Applied Artificial Intelligence

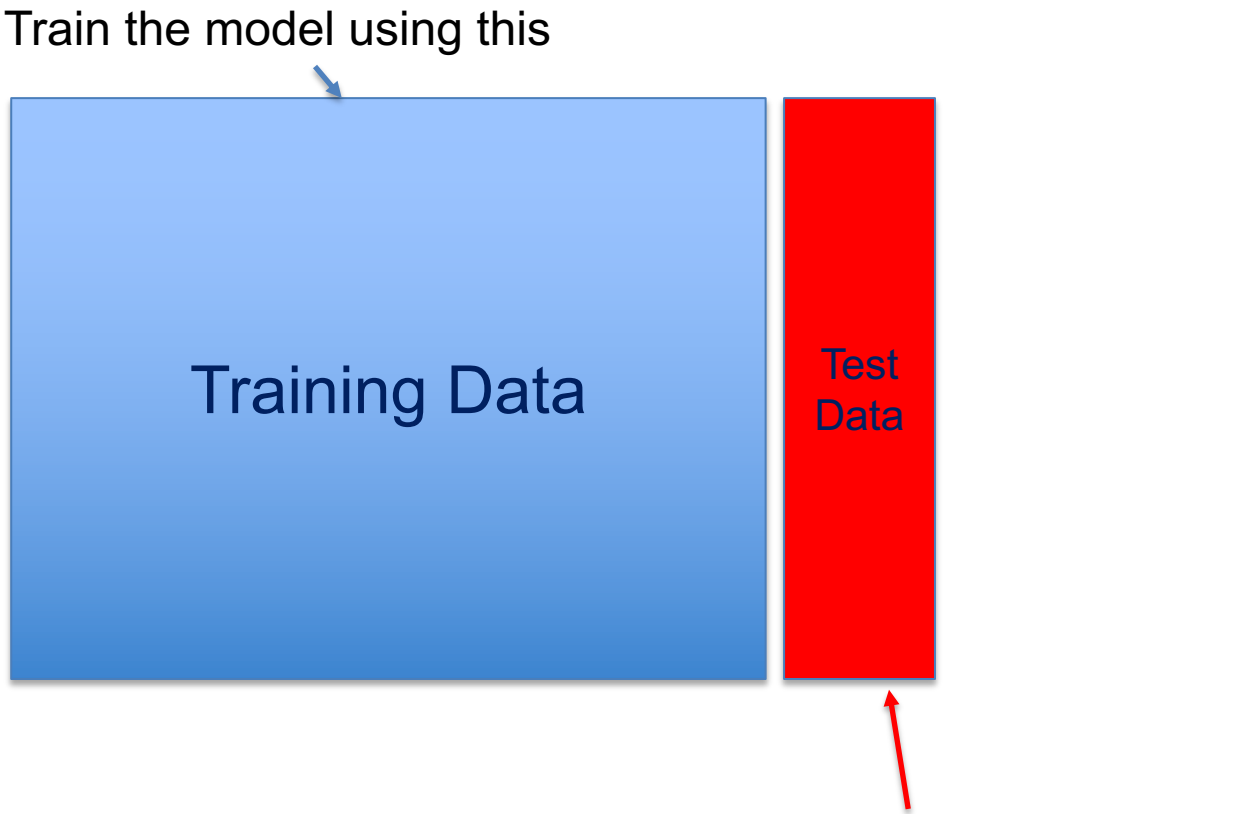
Session 14: Feedforward Multilayer Neural Networks Design and Training

Fall 2018

NC State University
Lecturer: Dr. Behnam Kia

Course Website: https://appliedai.wordpress.ncsu.edu/


Data


Train the model using this
Training Data
Test Data
Test the trained model using this

Designing Neural Network

- How many hidden layers?
- How many neurons in each hidden layer?
- What should be the activation functions?

.

.

.

Designing Neural Network

- How many hidden layers?
- How many neurons in each hidden layer?
- What should be the activation functions?

.
.
.

- We call these variables hyperparameters – opposed to parameters (*W*) that the model learns during training.
- These hyperparameters need to be set before the training starts. But how?

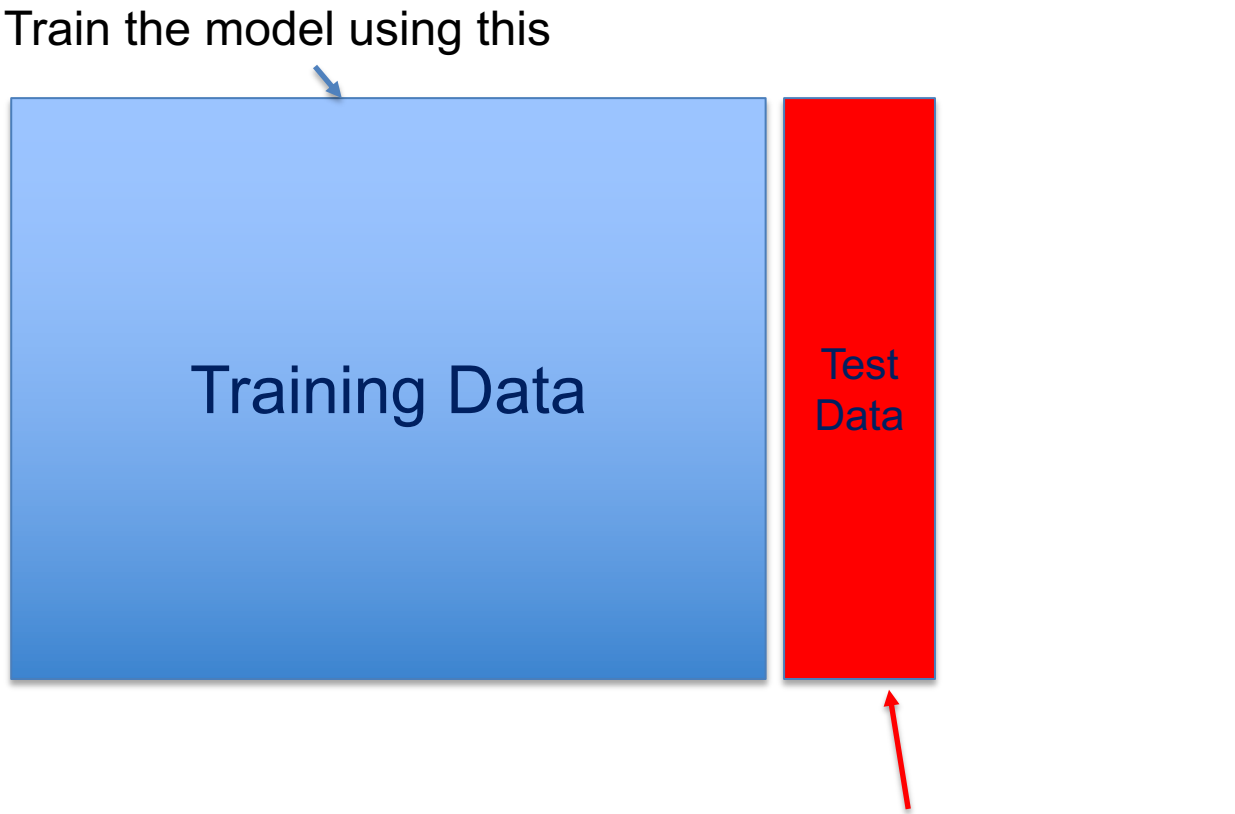
What do you think about this design method to tune hyperparameters?

For ***h*** in range (all possible hyperparameter values):

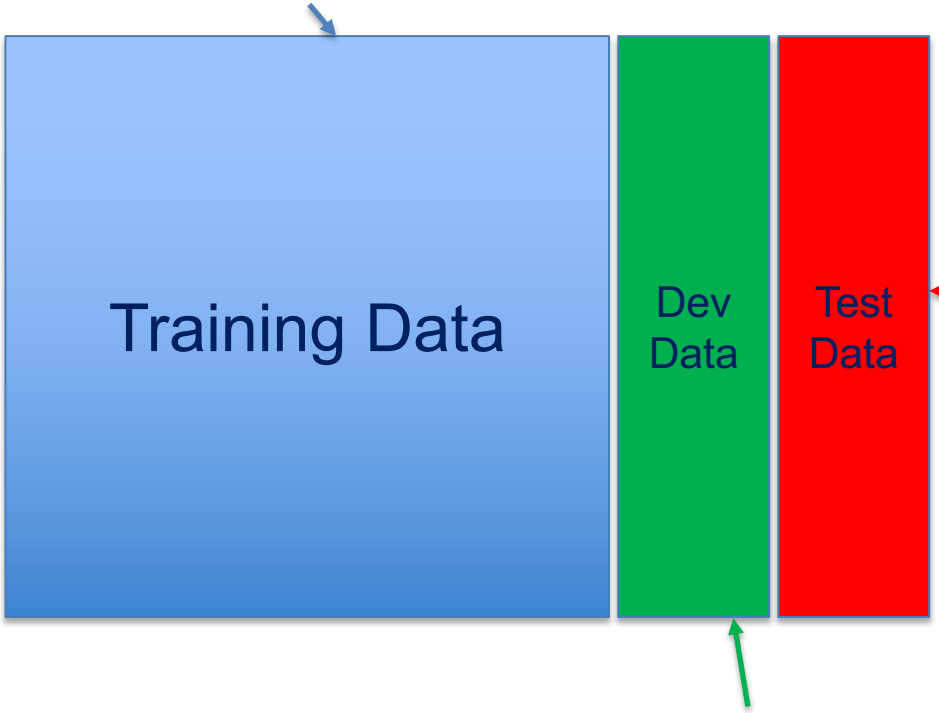
Train the model with ***h***

Test the trained model on the test data & keep scores

Use ***h*** value that results in the best score when applied on the test data.


Train the model using this
Training Data
Test Data
Test the trained model using this

Train the model using this data for different *h* values.



Test the trained model with optimal h using this

Take the model with *h* value that performs the best on this data.

- **Training set:** Which you run your learning algorithm on.
- **Development set (AKA Validation set):** Which you use to tune parameters, select features, and make other decisions regarding the learning algorithm.
- **Test set:** which you use to evaluate the performance of the algorithm, but not to make any decisions regarding what learning algorithm or parameters to use.

Sizes

- **Training set:** As much data as we can get.
- **Development (Validation) set:** Large enough to capture the statistical performance of the classifier. A development set of 100 examples cannot measure or detect performance improvement of 0.1%.
- **Test Set:** large enough to give high confidence in the overall performance of your system.

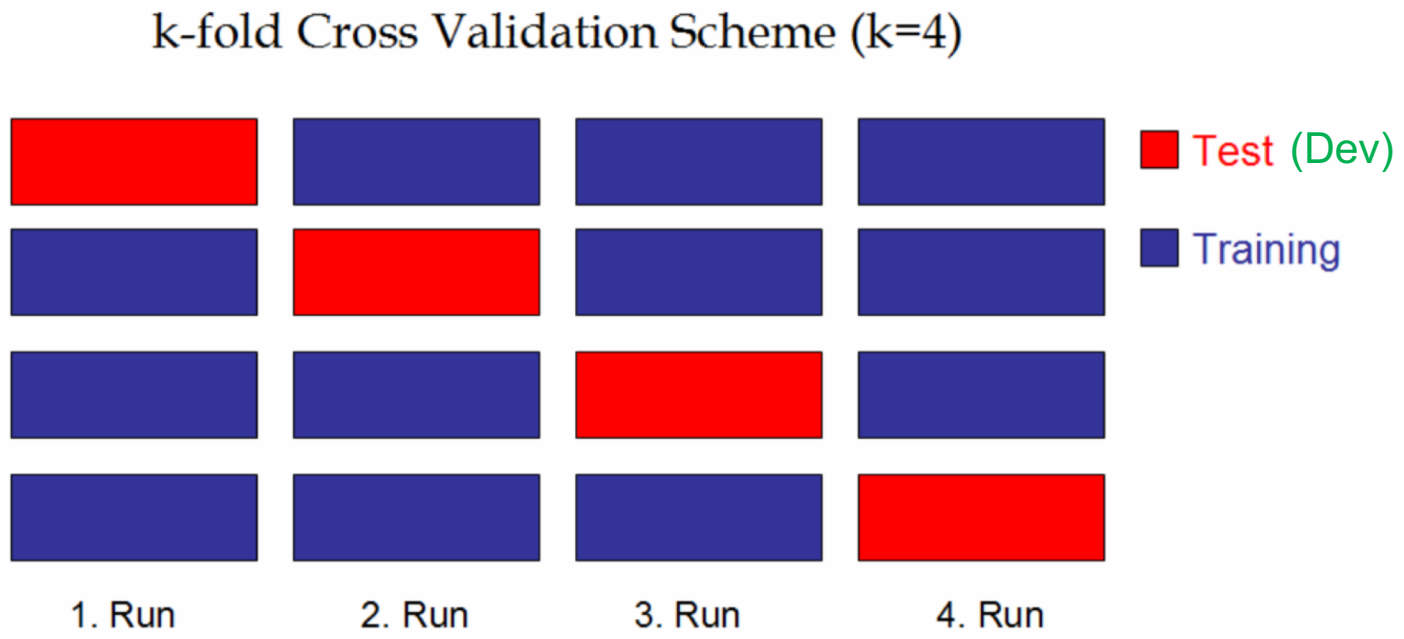
Sizes

- **Common Rule of thumb (classic ML)**
  - **Training set and Development set:** 70-80%
    - 70-80% goes for training, and the rest for development.
  - **Test Set:** 30-20% of data

- In the era of big data, we see allocation of a higher percentage of data to training, 90% and even higher.

K-Fold Cross Validation

- After dividing the data to two parts, and training on one and testing or validating on the second, we can then train on the second section and test or validate on the first (2-fold cross validation).



Picture from: H. Bisgin, et. al. Diagnosis of long QT syndrome via support vector machines classification

But how to search the hyperparameter space *H* to find the optimal *h?*

- This is a search mechanism question.

But how to search the hyperparameter space *H* to find the optimal *h?*

- Grid search (exhaustive search): we try a select possible (every possible) combination of hyperparameters.

# hidden layers:	2	3	4

But how to search the hyperparameter space *H* to find the optimal *h?*

- Grid search (exhaustive search): we try a select possible (every possible) combination of hyperparameters.

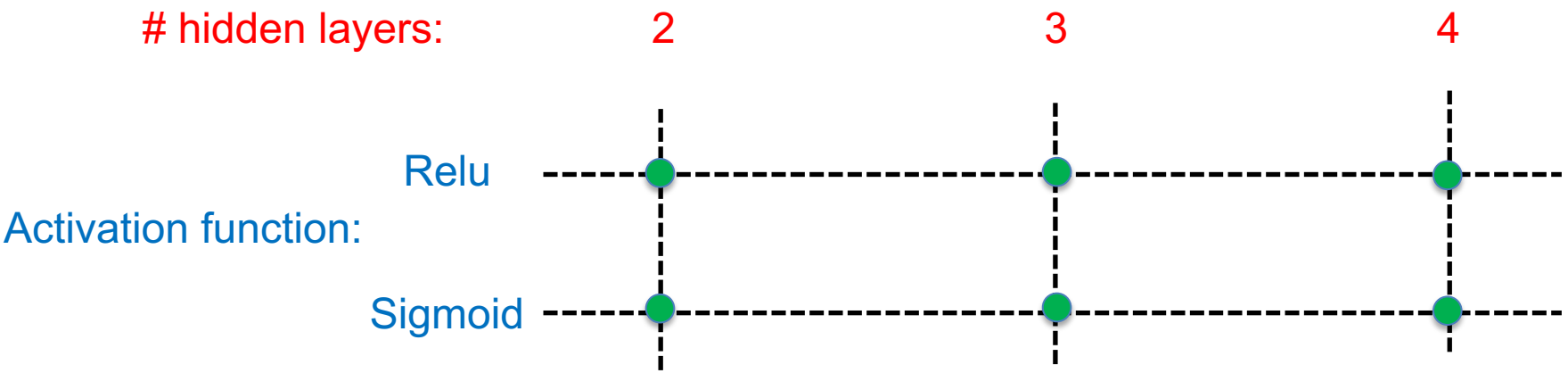
hidden layers: 2 3 4

Activation function: Relu

Sigmoid

But how to search the hyperparameter space *H* to find the optimal *h?*

- Grid search (exhaustive search): we try a select possible (every possible) combination of hyperparameters.



Tuning hyperparameters with development data

For ***h*** in range (all possible hyperparameter values):

Train the model with ***h***

Test the trained model on the test data & keep scores

Use ***h*** value that results in the best score when applied on the development data.

Test the resulting model on the test data.

But how to search the hyperparameter space *H* to find the optimal *h?*

- Grid search (exhaustive search): we try a select possible (every possible) combination of hyperparameters.
- Random search.

But how to search the hyperparameter space *H* to find the optimal *h*?

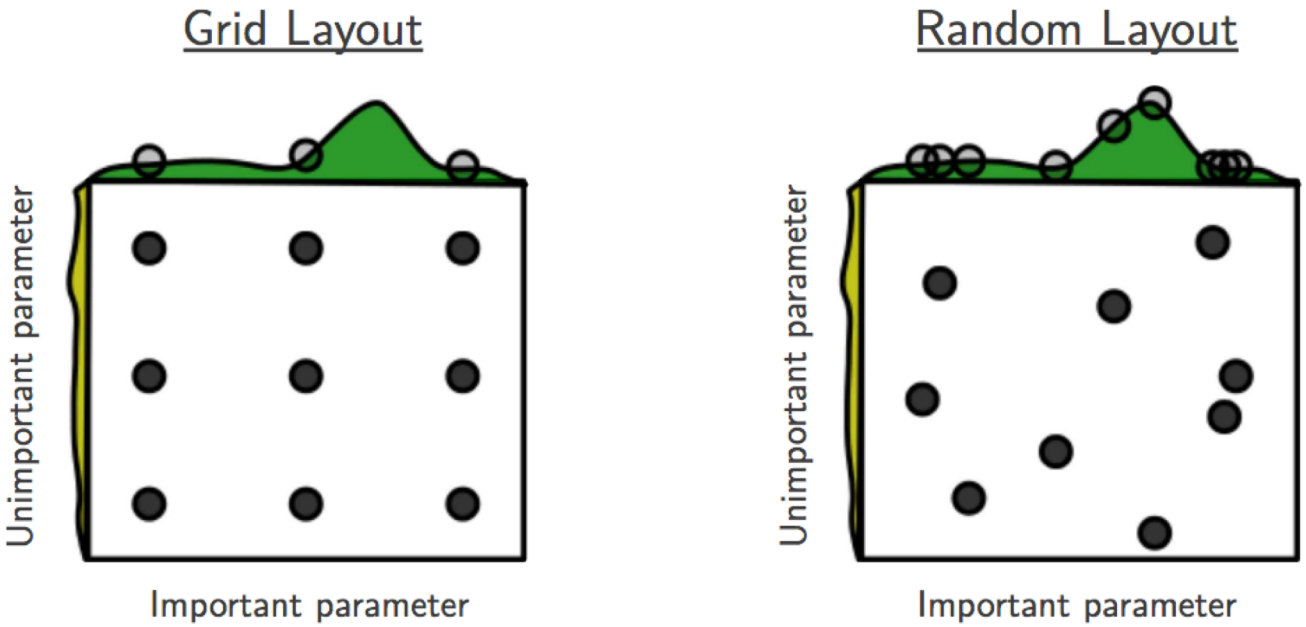
Journal of Machine Learning Research 13 (2012) 281-305 Submitted 3/11; Revised 9/11; Published 2/12

Random Search for Hyper-Parameter Optimization

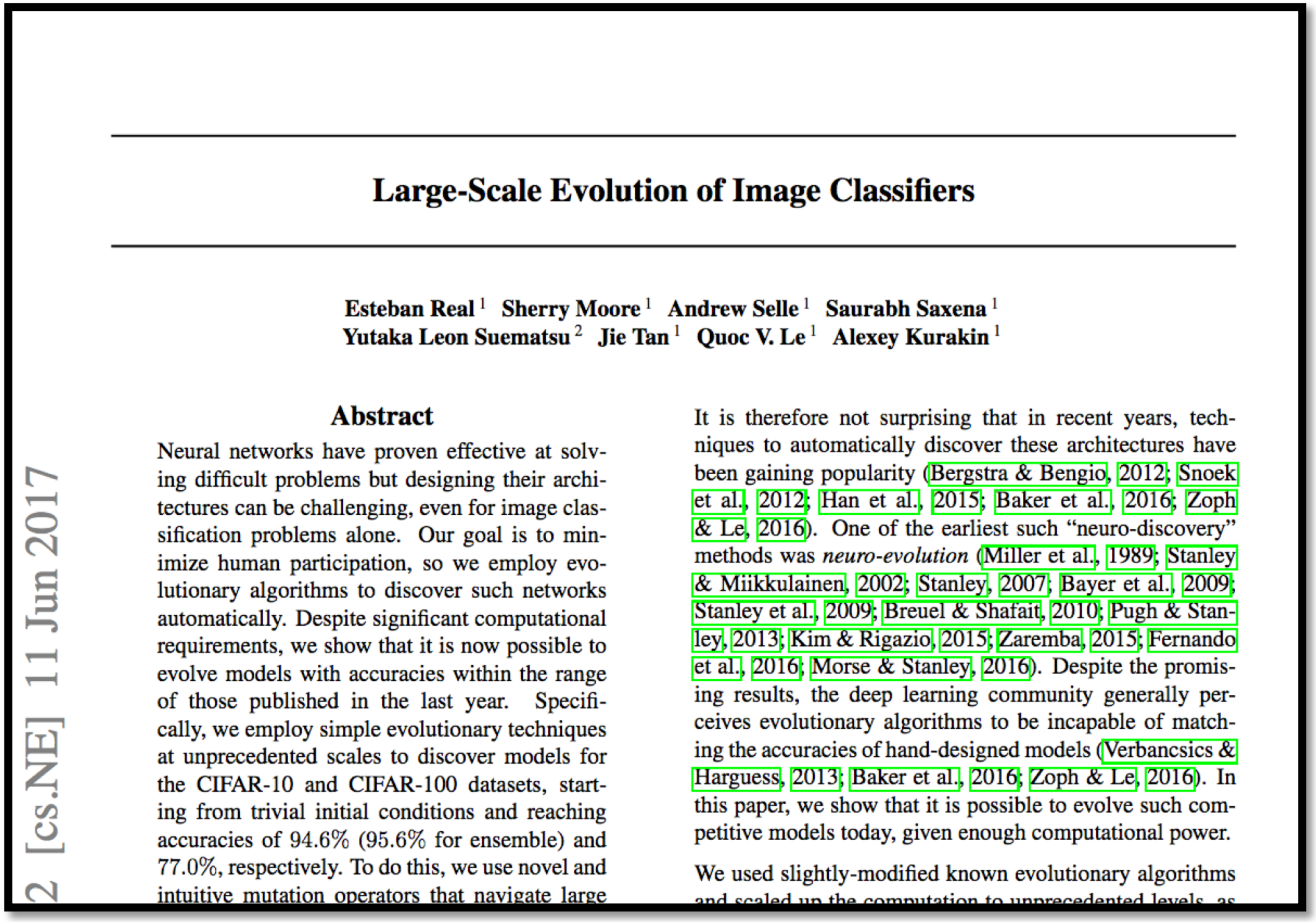
James Bergstra JAMES.BERGSTRA@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Département d'Informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC, H3C 3J7, Canada

Editor: Leon Bottou

But how to search the hyperparameter space *H* to find the optimal *h*?



But how to search the hyperparameter space *H* to find the optimal *h*?



Large-Scale Evolution of Image Classifiers

Esteban Real [1] Sherry Moore [1] Andrew Selle [1] Saurabh Saxena [1]
Yutaka Leon Suematsu [2] Jie Tan [1] Quoc V. Le [1] Alexey Kurakin [1]

Abstract

Neural networks have proven effective at solving difficult problems but designing their architectures can be challenging, even for image classification problems alone. Our goal is to minimize human participation, so we employ evolutionary algorithms to discover such networks automatically. Despite significant computational requirements, we show that it is now possible to evolve models with accuracies within the range of those published in the last year. Specifically, we employ simple evolutionary techniques at unprecedented scales to discover models for the CIFAR-10 and CIFAR-100 datasets, starting from trivial initial conditions and reaching accuracies of 94.6% (95.6% for ensemble) and 77.0%, respectively. To do this, we use novel and intuitive mutation operators that navigate large

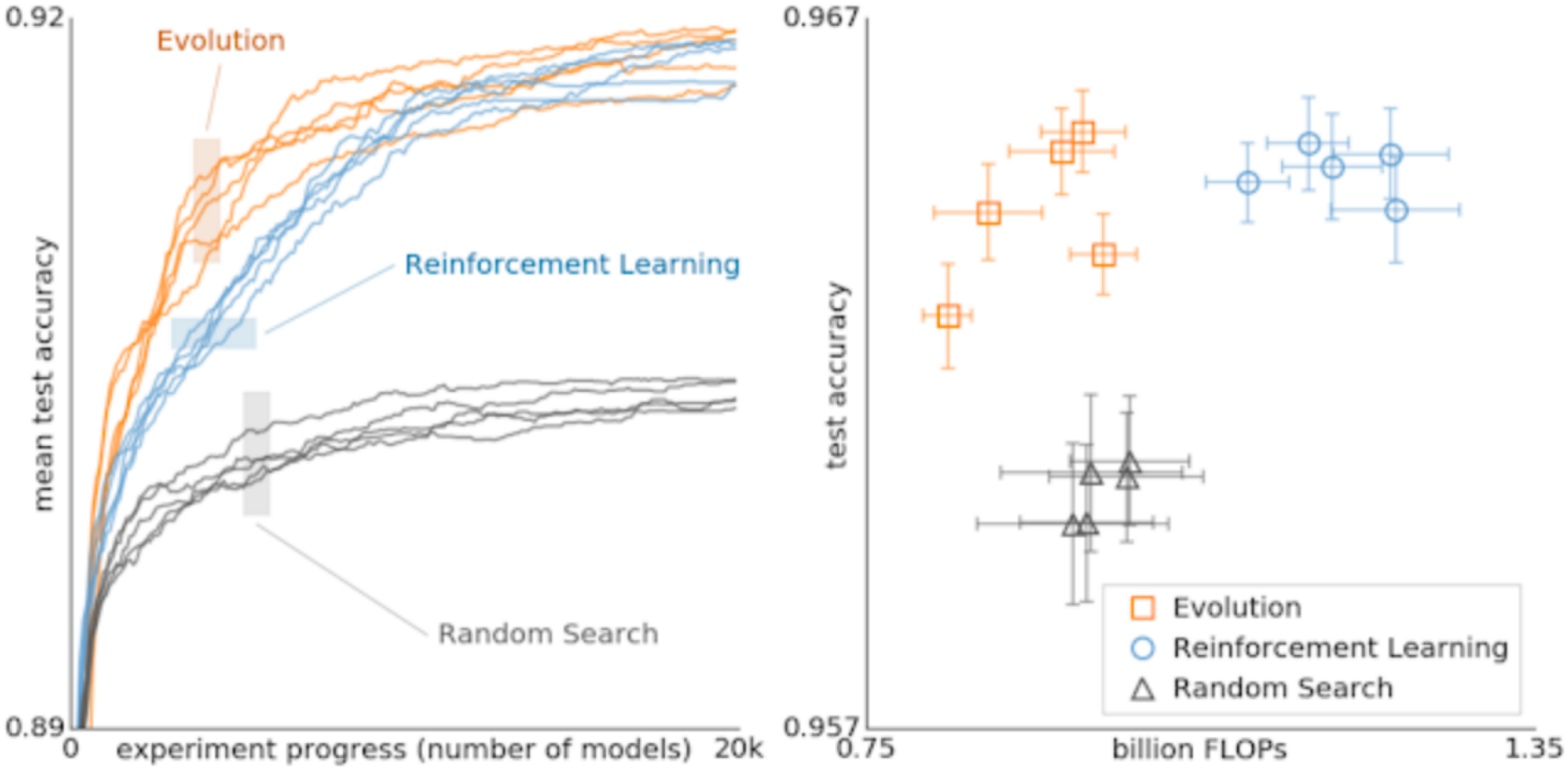
It is therefore not surprising that in recent years, techniques to automatically discover these architectures have been gaining popularity (Bergstra & Bengio, 2012; Snoek et al., 2012; Han et al., 2015; Baker et al., 2016; Zoph & Le, 2016). One of the earliest such "neuro-discovery" methods was *neuro-evolution* (Miller et al., 1989; Stanley & Miikkulainen, 2002; Stanley, 2007; Bayer et al., 2009; Stanley et al., 2009; Breuel & Shafait, 2010; Pugh & Stanley, 2013; Kim & Rigazio, 2015; Zaremba, 2015; Fernando et al., 2016; Morse & Stanley, 2016). Despite the promising results, the deep learning community generally perceives evolutionary algorithms to be incapable of matching the accuracies of hand-designed models (Verbancsics & Harguess, 2013; Baker et al., 2016; Zoph & Le, 2016). In this paper, we show that it is possible to evolve such competitive models today, given enough computational power.

We used slightly-modified known evolutionary algorithms and scaled up the computation to unprecedented levels, as

Not operational or very practical yet (or back in 2017 when it was published!)!

But how to search the hyperparameter space *H* to find the optimal *h?*

Google reported even more exciting follow-up results!



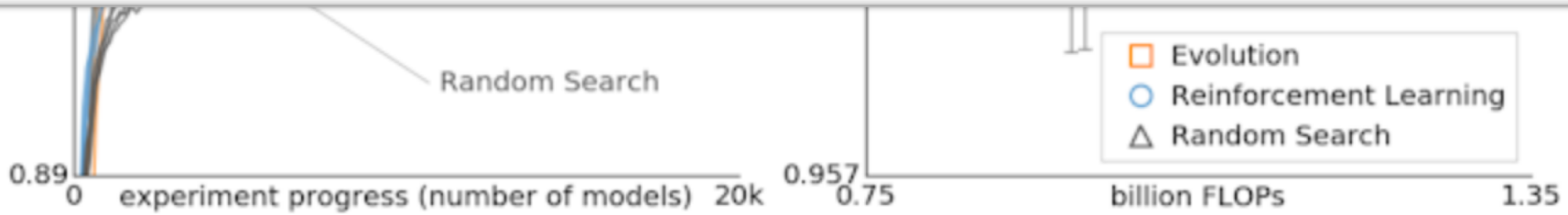
https://ai.googleblog.com/2018/03/using-evolutionary-automl-to-discover.html

But how to search the hyperparameter space *H* to find the optimal *h*?

Google reported even more exciting follow-up results!

Aging Evolution for Image Classifier Architecture Search

Esteban Real[*†] and **Alok Aggarwal**[†] and **Yanping Huang**[†] and **Quoc V. Le**
Google Brain, Mountain View, California, USA
[†]Equal contribution. *Correspondence: ereal@google.com



https://ai.googleblog.com/2018/03/using-evolutionary-automl-to-discover.html

- Homework and codes
  - Redesign MNIST classifier, and choose the number of hidden layers and the number of neurons (and any other hyperparameter that you choose, i.e. optimizer, cost function) in each layer using Grid Search, or Random Search, or PREFERABLY GA.