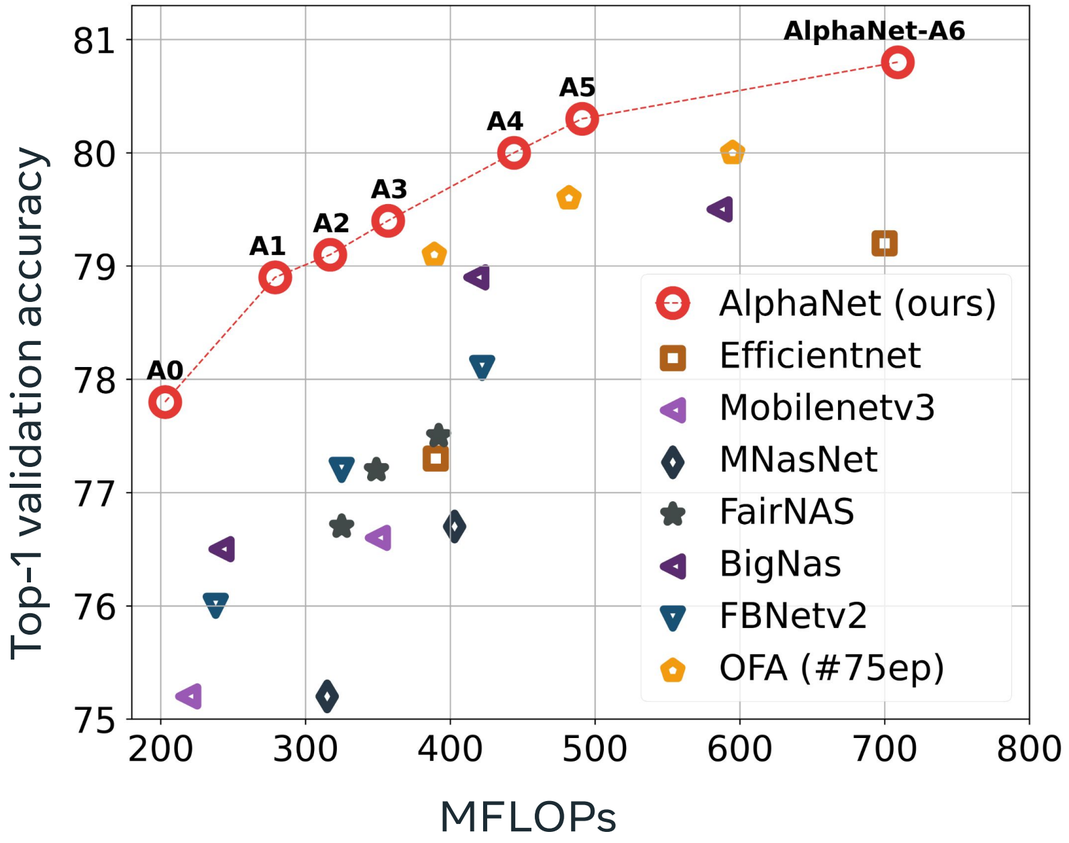
Weight-sharing NAS results

- Following AttentiveNAS (Wang et al., CVPR'21).
- First train supernets with different KD strategies; then run evolutionary search to sample from the supernets



Improvements on SOTA

- We call our model as AlphaNets
- Directly evaluating the discovered models from AttentiveNAS.



Model	MFLOPs	Top-1
AlphaNet-A0	203	77.9
AlphaNet-A1	279	78.9
AlphaNet-A2	317	79.2
AlphaNet-A3	357	79.4
AlphaNet-A4	444	80.0
AlphaNet-A5 (small)	491	80.3
AlphaNet-A5 (base)	596	80.6

Improvements on standard KD settings

- Training a single neural network with a pretrained teacher model, as in conventional KD setup

Teacher	MobileNetV1 1.0x		MobileNetV2 1.0x		RegNetY
Student	ShuffleNet 0.5x	ShuffleNet 1.0x	MobileNetV2 0.25x	MobileNetV2 0.5x	DeiT-tiny
w/ KL-KD (T=1)	60.3	69.3	54.4	65.3	74.6
Adaptive-KD (Ours)	**61.1**	**69.5**	**55.0**	**65.7**	**75.2**

Table: Additional KD results on ImageNet. Our MobileNet V1 and V2 teacher has a top-1 accuracy of 73.2% and 72.9%, respectively. All ShuffleNets (Ma et al., 2018) and MobileNetV2 models are trained for 120 epochs with standard random crop and resize data augmentation. For DeiT-tiny (Touvron et al., 2020), we exactly follow the settings of DeiT for training and use a RegNetY (Radosavovic et al., 2020) as the teacher model.

Conclusions

- Proposed an improved KD strategy to train supernets
  - Penalizing both uncertainty overestimation and underestimation
  - Easy to implement: a single setting works well for all the cases
  - Improved accuracy vs. efficiency tradeoffs on ImageNet

SCAN ME

Code and pretrained models:
https://github.com/facebookresearch/AlphaNet

facebook

TEXAS
The University of Texas at Austin