NADS: Neural Architecture Distribution Search for Uncertainty Awareness

Chen Zhao

ICML 2020 (`https://arxiv.org/pdf/2006.06646.pdf`)

1 Motivation

- The widespread use of deep learning models within autonomous systems have aggravated the issue of detection abnormal inputs.
- Existing OoD detection approaches are prone to errors and even sometimes assign higher likelihoods to OoD samples.
- Unlike common machine learning tasks such as image classification, segmentation, and speech recognition, there are currently no well established guidelines for designing architectures that can accurately screen out OoD data and quantify its predictive uncertainty. Such a gap makes Neural Architecture Search (NAS) a promising option to explore the better design of uncertainty-aware models.
- Existing efforts have primarily focused on searching for architectures that perform well on image classification or segmentation.However, it is unclear whether architecture components that are beneficial for image classification and segmentation models would also lead to better uncertainty quantification (UQ) and thereafter be effective for OoD detection.
- NADS is proposed to identify common building blocks that naturally incorporate model uncertainty quantification and compose good OoD detection models, which searches for a distribution of well-performing architectures, instead of a single best architecture, by formulating the architecture search problem as a stochastic optimization problem.

2 The Framework

For simplicity (see Figure 2), in a neural network, suppose we only have one hidden layer and hence there is only one operation which can be chosen from K candidate operations. Each operation $i \in \{1....K\}$ has a weight ϕ_i which is needed to be learned from the model. X denotes the input feature vector and $y = P(X)$ is the density function. Besides, we add a softmax function (*i.e.* $\sigma(\cdot)$) to normalize the probability vector $[\phi_1, ..., \phi_k]$.

In this simplified case, $\alpha = \{0, 1\}^k$ is a dummy binary vector of size k. This means, only one α_i equals 1 and rests are zeros. α is sampled from multi-normal distribution based on the probability vector $\sigma[\phi_1, ..., \phi_k]$, where $P_\phi(\alpha)$ is the probability density function. In the paper, architectures α are encoded by b which is also a binary vector. Then y is calculated as $y = \sum b_i o_i(x)$. See Figure 1 for more details.

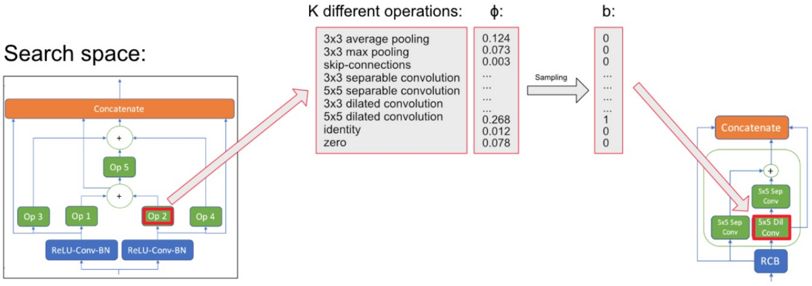

Figure 1:

In this paper, authors seek architectures that can perform entropy estimation by maximizing the Widely Applicable Information Criteria (WAIC) of the training data. The WAIC score is a Bayesian adjusted metric to calculate the marginal likelihood.

$$WAIC(X) = \mathbb{E}_{\alpha\sim p(\alpha)}[\mathbb{E}_{p(x)}[\log p(x|\alpha)]] - \mathbb{V}_{\alpha\sim p(\alpha)}[\mathbb{E}_{p(x)}[\log p(x|\alpha)]] \tag{1}$$

where $\mathbb{E}[\cdot]$ and $\mathbb{V}[\cdot]$ denote expectation and variance respectively.

As several difficulties arise when naively attempting to optimize this setup, this problem is alleviated by approximating the WAIC objective through Monte Carlo sampling. Specifically, we can sample M architectures from $P_\phi(\alpha)$ and approximate the WAIC score expectation and variance terms as

$$WAIC(X) \approx \sum_{i=1}^{N}(\sum_{j=1}^{M}\log p(x_i|\alpha_j) - (\sum_{j=1}^{M}(\log p(x_i|\alpha_j)^2) - (\sum_{j=1}^{M}\log p(x_i|\alpha_j))^2)) \tag{2}$$

To make optimization tractable, the discrete mask b is relaxed to be a continuous random variable $\tilde{b}$ using the Gumbel-Softmax reparameterization.

$$\tilde{b}_i = \frac{\exp((\log(\phi_i) + g_i)/\tau)}{\sum_{j=1}^{k}\exp((\log(\phi_i) + g_i)/\tau)} \text{ for } i = 1, ..., K \tag{3}$$

Here, $g_1 ... g_k \sim -\log(-\log(u))$ where $u \sim \text{Unif}(0, 1)$, and τ is a temperature parameter.

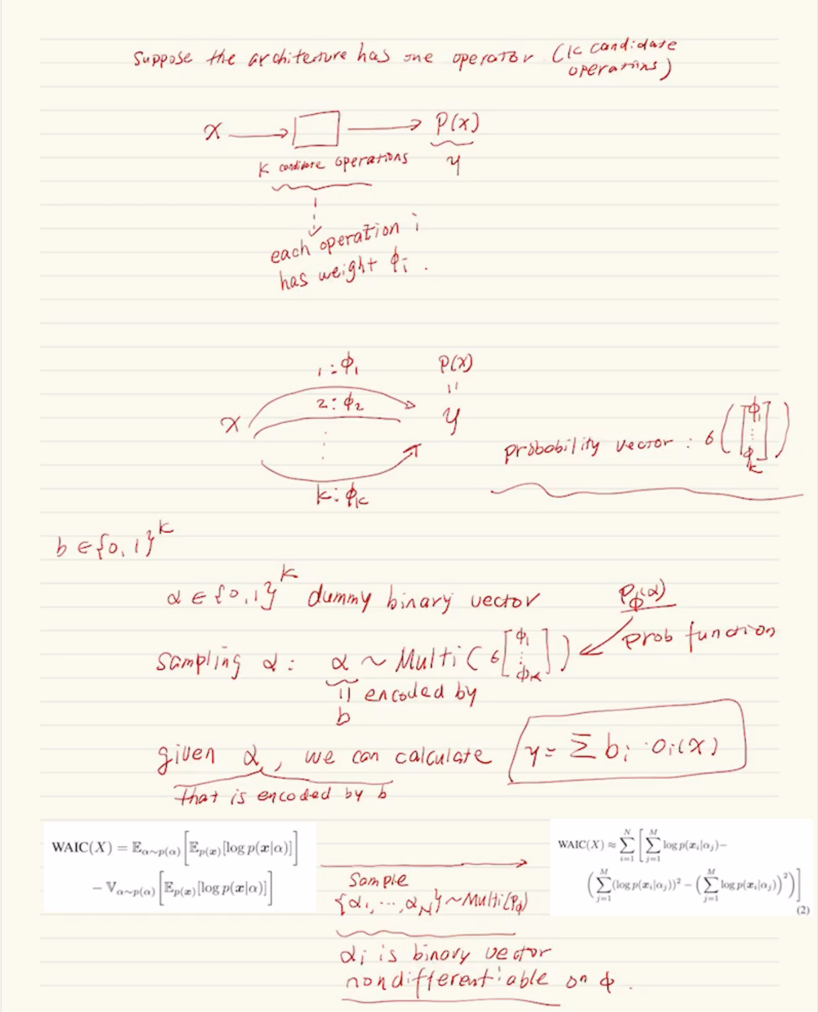
Suppose the architecture has one operator (k candidate operations)
x
P(x)
y
k candidate operations
each operation i has weight φ_i.
1: φ_1
2: φ_2
k: φ_k
P(x)
y
probability vector : σ([φ_1 ... φ_k])
b ∈ {0, 1}^k
α ∈ {0, 1}^k dummy binary vector
P_φ(α)
prob function
Sampling α : α ~ Multi(σ[φ_1 ... φ_K])
encoded by b
given α, we can calculate y = Σ b_i · O_i(x)
That is encoded by b
WAIC(X) = E_{α∼p(α)}[E_{p(x)}[log p(x|α)]] − V_{α∼p(α)}[E_{p(x)}[log p(x|α)]]
Sample {α_1, ..., α_N} ~ Multi(P_φ)
WAIC(X) ≈ Σ_{i=1}^{N}[Σ_{j=1}^{M} log p(x_i|α_j)− (Σ_{j=1}^{M}(log p(x_i|α_j))² − (Σ_{j=1}^{M} log p(x_i|α_j))²)]
(2)
α_i is binary vector
nondifferentiable on φ.

Figure 2: Overall Idea of NADS