NAS Spaces contain a vast number of architectures.

It is intractable to consider all candidate architectures, necessitating search.

...what if, *instead* of evaluating **whole architectures**, we quantified the importance of small modules – the *building blocks* of neural networks?

We then select a small number of quality modules and *automatically build* high-performance architectures.

Building Optimal Neural Architectures using Interpretable Knowledge

CVPR

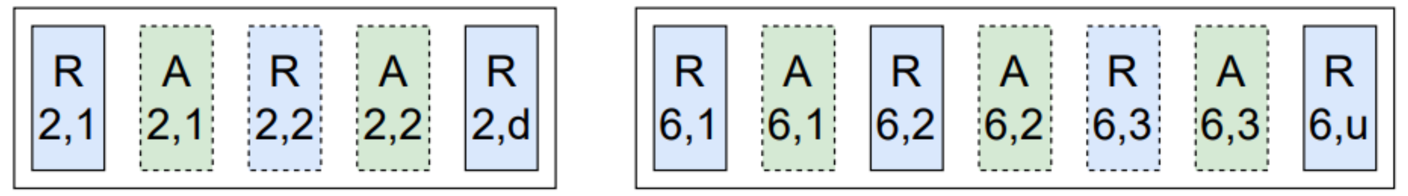
Keith G. Mills[1,2], Fred X. Han[2], Mohammad Salameh[2], Shengyao Lu[1], Chunhua Zhou[3], Jiao He[3], Fengyu Sun[3] and Di Niu[1]

[1]Dept. ECE, University of Alberta, [2]Huawei Technologies Canada, [3]Huawei Kirin Solution, Shanghai, China

Quantifying Arch. Module Importance

What are architecture modules?

- Individual operations like nn.Conv2d/nn.Linear.
- Structures like ResNet/Transformer blocks.
- Larger sequences like network *stages*.

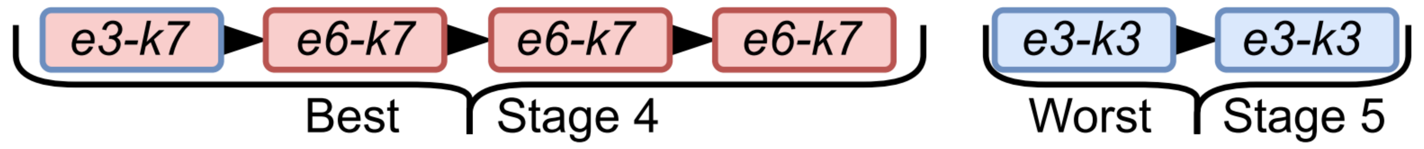


Why is this important?

- Architectures are combinations of modules.
- There are fewer modules than architectures.

The key idea!

- Limiting the number of modules severely restricts the search space size.
- Base restriction on module quality, e.g., how a module contributes to desirable architectures.

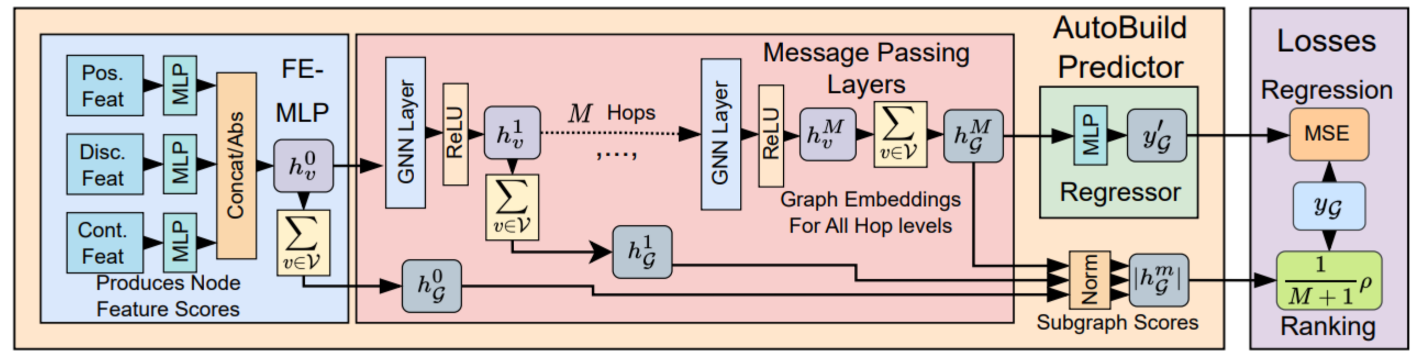


(a) $y = 100^{Acc}$

(b) $y = 100^{Acc}/log_{10}(Lat)$

Why is this difficult?

- Modules do not exist in a vacuum.
- They combine to form architectures.
- Typically, we only have access to end-to-end architecture metrics, e.g., accuracy/latency.



Magnitude Ranked Embedding

Preliminary: We cast architectures as *graphs* G.

E.g., data format, $(arch, perf) = (G_1, y_1)$

Train a GNN predictor; $y'_1 = GNN(G_1)$

➢ Node embeddings $\forall v \in V_G$, there is h_v

➢ Graph embedding: $h_G = \frac{1}{|V_G|}\sum_{v \in V_G} h_v$

Key learning constraint:

$$\text{if } y_1 > y_2 \text{, then } \|h_{G_1}\|_1 > \|h_{G_2}\|_1$$

In plain language:

- Architectures with better performance have higher embedding norms.

Intuition:

- Graph embed from node embeddings.
- Force GNN to learn in a specific way.
- Identify nodes that lead to high performance!
- Node embeddings = module scores!

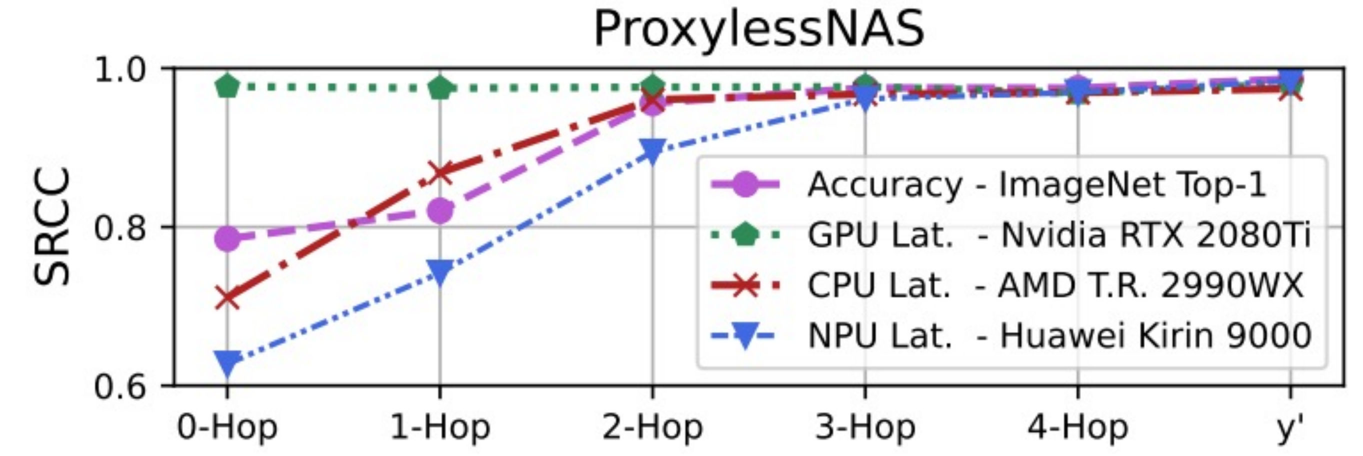
How to enforce?

- Differentiable Spearman ρ loss, per batch.

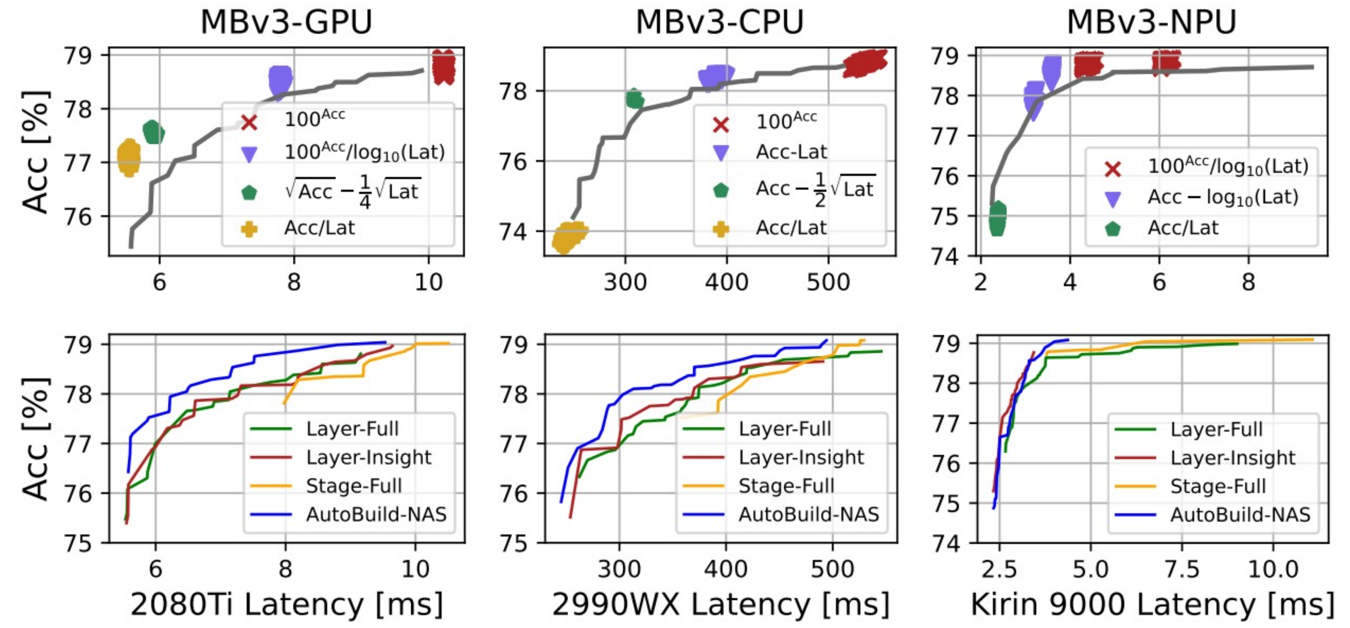
$$Loss = 1 - \rho(y_G\ \|h_G\|_1)$$

Experimental Results

Efficacy of the hop-level ranking loss.



Application to MobileNets & augmenting NAS.



High-quality SDv1.4 architecture with <100 samples

Arch Set	Eval Archs (68)	Exhaustive Search (4)	AutoBuild (4)
Ave. FID	22.13	10.82	**10.13**
Best FID	10.54	10.29	**9.96**

SDv1.4 Inpainting Visual Examples

(a) Original (b) ES (c) AutoBuild