IACS | HARVARD School of Engineering and Applied Sciences

Investigating Differentiable Neural Architecture Search

Harvard Data Science Capstone (Fall 2019) Final Presentation

Team

Michael S. Emanuel
Julien Laasri
Dylan Randle
Jiawei Zhuang

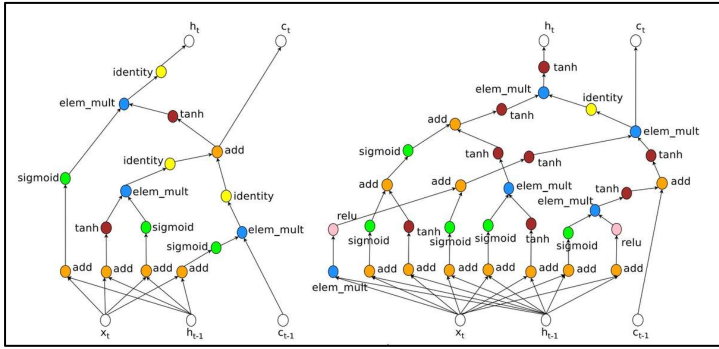
Outline

1. Problem & Motivation
    a. Neural Architecture Search
    b. Datasets
2. Introduction to DARTS (Differentiable Neural Architecture Search)
3. Results
    a. MNIST
    b. Graphene
    c. Galaxy Zoo
    d. Chest X-Ray
4. Conclusions & Future Work

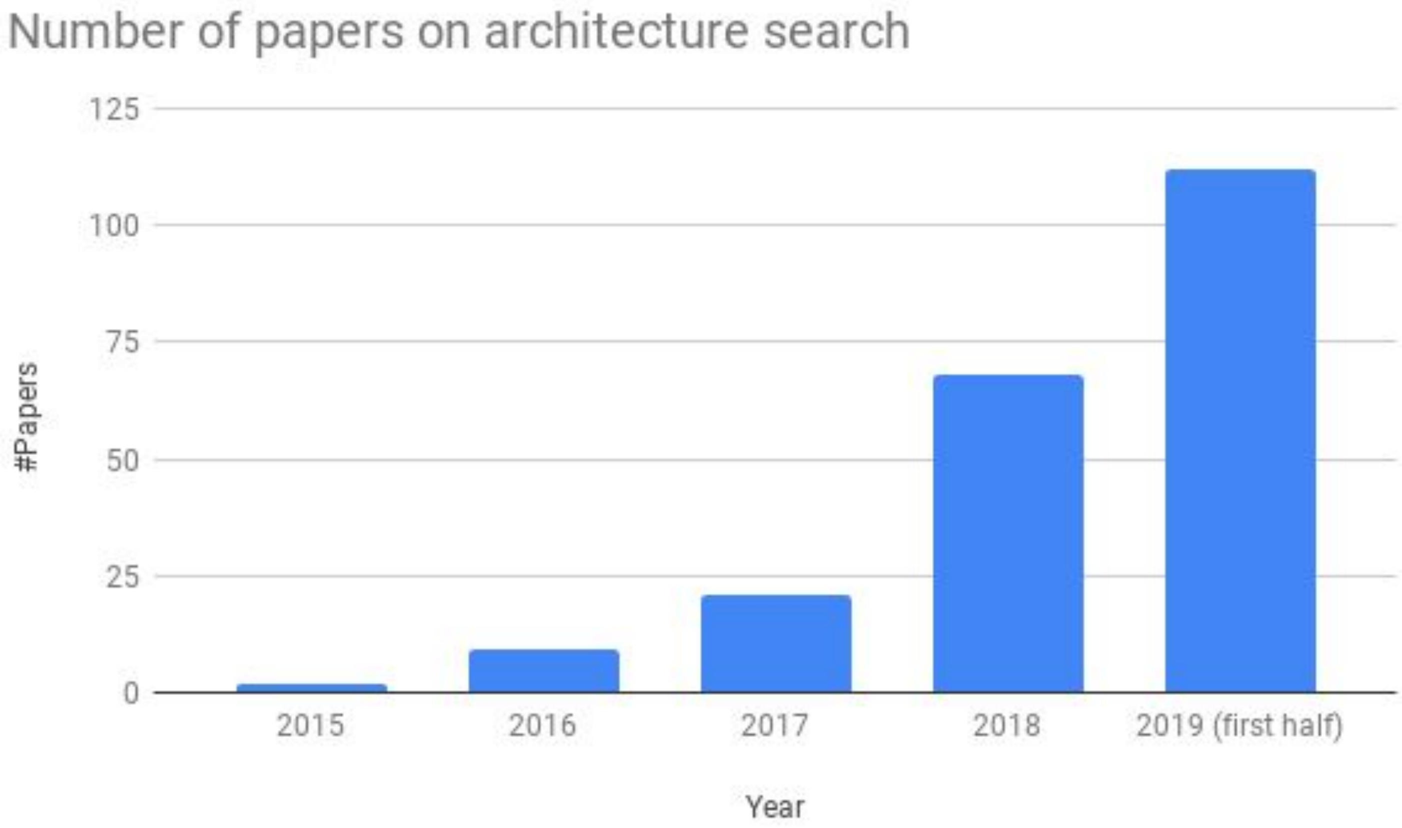
Problem & Motivation

What is Neural Architecture Search?

- Deep learning frees us from feature engineering, but has led us to spend valuable time on **architecture engineering**
- Today: designed by experts
  - Labor-intensive
- Tomorrow: **Neural Architecture Search (NAS)**
  - Automatically find best architecture
- Interest in NAS is increasing rapidly: there is now far more demand for neural network models than available experts who can design model architectures

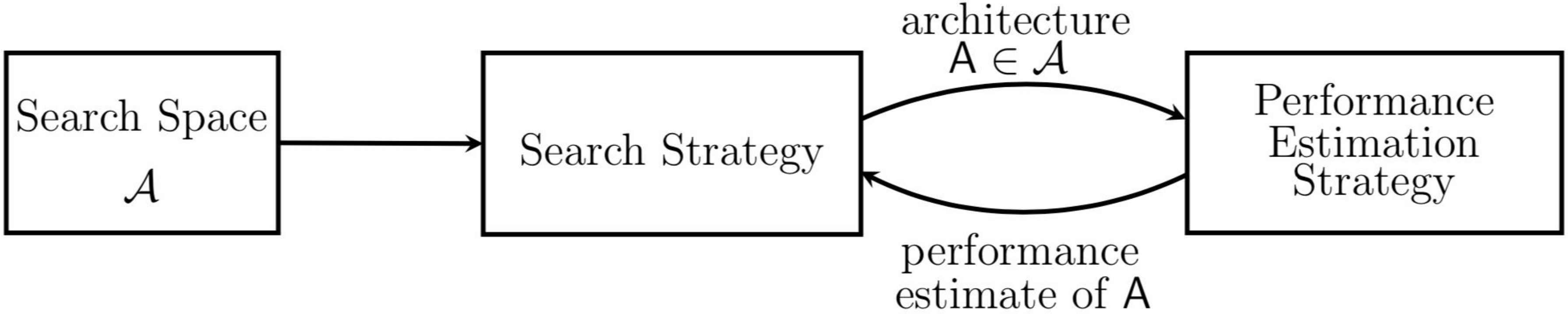


LSTM vs learned recurrent cell using reinforcement learning approach



NAS papers per year based on the literature list on automl.org.
The number for 2019 only considers the first half of 2019.
(Lindauer and Hutter, 2019)

Neural Architecture Search Workflow



Credit: Elskin et. al, 2019

NAS can be very expensive

Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$\text{Transformer}_{base}$	P100x8	1415.78	12	27	26	\$41–\$140
Transformer_{big}	P100x8	1515.43	84	201	192	\$289–\$981
ELMo	P100x3	517.66	336	275	262	\$433–\$1472
BERT_{base}	V100x64	12,041.51	79	1507	1438	\$3751–\$12,571
BERT_{base}	TPUv2x16	—	96	—	—	\$2074–\$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	\$942,973–\$3,201,722
NAS	TPUv2x1	—	32,623	—	—	\$44,055–\$146,848
GPT-2	TPUv3x32	—	168	—	—	\$12,902–\$43,008

Table 3: Estimated cost of training a model in terms of CO_2 emissions (lbs) and cloud compute cost (USD).[7] Power and carbon footprint are omitted for TPUs due to lack of public information on power draw for this hardware.

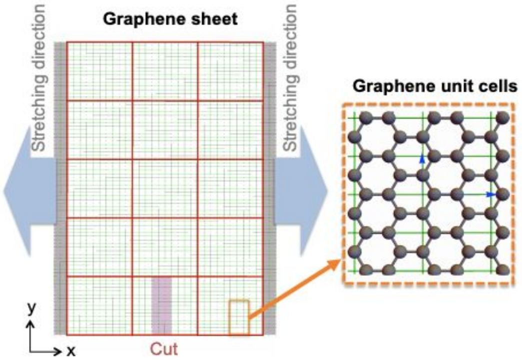
NAS cost is from evolutionary architecture search on Transformer (Strubell et al. 2019)

Scientific Datasets

Most NAS studies only use standard image datasets like CIFAR and ImageNet. However, deep learning also shows large potential for various physical sciences. Thus we want to see whether DARTS is useful for scientific datasets.

- ❑ **MNIST:** classifying images of handwritten digits (non-scientific baseline)
- ❑ **Graphene Kirigami:** cutting graphene to optimize stress/strain
- ❑ **Galaxy Zoo:** classifying galaxy morphology from telescope images
- ❑ **Chest X-Ray:** predicting diseases from chest x-rays

Not “scientific” but good “hello world”



Introduction to DARTS

(Differentiable Neural ARchiTecture Search)

Issues with brute-force or traditional Neural Architecture Search (NAS) approaches

Proposed solutions in DARTS (Differentiable ARchiTecture Search, Liu 2019, ICLR)

Extremely large search space: arbitrary connections and operations between neural network nodes

→ Only search for the optimal "cell", i.e. a small unit of convolutional layers. Construct the complete model by stacking identical cells. (following NASNet, Zoph 2018, Google Brain)

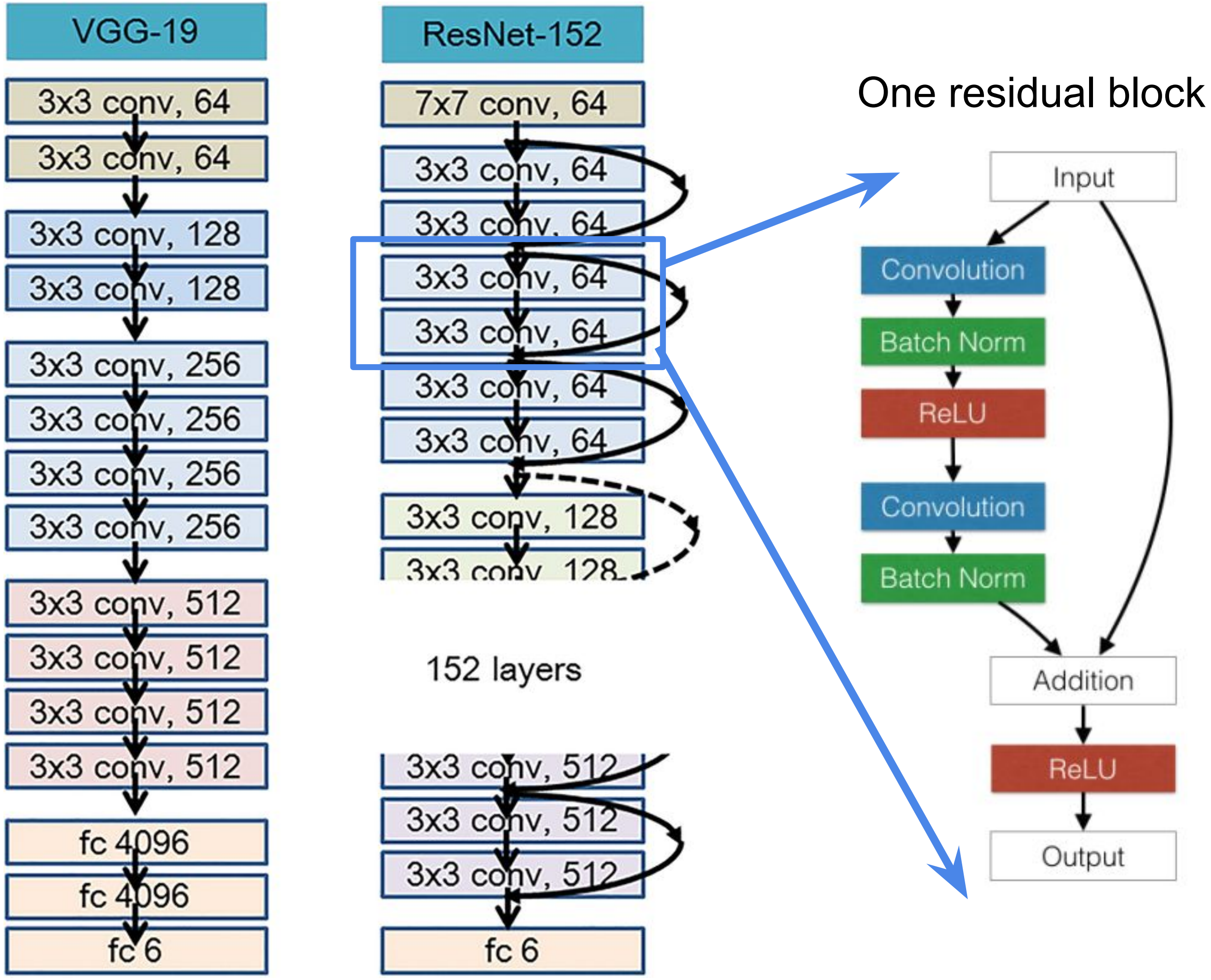
Every "trial architecture" is re-trained from scratch, taking many GPU hours

→ Share weights/parameters between child models, which can be "trained together" (similar to ENAS, Pham 2018, Google Brain)

The choice of operations (e.g. Conv, Pooling) is discrete, requiring expensive optimization

→ Continuous relaxation: parameterize the choice of operation by "architecture parameter" α, which can be optimized by gradient descent

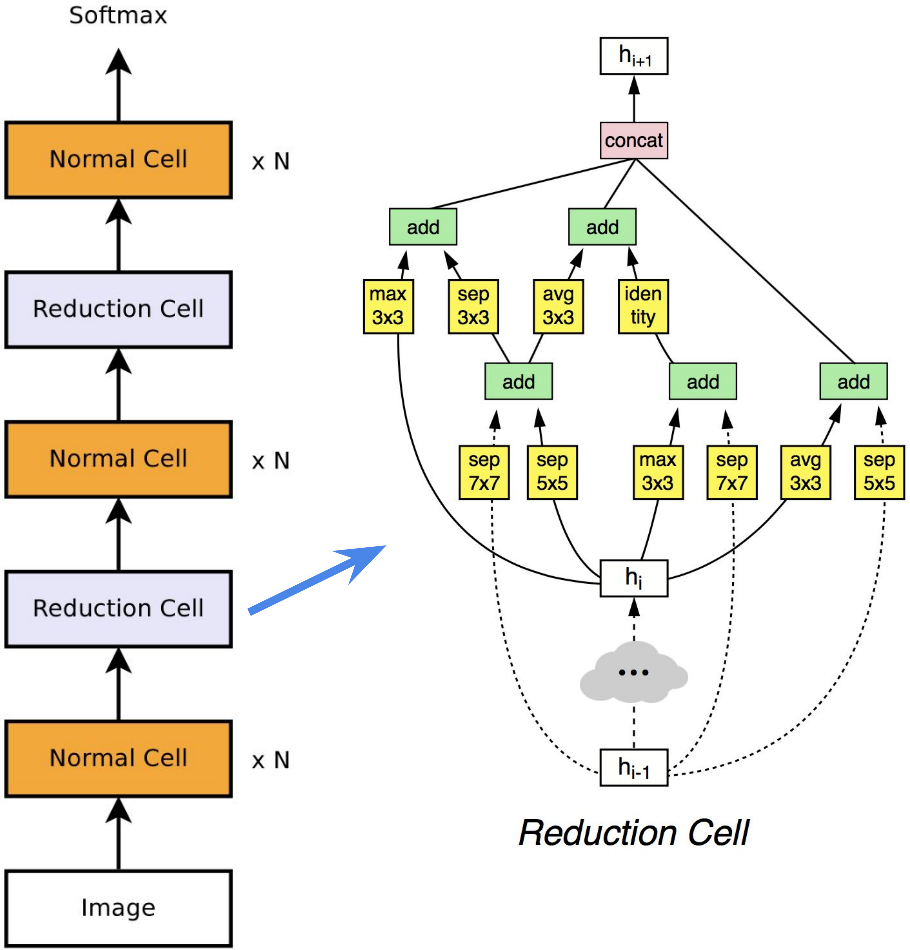
Key observation: popular CNN architectures often contain repeating blocks, stacked sequentially



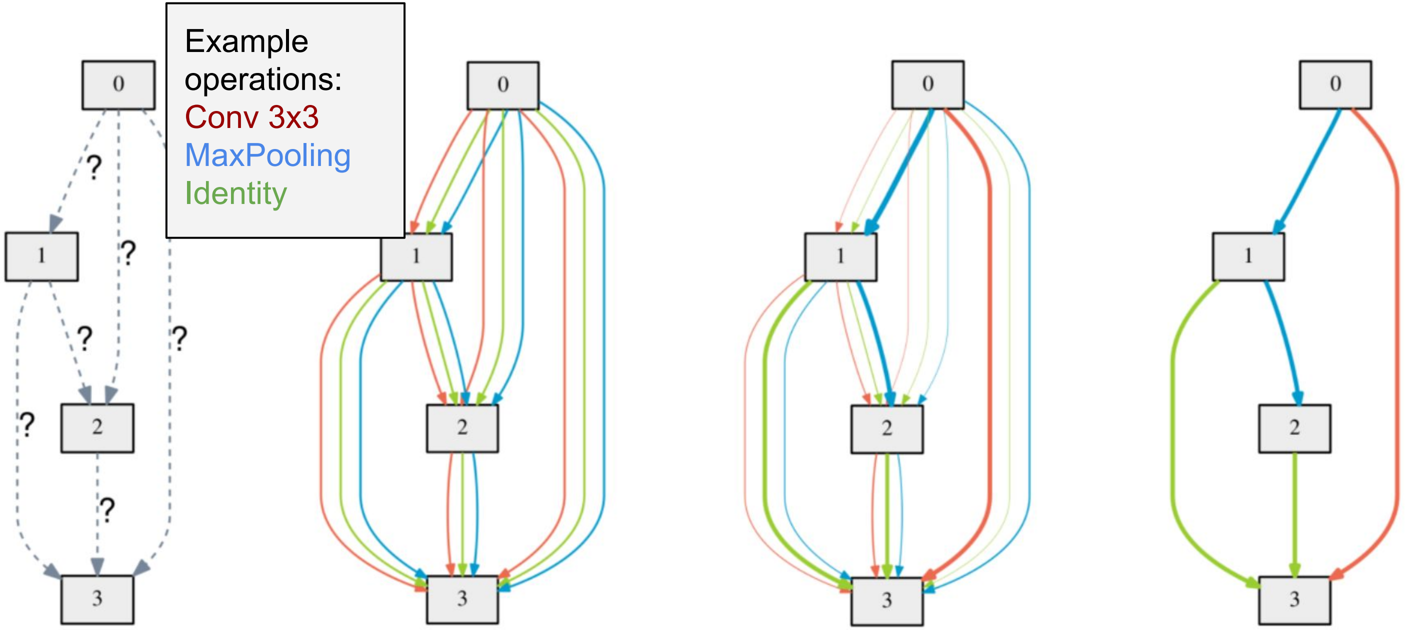
DARTS searches for the optimal "cell", not whole model

- Two types of cells:
  - **Normal Cell:** output same dimension
  - **Reduction Cell:** output half dimension
- Stack cells sequentially to form model
- Each cell type **share the same architecture but have independent weights**

(following NASNet, Zoph 2018, Google Brain)



Continuous relaxation of discrete operations enables gradient descent



(a)

Goal: Find the optimal cell, by placing proper operations (e.g. conv, pooling) at edges

(b)

Superpose: each edge is the sum over the outputs of multiple operations, weighted by continuous "architecture parameters" α

(c)

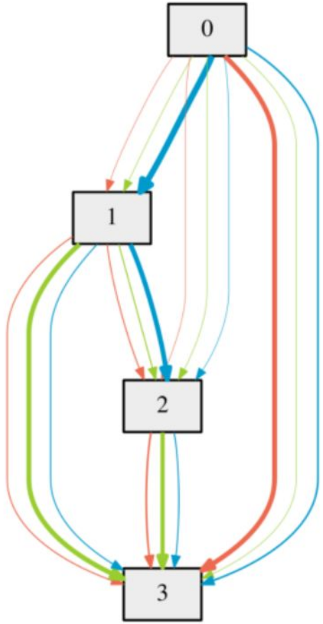
Search: Optimize the architecture weights α, using gradient descent on validation loss

(d)

Discretize: select the operation with the highest architecture weight, to be the final architecture

Gradient-based optimization for architecture parameter α

Example operations:
Conv 3x3
MaxPooling
Identity



The actual operation at edge (i, j) is the average of all candidate operations o(x), weighted by α:

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

With a certain choice of architecture weight α, the corresponding architecture can be in principle trained to convergence, leading to the optimal model weights w*(α) and the final validation loss $L_{val}(w^*(\alpha), \alpha)$.

The gradient of L_{val} w.r.t to α gives the direction for gradient descent!

One-shot evaluation to avoid re-training

Computing the true loss L_{val} by training w to the end is too expensive; thus DARTS just trains w for one step to get a proxy loss:

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$$

where the optimal model weights w*(α) is approximated by the one-step training

The training of α and w is performed in an alternate way:

while *not converged* **do**
1. Update architecture α by descending $\nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$ ($\xi = 0$ if using first-order approximation)
2. Update weights w by descending $\nabla_w \mathcal{L}_{train}(w, \alpha)$

Results

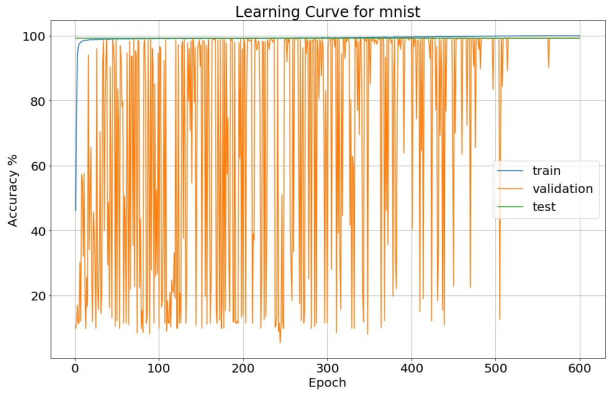
Overview

Model	MNIST	Graphene	Galaxy Zoo	Chest X-Ray
DARTS (Continuous)	99.07	0.89	**0.094**	**0.157**
DARTS (Discrete)	99.27	**0.92**	0.114	0.163
Random Search	99.31	0.90	0.098	0.169
ResNet	**99.40**	**0.92**	0.095	0.163
Metric	Accuracy	R^2	RMSE	BCE

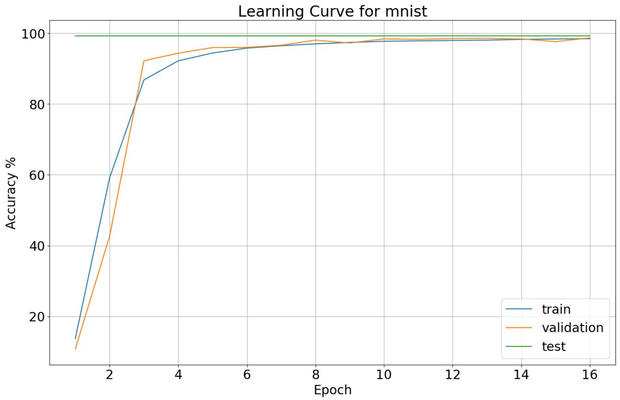
Results: MNIST

- Training MNIST with default learning rate fails
- Had to tune learning rate
- Key point: even on easy problem, DARTS is sensitive to hyperparameters

Default Hyper-parameters



Tuned Hyper-parameters



Model	MNIST
DARTS (Continuous)	99.07
DARTS (Discrete)	99.27
Random Search	99.31
ResNet	**99.40**
Metric	Accuracy

Large number of hyperparameters (~2x standard)

```
[--layers LAYERS] [--learning_rate LEARNING_RATE]
[--learning_rate_min LEARNING_RATE_MIN]
[--weight_decay WEIGHT_DECAY] [--L1_lambda L1_LAMBDA]
[--arch_learning_rate ARCH_LEARNING_RATE]
[--arch_weight_decay ARCH_WEIGHT_DECAY] [--cell_steps CELL_STEPS]
[--cell_multiplier CELL_MULTIPLIER] [--epochs EPOCHS]
[--batch_size BATCH_SIZE] [--optimizer OPTIMIZER]
[--momentum MOMENTUM] [--gz_dtree] [--fc1_size FC1_SIZE]
[--fc2_size FC2_SIZE] [--primitives PRIMITIVES]
[--train_portion TRAIN_PORTION] [--grad_clip GRAD_CLIP]
[--unrolled] [--cutout] [--cutout_length CUTOUT_LENGTH]
```

Train DARTS search

```
[--init_channels INIT_CHANNELS] [--layers LAYERS]
[--learning_rate LEARNING_RATE] [--drop_path_prob DROP_PATH_PROB]
[--weight_decay WEIGHT_DECAY] [--arch ARCH] [--epochs EPOCHS]
[--batch_size BATCH_SIZE] [--optimizer OPTIMIZER]
[--momentum MOMENTUM] [--fc1_size FC1_SIZE] [--fc2_size FC2_SIZE]
[--gz_dtree] [--primitives PRIMITIVES]
[--train_portion TRAIN_PORTION] [--grad_clip GRAD_CLIP]
[--auxiliary] [--auxiliary_weight AUXILIARY_WEIGHT] [--cutout]
[--cutout_length CUTOUT_LENGTH]
```

Train discretized

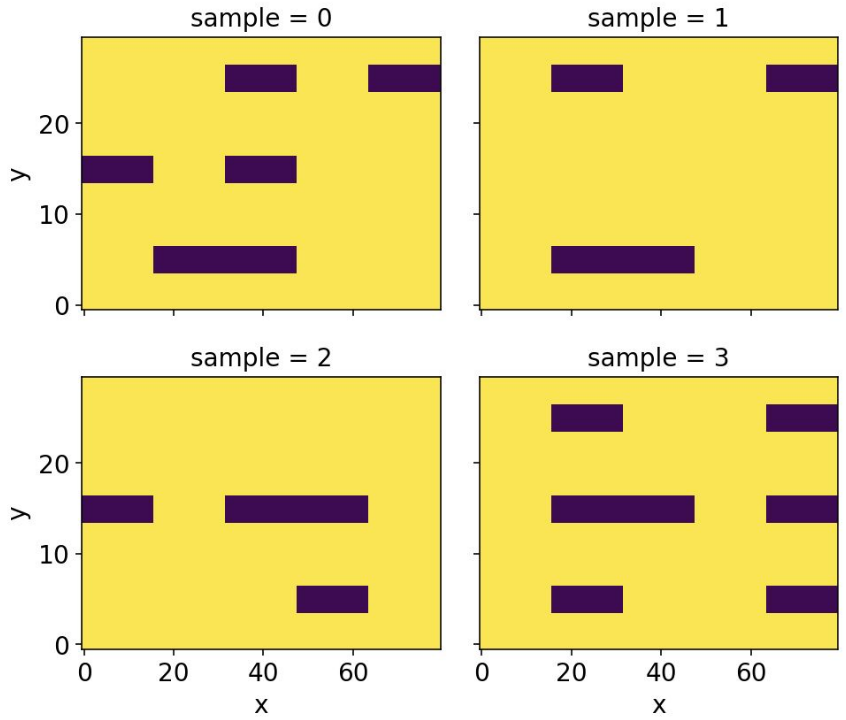
Problem: Graphene Kirigami

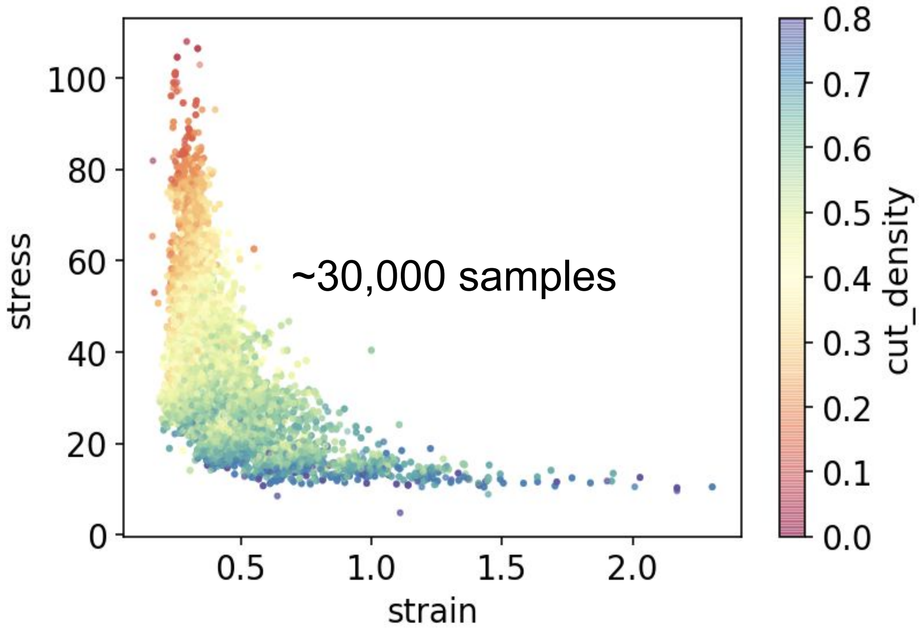
Hanakata et al. 2018, PRL

cut configuration

predict

stretch property





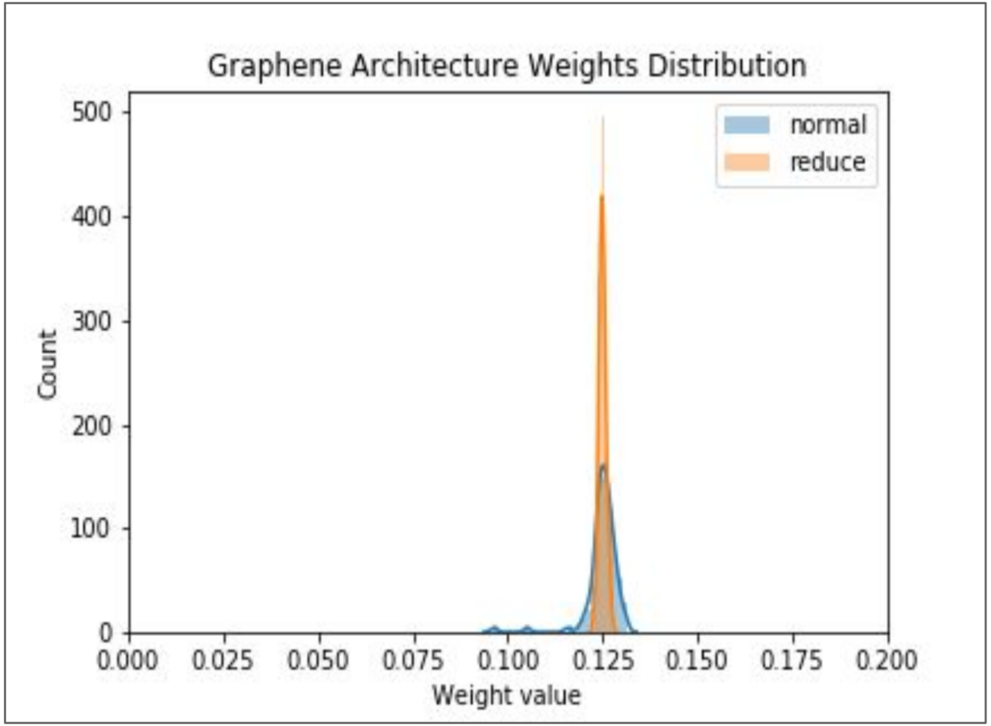
Results: Graphene Kirigami

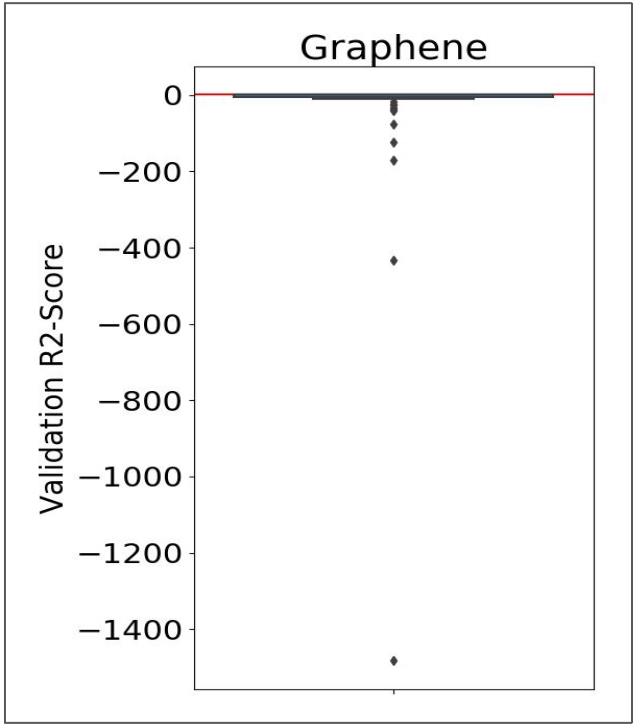
Method / model	No. Parameters	Training time* (minutes)	Test R^2
DARTS	195,236	1042	0.92
ResNet-18	11,168,193	11	0.92
"Tiny ResNet" (10 layers, 8x less filters)	77,273	4	0.92

1. Time reported to train for 30 epochs on a single GPU.

Conclusion: Graphene Kirigami dataset is too simple for DARTS to be useful.

Graphene Architecture & Random Search

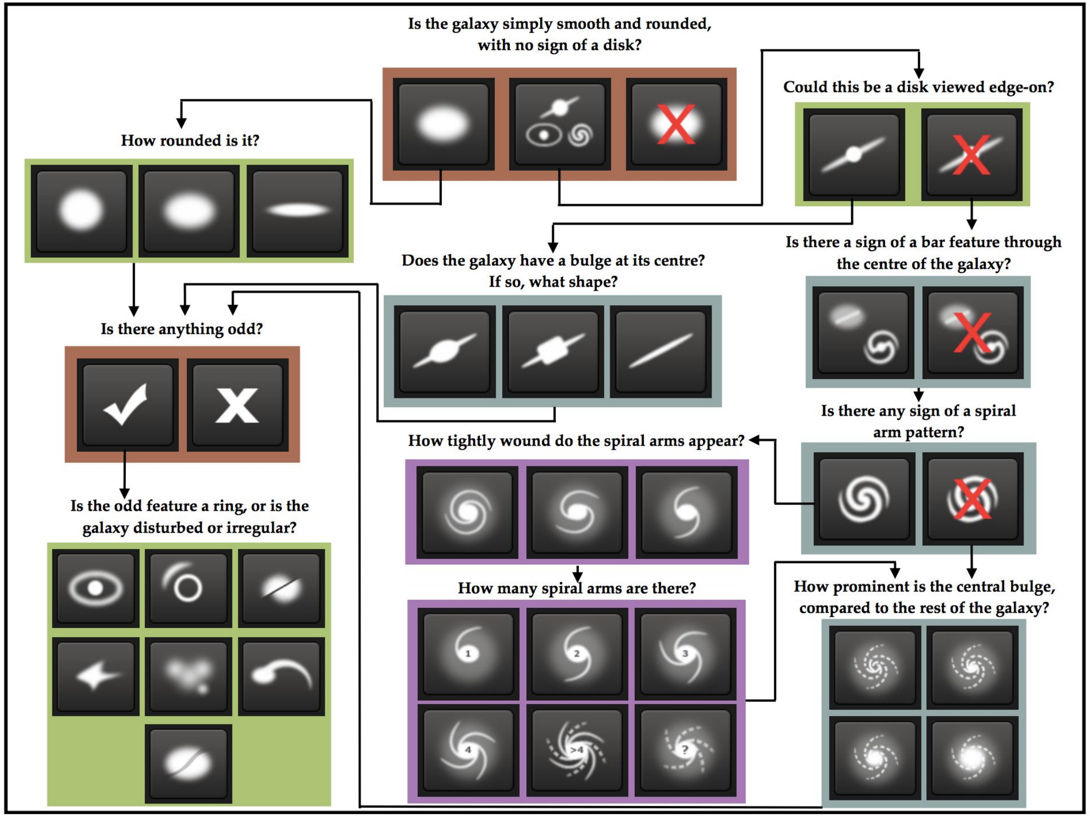




Simple problem → many good architectures → low sparsity in architecture weights

Problem: Galaxy Zoo

- Predict galaxy morphology from images
- 37 distinct binary attributes
- Scored on root mean squared error (RMSE)

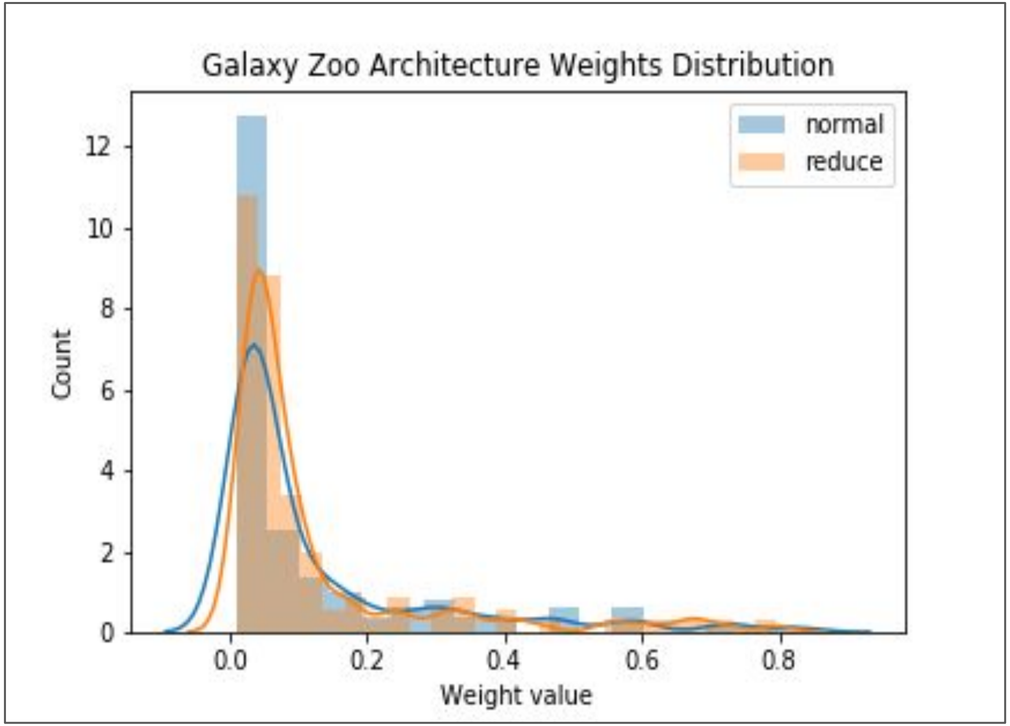


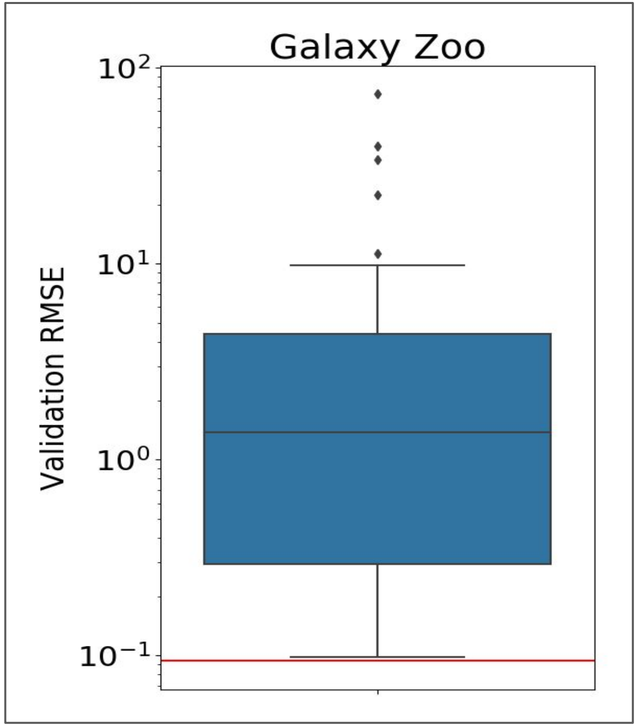
Results: Galaxy Zoo

- Continuous DARTS better than ResNet
- “Discretized” architecture is **worse**
- Shows heuristic discretization step can fail
- Kaggle winner score is ~0.075 (with extensive data augmentation and model ensembling)

Model	Galaxy Zoo
DARTS (Continuous)	**0.094**
DARTS (Discrete)	0.114
Random Search	0.098
ResNet	0.095
Metric	RMSE

Galaxy Architecture & Random Search

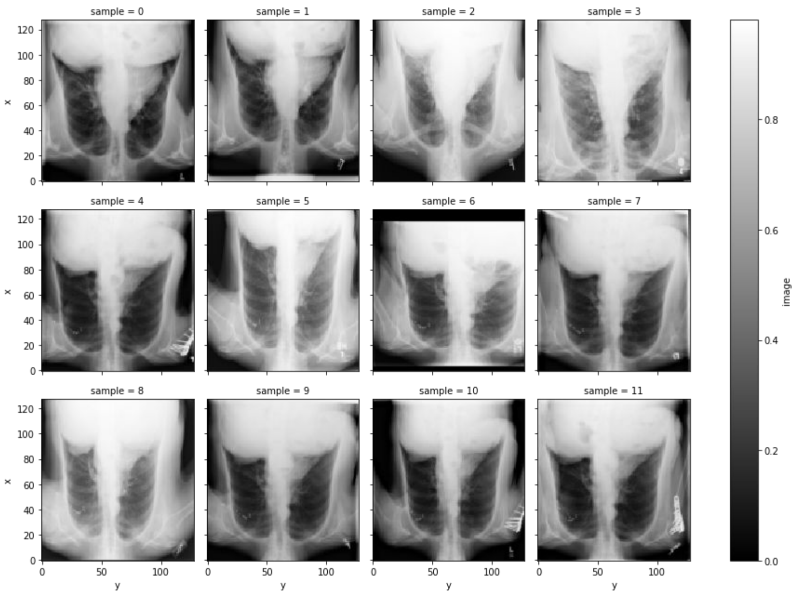




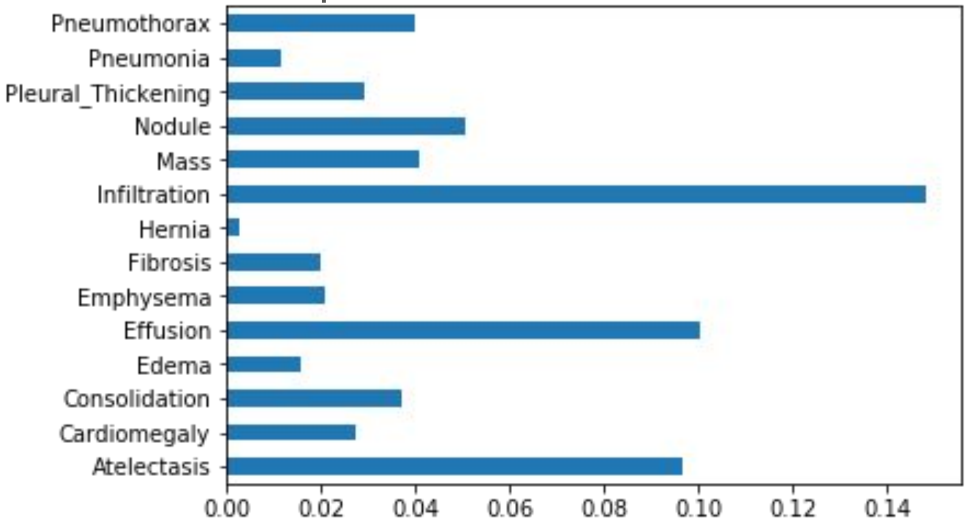
Large variance in architectures → sparse cell learned by DARTS

Problem: Chest X-Ray

- 39,589 chest X-rays
- 14 independent disease labels (confirmed diagnoses)
- Models assessed with mean binary cross entropy (BCE)



Proportion of each disease

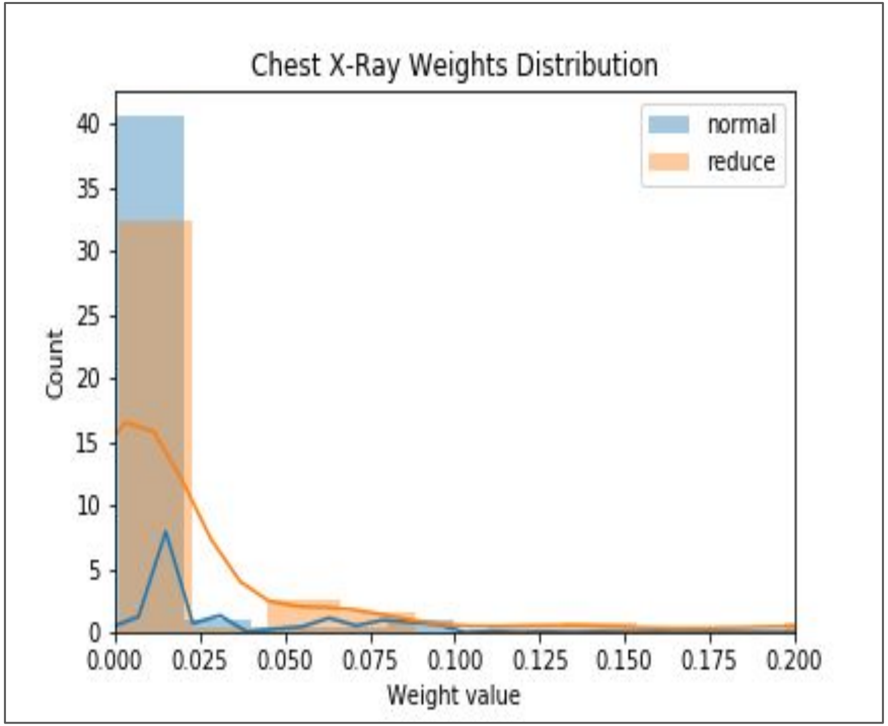


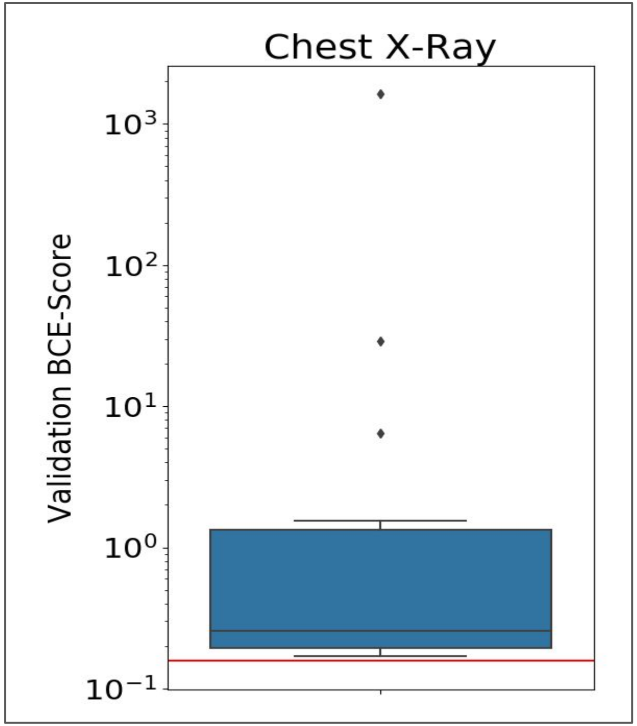
Results: Chest X-Ray

- DARTS performs well after some hyperparameter tuning
- The discretized network was worse (same as ResNet)
- Discretization step failed again

Model	Chest X-Ray
DARTS (Continuous)	**0.157**
DARTS (Discrete)	0.163
Random Search	0.169
ResNet	0.163
Metric	BCE

Chest X-Ray Architecture & Random Search





Again: large variance in architectures → sparse cell learned by DARTS

Conclusions & Future Work

Conclusions

- DARTS a useful tool, but overkill on simple tasks
- ResNet and random search could be good enough
- DARTS introduces additional hyperparameters (~2x regular)
- DARTS "discretization step" is heuristic and can fail
- Computational cost: ~10x more expensive than single (discrete) model, due to overlapping 10 ops. Batch size reduced by ~10x due to memory footprint

Recommendation

- If small increase in performance important, DARTS worthwhile

Future Work

- Automatic tuning and/or better defaults for hyperparameters
- Fix discretization heuristic, a crucially overlooked step (or eliminate it)
    - Encourage sparsity with sparsemax in place of softmax, or L_p regularization on the architecture weights
    - Dynamically prune architecture to remove components during training, eliminate need to re-train

Thank you for a great semester!