Geometry-Aware Gradient Algorithms for Neural Architecture Search

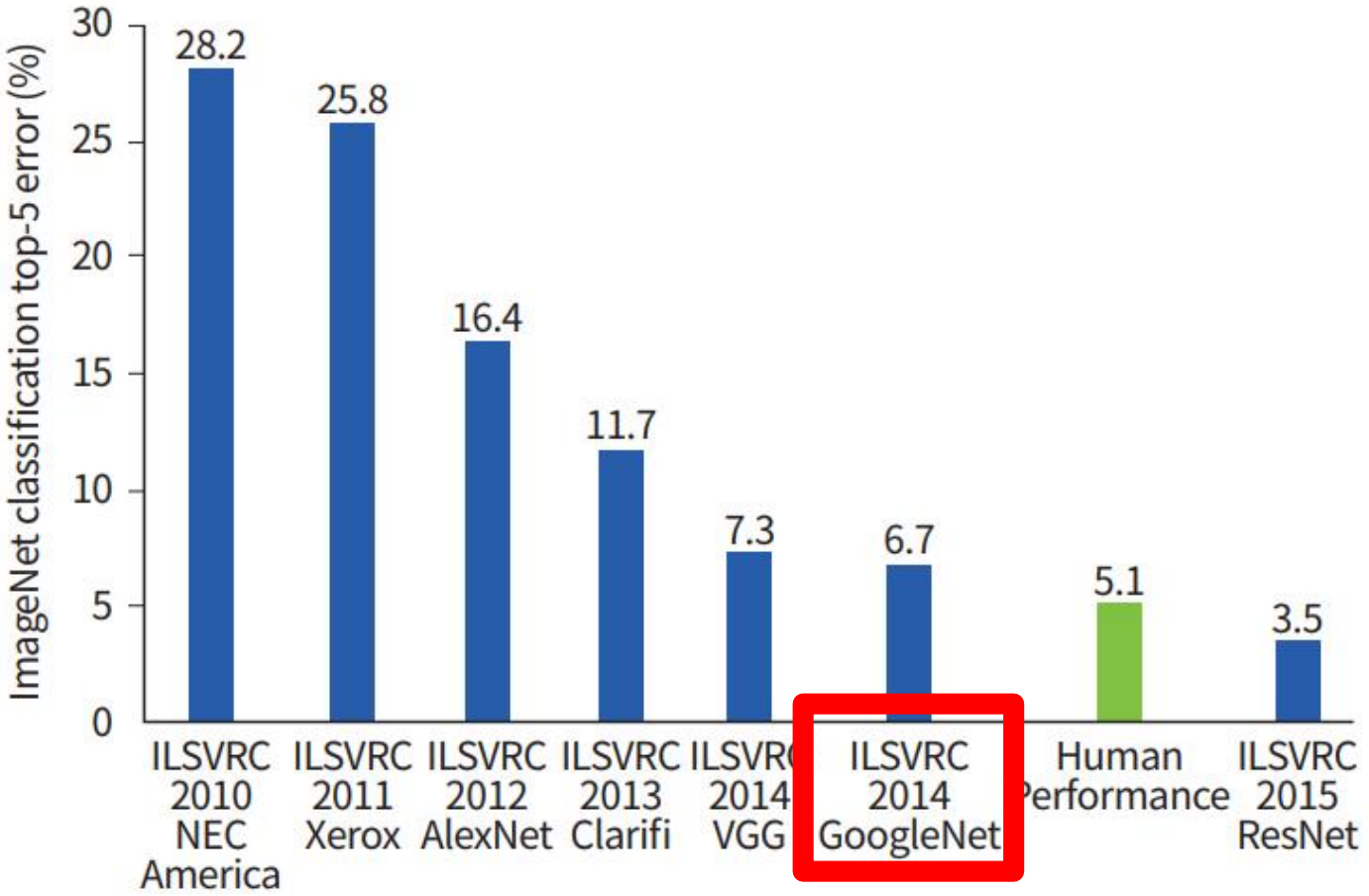
Misha Khodak
with Liam Li, Nina Balcan, and Ameet Talwalkar

Determined AI

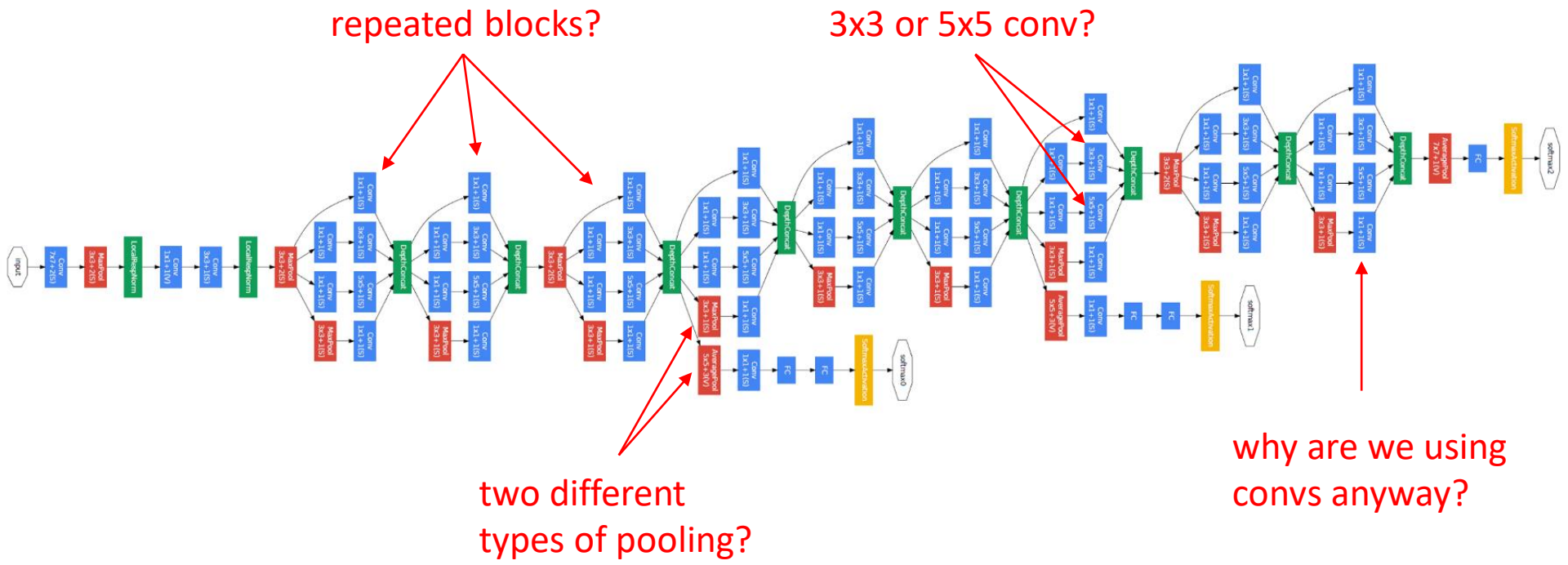
8 FEBRUARY 2021

AAAI-21 WORKSHOP ON LEARNING NETWORK ARCHITECTURE DURING TRAINING

What drives progress in machine learning?



Complex, expert-driven architecture design



Neural architecture search

Goal: automate away the design of neural architectures

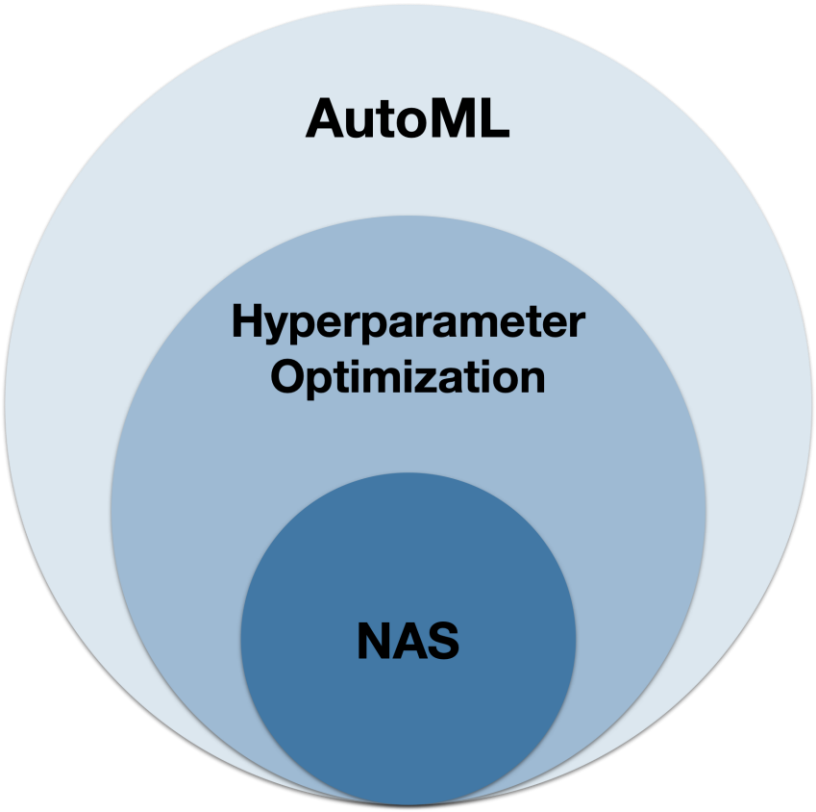
- help practitioners in advanced areas of ML (vision, language, …)
- accelerate progress in budding application areas (natural sciences, social sciences, …)
- reduce the need for ML expertise on domain-specific problems

Neural architecture search

Goal: automate away the design of neural architectures.

Reality: search through a pre-defined space of neural networks.

Usually posed as a subproblem of hyperparameter optimization.



Should NAS be viewed as hyperparameter optimization?

Our work:

1. NAS is perhaps better-viewed as regular ML over an extended function class
   - Reduces NAS to optimization and regularization over this class.
   - Revealed by the popular weight-sharing technique for NAS.

Should NAS be viewed as hyperparameter optimization?

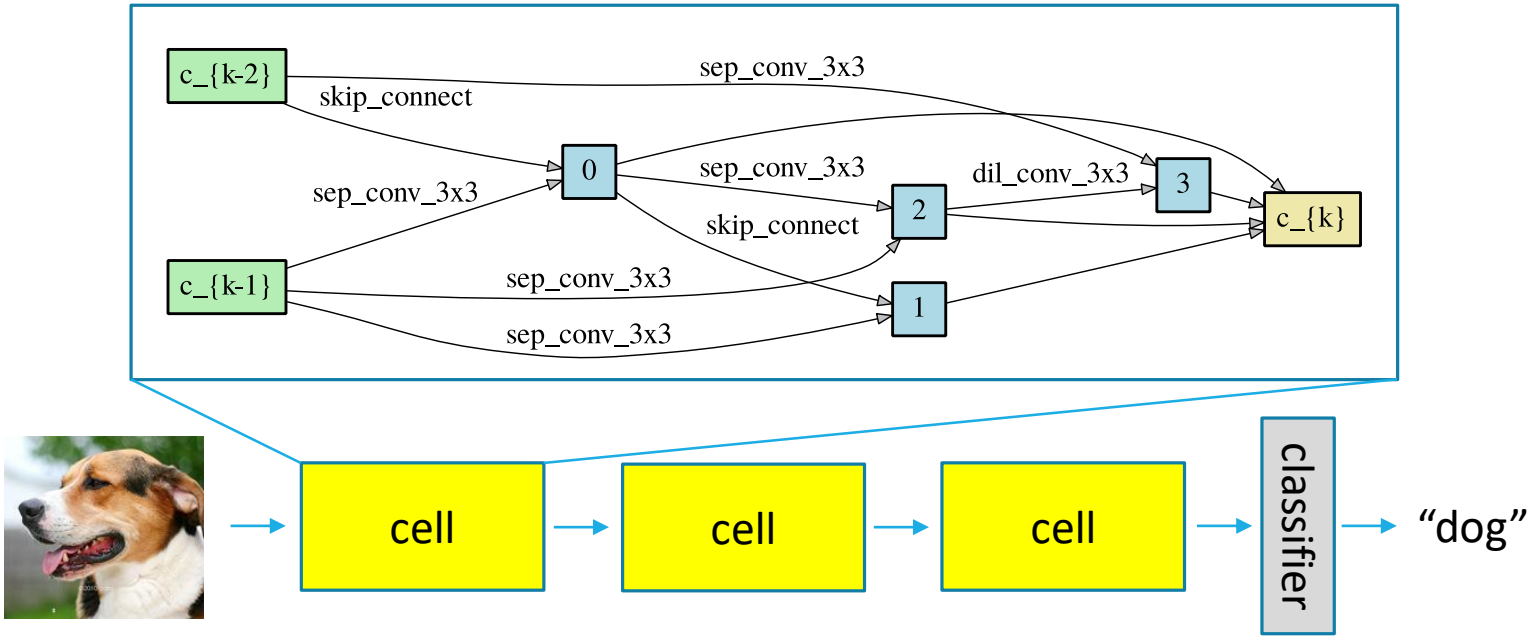
Our work:

1. NAS is perhaps better-viewed as regular ML over an extended function class
   - Reduces NAS to optimization and regularization over this class.
   - Revealed by the popular ~~weight-sharing technique for NAS~~ Learning Network Architecture During Training?

2. New class of mirror-descent based algorithms for NAS
   - Simple convergence guarantee.
   - New state-of-the-art results on the latest benchmarks.

Modern cell-block search spaces

- Cell - DAG with operation label on each allowed edge
- Repeat the same cell block several time until output



The hyperparameter optimization view of NAS

validation loss

$$\min_{a \in A} \quad \ell_V(w_a, a)$$

set of possible architectures

training loss

$$\text{s.t.} \qquad w_a \in \arg\min_{w \in \mathbb{R}^d} \quad \ell_T(w, a)$$

set of network weights

Evaluating the outer problem even once requires training a deep net.

Early NAS methods were exorbitantly expensive

Approach: apply standard hyperparameter optimization to search over the space

- Random search
- Bayesian optimization
- Reinforcement learning

Tens of thousands of GPU-hours needed for state-of-the-art accuracy.

Weight-sharing to the rescue
[Pham et al., ICML 2018]

Solution:

- Train one “supernet” containing all architectures in the search space
- Use its weights to pick which architectures to train to completion

10,000x reduction in search cost to achieve state-of-the-art accuracy.

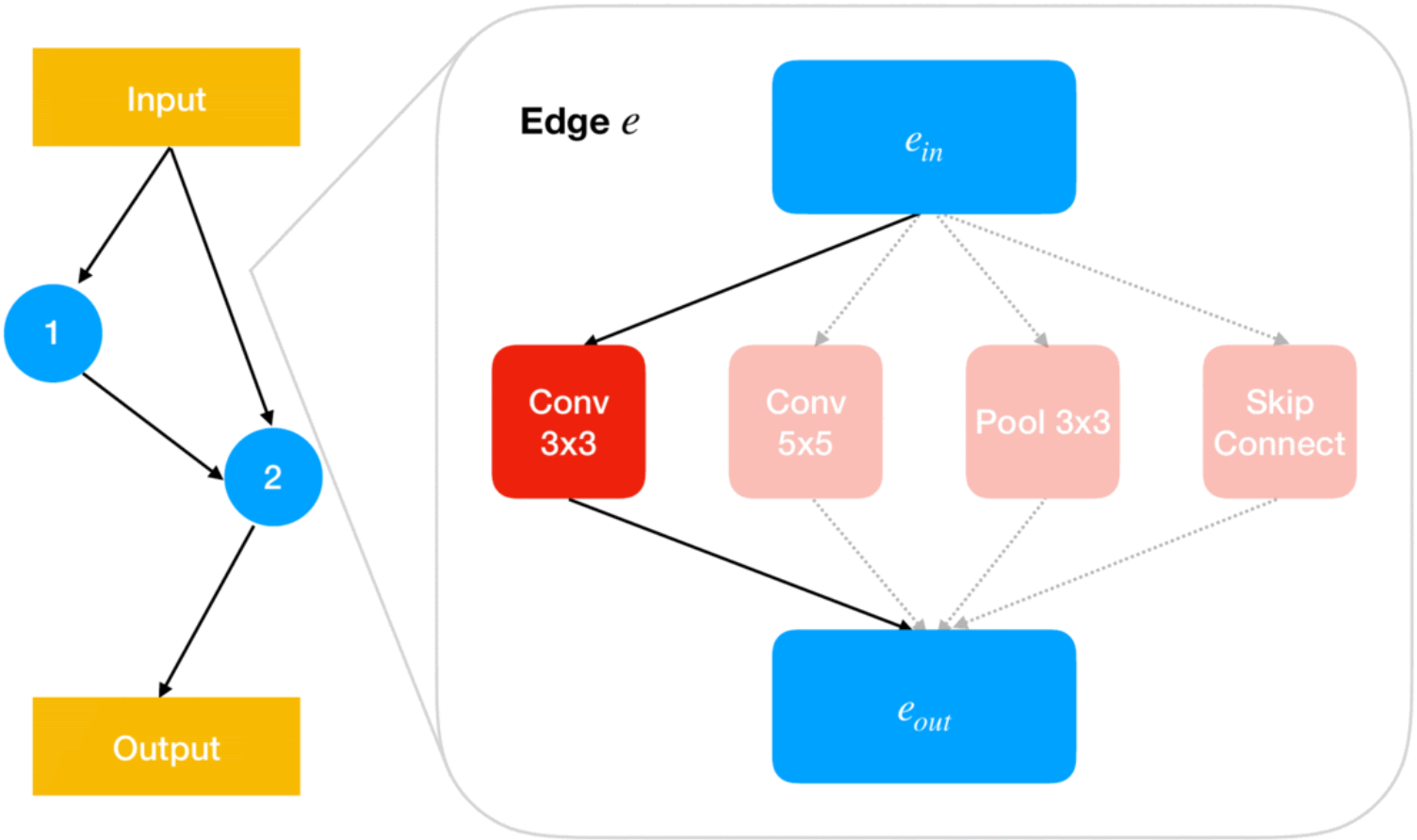
Random search with weight-sharing [Li & Talwalkar, UAI 2019]

Approach:

1. Sample an architecture from the search space
2. Update shared weights using a gradient step with this architecture
3. Repeat 1 and 2 until convergence
4. Use final shared weights to pick the output architecture

$$w_{\text{shared}} \in \arg\min \quad \mathbb{E}_{a \in \text{Unif}(A)} \ell_T(w, a)$$
$$a_{\text{output}} \in \arg\min_{a \in A} \quad \ell_V(w_{\text{shared}}, a)$$

Continuous relaxation for weight-sharing [Liu et al., ICLR 2019]



Optimization-based methods such as DARTS [Liu et al., ICLR 2019]

operation weight on edge e

$$\theta_{e,o} = \frac{\exp(\alpha_{e,o})}{\sum_{o' \in O} \exp(\alpha_{e,o'})}$$

architecture parameter

Recent methods (DARTS, GDAS, PC-DARTS) based on alternating optimization

1. **initialize** $w^{(0)} \in \mathbb{R}^d, \alpha^{(0)} \in \mathbb{R}^{|E||O|}$
2. **for iteration** $i = 0, \ldots, n-1$ **do:**
3. $\quad w^{(i+1)} \leftarrow w^{(i)} - \eta \nabla_w \ell_T(w^{(i)}, \alpha^{(i)})$
4. $\quad \alpha^{(i+1)} \leftarrow \alpha^{(i)} - \eta \nabla_\alpha \ell_V(w^{(i+1)}, \alpha^{(i)})$
5. **return** $\theta = \mathrm{Softmax}(\alpha^{(n)})$

Weight-sharing to the rescue?

- Popular methods: ENAS, DARTS, RS-WS, GDAS, PC-DARTS

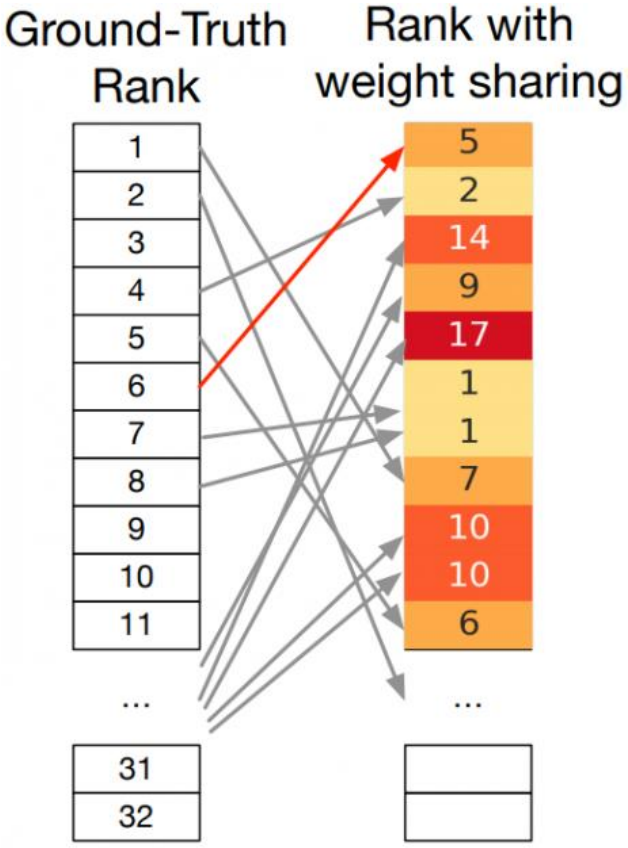
- Pros:
  - 10000x reduction in search cost

- Cons:
  - Inconsistent results on recent benchmarks
  - Rank-disorder problem
  - Post-hoc pruning step
  - People are surprised that it works

Problems with weight-sharing: Rank-disorder [Yu et al., ICLR 2020]

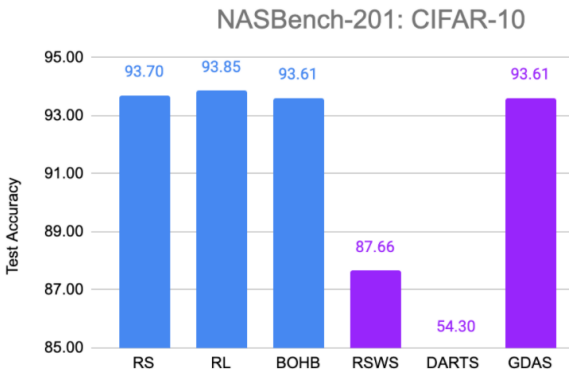
Rank-disorder:

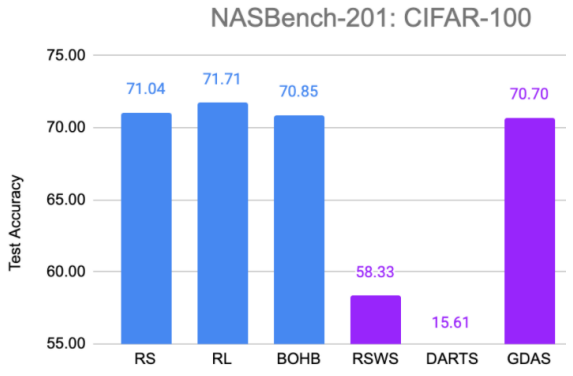
Validation loss of architectures computed using shared weights does not correlate with that computed using standalone weights.

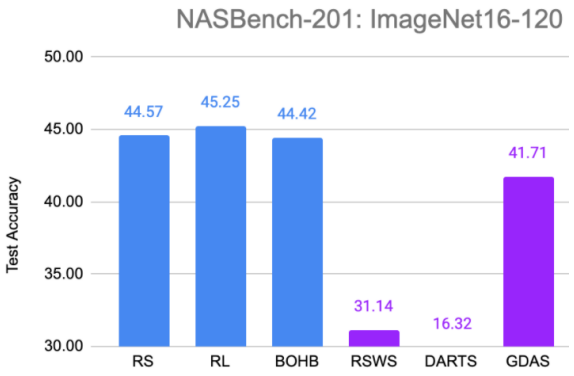


Problems with weight-sharing: Inconsistency [Dong & Yang, ICLR 2020]

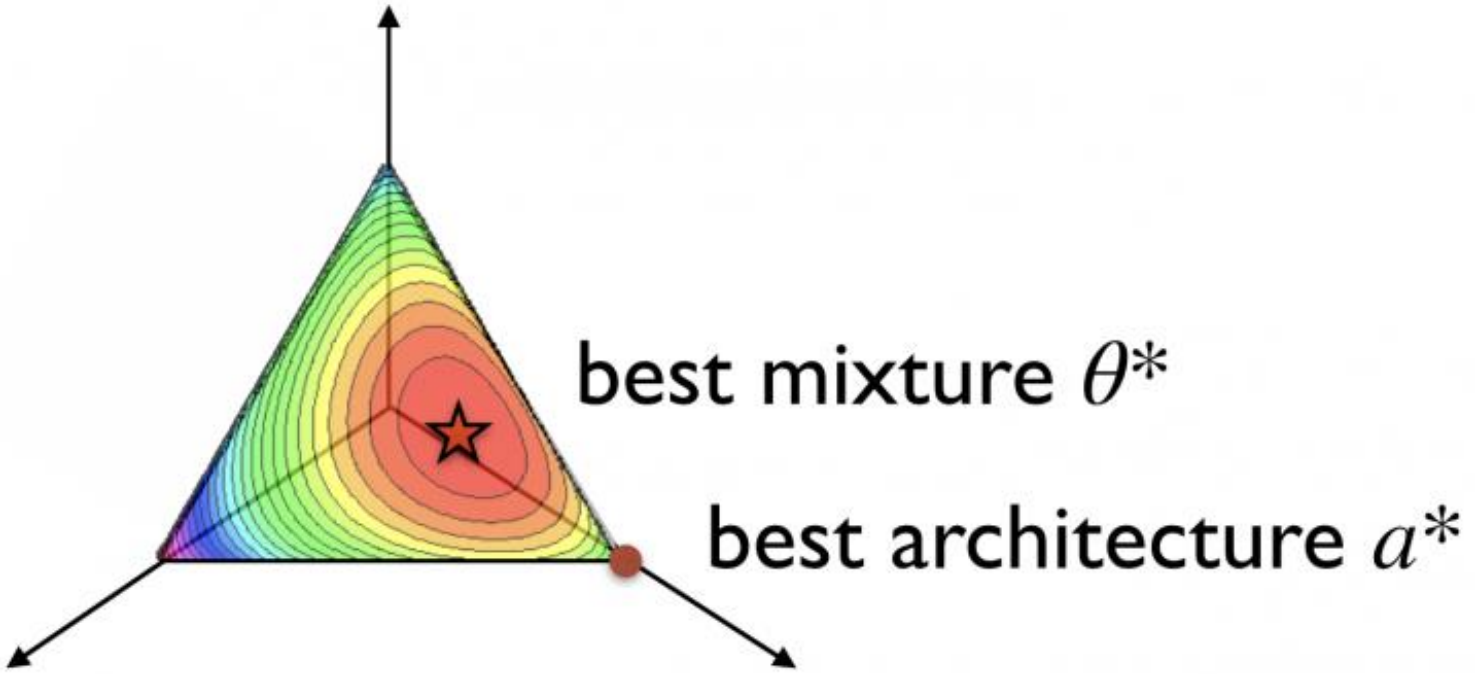
Across-the-board bad performance on NAS-Bench-201.







Problems with weight-sharing: Post-hoc pruning



Validation Accuracy:

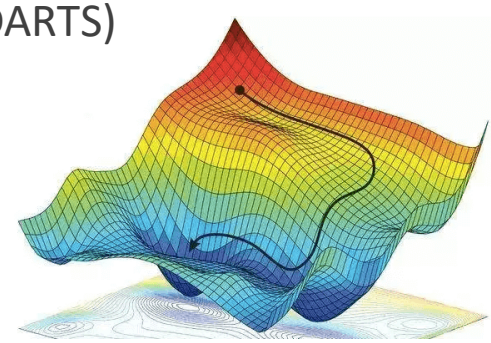
72%

15%

Our work:

- Understand and study the NAS problem as just vanilla empirical risk minimization:
  - Effectiveness of weight-sharing is not surprising
  - Don’t be afraid of rank-disorder
  - Fixing NAS comes down to better regularization and optimization, which we know how to do well in machine learning.

- New geometry-aware optimization fix for NAS
  - Applicable to many modern methods (DARTS, GDAS, PC-DARTS)
  - Sparser architectures – less post-hoc pruning
  - State-of-the-art performance



Vanilla NAS with weight-sharing is just empirical risk minimization

- Weight-sharing is applicable only for tuning *architectural* hyperparameters, where changing the hyperparameter directly affects the loss without changing the model weights.

- The same hyperparameters can simply be learned using vanilla ERM:

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

Vanilla NAS with weight-sharing is just empirical risk minimization

By taking the ERM view, we see that:

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

Vanilla NAS with weight-sharing is just empirical risk minimization

By taking the ERM view, we see that:

- NAS with weight-sharing is just optimizing a non-convex over-parameterized model.

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

Vanilla NAS with weight-sharing is just empirical risk minimization

By taking the ERM view, we see that:

- NAS with weight-sharing is just optimizing a non-convex over-parameterized model. *We do this all the time in deep learning.*

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

Vanilla NAS with weight-sharing is just empirical risk minimization

By taking the ERM view, we see that:

- NAS with weight-sharing is just optimizing a non-convex over-parameterized model. *We do this all the time in deep learning.*
- We only care about a single solution, not a ranking of the architectures in the search space.

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

Vanilla NAS with weight-sharing is just empirical risk minimization

By taking the ERM view, we see that:

- NAS with weight-sharing is just optimizing a non-convex over-parameterized model. *We do this all the time in deep learning.*
- We only care about a single solution, not a ranking of the architectures in the search space. *Don't worry about rank-disorder.*

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

So why is weight-sharing failing?

- For regular ERM in deep learning, we've spent decades
  - developing optimization routines (initializations, momentum, Adam, …)
  - developing regularization techniques (weight-decay, dropout, …)

- We know a lot about optimization and regularization for ERM over regular deep nets.

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

shared weights

architecture parameters

So why is weight-sharing failing?

- For regular ERM in deep learning, we've spent decades
  - developing optimization routines (initializations, momentum, Adam, …)
  - developing regularization techniques (weight-decay, dropout, …)

- We know a lot about optimization and regularization for ERM over regular deep nets. We need more time to get it right for NAS with weight-sharing.

loss over all data

$$\min_{w \in \mathbb{R}^d, \theta \in \Theta} \ell_T(w, \theta)$$

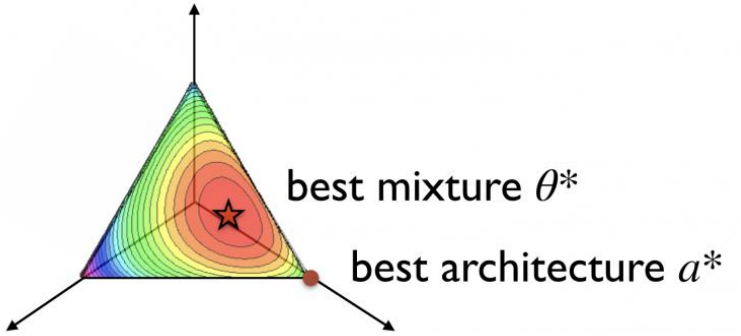
shared weights

architecture parameters

Lots of recent work on regularization

- Partial channel connections (Xu et al., ICLR 2020)
- Penalizing the Hessian of the validation loss (Zela et al., ICLR 2020)
- The bilevel formulation itself (Khodak et al., AutoML@ICML 2020)
- We focus on optimization instead.

Our work: fast convergence to sparse architecture parameters



$$a_{\text{output}} = \theta^{(n)} + \delta_{\text{pruning}}$$

$$\ell_V(a_{\text{output}}) \approx \ell_V(\theta^{(n)}) + \mathcal{O}\left(\|\nabla^2 \ell_V(\theta^{(n)})\| \|\delta_{\text{pruning}}\|\right)$$

minimize loss due to pruning by outputting sparse architectures

How do existing methods optimize architecture parameters?

- We want architecture parameters to lie in the simplex

operation weight on edge e

$$\theta_{e,o} = \frac{\exp(\alpha_{e,o})}{\sum_{o' \in O} \exp(\alpha_{e,o'})}$$

architecture parameter

- Existing methods (DARTS, GDAS, PC-DARTS) do this by SGD over unconstrained parameters, followed by softmax.
  - Pros: no projection step
  - Cons: simplex parameters are not sparse (bad for post-hoc pruning)

How *should* we optimize architecture parameters?

- Our solution: optimize directly over the simplex using exponentiated gradient descent:

gradient w.r.t. architecture parameter

$$\tilde{\theta} \leftarrow \theta^{(i)} \odot \exp\left(-\eta \nabla_{\theta} \ell(w^{(i+1)}, \theta^{(i)})\right)$$

$$\textbf{for } e \in E, o \in O \textbf{ do: } \theta_{e,o}^{(i+1)} \leftarrow \frac{\tilde{\theta}_{e,o}}{\sum_{o' \in O} \tilde{\theta}_{e,o'}}$$

cheap projection step to the simplex

- Pros:
  - Provably faster convergence.
  - Encourages sparsity by moving quickly when far away from edge of simplex.

Convergence notes

Our algorithm is a type of stochastic block mirror descent:

1. Update shared weights using regular gradient step
2. Update architecture parameters using exponentiated gradient step (mirror descent with KL divergence)

$$x_{i+1} \leftarrow \arg\min_{x \in X} \quad \langle \nabla f(x_i), x \rangle + \eta D(x, x_i)$$

learning rate

Bregman divergence

$$\text{for GD:} \quad D(x, y) = \frac{1}{2}\|x - y\|_2^2$$

$$\text{for EG:} \quad D(x, y) = KL(x||y)$$

Convergence notes

Existing results [Dang & Lan, 2015]: convergence for alternating mirror descent with smooth divergences (excludes KL divergence)

Our result:

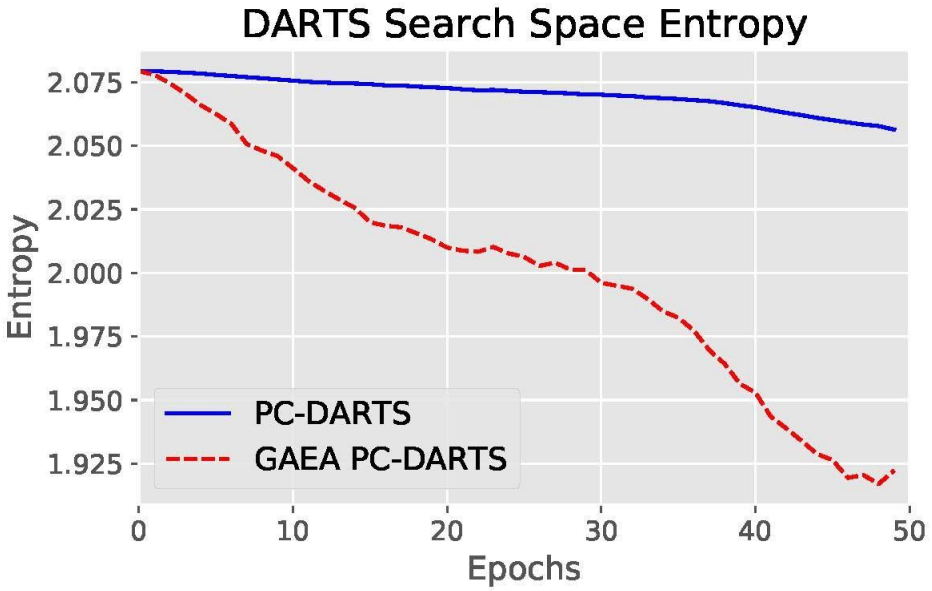
- Stationary point convergence for non-smooth divergence
- Proof technique: reduction to non-alternating mirror descent made possible by new convergence measure [Zhang & He, 2018]

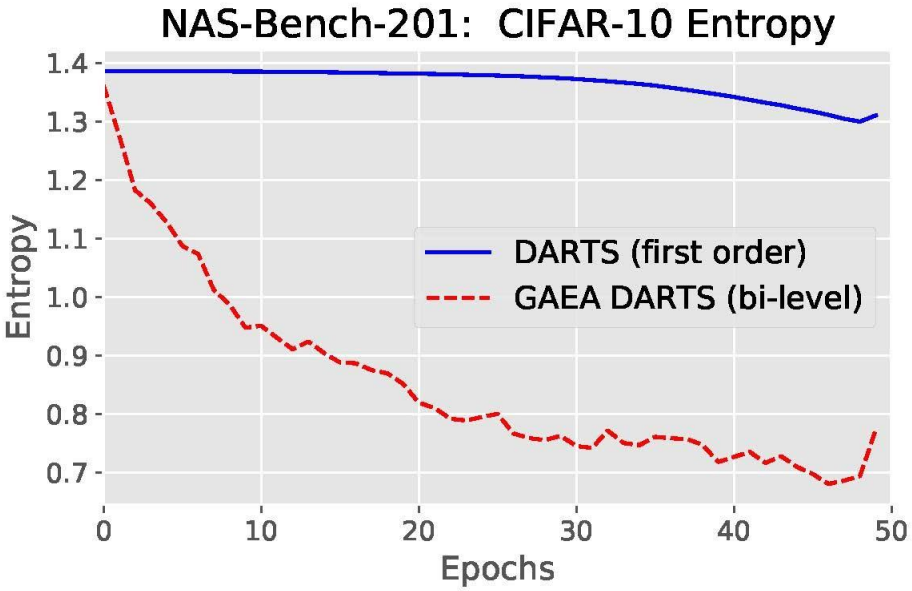
Number of iterations until ε-stationary point

GD over k-simplex: $\mathcal{O}\left(k/\varepsilon^2\right)$

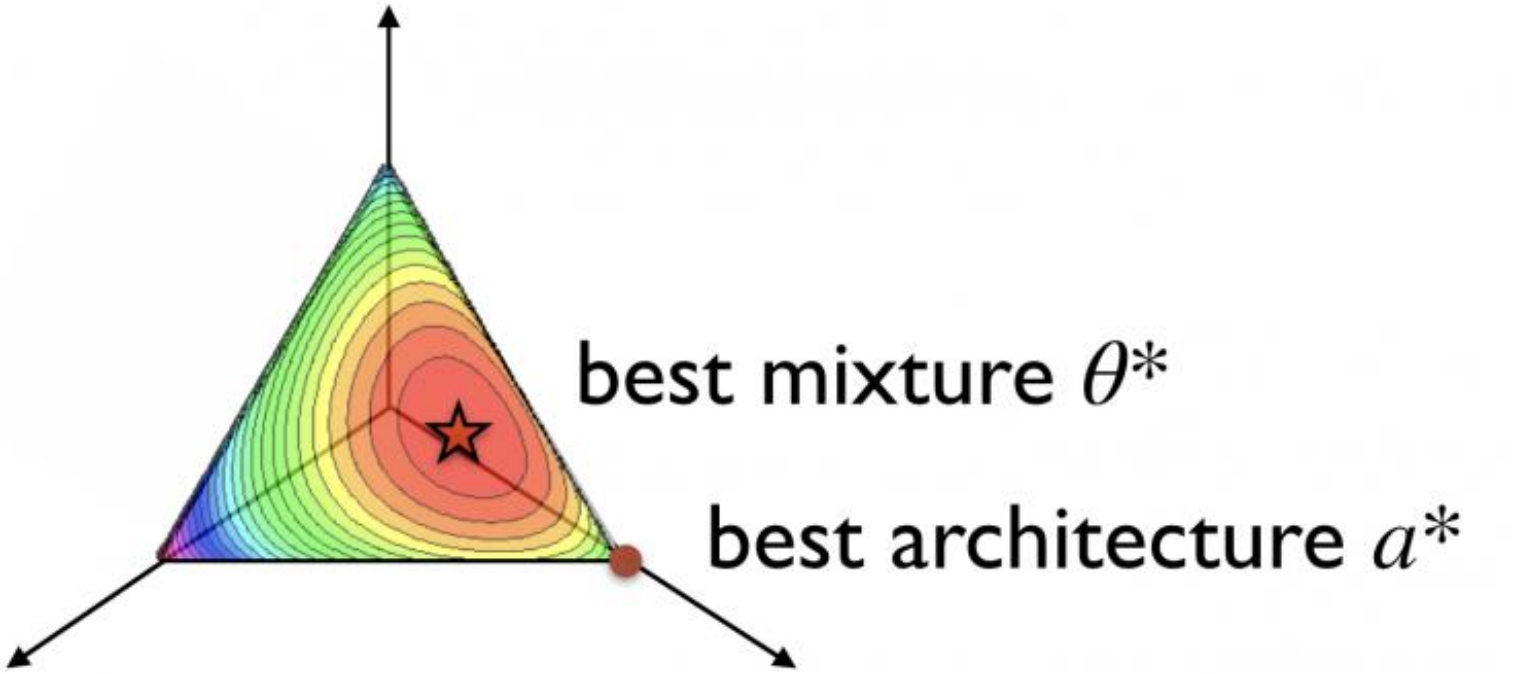
EG over k-simplex: $\mathcal{O}\left(\log k/\varepsilon^2\right)$

Geometry-Aware Exponentiated Algorithm (GAEA): Sparsity





Better optimization and less loss due to pruning

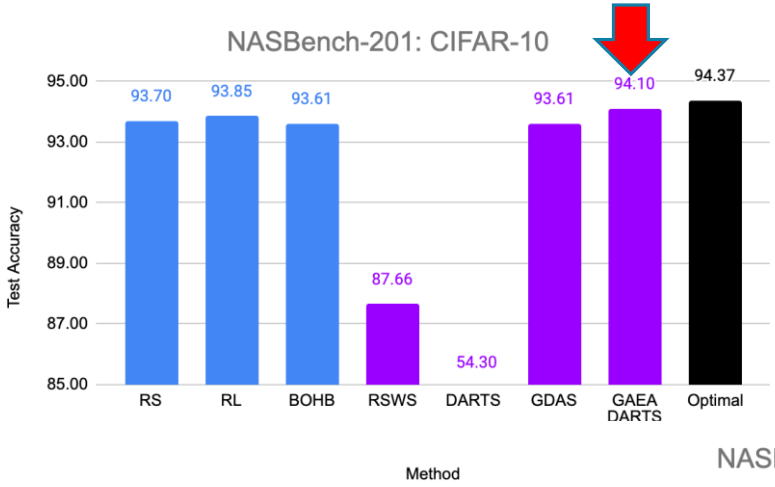


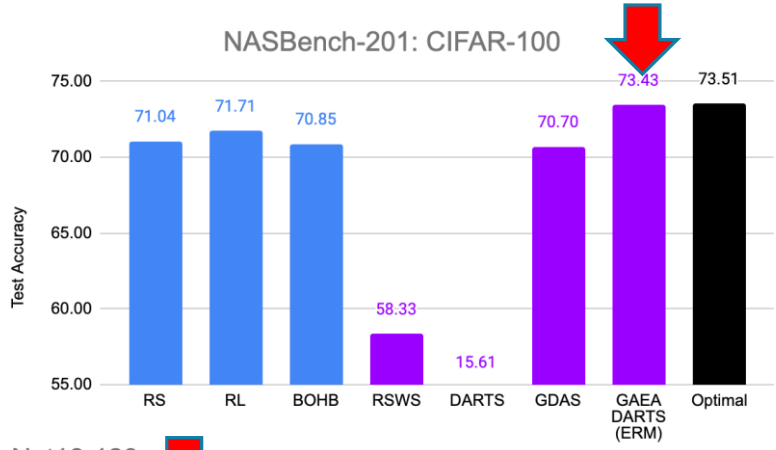
Validation Accuracy:

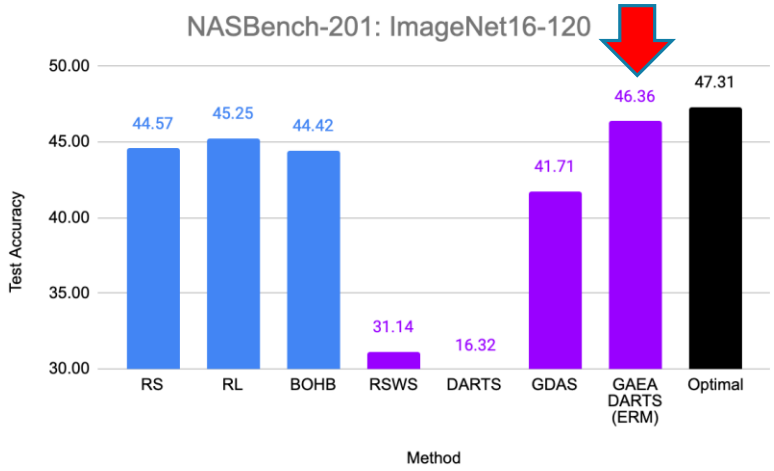
~~72%~~ 75%

~~15%~~ 33%

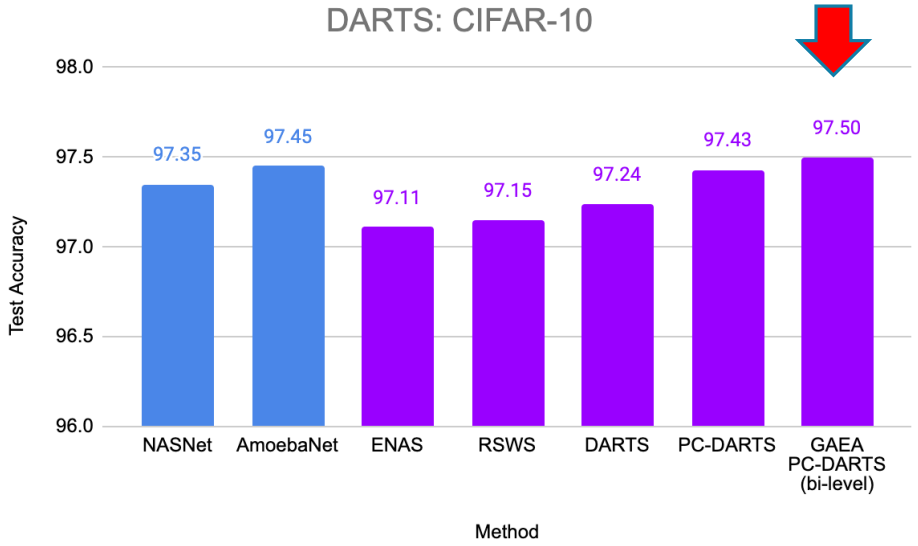
Geometry-Aware Exponentiated Algorithm (GAEA): NAS-Bench-201

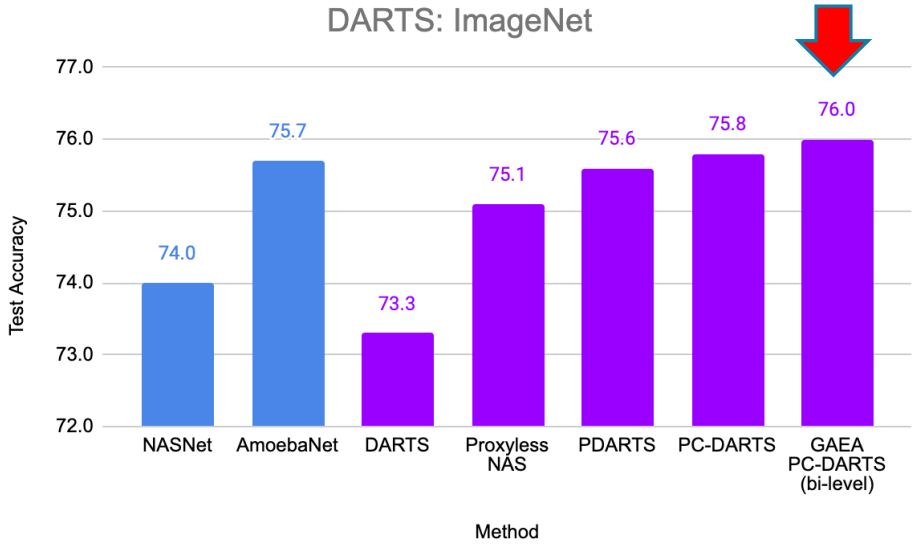






Geometry-Aware Exponentiated Algorithm (GAEA): DARTS Space





Main takeaways

- Rather than hyperparameter tuning, NAS can be viewed as just adding new model parameters that need to be optimized differently.

- There's a lot we still don't understand about NAS and weight-sharing, but success seems to come down to two things we know how to do:
  - Optimization
  - Regularization

- Our new NAS optimization method, GAEA:
  - Applicable to fix any method that uses softmax relaxation (DARTS, PC-DARTS, GDAS)
  - State-of-the-art results

Future Directions

- Geometry-aware optimization
  - Over network graphs, structured matrices encoding operations, activation function spaces, adding neurons, …
  - New applications of Riemannian SGD, natural gradient, boosting, …

- Other:
  - How to do post-hoc rounding?
  - Getting good network weights without re-training

Thank-you!

Paper: arxiv.org/abs/2004.07802

Code: github.com/liamcli/gaea_release

Blog: blog.ml.cmu.edu/2020/07/17/in-defense-of-weight-sharing-for-nas/