Intrinsic Dimension

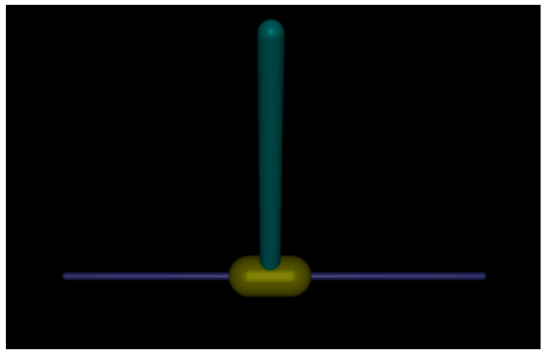
Learning goals

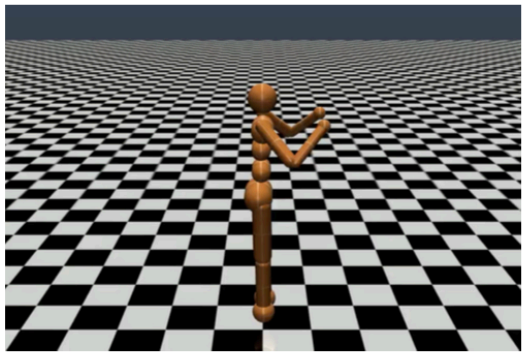
- Understand the notion of intrinsic dimension of objective functions
- Illustrate how intrinsic dimension quantify the complexity of a deep neural network architecture
- Measure intrinsic dimension using random projections
- Discuss the implications of measuring the intrinsic dimension and how this is related to model compression

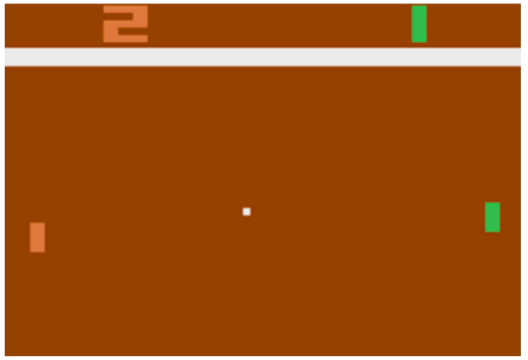
Which one is more difficult to solve?

- MNIST vs CIFAR-10?

Which one is more difficult to solve?







MEASURING THE INTRINSIC DIMENSION OF OBJECTIVE LANDSCAPES

Chunyuan Li *
Duke University
cl319@duke.edu

Heerad Farkhoor, Rosanne Liu, and Jason Yosinski
Uber AI Labs
{heerad,rosanne,yosinski}@uber.com

Gradient descent

repeat until convergence $\{$

$$\theta \leftarrow \theta - \gamma \nabla_\theta J(\theta)$$

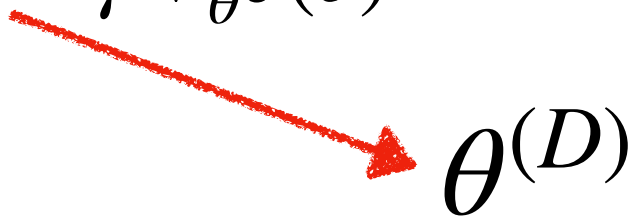
$\}$

Evaluating the gradient of a loss with respect to parameters and taking steps directly in the space of θ(D)

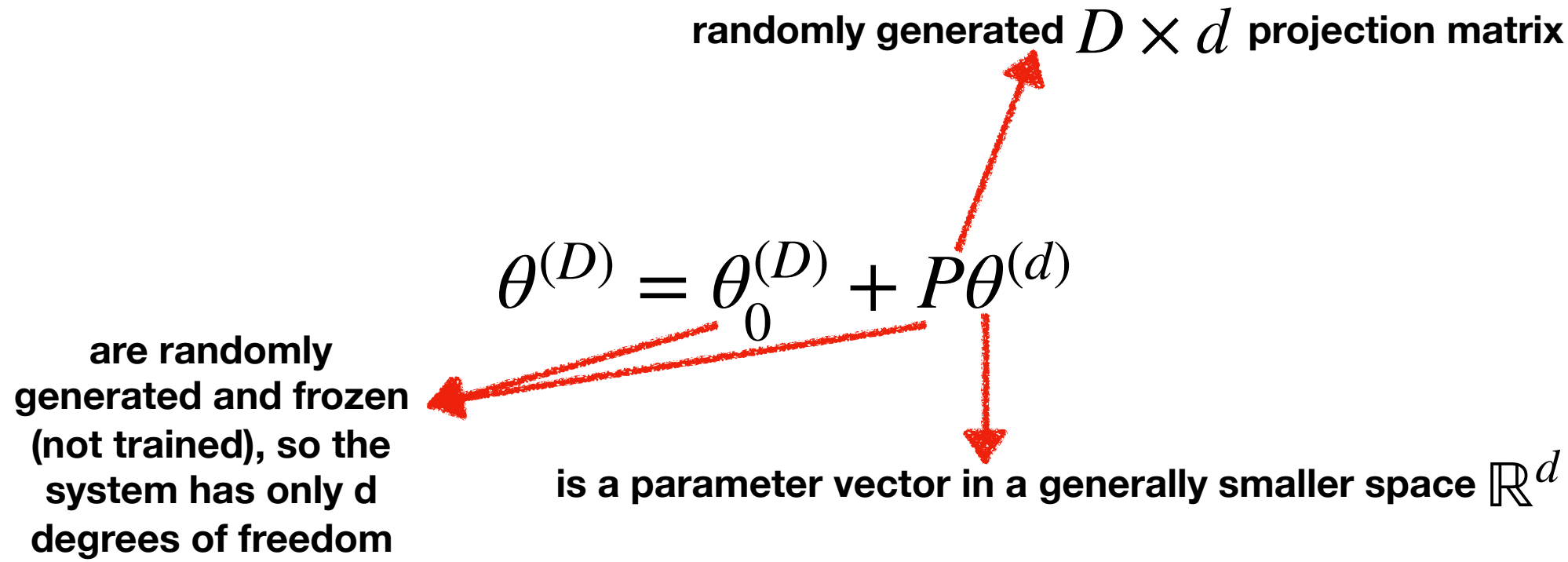
repeat until convergence {

$$\theta \leftarrow \theta - \gamma \nabla_\theta J(\theta)$$

}



Lets instead define θ(D) in the following way



Start of training

- We initialize θ(d) to a vector of all zeros,
- So initially $\theta^{(D)} = \theta_0^{(D)}$

It allows the network to start off well-conditioned

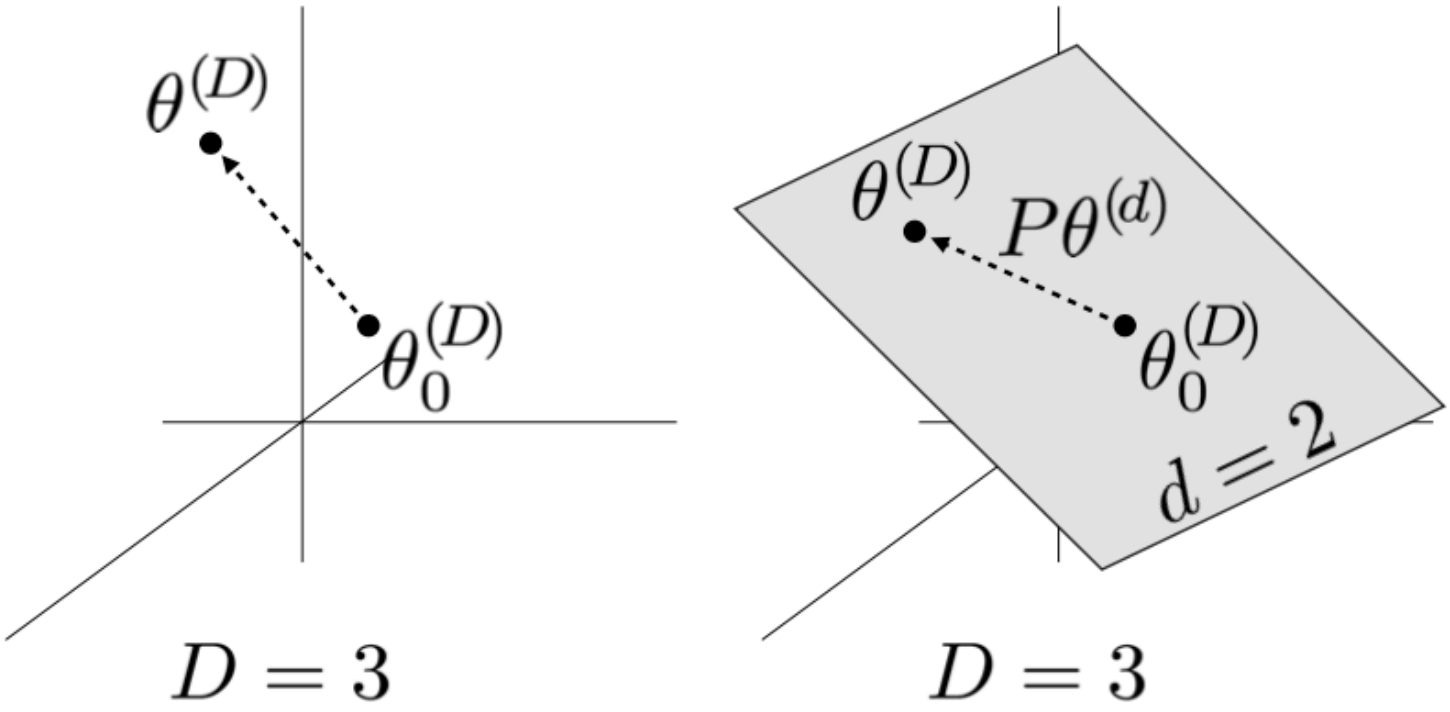
- This convention serves an important purpose for neural network training: it allows the network to benefit from beginning in a region of parameter space designed by any number of good initialization schemes, see (Glorot & Bengio, 2010; He et al., 2015) to be well-conditioned,

- Such that gradient descent via commonly used optimizers will tend to work well.

Training procedure

- Training proceeds by computing gradients with respect to θ(d) and taking steps in that space.
- Columns of P are normalized to unit length, so steps of unit length in θ(d) chart out unit length motions of θ(D).
- Columns of P may also be orthogonalized if desired.
- By this construction P forms an approximately orthonormal basis for a randomly oriented d dimensional subspace of RD, with the origin of the new coordinate system at θ_0(D).

Illustration of parameter vectors



Let’s review few properties of this approach

How about d = D and P is a large identity matrix?

When d = D and P is a large identity matrix

- If d = D and P is a large identity matrix, we recover exactly the direct optimization problem.

How about If d = D but P is instead a random orthonormal basis for all of RD (just a random rotation matrix)?

How about If d = D but P is instead a random orthonormal basis

- We recover a rotated version of the direct problem.

Do we need to change the way how we optimize DNN architectures?

- Note that for some “rotation-invariant” optimizers, such as SGD and SGD with momentum, rotating the basis will not change the steps taken nor the solution found,

- But for optimizers with axis-aligned assumptions, such as RMSProp and Adam, the path taken through θ(D) space by an optimizer will depend on the rotation chosen.

Choosing a heuristic to compare performance

- Choose a heuristic for classifying points on the objective landscape as solutions vs. non-solutions.
- The heuristic we choose is to threshold network performance at some level relative to a baseline model, where generally we take as baseline the best directly trained model.
- In supervised classification settings, validation accuracy is used as the measure of performance, and in reinforcement learning scenarios, the total reward is used.
- Accuracy and reward are preferred to loss to ensure results are grounded to real-world performance and to allow comparison across models with differing scales.

Selecting baseline

- We define d_{int100} as the intrinsic dimension of the “100%” solution: solutions whose performance is statistically indistinguishable from baseline solutions.

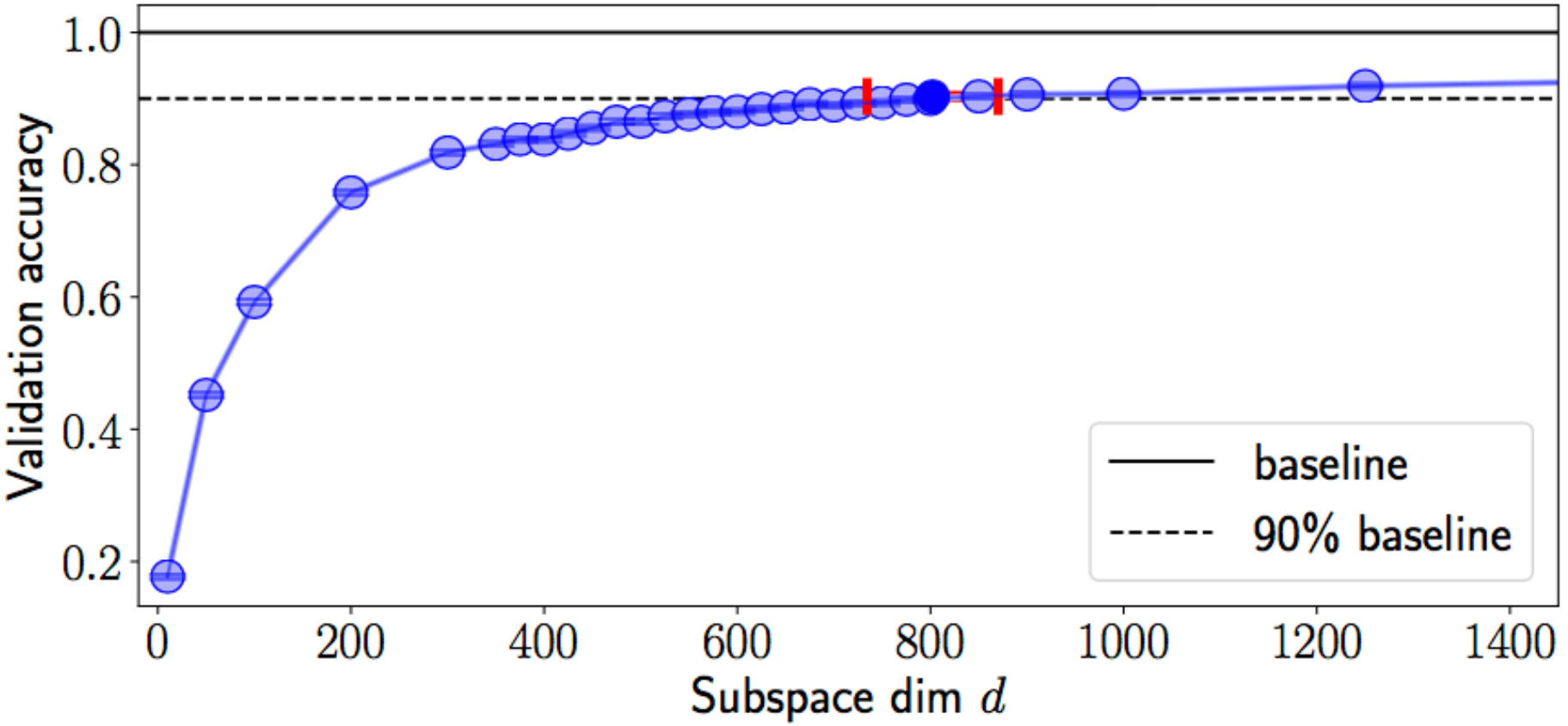
- However, when attempting to measure d_{int100}, we observed it to vary widely: it can be very high — nearly as high as D — when the task requires matching a very well-tuned baseline model,

- but can drop significantly when the regularization effect of restricting parameters to a subspace boosts performance by tiny amounts.

Selecting baseline

- Thus, we found it more practical and useful to define and measure d_{int90} as the intrinsic dimension of the “90%” solution: solutions with performance at least 90% of the baseline.

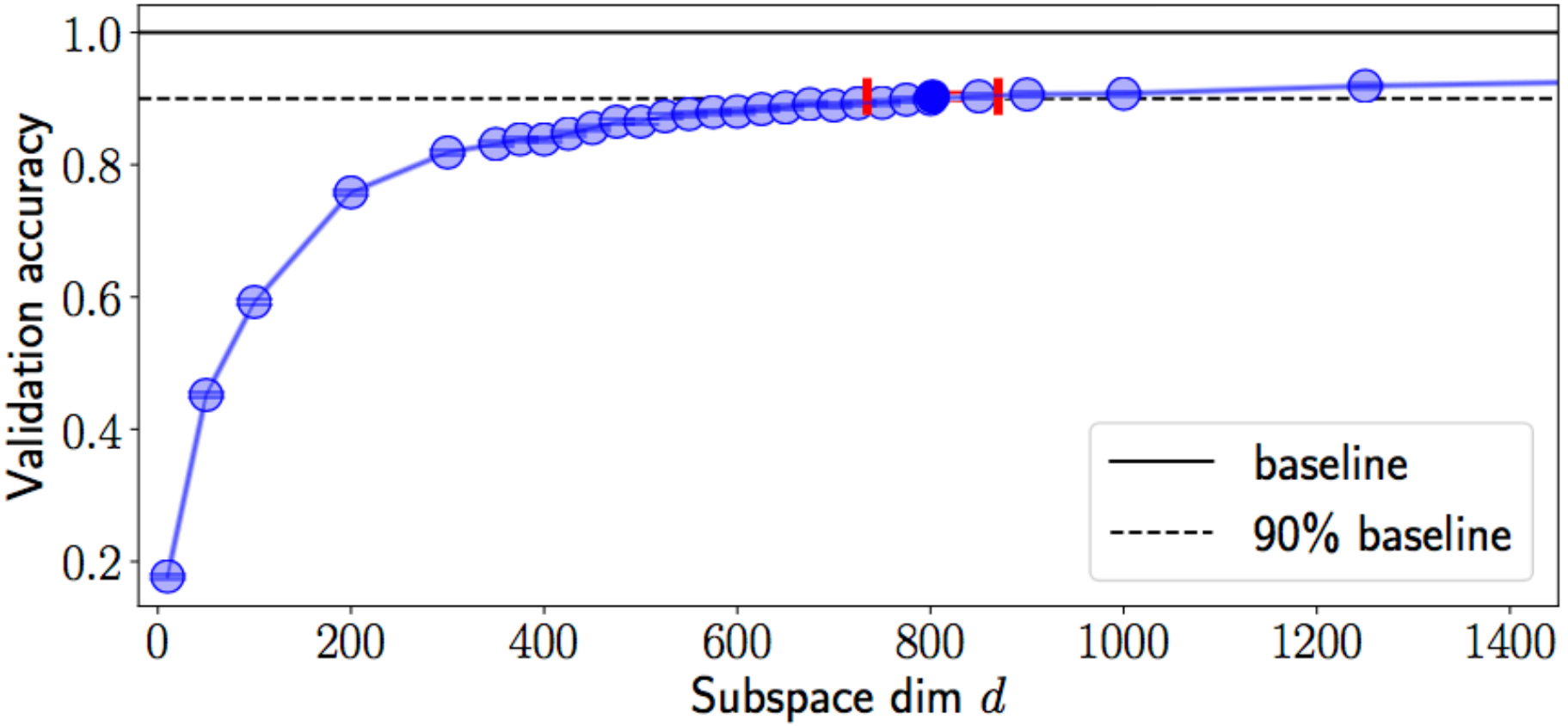
Performance (validation accuracy) vs. subspace dimension d



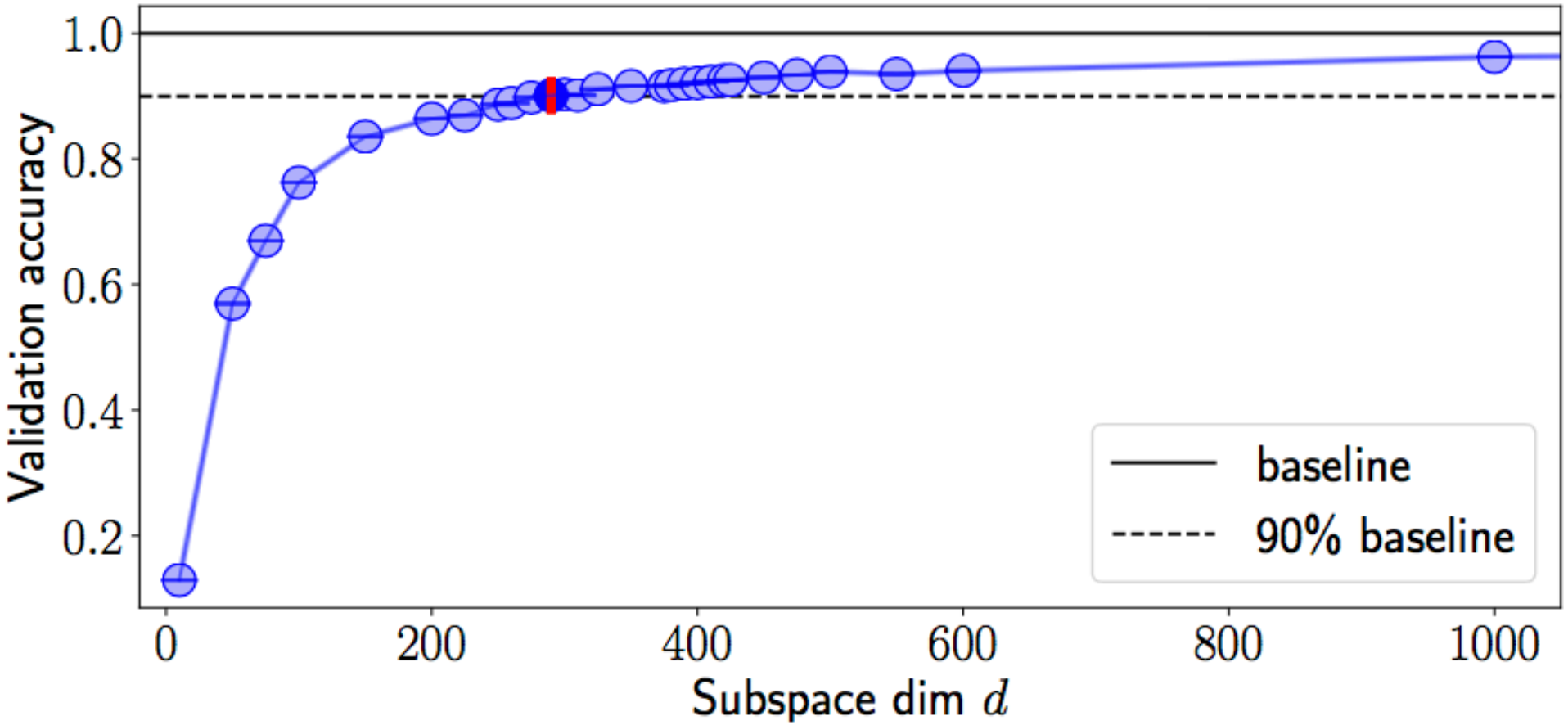
Results

- We begin by analyzing a fully connected (FC) classifier trained on MNIST.
- We choose a network with layer sizes 784–200–200–10,;
- This results in a total number of parameters D = 199, 210.

Fully-connected (FC) network (D = 199,210)



a convolutional network, LeNet (D = 44,426)



Some networks are very compressible

- A salient initial conclusion is that 750 is quite low.
- At that subspace dimension, only 750 degrees of freedom (0.4%) are being used and 198,460 (99.6%) unused to obtain 90% of the performance of the direct baseline model.

So what? What can we do with that?

Practical application: Network compression

- New way of creating and training compressed networks,
- Particularly networks for applications in which the absolute best performance is not critical.

More details? What we need to store?

We only need to store

- The random seed to generate the frozen $\theta_0^{(D)}$
- The random seed to generate P
- The 750 floating point numbers in $\theta_*^{(D)}$

Compression rate

- It leads to compression(assuming 32-bit floats) by a factor of 260× from 793kB to only 3.2kB,
- Or 0.4% of the full parameter size.

So what? In what scenarios this might be useful?

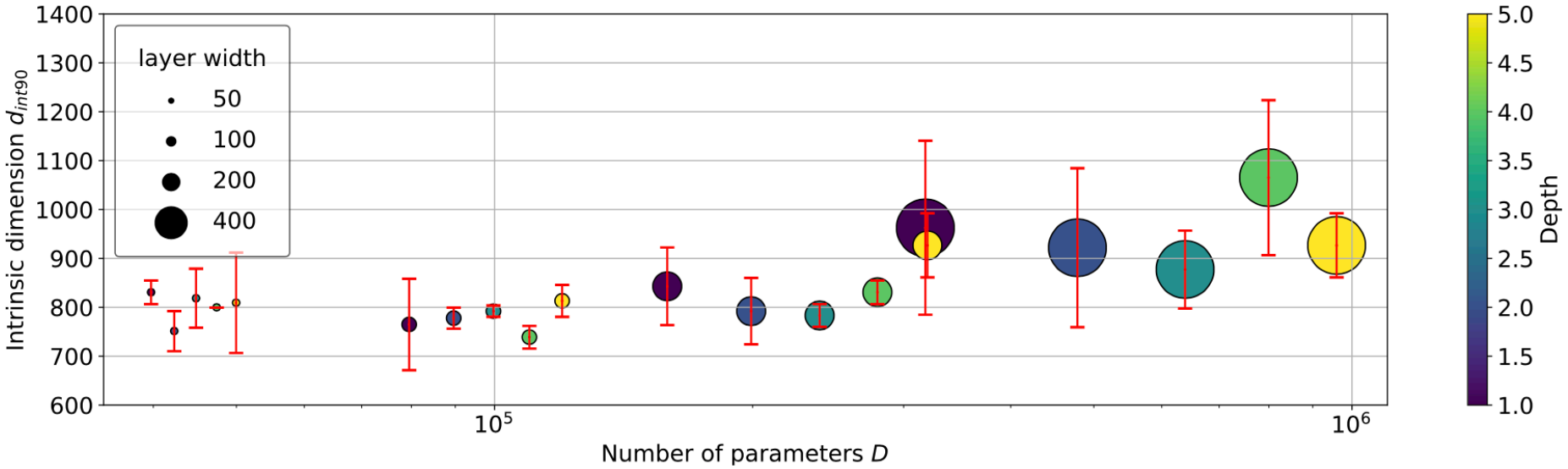
Application of the model compression

- Such compression could be very useful for scenarios where storage or bandwidth are limited,
- E.g. including neural networks in downloaded mobile apps or on web pages.

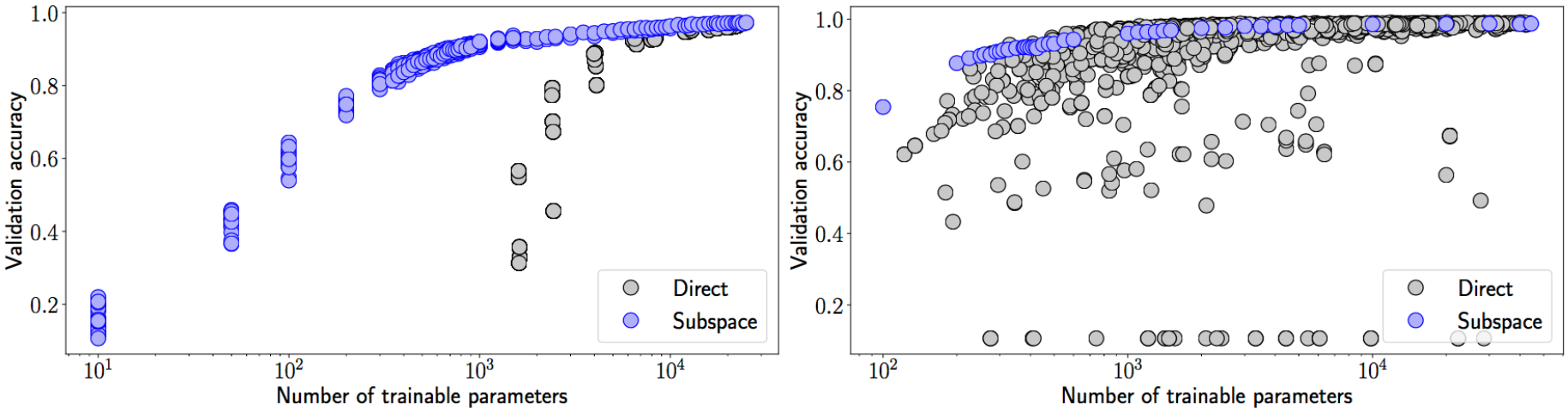
Contrasts to other existing model compression approaches

- Post-hoc vs once during training
- Layerwise compression models vs entire parameter space, which could work better or worse
- Simple vs difficult approaches that put hierarchical priors on the weights
- reduces the number of degrees of freedom, not the number of bits required to store each degree of freedom, complimentary to weight quantization
- Training vs inference

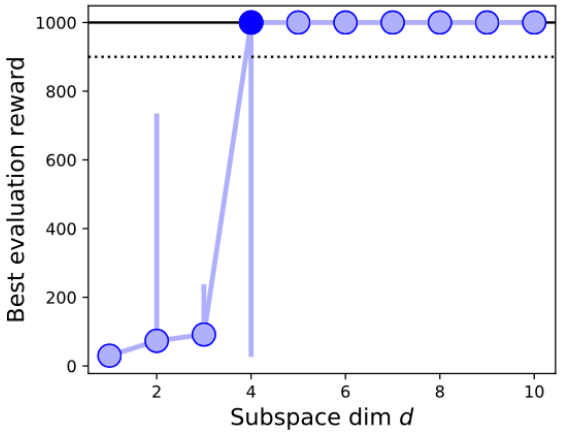
Robustness of intrinsic dimension

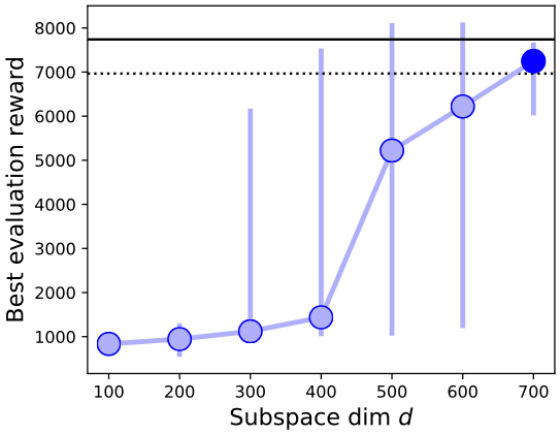


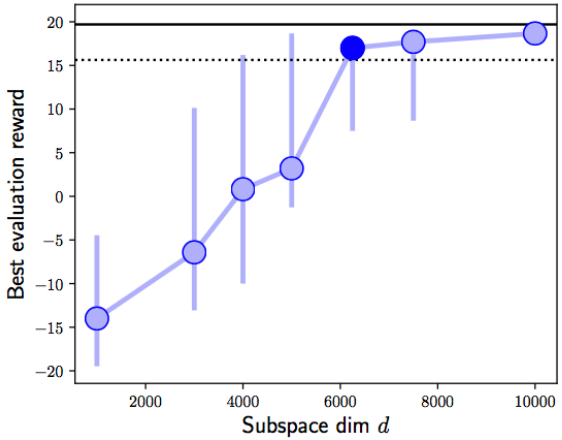
Performance vs. number of trainable parameters

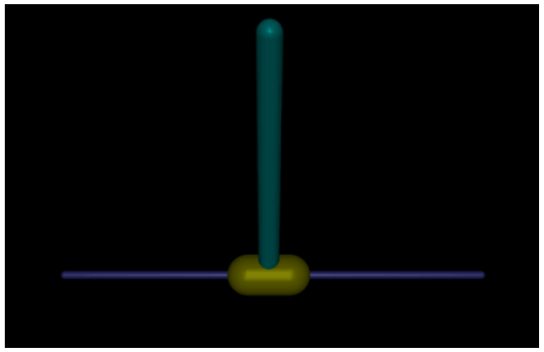


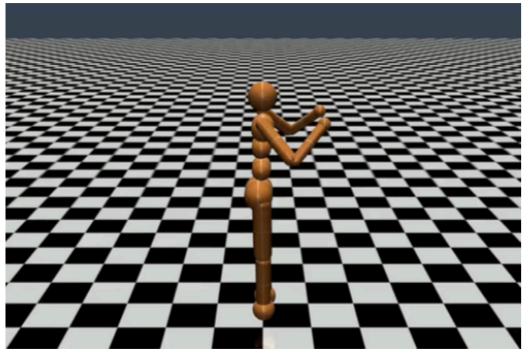
Results using the policy-based ES algorithm to train agents on (left column) InvertedPendulum–v1, (middle column) Humanoid–v1, and (right column) Pong–v0

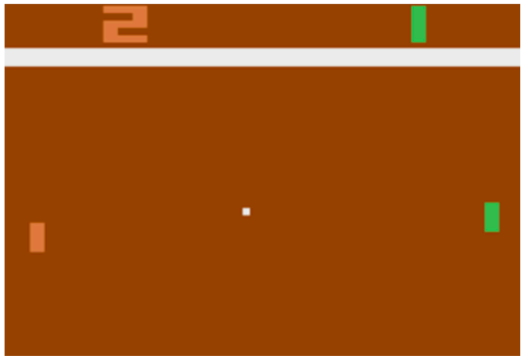












Summary

- We understood the intrinsic dimension of objective landscapes and shown a simple method — random subspace training — of approximating it for neural network modeling problems.

- We are now able to use this approach to compare problem difficulty within and across domains.

- We saw examples that the intrinsic dimension is much lower than the direct parameter dimension, and hence enable network compression, and in other cases the intrinsic dimension is similar to that of the best tuned models.