Structure of Learning Tasks and the Information in the Weights of a Deep Network

Giovanni Paolini (paolini@caltech.edu), Alessandro Achille (achille@cs.ucla.edu)
Amazon Web Services

Machine Learning

In a typical supervised learning problem, we are given a training dataset $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{N}$.

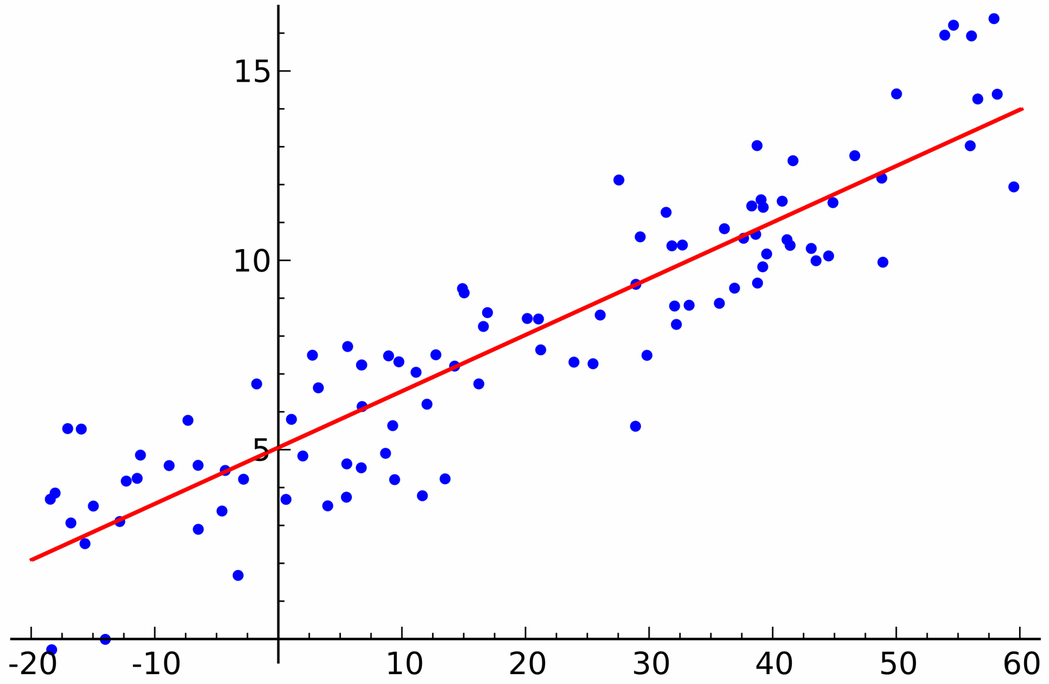
We want to learn a model that predicts the right output y for future inputs x.

We start from a parametric function $f_w(x)$, and look for a good set of parameters w by minimizing a loss function $\mathscr{L}_{\mathscr{D}}(w)$.

Example (linear regression)

Parametric function: $f_w(x) = w \cdot x$

L^2 loss: $\mathscr{L}_{\mathscr{D}}(w) = \frac{1}{N} \sum_{i=1}^{N} \left(f_w(x_i) - y_i\right)^2$

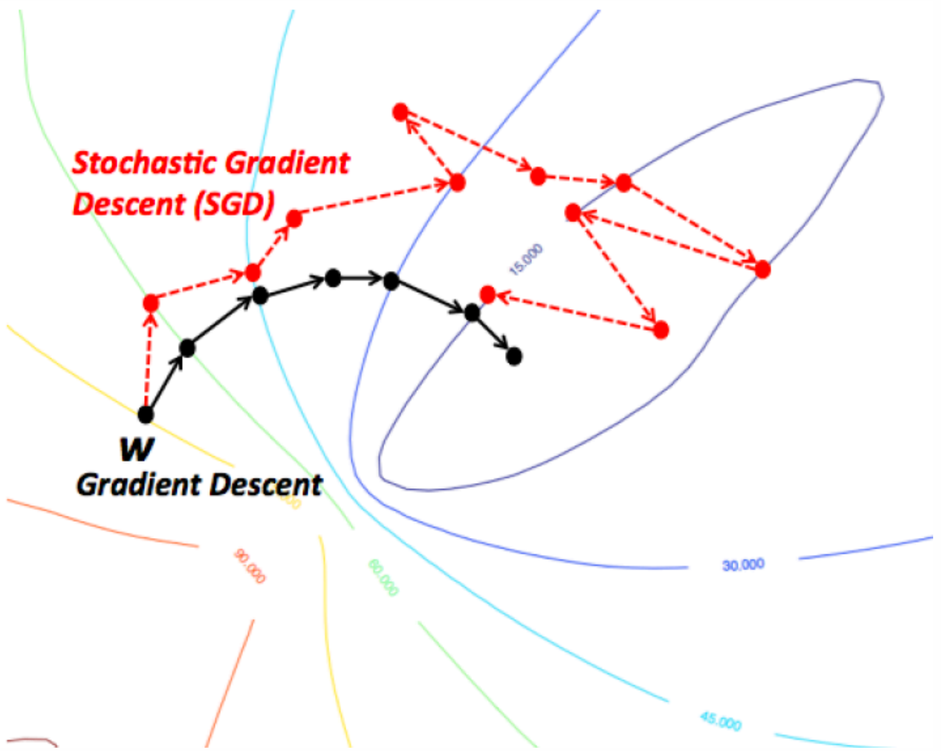


Picture by Sewaqu, https://commons.wikimedia.org/w/index.php?curid=11967659

How to find the parameters: Stochastic Gradient Descent

We want to find the parameters *w* that minimize a loss function

$$\mathscr{L}_{\mathscr{D}}(w) = \frac{1}{N}\sum_{i=1}^{N} \ell_w(x_i, y_i)$$



Algorithm (SGD):

1. Sample an example (x_i, y_i) from the dataset.
2. Compute the gradient of the per-sample loss $g_i := \nabla_w \ell_w(x_i, y_i)$
3. Update the network parameters $w' \leftarrow w - \eta\, g_i$

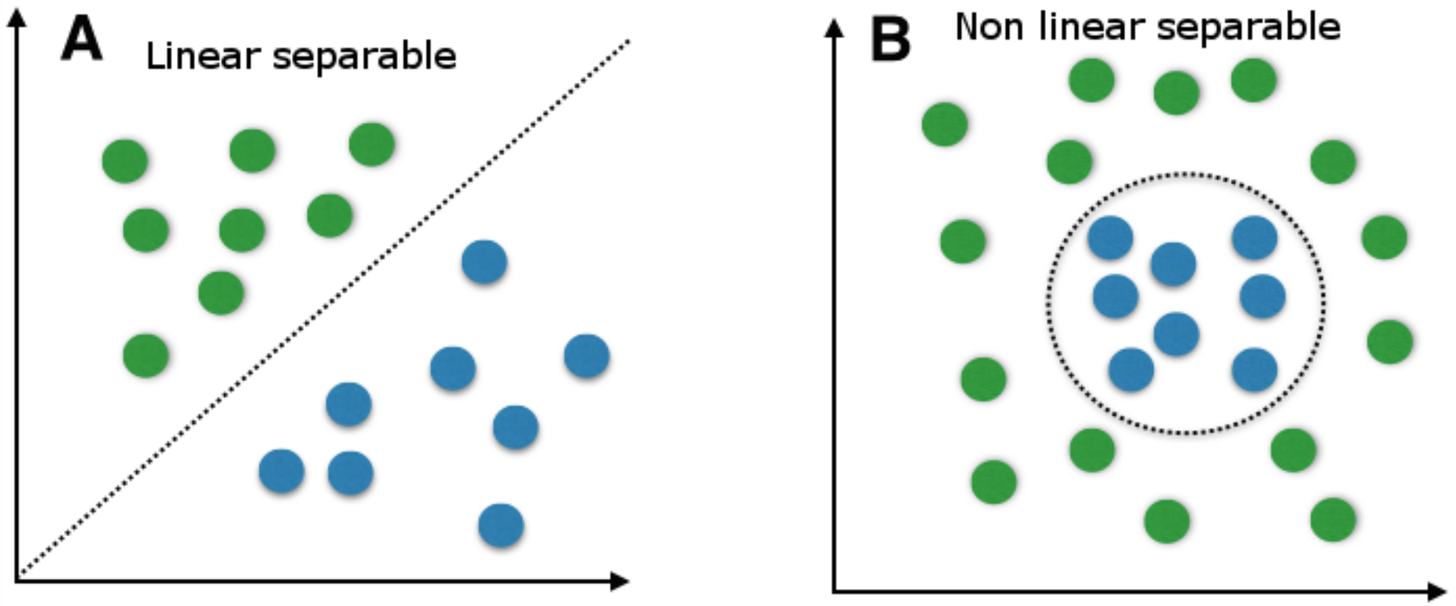
A prototype problem

Suppose that our task is to predict some real number associated with an input image.

Example: predict the age of a dog.

We want to learn a model $f_w(x)$ that predicts the right target value of future images.

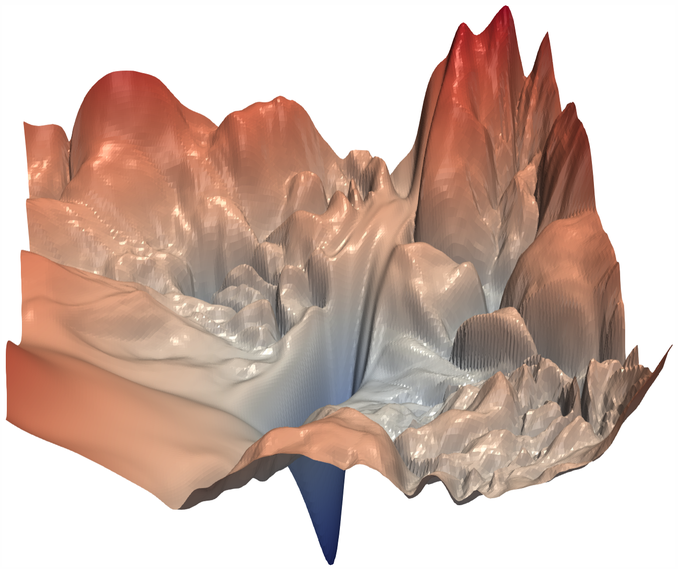
Linear is not enough: Deep Neural Networks



Deep neural networks: we can approximate any continuous function by alternating (parametric) linear functions and point-wise non-linearities σ, such as $\sigma(x) = \max(0, x)$.

$$f_w(x) = \sigma \circ \Phi_{w_L} \circ \sigma \circ \Phi_{w_{L-1}} \circ \ldots \circ \sigma \circ \Phi_{w_0}(x)$$

The loss landscape



$$\mathscr{L}_{\mathscr{D}}(w) = \frac{1}{N}\sum_{i=1}^{N}\left(f_w(x_i) - y_i\right)^2$$

The loss function has now many local minima with similar training error (overparametrization), but they can have very different test error (generalization).

This happens because the neural network can overfit to irrelevant information.

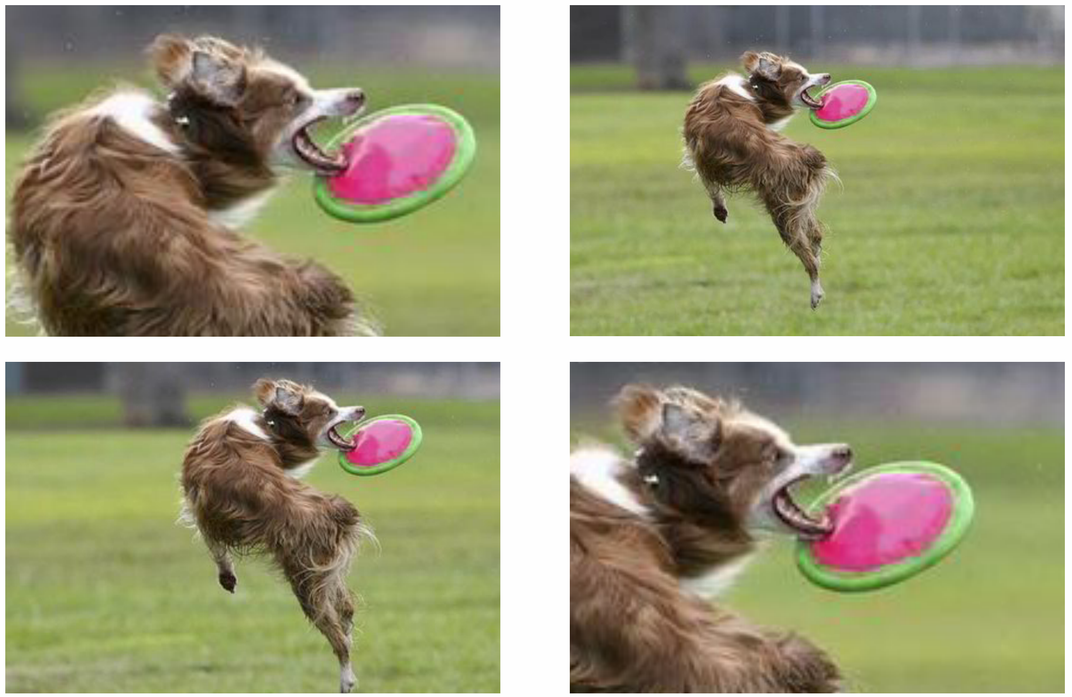
Li et al., *Visualizing the Loss Landscape of Neural Nets*, 2018

Geometric nuisances

Let a group G act geometrically on the data.

$x =$ “dog”

$g \cdot x =$ “dog” $\quad \forall g \in \mathrm{Aff}(\mathbb{R}^2)$



We want our function $f_w(x)$ to be group invariant.

G-invariance and G-equivariance

Let G act on two sets X and Y. A function $f\colon X \to Y$ is:

G-invariant if $f(g \cdot x) = f(x)$.

G-equivariant if $f(g \cdot x) = g \cdot f(x)$.

The composition of equivariant functions is equivariant.

Any equivariant function f can easily be made invariant, for example using
$\hat{f}(x) = \max_{g \in G} g \cdot f(x)$.

We can write an invariant function as a composition of simpler equivariant functions.

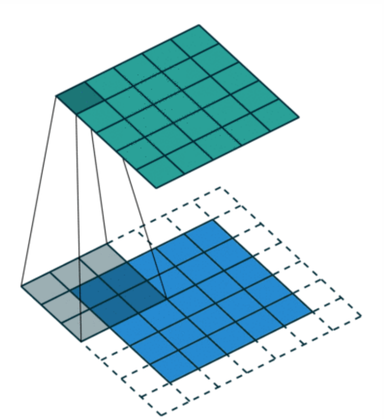
G-convolutions

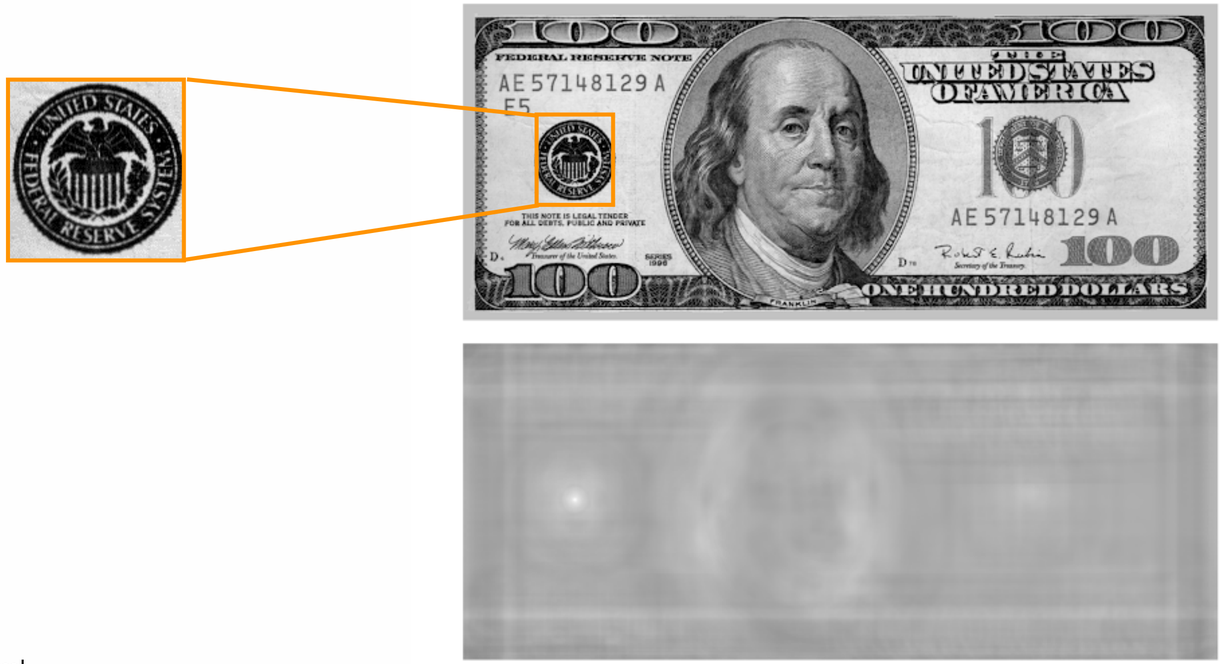
The only *G*-equivariant linear functions are *G*-convolutions:

$$x \star_G k\,(u) = \int_G x(ug^{-1})k(g)d\mu(g)$$

Example: Let $G = \mathbb{Z}^2$ be the translation group on $\mathbb{Z}^2$. We can think of an image as a function $x\colon \mathbb{Z}^2 \to \mathbb{R}$. The only translation equivariant operators are $\mathbb{Z}^2$-convolutions:

$$x \star k\,(u) = \sum_{t \in \mathbb{Z}^2} x(u-t)\,k(t)$$





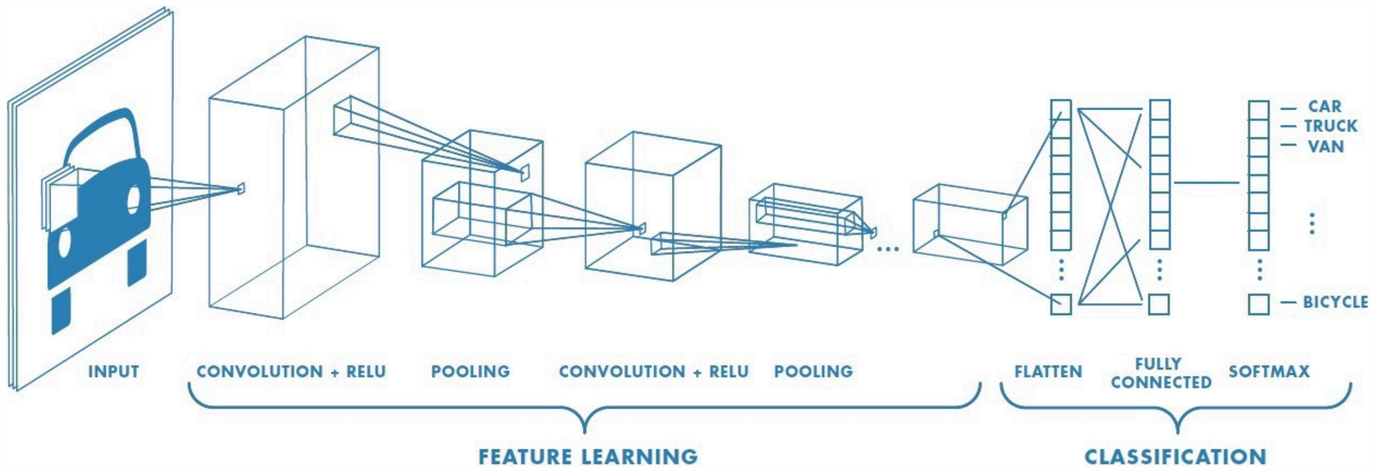
Picture from http://diplabs.blogspot.com/2012/04/template-matching-using-normalised.html

Deep Convolutional Neural Networks

We can approximate any G-equivariant function by alternating linear G-equivariant operations (convolutions) and point-wise non-linearities σ.

$$f_w(x) = \sigma \circ \Phi_{w_L} \circ \sigma \circ \Phi_{w_{L-1}} \circ \ldots \circ \sigma \circ \Phi_{w_0}(x)$$

We learn the convolution kernels by the loss function using SGD.

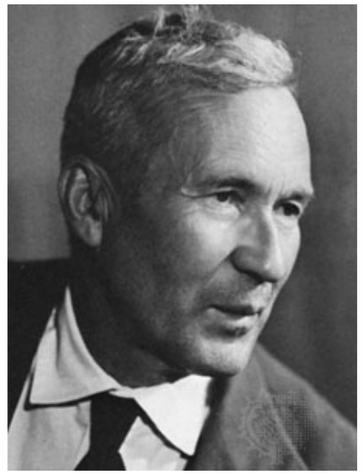


Structural information vs Noise

The fact that is a rainy outdoor scene is structural information of the image, the positions of the rain drops is pure randomness (noise).

Picture from https://buffalonews.com/2019/06/16/rain-will-return-this-week-after-record-setting-saturday/

Kolmogorov's complexity



The Kolmogorov complexity of a string is the length of the shortest program that can output that string. (Defined up to an *O(1)* factor)

Examples

A random sequence of length n of 0 and 1's:

$$x = 10001110110\ldots1001010011010 \qquad K(x) = n + O(1)$$

A repeating pattern of 0 and 1's has:

$$x = 10101010101\ldots.101010101010 \qquad K(x) = O(1)$$

The digits of π are statistically random, but have low complexity:

$$x = 3.14159265358979323846264 3\ldots \qquad K(x) = O(1)$$

The Kolmogorov Structure Function

Define the Kolmogorov Structure Function as:

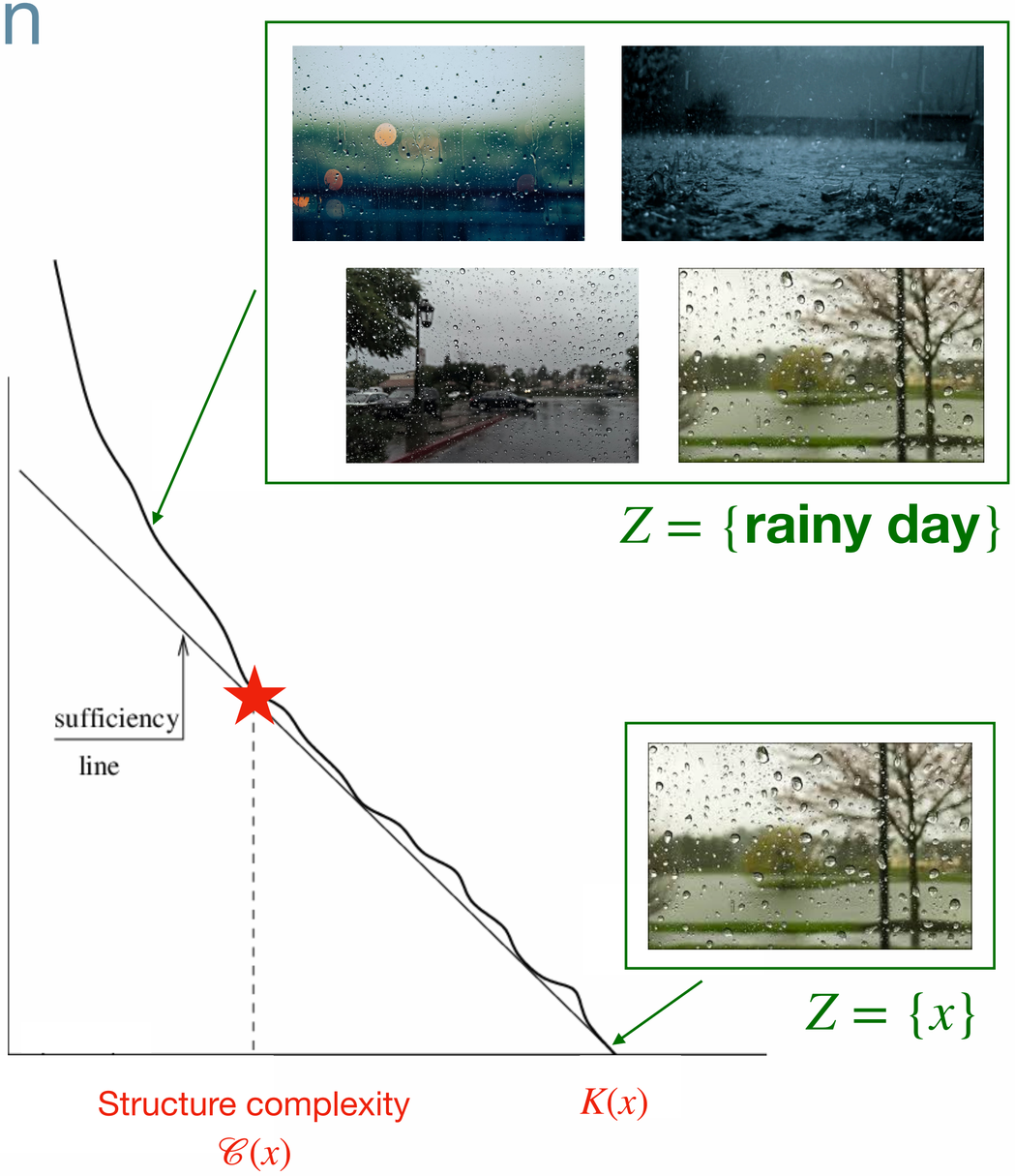
$$S_x(t) = \min_{K(Z) \leq t,\ x \in Z} \log |Z|$$

Cost of encoding the "model" Z

Cost of encoding the string x using the model

And the Structure Complexity as:

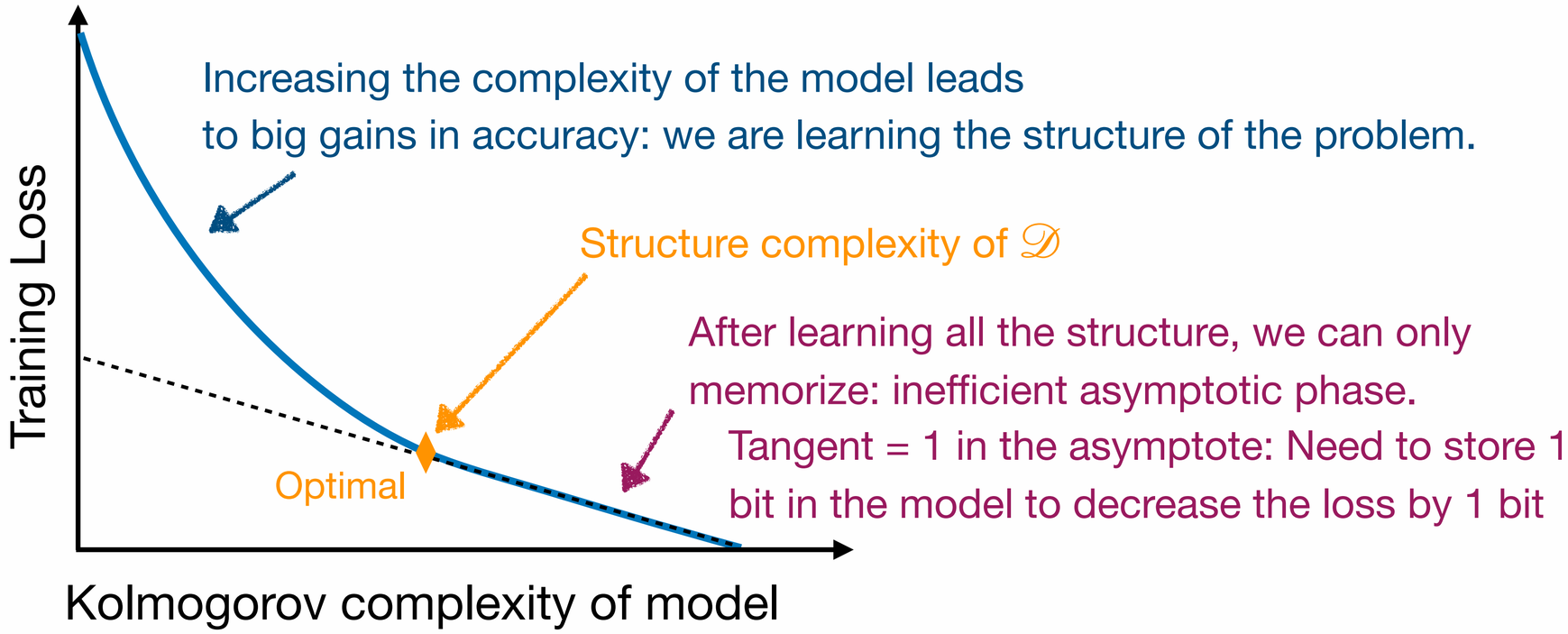
$$\mathscr{C}(x) = \min_{K(Z)+\log|Z|=K(x)} K(Z)$$



Vereshchagin and Vitanyi, *Kolmogorov's Structure Functions and Model Selection*, 2002

The Structure Function of a dataset

The structure function of the dataset $\mathscr{D}$ is: $S_{\mathscr{D}}(t) = \min_{K(w) \leq t} L_{\mathscr{D}}(w)$



But how is $K(w)$ measured?

A., Mbeng, P., Soatto, *Information Complexity of Tasks, their Structure and their Distance*, 2019

The Information in the Weights

$$S_{\mathscr{D}}(t) = \min_{K(w) \leq t} L_{\mathscr{D}}(w) \quad \text{equivalently} \quad \boxed{\min_{L_{\mathscr{D}}(w) \leq t'} K(w)}$$

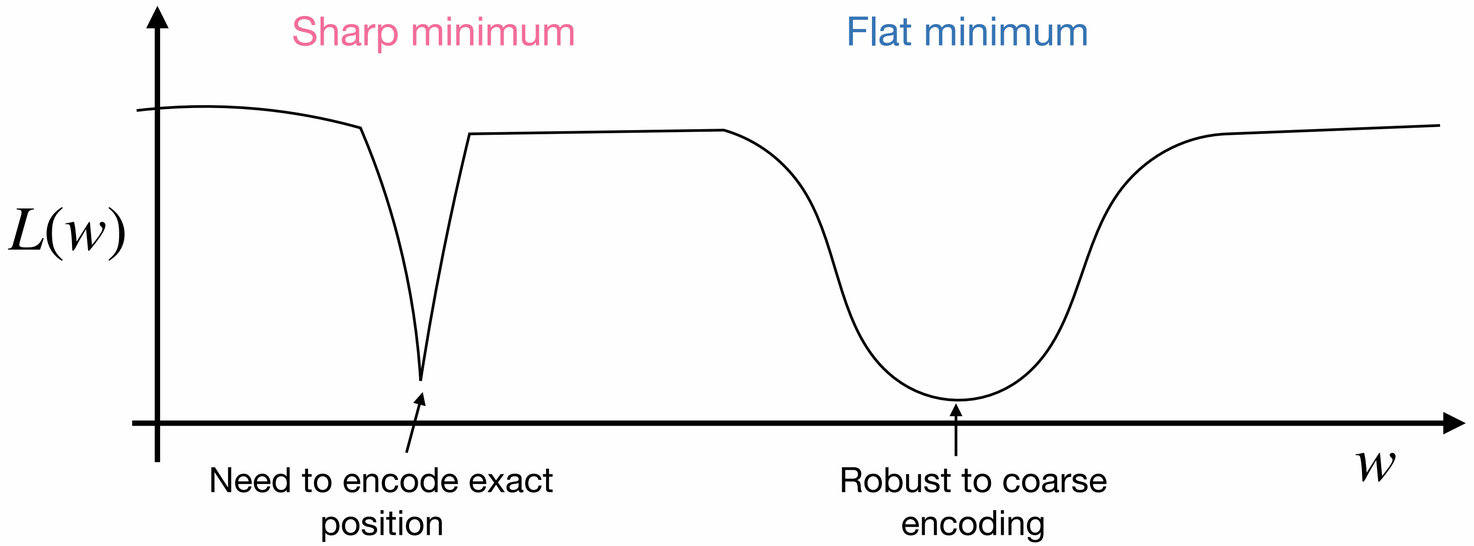
How do we measure the complexity of a DNN? Codifying a particular set of weights as real numbers requires infinite information.

Keeping Neural Networks Simple by Minimizing the Description Length of the Weights

Geoffrey E. Hinton and Drew van Camp
Department of Computer Science
University of Toronto
10 King's College Road
Toronto M5S 1A4, Canada

Idea: Codifying noisy weights requires *finite* information

Information in weights and flatness



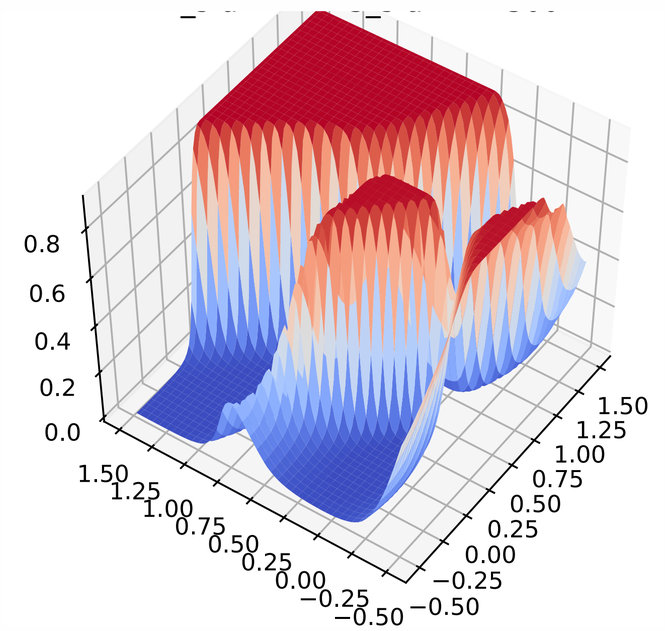
Proposition. The minimum encoding length of the weights (to obtain a given loss) is upper-bounded by the flatness of the minimum:

$$K(w) \leq \frac{\|w\|^2}{\lambda^2} + \det(\lambda^2 H(w) + I)$$

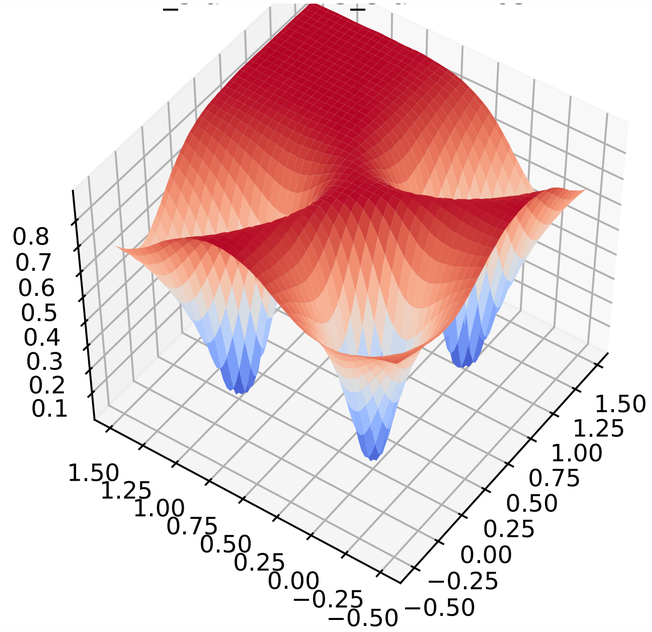
Where *H(w)* is the Hessian of the loss function computed in *w*. Changing λ changes the trade-off between *L(w)* and *K(w).*

Sharp vs Flat Minima

Fitting random labels (pure memorization) leads to sharper minima in practice.



(a) natural label

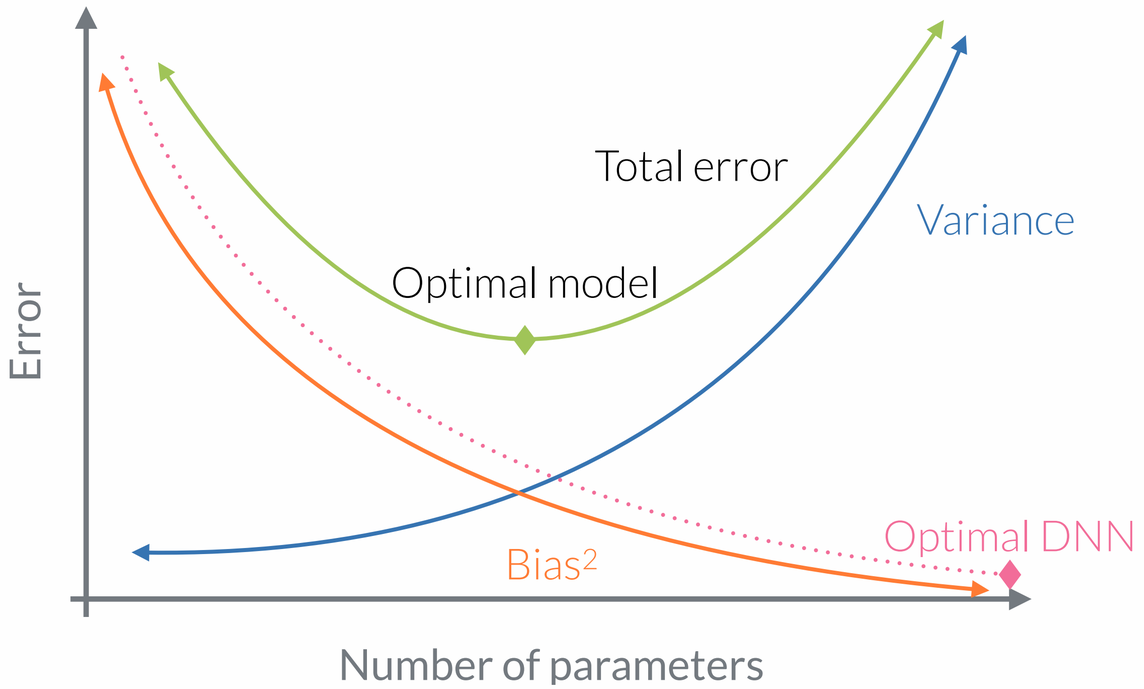


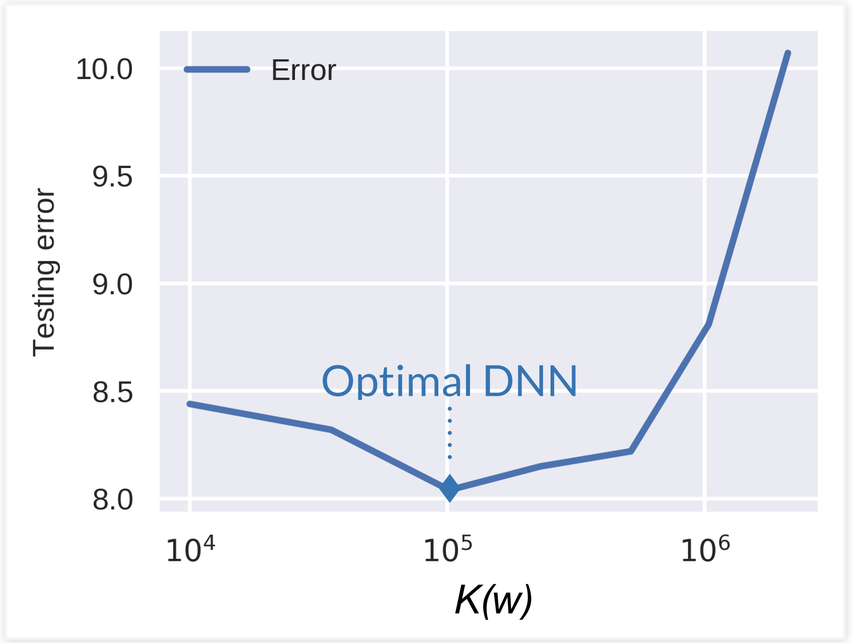
(b) random label

This is compatible with sharpness being a measure of information in the weights.

Pictures from Zhang et al, Theory of Deep Learning III: Generalization Properties of SGD, 2017

Bias-variance tradeoff

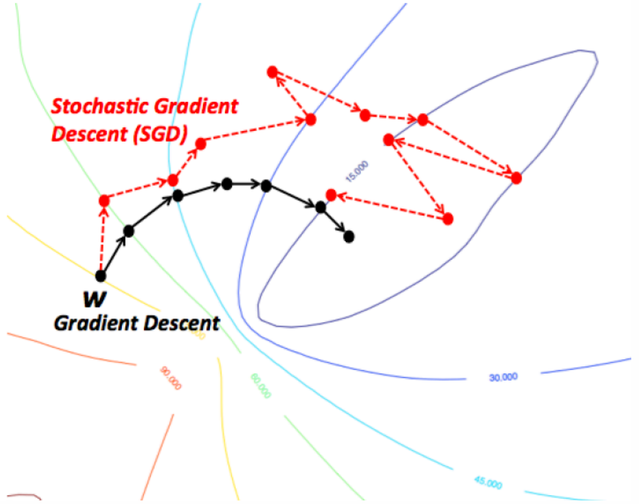




What matters for generalization is not the number of weights, but the information they contain.

Achille and Soatto, Emergence of Invariance and Disentanglement in Deep Representations, JMLR 2018

How to find flat minima: SGD



Recall that we minimize the loss function using SGD.

$$L_D(w) = \frac{1}{N}\sum_{i=1}^{N} \ell_w(x_i, y_i)$$

Let $g_i = \nabla_w \ell_w(x_i, y_i)$ be the gradient of the *i*-th sample. Note that $\nabla_w L_D(w) = \mathbb{E}[g_i]$.

Hence: we can think of $g_i = \nabla_w L_D(w) + n_i$ as being a noisy version of the real gradient:

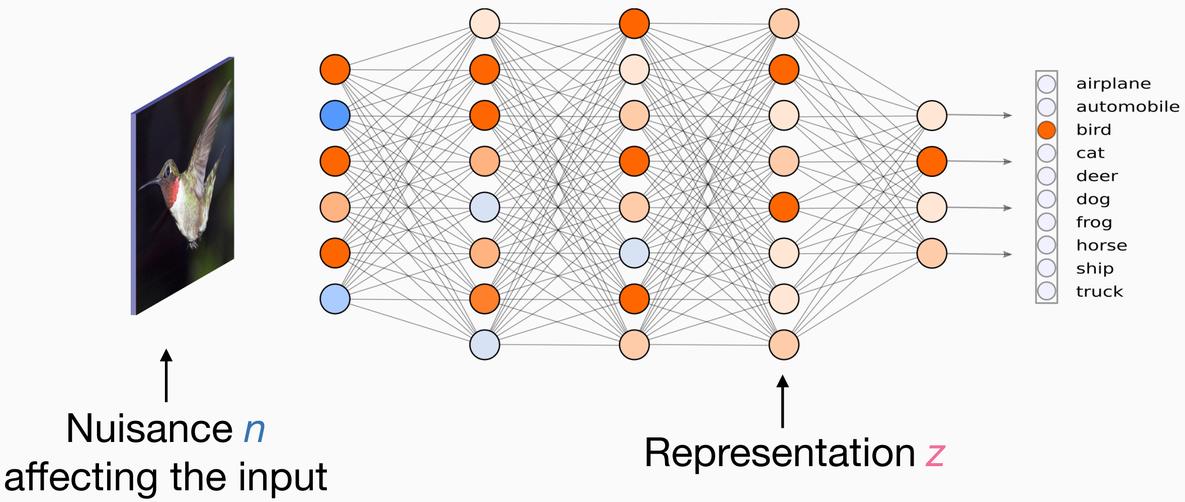
GD $\quad w_{t+1} \leftarrow w_t - \eta \nabla_w L_D(W)$

SGD $\quad w_{t+1} \leftarrow w_t - \eta \nabla_w L_D(W) + \boxed{\eta\, n_i}$

SGD is gradient descent plus noise

Proposition. SGD is more likely to escapes from sharp minima than GD.

Flatness and Invariance to Nuisances



Proposition. Let z = f_w(x) be a layer of a network. To a first order approximation, the information that it contains about a nuisance is bounded by:

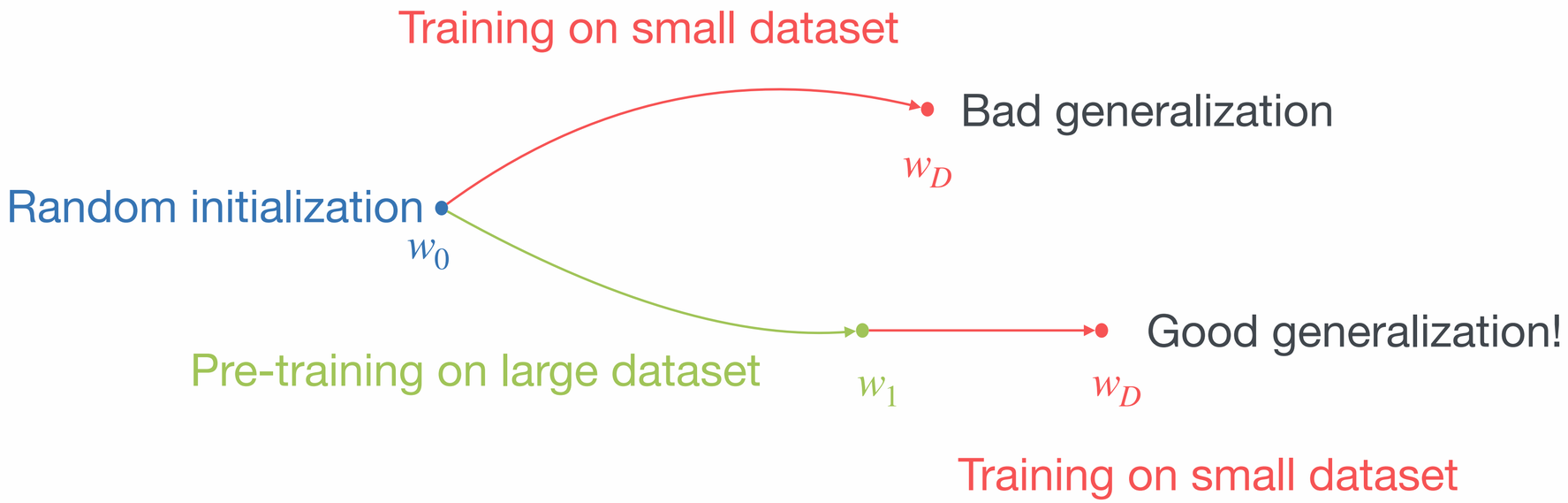
$$I_{\text{eff}}(z; n) \leq C + \log\det\left(\left|\nabla_x f_w(x)^t\, J_f^t\, \underbrace{H(w)}_{\text{Flatness}}\, J_f\, \nabla_x f_w(x)\right|\right)$$

where *F(w)* is the Fisher Information of the weights, J_f is the Jacobian of f_w w.r.t. *w.*

A. and Soatto, *Emergence of Invariance and Disentanglement in Deep Representations*, JMLR 2018

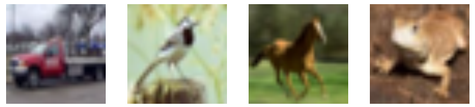
Transfer learning

Deep Networks have wide applications because they can transfer information between tasks.



What is the transferable information between tasks?

What is the distance between two tasks?



CIFAR-10

ImageNet

KITTI

SVHN

MNIST

Fashion MNIST

A Topology on the Space of Tasks

Distance between tasks:

$$d(\mathscr{D}_1 \to \mathscr{D}_2) = C(\mathscr{D}_1\mathscr{D}_2) - C(\mathscr{D}_1)$$

Complexity of learning together

Complexity of learning one

That is, how much more structure do we need to learn?

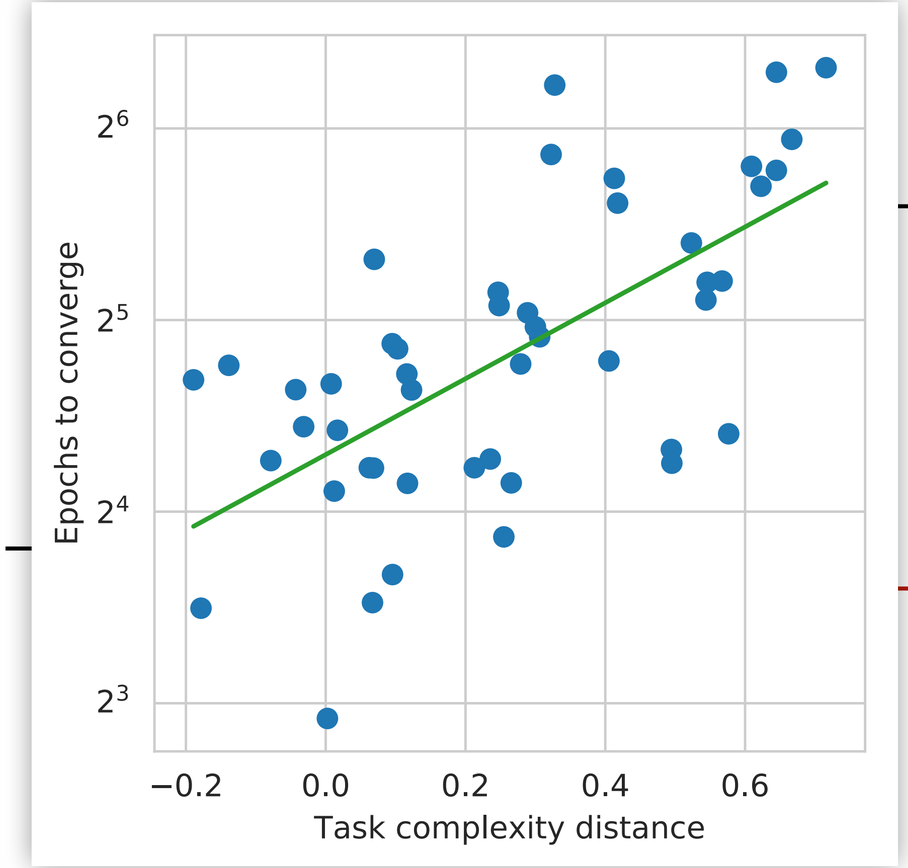
Notice that this is an asymmetric distance

A., Paolini, Mbeng, Soatto, *The Information Complexity of Learning Tasks, their Structure and their Distance*, 2019

A Topology on the Space of Tasks

$$d(\mathscr{D}_1 \to \mathscr{D}_2) = K(\mathscr{D}_1\mathscr{D}_2) - K(\mathscr{D}_1)$$

Easy task to difficult task →

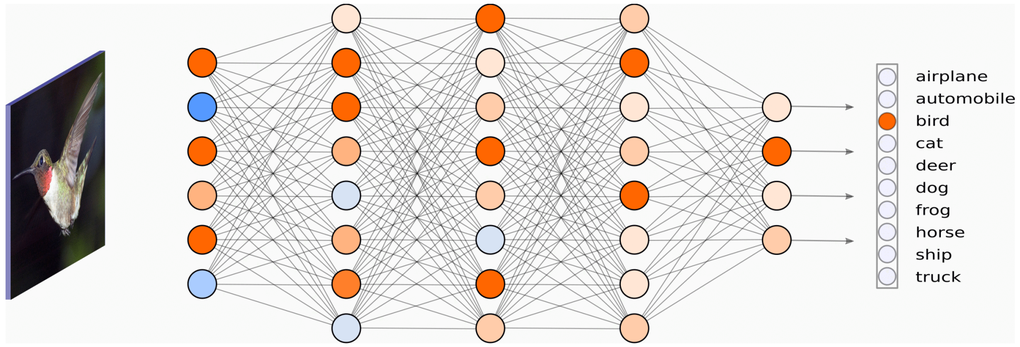


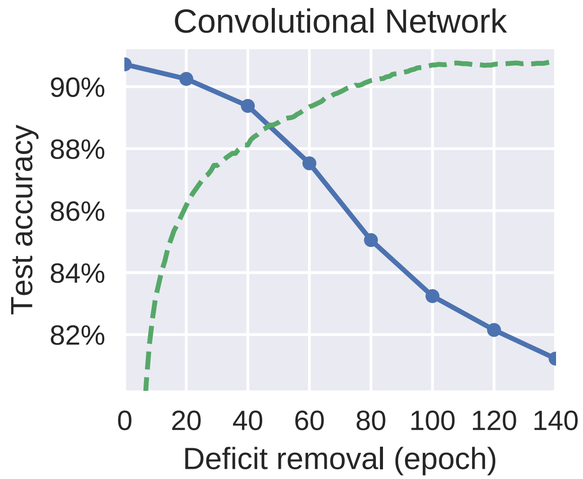
→ Difficult task to easy task

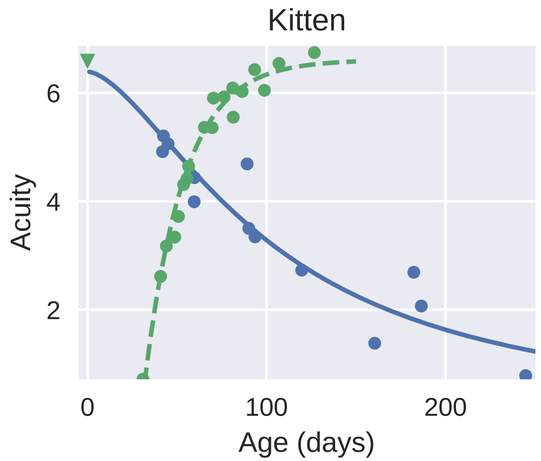
→ Similar tasks cluster together

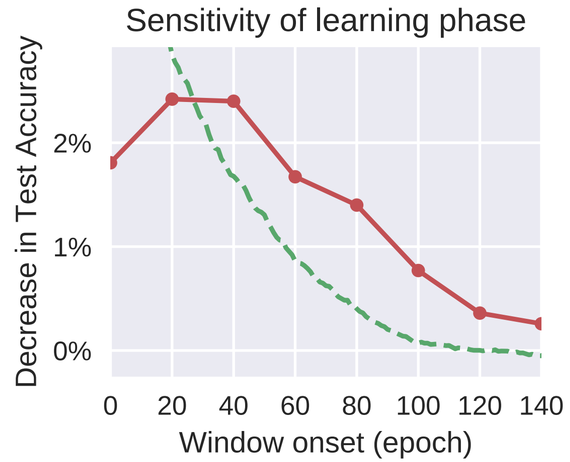
A., Paolini, Mbeng, Soatto, *The Information Complexity of Learning Tasks, their Structure and their Distance*, 2019

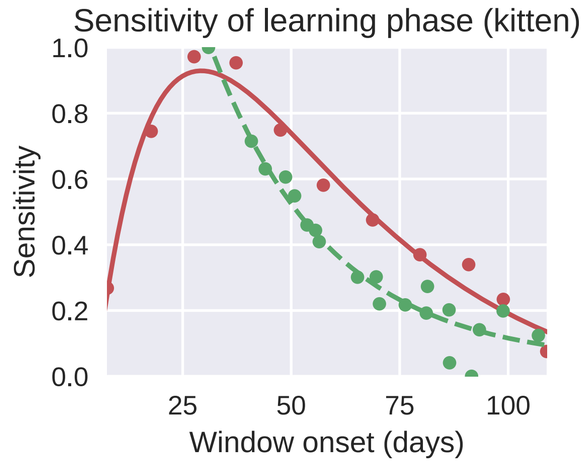
Critical Learning Periods in Deep Networks









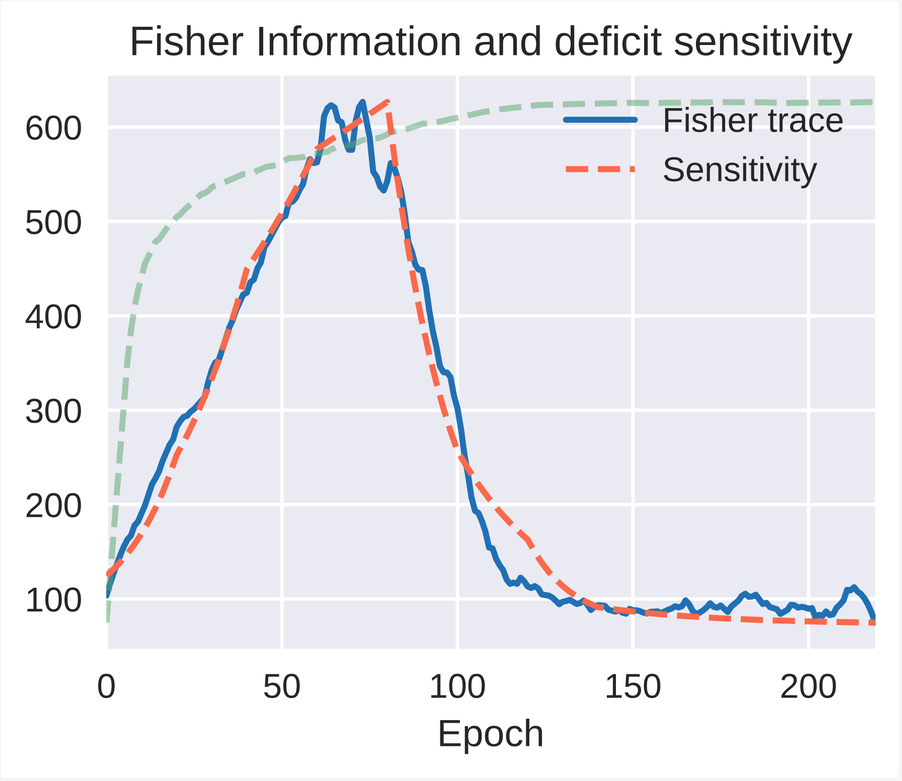


A., Rovere, Soatto, *Critical Learning Periods in Deep Networks, 2018*

Critical learning periods and Information in Weights

Sensitivity to deficits peaks when network is absorbing information.

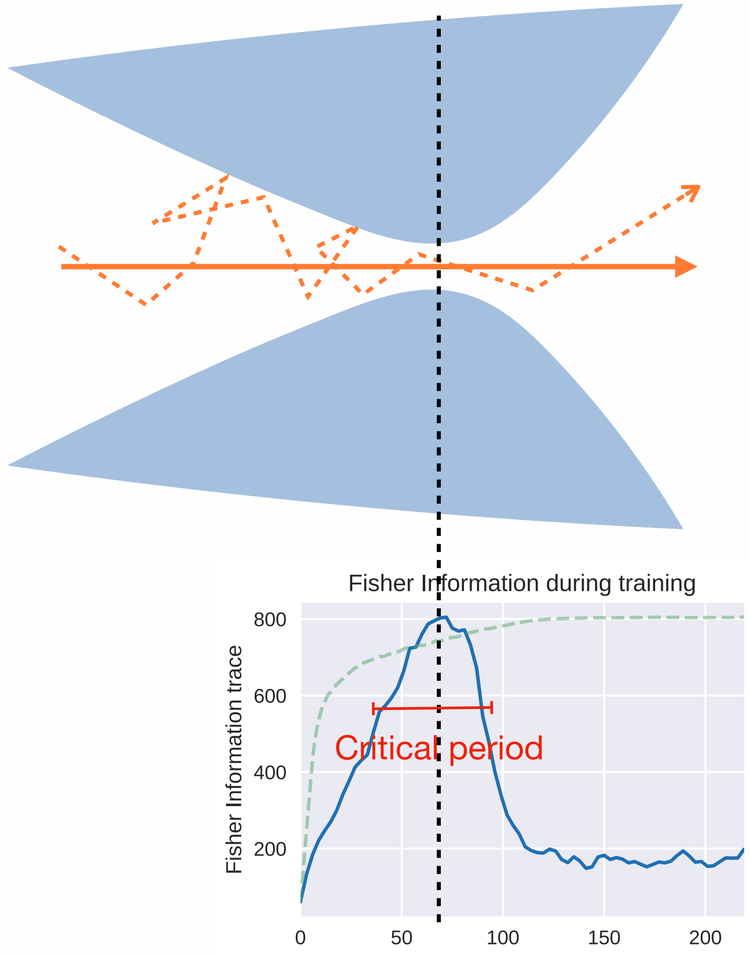
Is minimal when the network is consolidating information.



Achille, Rovere, Soatto, *Critical Learning Periods in Deep Networks, 2018*

Information is physical

How can the Fisher Information affect the learning dynamics?

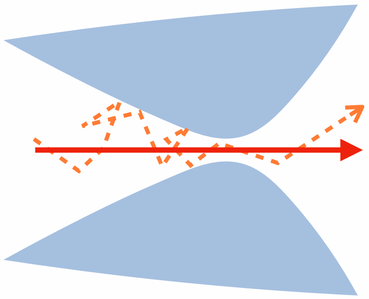


A path-integral approximation

1) Approximate SGD with gradient descent + white noise. Use MSR formalism to obtain probability of following a path *w(t)*:

$$p(w(t)|w_0, t_0) = e^{\frac{1}{D}\int \mathcal{L}}$$

2) Assume most path are perturbations of distinct “critical” paths:



3) Approximate the loss function quadratically along critical paths, and integrate out the perturbations to find total probability of crossing bottleneck:

$$p(w_f, t_f|w_0, t_0) = e^{-\Delta\mathcal{L}(w;\mathcal{D})} \int_{w_0}^{w_f} e^{-\frac{1}{2D}\int_{t_0}^{t_f} \frac{1}{2}\dot{u}(t)^2 + V(u(t))dt} du(t)$$

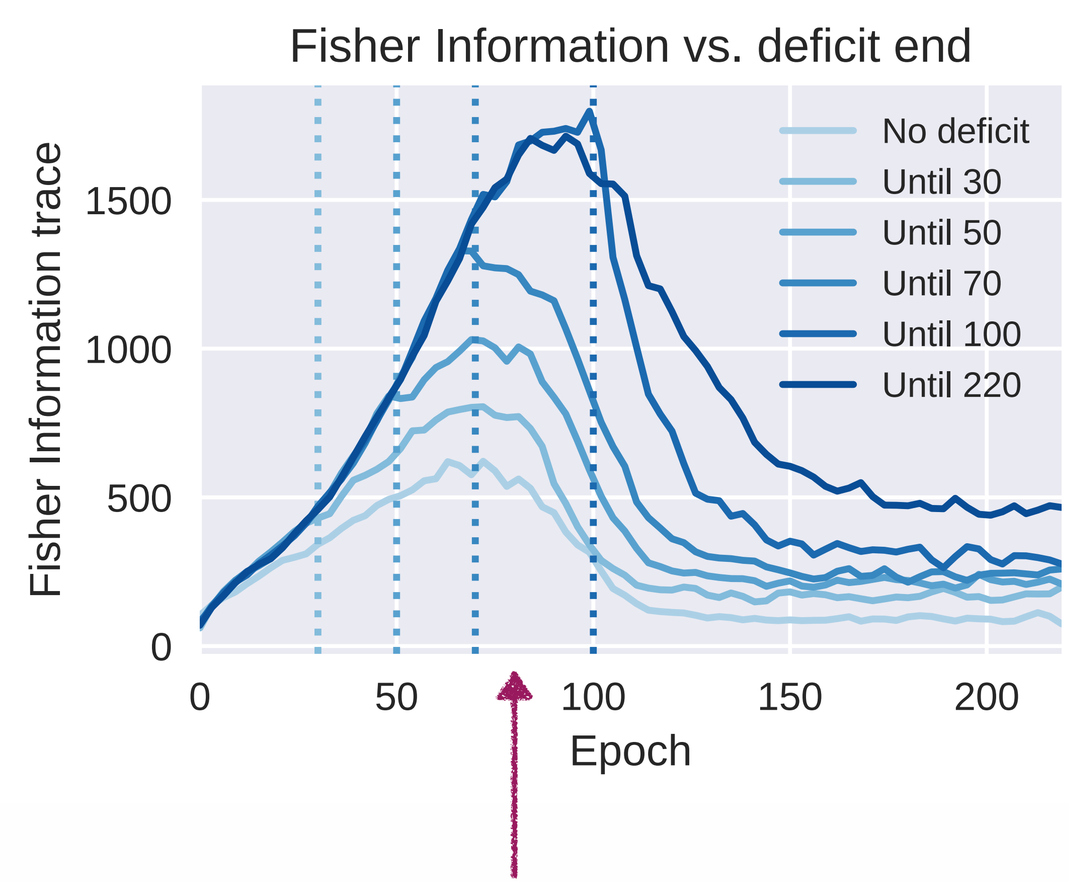
Static part

Depends only on the difference in information between initial and final point

Dynamic part

Depends on the existence of likely path between the two

Are flat minima an epiphenomenon?



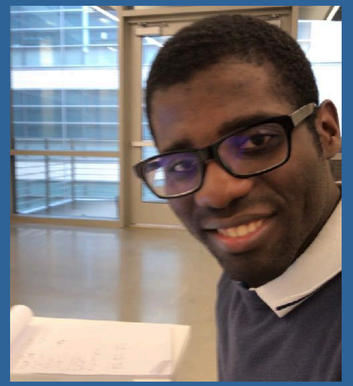
Final sharpness correlates with generalization…

…but generalization quality is decided here, far from convergence to minima

THANKS!

Matteo Rovere

Giovanni Paolini



Glen Mbeng

Stefano Soatto

The PAC-Bayes generalization bound

PAC-Bayes bound on the test error: (Catoni, 2007; McAllester 2013)

$$L_{\text{test}} \leq \frac{1}{1 - \frac{1}{2\beta}} \left[\mathbb{E}_w [L_{\mathcal{D}}(w)] + \beta \, \mathrm{KL}(q(w|\mathcal{D}) \,\|\, p(w)) \right]$$

Moreover, the sharpest bound is obtained when $E[\mathrm{KL}] = I(w; D)$.

What matters for generalization is not the number of weights, but the information they contain.

This gives non-vacuous generalization bounds.

A. and Soatto, *Emergence of Invariance and Disentanglement in Deep Representations*, JMLR 2018
Dziugaite and Roy, *Computing non-vacuous generalization bounds for deep neural networks, UAI 2017*