Neural ODE Problem Setup

- We have a func $F(x,t)$
  we want to train a dynamic NN to compute $F(x,t_1)$ given $F(x,t_0)$ and samples at $F(x,t_1)$
  (t_1: whatever time step)
- an initial value problem
  $\frac{dF}{dt} = f = F(x,t,p)$ derivative of F w.r.t t
  can be solved by ODE solver
  But we don't know f (f: analytical) so we use a NN to learn it

Deep Learning Theory and Applications

Representation Learning in Neural Networks

CPSC/AMTH 663

Yale

Summary How to design NN

① initial: deep and wide } lottery ticket hypoth

② prune (neuron or edge) }

③ narrower subsequent hidden layers : J L Lemma

What is happening inside a neural network?

- The output of a neural network is a function (continuous)
- We learned last time that it can be ANY function
  - Should be able to approximate it
- Now what about the other layers?
- How are they facilitating the learning?

Dimensionality Reduction

- Neural networks typically learn a dimensionality reduced embedding in their internal layers
- These embeddings form GOOD projections of the data
  - Projections that maintain distances between datapoints
  - Projections that can show groupings, clusters etc $\|x_1 - x_2\| \approx \|f(x_1) - f(x_2)\|$
- This can be shown mathematically

keep important info and remove useless info

Johnson Lindenstrauss Lemma 定理

sample size

Theorem 2.1. *For any $0 < \epsilon < 1$ and any integer n, let k be a positive integer such that*

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n. \tag{2.1}$$

Then for any set V of n points in $\mathbf{R}^d$, there is a map $f : \mathbf{R}^d \to \mathbf{R}^k$ such that for all u, $v \in V$,

$$(1 - \epsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon)\|u - v\|^2.$$

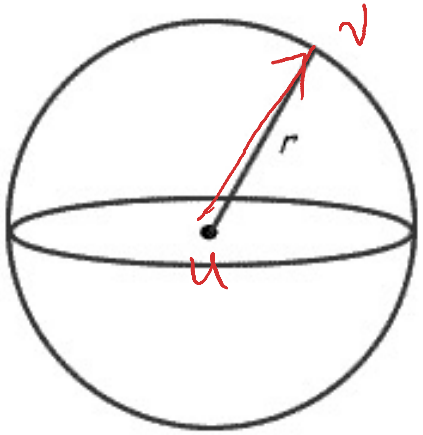
squared distance　　close to original distance

Furthermore, this map can be found in randomized polynomial time. $T(O^n)$

d : input size

k : embedding size

Length of projected unit vector



- What is the expected length of a unit vector in d dimensions when it is projected down to K?
- If several such vectors are projected down would their relative magnitudes be preserved?

Proof Concept

- It shows that the squared length of a random vector is sharply concentrated around its mean when the vector is projected onto a random k-dimensional subspace. Specifically, with probability O(1/n2), its (scaled) length is not distorted by more than (1+ ϵ)
- Let $X_1,\ldots, X_d$ be d independent Gaussian N(0, 1) random variables, and let Y = $\frac{1}{||x||}$ ($X_1,\ldots$, Xd). Y is a point chosen uniformly at random from the surface of the d-dimensional sphere S^d. Let the vector Z $\in R^k$ be the projection of Y onto its first k coordinates, and let $L=||Z||^2$.
- The expected squared length of Z is $\mu = E[L]=k/d$
  - And the probability is tightly concentrated around the mean

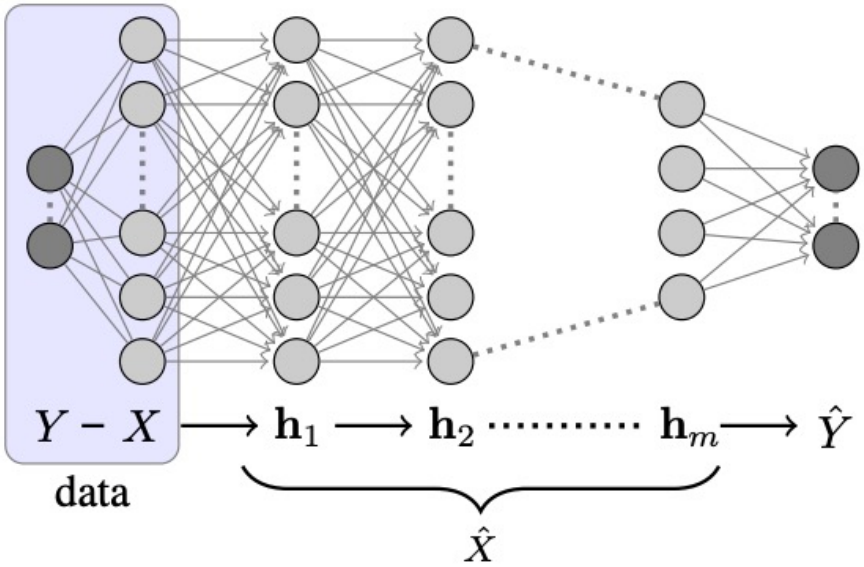
Johnson-Lindenstrauss lemma

- How does this connect to neural networks?
- This means that random initialization helps find good results, local minima are pretty good
- High probability of cluster patterns, manifolds, existing bc preserve distance will preserve most info
- SGD will only improve upon it.

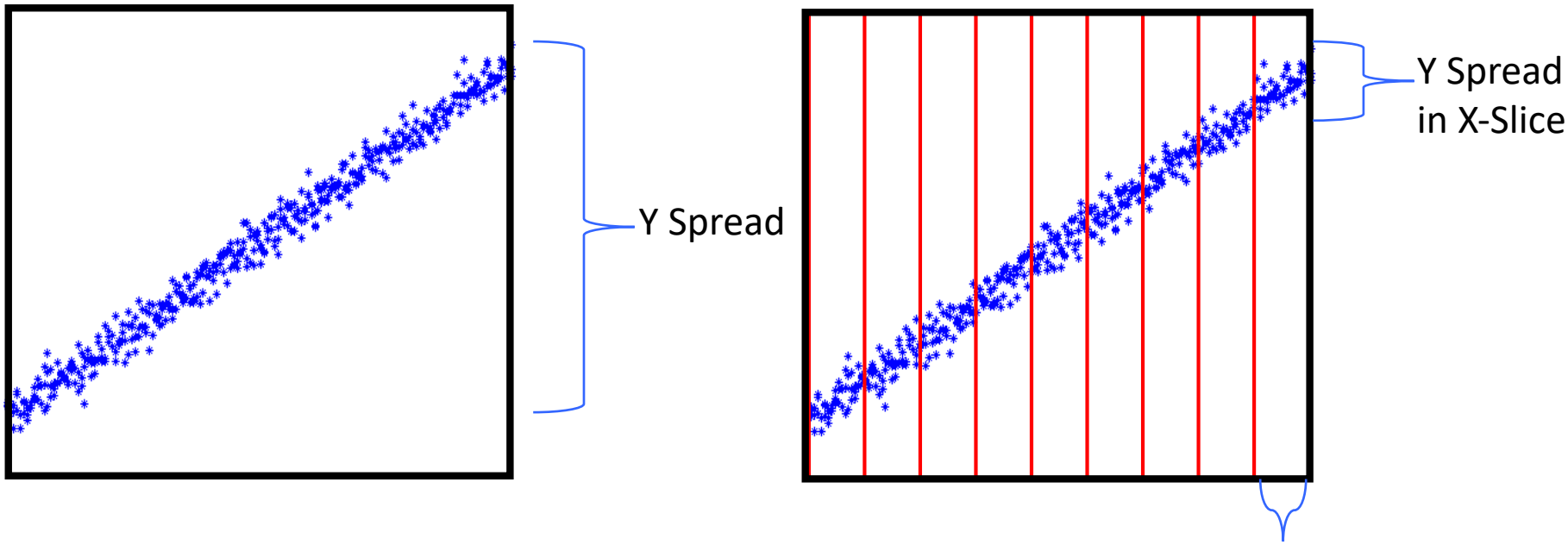
Deep Projections

- We have projections, but why make them deep?

- What do subsequent layers do?



Entropy as a measure of uncertainty



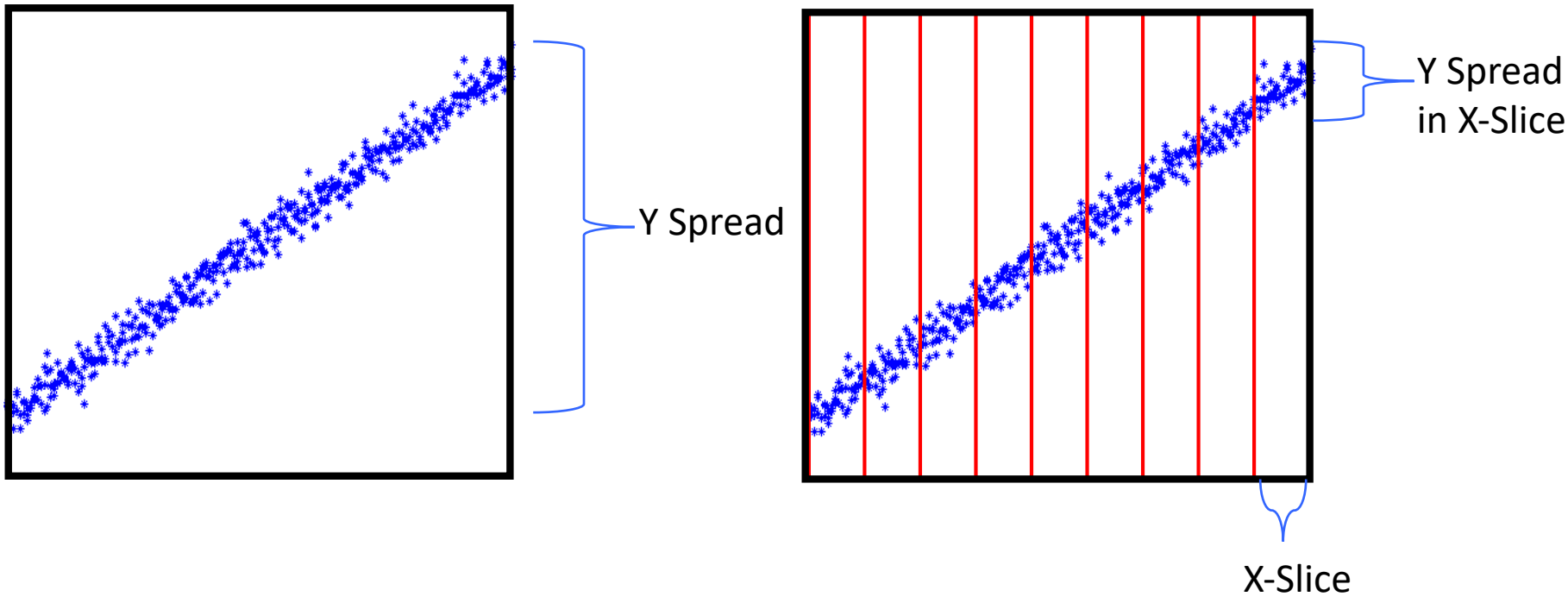
Measure of Uncertainty in a random variable.

$$-\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i),$$

Shannon Entropy

Units of bits tells us how many bits are needed to represent the outcome

Mutual Information

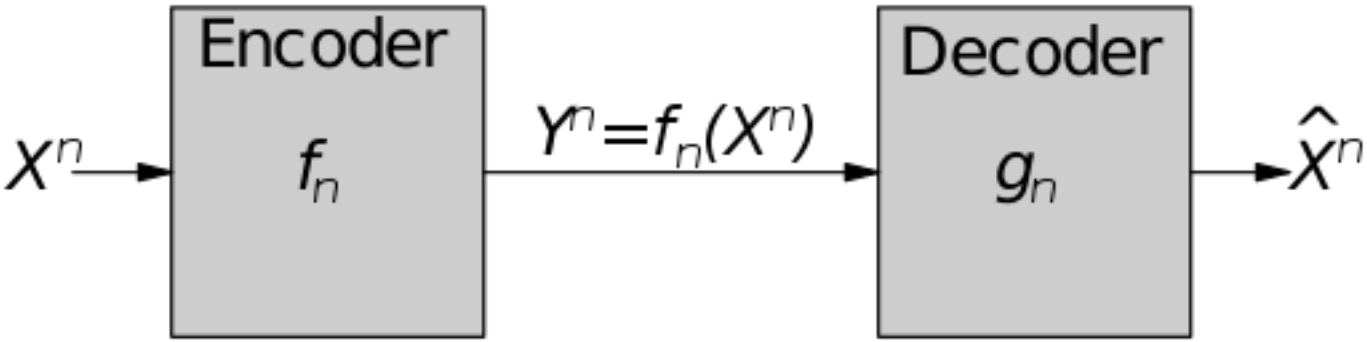


How much info X on Y

Mutual Information: $I(X,Y) = H(Y) - H(Y|X)$

↑ uncontional entropy ↑ conditional entropy

Rate Distortion Theory



$$\inf_{Q_{Y|X}(y|x)} I_Q(Y;X) \text{ subject to } D_Q \leq D^*.$$

Information Bottleneck Principle

- Introduced by Tishby et al. uses information theory to understand what about the input is relevant to the output
- Given a joint distribution p(X,Y), the information *relevant* to Y within X is given by I(X,Y) the *mutual information* between X and Y
- Therefore, a good compression of X should capture the mutual information and discard the remaining information

Information Bottleneck Loss

- Suppose the relevant information about Y is captured by minimum sufficient statistics $\hat{X}$ (compressed repr of X)
- We can define a an loss information bottleneck loss

$$\mathcal{L}\left[p\left(\hat{x}|x\right)\right] = I\left(X;\hat{X}\right) - \beta I\left(\hat{X};Y\right)$$

high loss: { large $I(X;\hat{X})$ — $\hat{x} \approx x$ — compression rate
small $I(\hat{X};Y)$ — $\hat{x}$ has no info on Y — info preserv

- This defines a tradeoff finding a compressed representation of X, and retaining information about Y based on the parameter Beta
- Compression rate vs information preservation

$$R^{(I)}(I_Y) = \min_{p(\hat{x}|x):I(\hat{X};Y)\geq I_Y} I(X;\hat{X})$$

↑ threshold

minimize $I(X;\hat{x})$ while mataining $I(\hat{x};Y) \geq$ some threshold I_Y

IB Loss

- The information bottleneck loss can be written as (only differ by constant)

$$\tilde{\mathcal{L}}\left[p\left(\hat{x}|x\right)\right] = I\left(X;\hat{X}\right) + \beta I\left(X;Y|\hat{X}\right)$$

condition on $\hat{x}$

- With this you can define a distortion

$$D_{IB} = E\left[d_{IB}\left(X,\hat{X}\right)\right] = I(X;Y|\hat{X})$$

- Distortion is the expected Kullback-Liebler divergence

$$d_{IB}\left(x,\hat{x}\right) = D\left[p\left(y|x\right)\|p\left(y|\hat{x}\right)\right]$$

DNNs as Markov Chains

- DNNs use multiple internal layers to compress representations
- These can be thought of Markov Chains and reasoning about them can be done via the Data Processing Inequality

$$I(Y;X) \geq I(Y;\mathbf{h}_j) \geq I(Y;\mathbf{h}_i) \geq I\left(Y;\hat{Y}\right)$$

input — hidden layer — $j < i$ — predicted output

- Achieving equality only happens when maximizing information to the output layer but also minimizing as much as possible information between subsequent layers

$$I(\mathbf{h}_{i-1};\mathbf{h}_i)$$

Relevant quantities

- It has been shown that can bound classification error in tasks with multiple classes $I(Y;\mathbf{h}_i)$

depends on info

- This is the mutual information between the representation in layer i and the class labels in Y

- This is the minimal description length

bits

$I(\mathbf{h}_{i-1};\mathbf{h}_i)$ Info between hidden layers

- Each layer can be compared to the theoretical optimal layer for the loss $I(\mathbf{h}_{i-1};\mathbf{h}_i) + \beta I(Y;\mathbf{h}_{i-1}|\mathbf{h}_i)$

Optimal Solutions for IB Loss

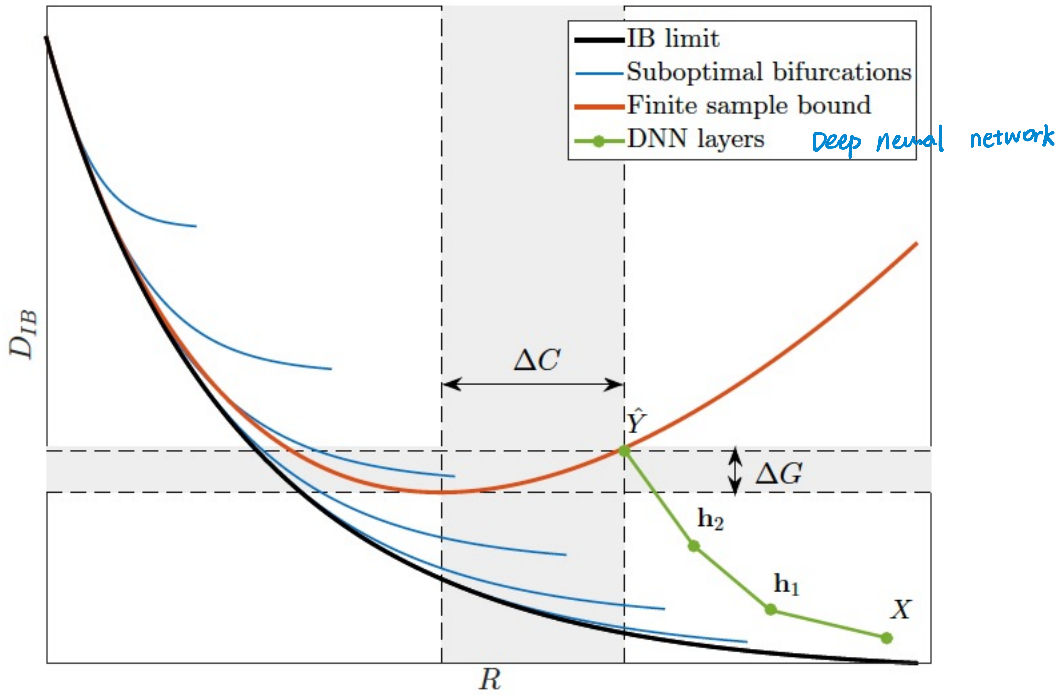
- Can be found by iterating these equations

$$
\begin{aligned}
p(\hat{x}|x) &= \frac{p(\hat{x})}{Z(x;\beta)} \exp\left(-\beta D\left[p(y|x)\,\|p(y|\hat{x})\right]\right) \\
p(y|\hat{x}) &= \sum_x p(y|x)\, p(x|\hat{x}) \\
p(\hat{x}) &= \sum_x p(x)\, p(\hat{x}|x)
\end{aligned}
$$

x

- This gives rise to an optimal rate distortion curve (not one that is reached by DNNs)

Rate Distortion Curve



β is the negative slope of this curve

Bifurcations happened where there are topological changes in the representation

Green are DNN layers are generally higher than the optimal

Multiple DNN Layers

- Each layer can only increase the distortion (based on data processing inequality)----can't add lost information
- However layers can compress information better in order to meet the finite sample bound curve
- This is the bound you can get using only a finite sample

$$I\left(X;\hat{X}\right) \leq \hat{I}\left(X;\hat{X}\right) + O\left(\frac{K}{\sqrt{n}}\right) \qquad K = |\hat{\mathcal{X}}|$$

IB-based Reason for Multiple Layers

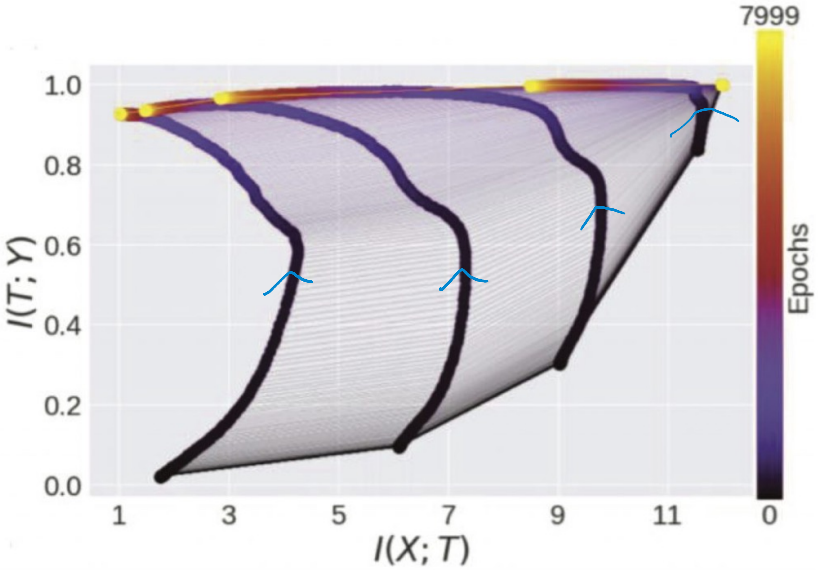
- Each layer in a neural network effectively only linearly seperates the information in the pervious layer
- Recall: One layer neural networks are equivalent to SVMs
- This is because layers are non-linear functions of dot products
- Linear separability is only possible when input units are conditionally independent given output classification
- This this is not necessarily true of the data distribution, multiple hidden layers are required!

Phase Transitions

- Interesting theory broached by Tisby et al. is that one could find the correct architecture
- Number of layers and number of neurons in each layer by looking to see phase transitions of the rate distortion curve using bifurcation analysis
- Namely finding critical values of β where one can move to simpler representations

During Training

- It has been observed that the mutual information between the input layer and the hidden layers initially increases as the neural network learns to encode the input
- Then falls rapidly as it learns to compress this information



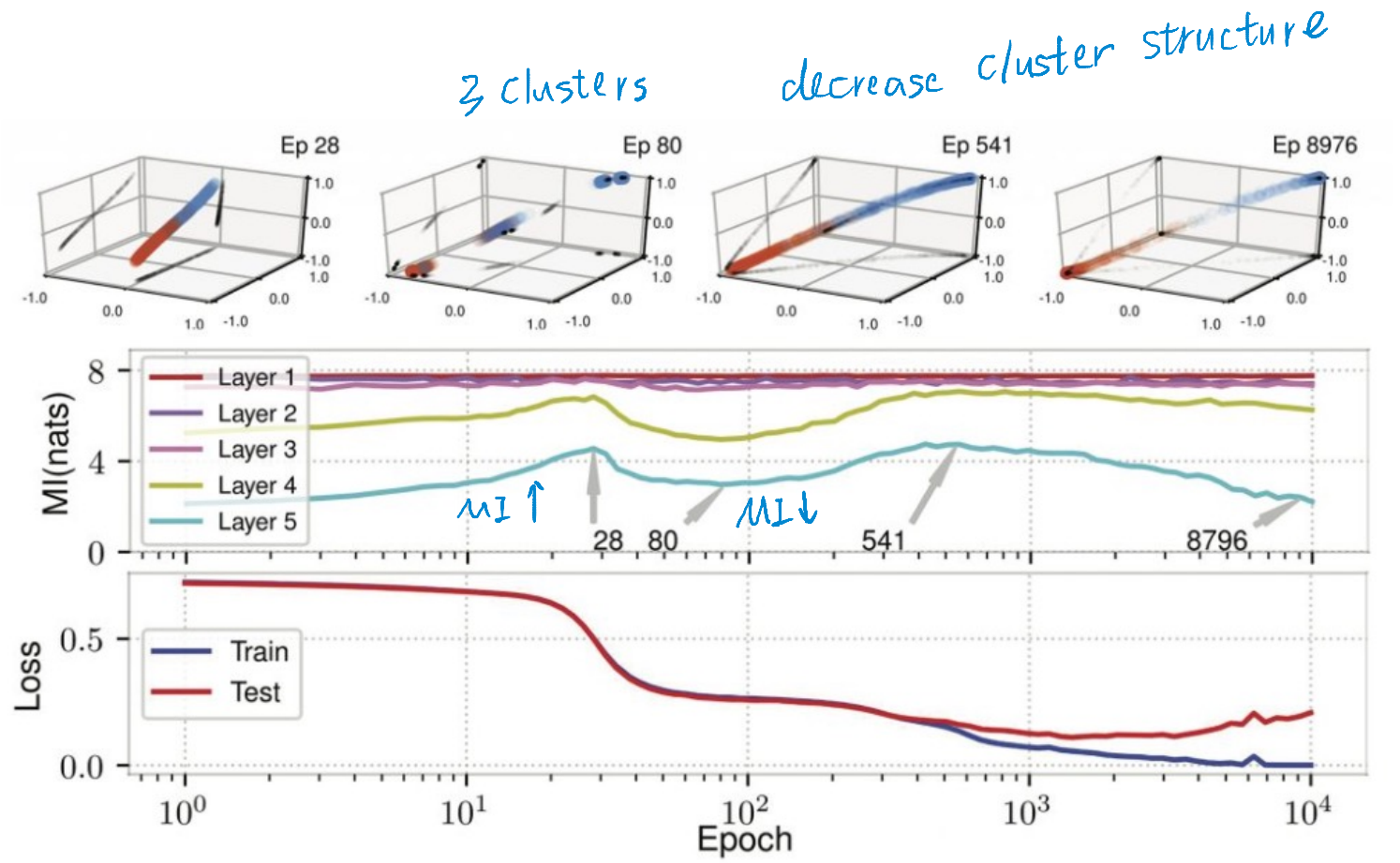
T: layer T

During training

I(X;T) 先↑后↓

I(T;Y) 一直↑

Evolution of representation



Rise and fall of MI corresponds to how clustered the representation will be

How do representations evolve?

- What happens during training

- We have shown that random initializations start well where do they go after training?

M-PHATE

Visualizing Internal Representations
(Gigante,S. et al. NeurIPS 2019)

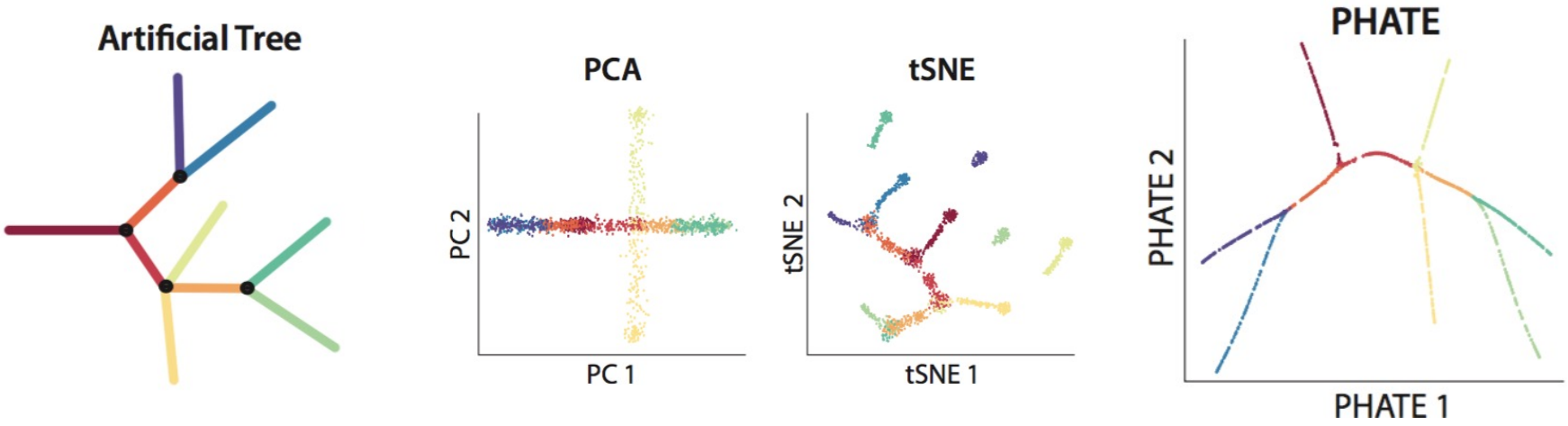
Scott Gigante

Gal Mishne

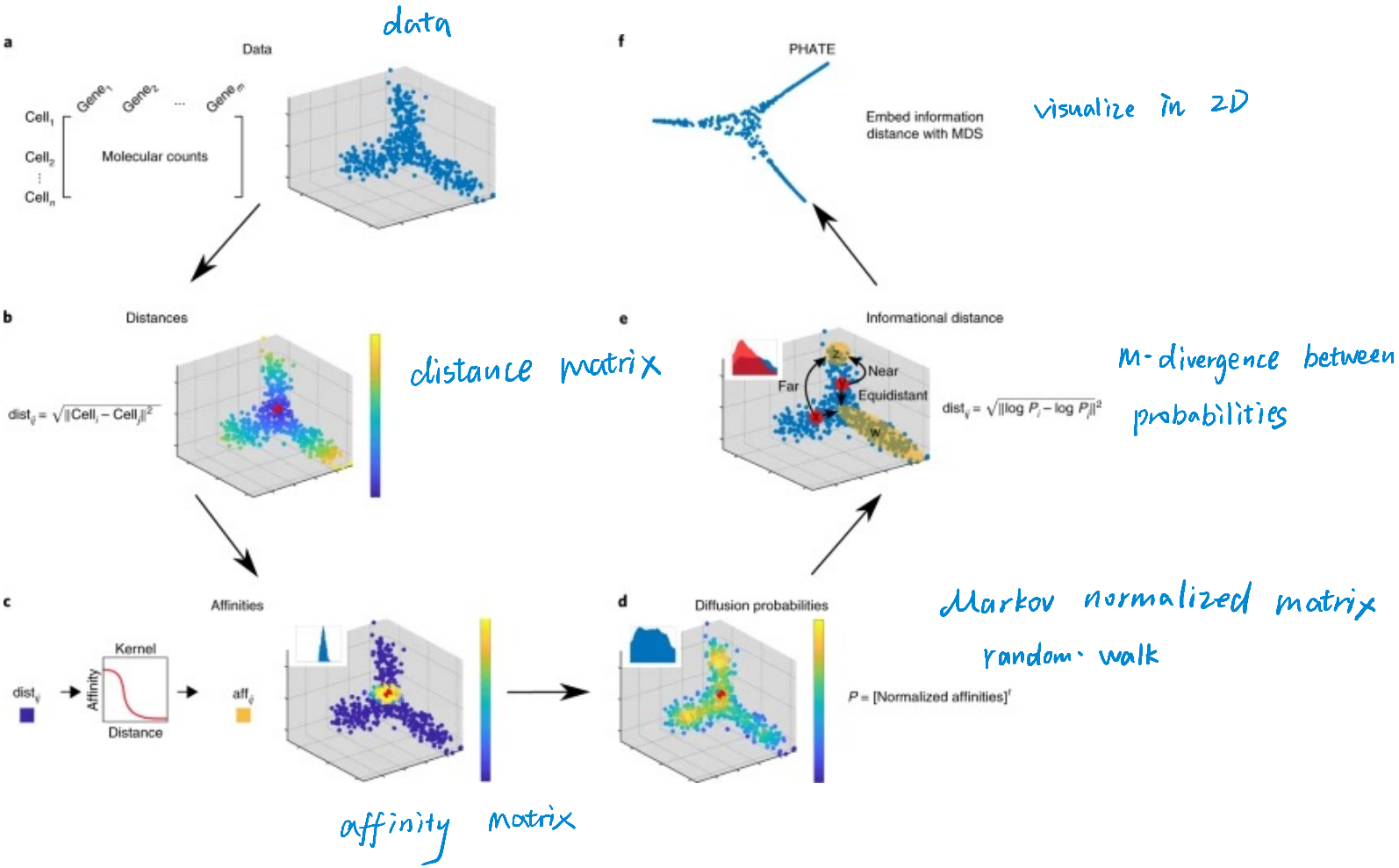
Main idea: Multi-slice kernel allows for PHATE to visualize dynamic behaviorin neural networks

Scott Gigante
Adam Charles
Gal Mishne

Visualization methods fail to capture global structure in high dimensional data



keep distance in manifold

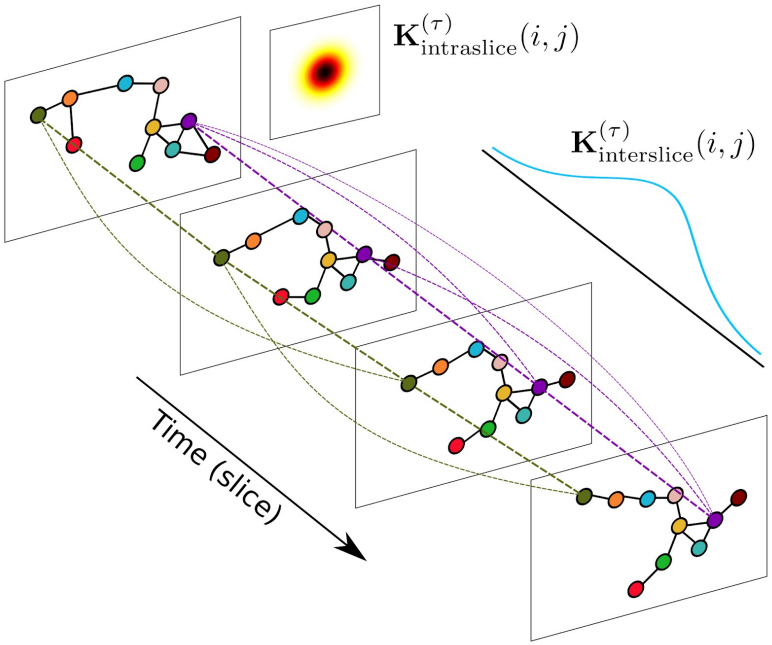

data
a
Data
Gene1
Gene2
...
Genem
Cell1
Cell2
Celln
Molecular counts
f
PHATE
Embed information distance with MDS
visualize in 2D
b
Distances
distij = √‖Celli − Cellj‖2
distance matrix
e
Informational distance
Near
Equidistant
Far
distij = √‖log Pi − log Pj‖2
M-divergence between probabilities
c
Affinities
Kernel
Affinity
Distance
distij
affij
affinity matrix
d
Diffusion probabilities
P = [Normalized affinities]r
Markov normalized matrix
random walk

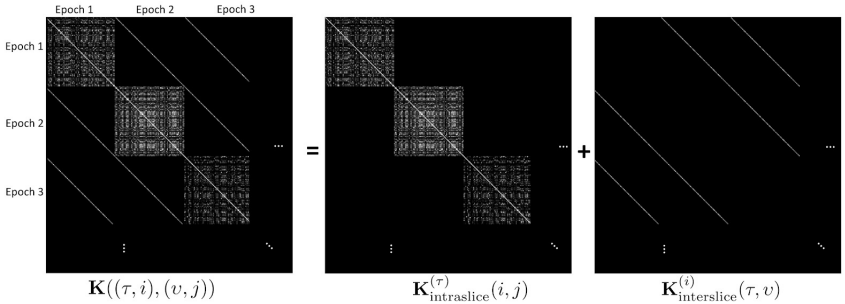
M-PHATE Neural Network Learning

Multi-slice PHATE

let PHATE sweep thru time slice

induce affinity between neural itself at different time points





Learning vs Memorization in Nnets

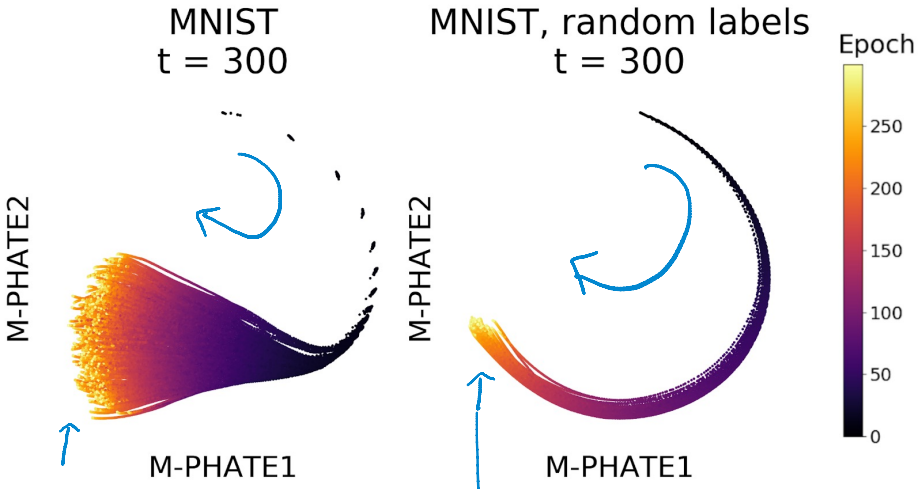
How to make NN memorize rather than learn sth ?

A: give scramble labels rather than true labels (meaningful)

无序的

visualize internal layers at epoch

Mnist handwritten digits



more diversity to encode more features

increase entropy

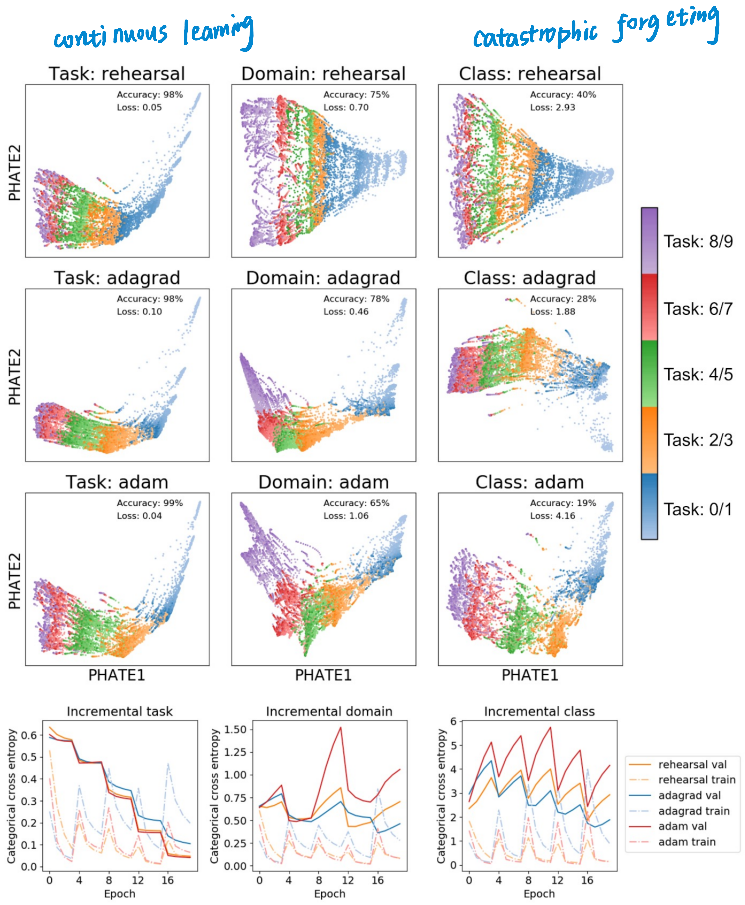
one neuron is 1 all others 0

bc it just memorize one input

Learning vs. Catastrophic Forgetting

when train NN to do 2nd task

it forget 1st task

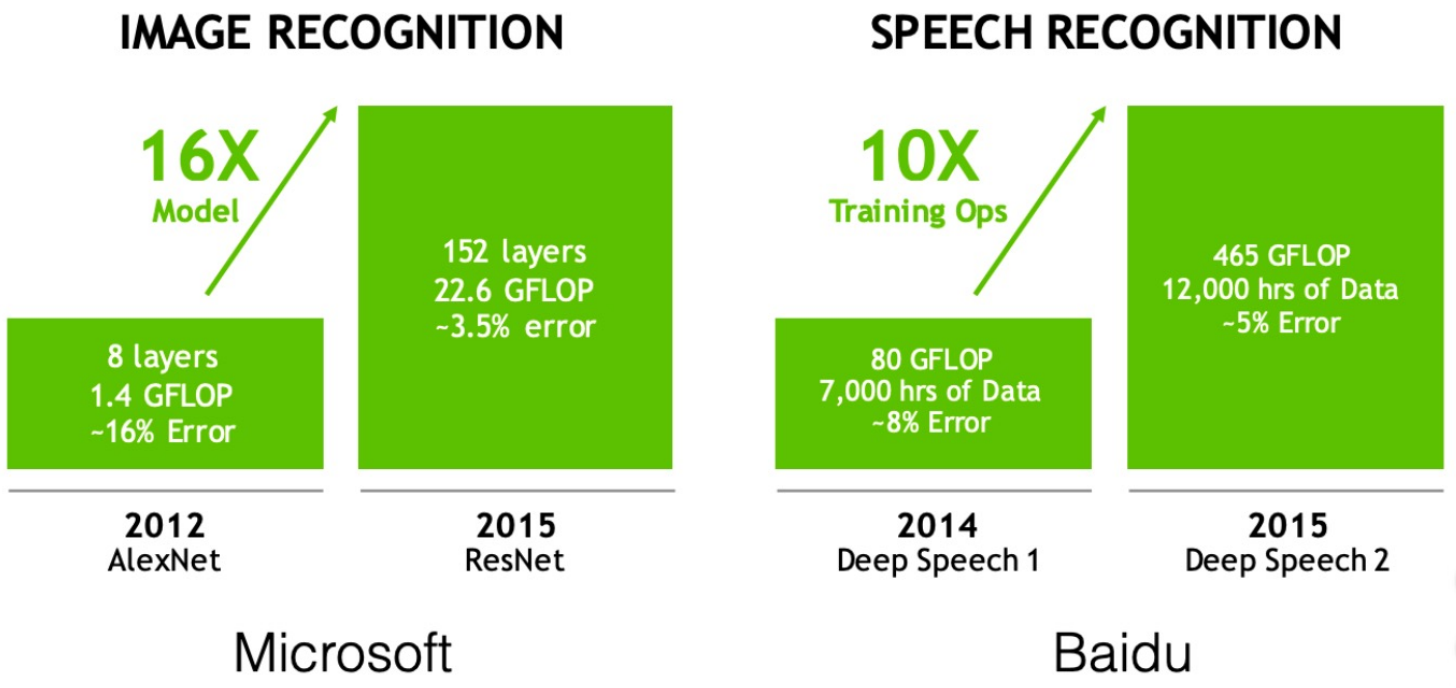


Expanding and shrinking

- We have shown that we can use depth to increase the power of a neural network
- Can we also shrink or prune a network?
- There could be many reasons to shrink or prune a network

Reducing Model Size

Larger Model => More Memory References => More Energy

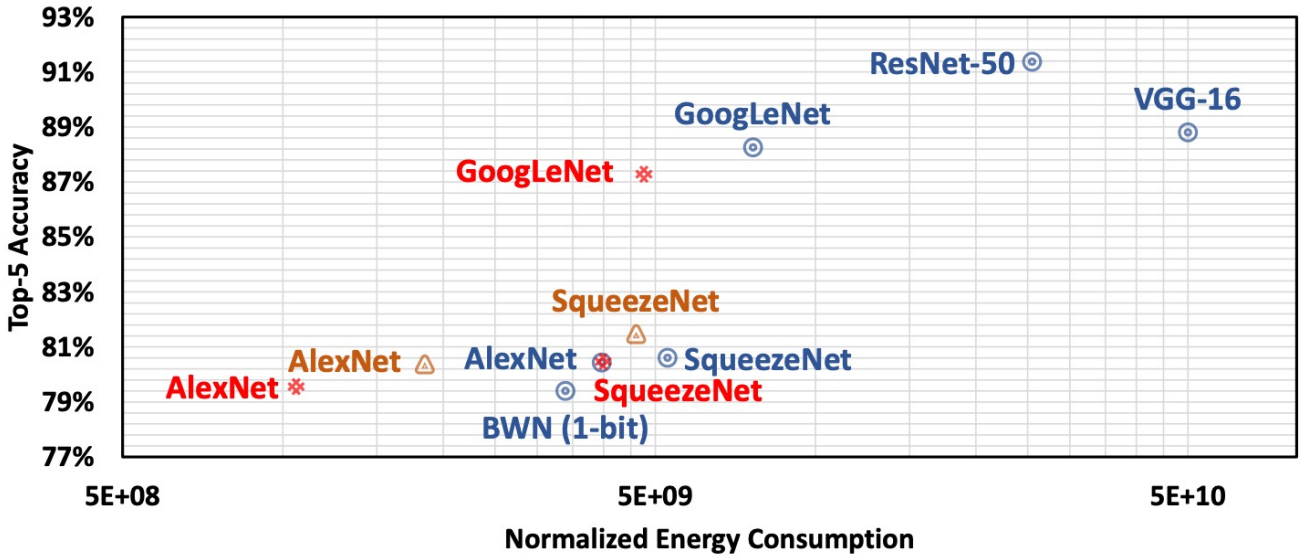


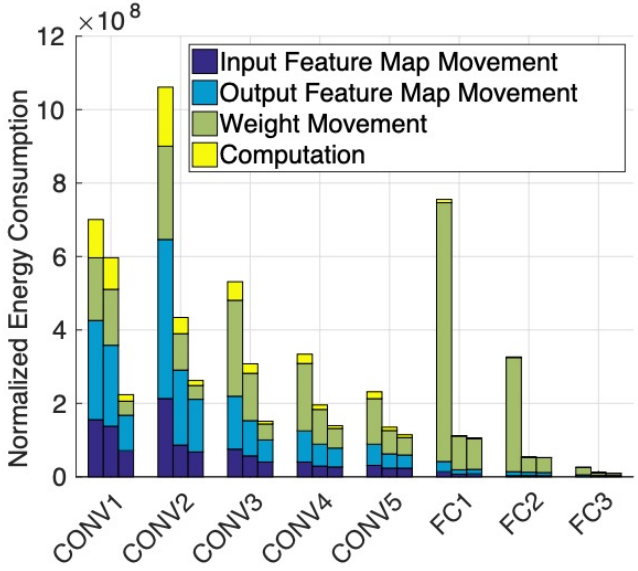
Difficulty Training Large Models

	Error rate	Training time
ResNet18:	10.76%	2.5 days
ResNet50:	7.02%	5 days
ResNet101:	6.21%	1 week
ResNet152:	6.16%	1.5 weeks

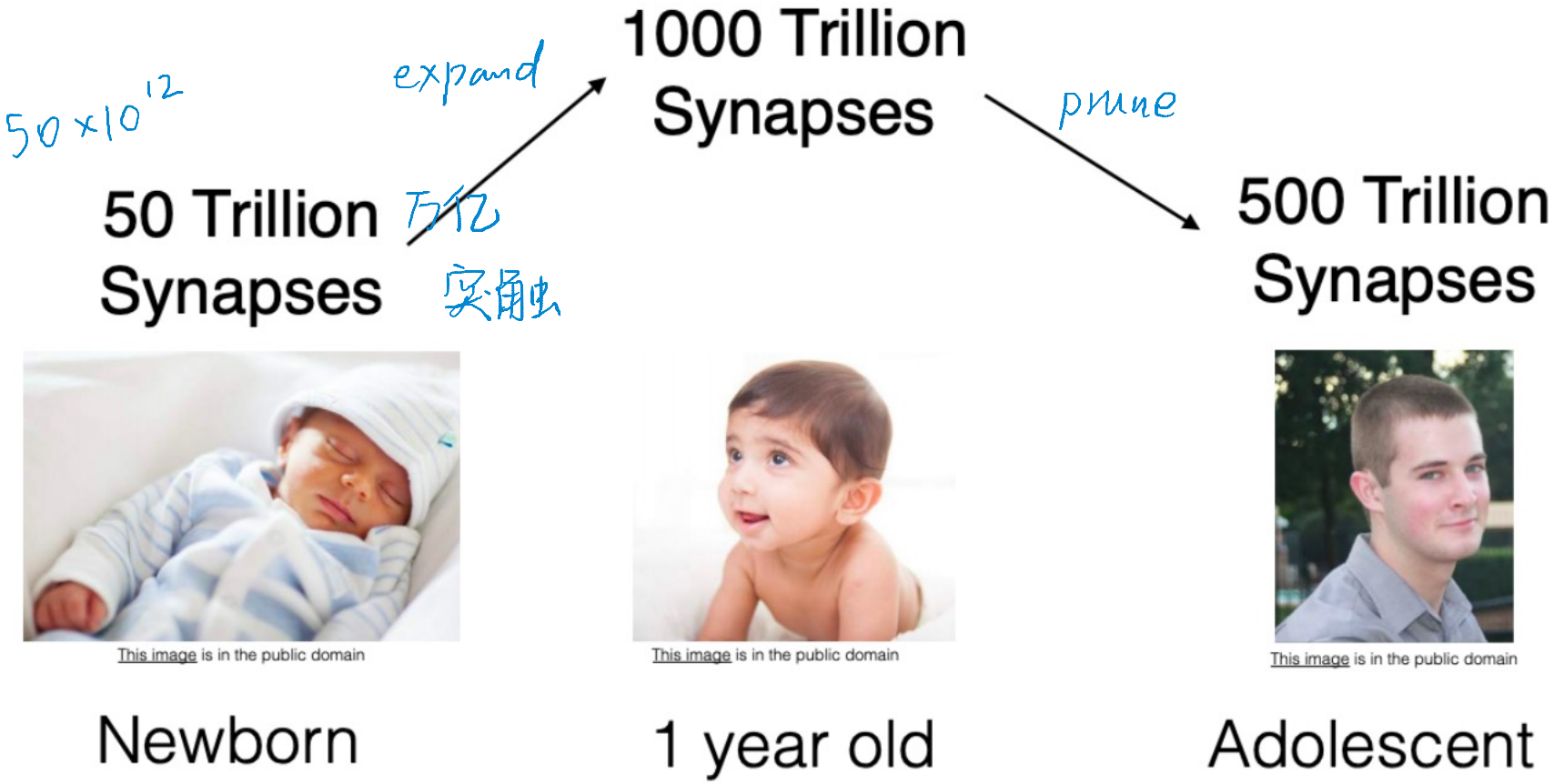
Training time benchmarked with fb.resnet.torch using four M40 GPUs

Energy consumption

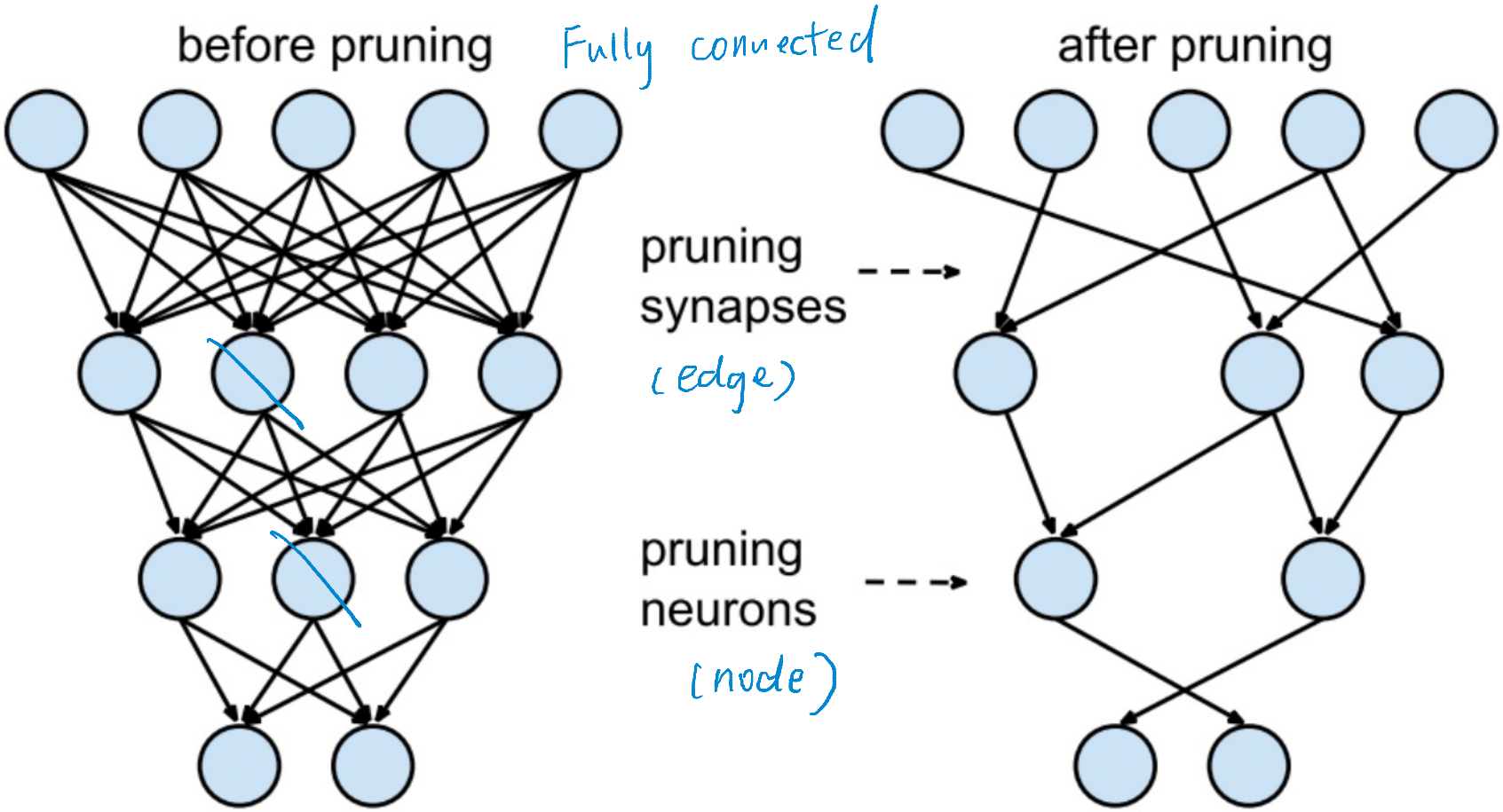




Biological Inspiration for Pruning



Neural Network Pruning



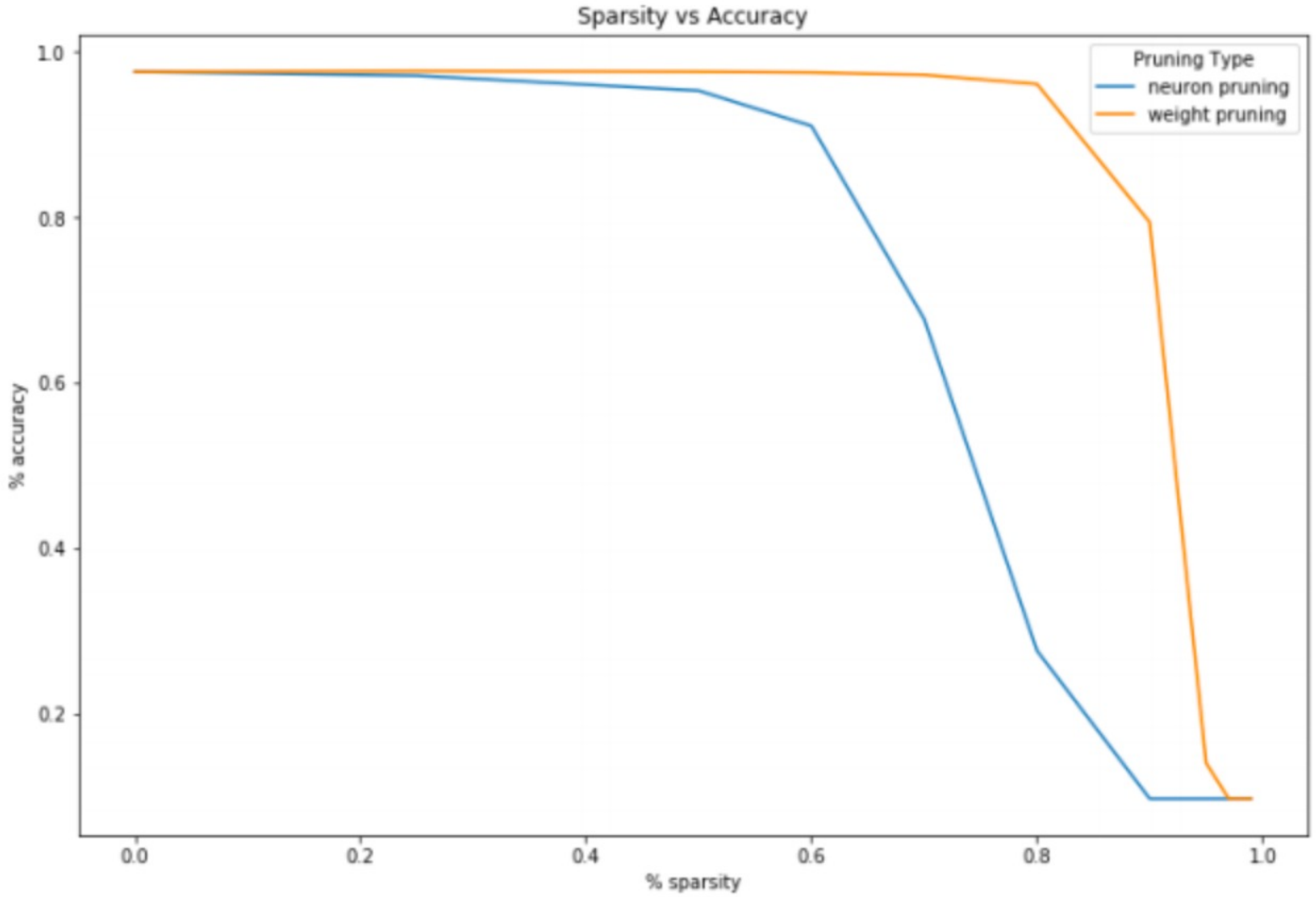
Weight Pruning

- Set individual weights in the weight matrix to zero. This corresponds to deleting connections
- To achieve k% sparsity rank the individual weights in weight matrix W according to their magnitude, and then set the smallest k% to zero.

Neuron Pruning

- Set entire columns to zero in the weight matrix to zero, in effect deleting the corresponding output neuron.
- Here to achieve sparsity of k% we rank the columns of a weight matrix according to their L2-norm and delete the smallest k%.

Pruning often Maintains Accuracy



Lottery Ticket Hypothesis

why?

- *Dense, randomly-initialized, feed-forward. networks contain subnetworks (winning tickets) that—when trained in isolation—reach test accuracy comparable to the original network in a similar number of iterations.* (all tickets)
- The winning tickets we find have won the initialization lottery: their connections have initial weights that make training particularly effective.

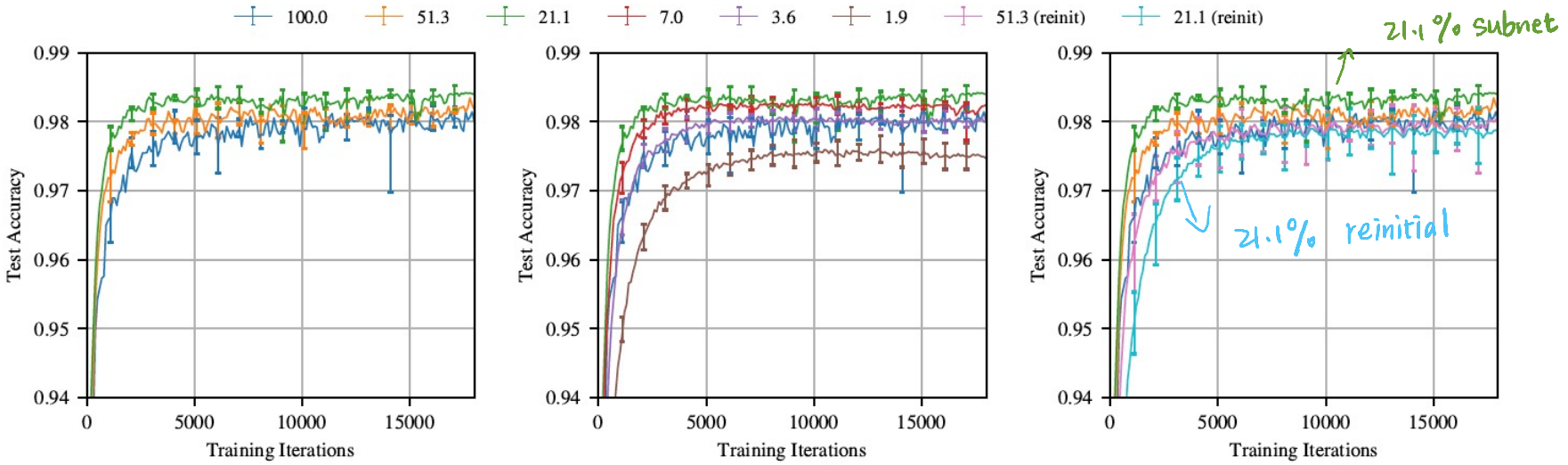
  but if initialize all weights to 0 no such lottery effect
- However, architectures uncovered by pruning are harder to train from the start
  - They reaching lower accuracy than the original networks

But if start from subnetwork → low accuracy

bc a good initialization isn't enough to capture connectivity pattern

Empirical Evidence

Test accuracy: 21.1% > 51.3% ≈ 7.0% ≈ 3.6% ≈ 100% > 51.3 reinit > 21.1 reinit > 1.9%



Further reading

- Tishby et al. DNNs and Information Bottleneck https://arxiv.org/abs/1503.02406
- Frankle & Carbin 2019 **THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE, TRAINABLE NEURAL NETWORKS** https://arxiv.org/pdf/1803.03635.pdf
- DasGupta & Gupta 2002 https://cseweb.ucsd.edu/~dasgupta/papers/jl.pdf
- Visualizing the PHATE of neural networks https://arxiv.org/abs/1908.02831