Adaptive Regularizing Tucker Decomposition for Knowledge Graph Completion

[1]Quanming YAO, [1,2]Shimin DI, [1]Yongqi ZHANG

[1] *4Paradigm Inc.*

[2]*The Hong Kong University of Science and Technology*

Outline

- Background
- Related Work
- Problem Formulation
- Search Algorithm
- Experiments

Background

Knowledge Graph

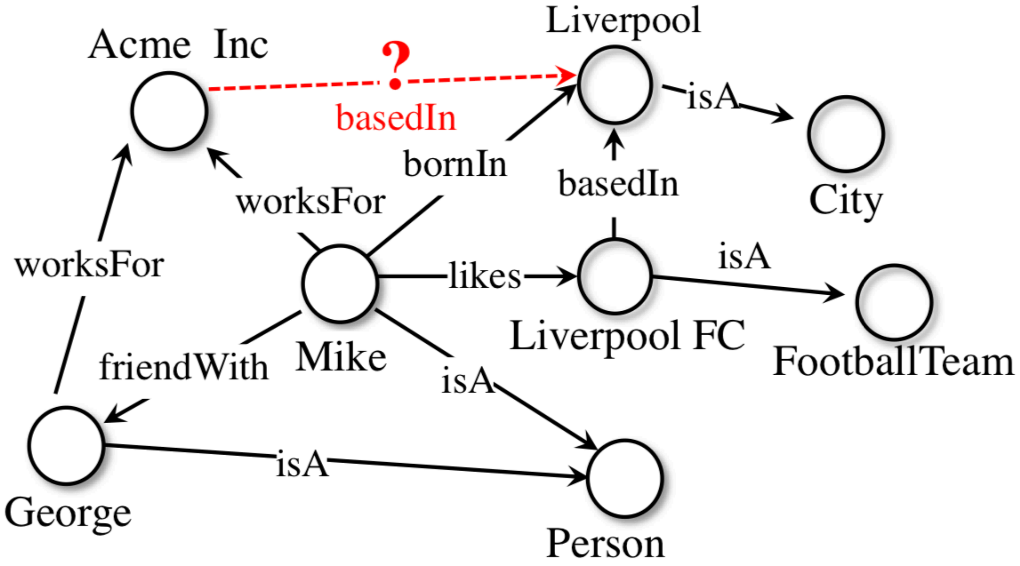
- Knowledge Graph (KG) $G = (E, R)$
  - Each node = an entity $e \in E$
  - Each edge = a relation $r \in R$

- Fact (i.e., triplet)
  - (*head entity, relation, tail entity*) or (h, r, t)

- Knowledge graph completion (KGC)
  - (Acme Inc, basedIn, ?)

Popular KGs

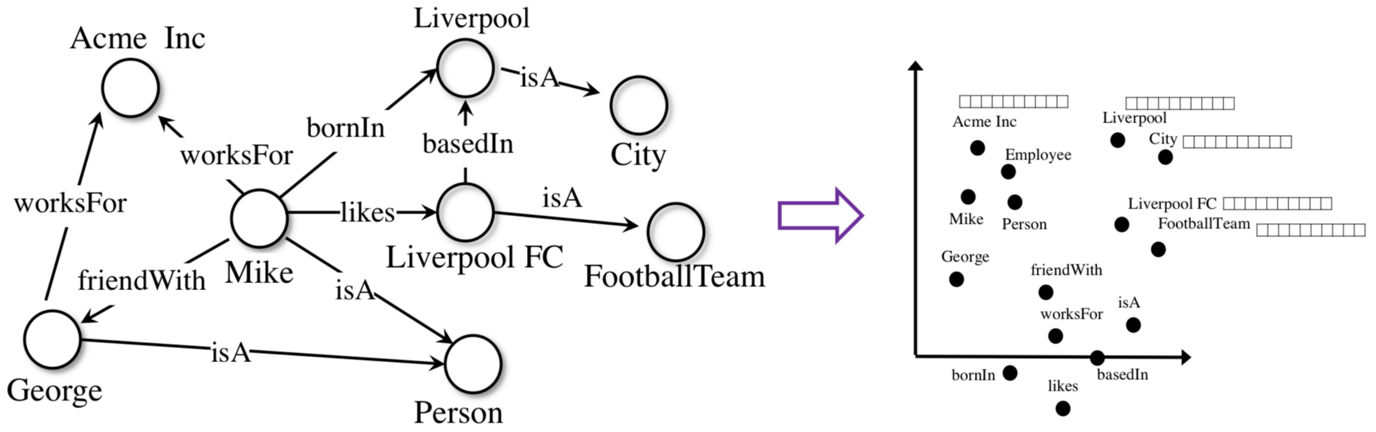
KG Instantiation



Background

Knowledge Graph Embedding

- Knowledge Graph Eembedding (KGE) approaches encode the KG $G = (E, R)$ into low-dimensional vector spaces, such as $\mathbf{E} \in \mathbb{R}^{n_e \times d_e}$ and $\mathbf{R} \in \mathbb{R}^{n_r \times d_r}$.

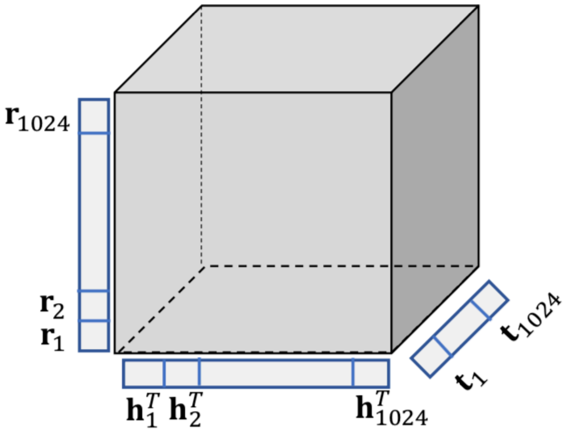


- Then the scoring function (SF) $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$ is utilized to measure whether a triplet (h, r, t) is real or not.
  - TransE: $f(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_1$

Outline

- Background
- Related Work
- Problem Formulation
- Search Algorithm
- Experiments

Related Work



Tensor Decomposition for KGC

- Among kinds of SFs, tensor decomposition models have been demonstrated their superiority due to the expressive guarantee and better empirical performance.
  - CANDECOMP/PARAFAC (CP) decomposition $f(h,r,t) =< \mathbf{h}, \mathbf{r}, \mathbf{t} >$
  - Tucker decomposition $f(h,r,t) = \boldsymbol{\mathcal{G}} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t}$
- TuckER utilizes a 3-order tensor $\mathcal{X} \in \{0,1\}^{n_e \times n_d \times n_e}$ to represents a KG.
  - $\mathcal{X}_{i,j,k} = 1$ represents that the fact (e_i, r_j, e_k) is known to exist
  - Otherwise, $\mathcal{X}_{i,j,k} = 0$
- Then TuckER factorizes $\mathcal{X}$ by Tucker Decomposition

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{E} \times_2 \mathbf{R} \times_3 \mathbf{E}$$

  - where $\mathcal{G} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the Tucker core tensor, $\mathbf{E} \in \mathbb{R}^{n_e \times d_e}$ and $\mathbf{R} \in \mathbb{R}^{n_r \times d_r}$ represents embeddings of entities and relations respectively.

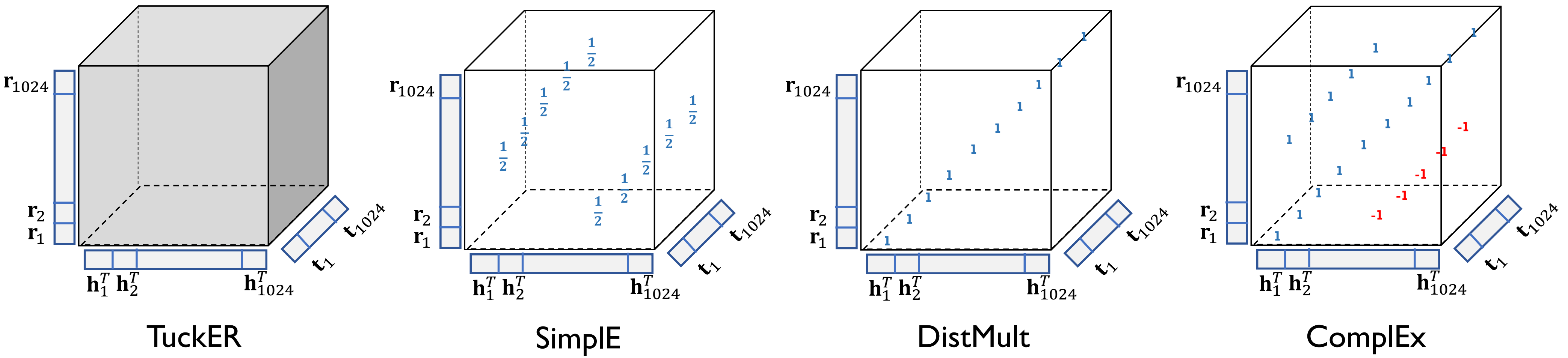
Outline

- Background
- Related Work
- Problem Formulation
- Search Algorithm
- Experiments

Problem Formulation

Motivation

- However, $\mathcal{G}$ in TuckER requires cubic complexity $O(d_e^2 d_r)$, which is hard to train and easy to overfit without sufficient data.
  - But CP-based tensor decomposition models (e.g., DistMult, ComplEx, SimplE) achieve relatively good performance without introducing the dense core tensor.
- CP-based tensor decomposition models can be regarded to have a special core tensor with **sparse** constraint.

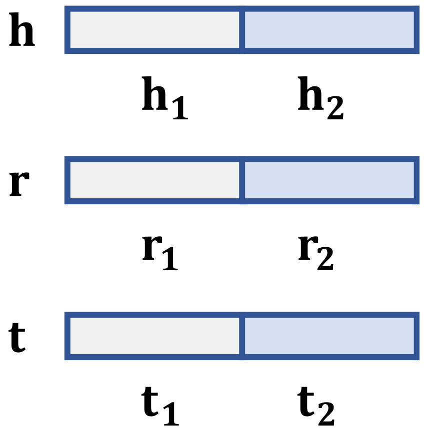


Problem Formulation

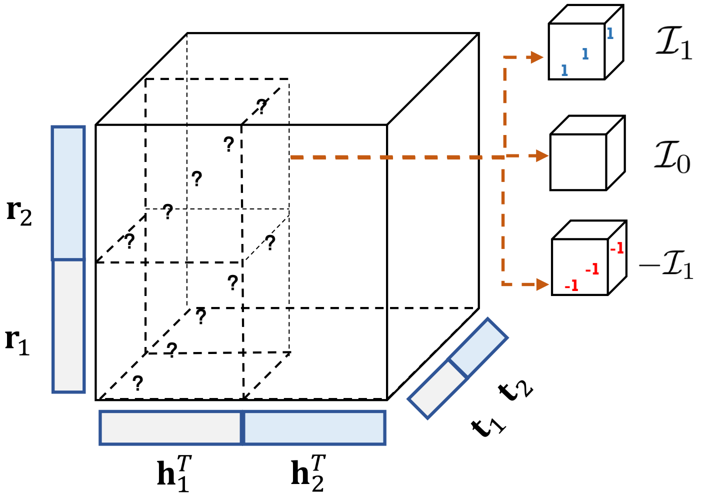
Regularizing Tucker Decomposition

- To alleviate over-parameterization issue, we propose to regularize Tucker decomposition.
    - Embedding segmentation: first divides an embedding $\mathbf{h} \in \mathbb{R}^d$ into m segmentations as $\mathbf{h} = [\mathbf{h_1}; \dots; \mathbf{h_m}]$, where $\mathbf{h_i} \in \mathbb{R}^{d/m}$, and same for $\mathbf{r}$ and $\mathbf{t}$.
    - Candidate diagonal tensors: $\mathbb{O} = \{\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_{-1}\}$.

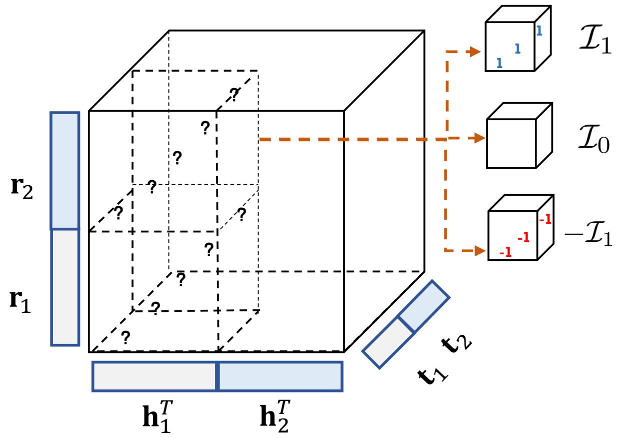
Embedding Segmentation



Regularize Core Tensor by Diagonal Tensor



Problem Formulation



Adaptive Regularizing Tucker Decomposition

- Given a tensor $\mathcal{G} \in \mathbb{R}^{d\times d\times d}$, let $\delta(\mathcal{G})$ divide $\mathcal{G}$ into m^3 cube segmentations $\mathbb{G} = \{\mathcal{G}^{ijk}\}$ where $\mathcal{G}^{ijk} \in \mathbb{O} = \{\mathcal{J}_0, \mathcal{J}_1, \mathcal{J}_{-1}\}$. The SF is defined as:

$$f_{\mathbb{G}}(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \sum_{ijk} \mathcal{G}^{ijk} \times_1 \mathbf{h}_i \times_2 \mathbf{r}_j \times_3 \mathbf{t}_k$$

- **Search Problem**: Inspired by automated machine learning (AutoML), we propose to adaptively regularize Tucker (ART) decomposition for any given KG data $\mathbb{S}$.

$$\bar{\mathbb{G}} = \arg\min_{\mathbb{G}} \sum_{(h,r,t)\in\mathbb{S}_{val}} \mathcal{L}\left(f_{\mathbb{G}}(\bar{\mathbf{h}}, \bar{\mathbf{r}}, \bar{\mathbf{t}})\right)$$

$$s.t. \{\bar{\mathbf{h}}, \bar{\mathbf{r}}, \bar{\mathbf{t}}\} = \arg\min_{\{\mathbf{h},\mathbf{r},\mathbf{t}\}} \sum_{(h,r,t)\in\mathbb{S}_{tra}} \mathcal{L}\left(f_{\mathbb{G}}(\mathbf{h}, \mathbf{r}, \mathbf{t})\right)$$

  - where $\mathcal{L}(\cdot)$ measures the loss of embeddings on the corresponding data.

Outline

- Background
- Related Work
- Problem Formulation
- Search Algorithm
- Experiments

Search Algorithm

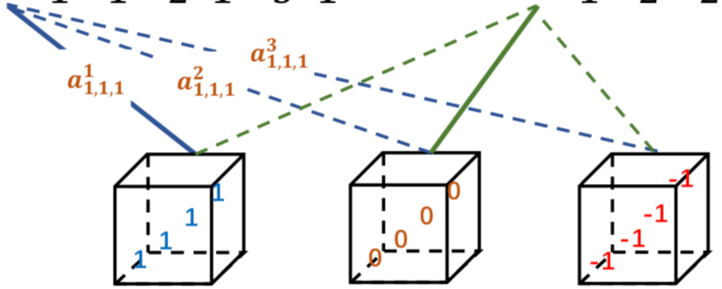
Optimization

- Optimizing search problem is a non-trivial task due to large amount of candidates.
  - For example, there are 3^{64} candidates when $m = 4$.
- To enable an efficient search method, we first design a continuous view as:

$$\mathcal{G}^{ijk} = \sum\nolimits_{o_p \in \mathbb{O}} a_{ijk}^{p} \cdot o_p$$

  - where a_{ijk}^{p} denotes the weight between o_p with $\mathcal{G}^{ijk}$.

$$f_{\mathbb{G}}(\mathrm{h}, \mathrm{r}, \mathrm{t}) = ? \times_1 \mathbf{h}_1 \times_2 \mathbf{r}_1 \times_3 \mathbf{t}_1 + \cdots + ? \times_1 \mathbf{h}_2 \times_2 \mathbf{r}_2 \times_3 \mathbf{t}_2$$



$$\bar{\mathbb{G}} = \arg\min_{\mathbb{G}} \sum\nolimits_{(h,r,t)\in\mathbb{S}_{val}} \mathcal{L}\left(f_{\mathbb{G}}(\bar{\mathbf{h}}, \bar{\mathbf{r}}, \bar{\mathbf{t}})\right)$$

$$s.t. \{\bar{\mathbf{h}}, \bar{\mathbf{r}}, \bar{\mathbf{t}}\} = \arg\min_{\{\mathbf{h},\mathbf{r},\mathbf{t}\}} \sum\nolimits_{(h,r,t)\in\mathbb{S}_{tra}} \mathcal{L}\left(f_{\mathbb{G}}(\mathbf{h}, \mathbf{r}, \mathbf{t})\right)$$

Search Algorithm

Optimization

- Initialize architecture weight $\boldsymbol{A}^{\mathbf{0}} = [a_{ijk}^{p}]$ and embeddings $\boldsymbol{X}^{\mathbf{0}} = \{\boldsymbol{E}, \boldsymbol{R}\}$;
- **while** not converge **do**
  # optimize embeddings
  - Randomly sample a mini-batch $\mathbb{B}_{tra}$ from $\mathbb{S}_{tra}$;
  - Sample a regularized core tensor $\mathbb{G}$ based on weight $\boldsymbol{A}^{\boldsymbol{t}}$;
  - Update embeddings $\boldsymbol{X}$ as: $\boldsymbol{X}^{\boldsymbol{t}+\mathbf{1}} \leftarrow \boldsymbol{X}^{\boldsymbol{t}} - \eta \nabla_{\boldsymbol{X}} L(f_{\mathbb{G}}(\boldsymbol{X}^{\boldsymbol{t}}); \mathbb{B}_{tra})$;
  # optimize architecture weight
  - Randomly sample a mini-batch $\mathbb{B}_{val}$ from $\mathbb{S}_{val}$;
  - Update weight $\boldsymbol{A}$ as: $\boldsymbol{A}^{\boldsymbol{t}+\mathbf{1}} \leftarrow \boldsymbol{A}^{\boldsymbol{t}} - \epsilon \nabla_{\boldsymbol{A}} L(f_{\boldsymbol{A}^{\boldsymbol{t}}}(\boldsymbol{X}^{\boldsymbol{t}}); \mathbb{B}_{val})$;
- **end while**
- Derive the final $\mathbb{G}^*$ from the final $\boldsymbol{A}^*$, and achieve embedding $\boldsymbol{X}^*$ by training $\mathbb{G}^*$ from scratch to convergence.

Outline

- Background
- Related Work
- Problem Formulation
- Search Algorithm
- Experiments

Experiments

Link Prediction Performance

- Experiment Setup
  - KGC task: link prediction
  - Datasets: WN18, WN18RR, FB15k, FB15k237

Table 2: Comparison of the best SFs identified by ART and the state-of-the-art SFs on the link prediction task.

model	WN18		WN18RR		FB15k		FB15k237	
	MRR	Hit@10	MRR	Hit@10	MRR	Hit@10	MRR	Hit@10
RotatE [Sun *et al.*, 2019]	0.949	**95.9**	0.476	57.1	0.797	88.4	0.297	48.0
ConvE [Dettmers *et al.*, 2018]	0.943	95.6	0.430	52.0	0.657	83.1	0.325	50.1
HolEX [Xue *et al.*, 2018]	0.938	94.9	-	-	0.800	88.6	-	-
QuatE [Zhang *et al.*, 2019]	0.950	**95.9**	<u>0.488</u>	**58.2**	<u>0.833</u>	90.0	0.357	55.3
DistMult [Wang *et al.*, 2014]	0.821	95.2	0.443	50.7	0.817	89.5	0.349	53.7
ComplEx [Trouillon *et al.*, 2017]	<u>0.951</u>	95.7	0.471	55.1	0.831	<u>90.5</u>	0.347	54.1
SimplE [Kazemi and Poole, 2018]	0.950	**95.9**	0.48	55.5	0.830	90.3	0.350	54.4
TuckER [Balazevic *et al.*, 2019]	**0.953**	<u>95.8</u>	0.470	52.6	0.795	89.2	<u>0.358</u>	<u>54.4</u>
ART (ours)	0.950	**95.9**	**0.489**	<u>56.8</u>	**0.840**	**90.8**	**0.360**	**55.0**

Experiments

Efficiency

- The time cost of ART is cheaper than TuckER, while it is longer than the simplest tensor decomposition method, DistMult.

Case Study

- ART can search different $\mathbb{G}$ for various KGs.

Table 3: Running time (in hours) analysis of SFs on single GPU.

data set	ART		TuckER	DistMult
	Search	Training		
WN18	5.89	4.73	25.42	1.9
WN18RR	3.12	3.04	18.70	0.42
FB15k	13.61	10.79	38.67	8.36
FB15k237	5.66	3.86	21.33	2.6

Table 4: The example of searched $\mathbb{G}$ on WN18RR with $m = 2$.

data sets	$\mathcal{G}_{111}$	$\mathcal{G}_{112}$	$\mathcal{G}_{121}$	$\mathcal{G}_{122}$	$\mathcal{G}_{211}$	$\mathcal{G}_{212}$	$\mathcal{G}_{221}$	$\mathcal{G}_{222}$
WN18RR	$\mathcal{I}_1$	$\mathcal{I}_1$	$-\mathcal{I}_1$	$\mathcal{I}_0$	$\mathcal{I}_1$	$-\mathcal{I}_1$	$\mathcal{I}_1$	$-\mathcal{I}_1$
FB15k237	$\mathcal{I}_1$	$-\mathcal{I}_1$	$\mathcal{I}_1$	$\mathcal{I}_1$	$\mathcal{I}_0$	$-\mathcal{I}_1$	$-\mathcal{I}_1$	$\mathcal{I}_1$

References

- Bordes, Antoine, et al. "Translating embeddings for modeling multi-relational data." *Advances in neural information processing systems* 26 (2013): 2787-2795.
- Balažević, Ivana, Carl Allen, and Timothy M. Hospedales. "Tucker: Tensor factorization for knowledge graph completion." *arXiv preprint arXiv:1901.09590* (2019).
- Yang, Bishan, et al. "Embedding entities and relations for learning and inference in knowledge bases." *arXiv preprint arXiv:1412.6575* (2014).
- Trouillon, Théo, et al. "Complex embeddings for simple link prediction." International Conference on Machine Learning (ICML), 2016.
- Kazemi, Seyed Mehran, and David Poole. "Simple embedding for link prediction in knowledge graphs." *Advances in neural information processing systems*. 2018.
- Lacroix, Timothée, Nicolas Usunier, and Guillaume Obozinski. "Canonical tensor decomposition for knowledge base completion." *arXiv preprint arXiv:1806.07297* (2018).
- Yao, Quanming, et al. "Efficient Neural Architecture Search via Proximal Iterations." *AAAI*. 2020.

Thank you!

Q & A