TabNAS: Rejection Sampling for Neural Architecture Search on Tabular Datasets

Google Research

Stanford University

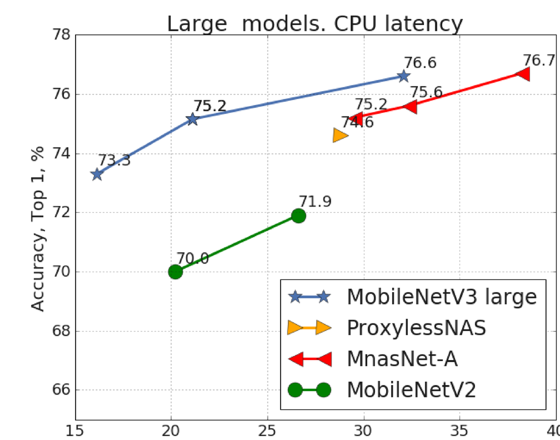
Chengrun Yang[1], Gabriel Bender[1], Hanxiao Liu[1], Pieter-Jan Kindermans[1], Madeleine Udell[2], Yifeng Lu[1], Quoc V. Le[1], Da Huang[1]

[1] Google Research, Brain Team [2] Stanford University

Neural architecture search (NAS)

People want neural networks that are **accurate** (low loss), **fast** (low latency), cheap, interpretable, fair, . . .

Neural architecture search (NAS) matters to **improve accuracy** while **meeting the latency desiderata**.



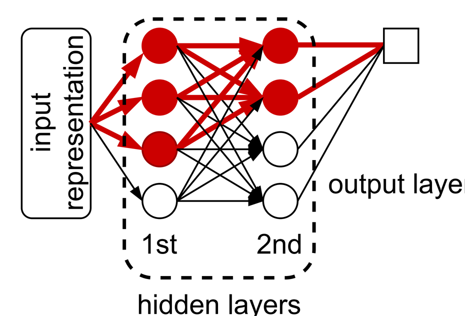
Source: Google AI MobileNet-EdgeTPU blog post

Question: How to find **the best architecture** within a user-given **resource limit**?

We do: **reinforcement learning (RL)** with **weight sharing** in a **factorized search space**

Example:

a 2-layer search space, candidate widths for each layer: $\{2, 3, 4\}$



Previous resource-aware RL rewards: With

- a resource target T_0
- a sampled architecture y, with quality reward $Q(y)$ and resource consumption $T(y)$

we have

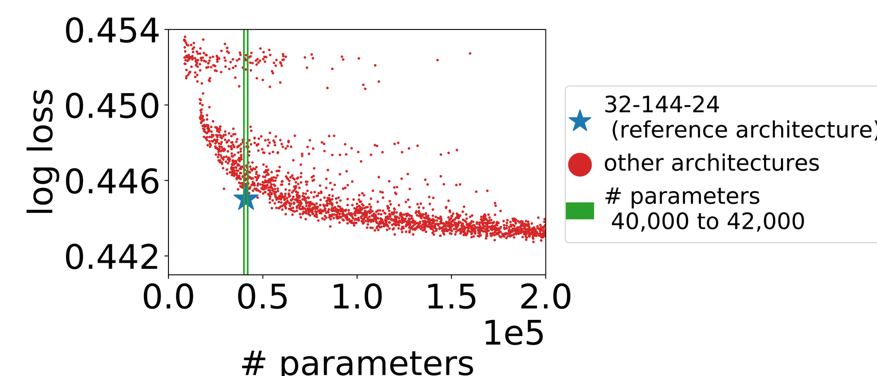
- in MnasNet [3]: $Q(y) \cdot (T(y)/T_0)^\beta$ (exponential decay over limit, $\beta < 0$)
- in TuNAS [1]: $Q(y) + \beta|T(y)/T_0 - 1|$ (absolute value reward, or **Abs Reward**. $\beta < 0$)

Tradeoff between performance and resource usage

- performance metric: loss
- resource metric: number of parameters

Example: 3-layer feedforward networks (FFNs) on the Criteo dataset

- candidate widths for each layer: $\{8, 16, 24, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256, 384, 512\}$
- embedding size: 1,027
- activation: ReLU



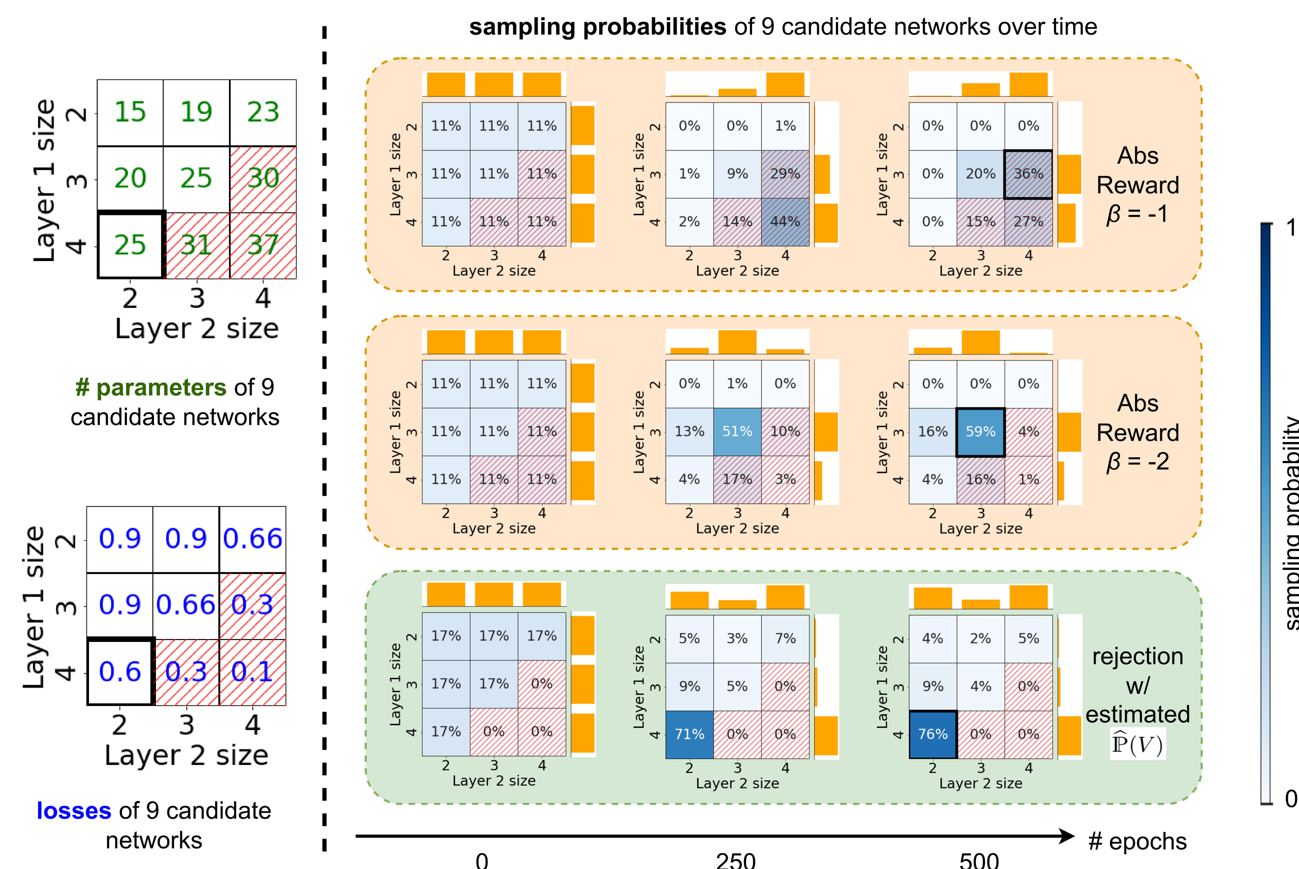
Bottleneck architectures often outperform among FFNs.

- Definition: a layer being much wider or narrower than its neighbors
- Example: 32-144-24
- Intuition for outstanding performance: the weights mimic the low-rank factors of wider networks
- Our hope for NAS: automatically determine whether to use bottlenecks, and their sizes

Our NAS setting: use a Pareto-optimal (often bottleneck) architecture as our **reference architecture** for NAS:

- The NAS controller only has information about its number of parameters.
- The NAS controller aims to find an architecture that matches its performance.

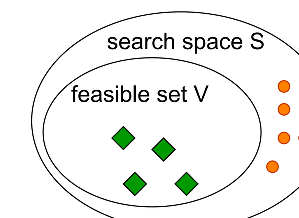
It is difficult for RL with the Abs Reward to find a bottleneck – a toy example



Our method: rejection-based reward + Monte-Carlo sampling

Part I: Use **rejection sampling** to learn a better probability distribution

- the set of feasible architectures: V
- a REINFORCE step on the logits: $\ell = \ell + \eta \nabla J(y)$ with single-step objective $J(y)$



Our algorithm: In each RL step

1. sample a child network y
2. if y is feasible:
   - compute **(or estimate)** a **differentiable** $\mathbb{P}(V)$: within S, the probability of sampling an architecture that falls in V
   - compute single-step objective with reweighted sampling probability: $J(y) = \text{stop_grad}(Q(y) - Q_{\text{avg}}) \log[\underbrace{\mathbb{P}(y)/\mathbb{P}(V)}_{\text{was } \mathbb{P}(y)}]$
3. else if y is infeasible: skip this step

Intuition: rejection sampling

- the distribution we want to sample from: $\mathbb{P}(y \mid y \in V)$, which requires **coupled** distributions across layers
- the distribution we have: layer-wise distributions $\mathbb{P}(y)$ in a **factorized** search space
- what we do: sample from $\mathbb{P}(y)$, accept and reweight it with $\mathbb{P}(V)$ when the sample y is feasible, reject otherwise

Part II: when the sample space is large, estimate $\mathbb{P}(V)$ by **Monte-Carlo sampling**

- what we want: $\widehat{\mathbb{P}}(V)$, an estimate of the differentiable $\mathbb{P}(V)$
- what we have: candidate architectures, each with a sampling probability
- what we do: sample from a proposal distribution q for N times, get an estimate

$$\widehat{\mathbb{P}}(V) = \frac{1}{N} \sum_{k \in [N]} \frac{p^{(k)}}{q^{(k)}} \cdot \mathbb{1}(z^{(k)} \in V)$$

In theory:

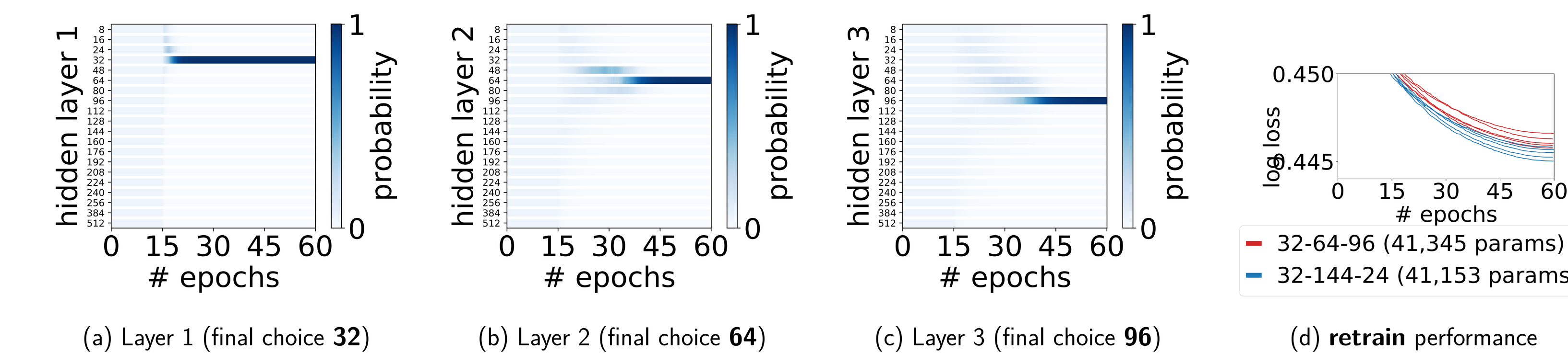
- $\widehat{\mathbb{P}}(V)$ is an unbiased and consistent estimate of $\mathbb{P}(V)$
- $\nabla \log[\mathbb{P}(y)/\widehat{\mathbb{P}}(V)]$ is a consistent estimate of $\nabla \log[\mathbb{P}(y \mid y \in V)]$

In practice:

- for simplicity: **set** $q =$ **stop_grad**(p), i.e. sample with the current distribution p
- to get an accurate estimate: have **a large enough** N

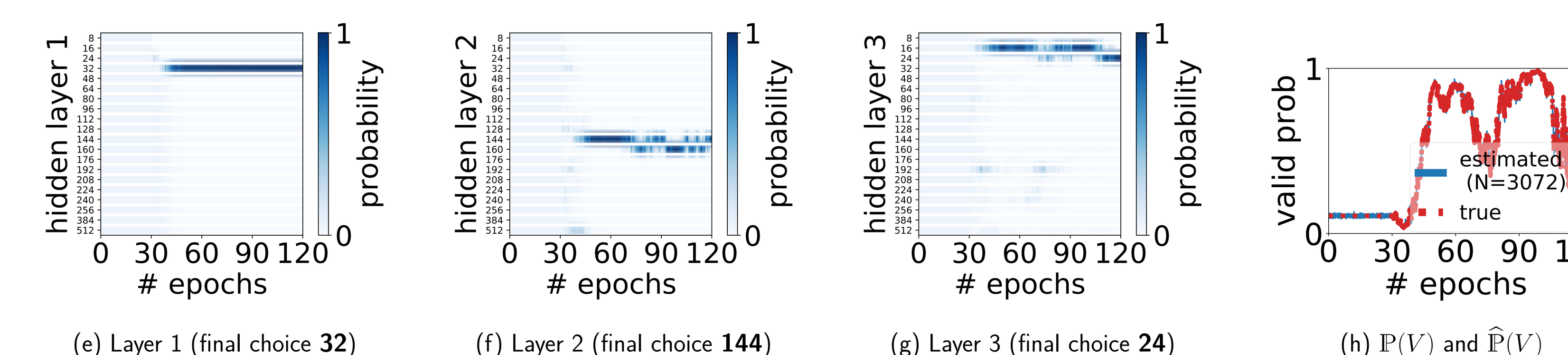
Experiments (more in paper!)

The **failure of Abs Reward** on Criteo, in the 3-layer search space with reference architecture 32-144-24:



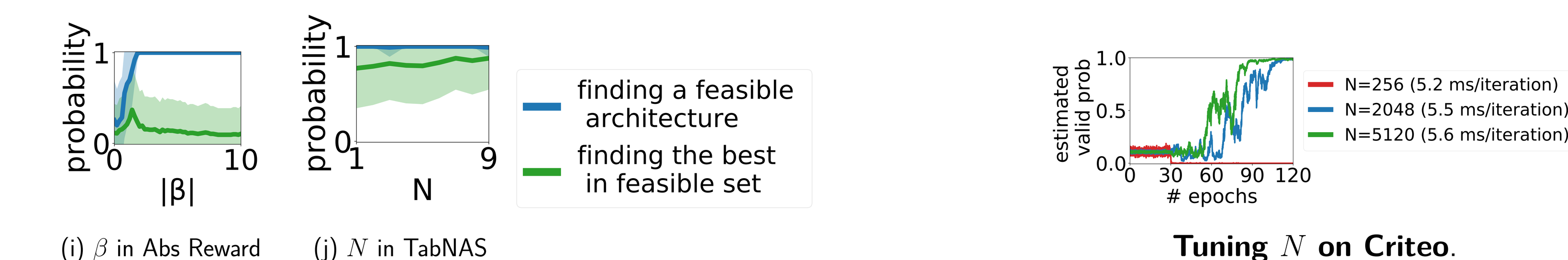
(a) Layer 1 (final choice **32**) (b) Layer 2 (final choice **64**) (c) Layer 3 (final choice **96**) (d) **retrain** performance

The **success of the rejection-based reward** on Criteo in the same setting:



(e) Layer 1 (final choice **32**) (f) Layer 2 (final choice **144**) (g) Layer 3 (final choice **24**) (h) $\mathbb{P}(V)$ and $\widehat{\mathbb{P}}(V)$

TabNAS has **easier-to-tune hyperparameters**:



(i) β in Abs Reward (j) N in TabNAS

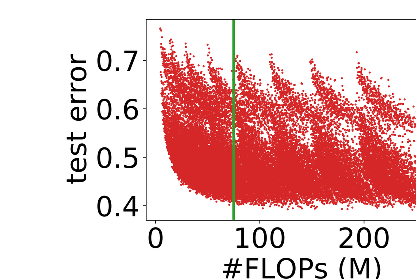
Tuning β and N on the toy example: the number of Monte-Carlo samples N in rejection-based reward is easier to tune than coefficient β in Abs Reward, and is easier to succeed.

Tuning N on Criteo.

On a **vision** task – NATS-Bench [2] channel size search space on CIFAR-100, with a 75M #FLOPs limit:

Table 1: **#FLOPs (M)** of architectures found by RL with the rejection-based reward and the Abs Reward.

RL learning rate	0.01	0.05	0.1	0.5
rejection-based reward, N=200	74.7 ± 0.3 (median 74.7)	74.7 ± 0.2 (median 74.7)	74.6 ± 0.3 (median 74.7)	70.7 ± 4.5 (median 72.4)
Abs Reward, β=-10	77.0 ± 12.7 (median 76.4)	75.1 ± 4.5 (median 74.8)	75.2 ± 1.7 (median 75.1)	75.7 ± 3.8 (median 75.2)
Abs Reward, β=-5	78.1 ± 13.1 (median 77.0)	75.6 ± 4.3 (median 75.5)	75.4 ± 1.7 (median 75.2)	76.0 ± 4.4 (median 75.3)
Abs Reward, β=-1	83.0 ± 13.3 (median 81.9)	77.4 ± 4.4 (median 77.1)	76.0 ± 2.1 (median 75.8)	76.7 ± 5.3 (median 75.6)
Abs Reward, β=-0.5	87.1 ± 12.7 (median 86.7)	78.8 ± 4.5 (median 78.7)	77.0 ± 2.6 (median 76.6)	76.9 ± 6.0 (median 76.1)
Abs Reward, β=-0.1	101.8 ± 13.3 (median 103.0)	84.2 ± 7.6 (median 83.0)	81.1 ± 5.8 (median 80.5)	80.1 ± 10.6 (median 78.6)



Data source: NATS-Bench

Table 2: **Test errors** of architectures found by RL with the rejection-based reward and the Abs Reward.

RL learning rate	0.01	0.05	0.1	0.5
rejection-based reward, N=200	0.468 ± 0.050	0.435 ± 0.023	0.425 ± 0.014	0.420 ± 0.008
Abs Reward, β=-10	0.483 ± 0.056	0.468 ± 0.034	0.473 ± 0.046	0.490 ± 0.074
Abs Reward, β=-5	0.474 ± 0.049	0.461 ± 0.029	0.463 ± 0.039	0.481 ± 0.061
Abs Reward, β=-1	0.449 ± 0.027	0.442 ± 0.016	0.445 ± 0.020	0.456 ± 0.035
Abs Reward, β=-0.5	0.436 ± 0.019	0.434 ± 0.014	0.435 ± 0.015	0.444 ± 0.023
Abs Reward, β=-0.1	0.414 ± 0.009	0.416 ± 0.008	0.418 ± 0.009	0.426 ± 0.014

More experiments in paper: **tradeoffs** between loss and number of parameters **in more search spaces**; performance of RL-based NAS with **more reward functions**; **ablation** studies; **comparison with Bayesian optimization and evolutionary search** in one-shot NAS

- Paper: `https://arxiv.org/abs/2204.07615`
- Contact: Chengrun Yang (`chengrun@google.com`)

Bibliography

[1] G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P.-J. Kindermans, and Q. V. Le. Can weight sharing outperform random architecture search? An investigation with TuNAS. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14323–14332, 2020.

[2] X. Dong, L. Liu, K. Musial, and B. Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[3] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.