

AutoML: Neural Architecture Search

Part 2: One-shot Neural Architecture Search

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

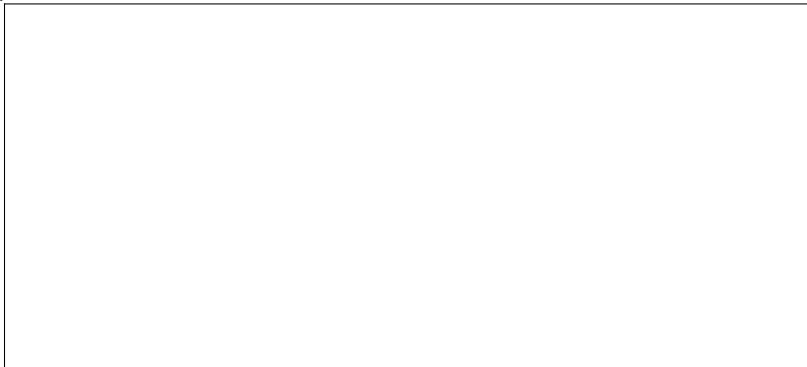
The One-Shot Model

One-shot models: convolutional neural fabrics [Saxena and Verbeek. 2017]

- A **one-shot model** is a big model that has all architectures in a search space as submodels
 - ▶ This allows weights sharing across architectures
 - ▶ One **only needs to train the single one-shot model**,
and implicitly trains an exponential number of individual architectures

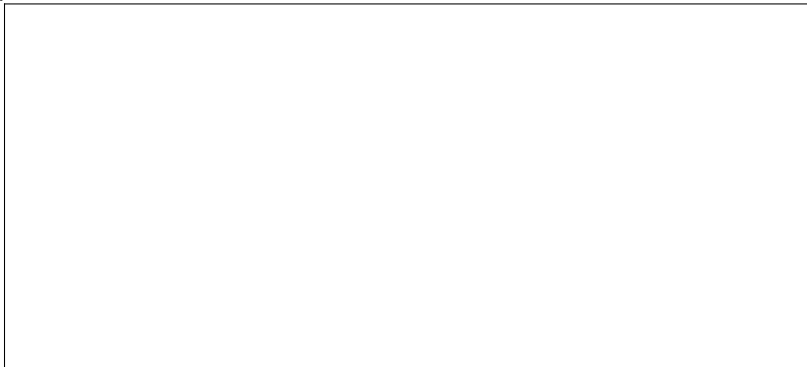
One-shot models: convolutional neural fabrics [Saxena and Verbeek. 2017]

- A **one-shot model** is a big model that has all architectures in a search space as submodels
 - ▶ This allows weights sharing across architectures
 - ▶ One **only needs to train the single one-shot model**,
and implicitly trains an exponential number of individual architectures
- The first type of one-shot models: **convolutional neural fabrics**



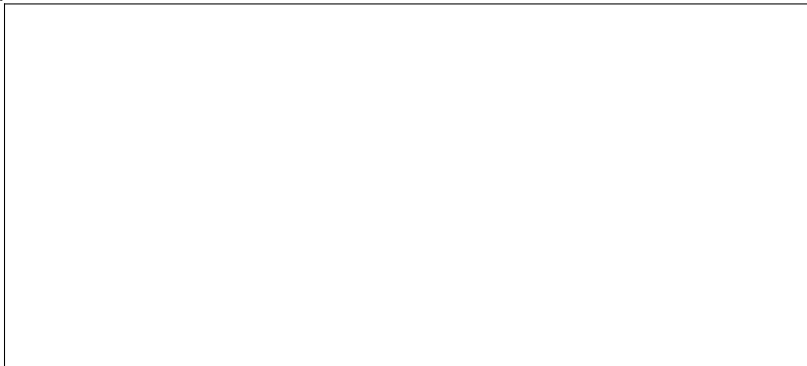
One-shot models: convolutional neural fabrics [Saxena and Verbeek. 2017]

- A **one-shot model** is a big model that has all architectures in a search space as submodels
 - ▶ This allows weights sharing across architectures
 - ▶ One **only needs to train the single one-shot model**,
and implicitly trains an exponential number of individual architectures
- The first type of one-shot models: **convolutional neural fabrics**



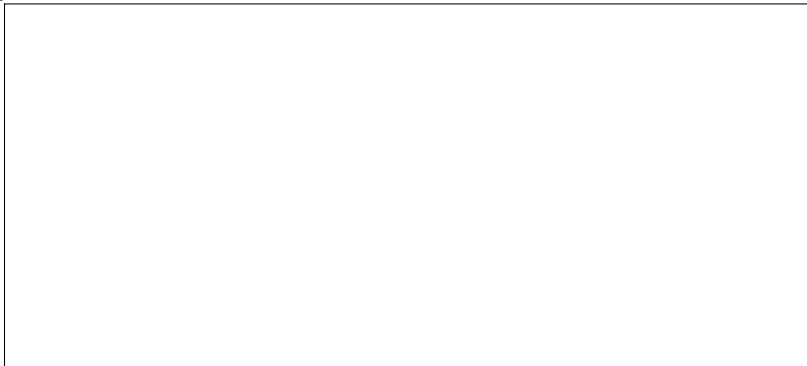
One-shot models: convolutional neural fabrics [Saxena and Verbeek. 2017]

- A **one-shot model** is a big model that has all architectures in a search space as submodels
 - ▶ This allows weights sharing across architectures
 - ▶ One **only needs to train the single one-shot model**,
and implicitly trains an exponential number of individual architectures
- The first type of one-shot models: **convolutional neural fabrics**



One-shot models: convolutional neural fabrics [Saxena and Verbeek. 2017]

- A **one-shot model** is a big model that has all architectures in a search space as submodels
 - ▶ This allows weights sharing across architectures
 - ▶ One **only needs to train the single one-shot model**,
and implicitly trains an exponential number of individual architectures
- The first type of one-shot models: **convolutional neural fabrics**

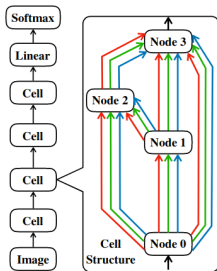


One-shot models: convolutional neural fabrics [Saxena and Verbeek. 2017]

- A **one-shot model** is a big model that has all architectures in a search space as submodels
 - ▶ This allows weights sharing across architectures
 - ▶ One **only needs to train the single one-shot model**,
and implicitly trains an exponential number of individual architectures
- The first type of one-shot models: **convolutional neural fabrics**

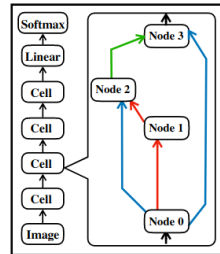
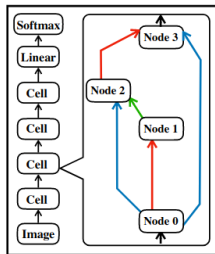
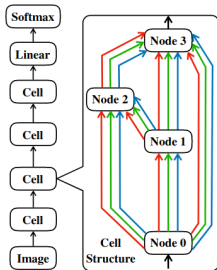
One-shot models for cell search spaces

- Directed acyclic multigraph to capture all (exponentially many) cell architectures
 - ▶ The nodes represent tensors
 - ▶ The edges represent computations (e.g., 3x3 conv, 5x5 conv, max pool, ...)
 - ▶ The results of operations on multiple edges between two nodes are combined (addition/concatenation)



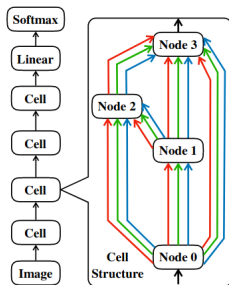
One-shot models for cell search spaces

- Directed acyclic multigraph to capture all (exponentially many) cell architectures
 - ▶ The nodes represent tensors
 - ▶ The edges represent computations (e.g., 3x3 conv, 5x5 conv, max pool, ...)
 - ▶ The results of operations on multiple edges between two nodes are combined (addition/concatenation)
- Individual architectures are subgraphs of this multigraph
 - ▶ Weights for the operation on an edge are shared across all (exponentially many) architectures that have that edge



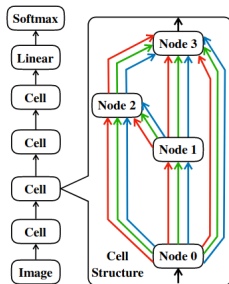
Training the one-shot model – standard SGD [Saxena and Verbeek. 2017]

- One-shot model is an acyclic graph; thus, backpropagation applies
 - ▶ Simplest method: standard training with SGD
 - ▶ This implicitly trains **an exponential number of architectures**



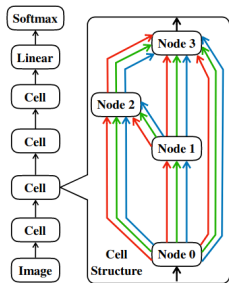
Training the one-shot model – standard SGD [Saxena and Verbeek. 2017]

- One-shot model is an acyclic graph; thus, backpropagation applies
 - ▶ Simplest method: standard training with SGD
 - ▶ This implicitly trains **an exponential number of architectures**
- Potential issue: co-adaptation of weights
 - ▶ Weights are implicitly optimized to work well on average across all architectures
 - ▶ They are **not** optimized specifically for the top-performing architecture



Training the one-shot model – DropPath [Bender et al. 2018]

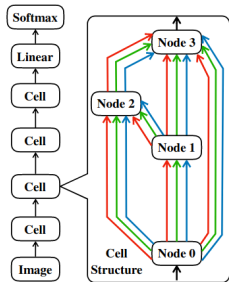
- To avoid coadaptation of weights, we can use **DropPath**, a technique analogous to Dropout [Srivastava et al., 2014]:



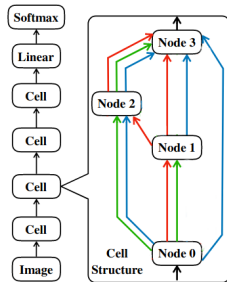
One-shot model

Training the one-shot model – DropPath [Bender et al. 2018]

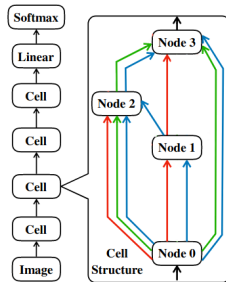
- To avoid coadaptation of weights, we can use **DropPath**, a technique analogous to Dropout [Srivastava et al., 2014]:
 - At each mini-batch iteration:
for each operation connecting 2 nodes, zero it out with probability p



One-shot model



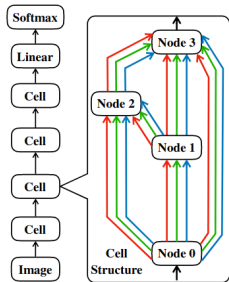
Architecture for batch 1



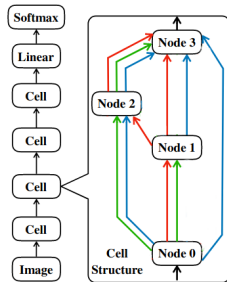
Architecture for batch 2

Training the one-shot model – DropPath [Bender et al. 2018]

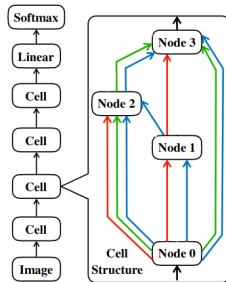
- To avoid coadaptation of weights, we can use **DropPath**, a technique analogous to Dropout [Srivastava et al., 2014]:
 - At each mini-batch iteration:
for each operation connecting 2 nodes, zero it out with probability p
 - **ScheduledDropPath**: starts with $p = 0$ and increases p linearly to p_{max} at the end of training



One-shot model



Architecture for batch 1



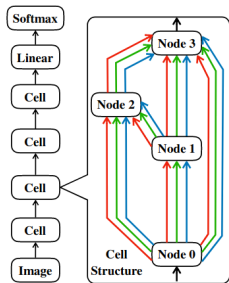
Architecture for batch 2

Training the one-shot model – Sampling

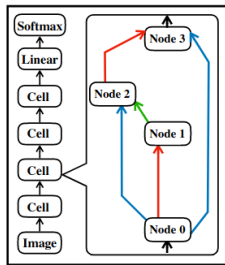
- At each mini-batch iteration during the training of the one-shot model **sample a single architecture** from the search space

Training the one-shot model – Sampling

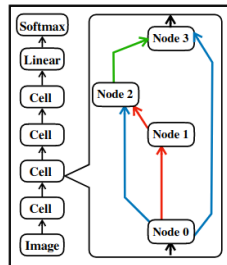
- At each mini-batch iteration during the training of the one-shot model **sample a single architecture** from the search space
- Update the parameters of the one-shot model** corresponding to only that architecture



One-shot model



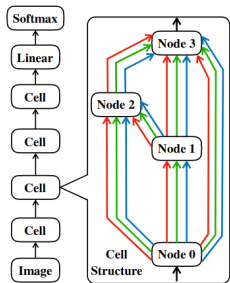
Architecture for batch 1



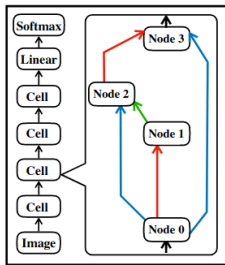
Architecture for batch 2

Training the one-shot model – Sampling

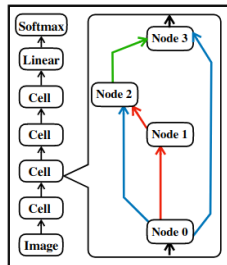
- At each mini-batch iteration during the training of the one-shot model **sample a single architecture** from the search space
 - Random Search with Weight Sharing** [Li and Talwalkar. 2020] → sample from uniform distribution
 - ENAS** [Pham et al. 2018] → sample from the learned policy of a RNN controller
- Update the parameters of the one-shot model** corresponding to only that architecture



One-shot model



Architecture for batch 1



Architecture for batch 2

How to utilize the trained one-shot model?

- After training the one-shot model we have to **select the best individual architecture** from it
- There are multiple ways one can approach this. Some of these are:

How to utilize the trained one-shot model?

- After training the one-shot model we have to **select the best individual architecture** from it
- There are multiple ways one can approach this. Some of these are:
 1. Sample uniformly at random M architectures and rank them based on their validation error **using the one-shot model parameters**

How to utilize the trained one-shot model?

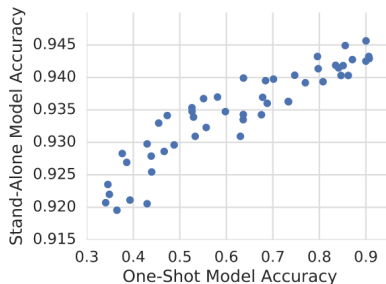
- After training the one-shot model we have to **select the best individual architecture** from it
- There are multiple ways one can approach this. Some of these are:
 1. Sample uniformly at random M architectures and rank them based on their validation error **using the one-shot model parameters**
 - 1b. (Optional) Select top K ($K < M$) and retrain them from scratch for a couple of epochs

How to utilize the trained one-shot model?

- After training the one-shot model we have to **select the best individual architecture** from it
- There are multiple ways one can approach this. Some of these are:
 1. Sample uniformly at random M architectures and rank them based on their validation error **using the one-shot model parameters**
 - 1b. (Optional) Select top K ($K < M$) and retrain them from scratch for a couple of epochs
 2. Return the top performing architecture to **retrain from scratch** for longer

How to utilize the trained one-shot model?

- After training the one-shot model we have to **select the best individual architecture** from it
- There are multiple ways one can approach this. Some of these are:
 1. Sample uniformly at random M architectures and rank them based on their validation error **using the one-shot model parameters**
 - 1b. (Optional) Select top K ($K < M$) and retrain them from scratch for a couple of epochs
 2. Return the top performing architecture to **retrain from scratch** for longer
- **Pitfall:** the correlation between architectures evaluated with the one-shot weights and retrained from scratch (stand-alone models) should be high
- If not, **selecting the best architecture based on the one-shot weights** is sub-optimal.



From [Bender et al. 2018]

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
How are the weights shared in the one-shot model?
- Repetition:
What is the difference between Random Search with Weight Sharing and ENAS?
- Discussion:
What might be some downsides of using the one-shot model for NAS?