TabNAS: Rejection Sampling for Neural Architecture Search on Tabular Datasets

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

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

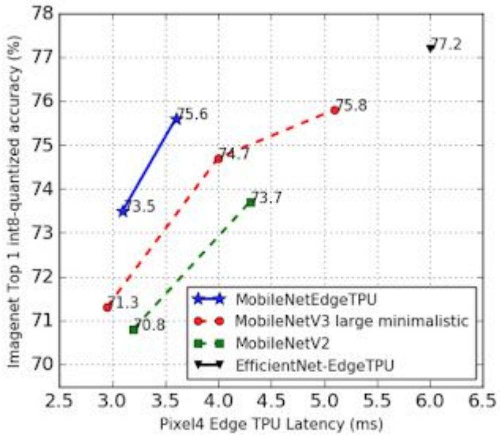
NeurIPS 2022 Poster #54518

Neural architecture search (NAS)

People want neural networks that are …

- accurate: low loss
- fast: low latency
- cheap: low power or memory usage
- interpretable
- fair
- …

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



Source: MobileNet-EdgeTPU blog post

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

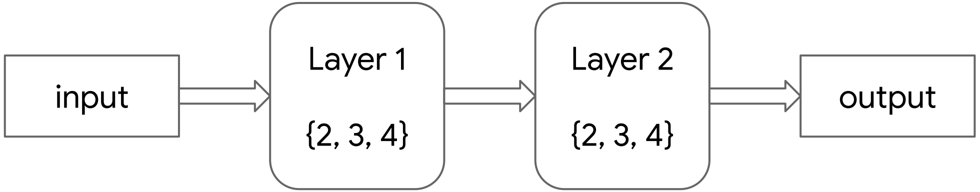
number of parameters, #FLOPs, latency, …

Our NAS on tabular datasets

- candidate choices: the number of units in each hidden layer
- **bottleneck structures** are critical to get good tradeoffs between network size and quality
  - Definition: a layer being much wider or narrower than its neighbors
  - Example: 48-**240**-24-**256**-8
  - Intuition for outstanding performance: the weights mimic the low-rank factors of wider networks

Factorized search space in weight-sharing NAS

- "Factorized": learn a separate distribution for each search component



- benefit: reduce the size of the RL action space from product to sum
- pitfall: ?

Previous works: resource-aware RL rewards

With a sampled architecture y with quality reward $Q(y)$ and resource consumption $T(y)$, and resource target $T0$, previously proposed resource-aware rewards:

- **MnasNet** [1]: making an architecture cheaper always improves its reward
  - $Q(y) * (T(y) / T0) \wedge \beta$
  - $Q(y) * \max\{1, (T(y) / T0) \wedge \beta\}$
- **Absolute Value Reward** in **TuNAS** [2]: prefer architectures with resource consumption close to our target

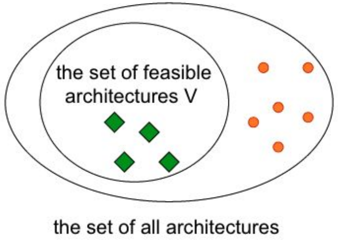
  $Q(y) + \beta * |T(y) / T0 - 1|$

in which $\beta < 0$, and we tune its absolute value.

[1] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. CVPR 2019.

[2] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, Quoc Le. Can weight sharing outperform random architecture search? An investigation with TuNAS. CVPR 2020.

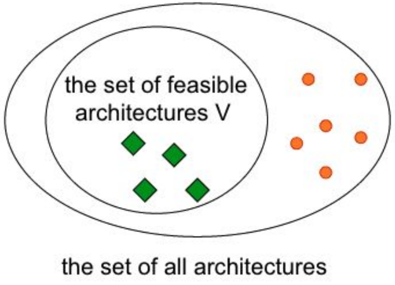
Intuition for the failure of resource-aware rewards



- With a feasible set *V*, we only want to sample among feasible architectures, in which **feasibility is determined by all layers**.
- However, in the factorized search space, **we learn a separate distribution for the choices of each layer**.

  => **Co-adaptation** makes it difficult to sample large layer sizes and thus choose a bottleneck structure.

We propose: rejection-based reward

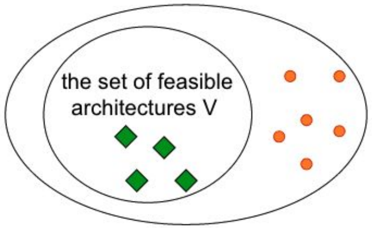


- the set of feasible architectures: V
- one step of the REINFORCE update: $\ell = \ell + \eta * \nabla J(y)$
- algorithm: In each RL step
  - sample a child network y
  - if y is feasible:
    - compute (or estimate) a **differentiable** $\mathbb{P}(V)$: the probability of sampling an architecture in V
    - single-step objective: $J(y)$ = stop_gradient[$Q(y)$ - Q_avg] * log ($P(y)$ / $P(V)$)

P(y) in previous works

  - else if y is infeasible: skip this step
- intuition: **rejection sampling**
  - we want to sample from: $P(y \mid y \in V)$, which requires coupled distributions across layers
  - we have: layer-wise distributions $P(y)$ in a factorized search space
  - **what we do: sample from P(*y*), accept when the sampled architecture *y* is feasible, reject otherwise**

When the sample space is large: estimate 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, obtain 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 $\nabla \log[\mathbb{P}(y)/\mathbb{P}(V)]$.

In experiments:

- 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***.

more contents in paper, including:

- performance on **real tabular (and vision!) datasets**
- **ablation** studies
- analysis on the difficulty of **hyperparameter tuning**
- comparison with Bayesian optimization and evolutionary search in our setting

Open questions: can TabNAS

- find better architectures in **more domains**?
- improve RL results for **more complex architectures**?
- be useful for **other resource-constrained RL problems**?

Thanks!

TabNAS: Rejection Sampling for Neural Architecture Search on Tabular Datasets

Poster ID 54518