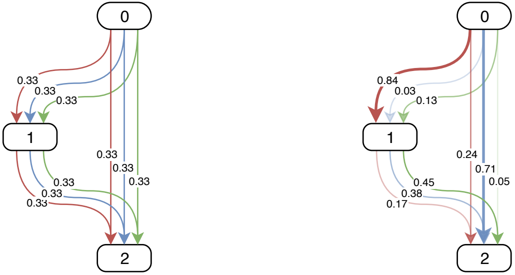
AutoML: Neural Architecture Search (NAS)

DARTS: Differentiable Architecture Search

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

DARTS: Differentiable Architecture Search [Liu et al, 2018]

- Use one-shot model with continuous architecture weight α for each operator

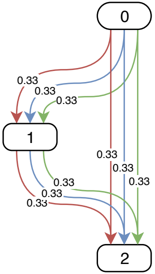


(a) Initialization (b) Search end

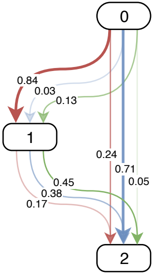
DARTS: Differentiable Architecture Search [Liu et al, 2018]

- Use one-shot model with continuous architecture weight α for each operator

$$x^{(j)} = \sum_{i<j} \tilde{o}^{(i,j)}(x^{(i)}) = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'}^{(i,j)})} o(x^{(i)})$$



(a) Initialization

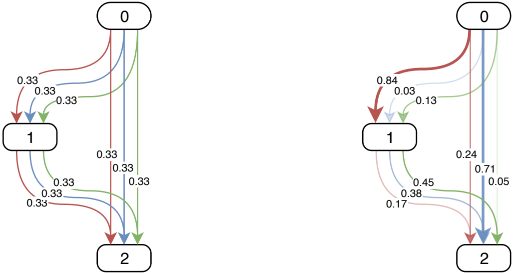


(b) Search end

DARTS: Differentiable Architecture Search [Liu et al, 2018]

- Use one-shot model with continuous architecture weight α for each operator

$$x^{(j)} = \sum_{i<j} \tilde{o}^{(i,j)}(x^{(i)}) = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'}^{(i,j)})} o(x^{(i)})$$



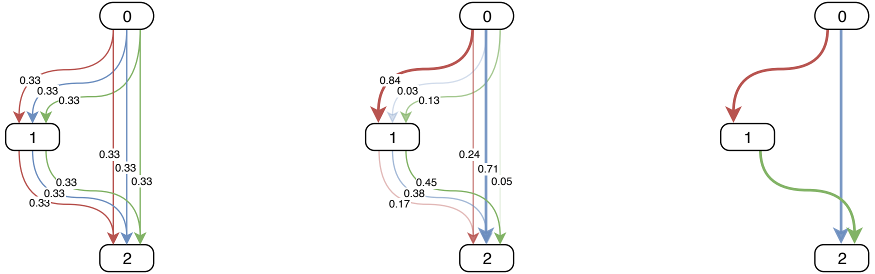
(a) Initialization (b) Search end

- By optimizing the architecture weights α, DARTS assigns importance to each operation
  - Since the α are continuous, we can optimize them with gradient descent

DARTS: Differentiable Architecture Search [Liu et al, 2018]

- Use one-shot model with continuous architecture weight α for each operator

$$x^{(j)} = \sum_{i<j} \tilde{o}^{(i,j)}(x^{(i)}) = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'}^{(i,j)})} o(x^{(i)})$$



(a) Initialization (b) Search end (c) Final cell

- By optimizing the architecture weights α, DARTS assigns importance to each operation
  - Since the α are continuous, we can optimize them with gradient descent
- In the end, DARTS discretizes to obtain a single architecture (c)

DARTS: Architecture Optimization

- The optimization problem (a $\rightarrow$ b) is a bi-level optimization problem:

$$\min_{\alpha} \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha)$$
$$s.t.\ w^*(\alpha) \in \operatorname{argmin}_w \mathcal{L}_{\text{train}}(w, \alpha)$$

DARTS: Architecture Optimization

- The optimization problem (a $\rightarrow$ b) is a bi-level optimization problem:

$$\begin{aligned} &\text{min}_\alpha \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \\ s.t.\ w^*(\alpha) \ &\in \ \text{argmin}_w \mathcal{L}_{\text{train}}(w, \alpha) \end{aligned}$$

- This is solved using alternating SGD steps on architectural parameters α and weights w

---

Algorithm: DARTS 1st order

---

while *not converged* **do**

  Update one-shot weights $\mathbf{w}$ by $\nabla_{\mathbf{w}} \mathcal{L}_{train}(\mathbf{w}, \alpha)$

  Update architectural parameters α by $\nabla_\alpha \mathcal{L}_{valid}(\mathbf{w}, \alpha)$

return $\arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)}$ for each edge (i, j)

DARTS: Architecture Optimization

- The optimization problem (a $\rightarrow$ b) is a bi-level optimization problem:

$$\min_\alpha \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha)$$
$$s.t.\ w^*(\alpha) \ \in \ \operatorname{argmin}_w \mathcal{L}_{\text{train}}(w, \alpha)$$

- This is solved using alternating SGD steps on architectural parameters α and weights w

---

Algorithm: DARTS 1st order

---

while *not converged* **do**

  Update one-shot weights $\mathbf{w}$ by $\nabla_{\mathbf{w}} \mathcal{L}_{train}(\mathbf{w}, \alpha)$

  Update architectural parameters α by $\nabla_\alpha \mathcal{L}_{valid}(\mathbf{w}, \alpha)$

return $\arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)}$ for each edge (i, j)

- Note: there is no theory showing that this process converges

Strong performance on some benchmarks

- E.g., original CNN search space
  - 8 operations on each MixedOp
  - 28 MixedOps in total
  - $> 10^{23}$ possible architectures
- Performance
  - $< 3\%$ error on CIFAR-10 in less than 1 GPU day of search

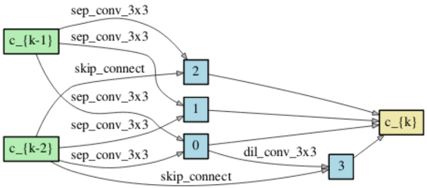


Figure 4: Normal cell learned on CIFAR-10.

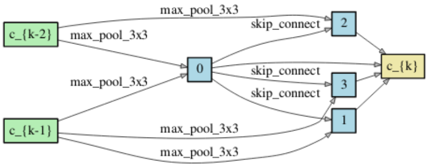


Figure 5: Reduction cell learned on CIFAR-10.

Issues – Non-robust behaviour

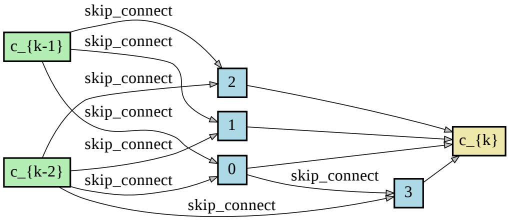
- DARTS is very sensitive w.r.t. its own hyperparameters (e.g. one-shot learning rate, L_2 regularization, etc.)

Issues – Non-robust behaviour

- DARTS is very sensitive w.r.t. its own hyperparameters (e.g. one-shot learning rate, L_2 regularization, etc.)
  - Tuning these hyperparameters for every new task/search space is computationally expensive

Issues – Non-robust behaviour

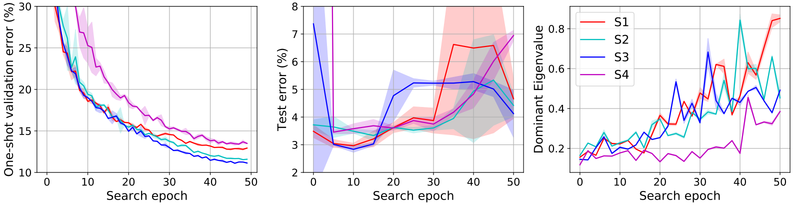
- DARTS is very sensitive w.r.t. its own hyperparameters (e.g. one-shot learning rate, L_2 regularization, etc.)
  - Tuning these hyperparameters for every new task/search space is computationally expensive
  - DARTS may return degenerate architectures, e.g., cells composed with only skip connections



Issues – Non-robust behaviour

- DARTS is very sensitive w.r.t. its own hyperparameters (e.g. one-shot learning rate, L_2 regularization, etc.)
  - Tuning these hyperparameters for every new task/search space is computationally expensive
  - DARTS may return degenerate architectures, e.g., cells composed with only skip connections

- RobustDARTS [Zela et al, 2020] – tracks the curvature of the validation loss and stops the search early based on that

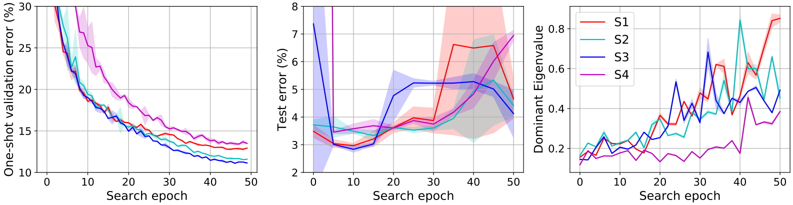


Issues – Non-robust behaviour

- DARTS is very sensitive w.r.t. its own hyperparameters (e.g. one-shot learning rate, L_2 regularization, etc.)
  - Tuning these hyperparameters for every new task/search space is computationally expensive
  - DARTS may return degenerate architectures, e.g., cells composed with only skip connections

- RobustDARTS [Zela et al, 2020] – tracks the curvature of the validation loss and stops the search early based on that

- SmoothDARTS [Chen and Hsieh, 2020] – applies random perturbation and adversarial training to avoid bad regions



Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors

Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors
  - This constrains the search space size and the fidelity used to train the one-shot model
  - Impossible to run on large datasets as ImageNet

Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors
  - This constrains the search space size and the fidelity used to train the one-shot model
  - Impossible to run on large datasets as ImageNet

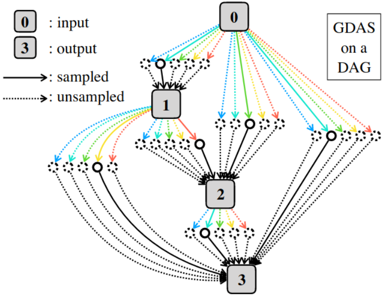
A lot of research aims to fix this issue:

Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors
  - This constrains the search space size and the fidelity used to train the one-shot model
  - Impossible to run on large datasets as ImageNet

A lot of research aims to fix this issue:

- GDAS [Dong et al, 2019] – samples from a Gumbel Softmax distribution to keep only a single architecture in memory

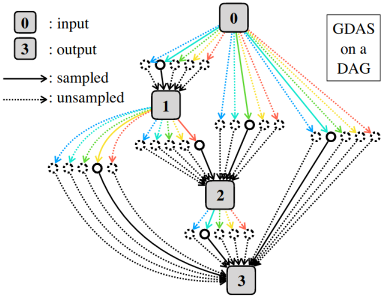


Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors
  - This constrains the search space size and the fidelity used to train the one-shot model
  - Impossible to run on large datasets as ImageNet

A lot of research aims to fix this issue:

- GDAS [Dong et al, 2019] – samples from a Gumbel Softmax distribution to keep only a single architecture in memory
- ProxylessNAS [Cai et al, 2019] – computes approximate gradients on α keeping only 2 edges between two nodes in memory at a time

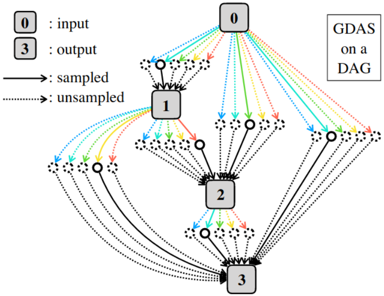


Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors
  - This constrains the search space size and the fidelity used to train the one-shot model
  - Impossible to run on large datasets as ImageNet

A lot of research aims to fix this issue:

- GDAS [Dong et al, 2019] – samples from a Gumbel Softmax distribution to keep only a single architecture in memory
- ProxylessNAS [Cai et al, 2019] – computes approximate gradients on α keeping only 2 edges between two nodes in memory at a time
- PC-DARTS [Xu et al, 2020] – performs the search on a subset of the channels in the one-shot model



Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  What is the main difference between DARTS and the other one-shot NAS methods we saw before?
- Repetition:
  How does DARTS optimize the architectural weights and one-shot weights?
- Repetition:
  What are DARTS' main issues and how can they be fixed?
- Discussion:
  RobustDARTS stops the architecture optimization early, before the curvature of the validation loss becomes high. Why do you think this works?
  [Hint: think about the discretization step after the DARTS search.]