Matteo Rovere Giovanni Paolini Glen Mbeng Stefano Soatto

Information in Deep Neural Networks

Alessandro Achille (achille@cs.ucla.edu), Stefano Soatto

University of California, Los Angeles

Overview

What is a task

Making a decision based on the data

Classification: Decide the class of an image (the prototypical supervised problem)

Survival: Decide the best actions to take to survive (Reinforcement Learning)

Reconstruction: Decide which information to store to reconstruct the data (generative models, unsupervised learning)

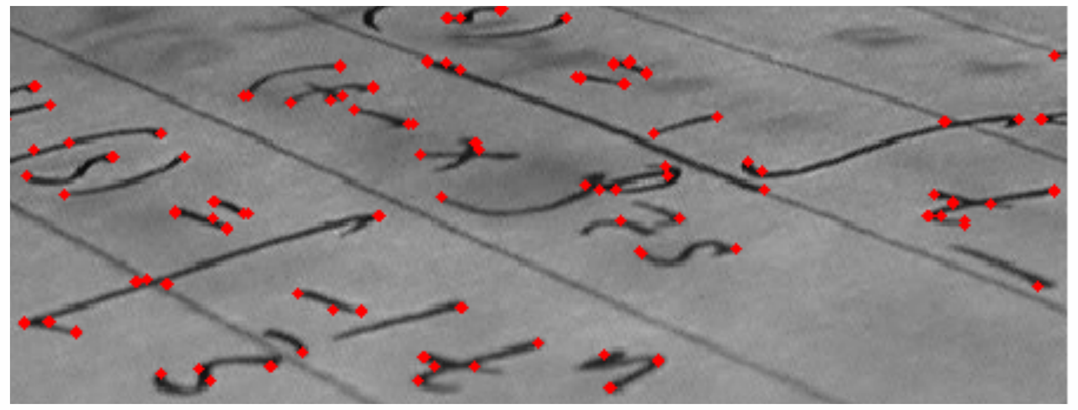
What is a representation

Any function of the data which is useful for a task.

Brightness

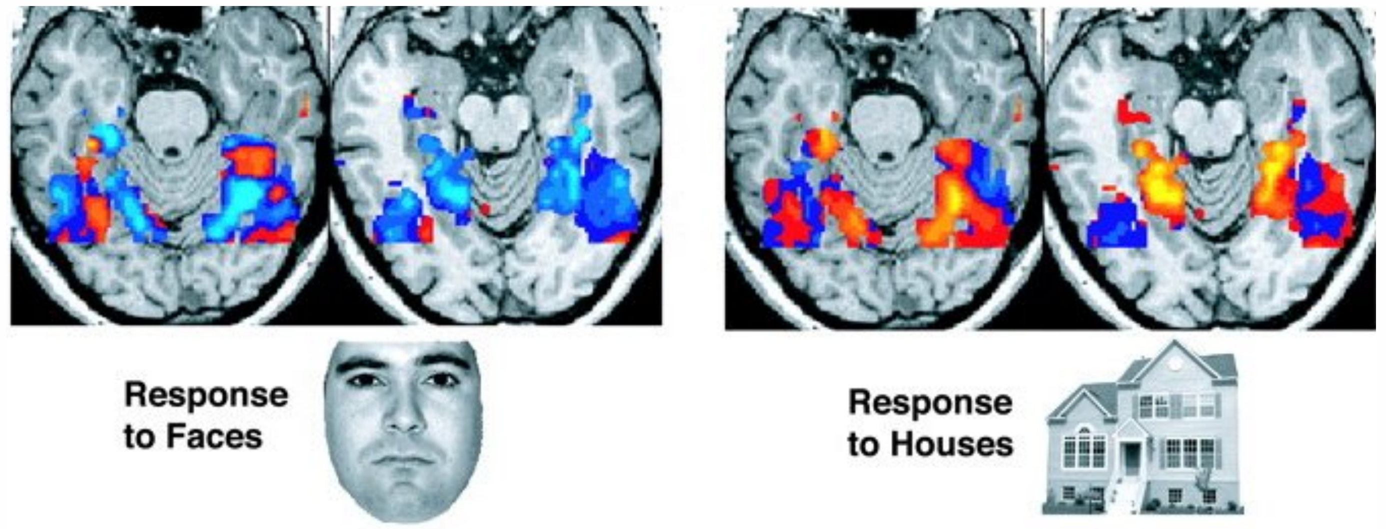
A simple organism may only need to know the direction of the light source

Corners



Popular in Computer Vision before DNNs, central to visual inertial systems and AR.

Neuronal activity



Hidden Layer

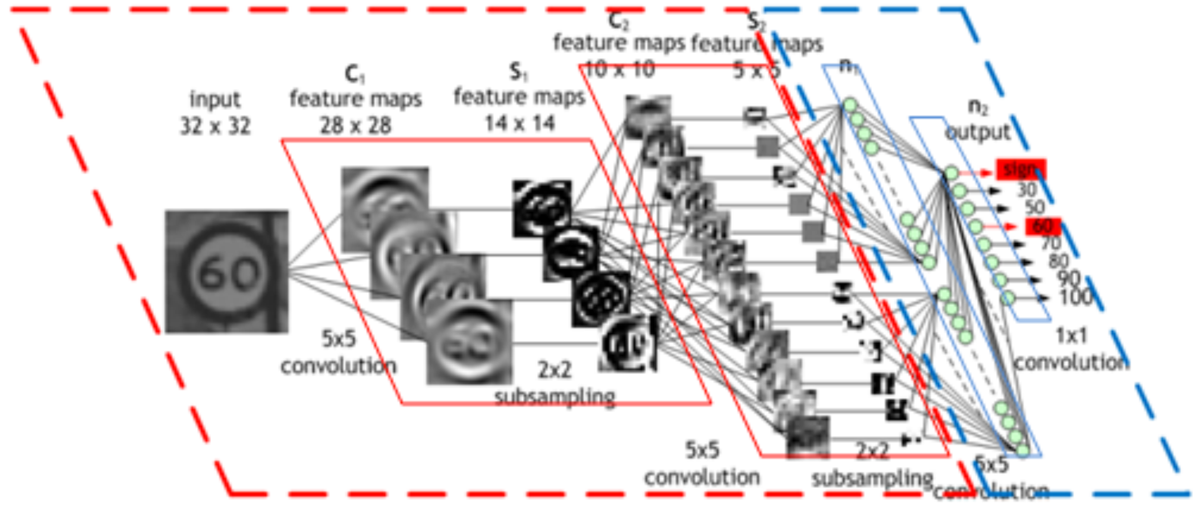
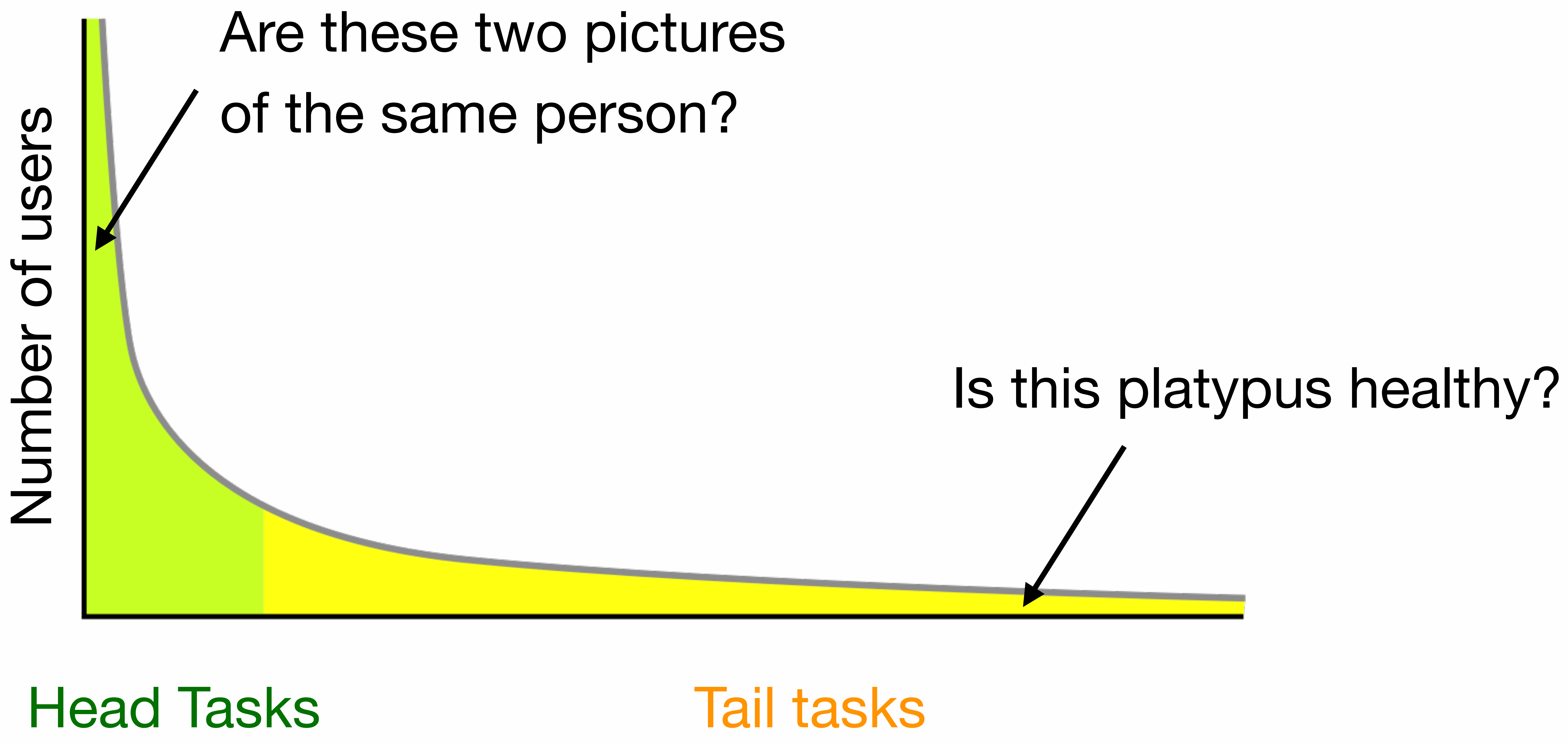


Image sources https://en.wikipedia.org/wiki/Functional_magnetic_resonance_imaging#/media/File:Haxby2001.jpg, https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Why studying representations in practice?

We can try to solve to the most common tasks, but what about the tails?



Idea: Provide the user with a powerful and flexible representation that allows them to easily solve their task.

Some questions in representation learning

1. What is the best representation for a task?
2. Which tasks can we solve using a given representation?

   The representation used by a health provider is probably not useful to a movie recommendation system.
3. Can we fine-tune a representation for a particular task?
4. Can we provide the user with error bounds? Privacy bounds?

But what is a good representation?

Data Processing Inequality:

No function of the data (representation) can be better than the data themself for decision and control (task).

However, most organisms and algorithms use complex representations that deeply alter the input. In Deep Learning we regularly torture the data to extract the results:

Three main ingredients of DNNs: Convolutions, ReLU, Max-Pool

Destroy information

Questions

Is the destruction of information necessary for learning?

Why some properties (invariance, hierarchical organization) emerge naturally in very different systems?

Why do we need to forget?

Let's assume we want to learn a classifier $p(y \mid x)$ given an input image x.

Curse of dimensionality: In general, to approximate $p(y \mid x)$ the number of samples should scale exponentially with the number of dimensions.

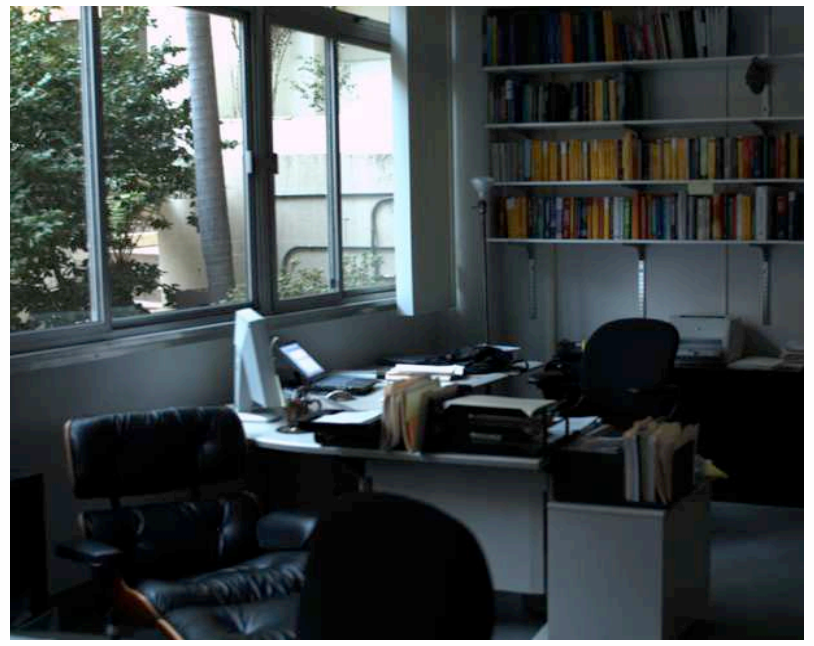
If x is a 256x256 image, this means we would need $\sim 10^{28462}$ samples.

Then, how can we learn on natural images?

1. Nuisance invariance (reduce the dimension of the **input**)
2. Compositionally (reduce the dimension of the **representation space**)
3. Complexity prior on the solution (reduce the dimension of **hypothesis space**)

Nuisance invariance

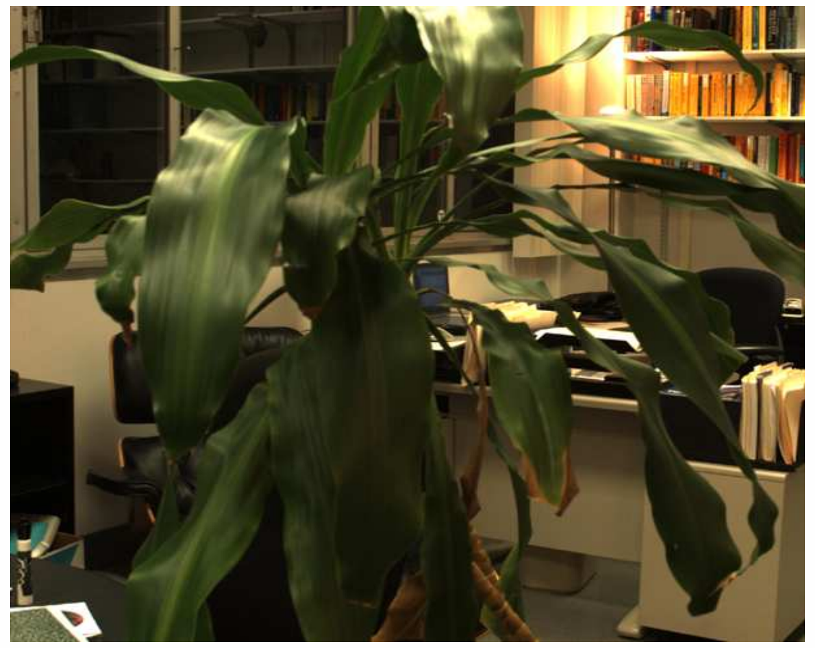
Nuisance variability



$I = h(\xi, \nu)$

Change of nuisance

$\tilde{I} = h(\xi, \tilde{\nu}), \quad \tilde{\nu} = \text{illumination}$



$\tilde{\nu} = \text{visibility}$

$\tilde{\nu} = \text{viewpoint}$

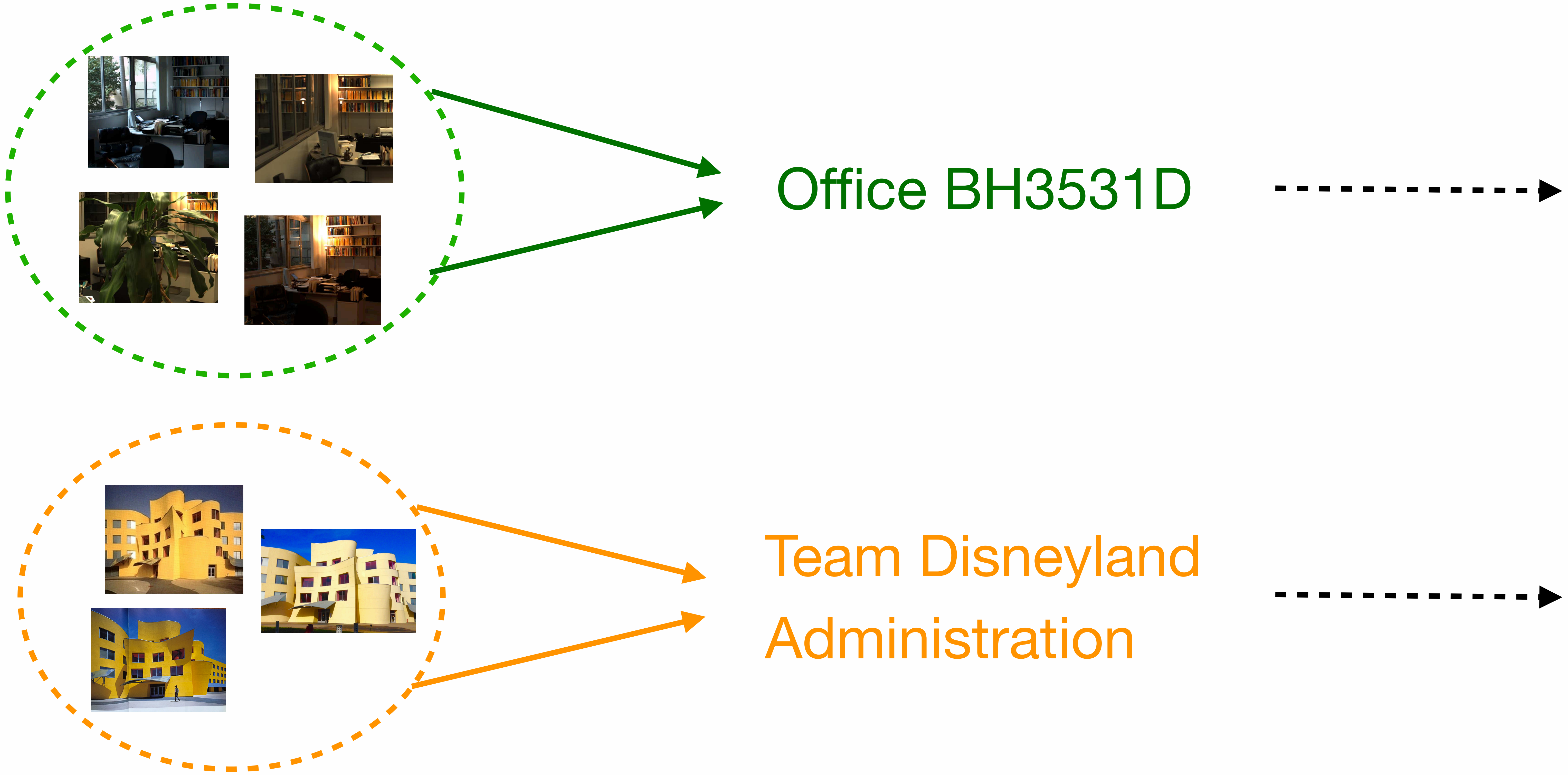
$\tilde{I} = h(\tilde{\xi}, \tilde{\nu}), \quad \tilde{\xi} \neq \xi$

Change of identity

Images from *Steps Toward a Theory of Visual Information*, S. Soatto, 2011

How to use nuisance variability

A good representation should collapse images differing only for nuisance variability.



Quotienting with respect to nuisances reduces the dimensionality of the space of images, and simplifies learning the successive parts of the pipeline.

Group nuisances

Examples: Translations, rotations, change of scale/contrast, small diffeomorphisms

Given a group *G* acting on the space of data *X*, we say that a representation f(x) is invariant to *G* if:

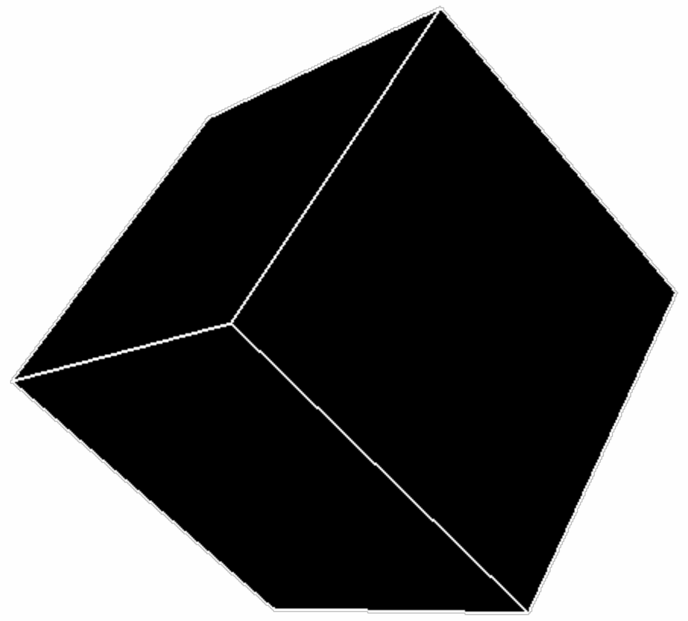
$$f(x) = f(g \circ x) \qquad \text{for all} \quad g \in G, x \in X$$

A representation is *maximal invariant* if all other invariant representations are a function of it.

Well understood for translation and scale. The solution inspired and justifies the use of convolutions and max-pooling (next class).

Problems with group nuisances

1. Rapidly becomes difficult for more complex groups
2. Groups acting on 3D objects do not act as groups on the image



3. Not all nuisances are groups (*e.g.*, occlusions)

More general nuisances

Idea: A nuisance as everything that does not carry information about the task.

Introduce the Information Bottleneck Lagrangian:

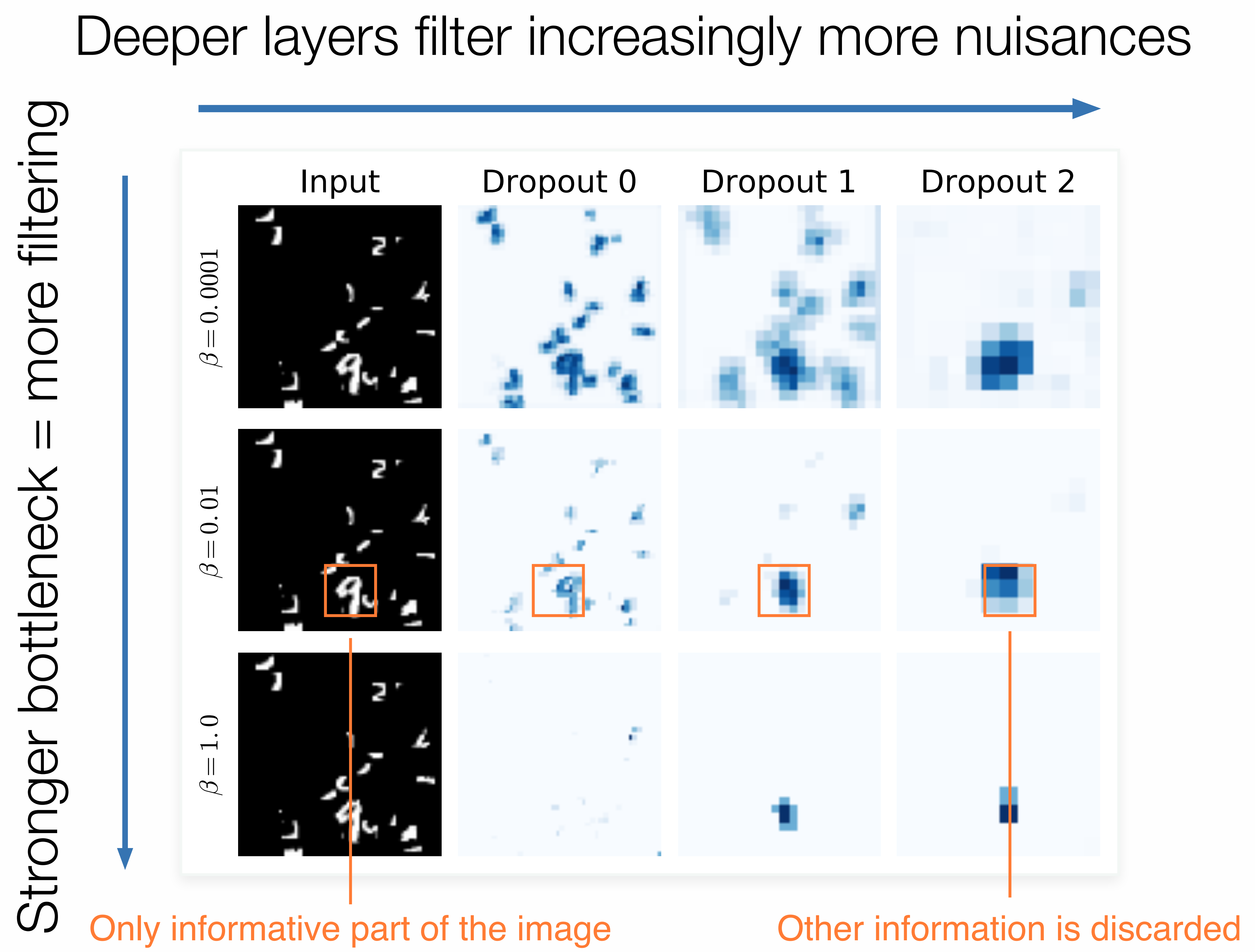
$$\min_f \underbrace{I(f(x); x)}_{\text{Total information}} - \lambda \underbrace{I(f(x); \text{task})}_{\text{Information the representation has about the task}}$$

where *I(x; y)* is the mutual information. The solution to the Lagrangian (for $\lambda \to +\infty$) is a maximally invariant representation for all nuisances.

We can thus rephrase the problem of nuisance invariance as a much simpler variational optimization problem.

Learning invariant representations

We can approximatively optimize the variational objective using DNNs (Tuesday).



A. and Soatto, "Information Dropout: Learning Optimal Representations Through Noisy Computation", PAMI 2018 (arXiv 2016)

Compositional representations

Compositional representations

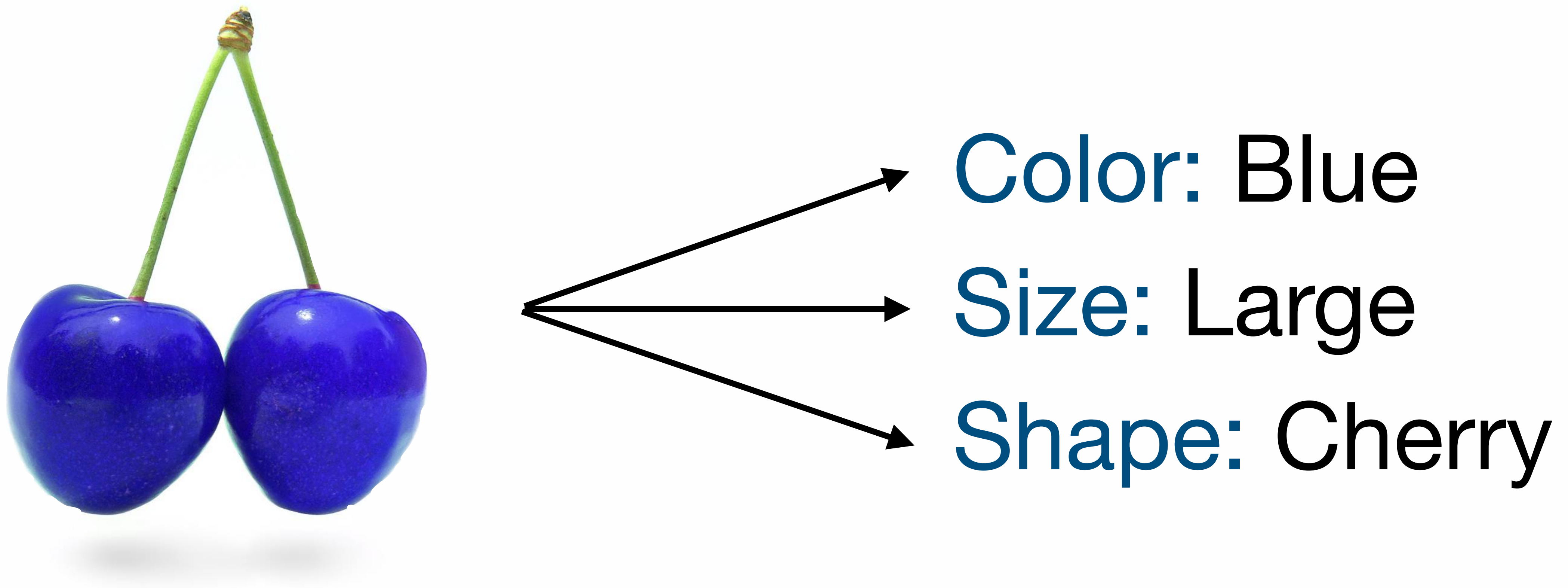
Humans can easily solve task by combining concepts:

"Find a blue large cherry"

We can easily solve this task, even if we have never seen a blue cherry before.

Compositionally requires disentanglement

To learn a good compositional representation, we first need to learn to decompose the image in reusable semantic factors:



This mitigates the curse of dimensionality: each factor is easy to learn, but combined they yield exponentially many objects.

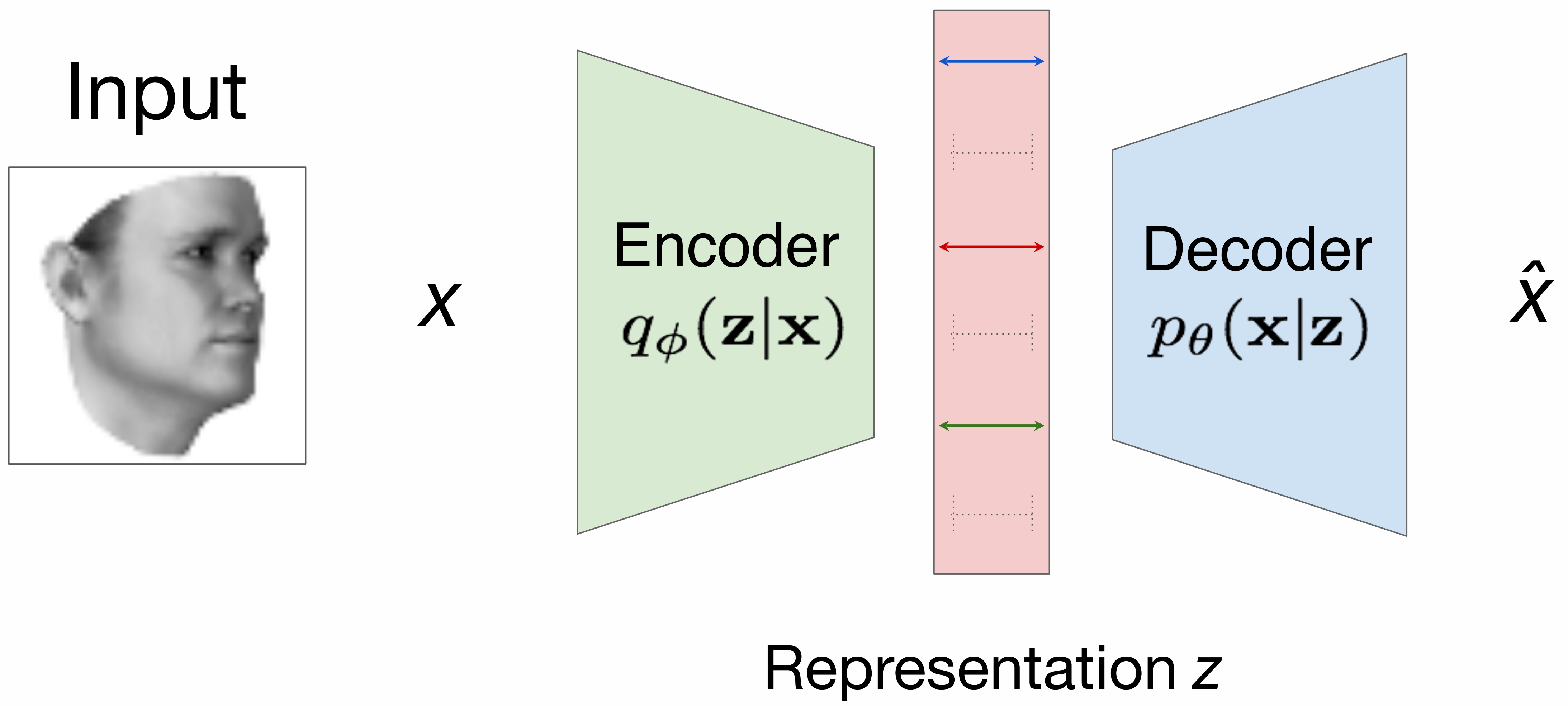
Factors of variation can be learnt in succession in a life-long learning setting and used in the future for one-shot or zero-shot learning.

Problem. But what are "semantic factors of variation"?

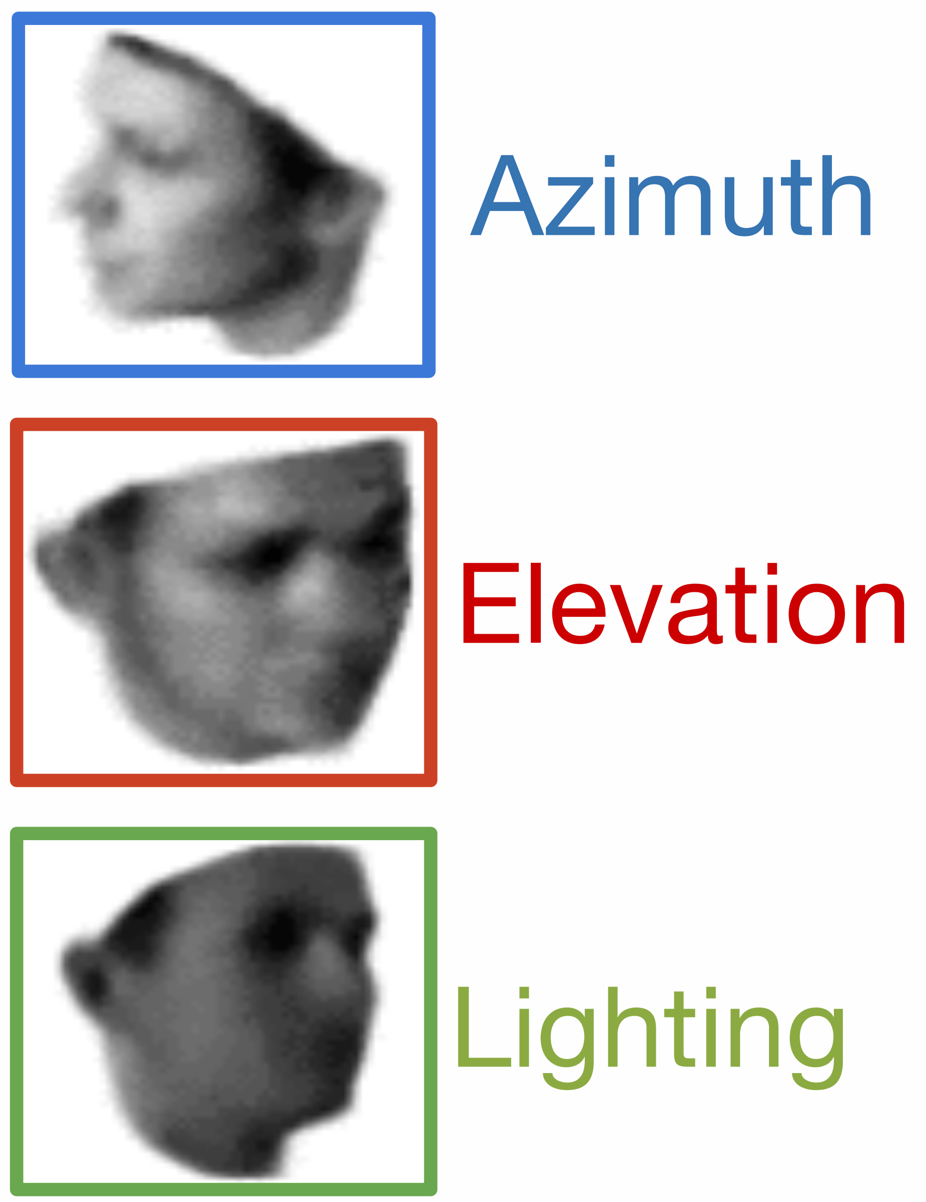
Learning disentangled representations

Possible answer through the Minimum Description Length principle:

$$\mathcal{L}_{\text{MDL}}(\phi,\theta)=\underbrace{\mathbb{E}_{\mathbf{z}^s\sim q_\phi(\cdot|\mathbf{x}^s)}[-\log p_\theta(\mathbf{x}\mid\mathbf{z}^s,s)]}_{\text{Reconstruction error}}+\gamma|\underbrace{\mathbb{KL}(q_\phi(\mathbf{z}^s|\mathbf{x}^s)||p(\mathbf{z}))}_{\text{Representation capacity}}-\underbrace{C}_{\text{Target}}|^2$$



Latent traversal



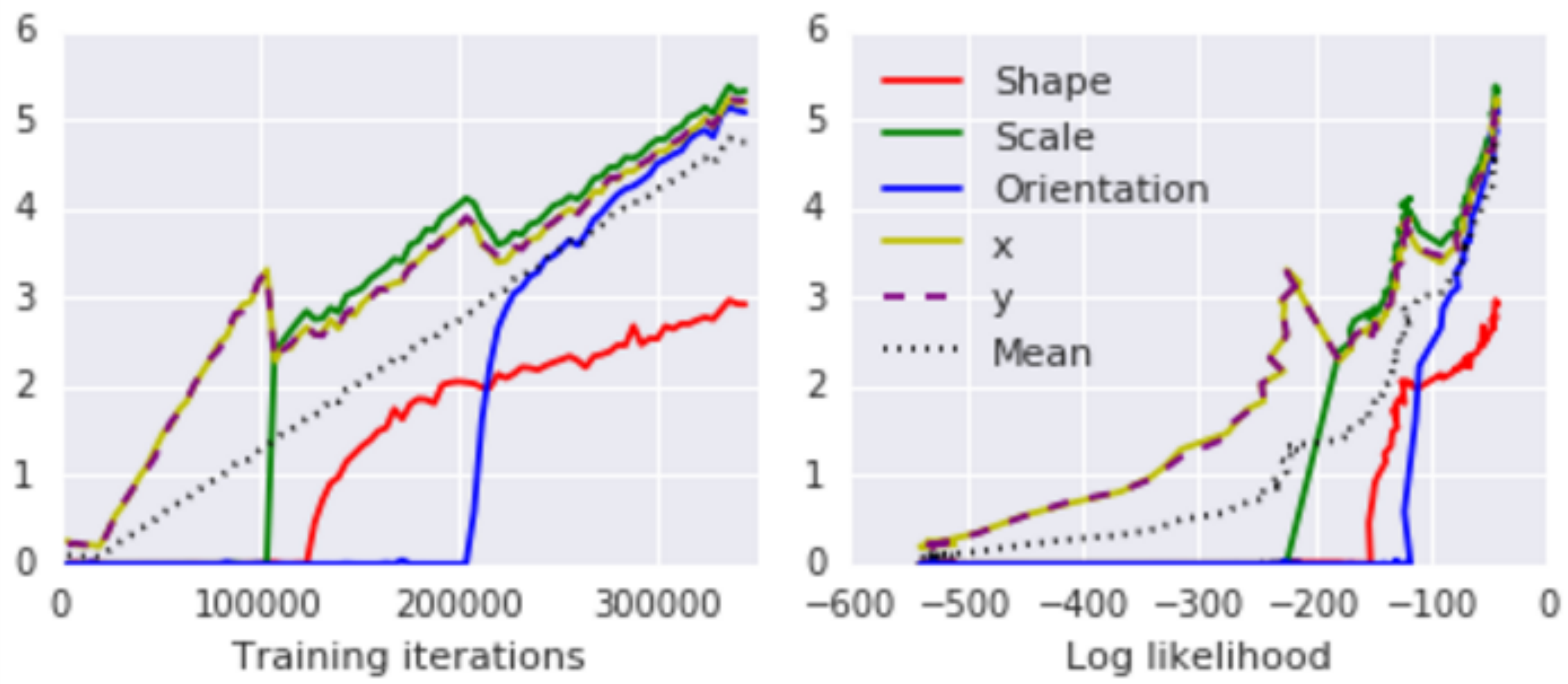
Higgins et al., β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, 2017
Burgess et al., Understanding Disentangling in beta-VAE” 2017

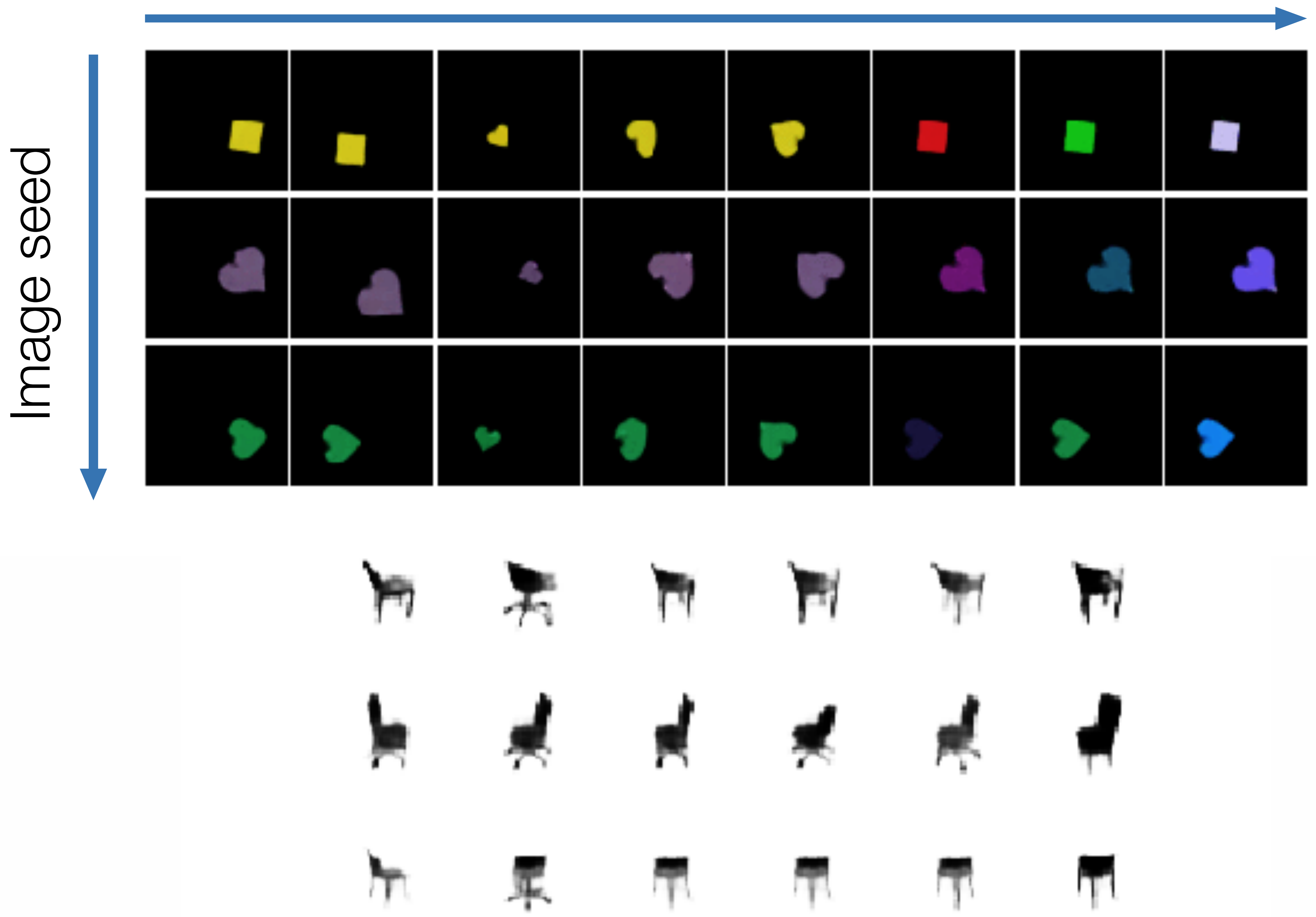
Pictures courtesy of Higgins et al., Burgess et al.

Learning disentangled representations

Possible answer through the Minimum Description Length principle:

Phase transitions during training



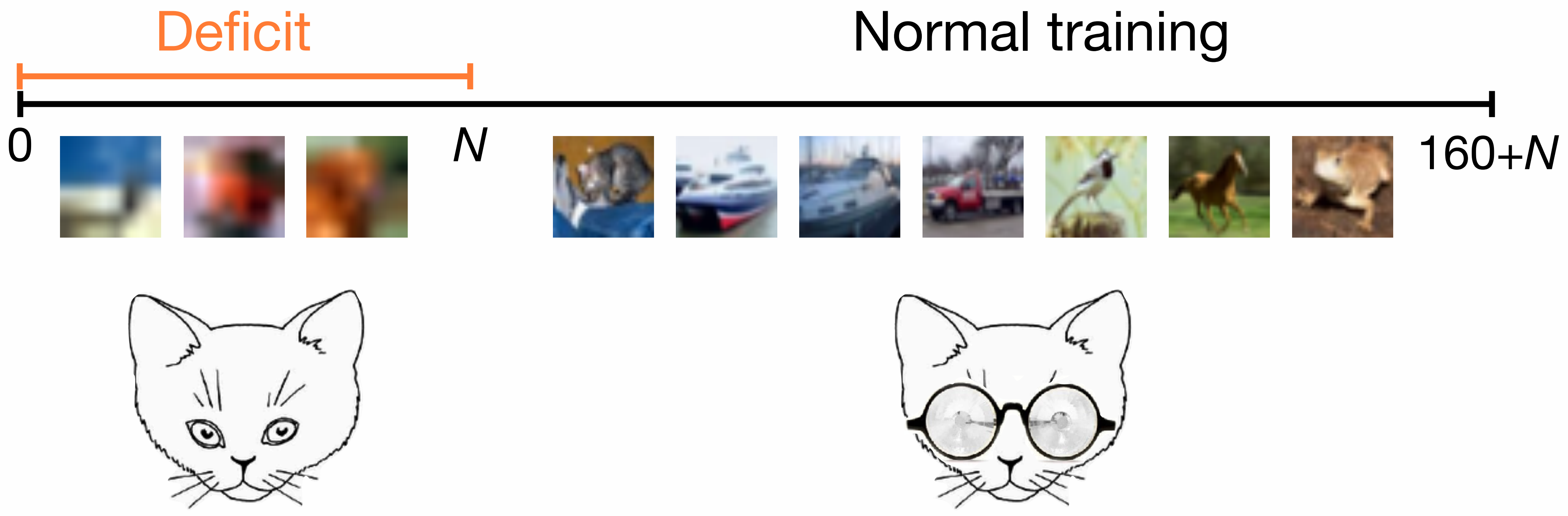


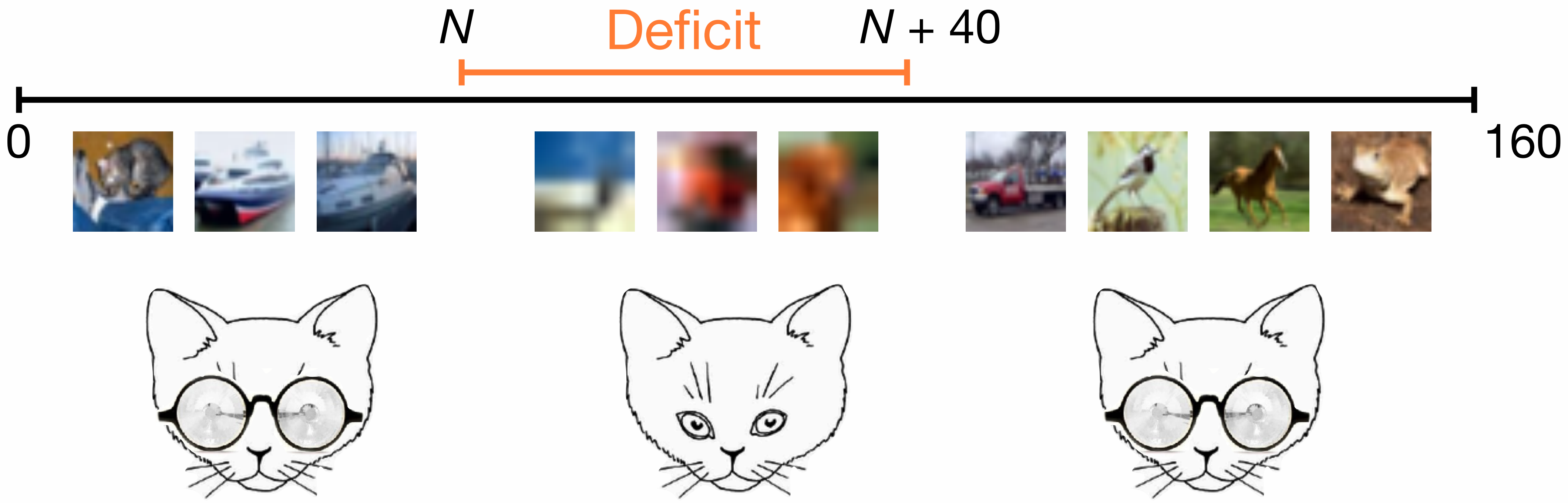
Higgins et al., *β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework*, 2017
Burgess et al., *Understanding Disentangling in beta-VAE"* 2017

Pictures courtesy of Higgins et al., Burgess et al

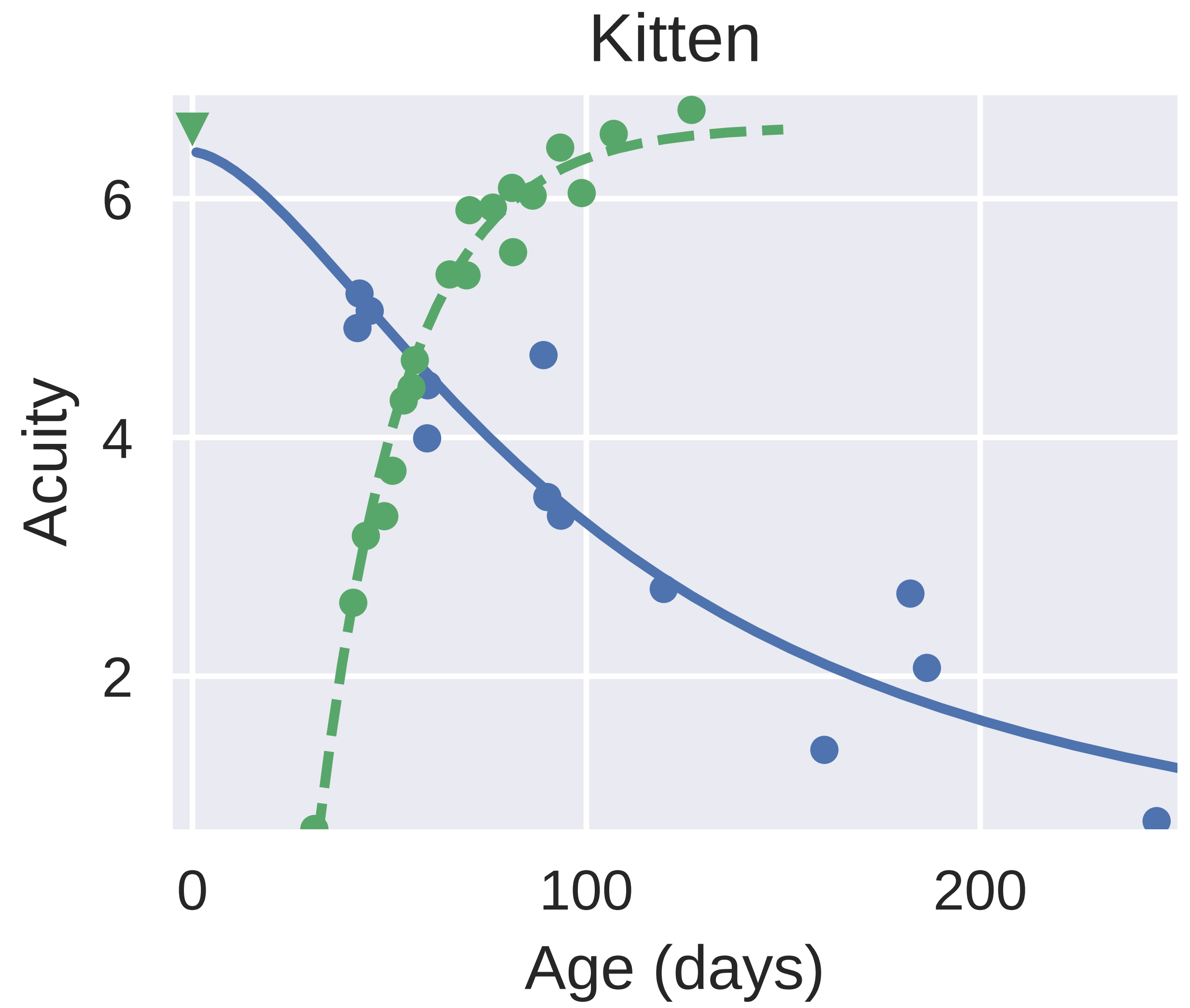
Information in the Weights and Dynamics of Learning

How, and *when*, do we learn good representations?





Hubel and Wiesel © Harvard University



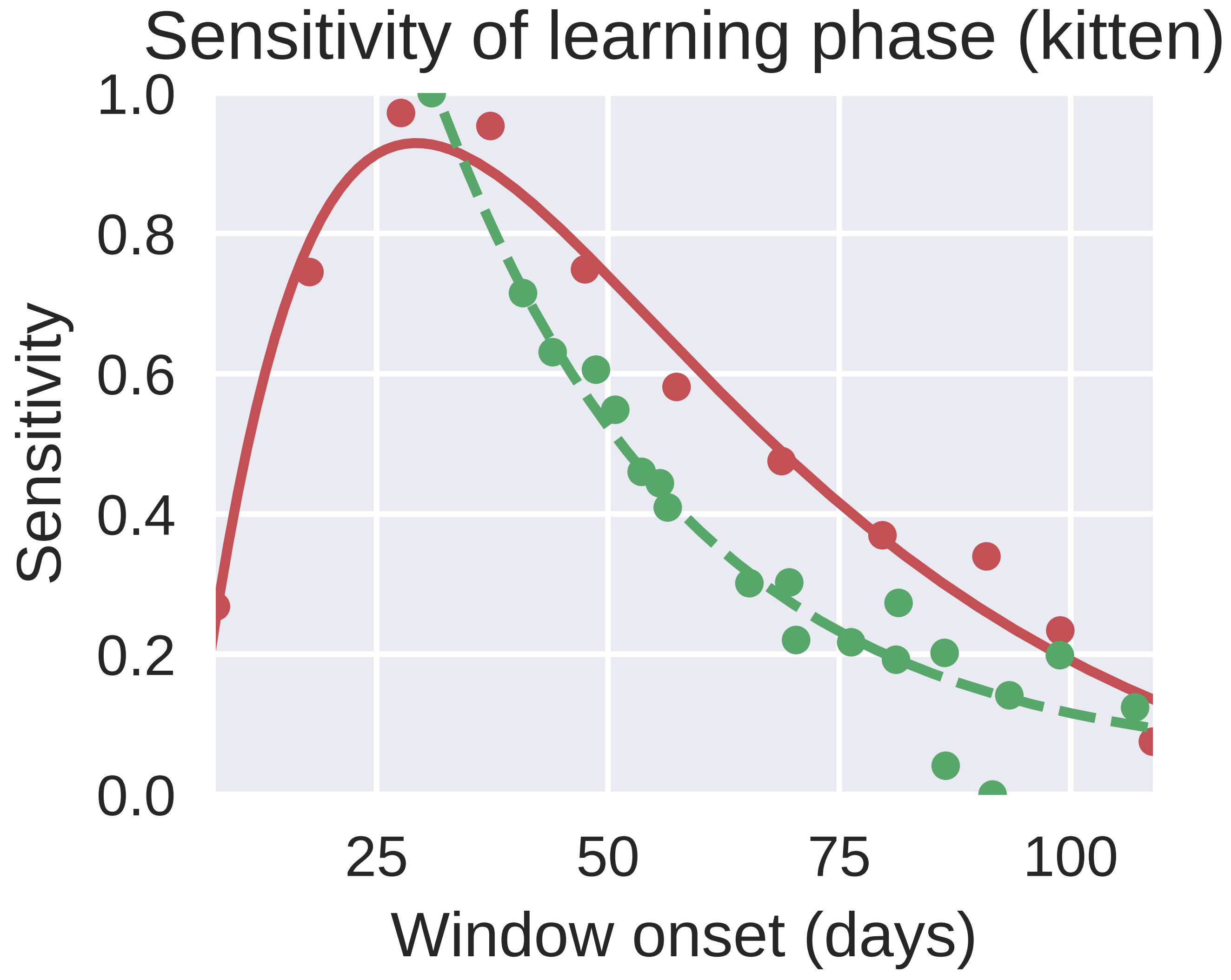
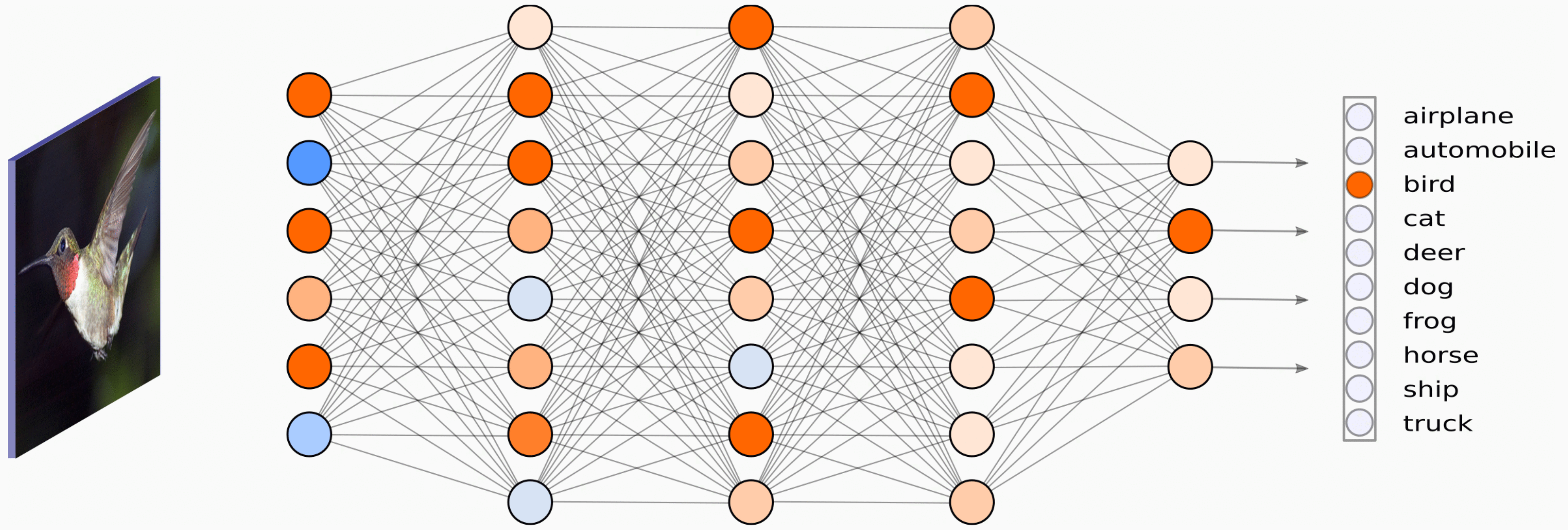
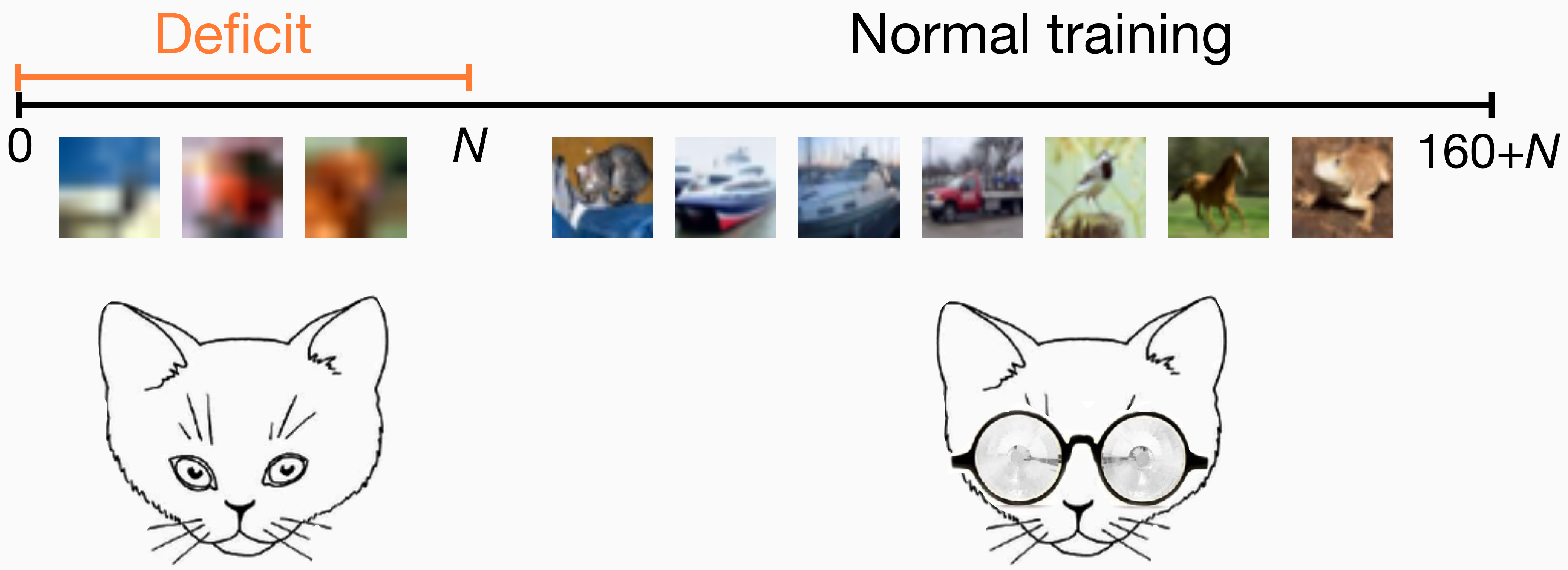
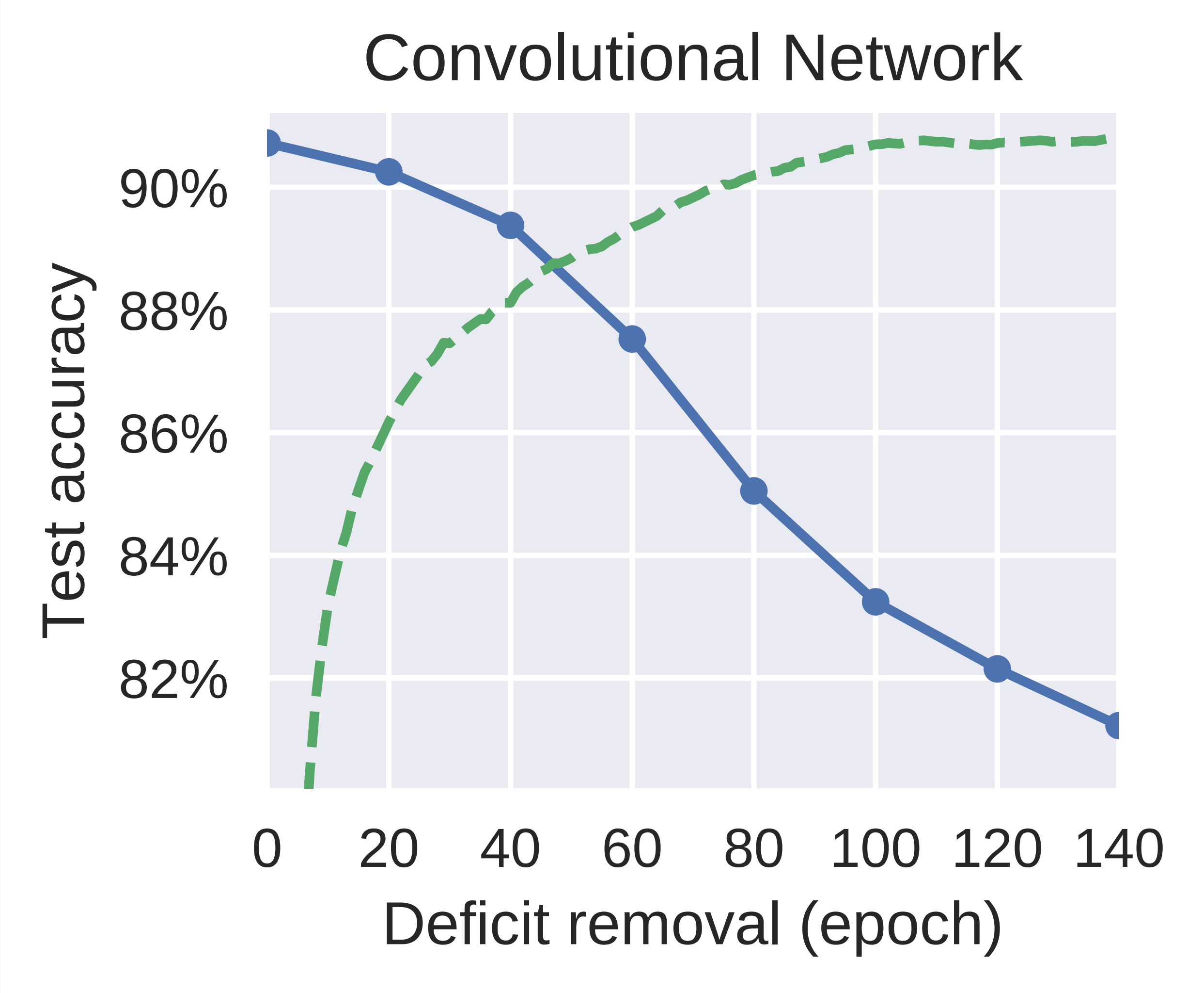


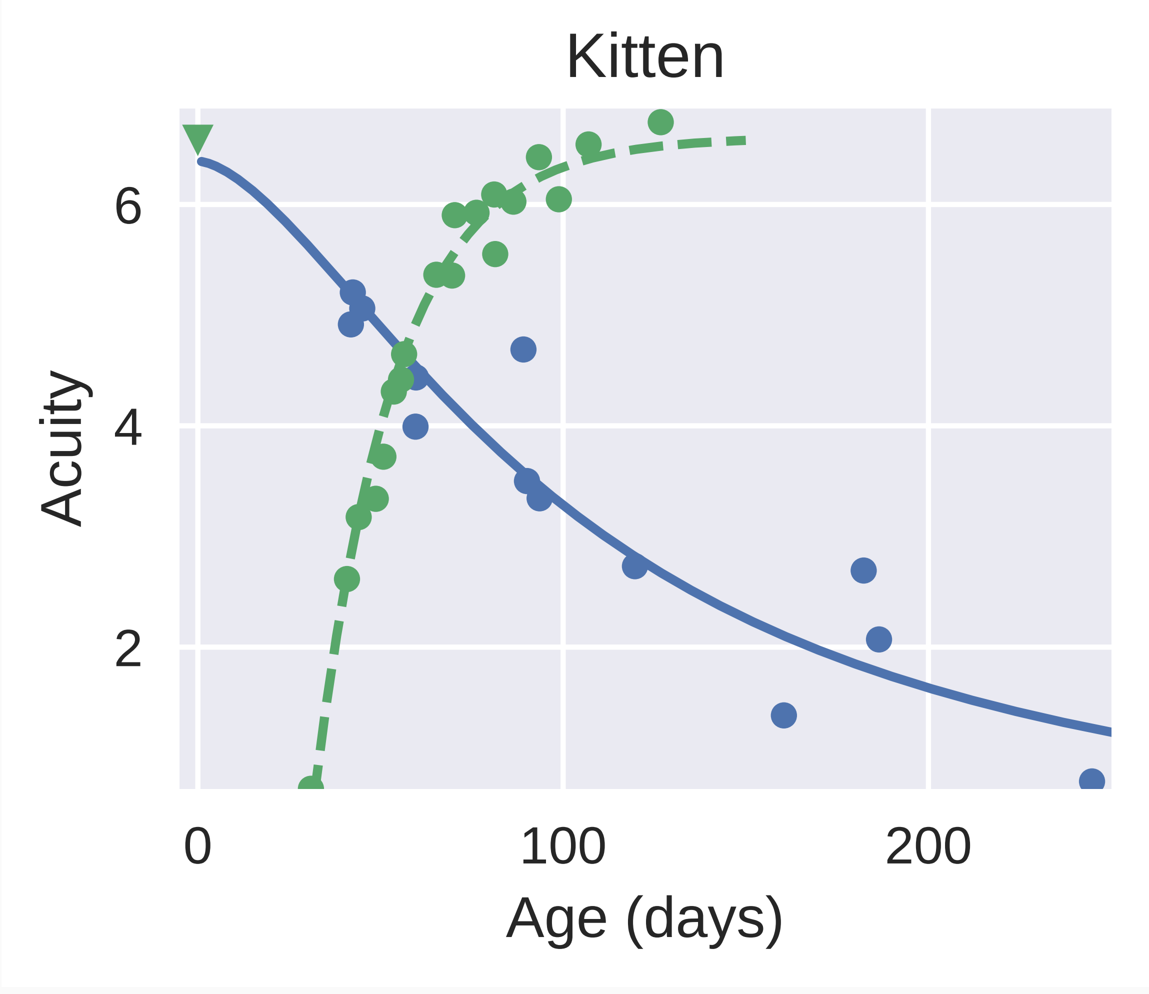
Image from Cnops et al., 2008

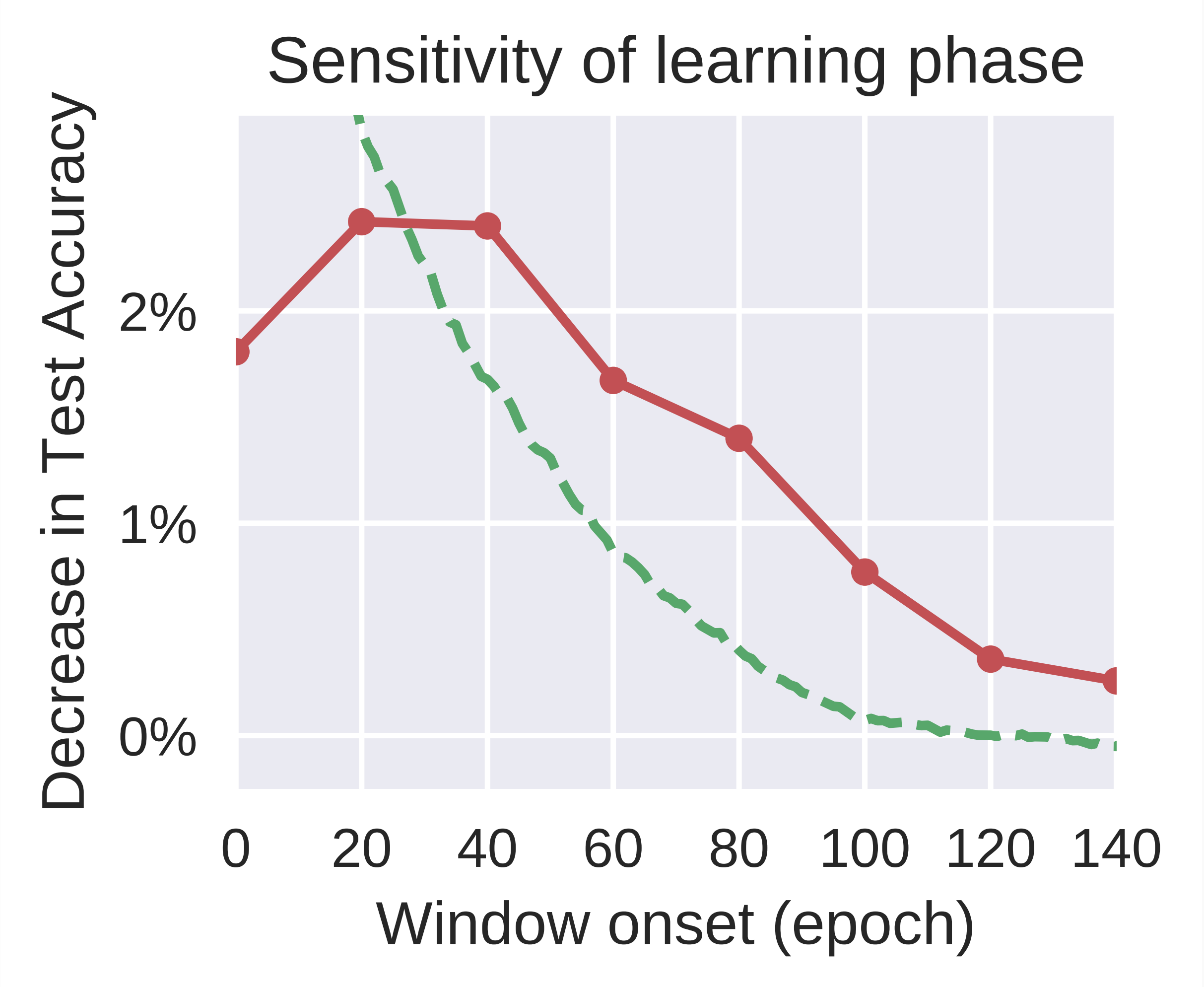
Critical Learning Periods in Deep Networks

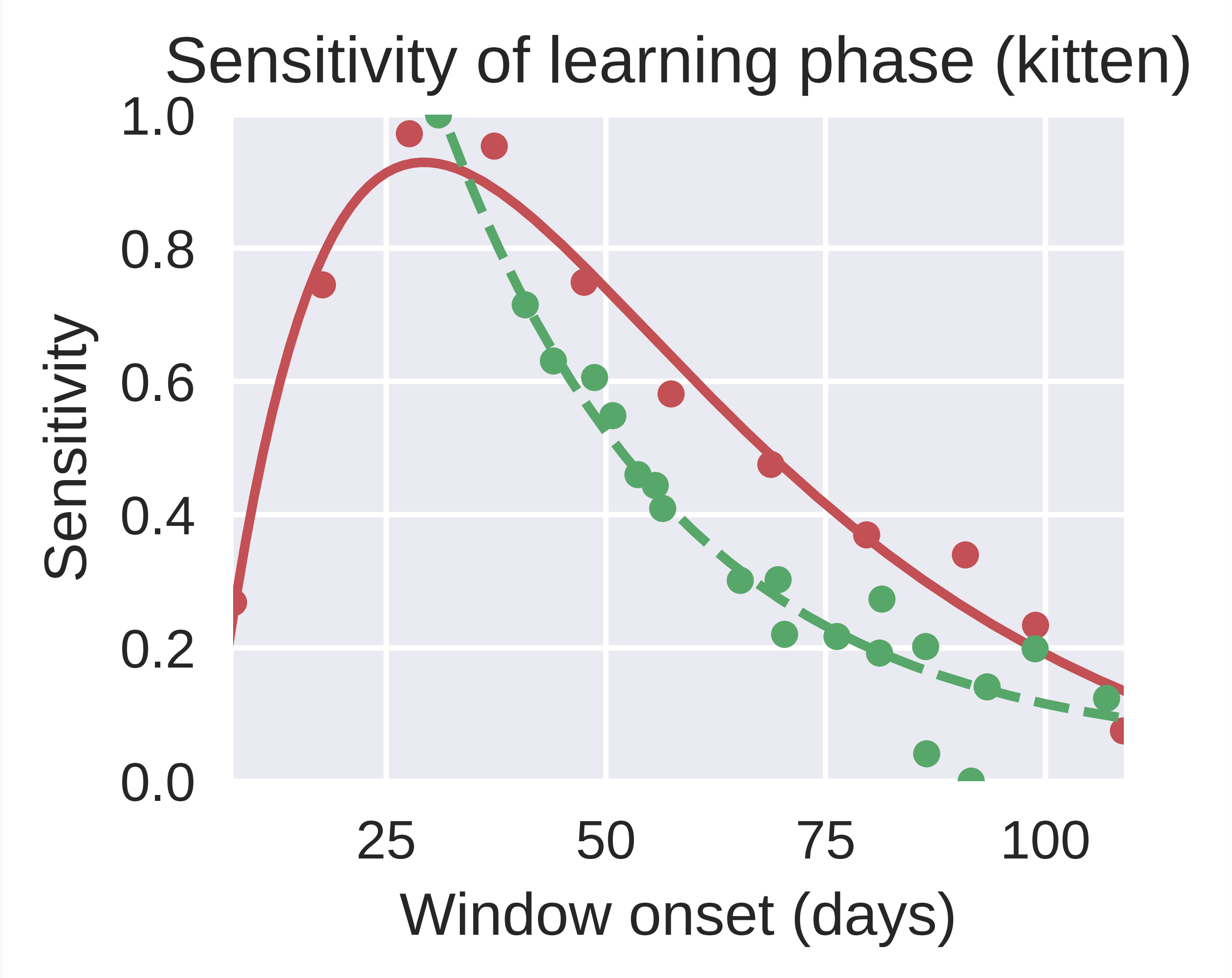




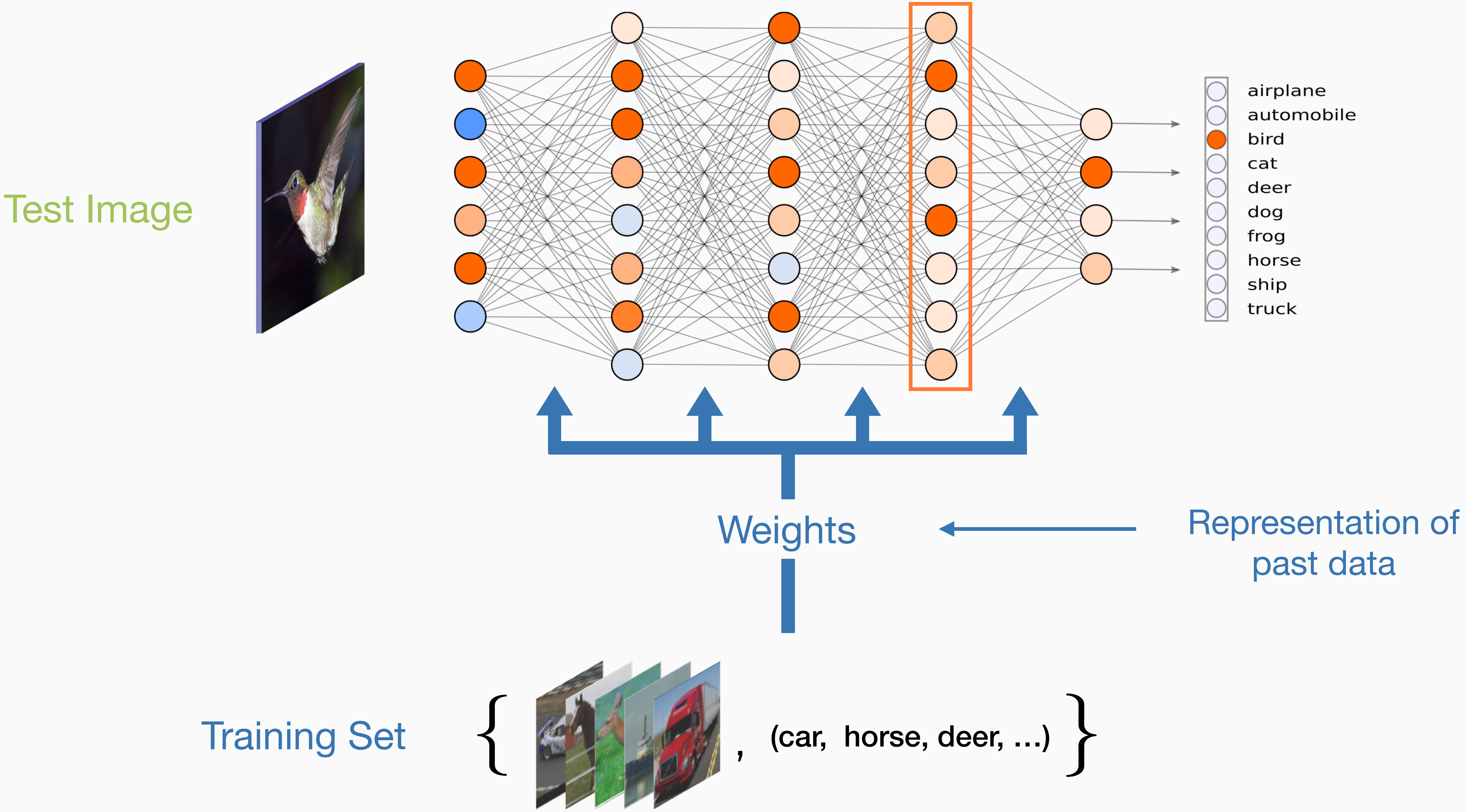








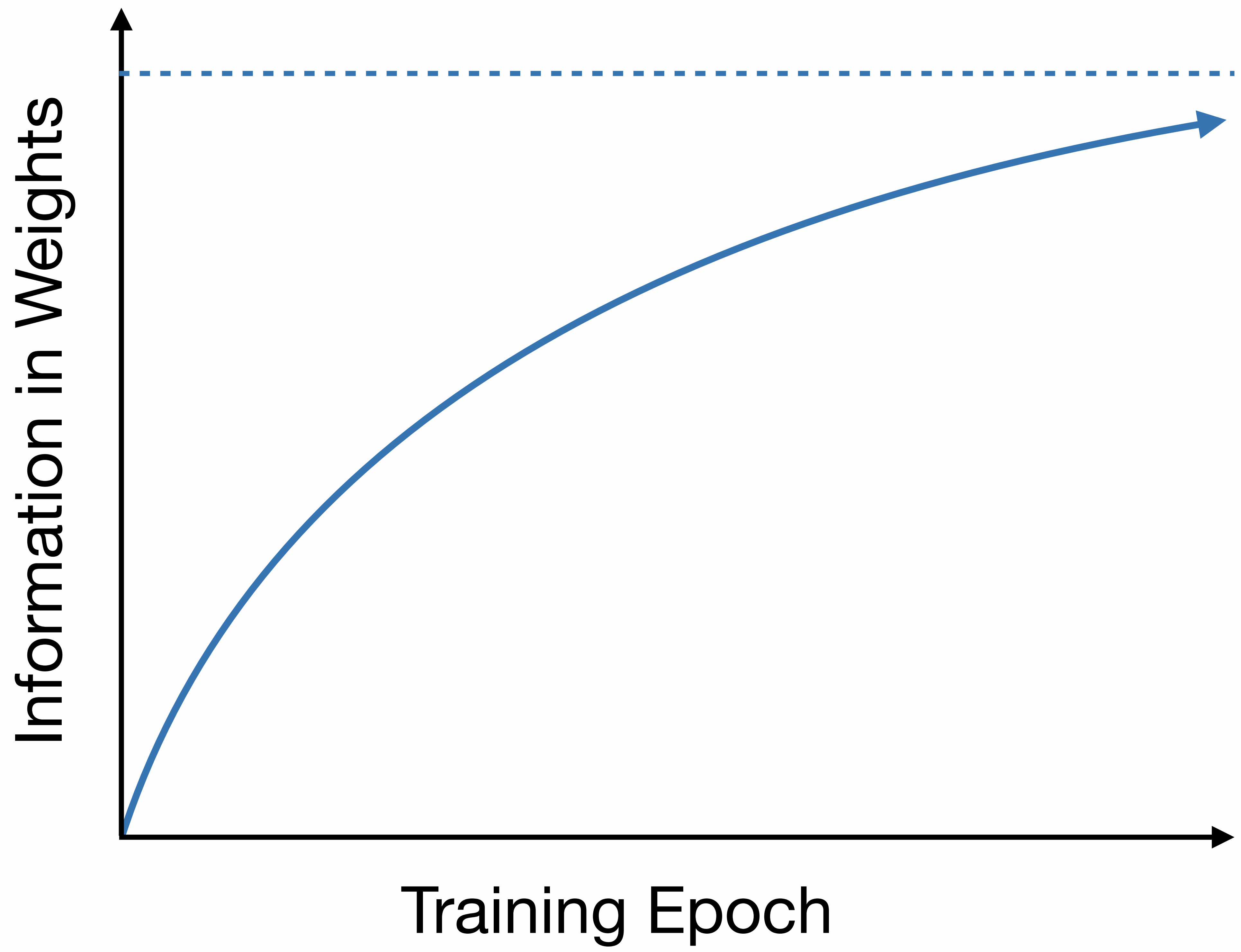
A., Rovere, Soatto, Critical Learning Periods in Deep Networks, 2018


Test Image
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Weights
Representation of
past data
Training Set
{ , (car, horse, deer, ...) }

Information in Weights during training

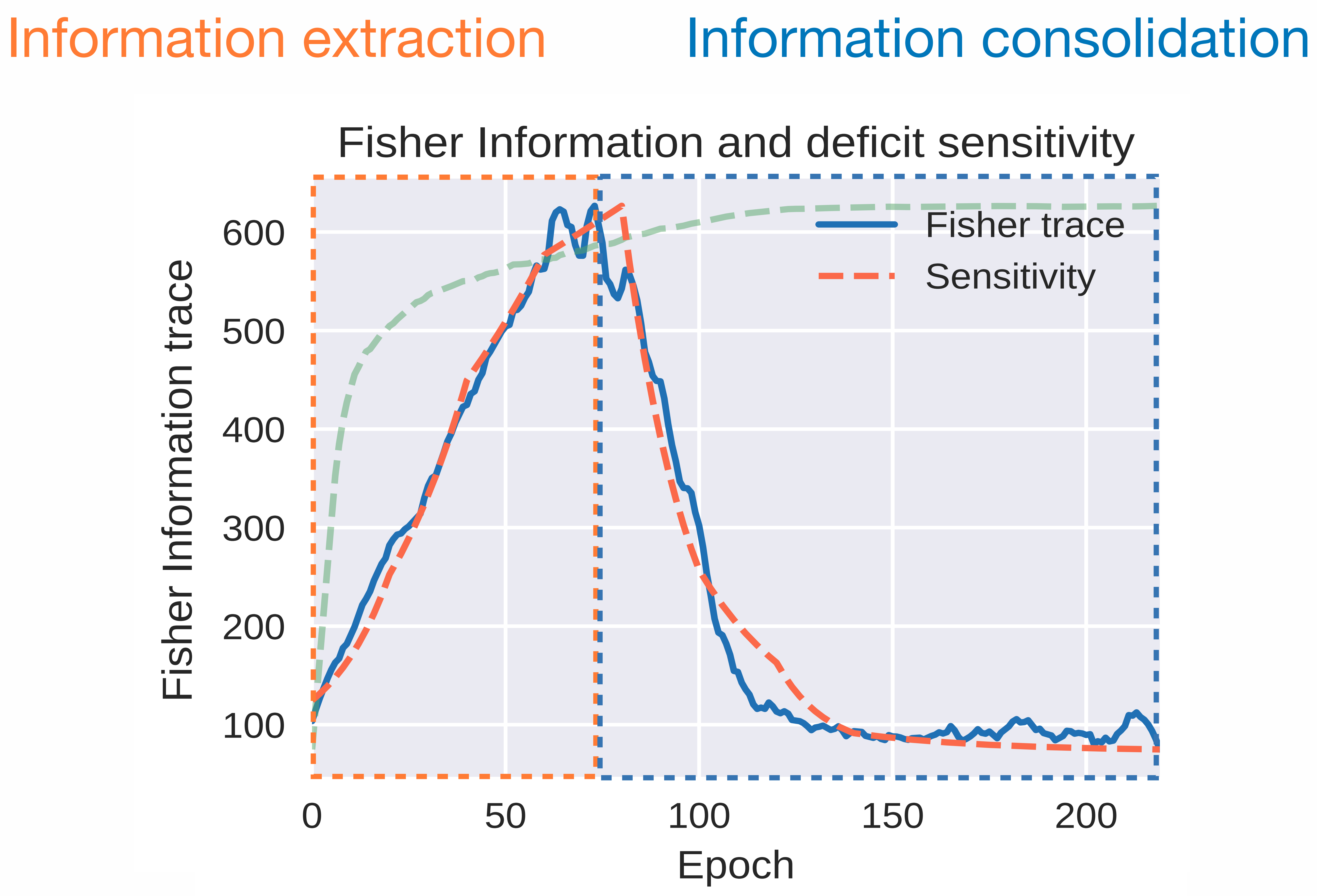
What should we expect from the information in the weights during training?

Maybe something like this?



Information in Weights during training

Not quite so.

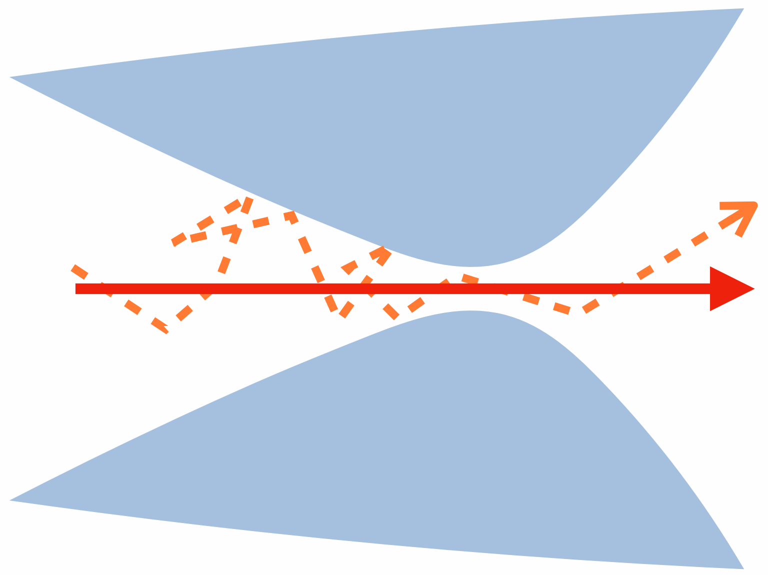


A., Rovere, Soatto, Critical Learning Periods in Deep Networks, 2018

Emergence of invariant and disentangled representations

Theorem 1 (informal). **Stochastic gradient descent** biases the optimization process toward recovering low information solutions.

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



Theorem 2 (informal). In DNNs, low-information classifier have invariant and disentangled representations.

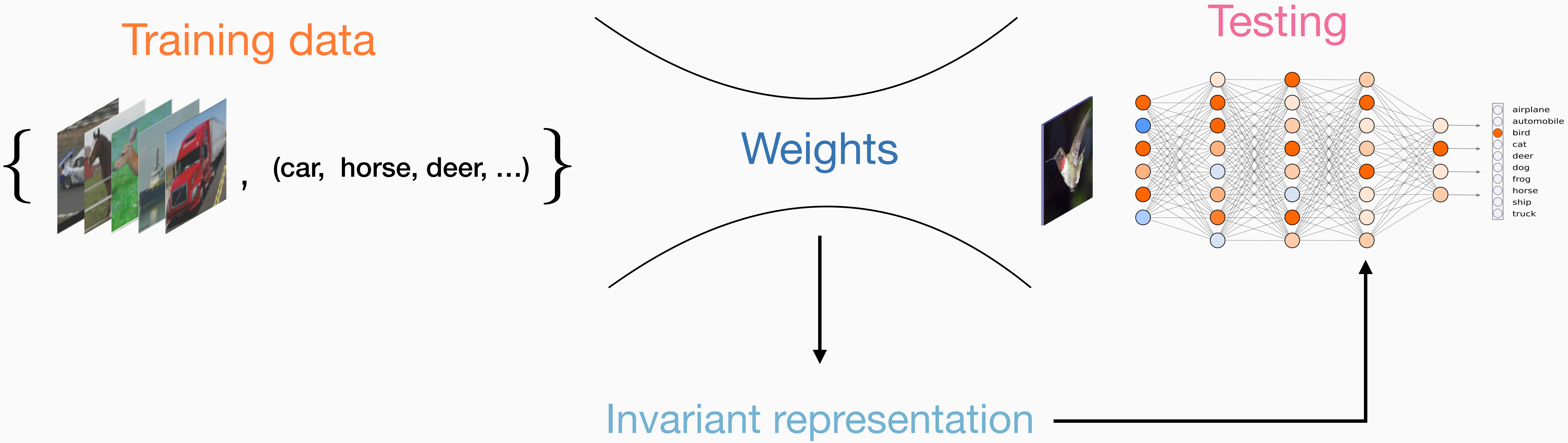
Minimality of activations

$$\boxed{I_{\text{eff}}(x;z)} \approx H(x) - \log\left(\frac{(2\pi e)^k}{|\nabla_x f_w(x)^t \; J_f^t \, \boxed{F(w)} \, J_f \; \nabla_x f_w(x)|}\right)$$

Fisher Information of Weights

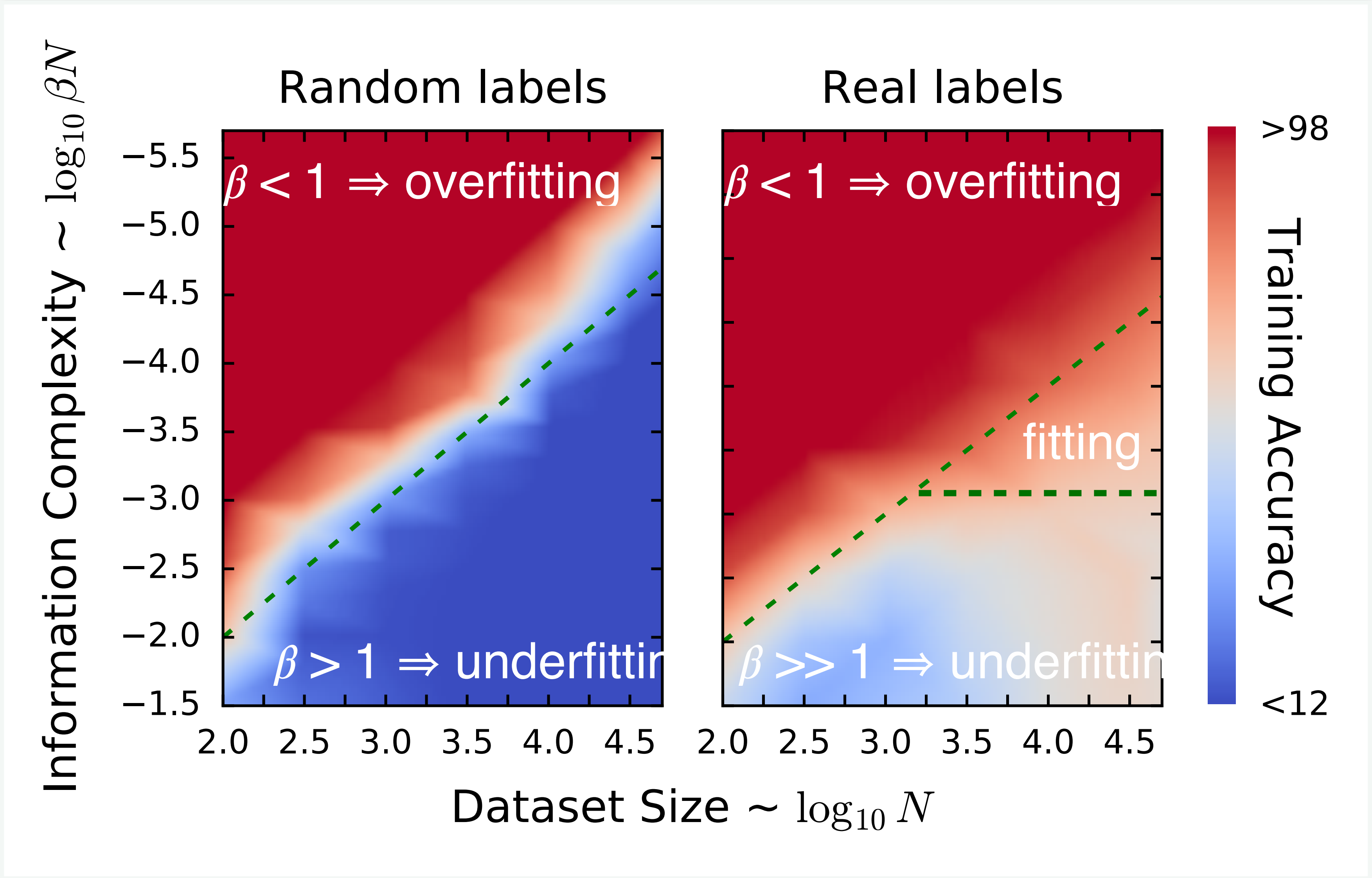
Corollary (Theorem 1 + 2). DNNs are biased toward learning invariant and disentangled representations.

Compression of the **weights** biases toward invariant and disentangled **representations.**



Some consequences

Phase transitions for learning.

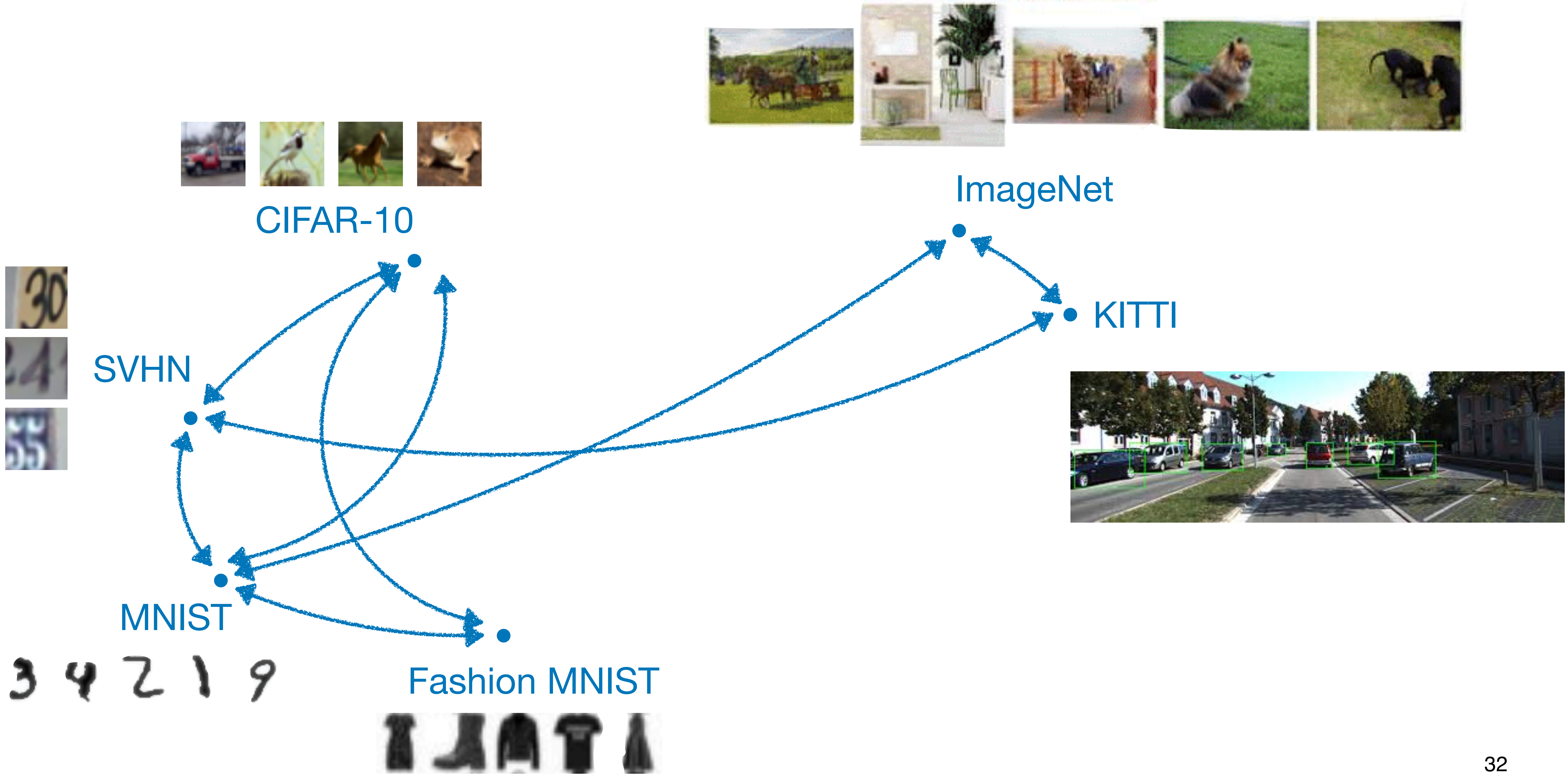


Error bounds for DNN.

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

Distance between tasks

Can we put a distance on the space of tasks?



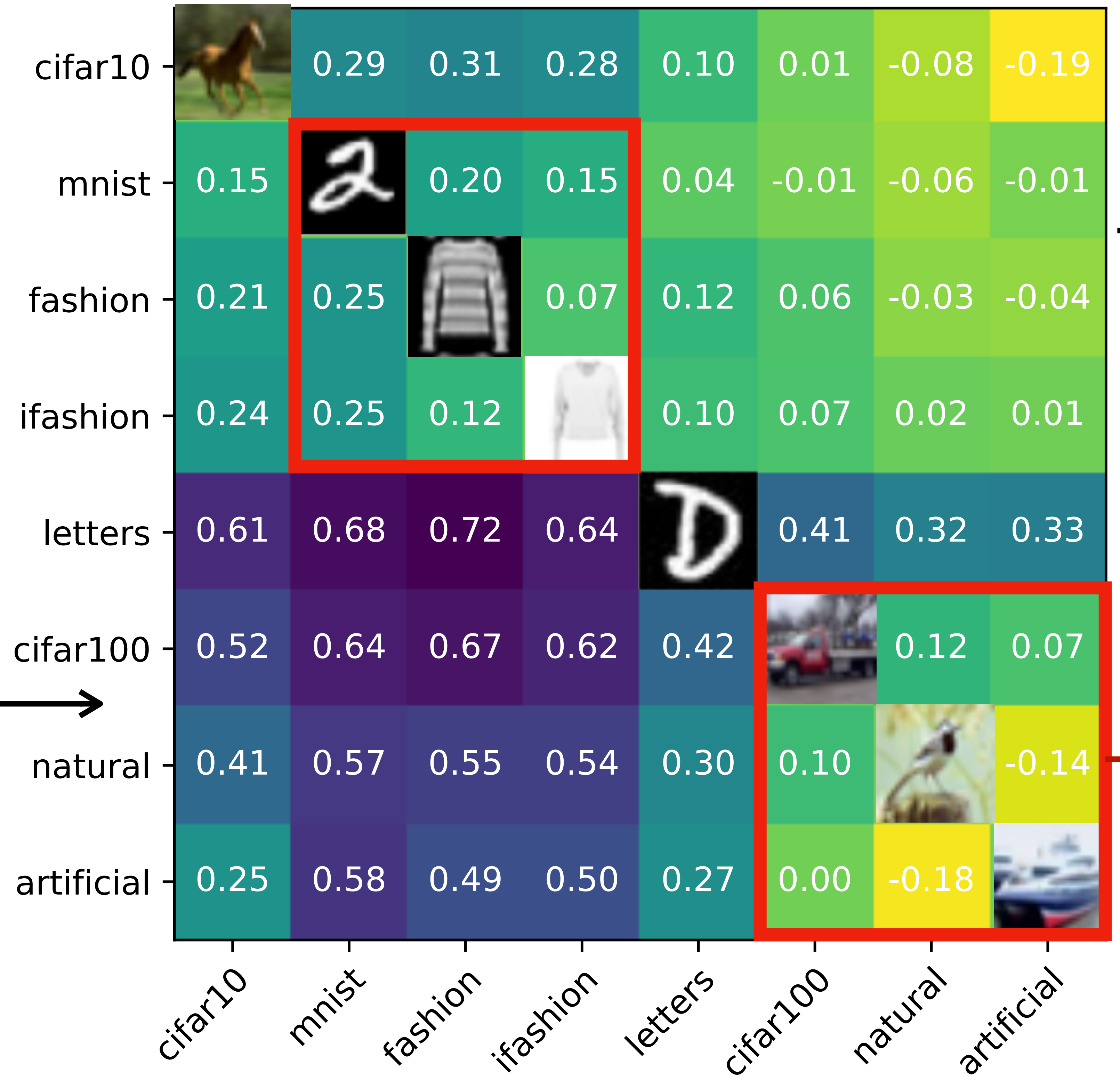
A Topology on the Space of Tasks

$$d(\mathscr{D}_1 \to \mathscr{D}_2) = I(\mathscr{D}_1\mathscr{D}_2; w) - I(\mathscr{D}_1; w)$$

Information in the joint datasets

Information in one of the two datasets

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

Lecture 1

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 family of function, and look for a good set of parameters w, by minimizing a loss function $\mathscr{L}_{\mathscr{D}}(w)$.

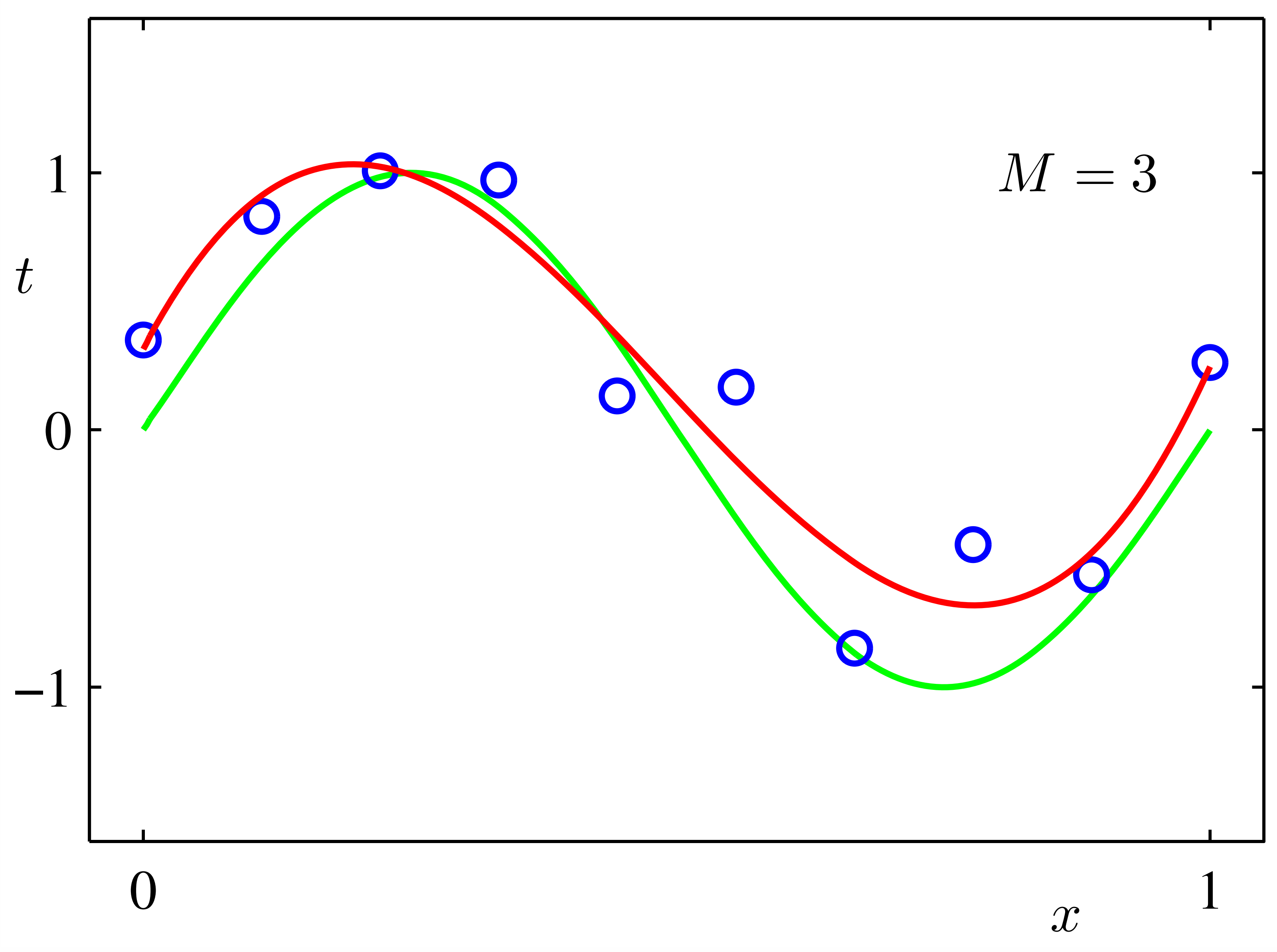
Example (Polynomial curve fitting):

Family of functions: polynomials of degree M

$$f_w(x) = w_0 + w_1 x + \cdots + w_M x^M$$

L_2 reconstruction loss:

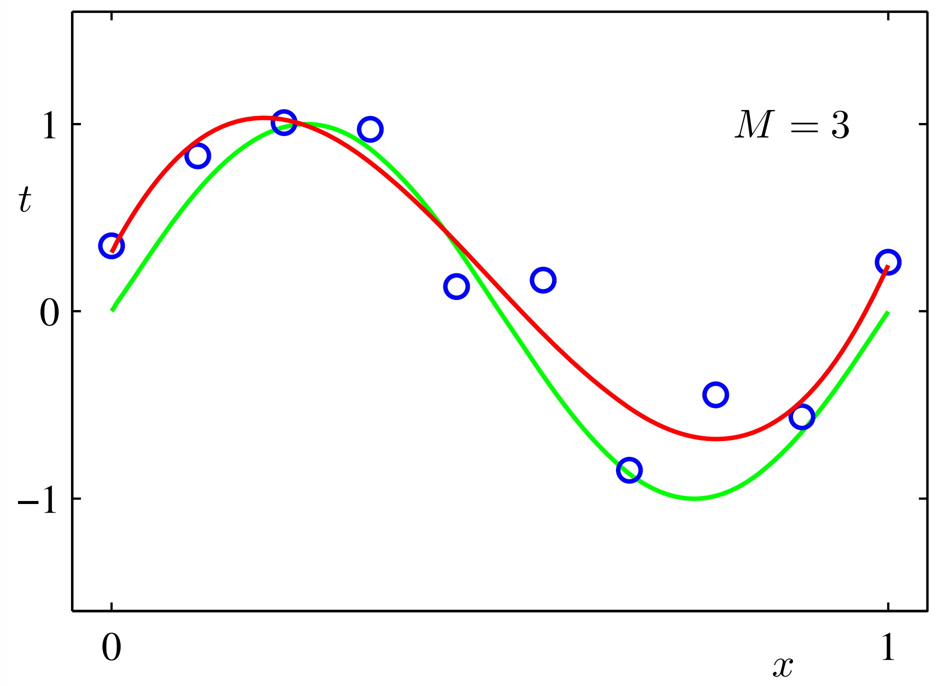
$$\mathscr{L}_{\mathscr{D}}(w) = \frac{1}{2} \sum_{i=1}^{N} \left(f_w(x_i) - y_i\right)^2$$



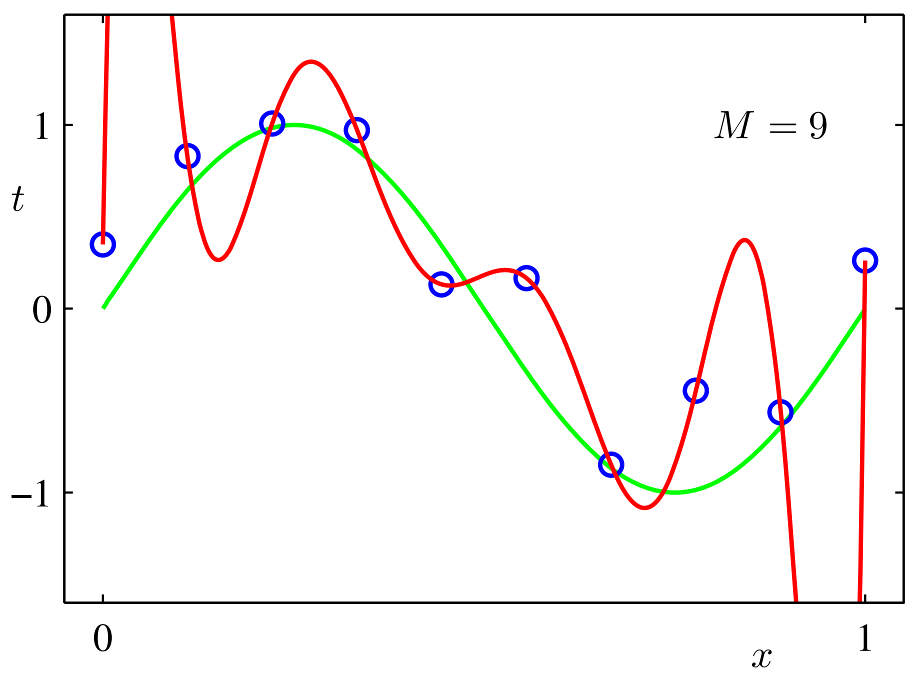
Picture from Christofer M. Bishop, *Pattern Recognition and Machine Learning*

Overfitting and regularization

If the model is too flexible expressive, overfitting can happen (bias variance trade-off).



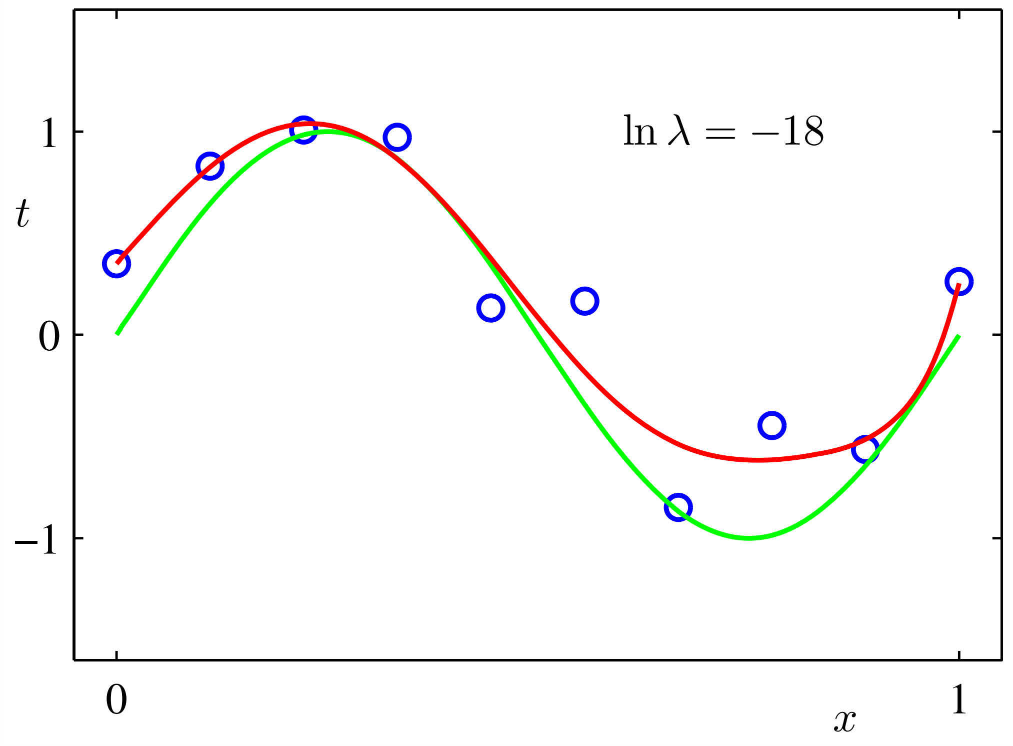
Increase degree M



One way to reduce overfitting is to constrain the parameters.

$$\mathscr{L}_{\mathscr{D}}(w) = \frac{1}{2}\sum_{i=1}^{N}\big(f_w(x_i) - y_i\big)^2 + \frac{\lambda}{2}\|w\|^2$$

Regularization term



Pictures from Christofer M. Bishop, *Pattern Recognition and Machine Learning*

How do we find the parameters: SGD

We want to find the parameters *w* that minimize the loss:

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

The gradient can be computed as:

$$\nabla_w \mathscr{L}_{\mathscr{D}}(w) = \mathbb{E}_{x_i, y_i \sim \mathscr{D}}[\nabla_w \ell_w(x_i, y_i)] \quad \Rightarrow$$

the sample gradient

$$\nabla_w \ell_w(x_i, y_i), \quad x_i, y_i \sim \mathscr{D}$$

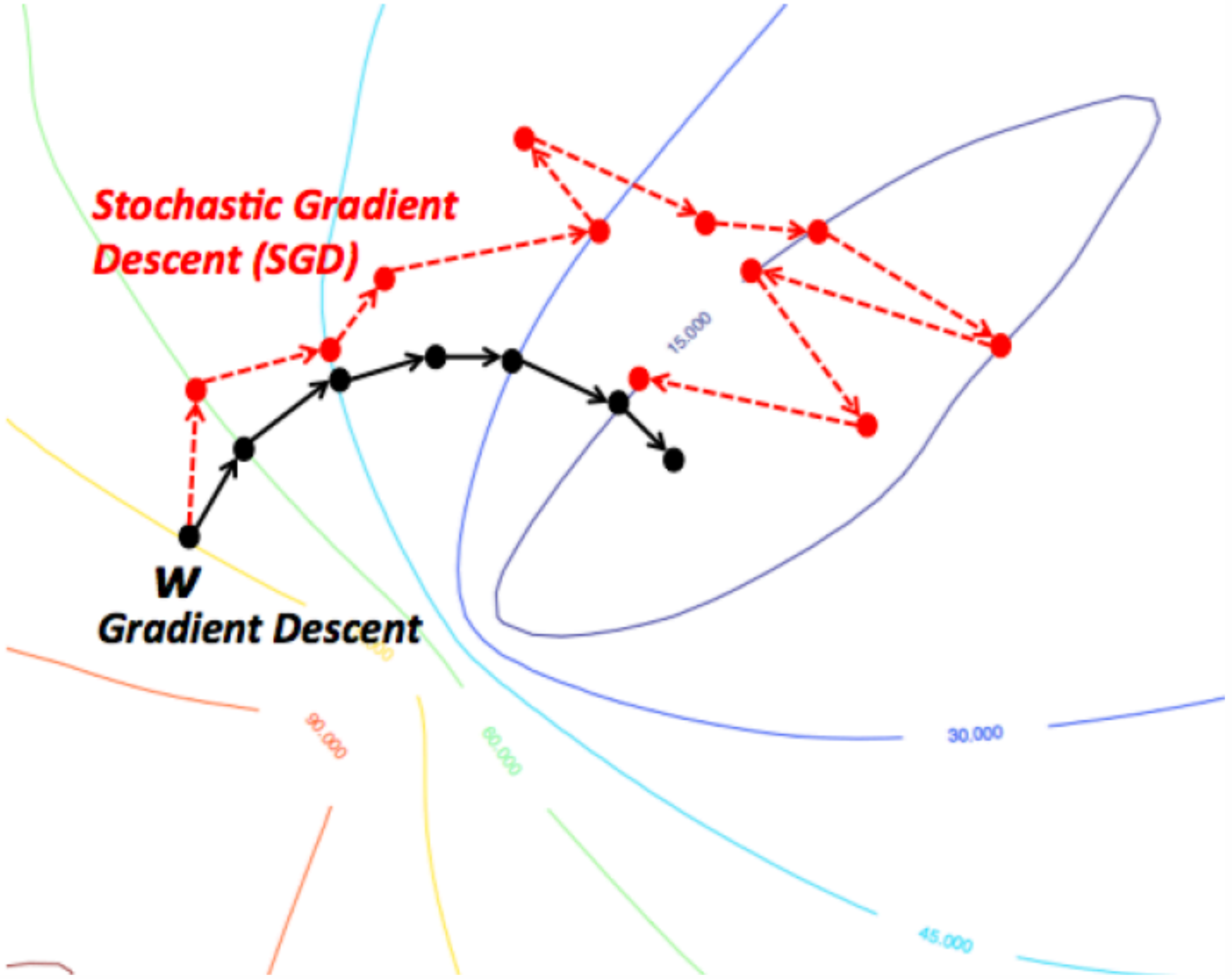
is an unbiased estimator of the real gradient

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 $w' \leftarrow w - \eta\, g_i$

SGD as gradient descent with noisy dynamics

$$\nabla_w \ell_w(x_i, y_i) = \nabla_w L(w) + \text{noise}$$



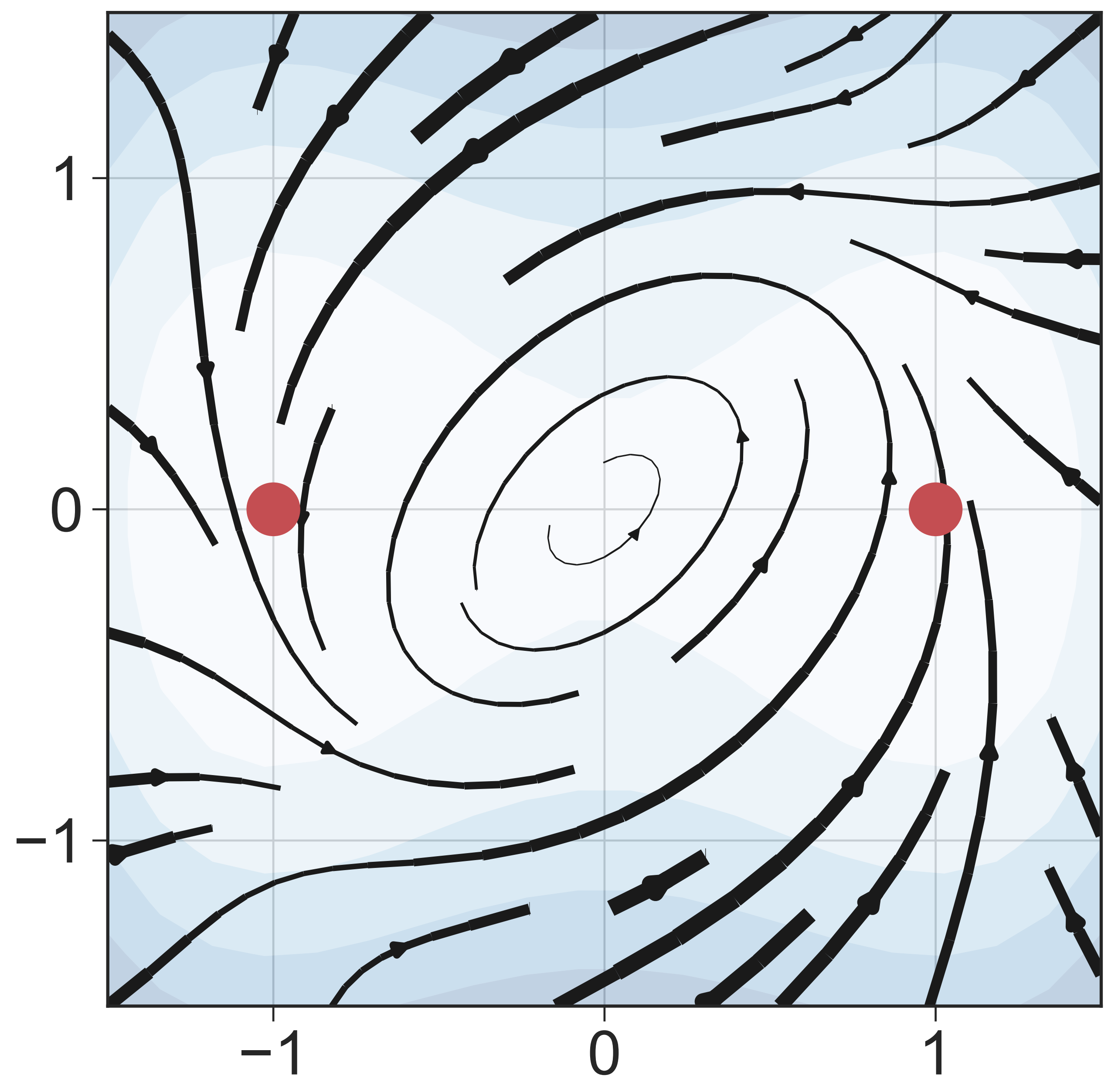
(Robbins and Morro, 1951). In a strongly convex optimization problem (e.g., linear regression), SGD converges to the global minimum provided the learning rate is annealed over time.

The shape of the noise

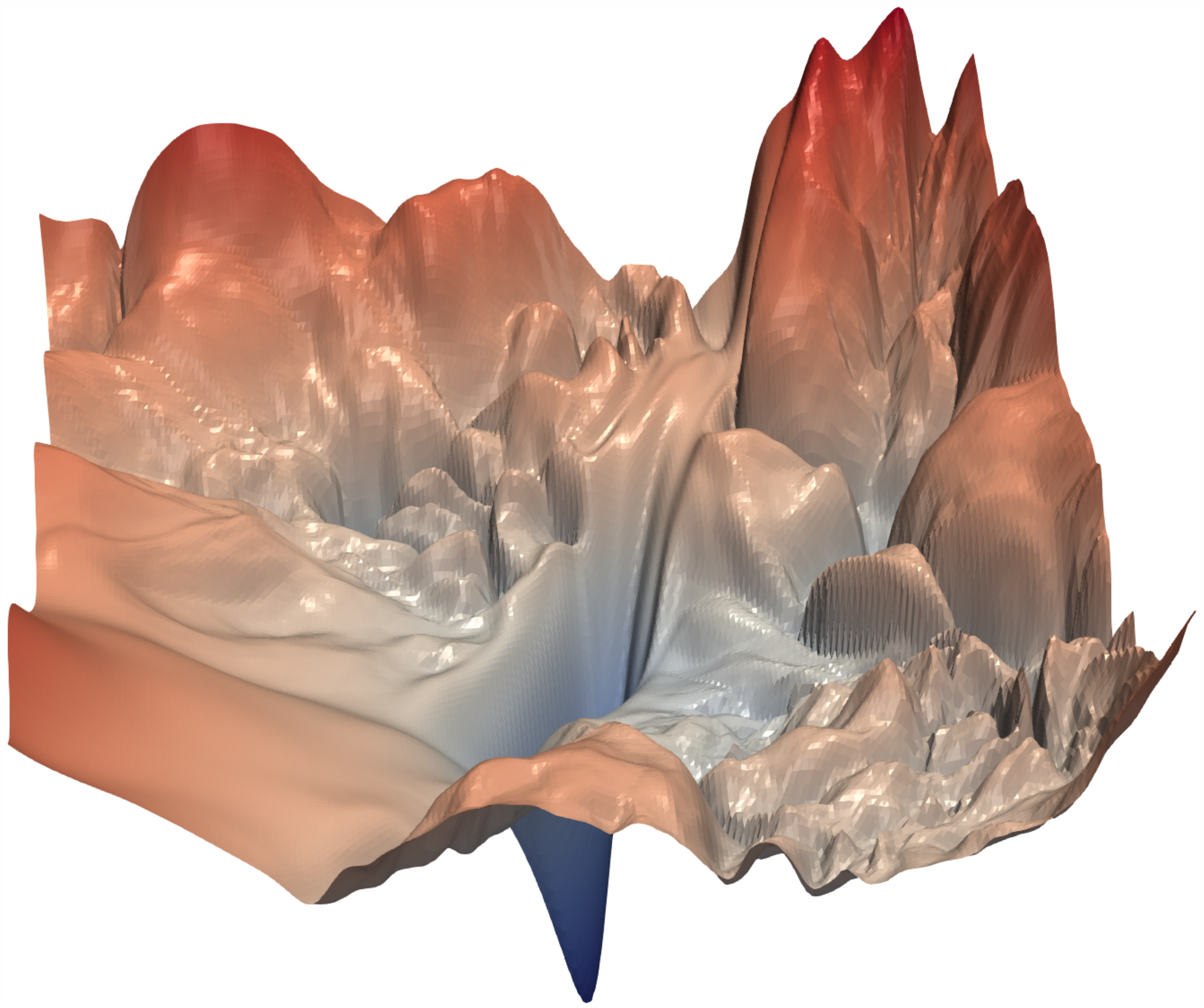
The noise term is non-gaussian and non-isotropic.

$$\nabla_w \ell_w(x_i, y_i) = \nabla_w L(w) + \text{noise}$$

Not an essential difference for a convex problem

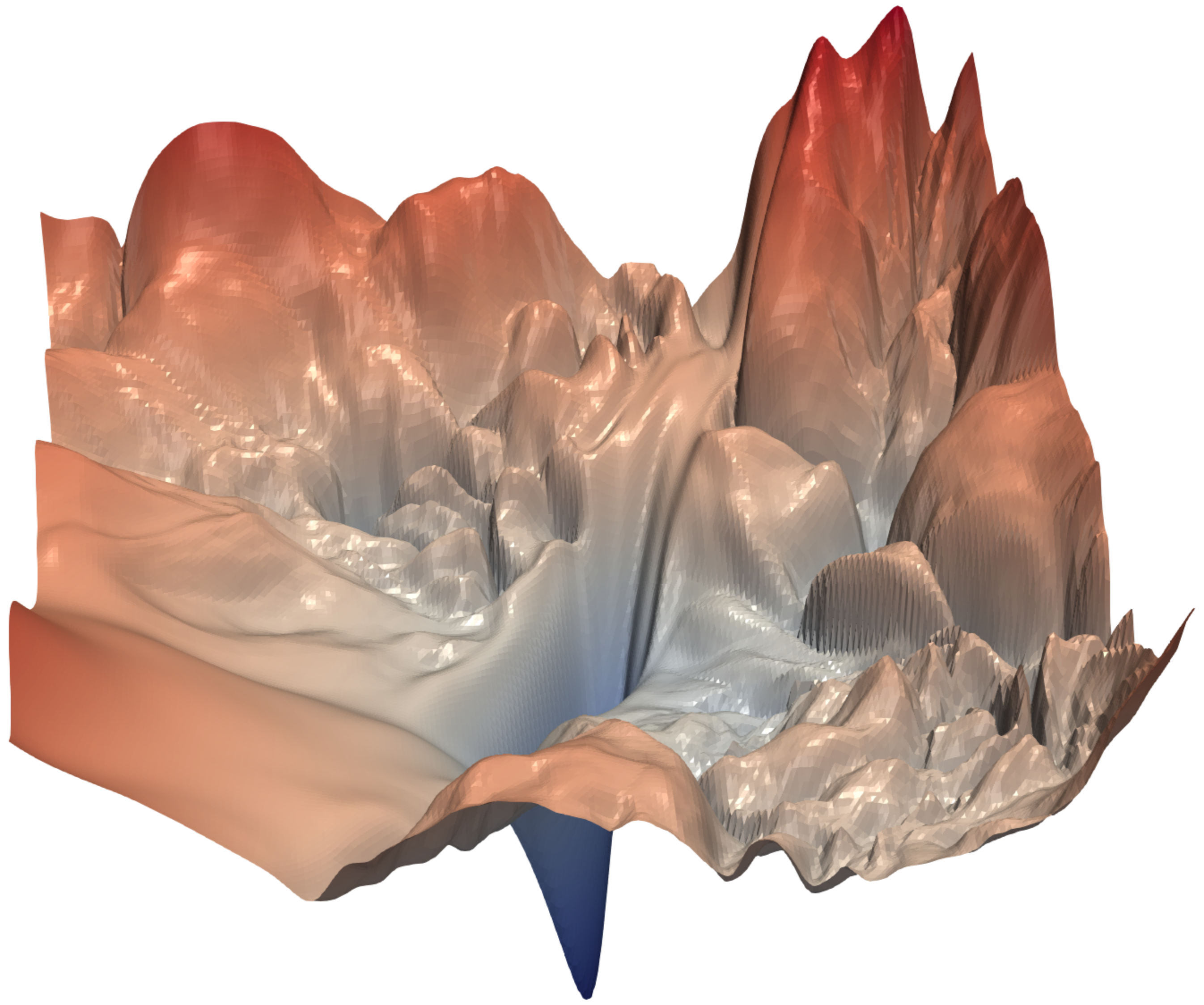


But really changes the dynamics in deep learning!



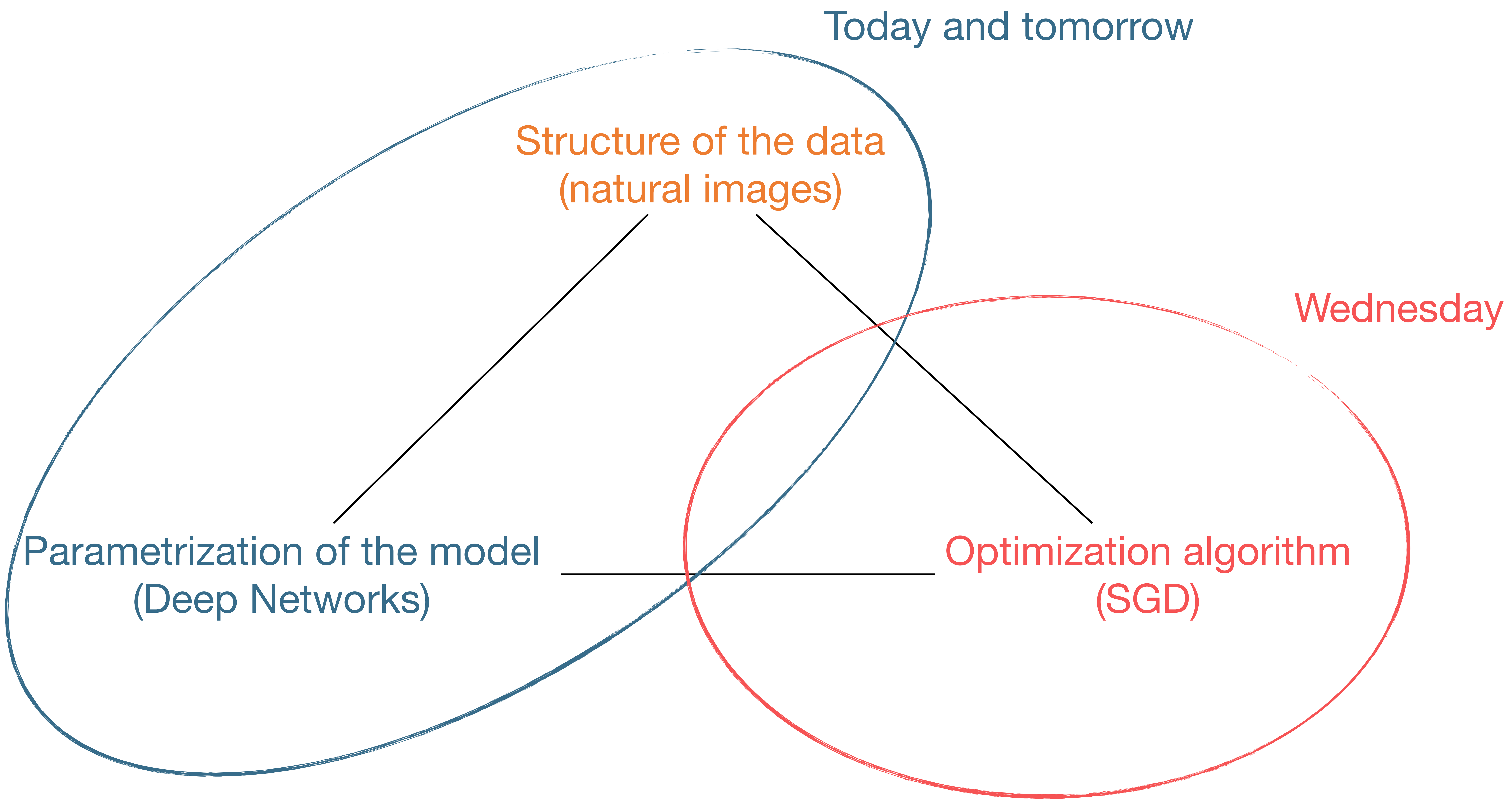
Chaudhari et al., *Stochastic gradient descent performs vatriational inference, converges to limit cycles*, 2018

Classic Machine Learning: minimize some loss function on the training data. Hope it generalizes to the test set



Deep Learning: We don’t want a global minimum, local minima are better. We don’t care about convergence speed (or about convergence at all). Over-parametrization makes things work better. Regularization is only needed at the beginning of training (!)

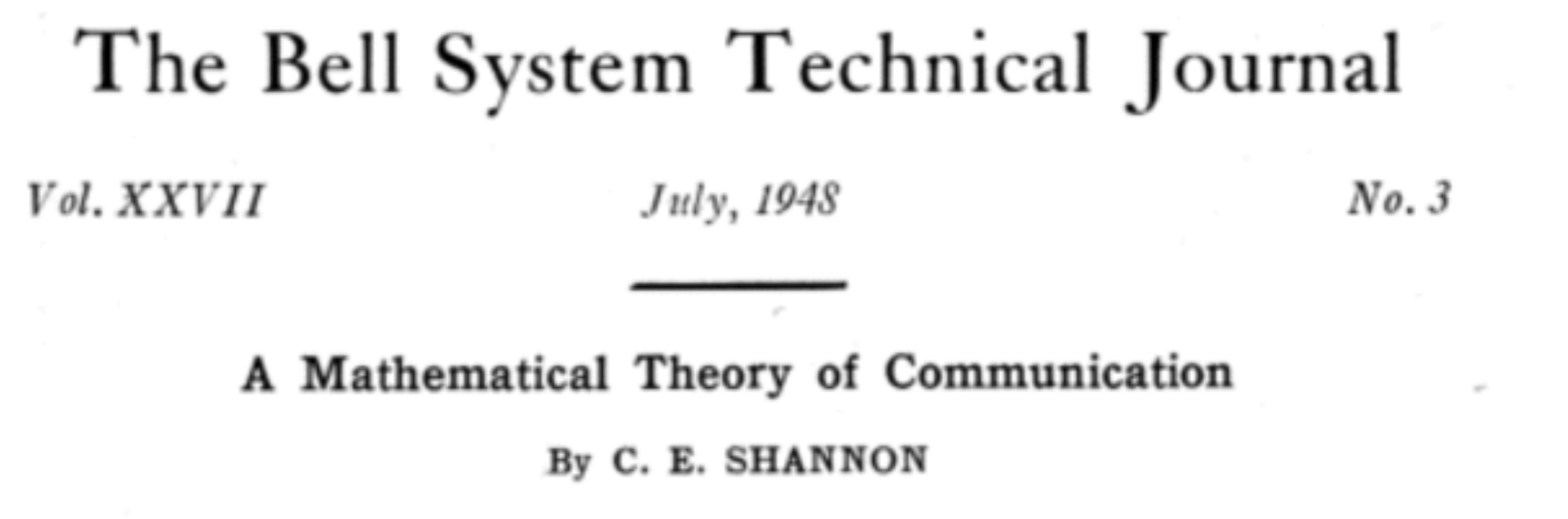
Li et al., *Visualizing the Loss Landscape of Neural Nets*, 2018


Today and tomorrow
Structure of the data
(natural images)
Wednesday
Parametrization of the model
(Deep Networks)
Optimization algorithm
(SGD)

Machine learning and information

Machine learning at its core is about extracting useful task information from the data.

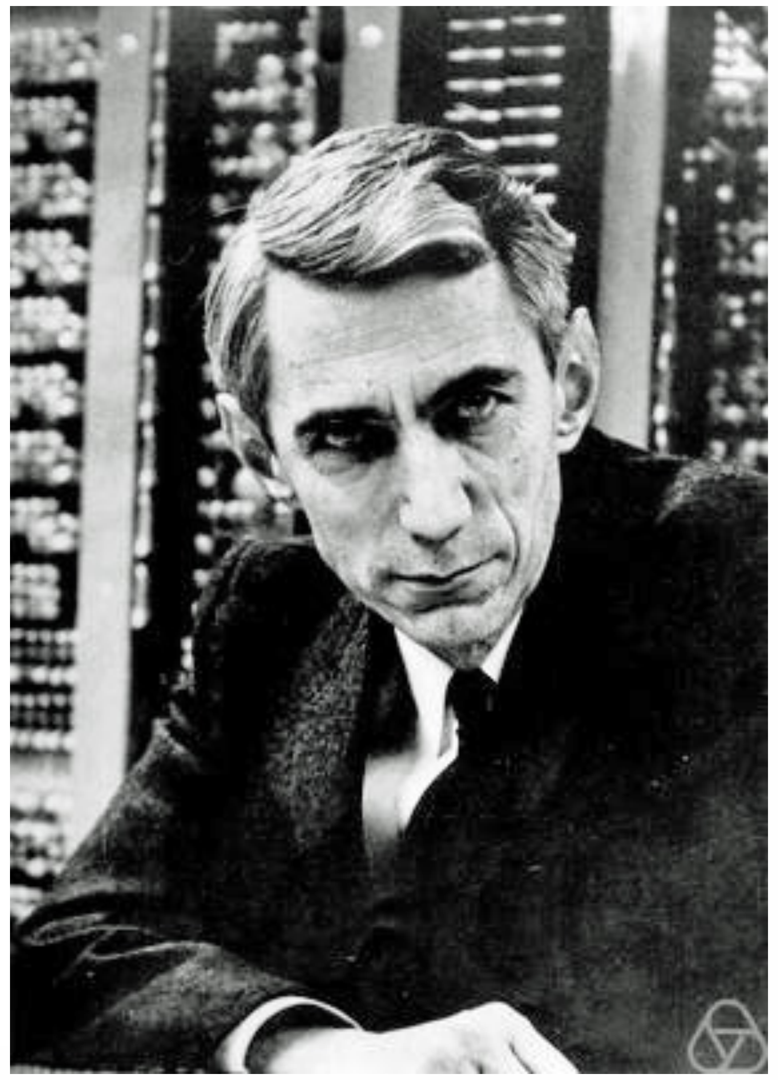
How do we define information?


The Bell System Technical Journal

Vol. XXVII July, 1948 No. 3

A Mathematical Theory of Communication

By C. E. SHANNON



"Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem."

Entropy

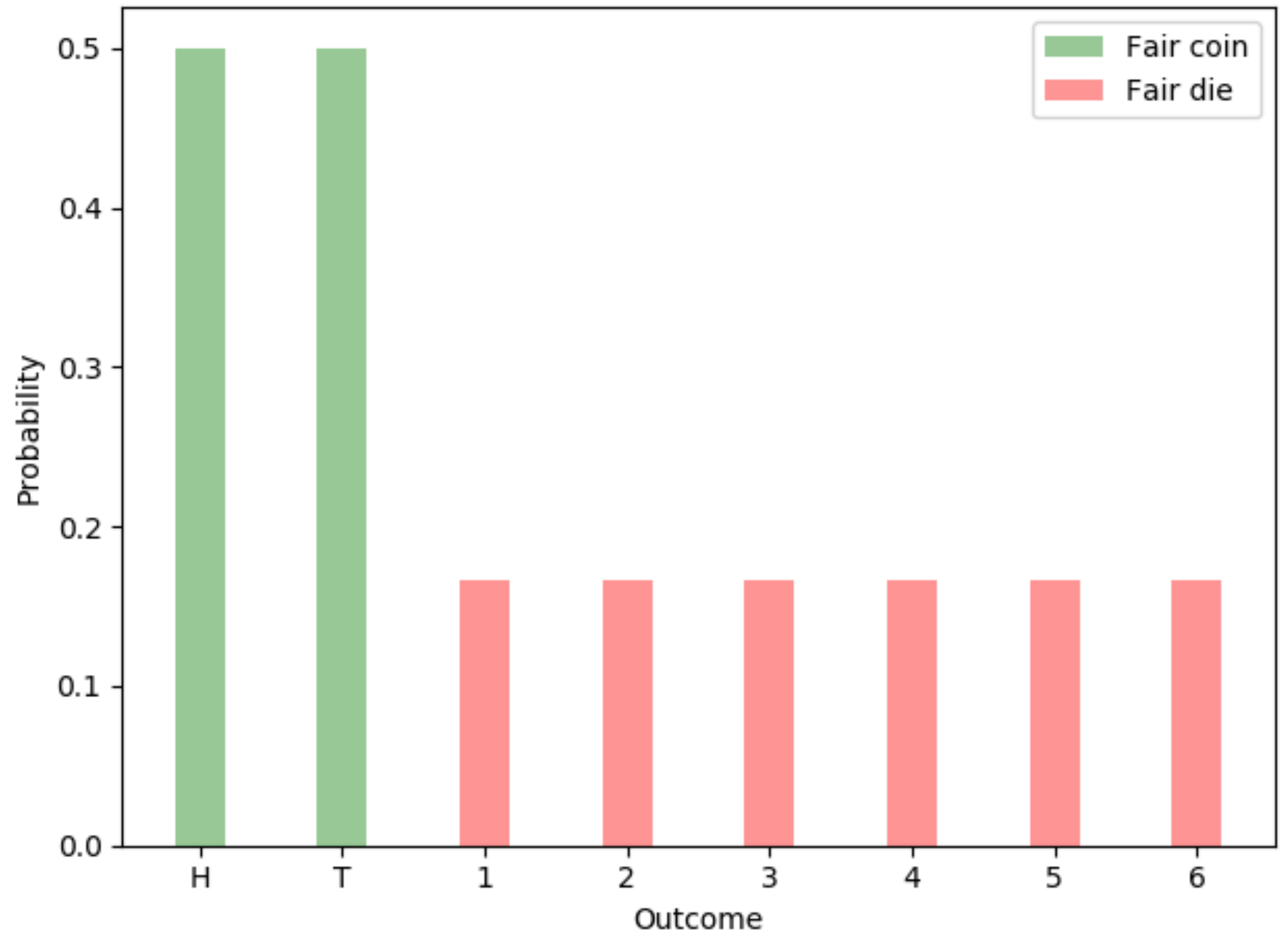
$$H_p(x) = \mathbb{E}_{x\sim p(x)}[-\log p(x)]$$

Shannon coding theorem. The expected minimum coding length (in bits) to encode a sample of the distribution without loss is equal to the entropy of the distribution.

That is, the entropy measures the "information content" of random variable.

Low entropy
2 bits

Higher entropy
2.58 bits



Kullback-Leibler divergence

Entropy is a particular case of the KL divergence (when *q(x)* is discrete and uniform).

$$\mathrm{KL}(p(x) \,\|\, q(x)) = \mathbb{E}_{x \sim p(x)}\left[-\log \frac{p(x)}{q(x)} \right]$$

By Jensen Inequality, the KL-divergence is always positive and it is zero if and only if *p(x)* = *q(x)*.

Cross-Entropy: $H_{q,p}(x) = \mathbb{E}_{x \sim p(x)}[-\log q(x)]$

$$H_{q,p}(x) = H_p(x) + \mathrm{KL}(p(x) \,\|\, q(y))$$

Corollary. The cross-entropy is minimized if and only if q(x) = p(x)

Conditional Entropy and Mutual Information

Conditional Entropy: How much information remains in *y* after having observed *x*

$$H(y \mid x) = \mathbb{E}_{x,y \sim p(y,x)}[-\log p(y \mid x)]$$

Mutual Information: How much information remains in *y* after having observed *x*

$$I(x; y) = H(y) - H(y \mid x)$$

The mutual information can also be interpreted as the expected divergence between the distribution of a random variable before and after an observation.

$$I(x; y) = \mathbb{E}_{x \sim p(x)}[\mathrm{KL}(p(y \mid x) \,\|\, p(y))]$$

Our prototype problem: Image classification

Suppose that our task is to classify images into a finite number of classes:

$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ where each x_i is an image, and each y_i is a label.

We want to learn a model $p_w(y \mid x)$ that predicts the right class of future images.

Example. A simple linear prediction model is $p_W(y \mid x) = \sigma(Wx)$, where $\sigma = (1 + e^{-x})^{-1}$

Cross-entropy loss

We want to maximize the amount of information about the task. Recall that

$$I(x; y) = H(y) - H_p(y \mid x) = \max_w H(y) - H_{p_w, p}(y \mid x)$$

We introduce the empirical cross-entropy loss:

$$\mathscr{L}(w) = H_{p_w}(y \mid x) = \frac{1}{N} \sum_{i=1}^{N} - \log p_w(y_i \mid x_i)$$

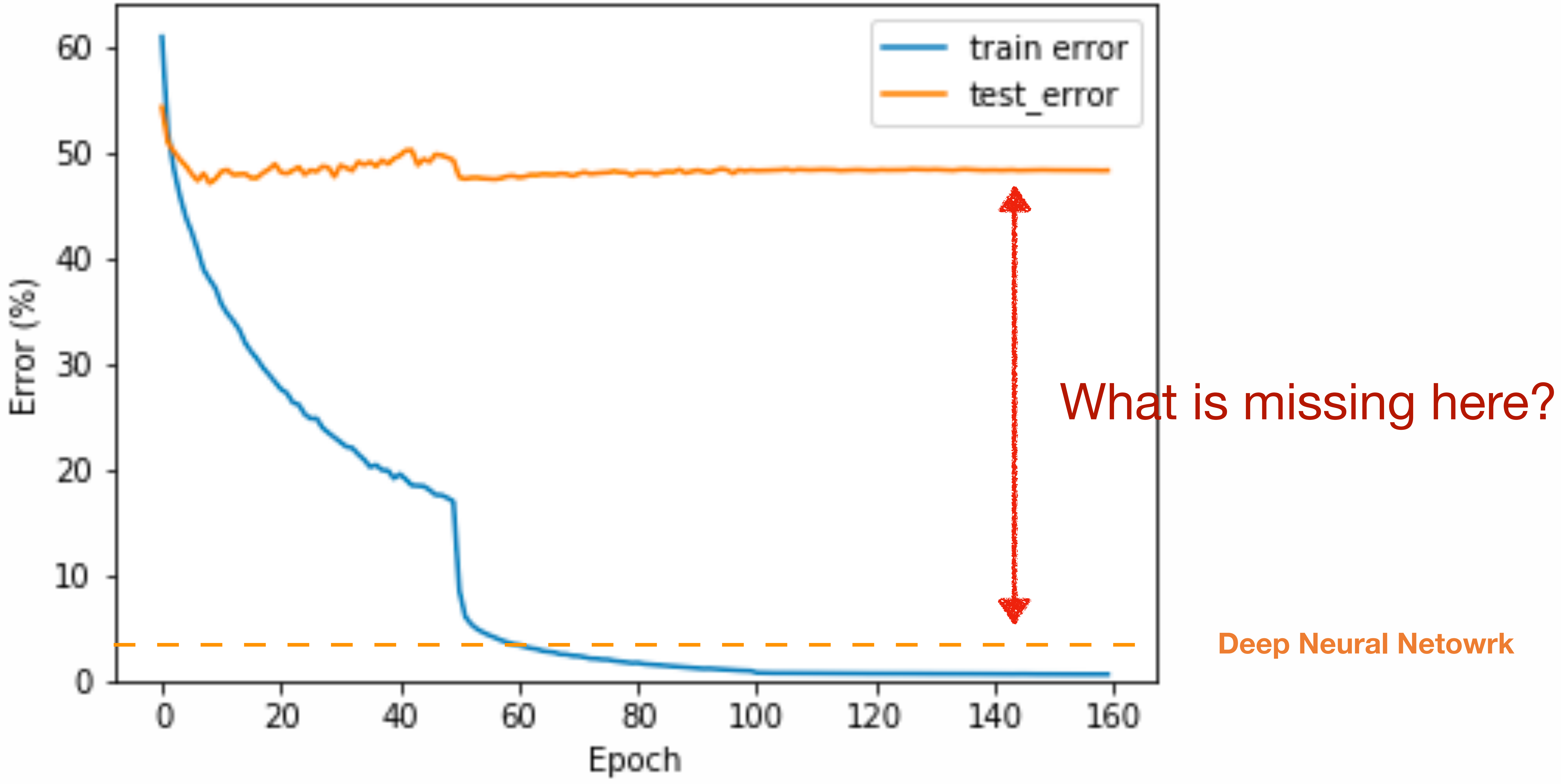
Recall that the cross-entropy is minimized when $p_w(y \mid x)$ is equal to the ground-truth data distribution.

Alternatively can be seen as computing the MLE of w.

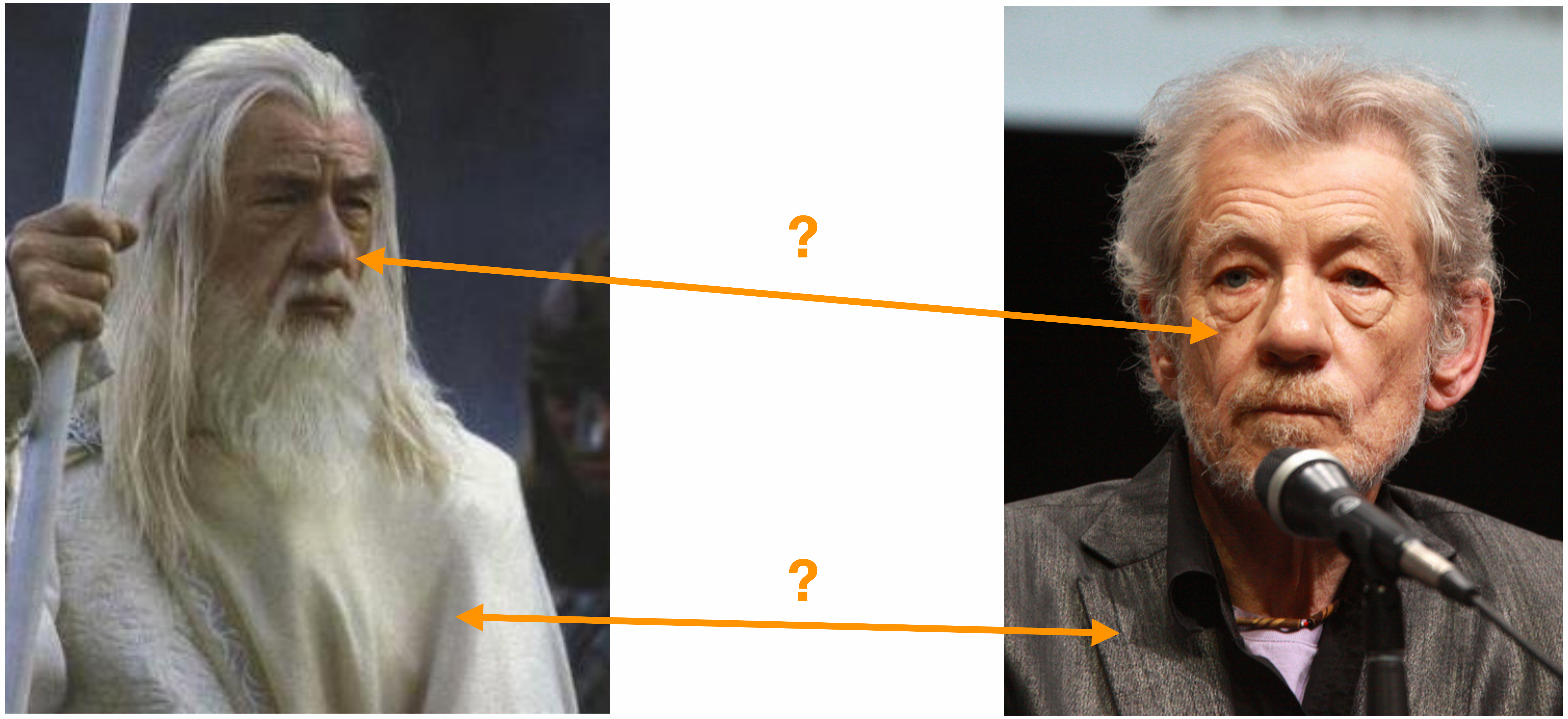
Notebook

A fully connected network can easily bring the train error to zero, but still fails to learn.

Fully connected network on CIFAR-10



What is a nuisance? It depends on the task!



Having different clothes/hairstyle/pose is a nuisance for the task of recognizing the person.
But what if our task is to tag the clothing style in the image?

Pictures of Ian McKellen from https://en.wikipedia.org/wiki/Ian_McKellen https://en.wikipedia.org/wiki/Gandalf

Nuisance for a task

Definition (Nuisance) Assume wlog that the input data can be written as $x = R(o, n)$ for a function $R(o, n)$ and $o, n \sim p(o, n)$. We say that n is a nuisance factor for the task y if:

$$p(y \,|\, R(o, n)) = p(y \,|\, R(o, n'))$$

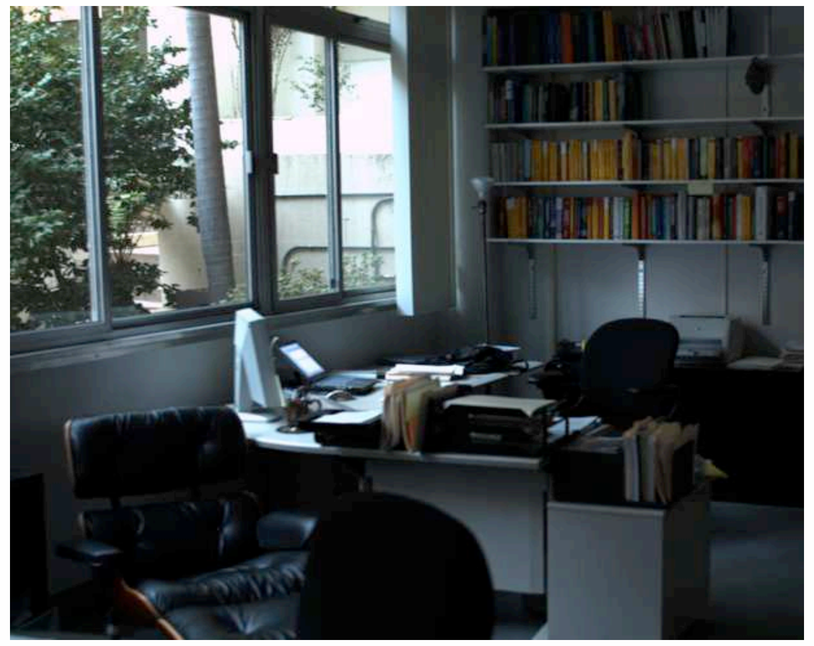
for all $o \in O$, $n \in N$.

Equivalently: A random variable n is a nuisance for y if $I(y; n) = 0$.

Definition (Invariance) A representation $z = \varphi(x)$ is invariant to a nuisance n if $I(z; n) = 0$.

Problem: How do we find a representation of x which is invariant to nuisances?

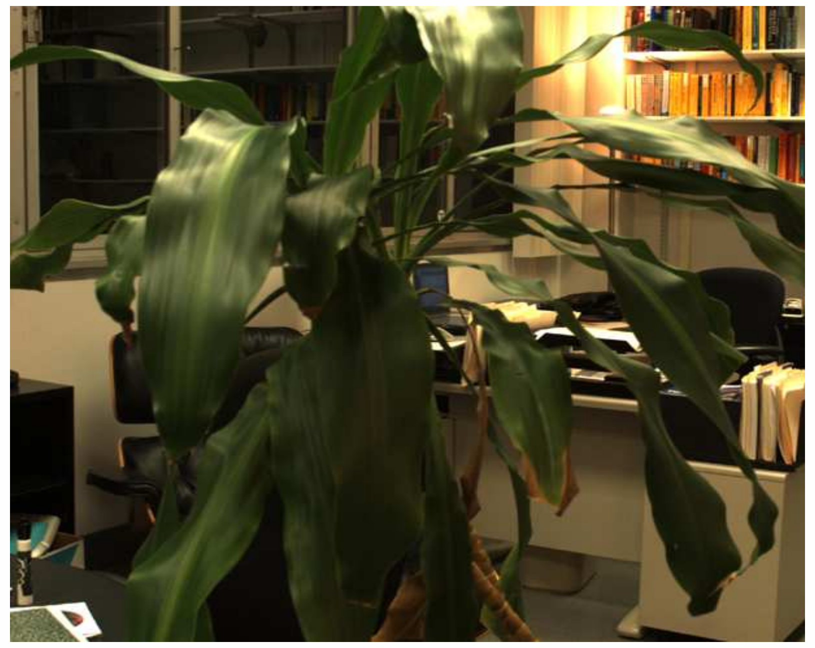
Nuisance variability



$I = h(\xi, \nu)$

Change of nuisance

$\tilde{I} = h(\xi, \tilde{\nu}), \quad \tilde{\nu} = \text{illumination}$



$\tilde{\nu} = \text{visibility}$

$\tilde{\nu} = \text{viewpoint}$

$\tilde{I} = h(\tilde{\xi}, \tilde{\nu}), \quad \tilde{\xi} \neq \xi$

Change of identity

Images from *Steps Toward a Theory of Visual Information*, S. Soatto, 2011

Group nuisances

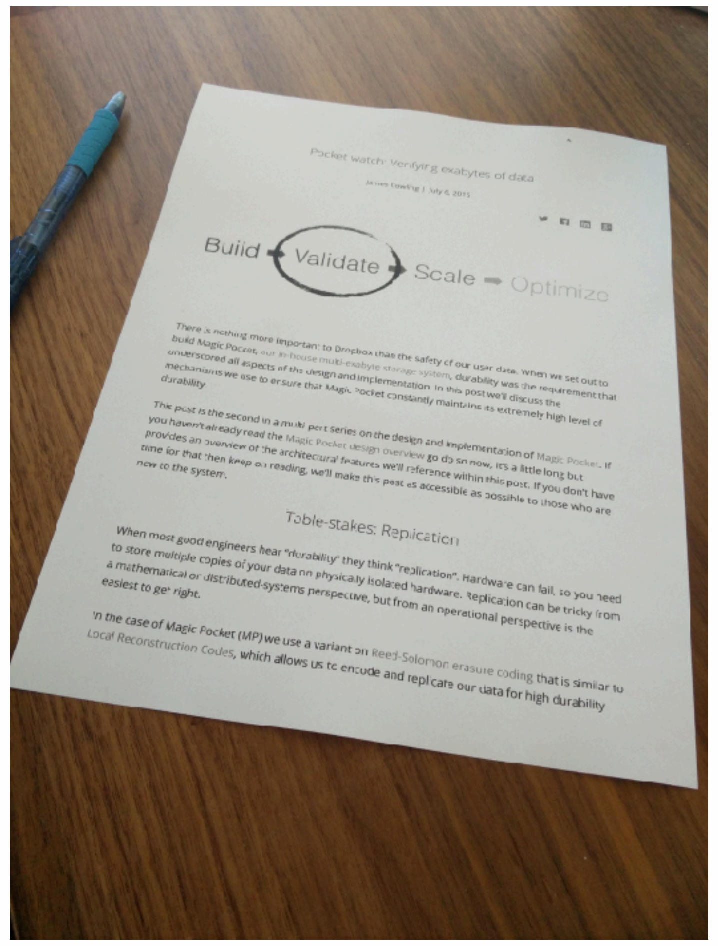
Assume that a nuisance $g \in G$ acts on the data x though a group action:

$$x = R(o, g) = g \cdot R(o, e) = g \cdot x'$$

Examples:

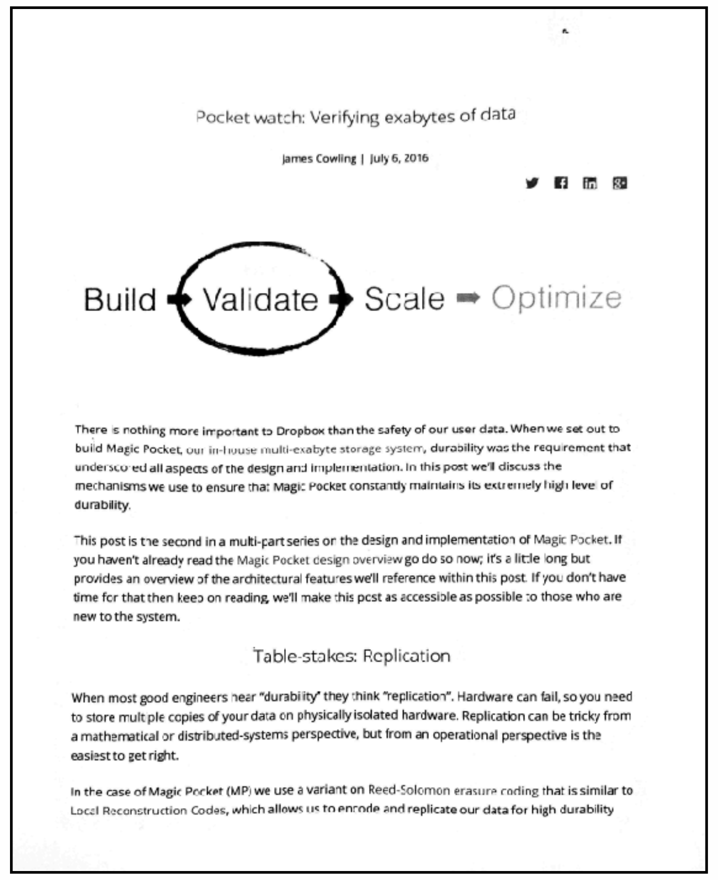
Translation, rotation, scale

$$\mathrm{Aff}\,(\mathbb{R}^2) = GL(2) \ltimes \mathbb{R}^2$$



Projective transform

Change of illumination and contrast



Change of pixel positions

$$PGL(\mathbb{R}^2) \times \mathrm{Diff}\,(\mathbb{R})$$

Change of pixel values (contrast)

Pictures from https://commons.wikimedia.org/wiki/File:Widok_na_Perast_z_zachodu_01.JPG, https://blogs.dropbox.com/tech/2016/08/fast-document-rectification-and-enhancement/

Local group-invariant descriptor

Reference frame need to be unique and robust.

Due to occlusions, we can only trust local features and need redundancy

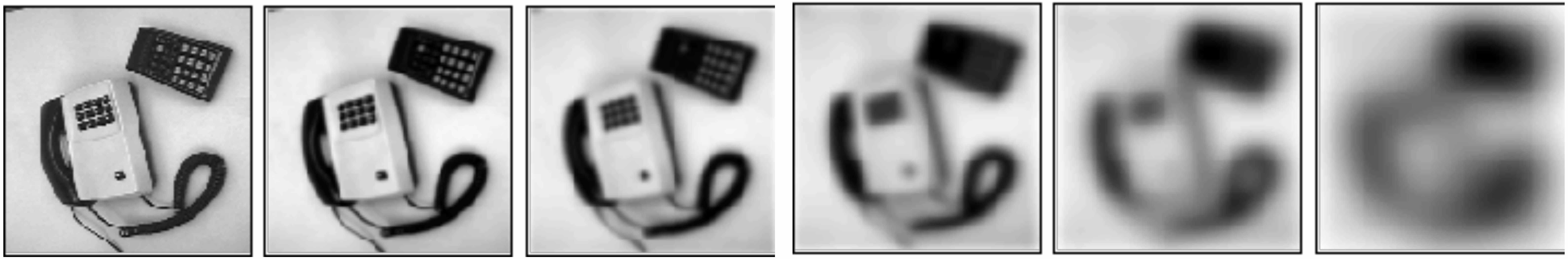
Occlusions Changes in appearance

Need to be robust to all geometric transformations and small deformations.

Need to be robust to changes of illuminations, shadows, …

SIFT: Finding the scale

Find “interesting points” (*i.e.*, local maxima and minima) at all scales.

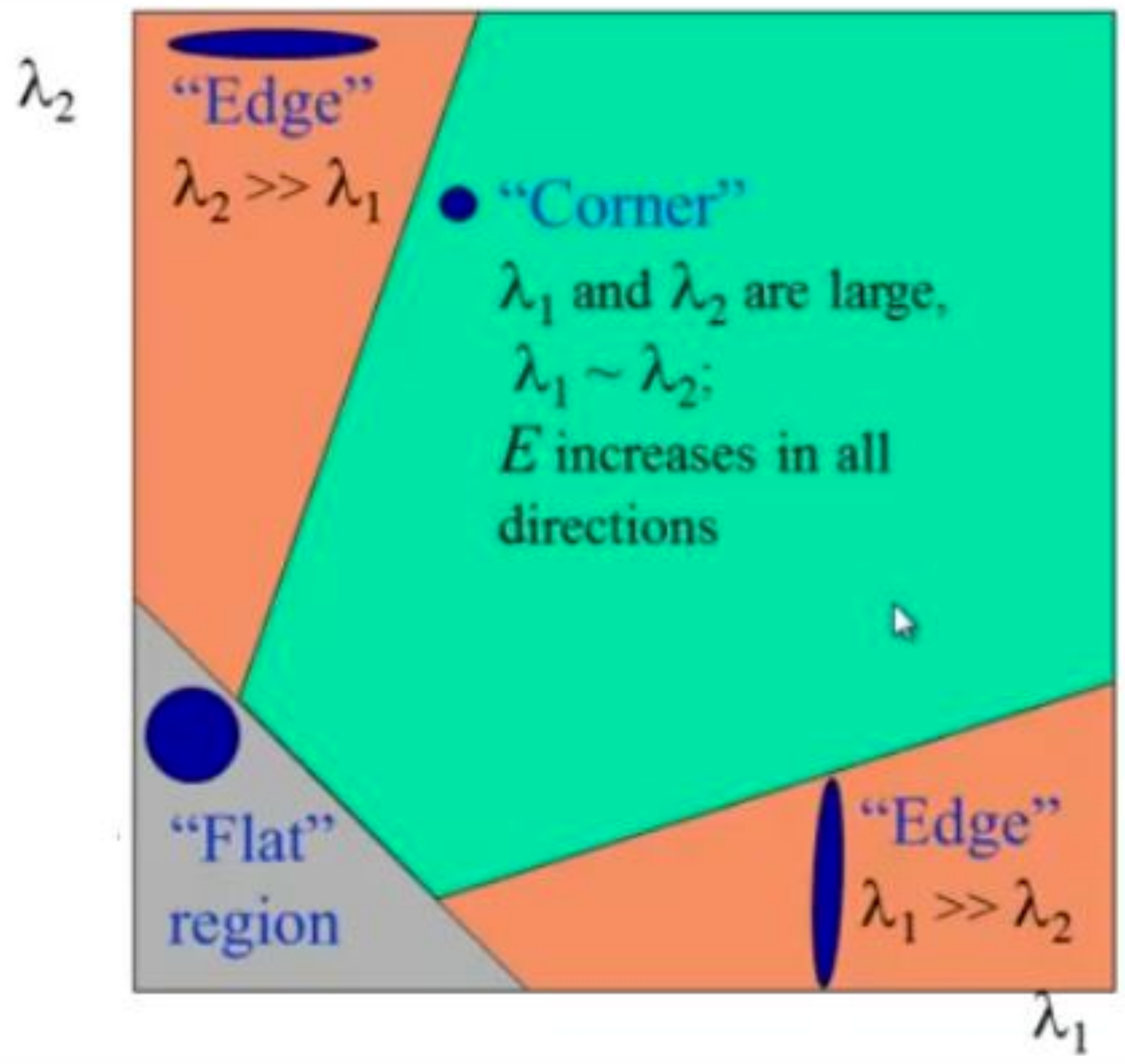


Done by constructing the scale space of the image and finding the first scale at which a local maximum (minimum) stops being a local maximum (minimum).

Harris corner detector

Points along edges are not useful keypoints, as they cannot be localized exactly.

Idea: Compute the Hessian at each interesting point. Consider only the points that have **large eigenvalues of the same magnitude**.



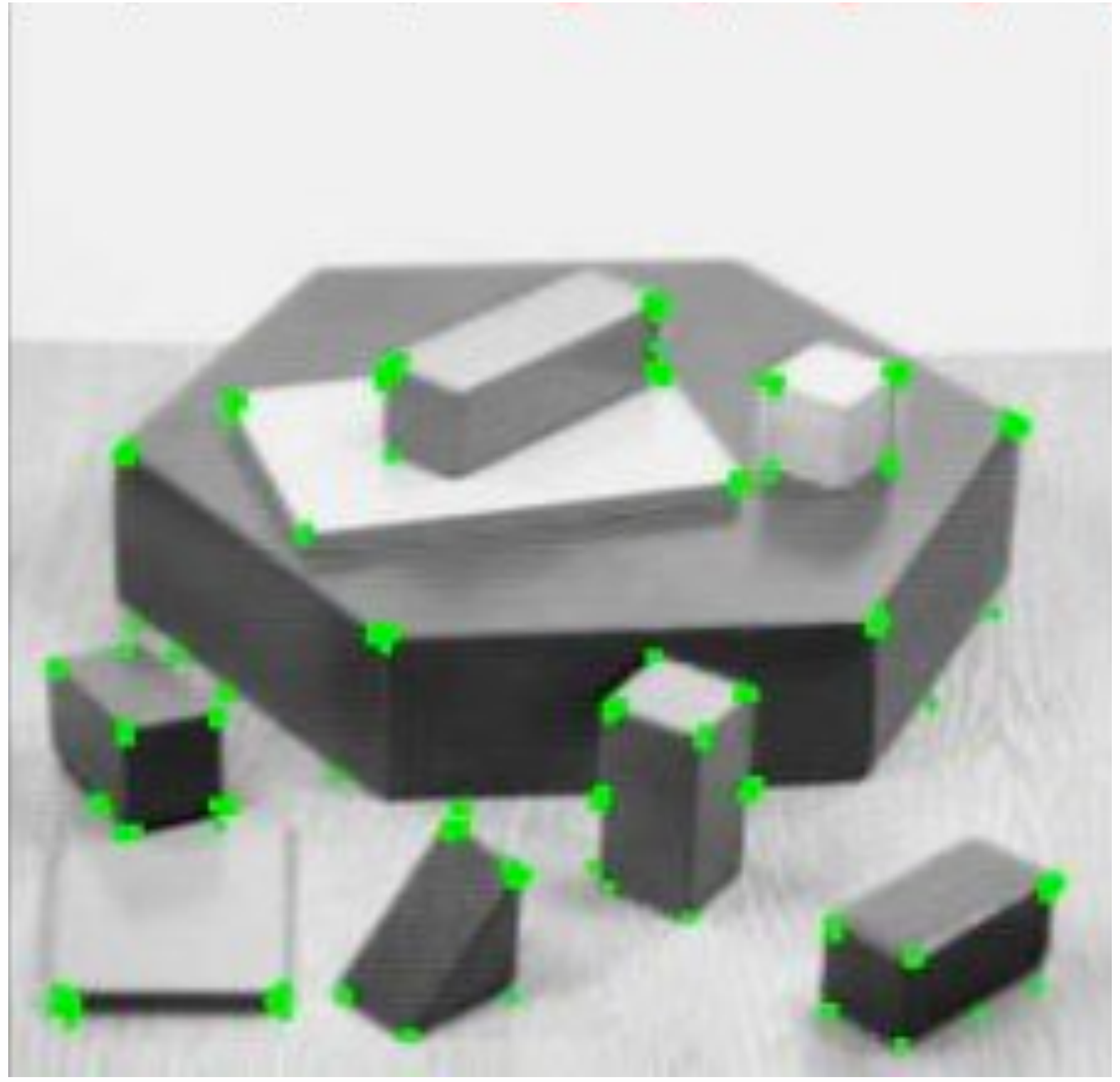
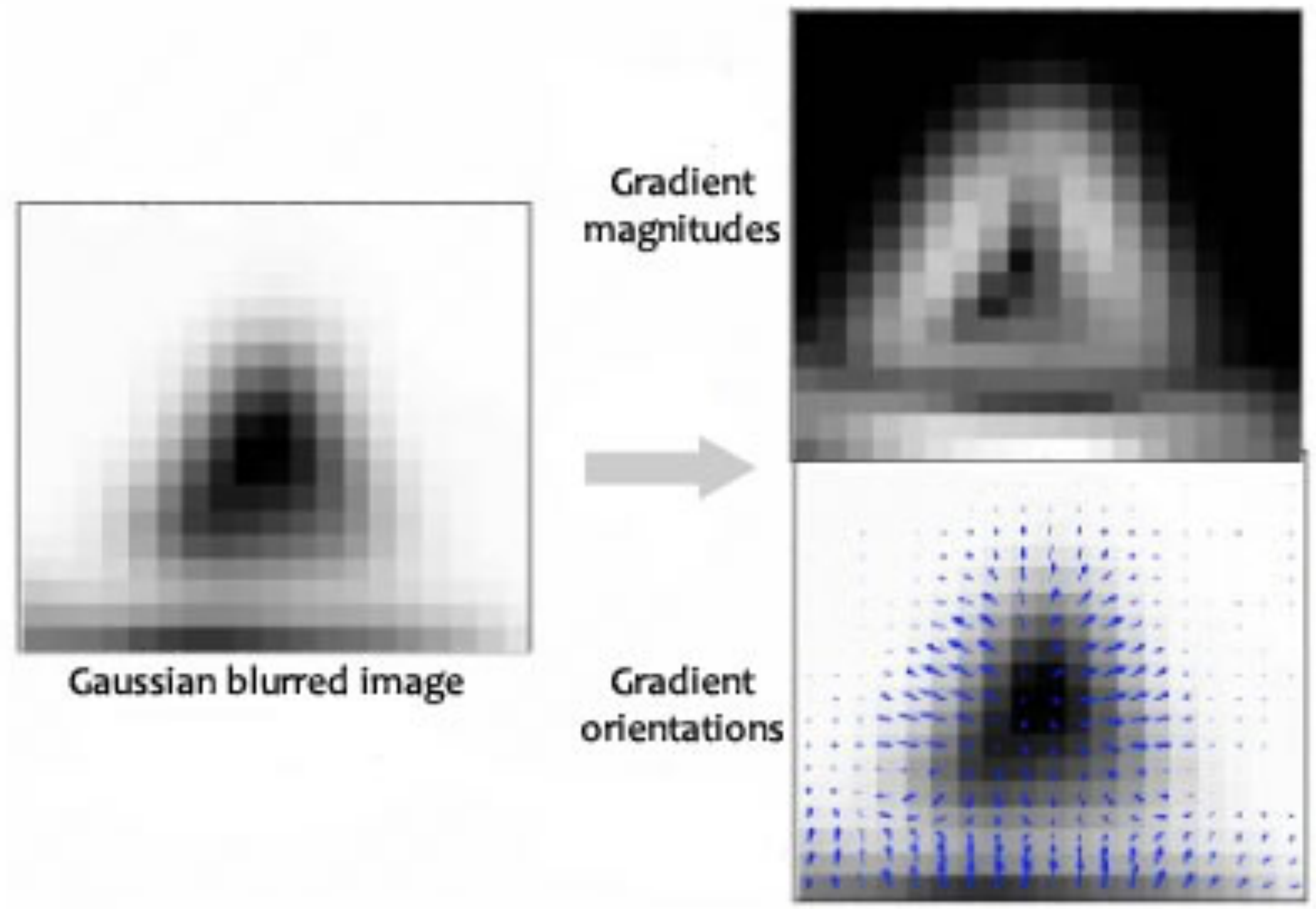
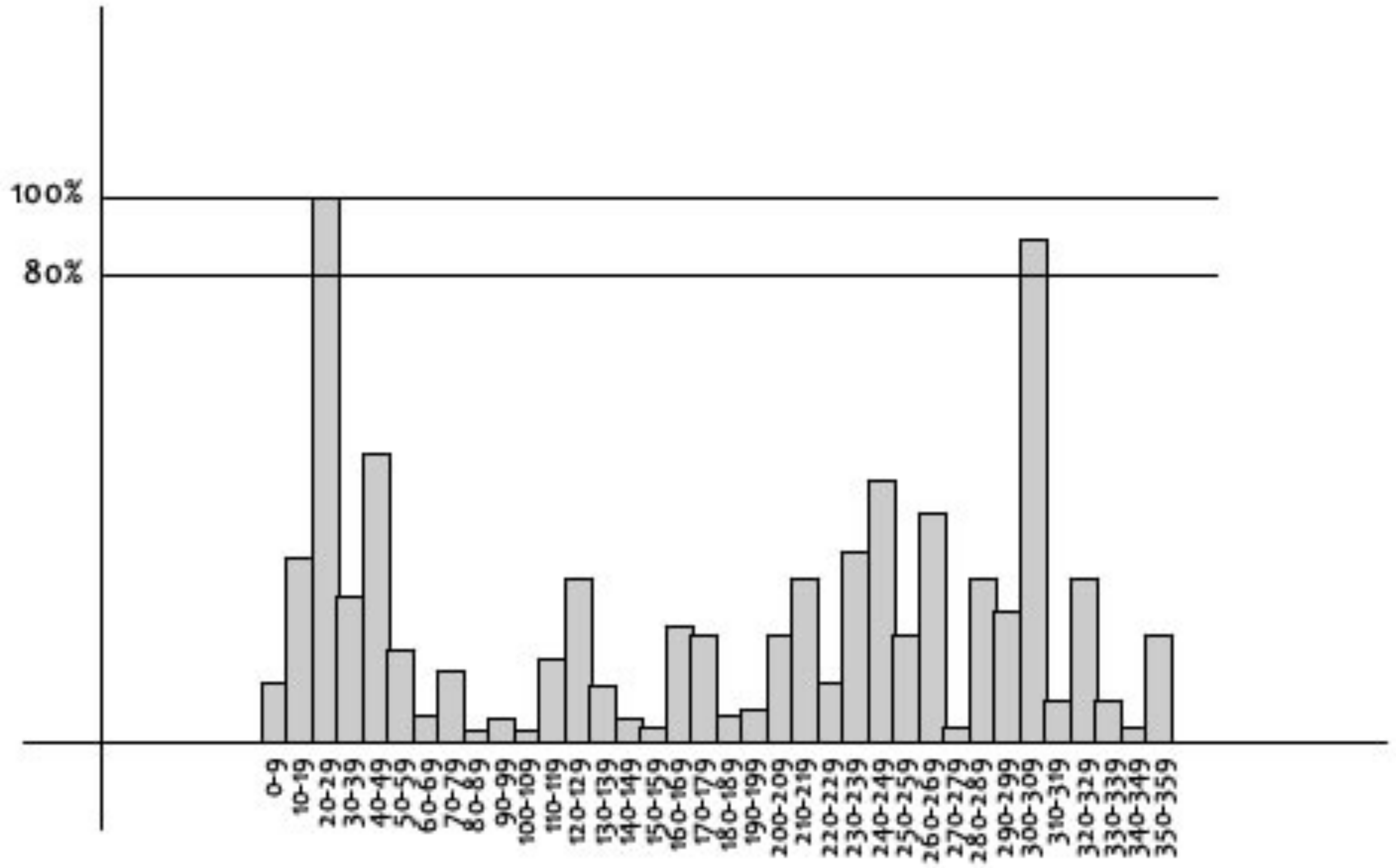


Image from https://docs.opencv.org/3.4.2/dc/d0d/tutorial_py_features_harris.html

Find corner orientation

Decide the orientation of the corner by plotting the histogram of the gradients orientation and find the most frequent orientation.





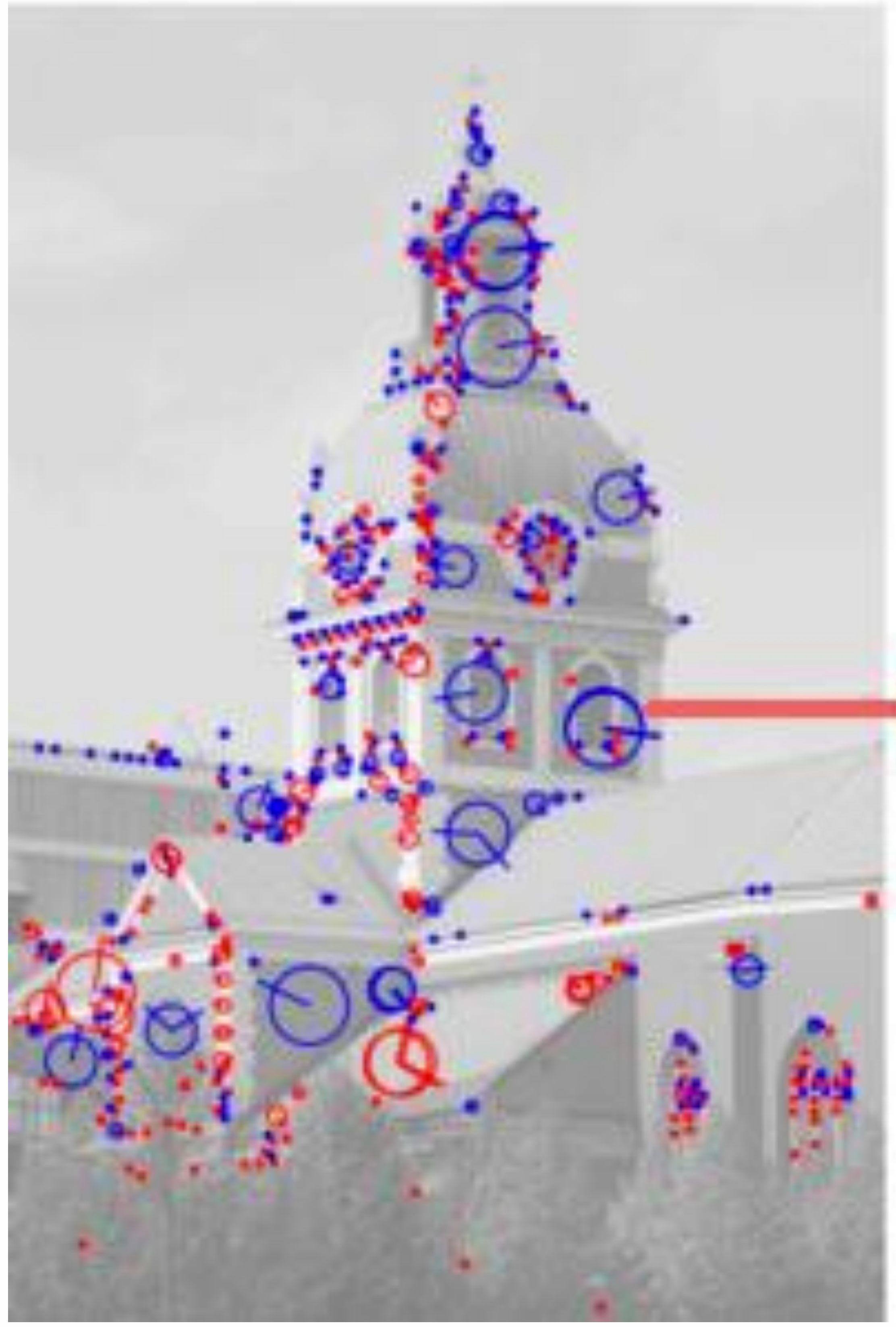
If multiple orientations are very frequent (> 0.8 * max), select all.

Image from http://aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation/

SIFT: Scale-Invariant Feature Transform

Gradient orientation is the only invariant to contrast changes.

Idea: Describe local patch around corner using orientations of the gradients.



Bin together gradients in a patch for robustness to small deformations

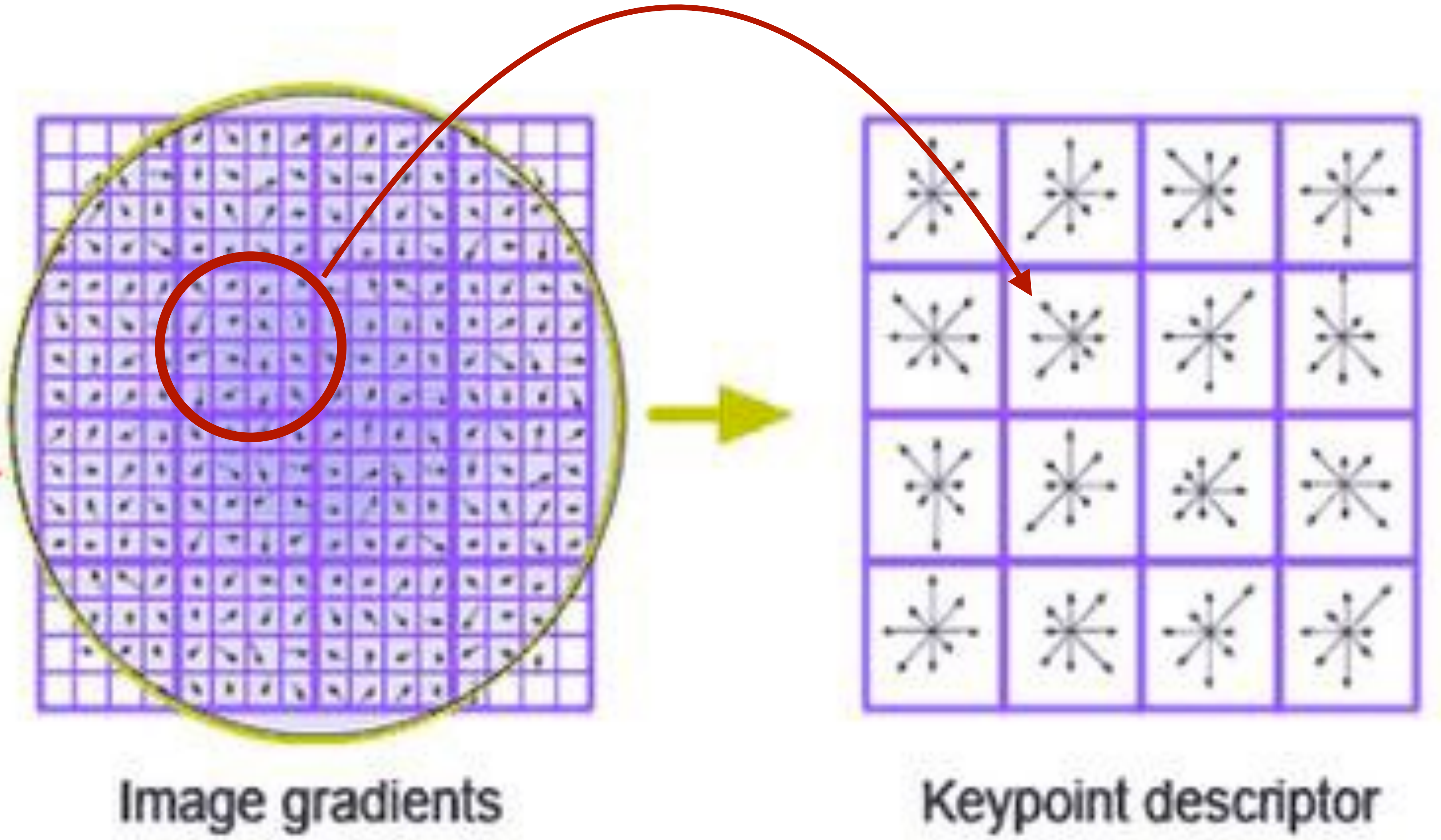
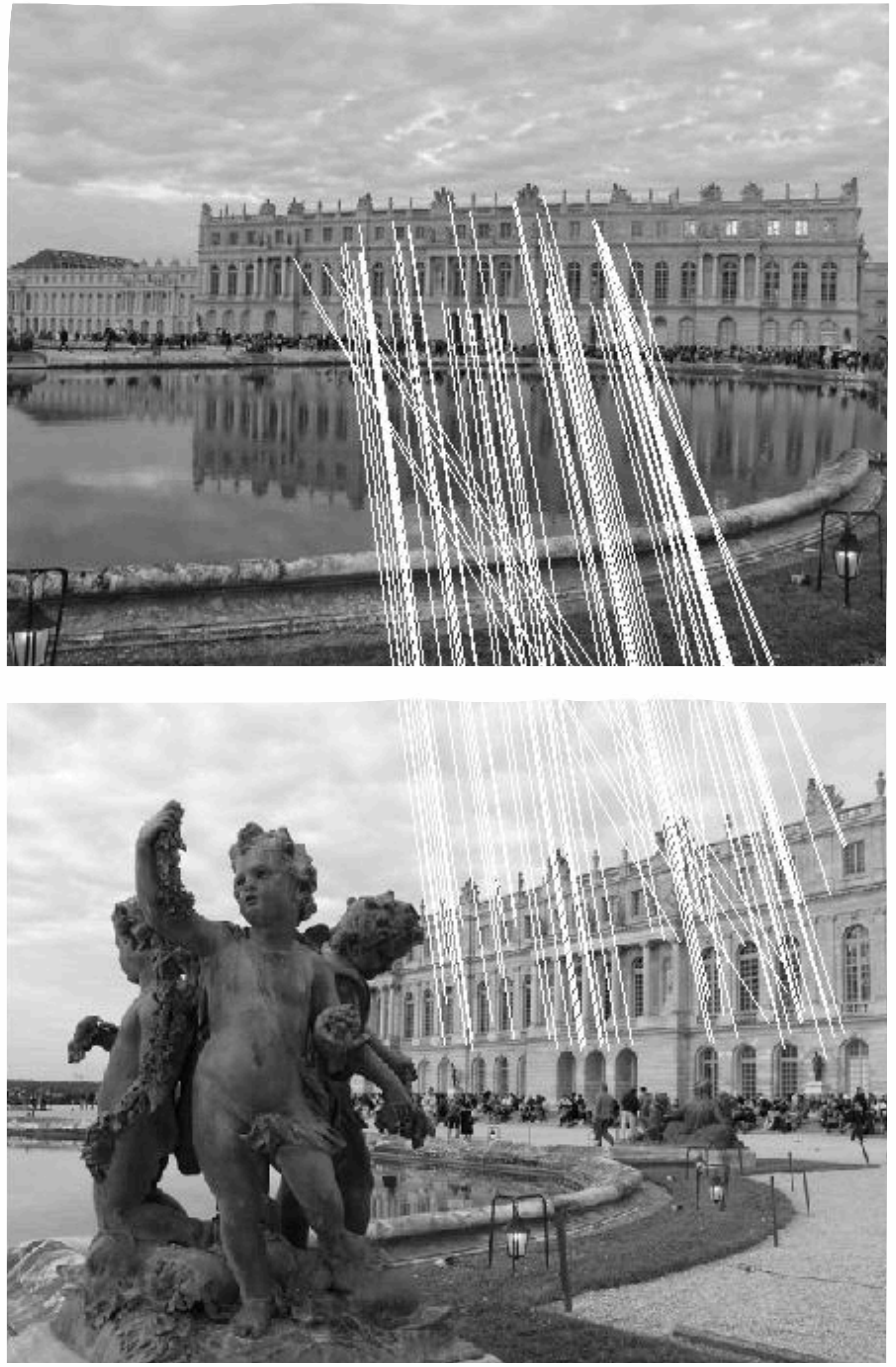


Image from http://aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation/

The state of Computer Vision, circa 2009



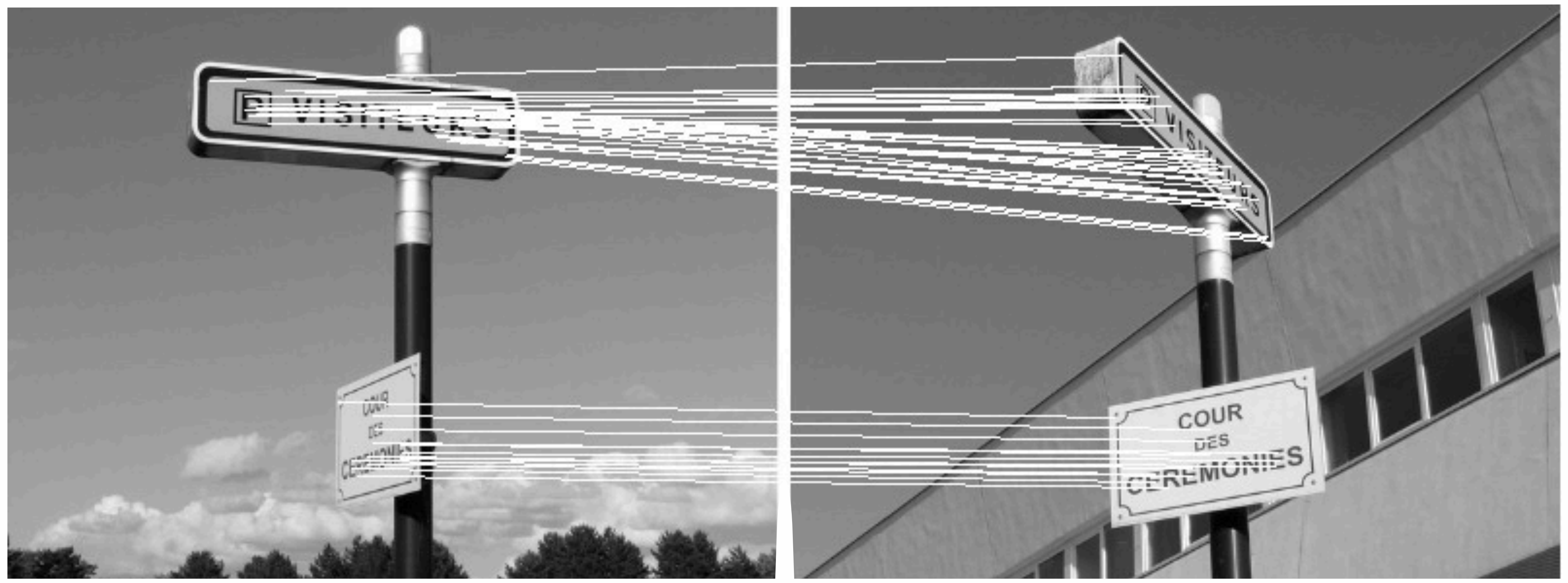


Image from http://www.cmap.polytechnique.fr/~yu/research/ASIFT/demo.html

Feature matching in Visual-Inertial SLAM system

Demo video from https://sites.google.com/site/ktsotsos/visual-inertial-sensor-fusion

Computer Vision now

How do we learn the complex variability of natural objects?

Lecture 2: Learning optimal representations

What is a representation?

$$\begin{tikzcd} \underset{\text{data}}{x} \arrow[r] & \underset{\text{representation}}{z} \arrow[r] & \underset{\text{task}}{y} \\ & \underset{\text{nuisances}}{n} \arrow[ul] & \end{tikzcd}$$

y = label, segmentation, reconstruction, …

Sufficient	$I(z; y) = I(x; y)$
Nuisance invariance	$n \perp\!\!\!\perp y \Rightarrow I(n; z) = 0$
Minimal	$I(x; z) =$ minimal
Compositional	Minimal component correlation?

The work-horse of representation learning: G-equivariant operators

Let a group G act on the data: We don’t want to learn the same thing over and over again.

$$f(x) = \text{“dog”}$$

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

How do we construct a general group invariant representation?

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
$\hat{f}(x) = \max g \cdot f(x)$.
We can write an invariant function as a composition of simpler equivariant functions.

Linear G-equivariant operators

G-convolution: Let G be a group with an Haar measure, and let $f, k : G \to \mathbb{R}$. We define the *G-convolution*:

$$f \star_G k\,(x) = \int f(xg^{-1})k(g)d\mu(g)$$

Proposition (Kondor et al., 2018) Let G be a compact group, and let $L(G) = \{f: G \to \mathbb{R}\}$. Then $\Phi: L(G) \to L(G)$ is a linear G-equivariant operator if and only if $\Phi(f) = f \star_G g$ for some kernel $k(x) : G \to \mathbb{R}$.*

Example: Let $G = \mathbb{Z}^2$ be the translation group on a lattice. We can think of an image as a function $f: \mathbb{Z}^2 \to \mathbb{R}$, so that $f \in L(G)$. The only translation equivariant operators are $\mathbb{Z}^2$-convolutions.

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

Kondor and Trivedi, *On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups, 2018*

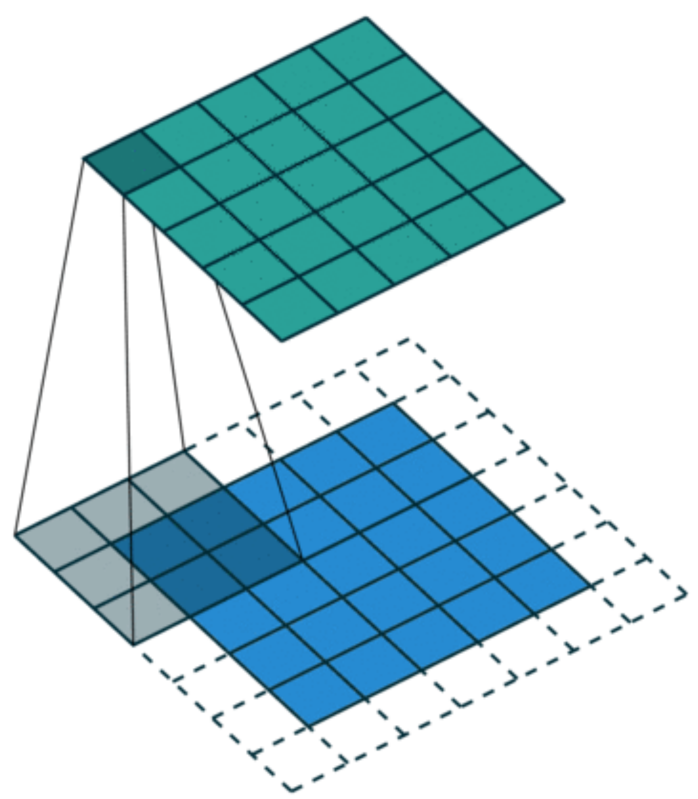
* The result extends to L(X) where X is any set on which G acts transitively.

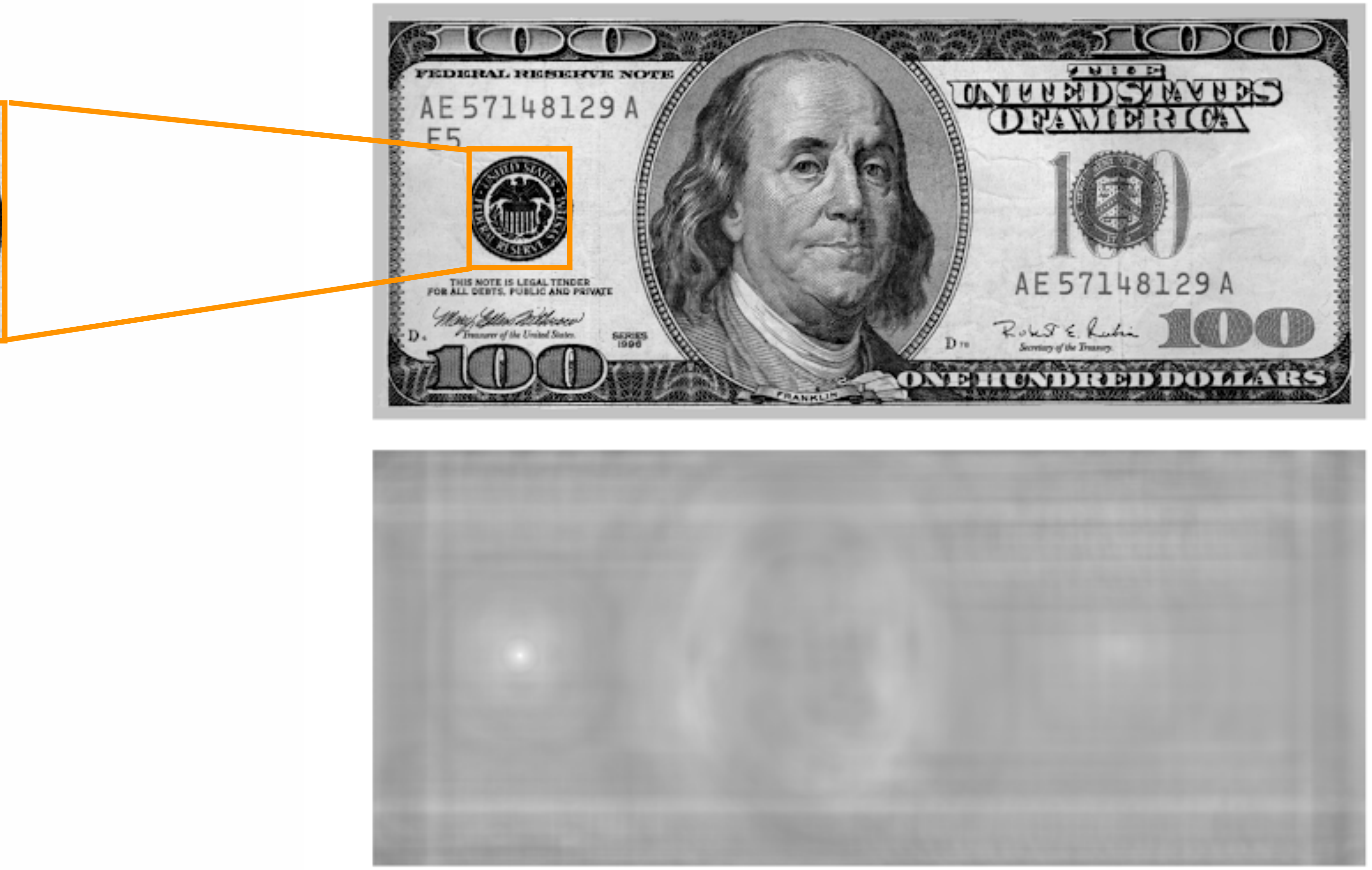
Convolutional Neural Networks

We want to be equivariant/invariant to translations in the image plane

$$f(x) \simeq \sigma \circ \Phi_{w_0}(x)$$

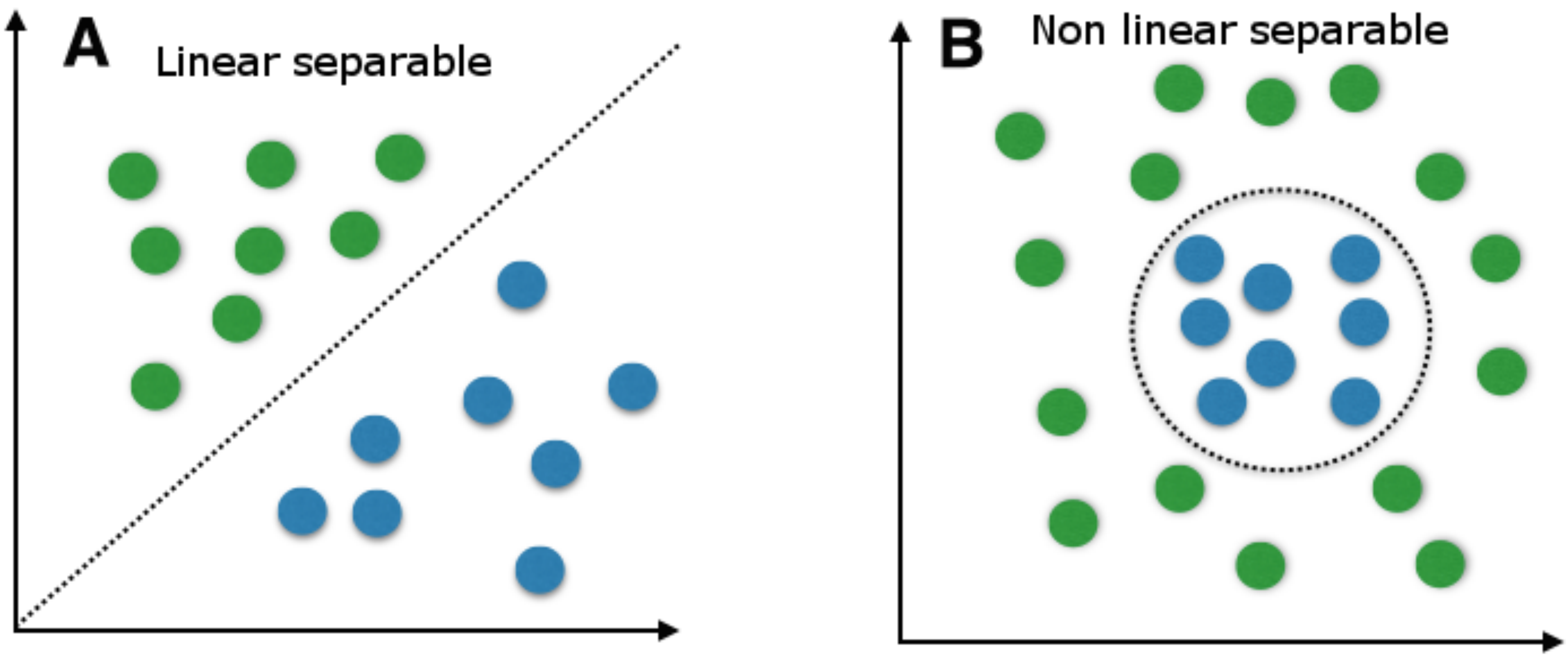
For the group $\mathbb{Z} \times \mathbb{Z}$ of planar translations of a discretized image:





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

Linear is not enough

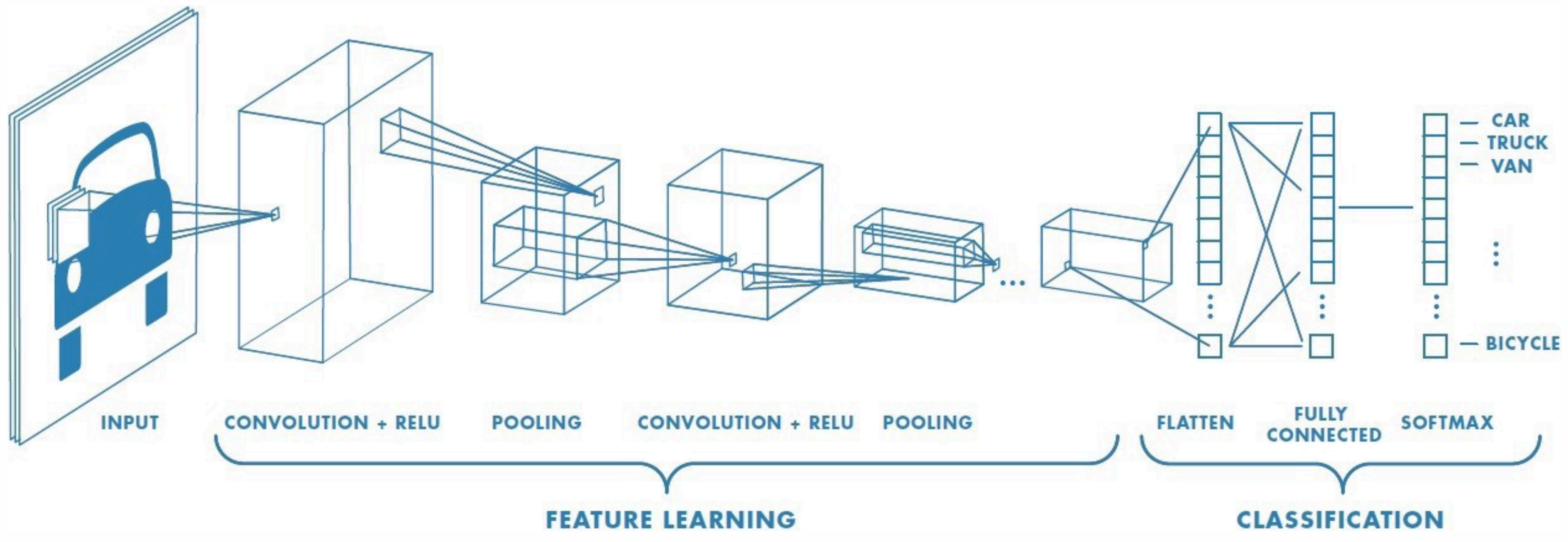


Deep Convolutional Neural Networks

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

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

We learn the convolution kernels by minimizing the cross-entropy loss with SGD.



What it looks like in practice

```
class AllCNN(nn.Module):
    def __init__(self, n_channels=3, n_classes=10):
        super(AllCNN, self).__init__()
        n_filter1 = 96
        n_filter2 = 192
        self.features = nn.Sequential(
            nn.Conv2d(n_channels, n_filter1, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(n_filter1, n_filter1, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(n_filter1, n_filter2, kernel_size=3, stride=2),
            nn.ReLU(),
            nn.Conv2d(n_filter2, n_filter2, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(n_filter2, n_filter2, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(n_filter2, n_filter2, kernel_size=3, stride=2),
            nn.ReLU(),
        )
        self.classifier = nn.Sequential(
            nn.Conv2d(n_filter2, n_filter2, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(n_filter2, n_filter2, kernel_size=1),
            nn.ReLU(),
            nn.Conv2d(n_filter2, n_classes),
            nn.AvgPool2d(8),
        )
    def forward(self, input):
        features = self.features(input)
        return self.classifier(features)
```

It's convolutions all the way down

Architecture from *Striving for Simplicity: The All Convolutional Net*, Spriengberg et al., 2014

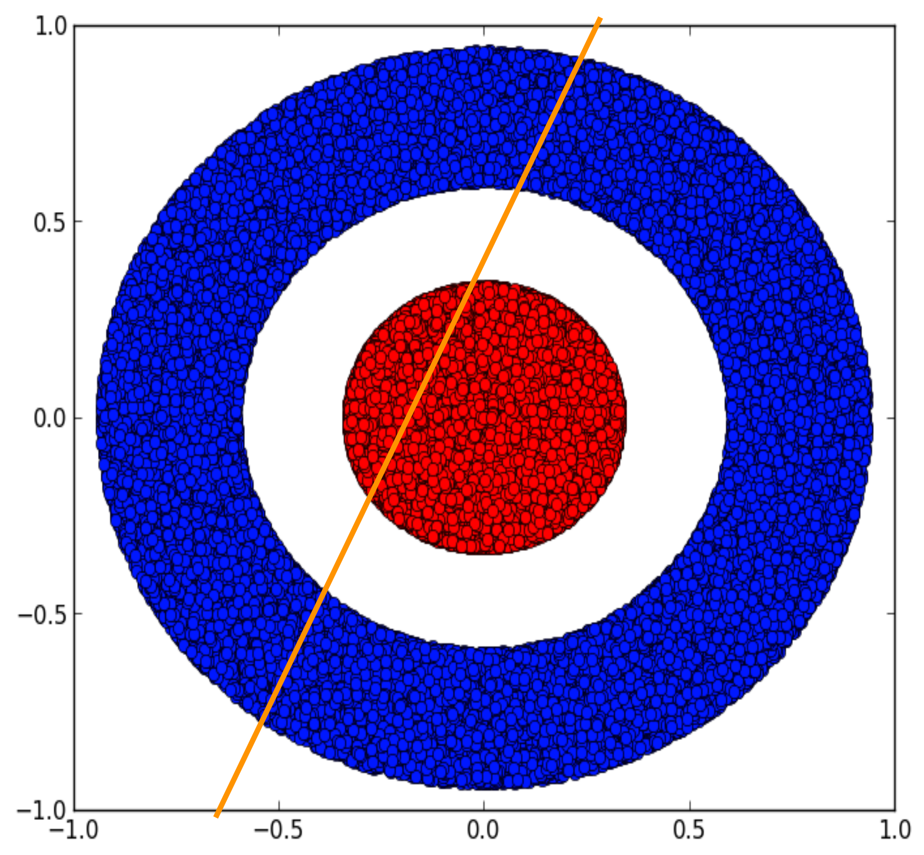
Notebook

Train this network

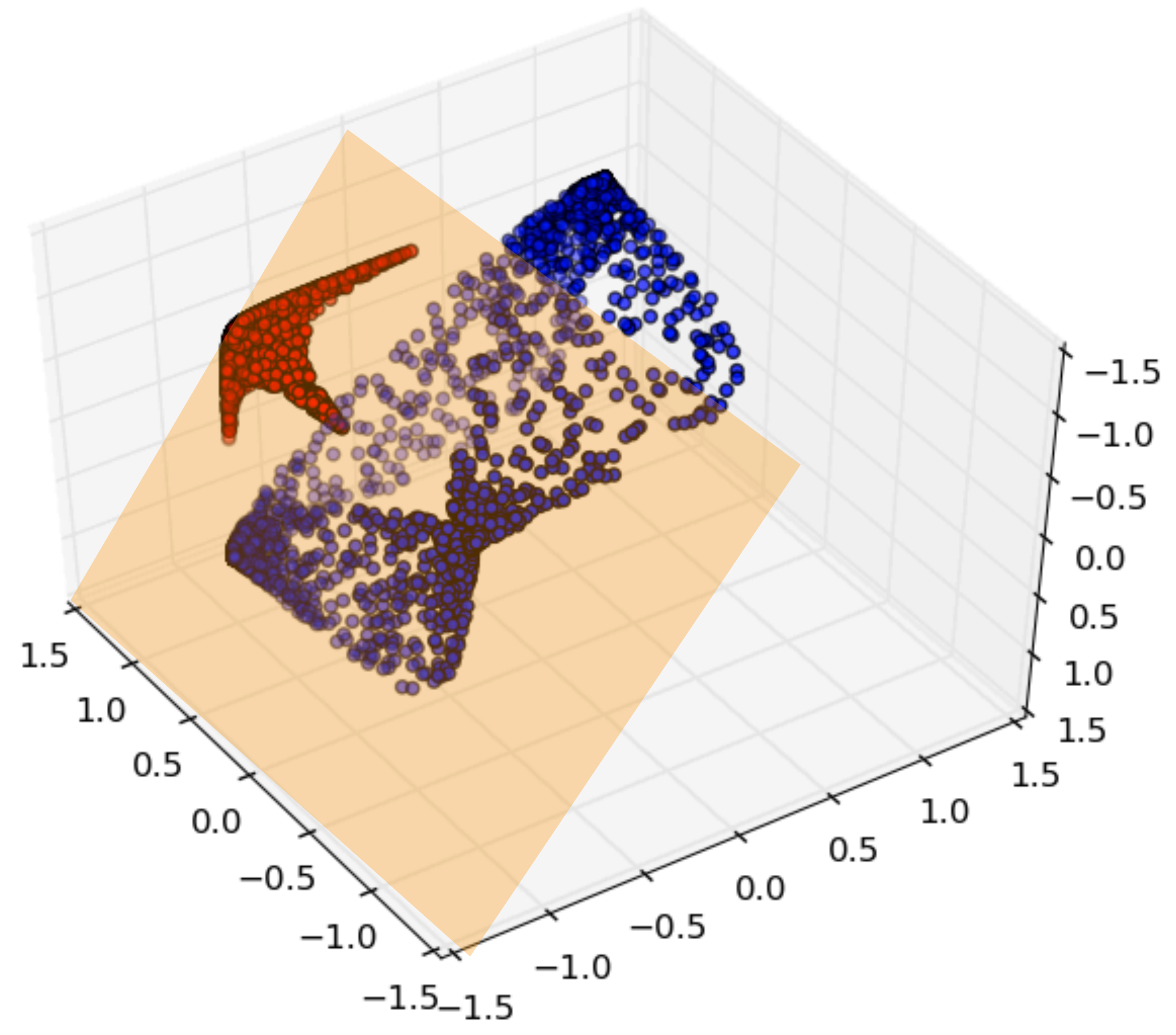
The effect of depth: increasing expressiveness

Depth is *necessary* to make the data linearly separable:

Non linearly-separable



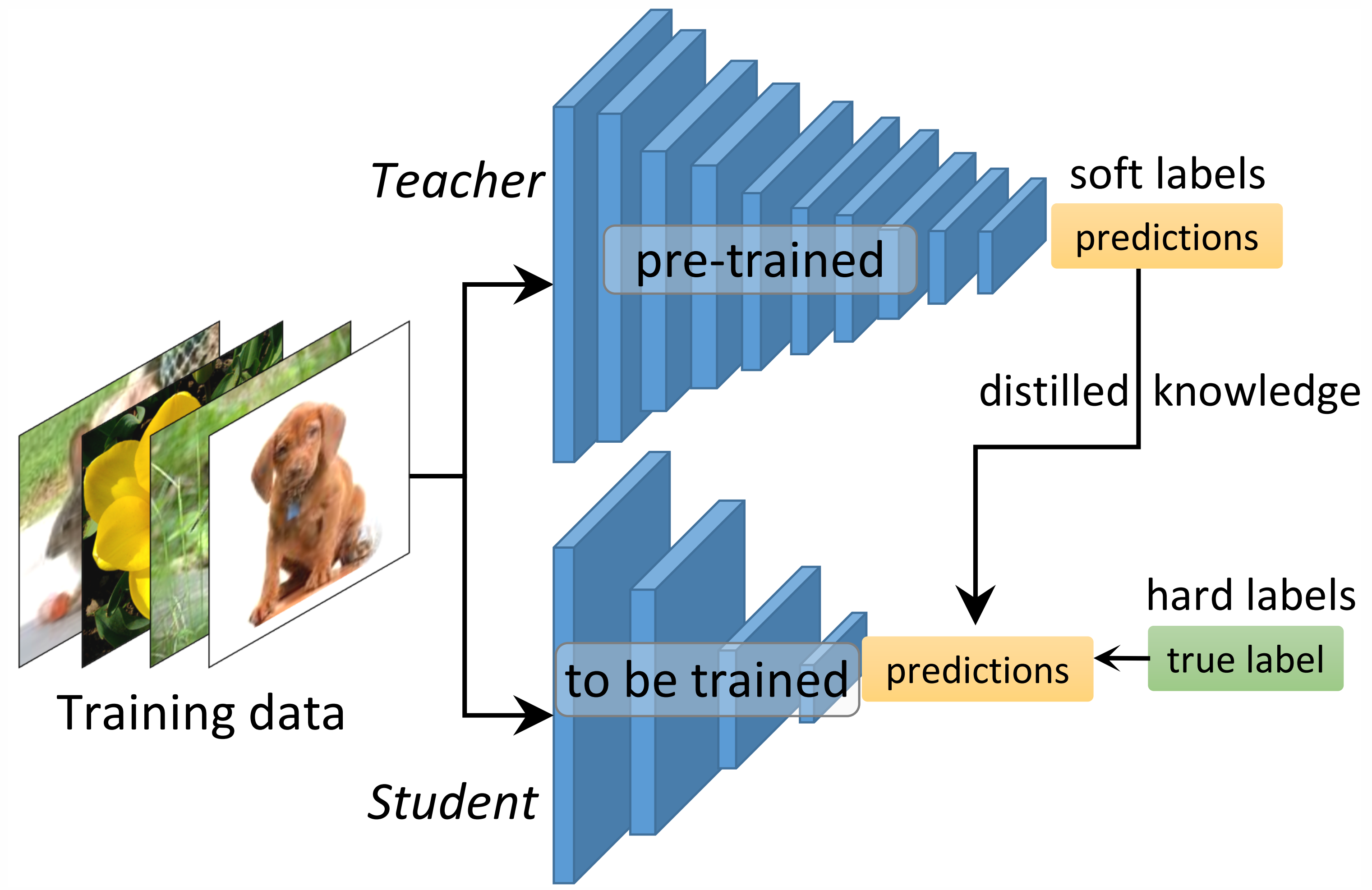
Linearly-separable



*Knowledge distillation (Hinton et al.) shows that a shallow student network can learn to imitate perfectly a deep teacher, even if it cannot learn directly from the data.

The effect of depth: changing the dynamics

But the main use of depth in modern DNN is to change the learning dynamics (more on this later), *not* to increase the expressiveness.



Knowledge distillation (Hinton et al., 2015) shows that a shallow student network can learn to imitate perfectly a deeper teacher, even if it cannot learn from the data equally well.

Picture from DeepVid, Wang et al., 2019

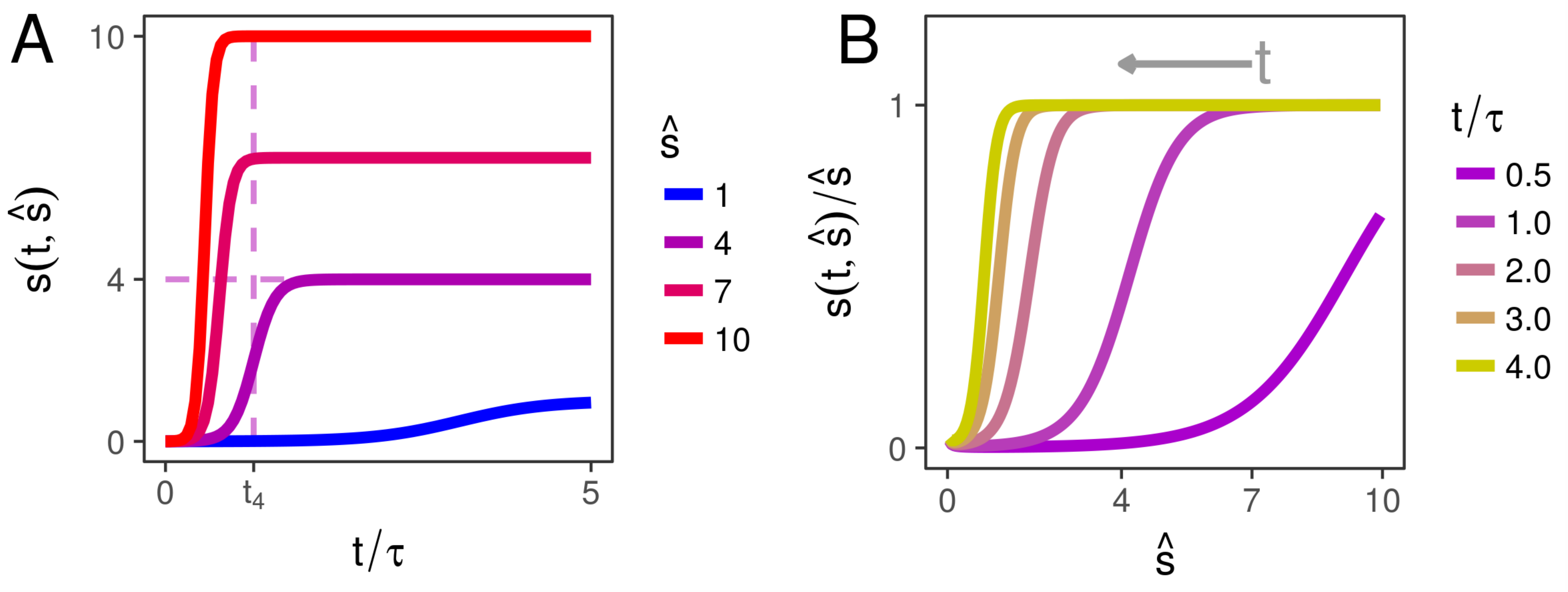
Deep parametrization makes linear networks non-linear

Consider a deep linear network:

$$f(x) = W_L W_{L-1} \dots W_0 \, x$$

While it still implements a simple linear function, the loss landscape is now non-convex and the SGD dynamics are much more complex.

Example: In a regression problem *y*=*Ax*, where *x* is gaussian, a deep linear network converges faster on components with the larger singular value.

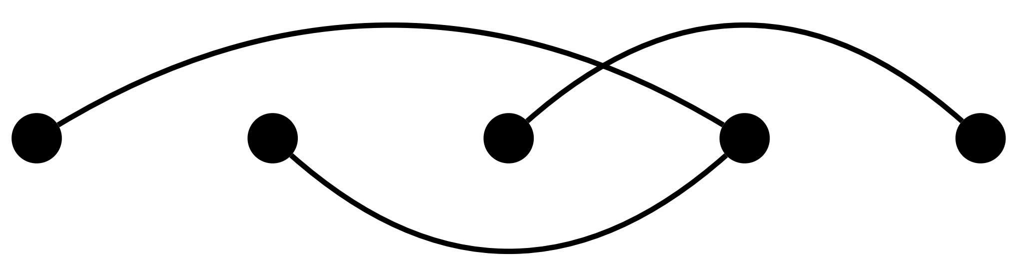


Saxe et al., Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2013

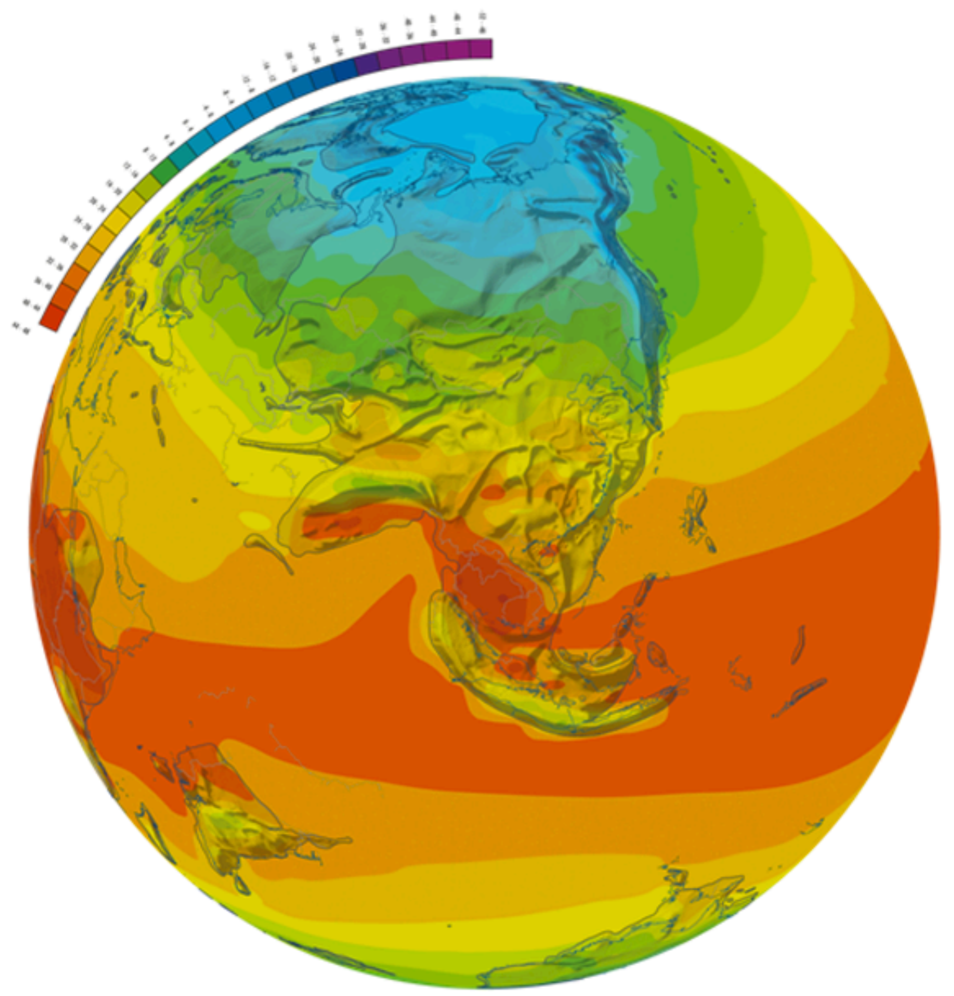
CNNs for non-image data

In general, we can construct DNNs tailored to some data by finding a group G that naturally acts on the data and using G-convolutions.

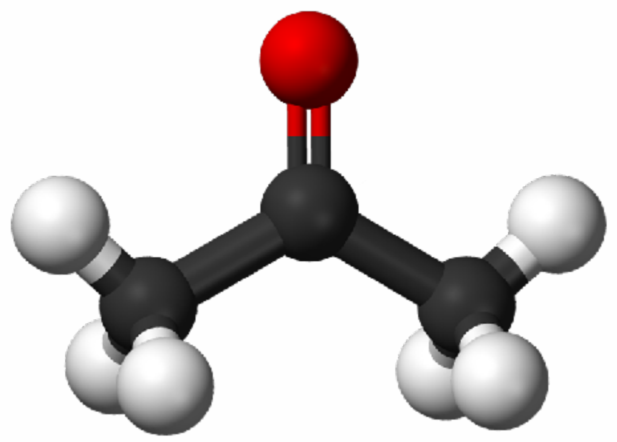
Example. The input is a set of n values $x = \{x_1, \ldots, x_n\}$. Since the output needs to be invariant to permutation of the elements, we can use π_n-convolutions.



Example. For weather forecast, we get measurements on a sphere (earth surface). We want the prediction to be $SO(2)$ invariant, use $SO(2)$-convolutions.



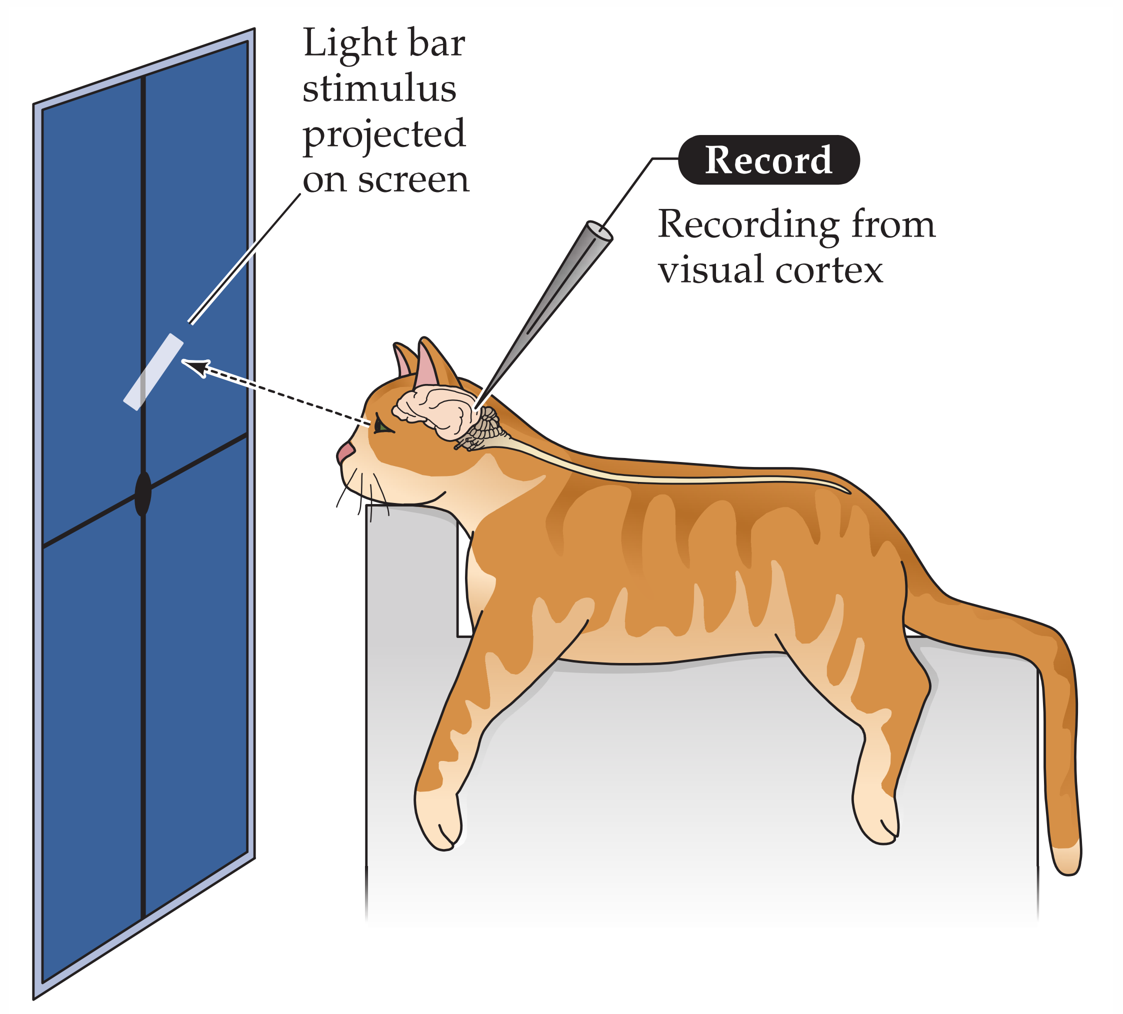
Example. Molecules, proteins, …

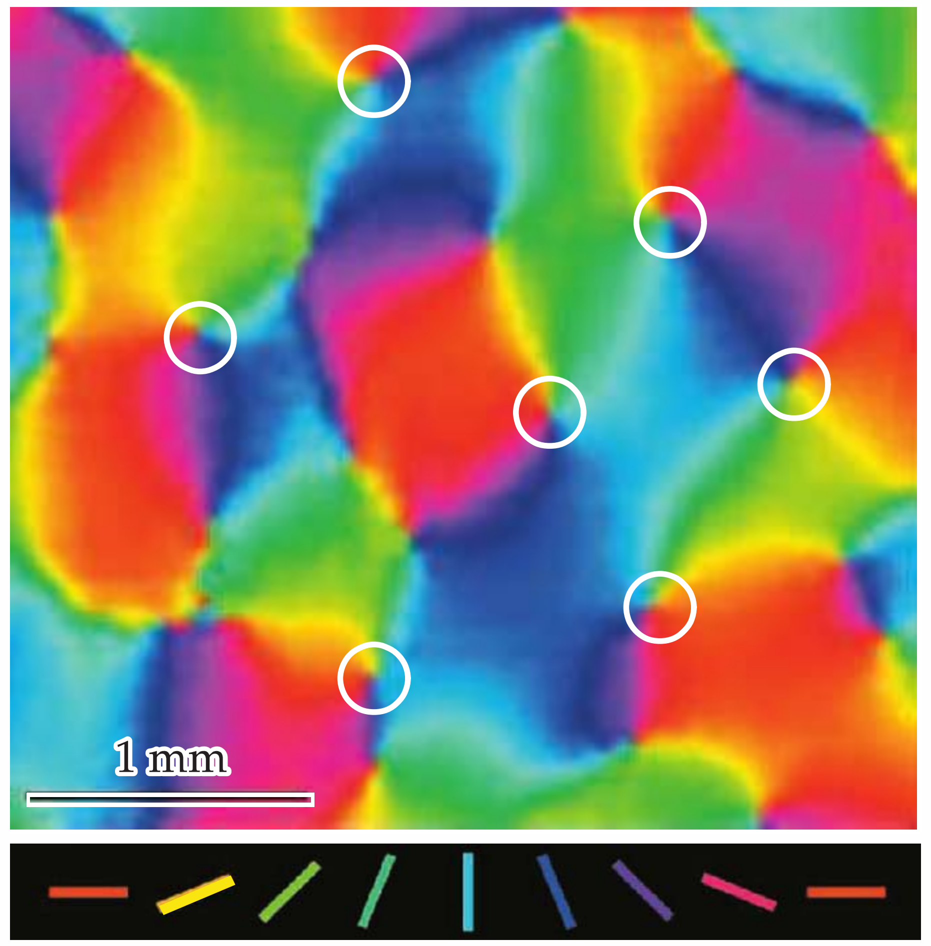


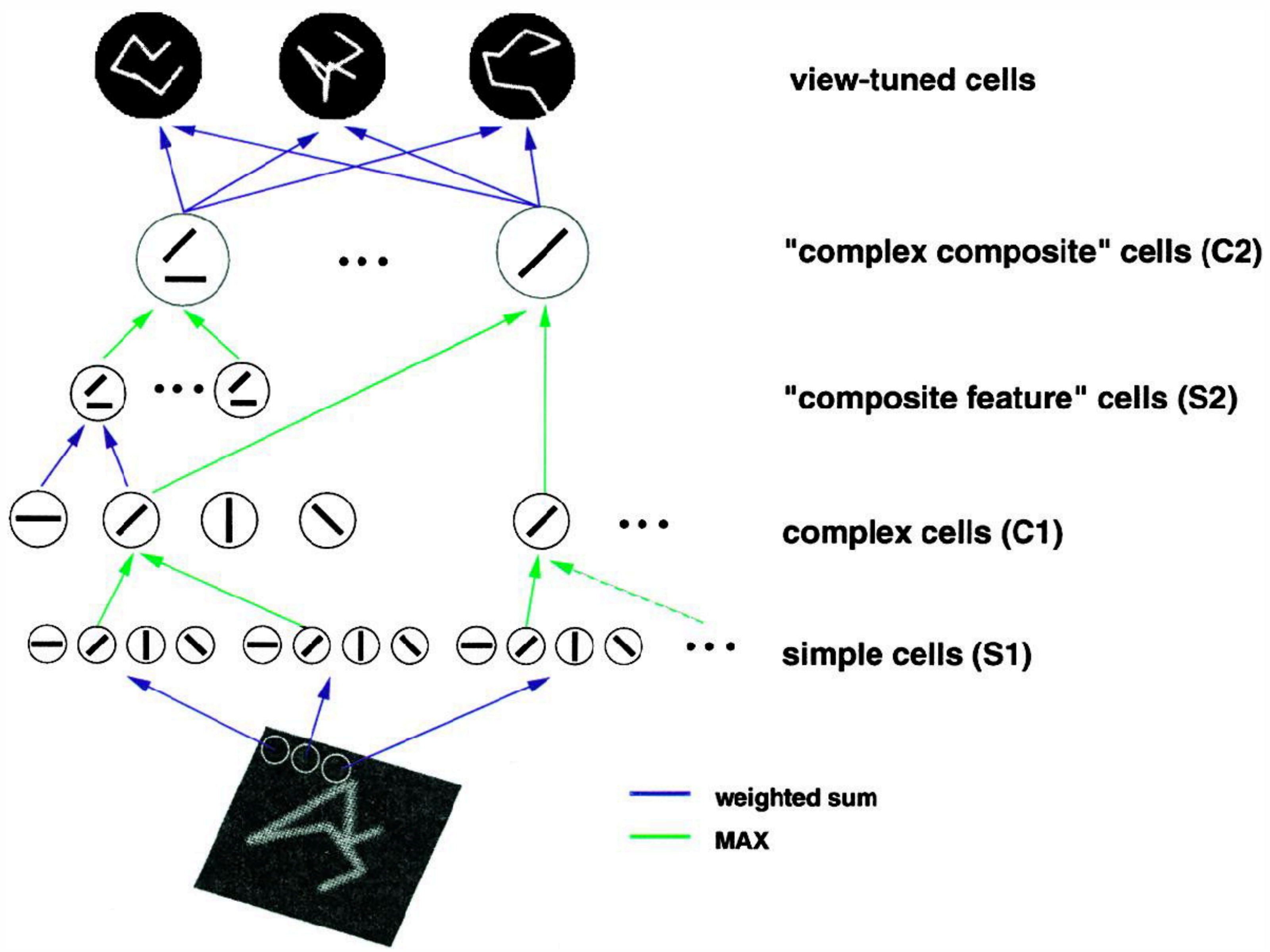
Why *neural* network?

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

By D. H. HUBEL and T. N. WIESEL







Pictures from *Neuroscience*, Purves et al., and *Are Cortical Models Really Bound by the "Binding Problem"?*, Riesenhuber and Poggio

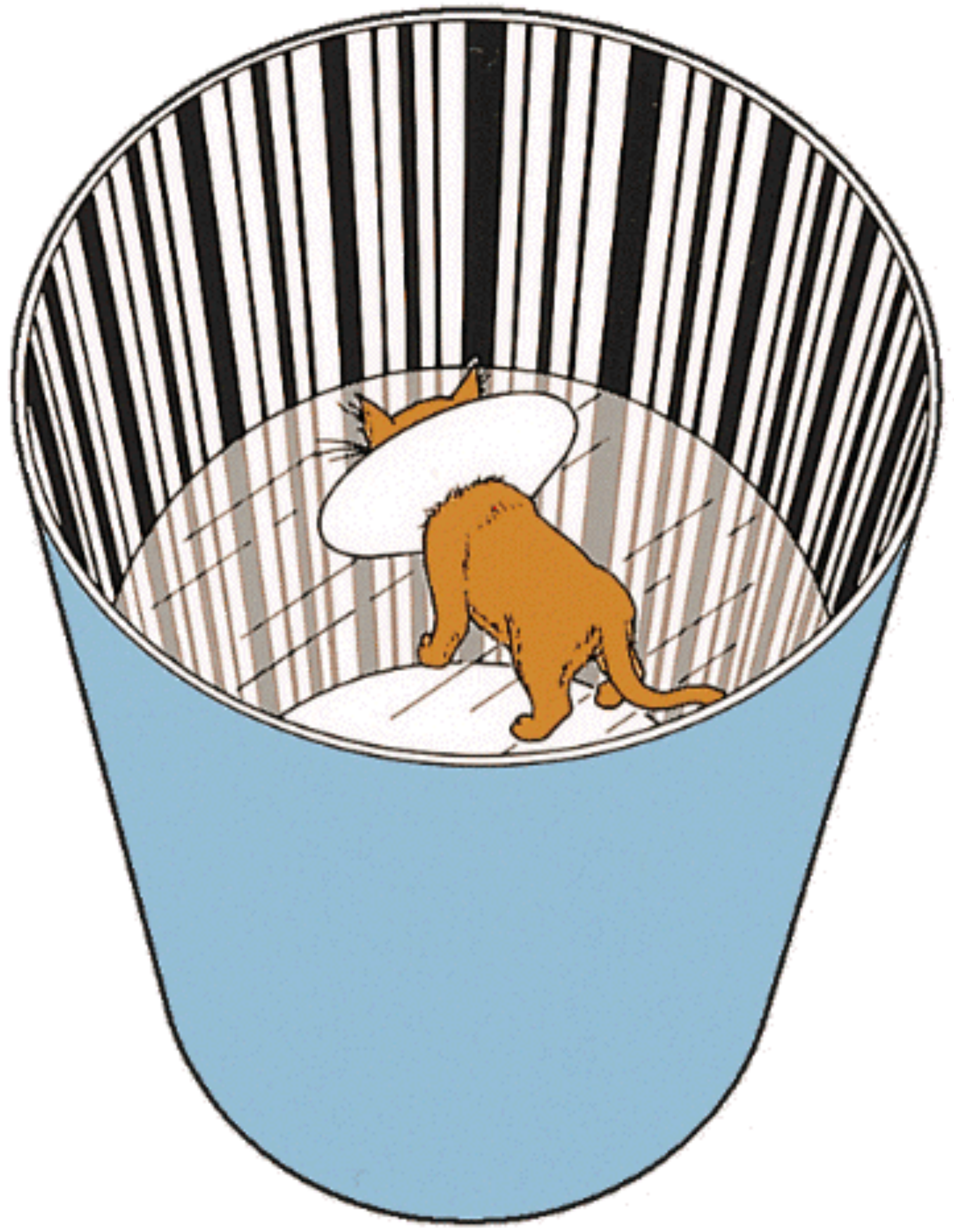
Why *neural* network?

Development of the Brain depends on the Visual Environment

COLIN BLAKEMORE
GRAHAME F. COOPER

The Physiological Laboratory,
University of Cambridge,
Cambridge CB2 3EG.

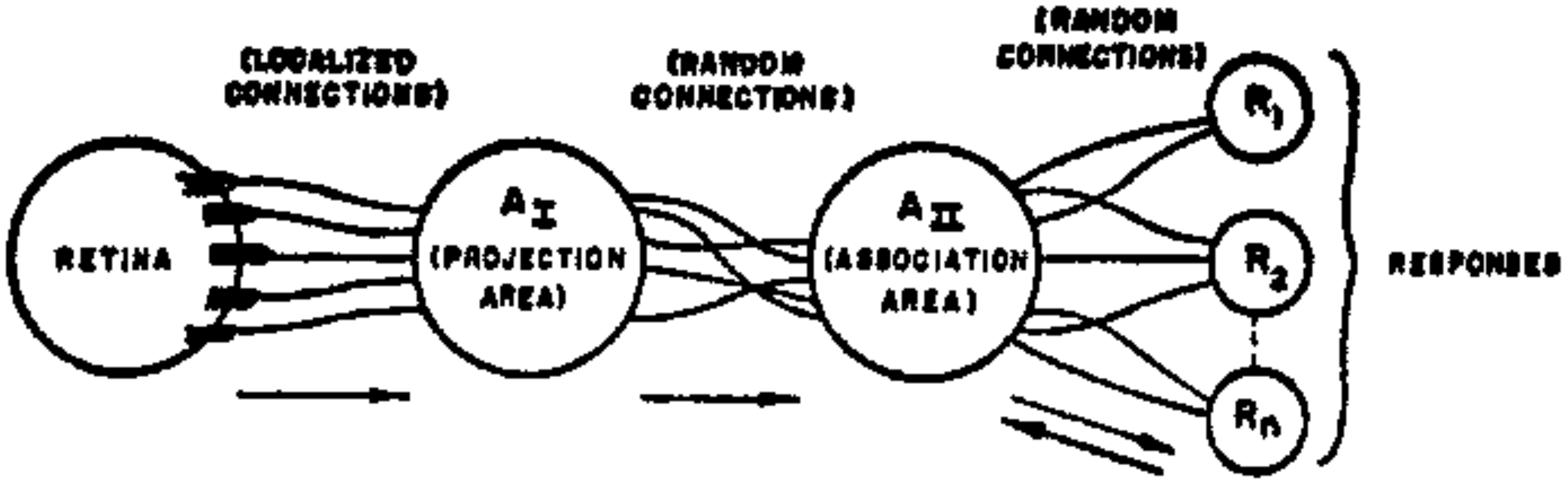
Received July 17, 1970.

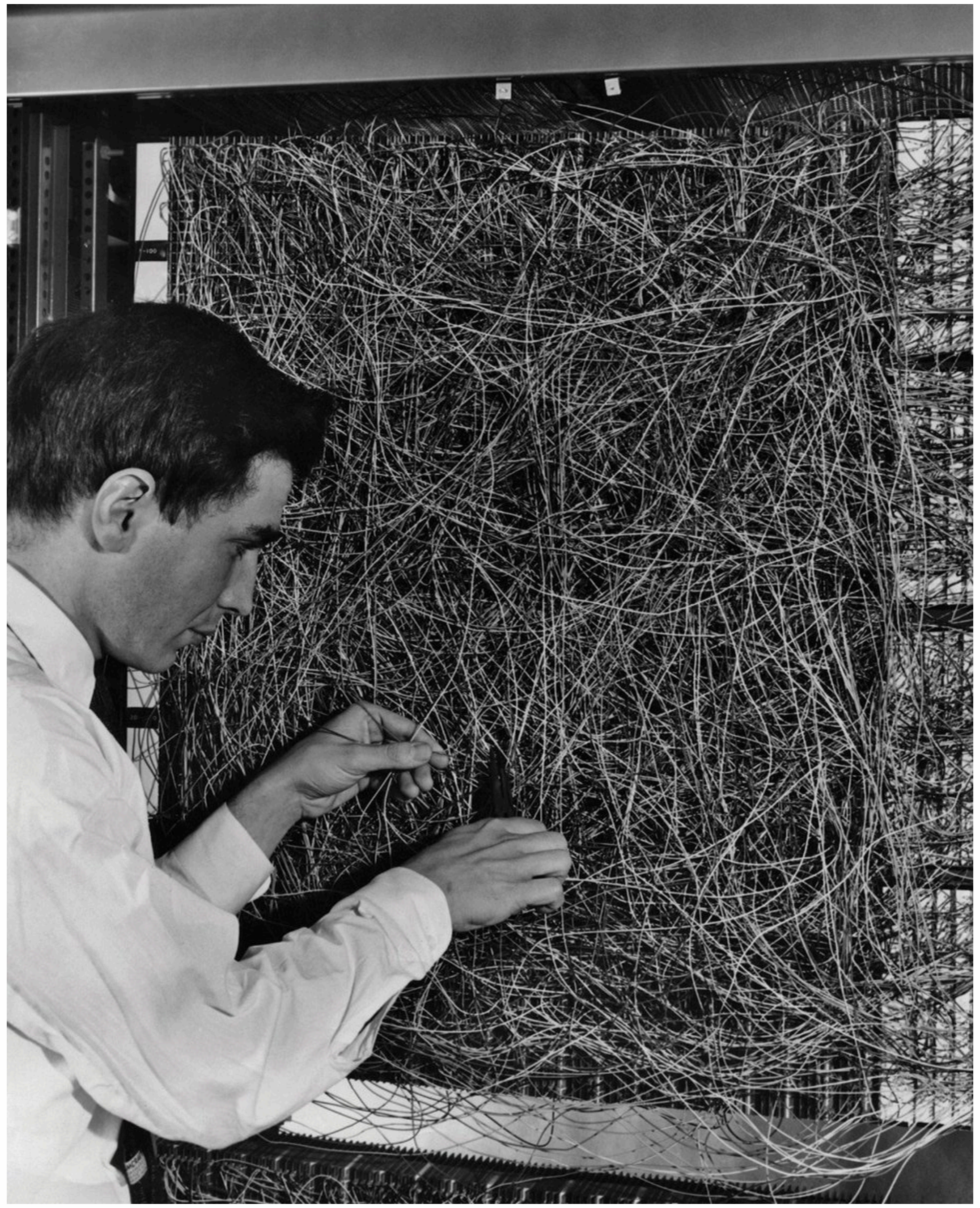


Why *neural* network?

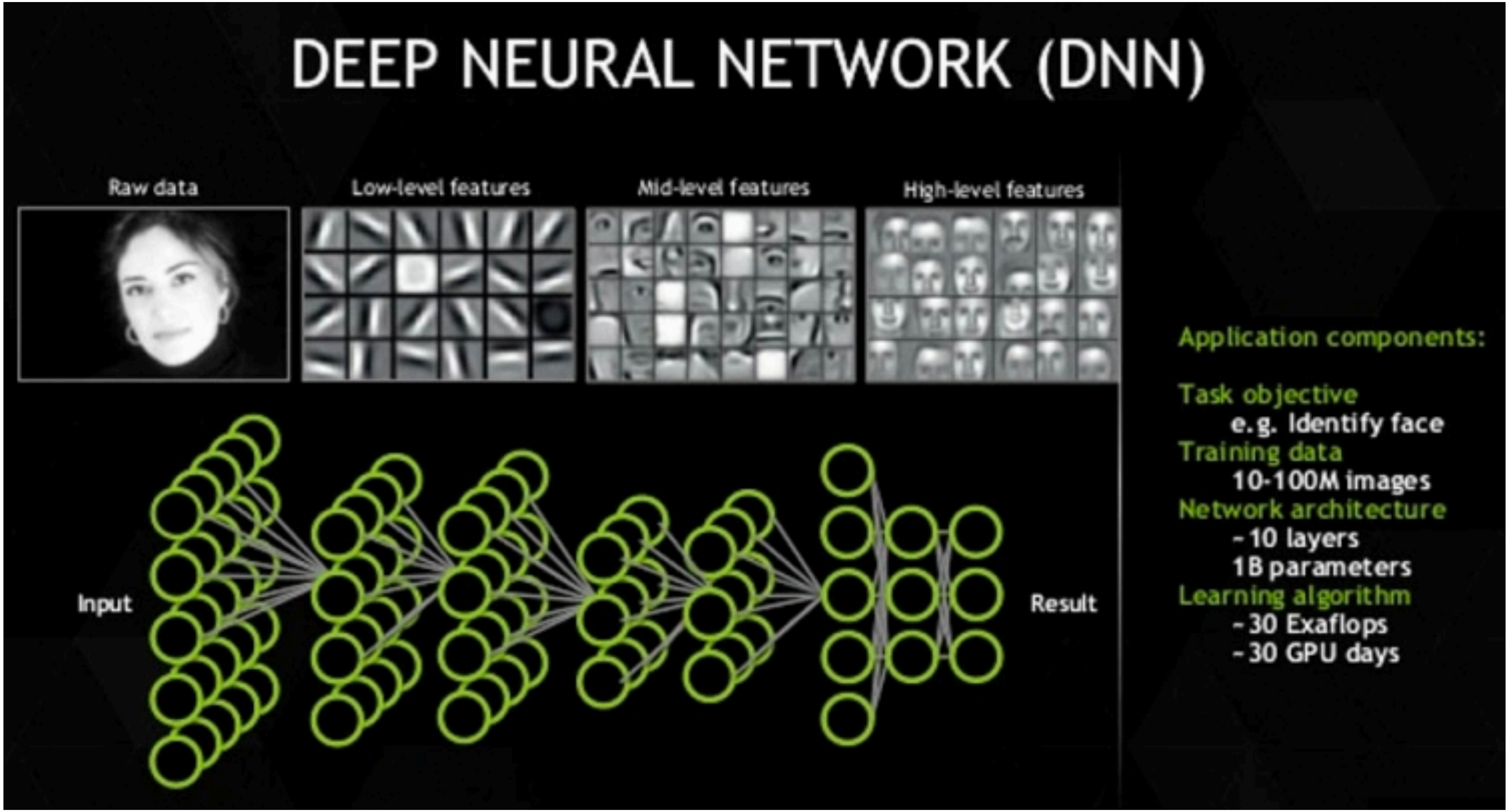
THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT



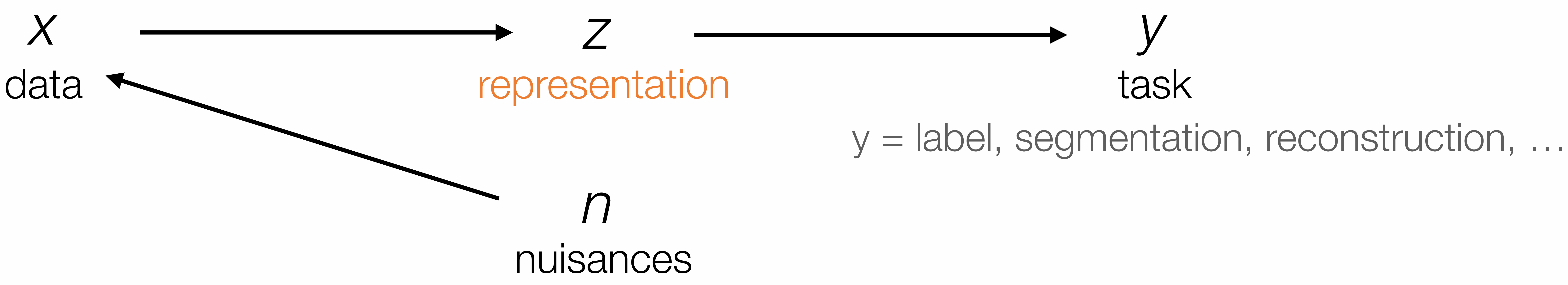


Features learned by a DNN



Lecture 2 — break

What is a representation?



Sufficient	$I(z;\, y) = I(x;\, y)$
Nuisance invariance	$n \perp\!\!\!\perp y \Rightarrow I(n;\, z) = 0$
Minimal	$I(x;\, z) =$ minimal
Compositional	Minimal component correlation?

A. and Soatto, Information Dropout: finding optimal representation through noisy computation, PAMI 2017

A Variational Principle for representation learning: The Information Bottleneck principle

A minimal sufficient representation is the solution to:

$$\begin{aligned} \text{minimize}_{p(z|x)} \quad & I(x;z) \\ \text{s.t.} \quad & H(y|z) = H(y|x) \end{aligned}$$

Information Bottleneck Lagrangian: (Tishby et al., 1999)

$$\mathcal{L} = \underbrace{H_{p,q}(y|z)}_{\text{cross-entropy}} + \underbrace{\beta I(z;x)}_{\text{regularizer}}$$

Invariant if and only if minimal

Definition. A representation z is minimal for the task y if it minimizes $I(z; x)$ among the sufficient representations.

Theorem (A., Soatto) (informal) Let z be a sufficient representation and n a nuisance. Then,

$$\underbrace{I(z; n)}_{\text{invariance}} \leq \underbrace{I(z; x)}_{\text{minimality}} - \underbrace{I(x; y)}_{\text{constant}}$$

Moreover, there exists a nuisance n for which equality holds.

Corollary: A representation is maximally invariant if and only if it is minimal

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

Compression without loss of *useful* information

Task Y = Is this the picture of a dog?

Less information $I(z; x)$ in the representation

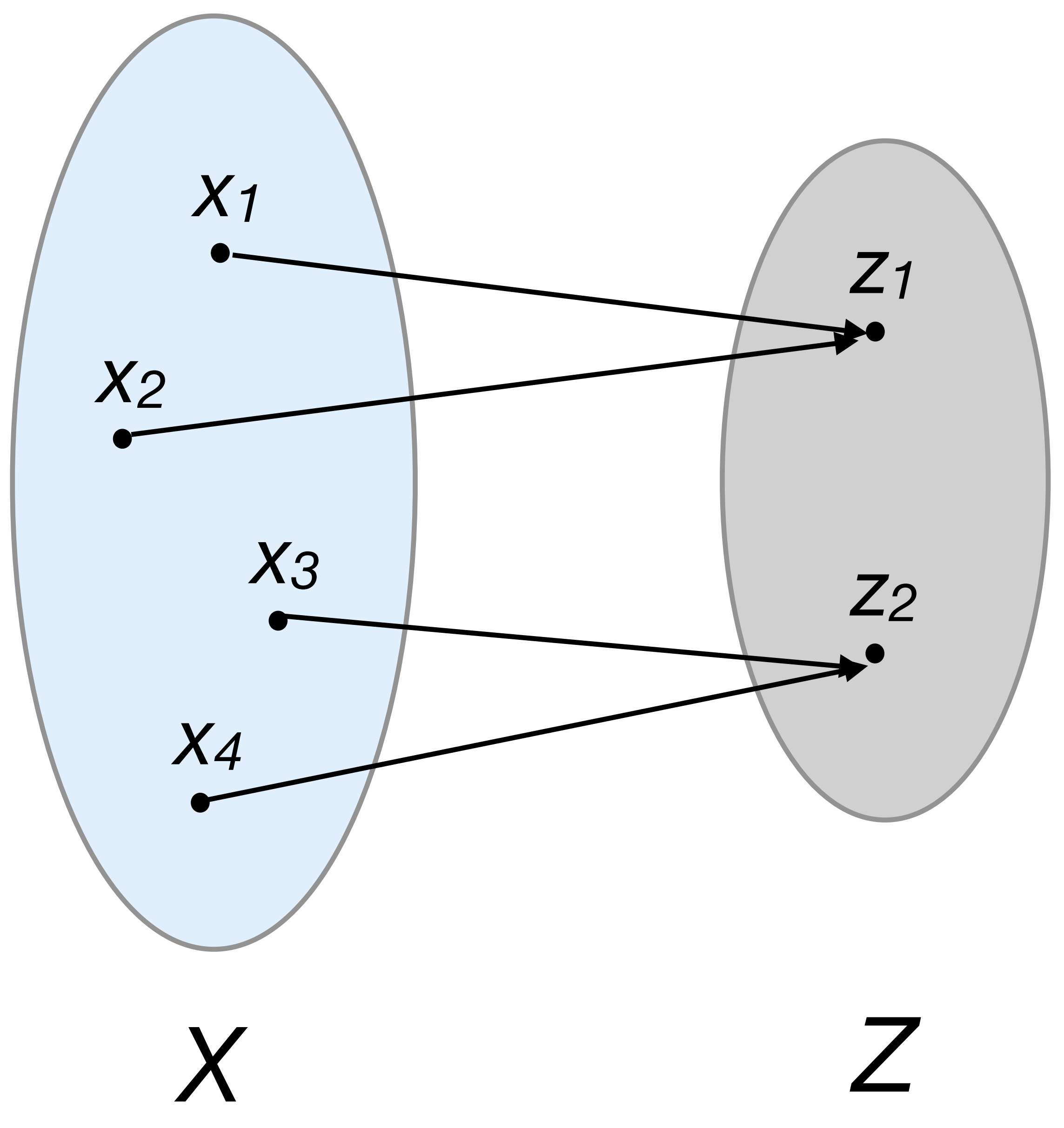
The task information $I(z; y)$ remains about the same

The IB Lagrangian $\mathscr{L}(\theta) = H_{p_\theta}(y \mid z) + \beta \, I_{p_\theta}(z; x)$ allows to interpolate between all the various representations by varying β.

Image source https://en.wikipedia.org/wiki/File:Terrier_mixed-breed_dog.jpg

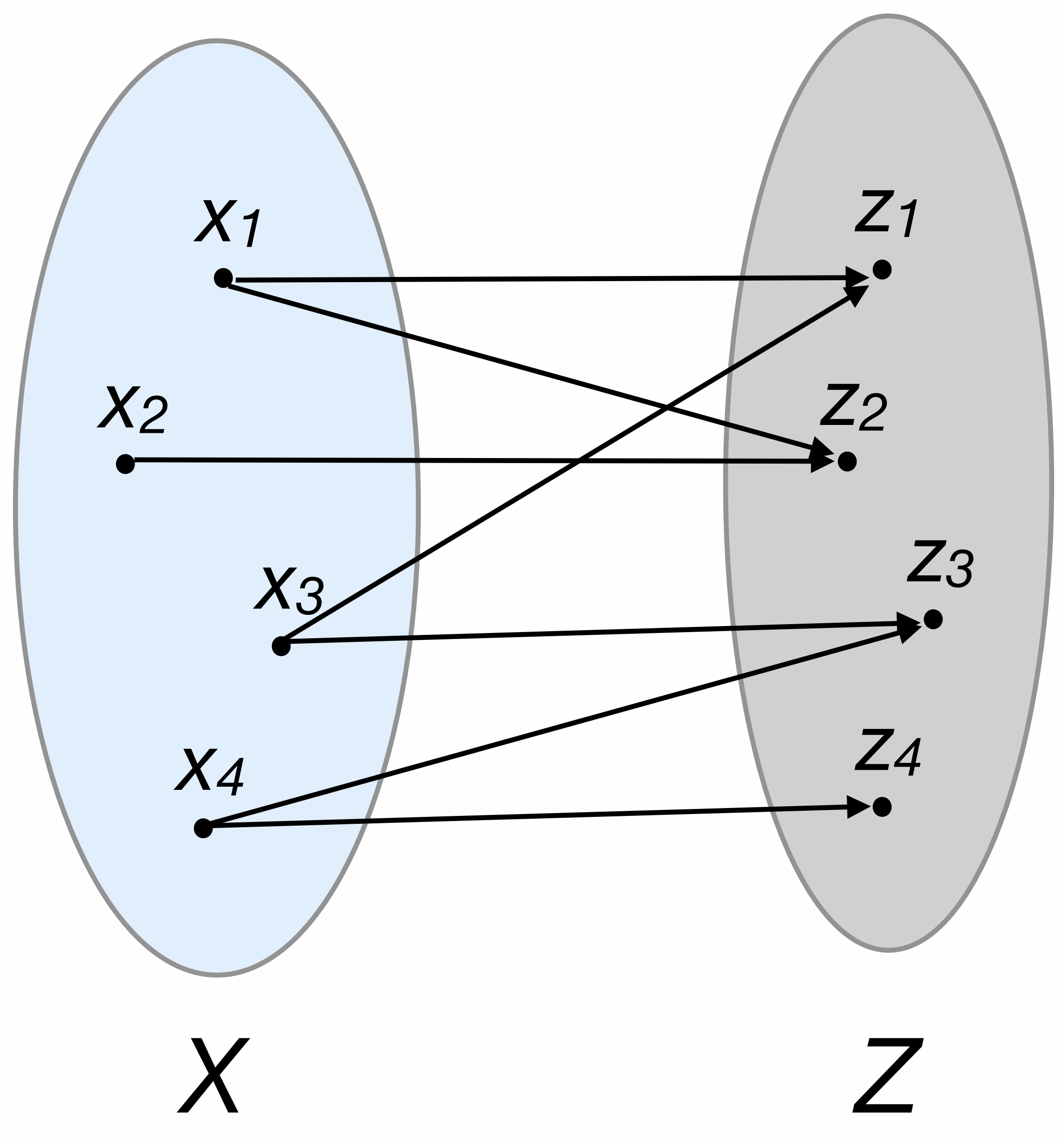
Compression in practice

Reduce the dimension



Examples: max-pooling, dimensionality reduction

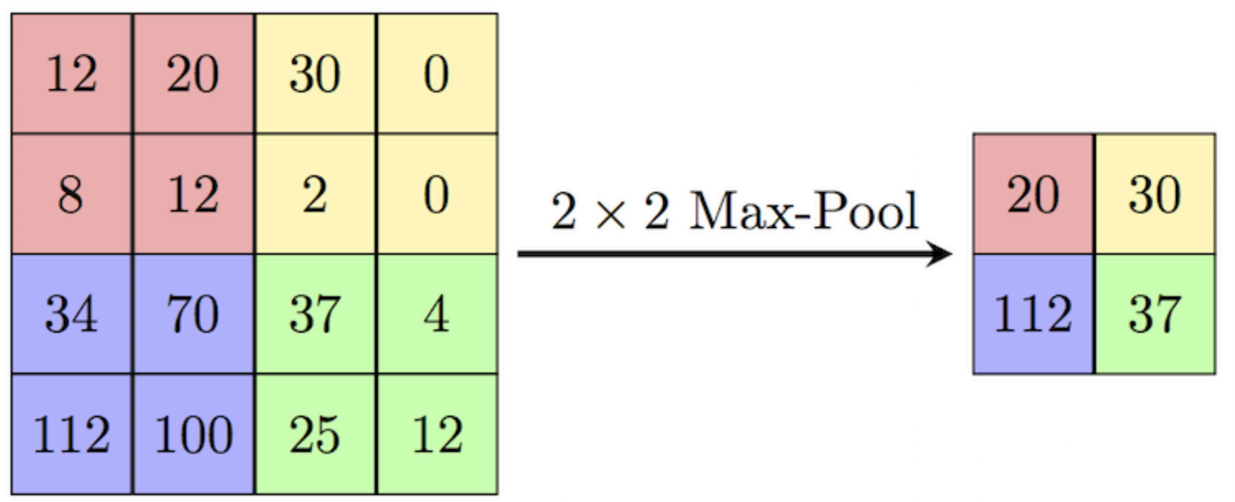
Increase dimension + Inject noise in the map



Examples: dropout, batch-normalization

MaxPooling: Reducing information by reducing the dimension

Downsample the spatial dimension by selecting only local maxima of the activations:



Nowadays replaced by more expensive (but better performing) strided convolutions.

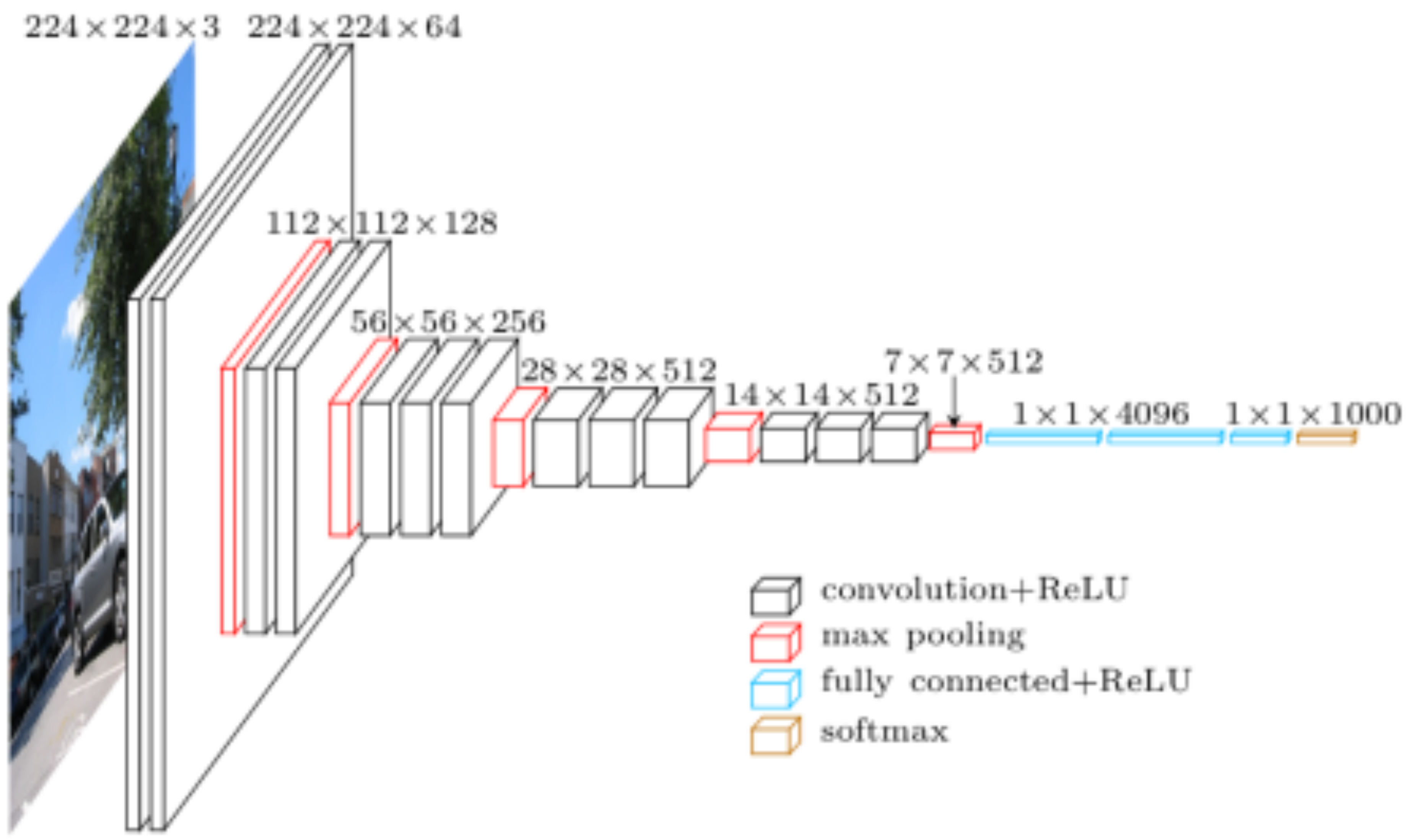


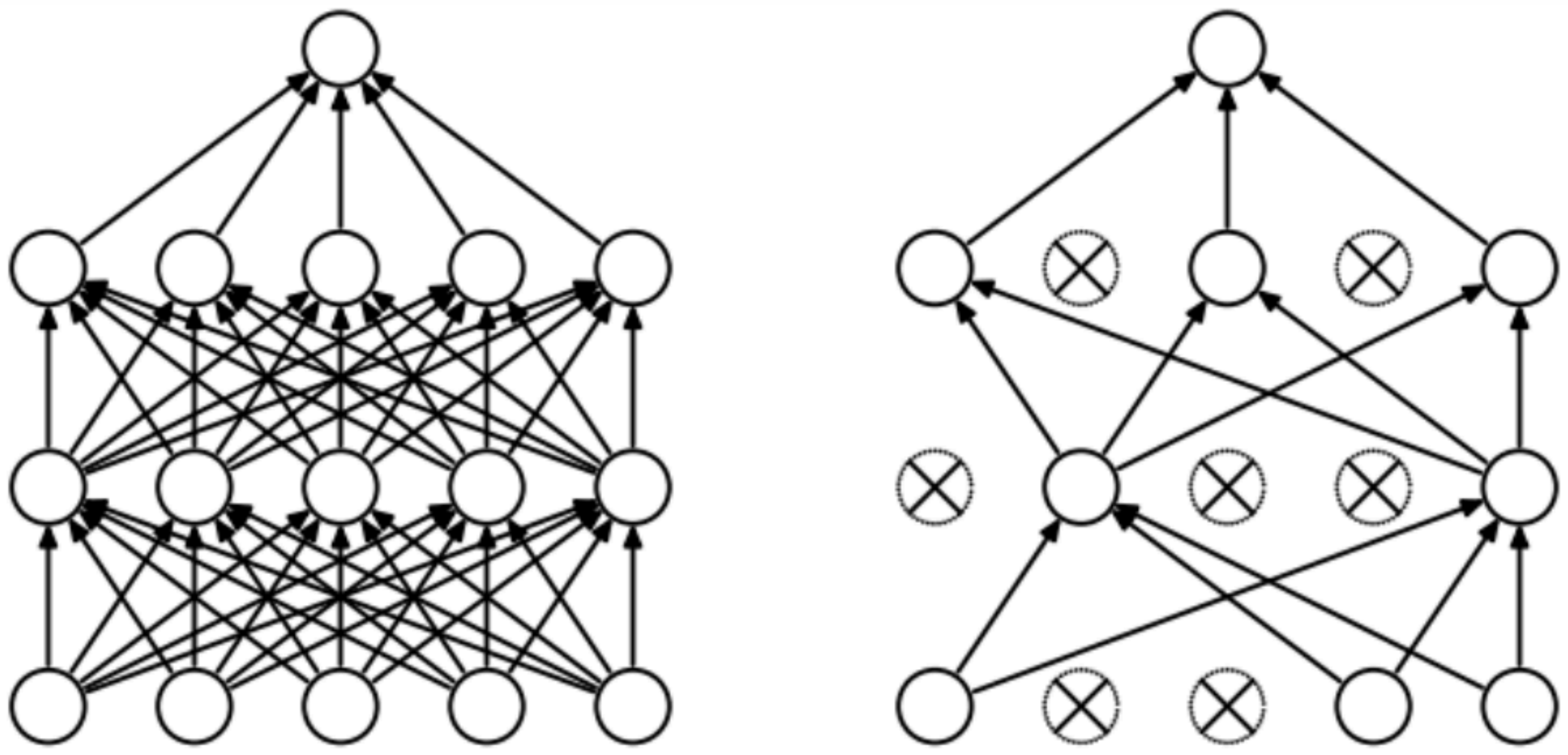
Image source https://neurohive.io/en/popular-networks/vgg16/

Dropout: Reducing information by adding noise

Introduce binary multiplicative noise in the activations:

$$z \rightarrow \frac{1}{p} z \odot \epsilon, \text{ where } \epsilon \sim \mathrm{Bernoulli}(p)$$

In practice disables random units during training:



(a) Standard Neural Net (b) After applying dropout.

Nowadays, batch normalization is used instead of dropout as it has a similar effect and performs much better.

Notebook

Back to the general Information Bottleneck

A minimal sufficient (and hence invariant) representation is the solution to:

$$\begin{aligned} \text{minimize}_{p(z|x)} \quad & I(x;z) \\ \text{s.t.} \quad & H(y|z) = H(y|x) \end{aligned}$$

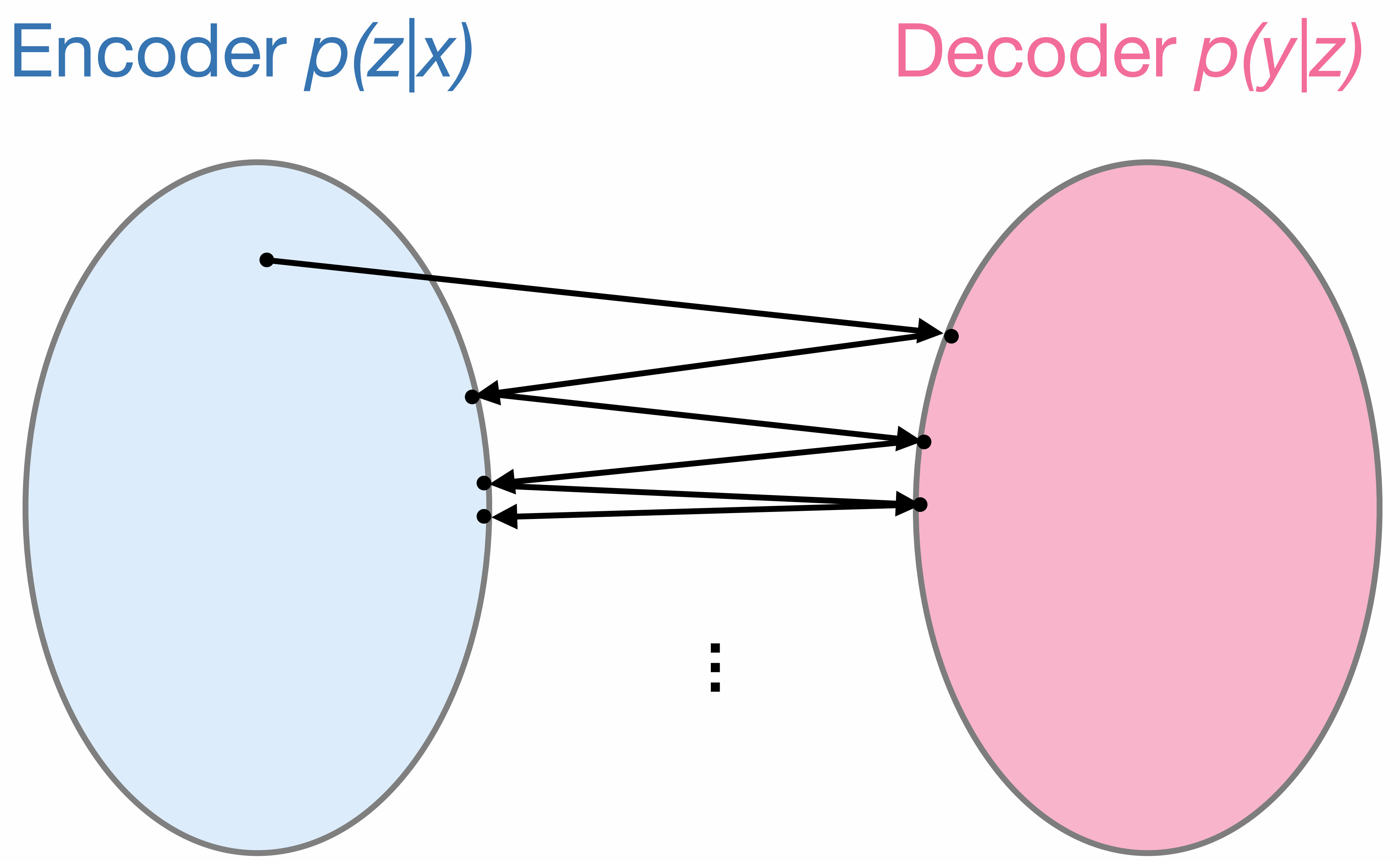
Information Bottleneck Lagrangian: (Tishby et al., 1999)

$$\mathcal{L} = \underbrace{H_{p,q}(y|z)}_{\text{cross-entropy}} + \underbrace{\beta I(z;x)}_{\text{regularizer}}$$

Blahut-Arimoto algorithm

In general, how do we minimize the IB Lagrangian $\mathscr{L}(\theta) = H_{p_\theta}(y \mid z) + \beta\, I_{p_\theta}(z; x)$ to find an optimal representation? We can use the following iterative algorithm:

$$
\begin{aligned}
p_t(z|x) &\leftarrow \frac{p_t(z)}{Z_t(x,\beta)} \exp(-1/\beta\, d(x,z)) \\
p_{t+1}(z) &\leftarrow \sum_x p(x) p_t(z|x) \\
p_{t+1}(y|z) &\leftarrow \sum_y p(y|x) p_t(x|z)
\end{aligned}
$$



Exploits the fact that the set of probability distributions is convex.

But what happens if *p(z|x)* is too large, or parametrized in a non-convex way?

Tishby et al,, The information bottleneck method, 2000

Minimizing the information by adding noise

How do we minimize $\mathscr{L}(\theta)$ when $p_\theta(z\,|\,x)$ is complex, e.g., computed by a DNN)?

Problem: the marginal distribution is too complex to compute

$$\mathscr{L}(\theta) = H_{p_\theta}(y\,|\,x) + \beta\, I(x;z)$$
$$= H_{p_\theta}(y\,|\,x) + \beta\, \mathbb{E}_x\Big[\ \mathrm{KL}\big(p_\theta(z\,|\,x)\,\|\,p_\theta(z)\big)\ \Big]$$

Lemma. $I(z;x) \le \mathbb{E}_x\big[\mathrm{KL}\big(p(z\,|\,x)\,\|\,q(z)\big)\big]$ for any *q(z)* and is equal if and only if *q(z)* = *p(z)*.

Using this:

$$\mathscr{L}(\theta) \le H_{p_\theta}(y\,|\,x) + \beta\, \mathbb{E}_x\Big[\ \mathrm{KL}\big(p_\theta(z\,|\,x)\|q_\phi(z)\big)\ \Big]$$
$$=: \mathscr{L}(\theta,\phi)$$

Hence: $\min_{\theta} \mathscr{L}(\theta) = \min_{\theta,\phi} \mathscr{L}(\theta,\phi)$, and the latter minimization problem is simpler.

A. and Soatto, Information Dropout: finding optimal representations by adding noise, 2016

Example of implementation

Learning a minimal sufficient representation z of the data.

$$\mathscr{L}(\theta,\phi) = H_{p_\theta}(y\,|\,x) + \beta\,\mathbb{E}_x\Big[\ \mathrm{KL}\Big(p_\theta(z\,|\,x) \| q_\phi(z)\Big)\ \Big]$$

Algorithm:

1. Choose a simple family of distributions $p_\theta(z\,|\,x)$ and $q_\phi(z)$, for example:

   $p_\theta(z\,|\,x) \sim N(f_\theta(x), \Sigma)$ and $q_\phi(z) \sim N(\mu_\phi, \Sigma_\phi)$

   Where $f_\theta(x)$ can be implemented by a DNN.

2. Train the network to minimize:

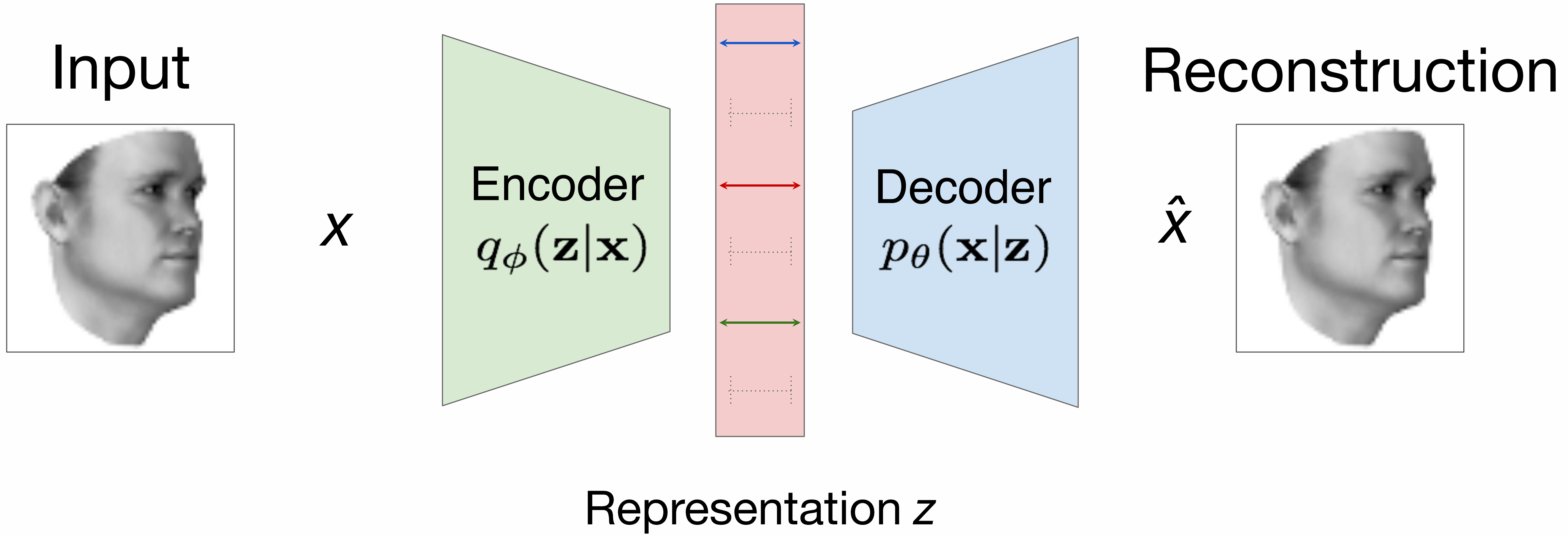
$$\mathscr{L}(\theta,\phi) = H_{p_\theta}(y\,|\,x) + \beta\,\mathbb{E}_x\Big[\ \mathrm{KL}\Big(N(\mu_\theta(x), \Sigma_\theta(x)) \,\|\, N(\mu_\phi, \Sigma_\phi)\Big)\ \Big]$$

$$= H_{p_\theta}(y\,|\,x) + \beta\,\mathbb{E}_x[(f(x) - \mu_\phi)^T \Sigma_\phi^{-1} (f(x) - \mu_\phi) + \mathrm{tr}(\Sigma/\Sigma_\phi - I) - \log \Sigma/\Sigma_\phi]$$

3. This can be seen equivalently as minimizing the loss with a noisy representation $z = f(z) + \epsilon$, $\epsilon \sim N(0, \Sigma)$ instead of with a deterministic representation as usual.

Example: Variational Auto-Encoders

Task: Train a network to encode and decode the input, while minimizing both the reconstruction error and the information *I(z; x)* used to encode it.

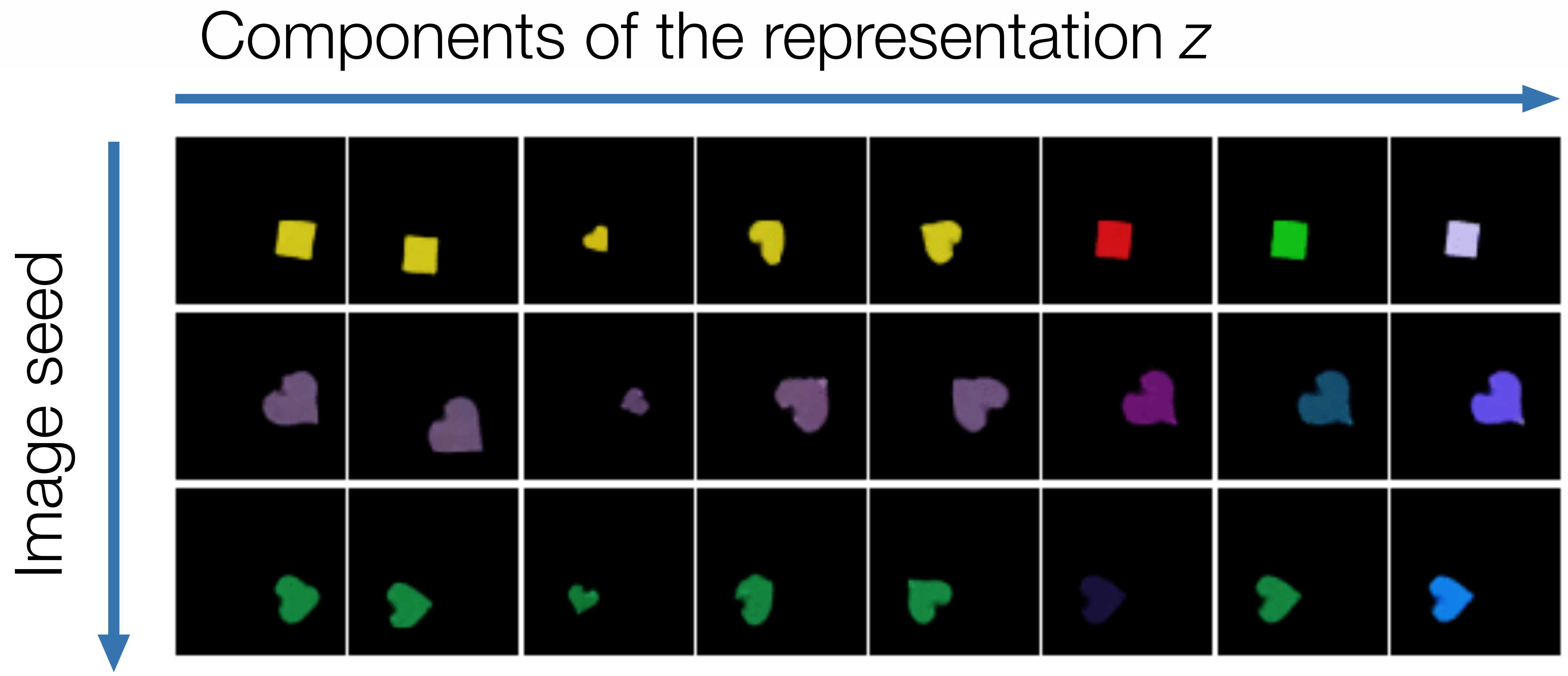


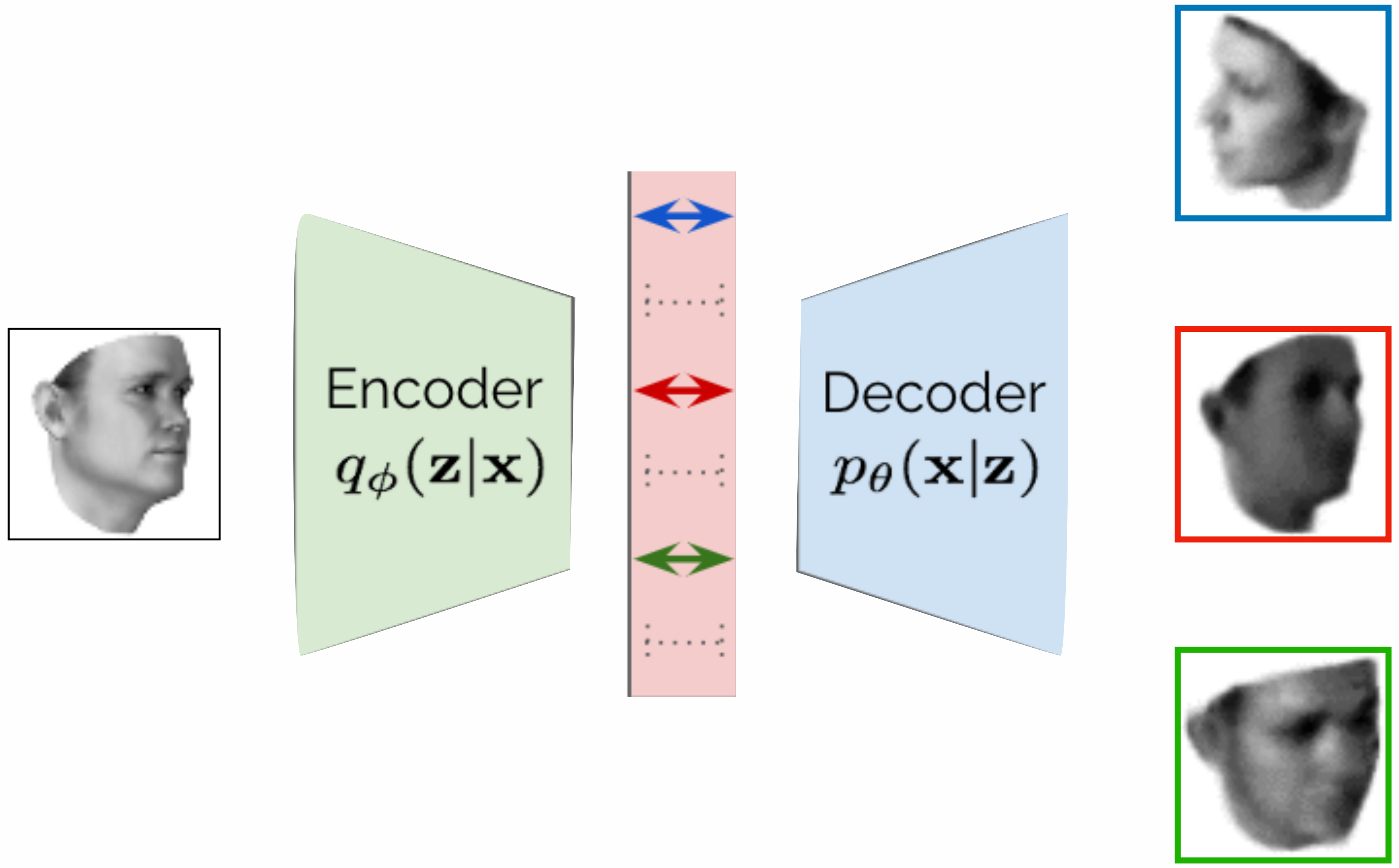
Minimize *I(z; x)*

Minimize $H(x \mid \hat{x})$

Example: Variational Auto-Encoders

Each component of the learned representation corresponds to a different semantic factor.





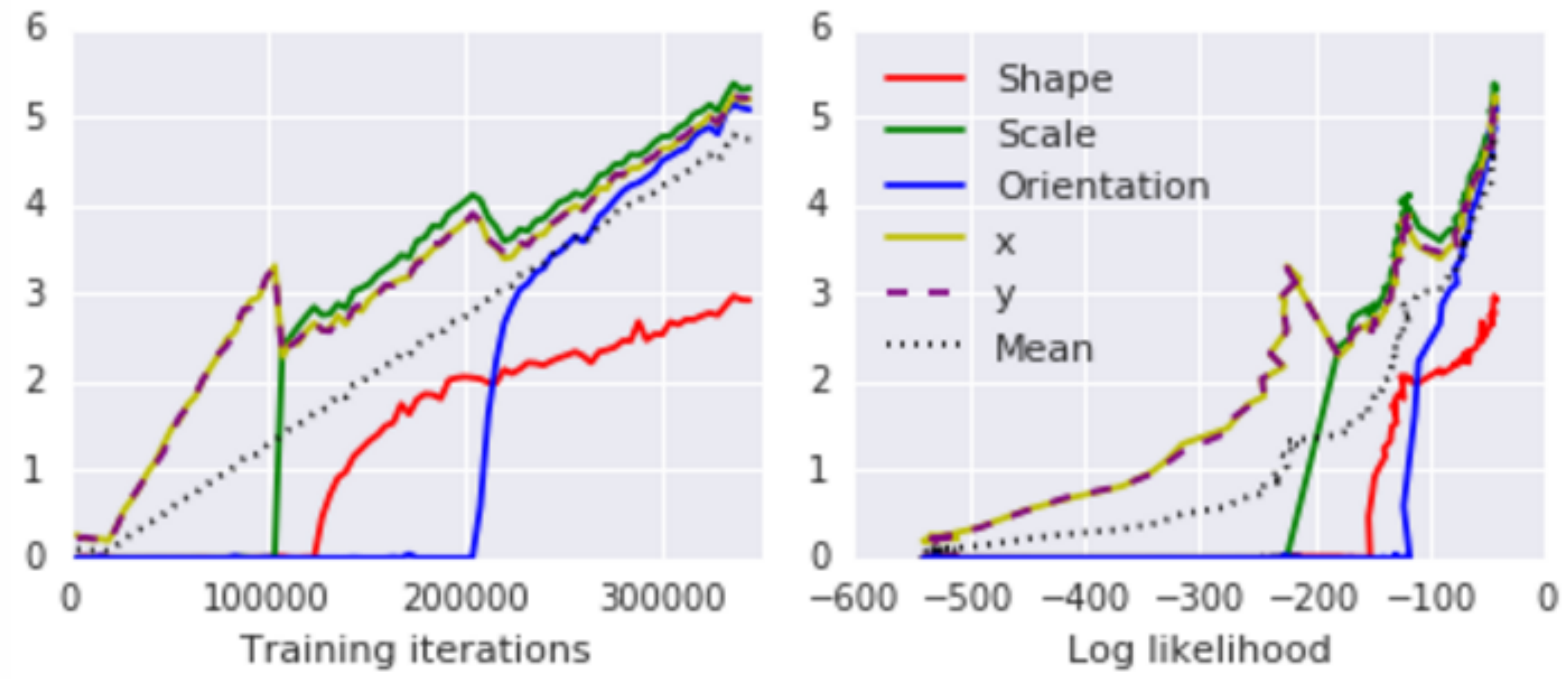
Higgins et al., β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, 2017
Burgess et al., Understanding Disentangling in beta-VAE" 2017

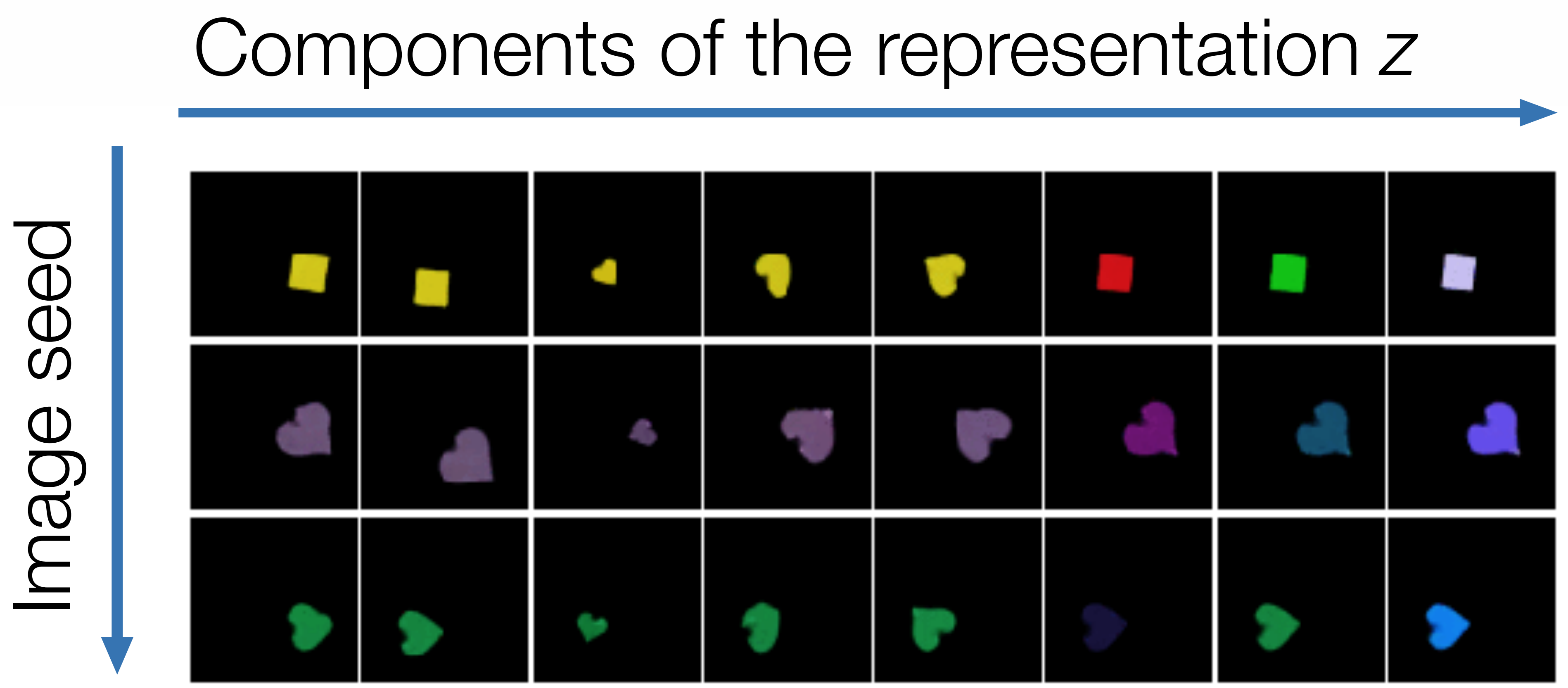
Pictures courtesy of Higgins et al., Burgess et al.

Learning disentangled representations

Start with very high β and slowly decrease during training.

Beginning: Very strict bottleneck, only encode most important factor
End: Very large bottleneck, encode all remaining factors



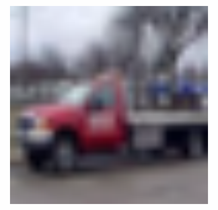
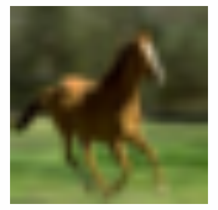


Think of it as a non-linear PCA, where *training time* disentangles the factors.

Higgins et al., β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, 2017
Burgess et al., Understanding Disentangling in beta-VAE" 2017

The catch

What if we just represent an image by its index in the training set (or by a unique hash)?

x		*z*		*y*
24,576 bits		*16 bits*		*4 bits*
	→	0000000000000000	→	0100
	→	0000000000000001	→	0001
	→	0000000000000010	→	0010
	→	0000000000000011	→	0101

It is a sufficient representation and it is close to minimal.

This Information Bottleneck is wishful thinking

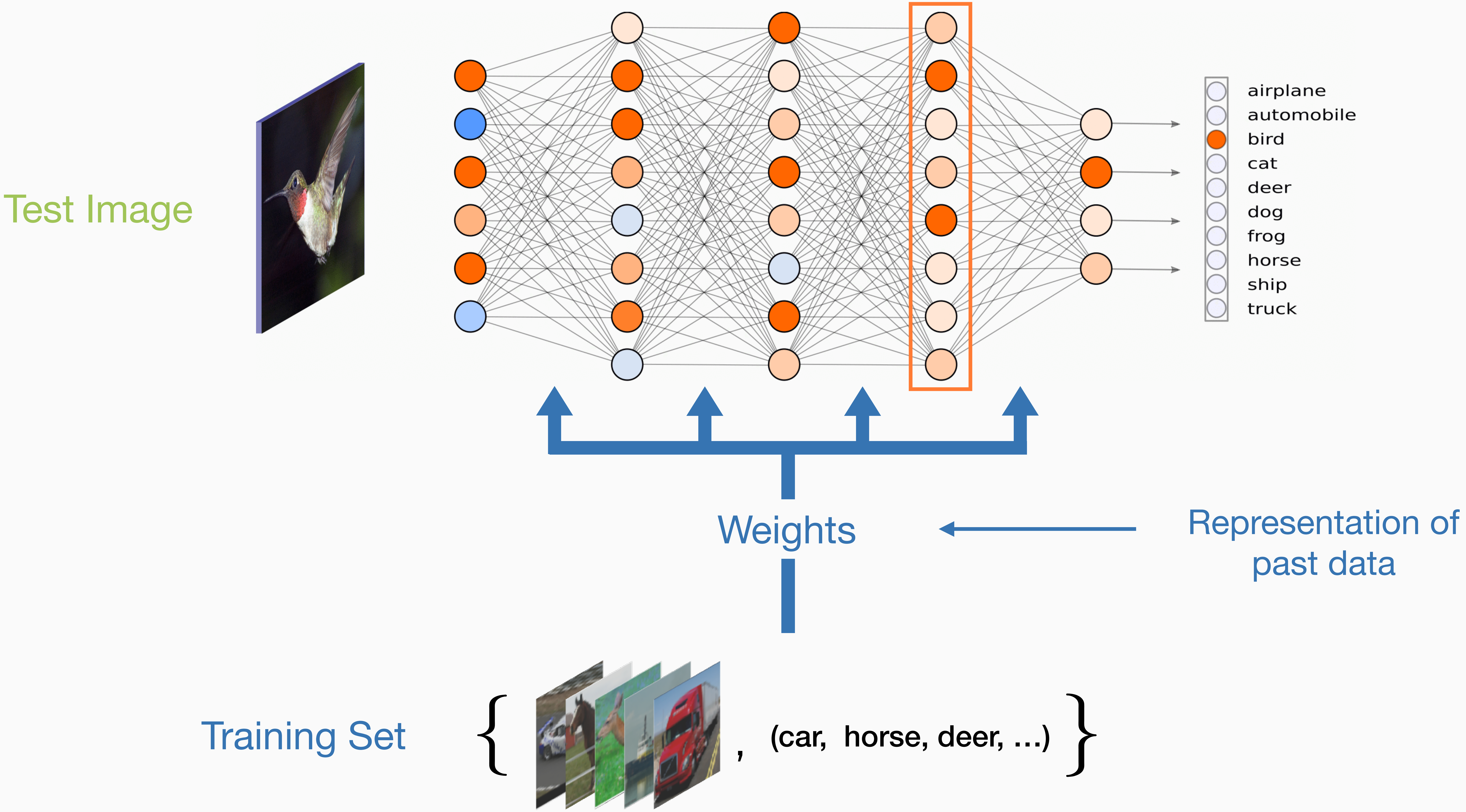
The IB is a statement of desire for future data we do not have:

$$\min_{q(z|x)} \mathcal{L} = H_{p,q}(y|z) + \beta\, I(z; x)$$

What we have is the data collected in the past.

What is the best way to use the past data in view of future tasks?

Lecture 3


Test Image
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Weights
Representation of past data
Training Set
{ , (car, horse, deer, ...) }

Can we separate structural information from noise?

Examples:

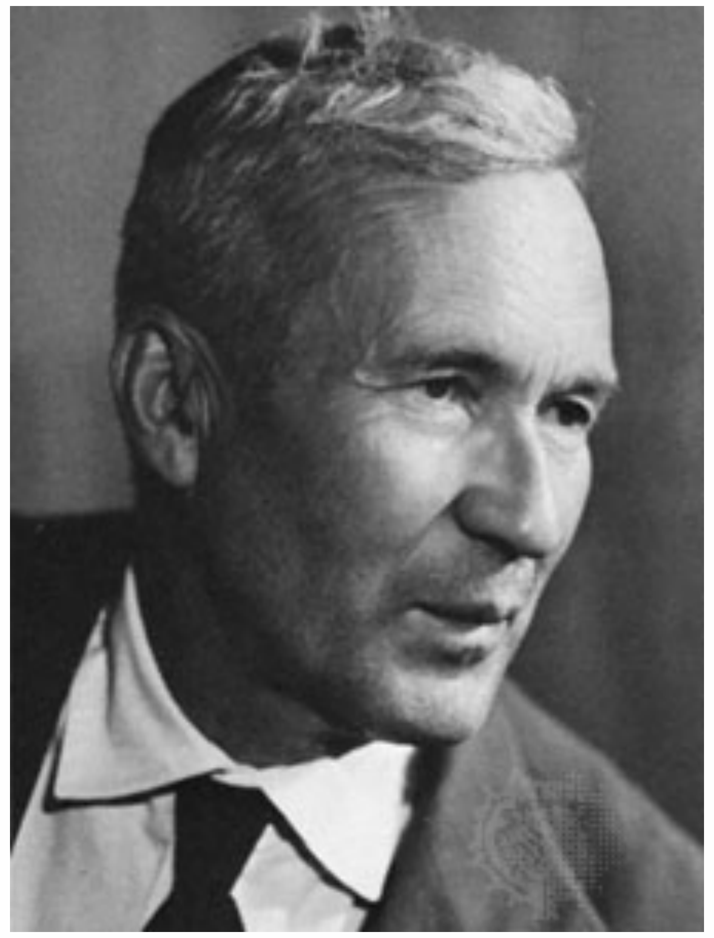
$$x = 110101111111011111101101110\ldots$$

x_i ~ Bernoulli(p=0.8), has entropy $H(x) = N \log(p)$. But the only “structural information” is that $p = 0.8$, the rest is randomness.

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

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:

$$S_x(t) = \min_{K(p)<t} -\log p(x)$$

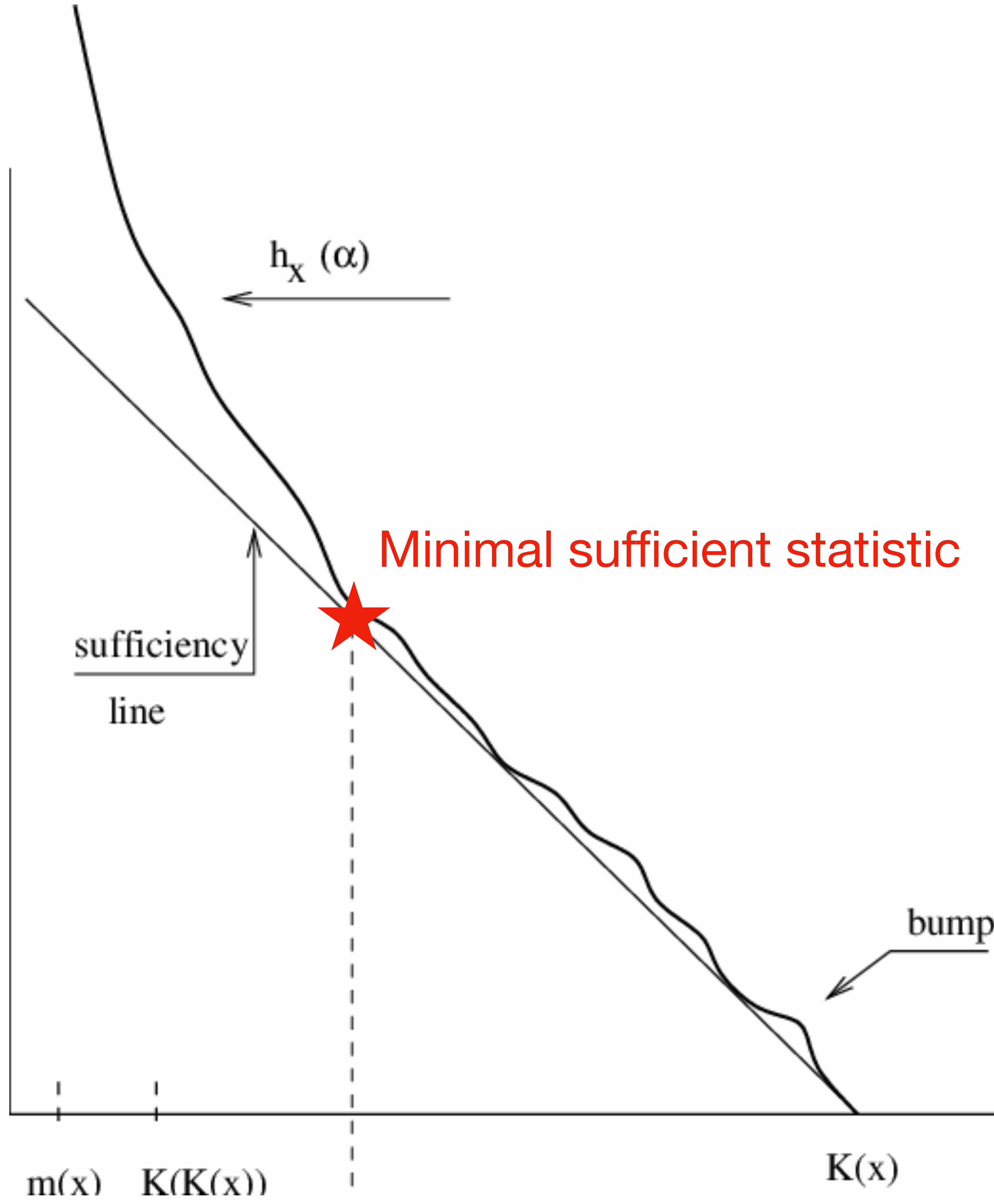
Cost of encoding the model

Cost of encoding the data using the model

Extreme cases:

$$p(z) = \mathrm{Unif}(z) \quad \Rightarrow \quad K(p) = 1 \text{ and } \log p(x) = N \log c$$

$$p(z) = \delta_x(z) \quad \Rightarrow \quad K(p) = K(x) \text{ and } \log p(x) = 0$$



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

The Kolmogorov Structure of a Task

How can we define the structure of a task?

Let $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^N$ be a dataset. For any $p_\theta(y\,|\,x)$ define $L(\mathscr{D}; p_\theta) = \sum_{i=1}^N p_\theta(y_i\,|\,x_i)$.

The structure function of the dataset $\mathscr{D}$ is defined by:

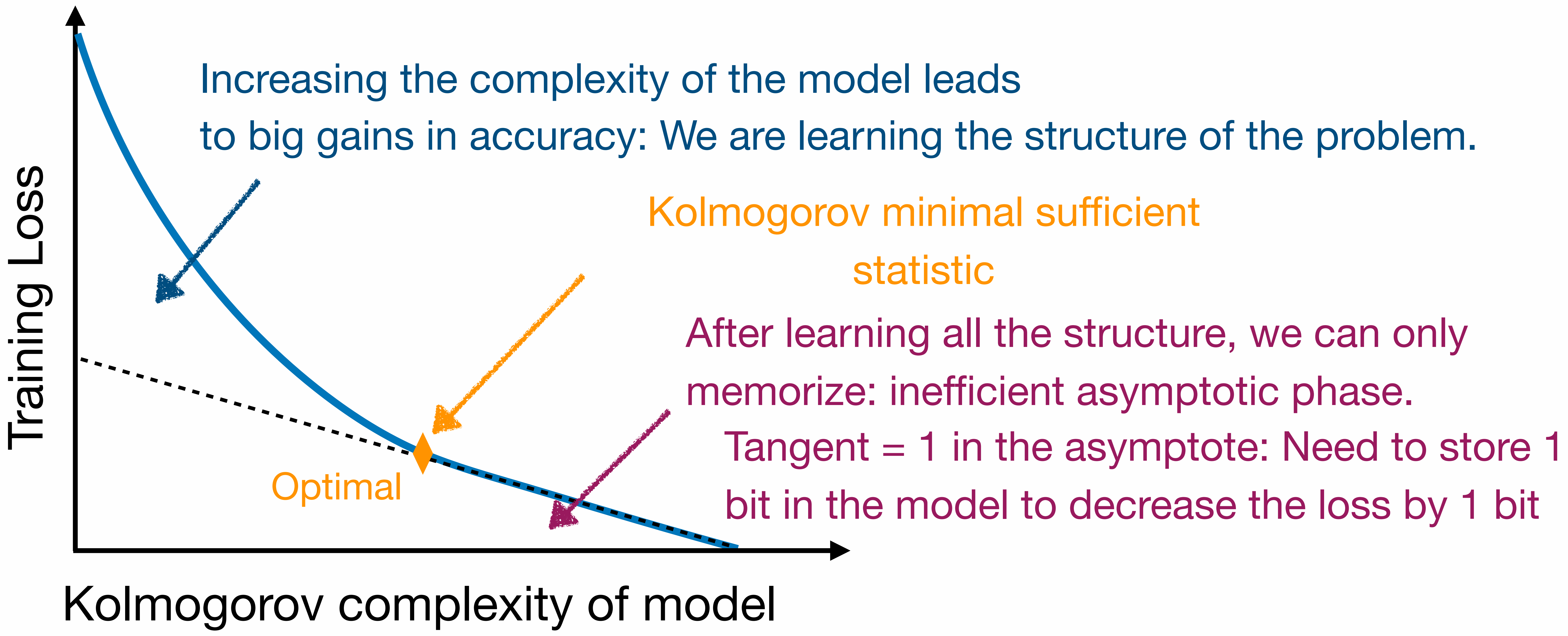
$$S_{\mathscr{D}}(t) = \min_{K(p_\theta)\leq t} L(\mathscr{D}; p_\theta)$$

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

Information Complexity of Tasks, their Structure and their Distance, Achille et al., 2018

The Kolmogorov Structure of a Task

$$S_{\mathscr{D}}(t) = \min_{K(p_\theta) \le t} L(\mathscr{D}; p_\theta)$$

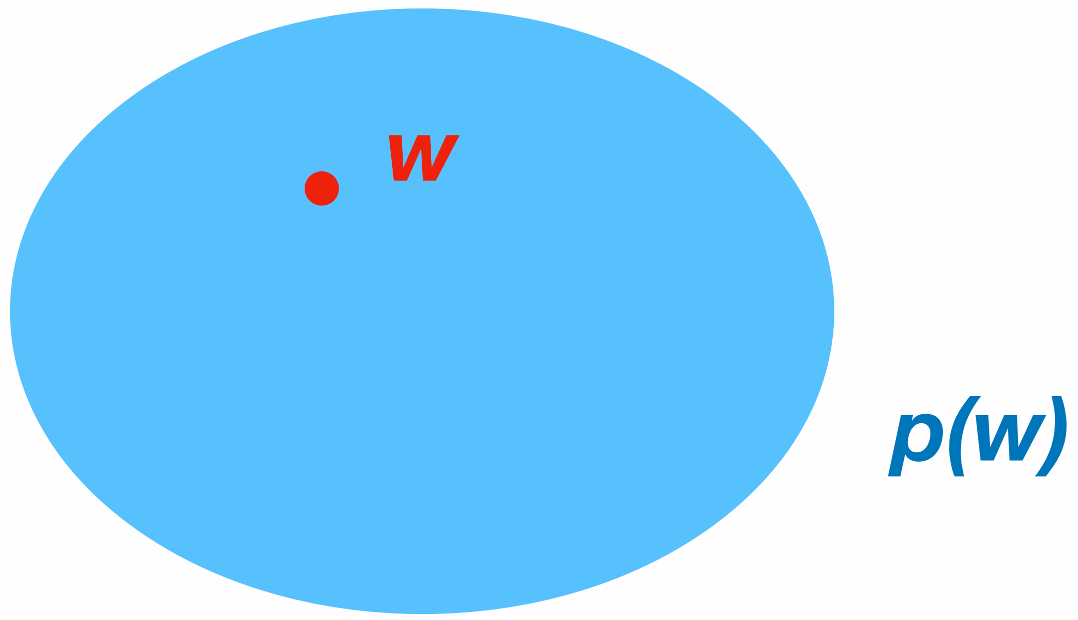


Kolmogorov's Structure Functions and Model Selection, Vereshchagin and Vitanyi, 2002
Information Complexity of Tasks, their Structure and their Distance, Achille et al., 2018

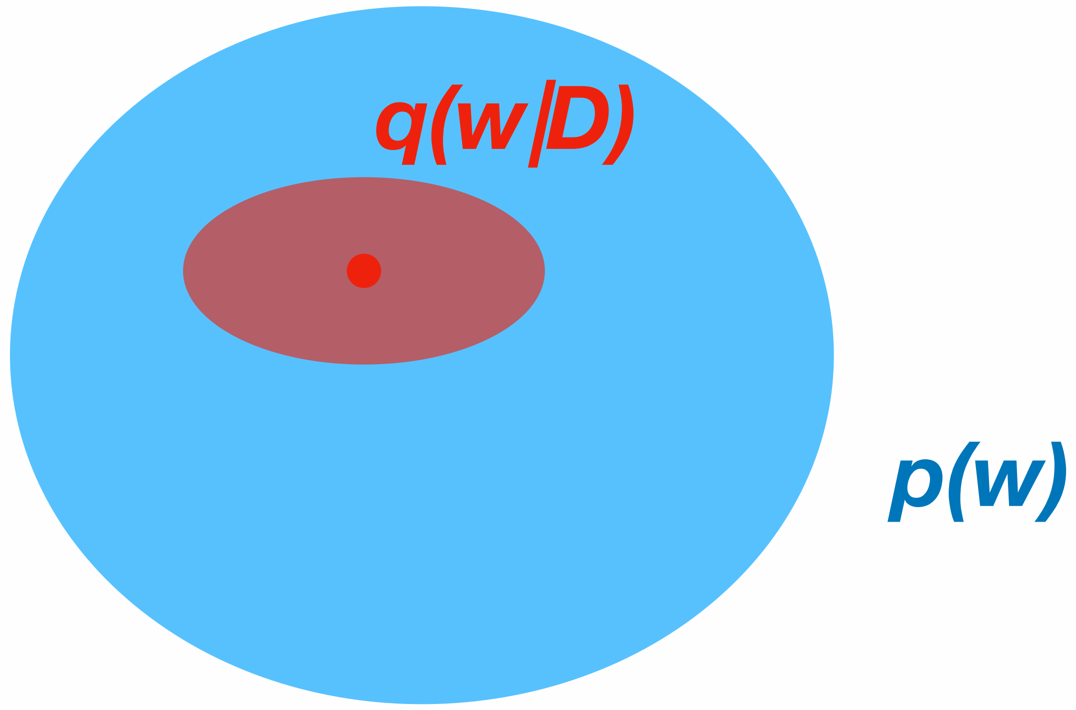
The Information in the Weights

How do we measure the complexity of a DNN?

Assume some prior distribution *p(w)* over the weights. Codifying a particular set of weights as real numbers requires infinite information.



Idea: Add noise to the weights to encode with a finite amount of information (Hinton, 1993)



$$|\text{encoding}| = \mathrm{KL}(\, q(w \mid \mathscr{D}) \,\|\, p(w)\,)$$

Hinton, Van Camp, 1993

The Information in the Weights

In our setting, consider a noisy weight distribution *q(w | D)*. Measure the amount of noise by its divergence KL(*q(w | D) || p(w)*) from a fixed prior *p(w).*

Expected loss over the noisy weights

$$S(t) = \min_{\mathrm{KL}(q(w|\mathcal{D})\,\|\,p(w))<t} \mathbb{E}_{w\sim q(w|\mathcal{D})}[L_{\mathcal{D}}(w)]$$

Or, equivalently the Lagrangian:

$$\mathcal{L} = \mathbb{E}_{w\sim q(w|\mathcal{D})}[L_{\mathcal{D}}(w)] + \beta\,\overbrace{\mathrm{KL}(\underbrace{q(w|\mathcal{D})}_{\text{optimal noise}}\,\|\,\underbrace{p(w)}_{\text{fixed prior}})}^{\text{Information in the Weights}}$$

For a given β we call Information in the Weights the value of the KL divergence of the optimal solution.

A., Paolini, Soatto, Information Complexity of Learning Tasks, 2019
A., and Soatto, Where is the Information in a DNN?, 2019

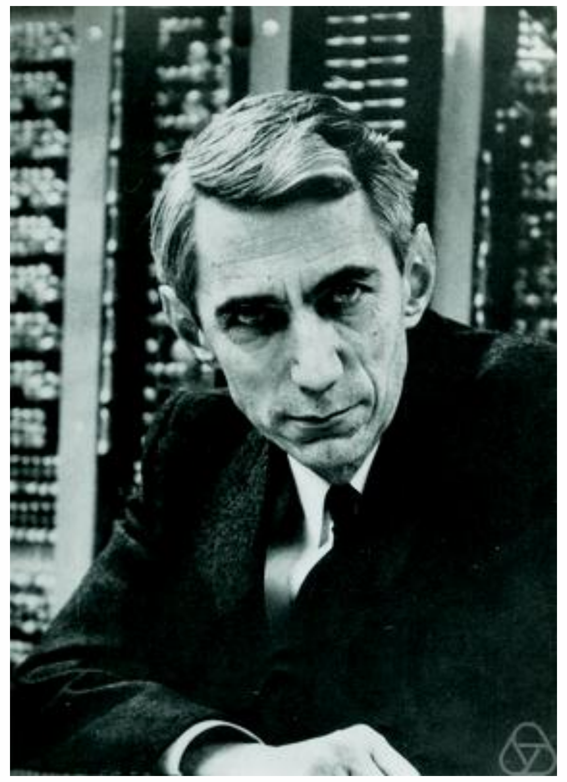
Example: Measuring Information by Adding Noise

Idea: We can estimate the amount of information contained in the weights by corrupting them with noise and measuring the decrease in performance.

Prediction and Entropy of Printed English

By C. E. SHANNON

(Manuscript Received Sept. 15, 1950)



Example: Shannon (1951) estimates the information content of the English language by corrupting random letters and measuring the reconstruction error of English speakers.

“Thif is a vevy moisy party” → “This is a very noisy party”

Let's rewrite this using Information Theory

We used an upperbound, what is the best we value it can assume?

$$\mathscr{L}(M) = \mathbb{E}_{w \sim q(w|\mathscr{D})}[H_{p,q}(\mathscr{D} \,|\, w)] + \lambda \, \mathrm{KL}(q(w \,|\, \mathscr{D}) \| p(w)) \,.$$

Recall that:

$$I(w; \mathscr{D}) \leq \mathbb{E}_{\mathscr{D}}[\mathrm{KL}(q(w \,|\, \mathscr{D}) \| p(w))],$$

which is obtained when *p(w)* = *q(w|D)*. Hence, on expectation over the datasets, the best function loss function to use to recover the task structure is:

$$\boxed{\mathscr{L}(M) = \mathbb{E}_{\mathscr{D}}[H(\mathscr{D} \,|\, w)] + \lambda I(w; \mathscr{D}) \,.}$$

IB Lagrangian for the weights

The Information in a Deep Neural Network

$$L(w) = H_{p,q}(\mathcal{D}|w) + \beta\,\mathrm{KL}(\underbrace{q(w|\mathcal{D})}_{\text{output of training}} \,\|\, \underbrace{p(w)}_{\text{fixed prior}})$$

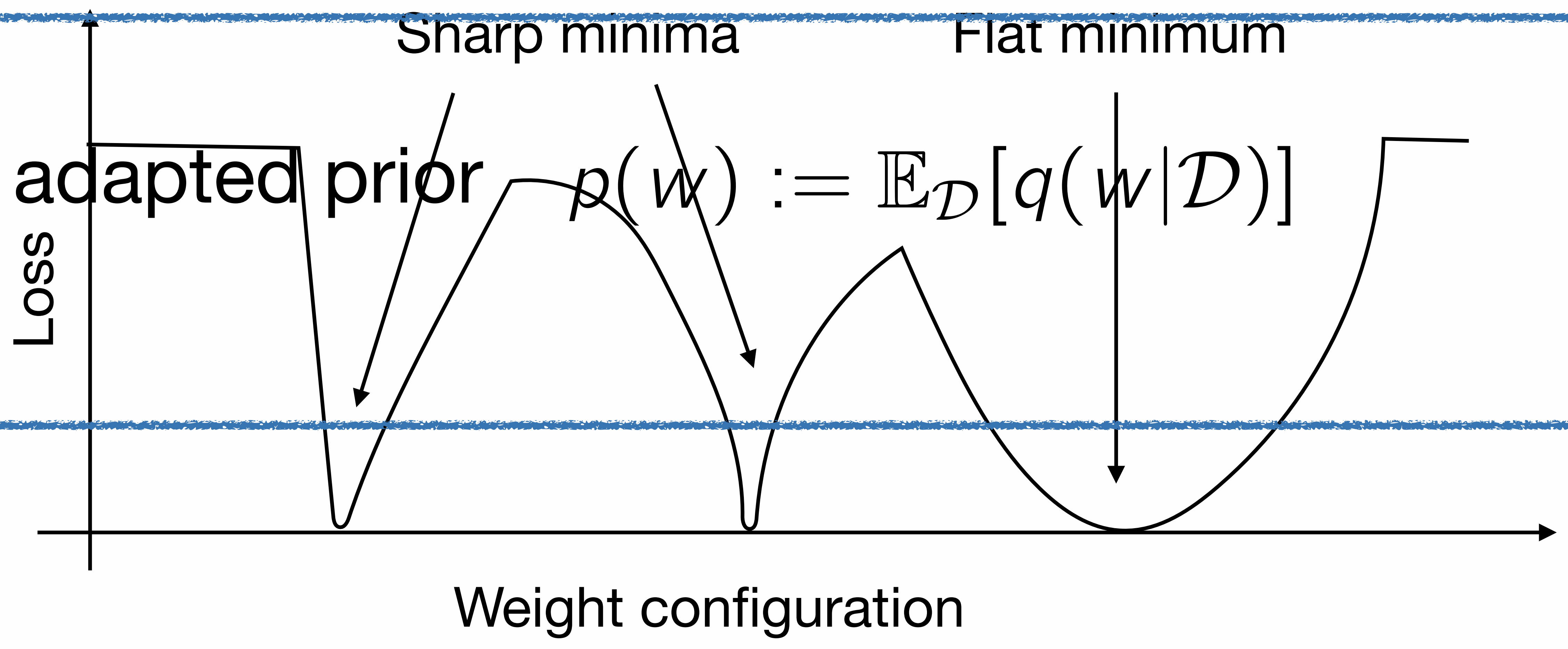
Fisher Information: *p(w)* = Gaussian prior, assume the loss is locally quadratic

$$\mathrm{KL} = \frac{\|w\|^2}{\lambda^2} + \log|2\lambda^2 NF + I|$$

F = curvature of loss landscape

⇒ Implicitly minimized by SGD

Shannon Information: adapted prior $p(w) := \mathbb{E}_{\mathcal{D}}[q(w|\mathcal{D})]$ $\mathbb{E}_{\mathcal{D}}[\mathrm{KL}] = I(w;\mathcal{D})$



A. et al., *The Information Complexity of Learning Tasks, their Structure and their Distance,* ArXiv 2019
Li et al., *Visualizing the Loss Landscape of Neural Nets*, ICLR 2018, Hochreiter and Schmidhuber, *Flat Minima*, *Neural Computation*1997

The PAC-Bayes generalization bound

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

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

Moreover, the sharpest bound is obtained when *E*[KL] = *I(w; D)*.

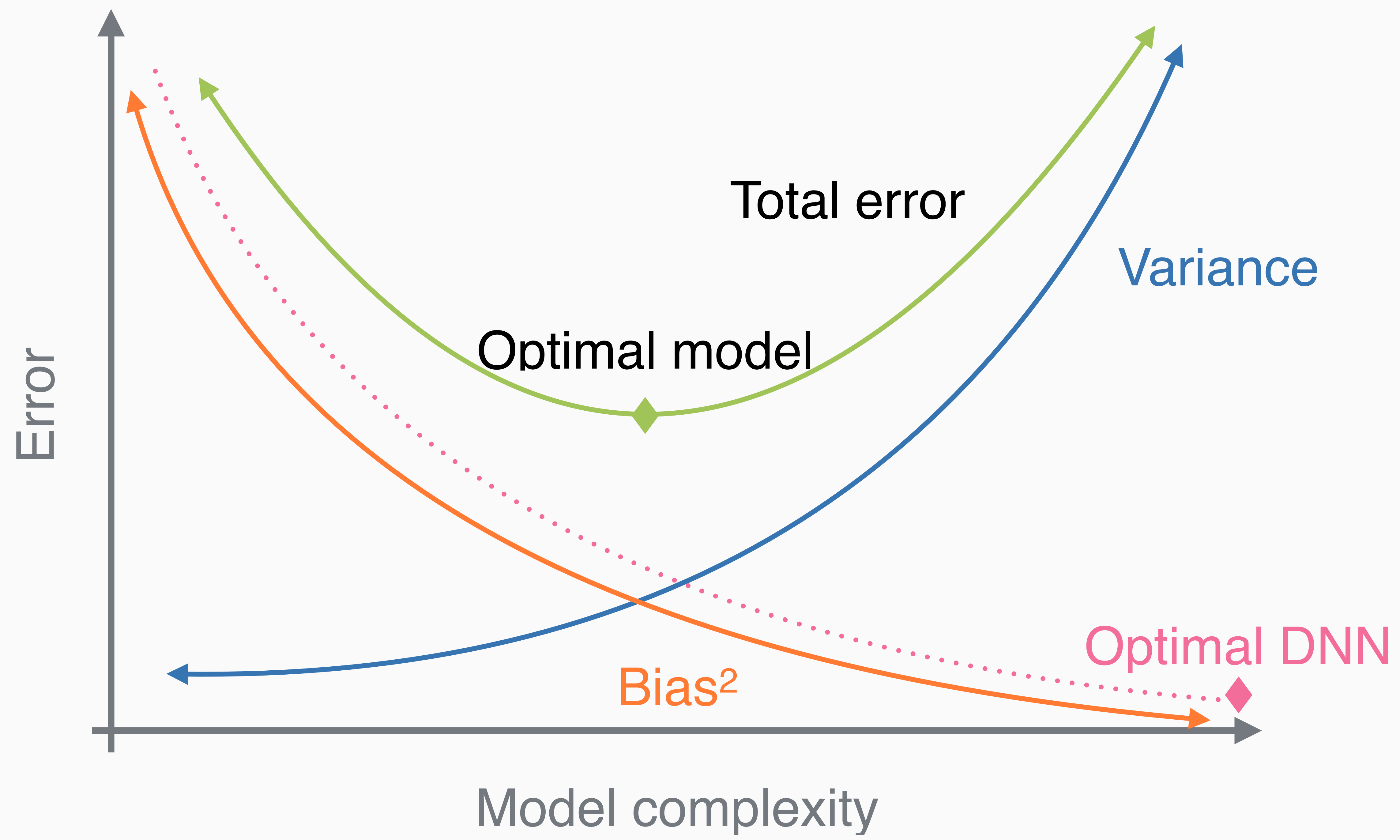
What matters for generalization is not the number of weights, but the information they contain.

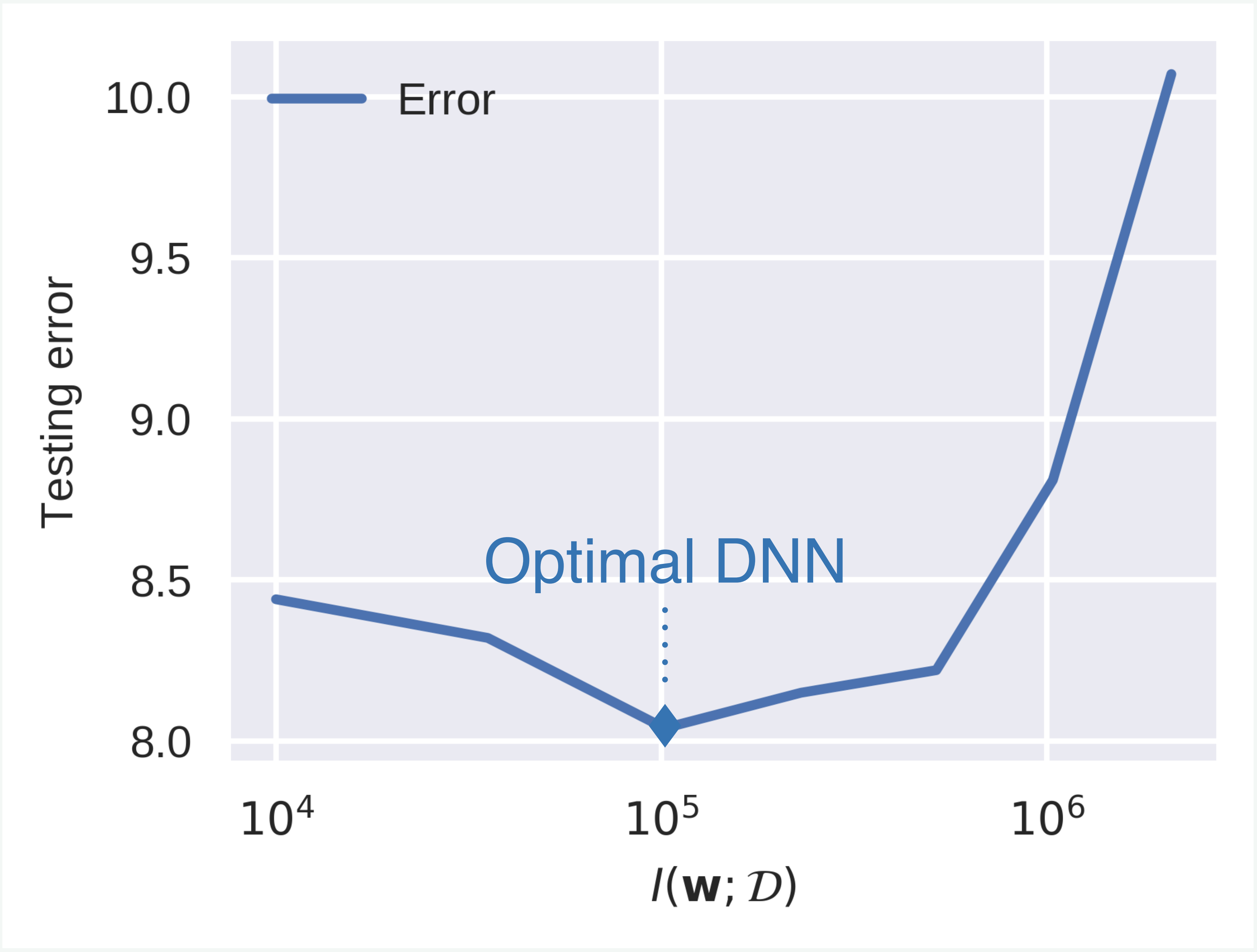
This gives non-vacuous generalization bounds.

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

Bias-variance tradeoff

Information is a better measure of complexity than number of parameters





Information complexity

Parametrizing the complexity with information in the weights, we recover bias-variance trade-off trend.

Achille and Soatto, Emergence of Invariance and Disentanglement in Deep Representations, JMLR 2018
Arora et al., Stronger generalization bounds for deep nets via a compression approach, ICML 2018

Relation between Fisher and Shannon

SGD minimizes the Fisher Information of the Weights. However, generalization is governed by the Shannon Information.

Proposition. Assuming the dataset is parametrized in a differentiable way, we have:

$$\boxed{I(w;\mathcal{D})} \approx H(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[\log\left(\frac{(2\pi e)^k}{|\nabla_{\mathcal{D}} w^* \, \boxed{F(w^*)} \, \nabla_{\mathcal{D}} w^{*T}|}\right)\right]$$

Where $w^* = w^*(D)$ is the result of running SGD on dataset D and F(w) is the Fisher Information Matrix in w.

A. and Soatto, Where is the Information in a Deep Network?, 2018

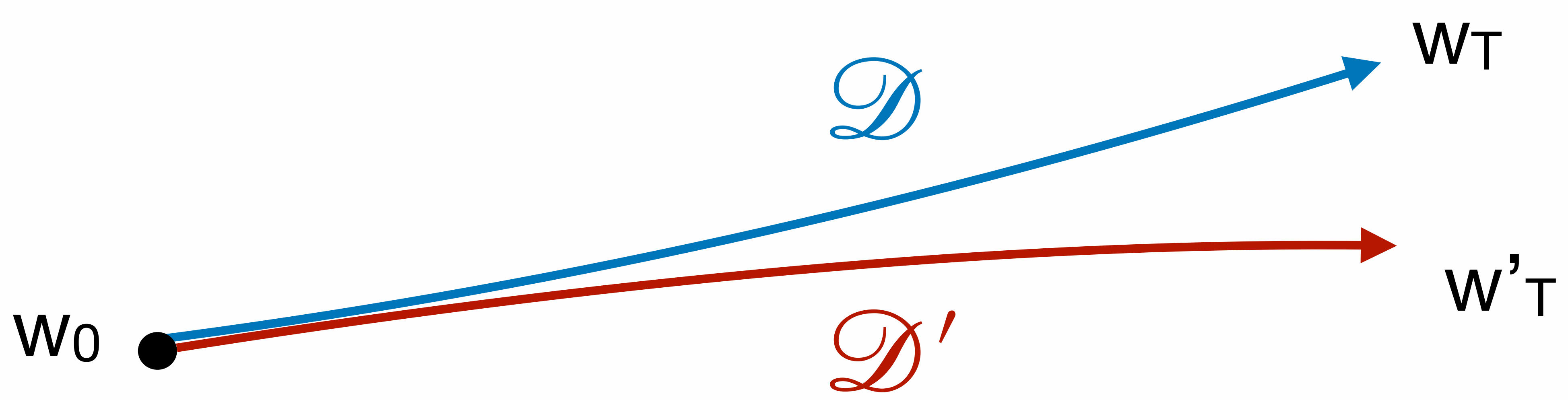
Stability of SGD

$$I(w;\mathcal{D}) \approx H(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[\log\left(\frac{(2\pi e)^k}{|\nabla_{\mathcal{D}} w^* \; F(w^*) \; \nabla_{\mathcal{D}} w^{*T}|}\right)\right]$$

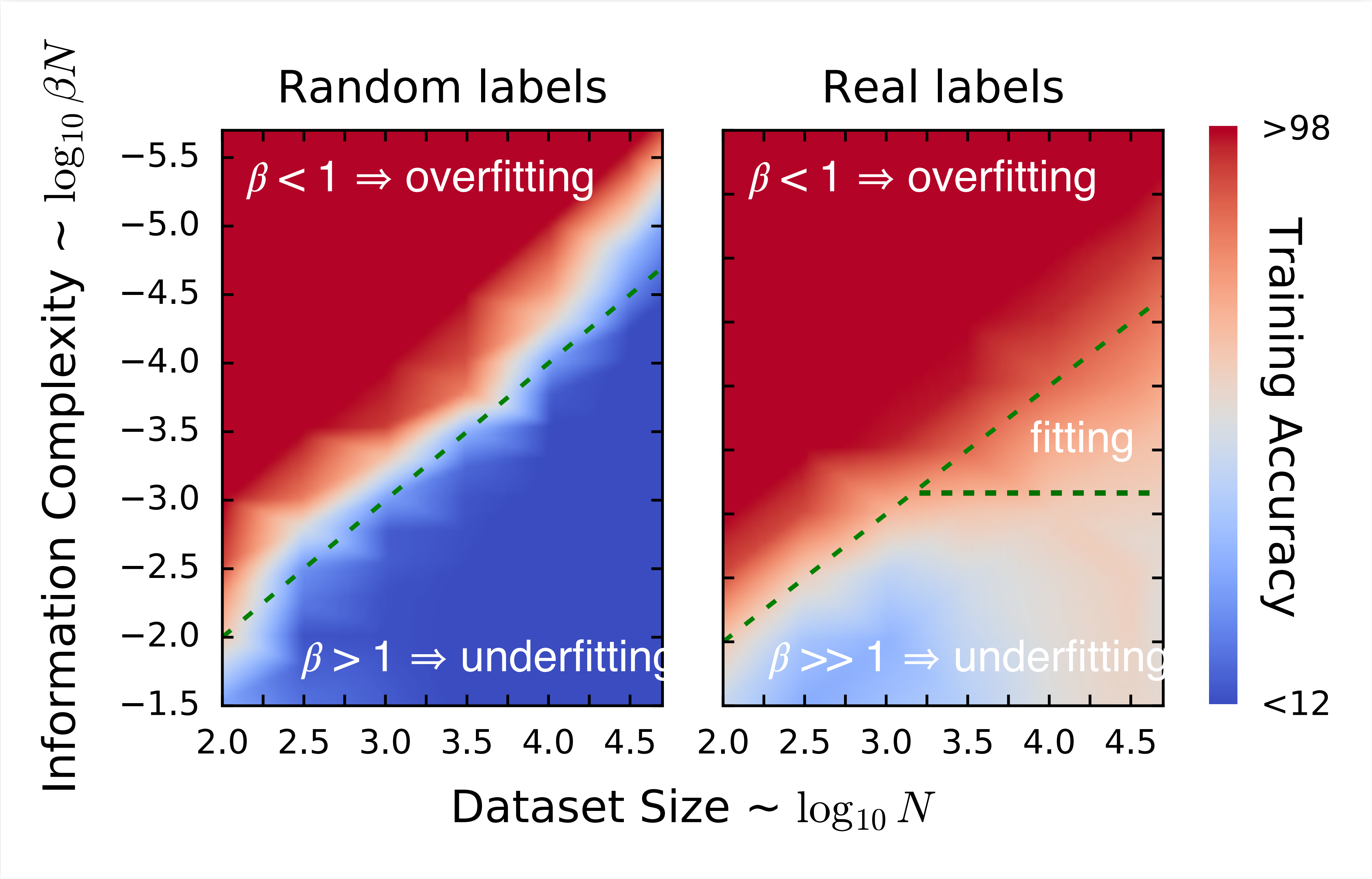
Dependency of final training point on the dataset

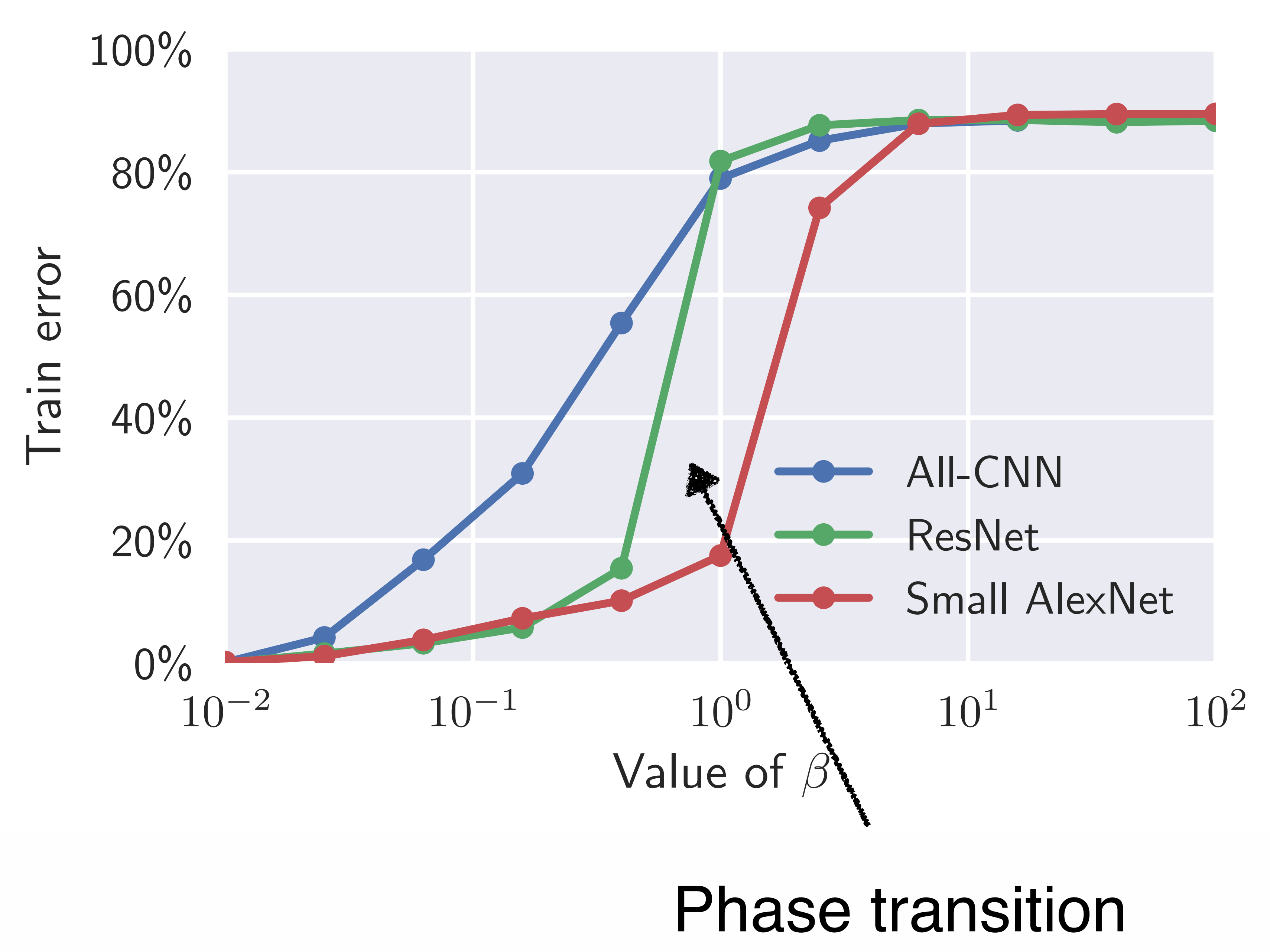
Curvature at the final point

Imagine training a network on a dataset *D* and on a slightly perturbed dataset *D'*



Phase transition





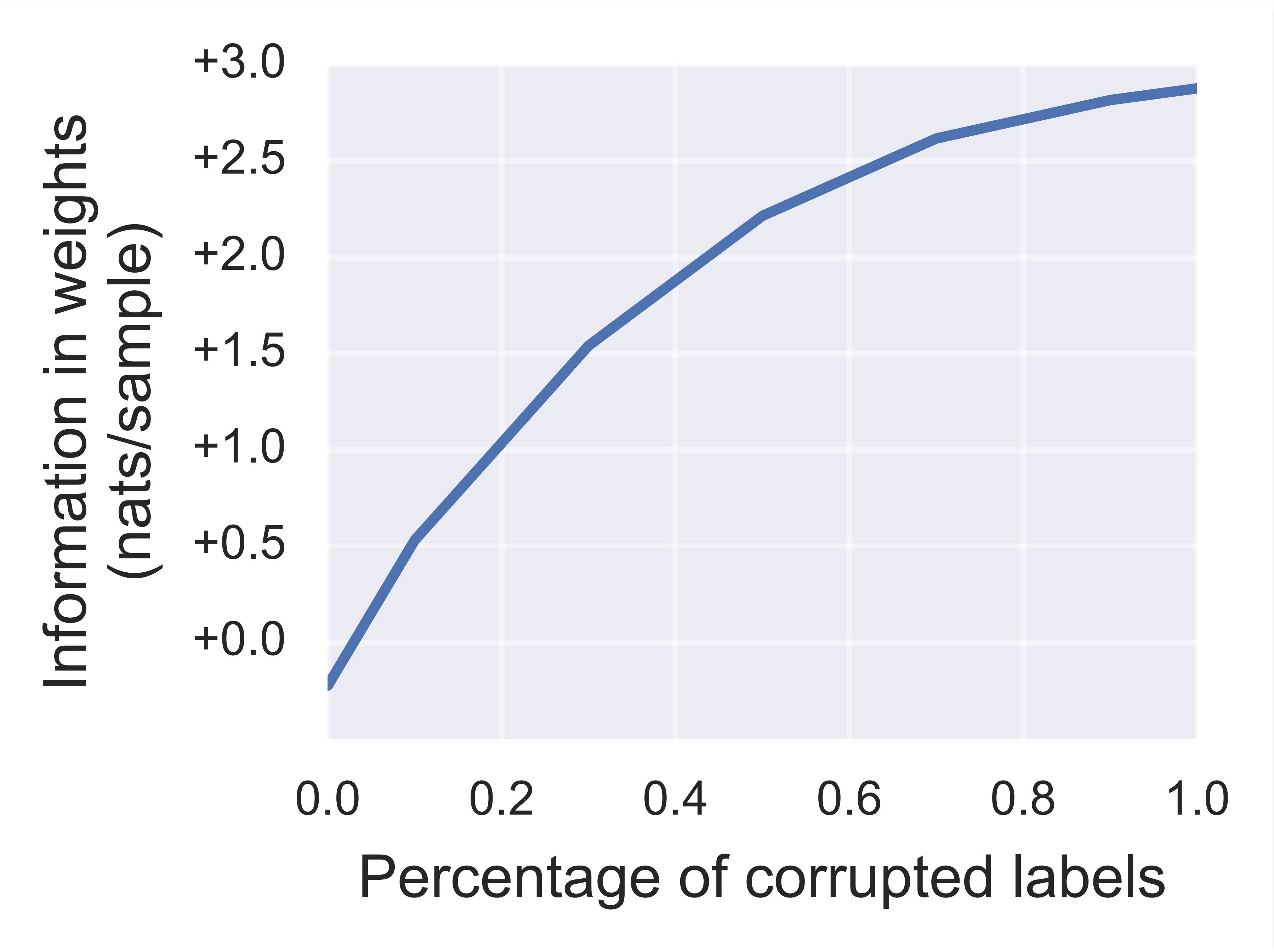
Using the regularized loss:

$$L(w) = H_{p,q}(\mathcal{D}|w) + \beta\, KL(q(w|\mathcal{D}) \| p(w))$$

For random labels there is a transition between over- and under-fitting at $\beta = 1$.

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

Networks can overfit, but they have to pay a price



Information in weights as a function of the number of corrupted labels.

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

Two Bottlenecks

Activations IB
Invariance

$x \longrightarrow z \longrightarrow y$

data — activations — label

$$\min_{q(z|x)} \mathcal{L} = H_{p,q}(y|z) + \beta I(z;x)$$

Weights IB
Generalization

$D \longrightarrow w \longrightarrow p(y|x)$

dataset — weights — real distribution

$$\min_{w} \mathcal{L} = H_{p,q_w}(y|z) + \beta I(\mathcal{D};w)$$

The Emergence Bound

Let z = f_w(x) be a layer of a network, and let z_n be the representation obtained by adding noise to the weigths. We define the effective information as $I_{eff}(x; z) = I(x; z_n)$

Proposition. Let z = f_w(x) be a layer of a network. To a first order approximation, the information in the activations is given by:

$$I_{\text{eff}}(x; z) \approx H(x) - \log \left(\frac{(2\pi e)^k}{|\nabla_x f_w(x)^t \ J_f^t \ F(w) \ J_f \ \nabla_x f_w(x)|} \right)$$

where $F(w)$ is the Fisher Infomation of the weights, J_f is the jacobian of f_w w.r.t. w.

Take-away: Reducing information in the weights reduces information in the activations, hence it promotes invariant classifiers.

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

A. and Soatto, Where is the Information in a Deep Network?, 2019

Explanation

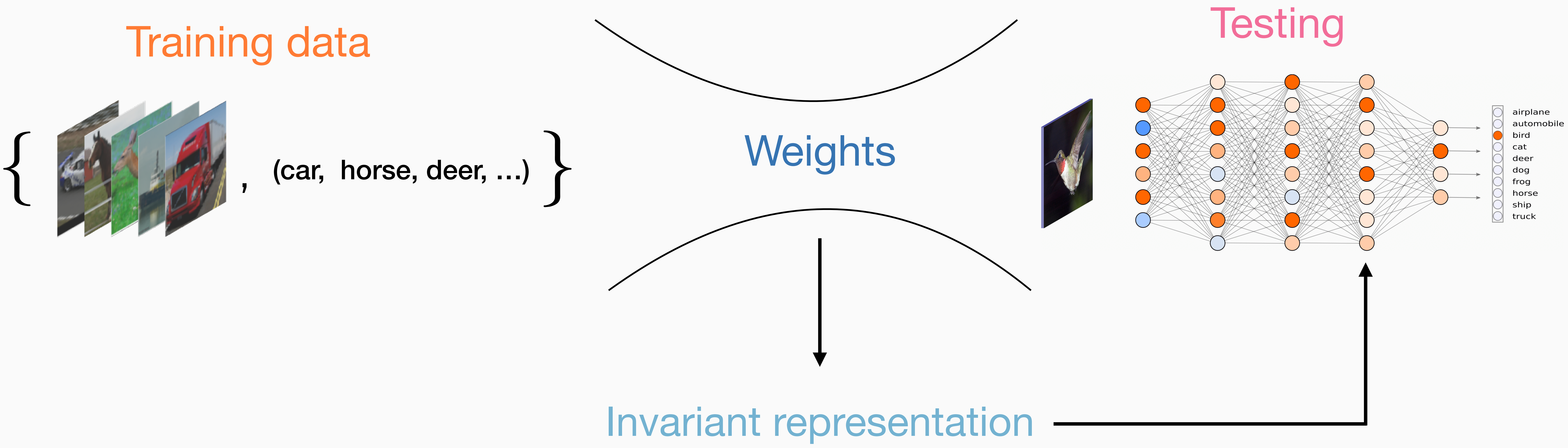
Jacobian of activations wrt inputs

$$I_{\mathrm{eff}}(x;z) \approx H(x) - \log\left(\frac{(2\pi e)^k}{|\nabla_x f_w(x)^t \ J_f^t \ F(w) \ J_f \ \nabla_x f_w(x)|}\right)$$

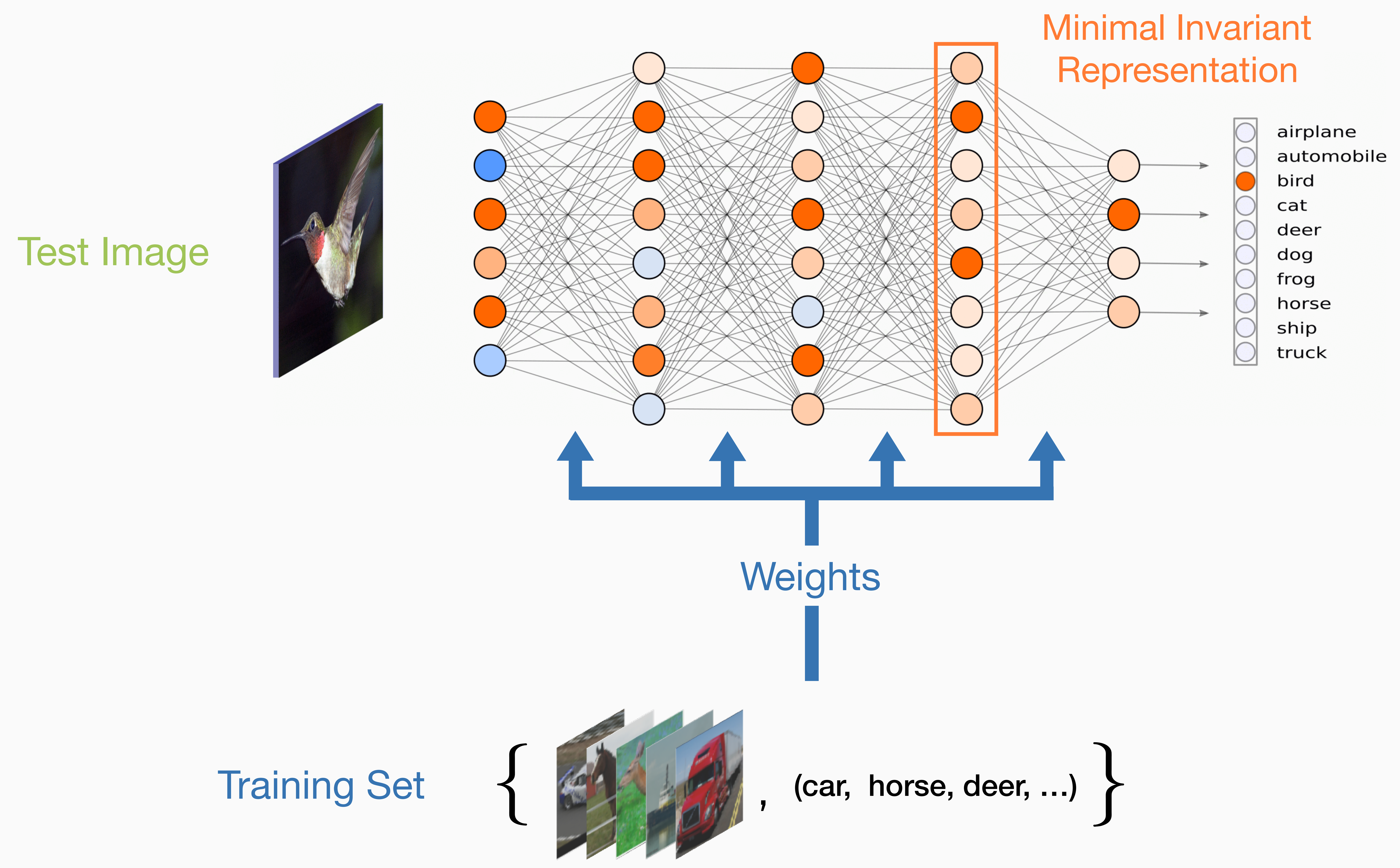
Curvature of loss function

Jacobian of representation wrt to weights

Compression of the **weights** biases toward invariant and disentangled **representations**.

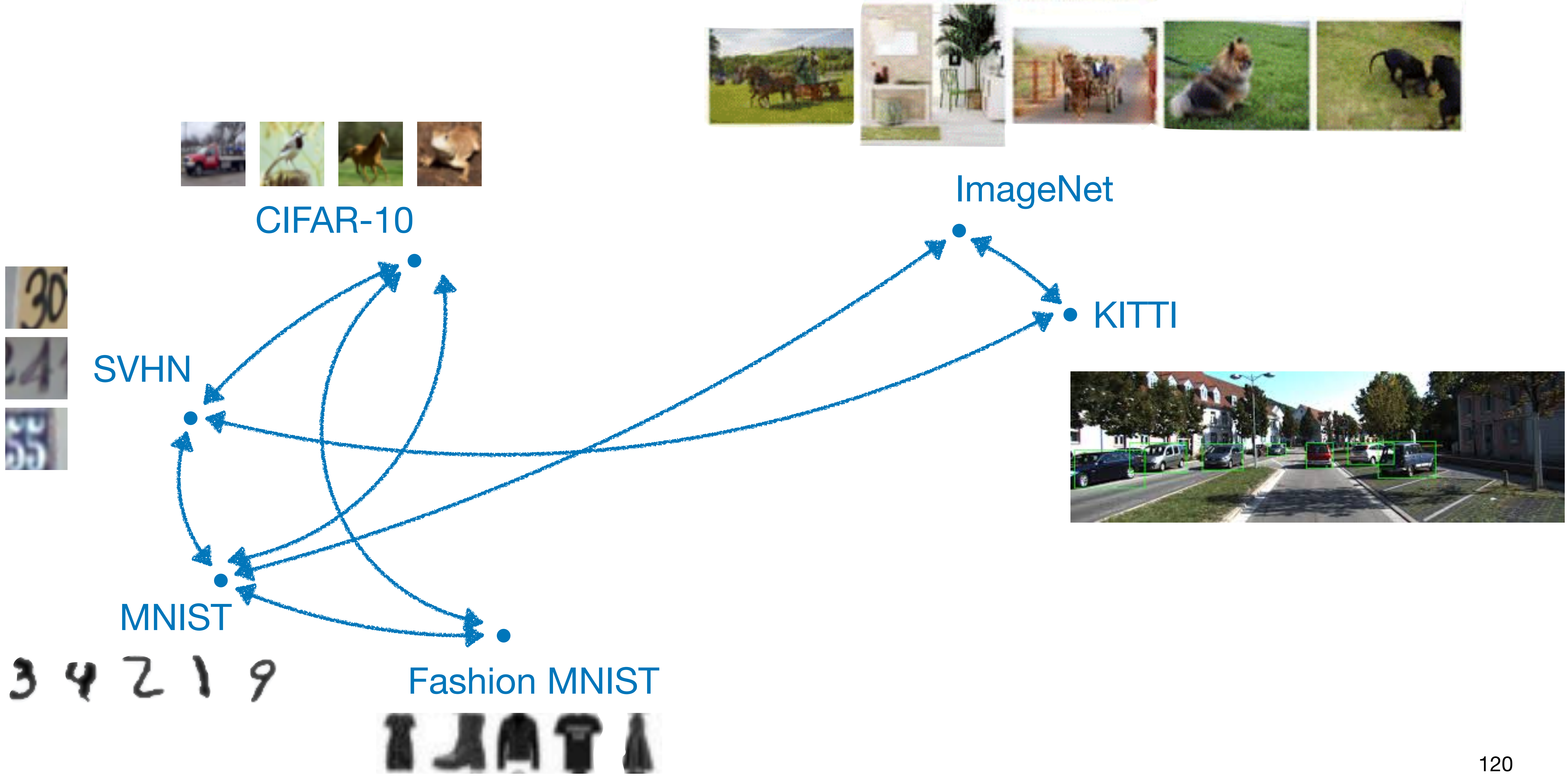


PAST	FUTURE
Weights	Activations
Generalization (PAC-Bayes)	Invariance (Emergence)
Minimality (Shannon)	Minimality (Fisher)



Minimal Invariant Representation
Test Image
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Weights
Training Set
(car, horse, deer, ...)

What is the distance between two tasks?



A Topology on the Space of Tasks

Distance between tasks:

$$d(\mathscr{D}_1 \to \mathscr{D}_2) = I(\mathscr{D}_1\mathscr{D}_2; w) - I(\mathscr{D}_1; w)$$

Complexity of learning together

Complexity of learning one

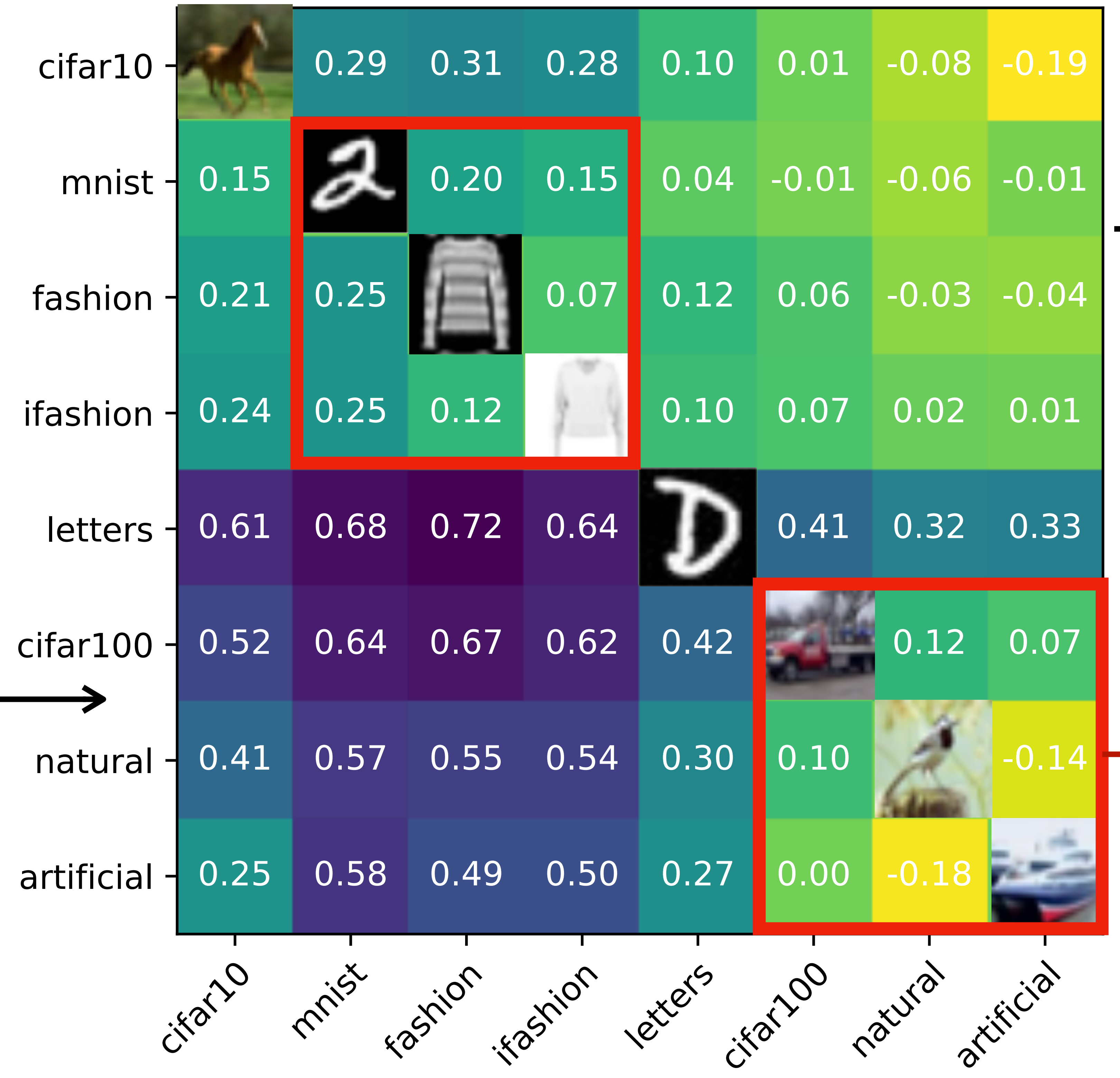
That is, how much more structure do we need to learn?

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

$$d(\mathscr{D}_1 \to \mathscr{D}_2) = I(\mathscr{D}_1 \mathscr{D}_2; w) - I(\mathscr{D}_2; w)$$



Difficult task to easy task

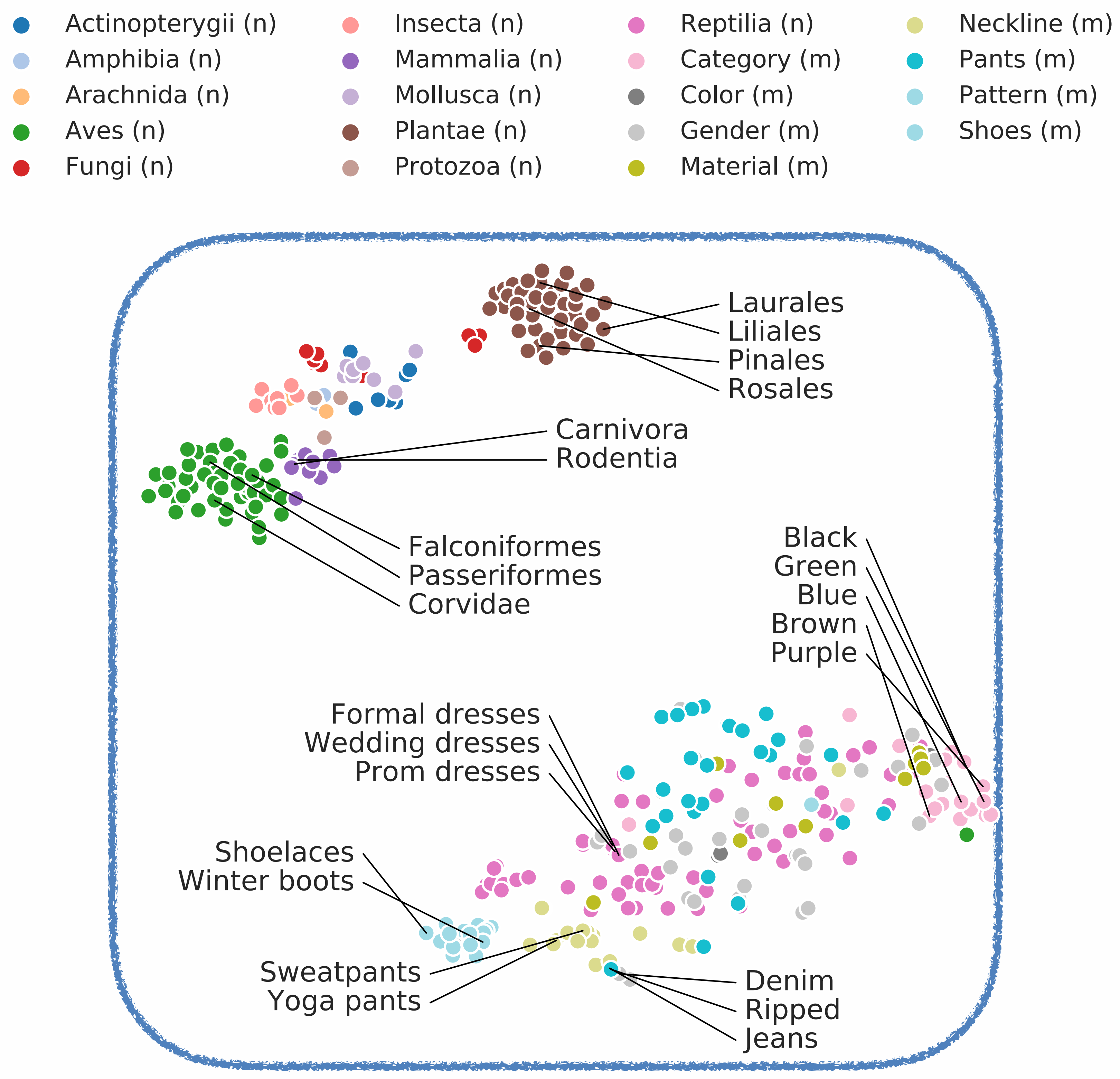
Easy task to difficult task

Similar tasks cluster together

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

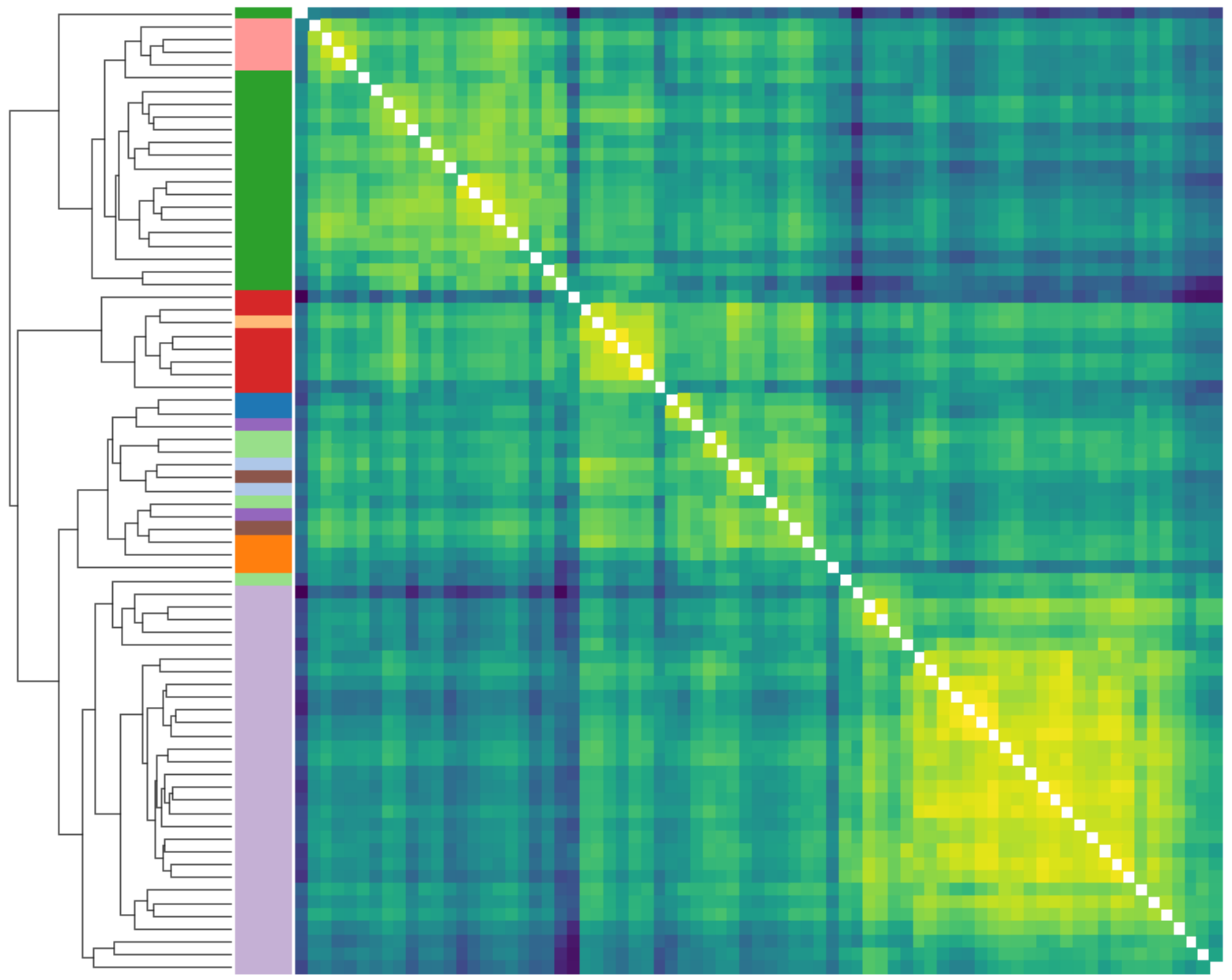
TASK2VEC: Embedding tasks in a metric space

Idea: Represent each tasks in a metric space using its Fisher Information Matrix diagonal.

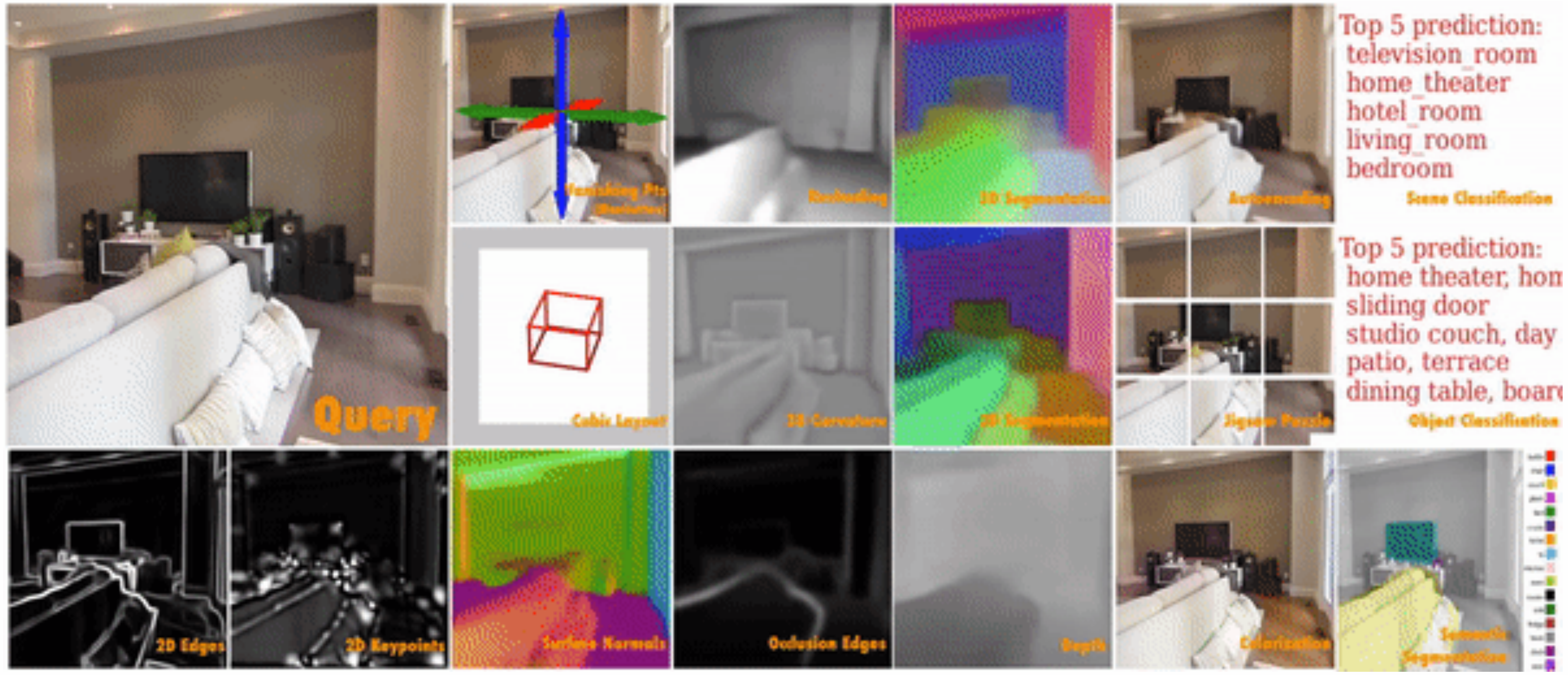


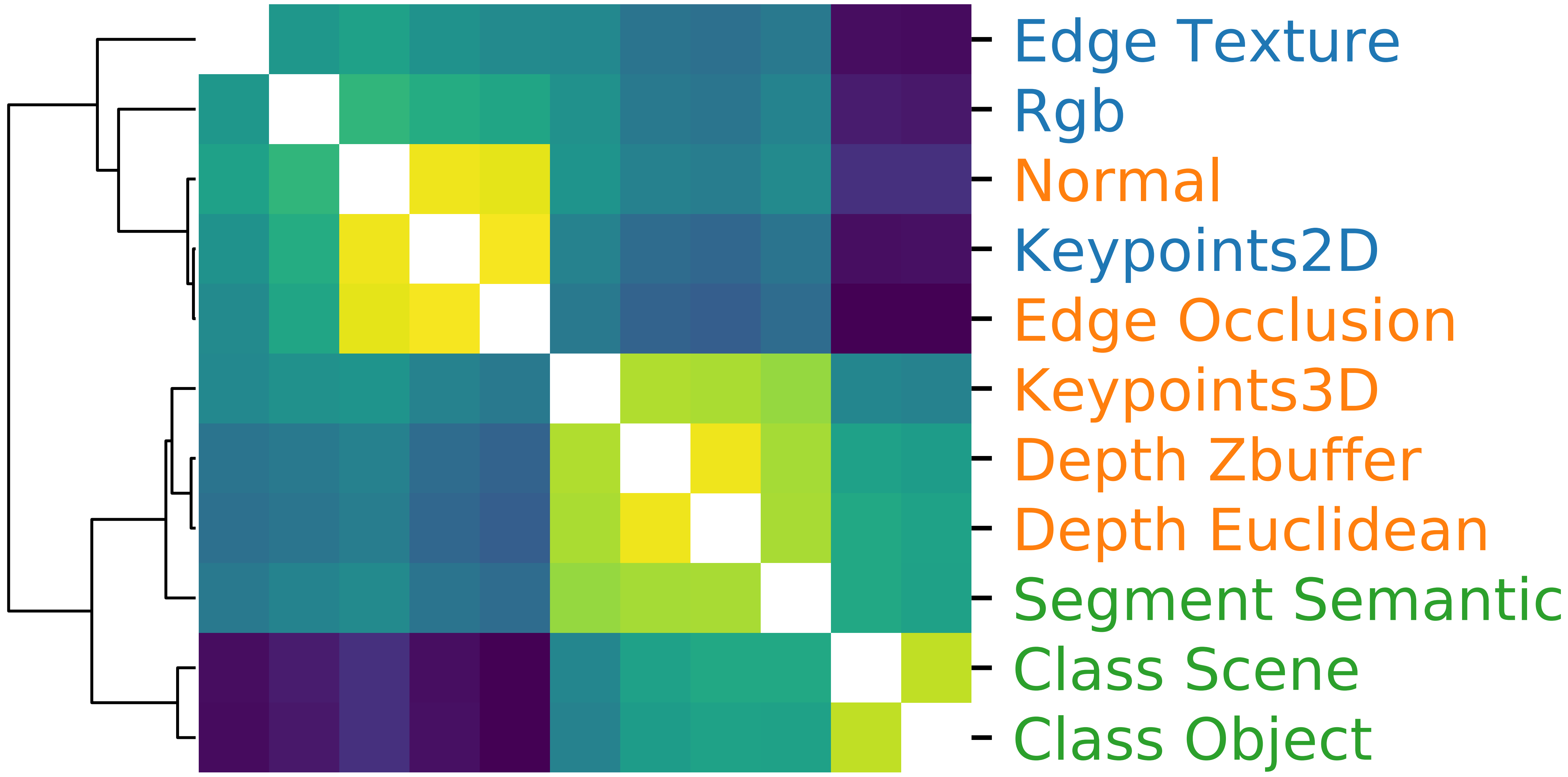
Recovers a meaningful topology on hundred of tasks

Recovers species taxonomy on iNaturalist

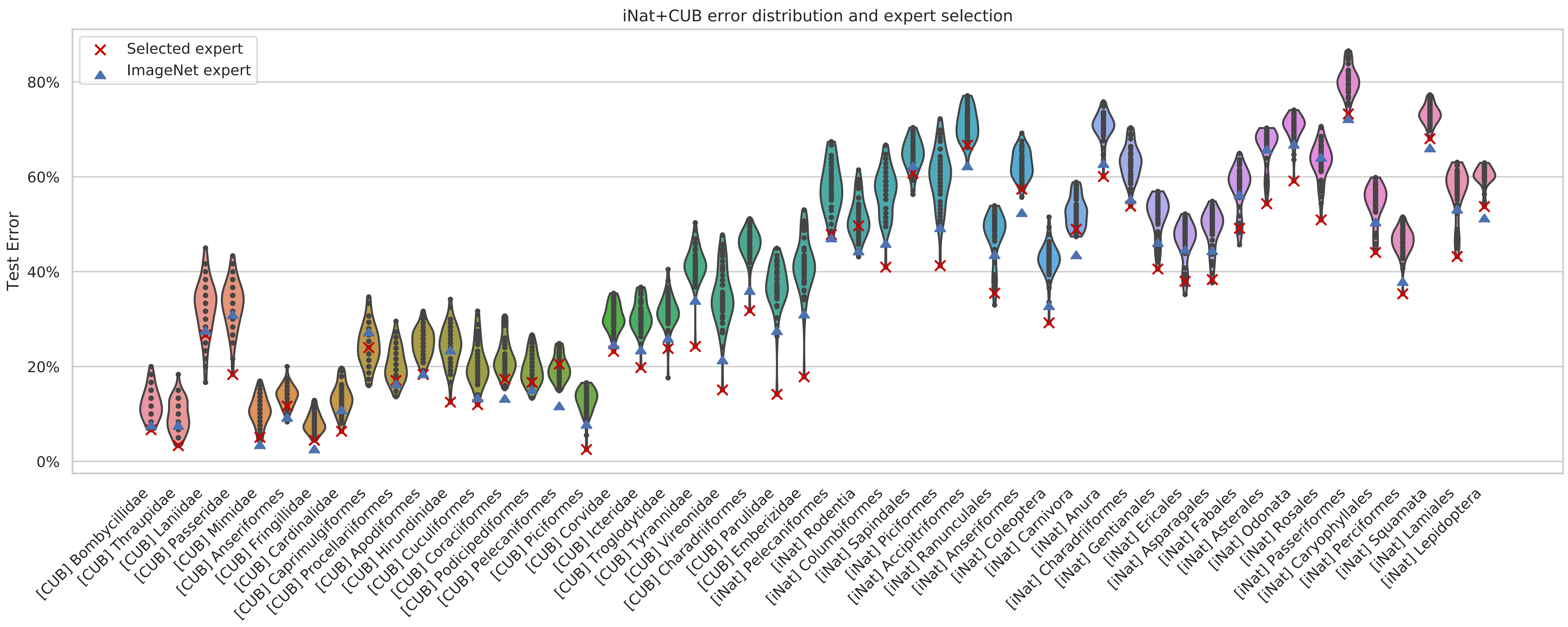


ARCAMA


Top 5 prediction:
television_room
home_theater
hotel_room
living_room
bedroom
Scene Classification
Top 5 prediction:
sliding door
patio, terrace
Object Classification
Query
Cubic Layout
Jigsaw Puzzle
2D Edges
2D Keypoints
Surface Normals
Occlusion Edges
Depth
Colorization


Edge Texture
Rgb
Normal
Keypoints2D
Edge Occlusion
Keypoints3D
Depth Zbuffer
Depth Euclidean
Segment Semantic
Class Scene
Class Object

Proposing an optimal expert for the task



Allows to select the best expert to solve a task and substantially reduce error and training time.

A snag: Critical Periods

Two almost identical tasks, yet it is not possible to fine-tune from one to the other.

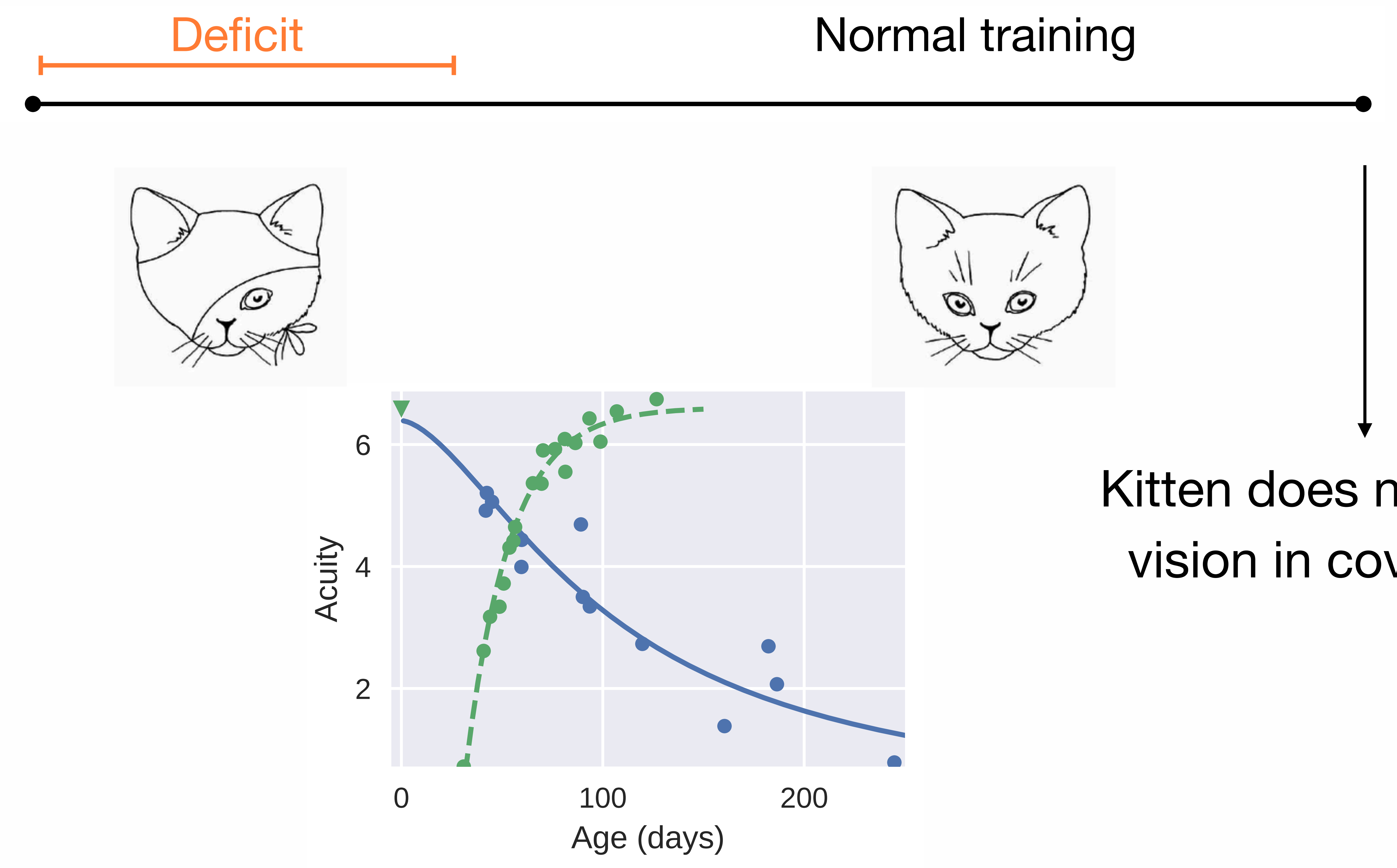
Excursus: Critical Periods for learning

Follow-up: Task reachability. Complexity is physical.

Critical periods

Critical periods: A time-period in early development where sensory deficits can permanently impair the acquisition of a skill

Examples: monocular deprivation, cataracts, imprinting, language acquisition

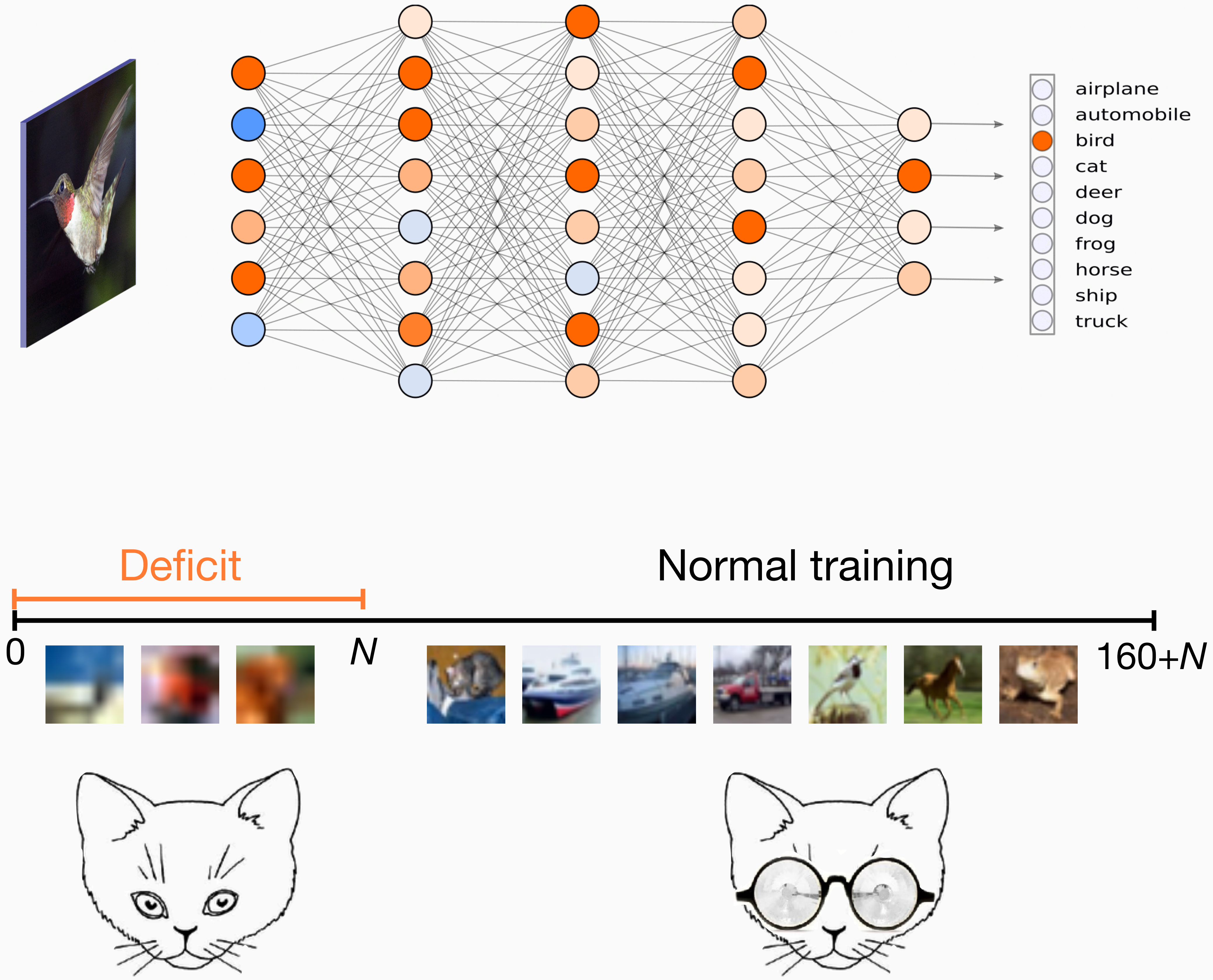


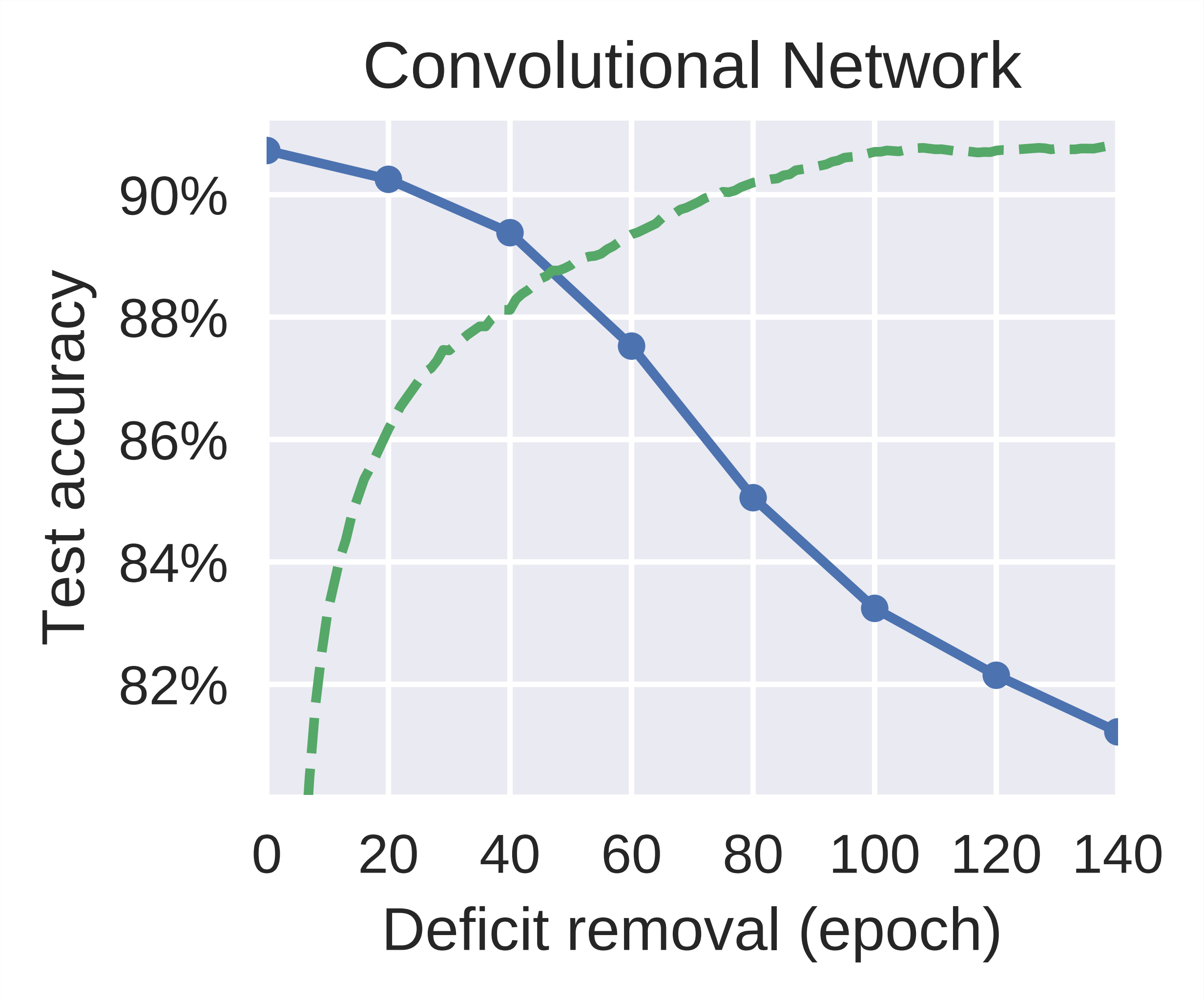
Kitten does not recover vision in covered eye

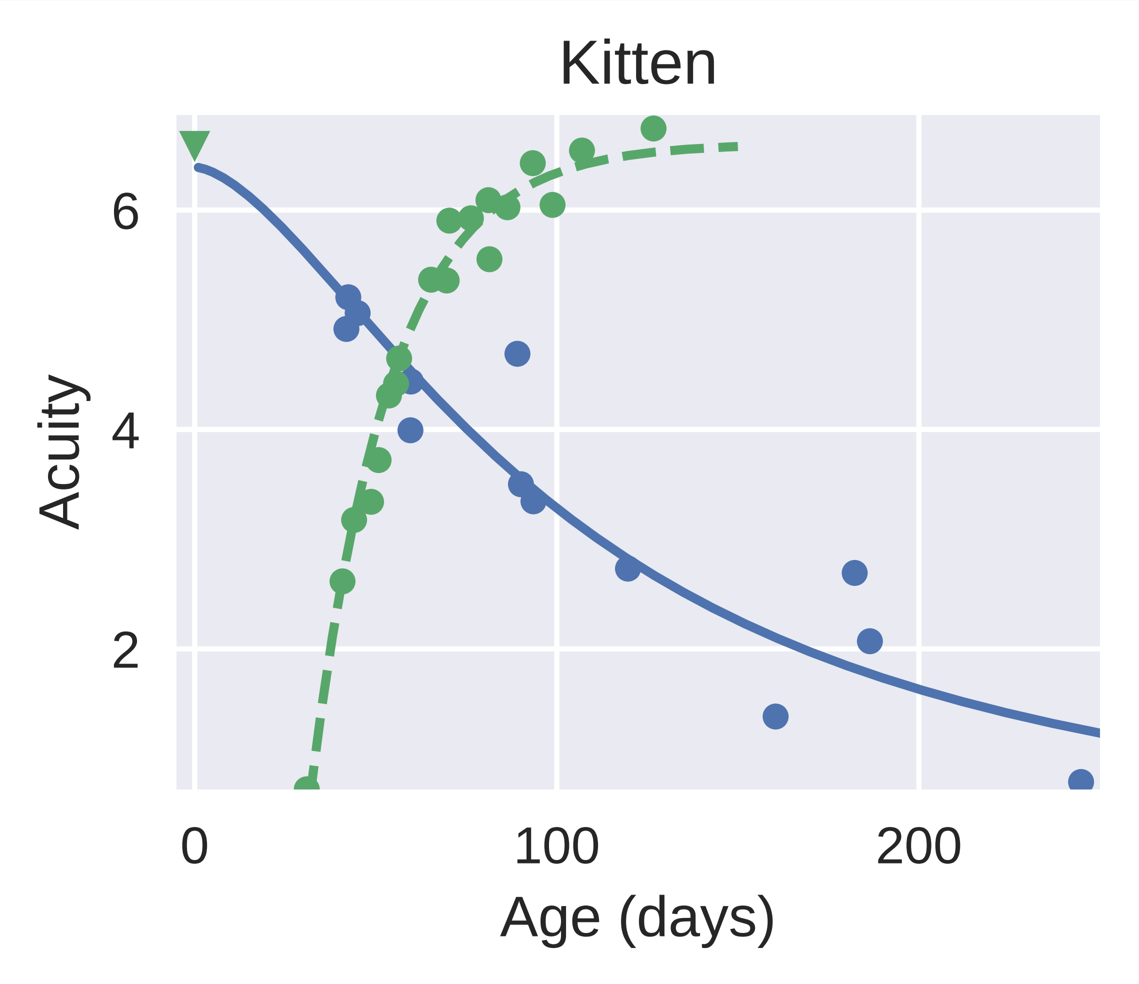
Hubel and Wiesel

Image from Cnops et al., 2008

Critical Learning Periods in Deep Networks

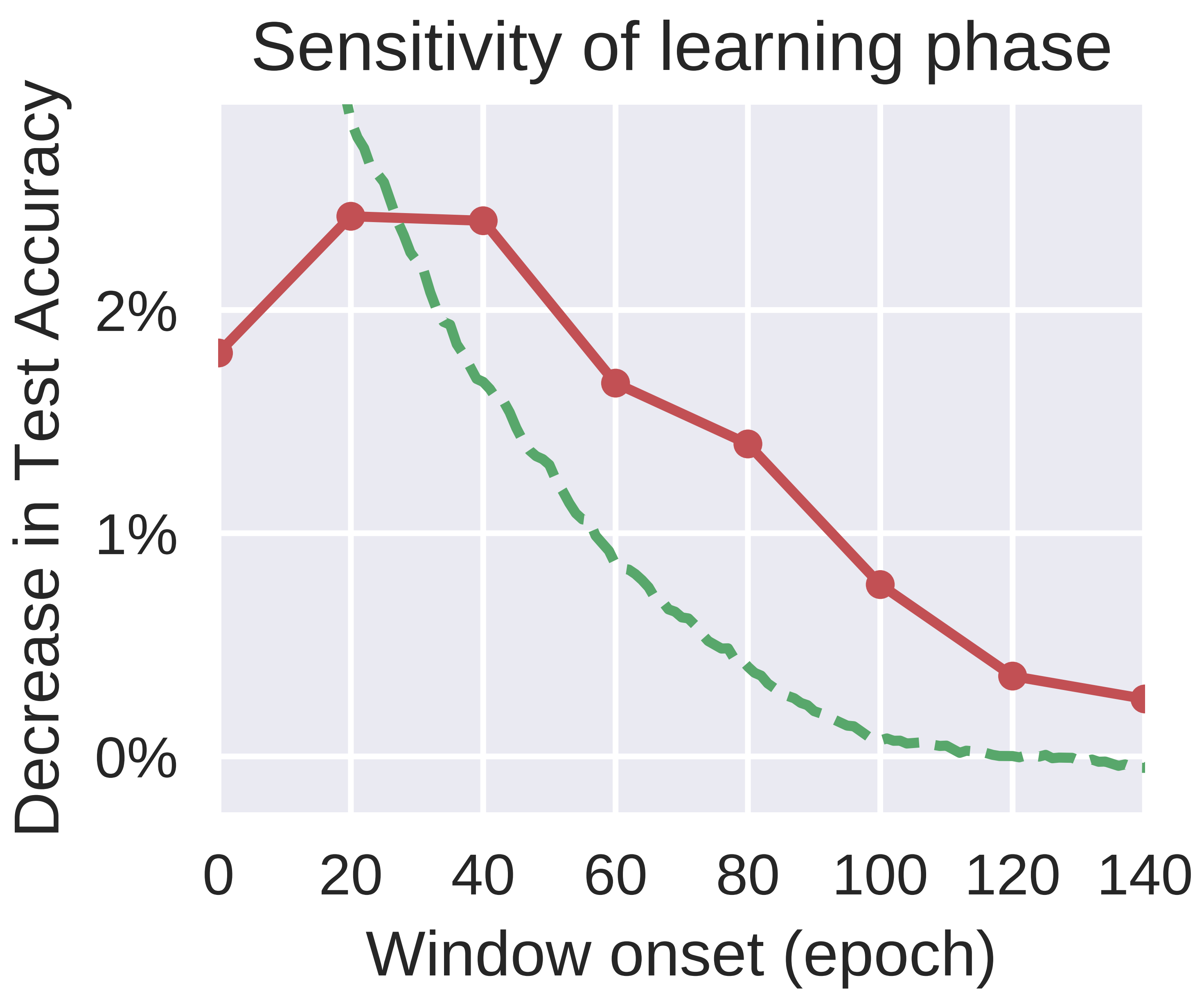


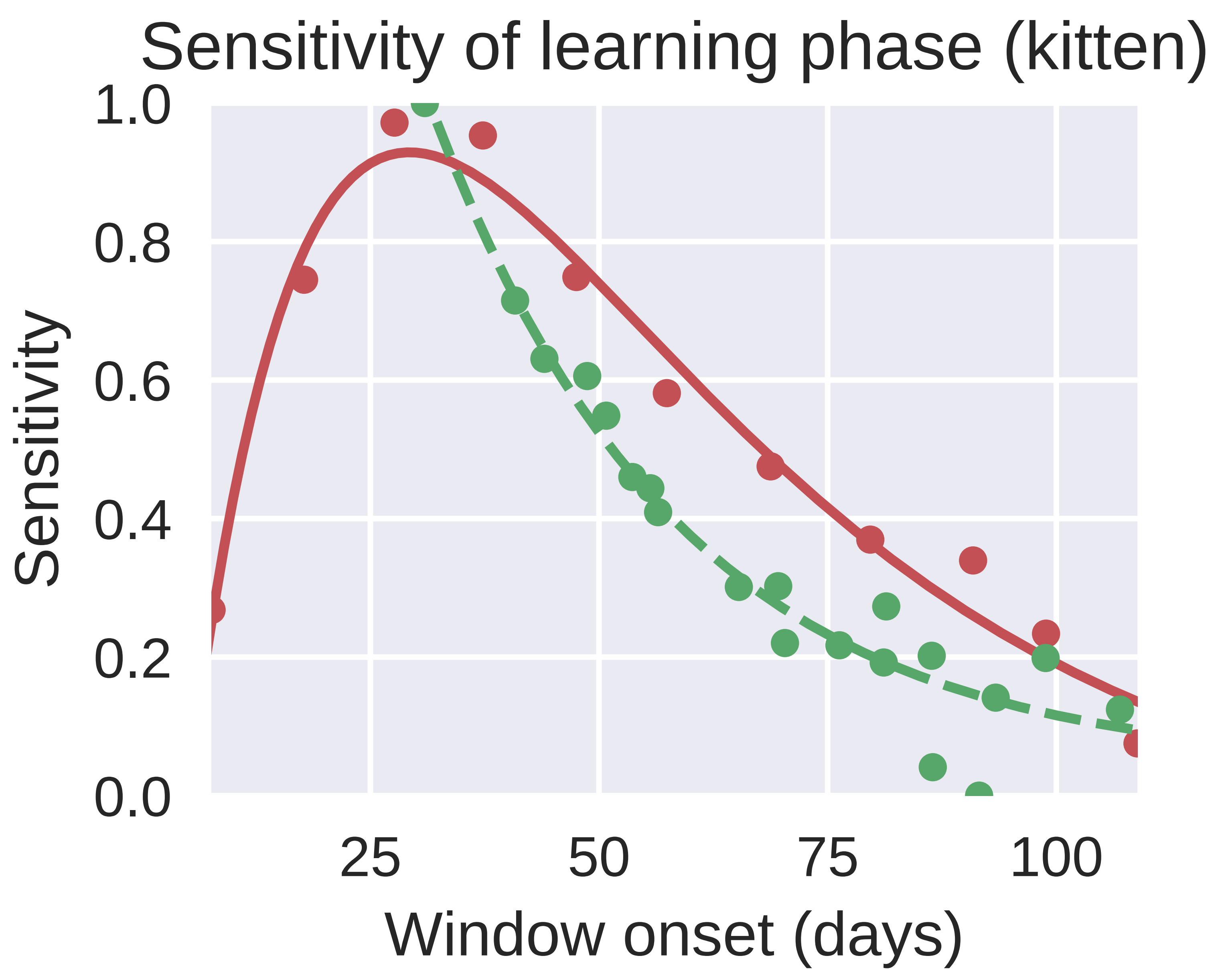


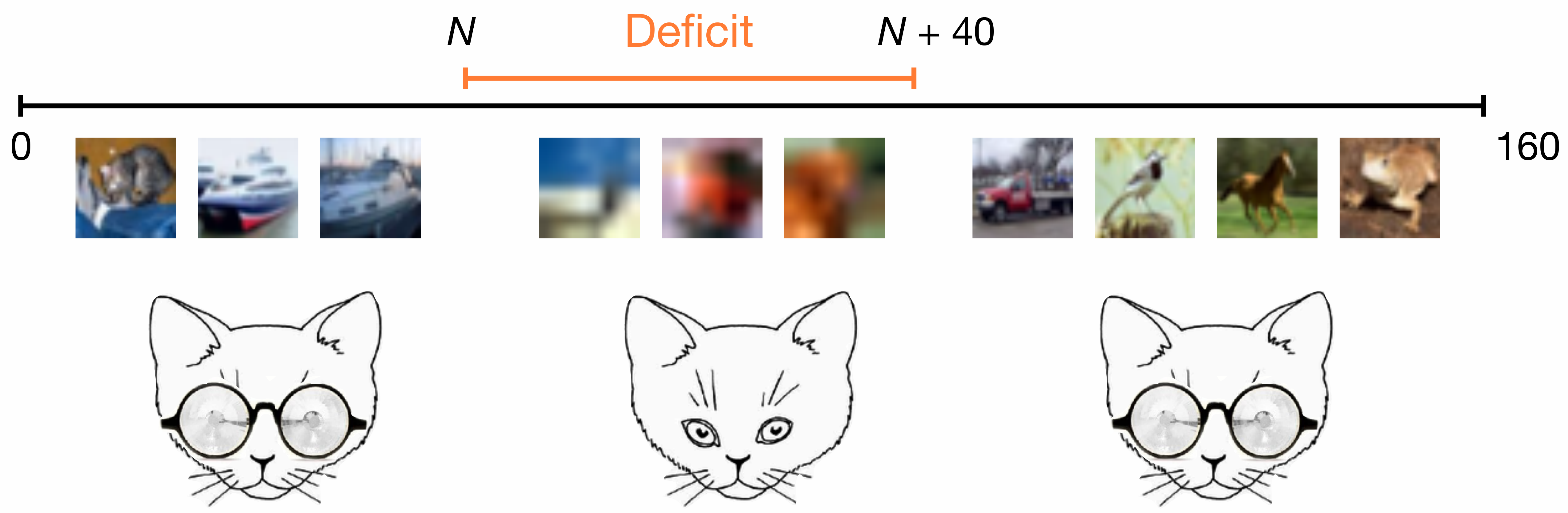


A., Rovere, Soatto, Critical Learning Periods in Deep Networks, 2018

Sensitivity to deficits

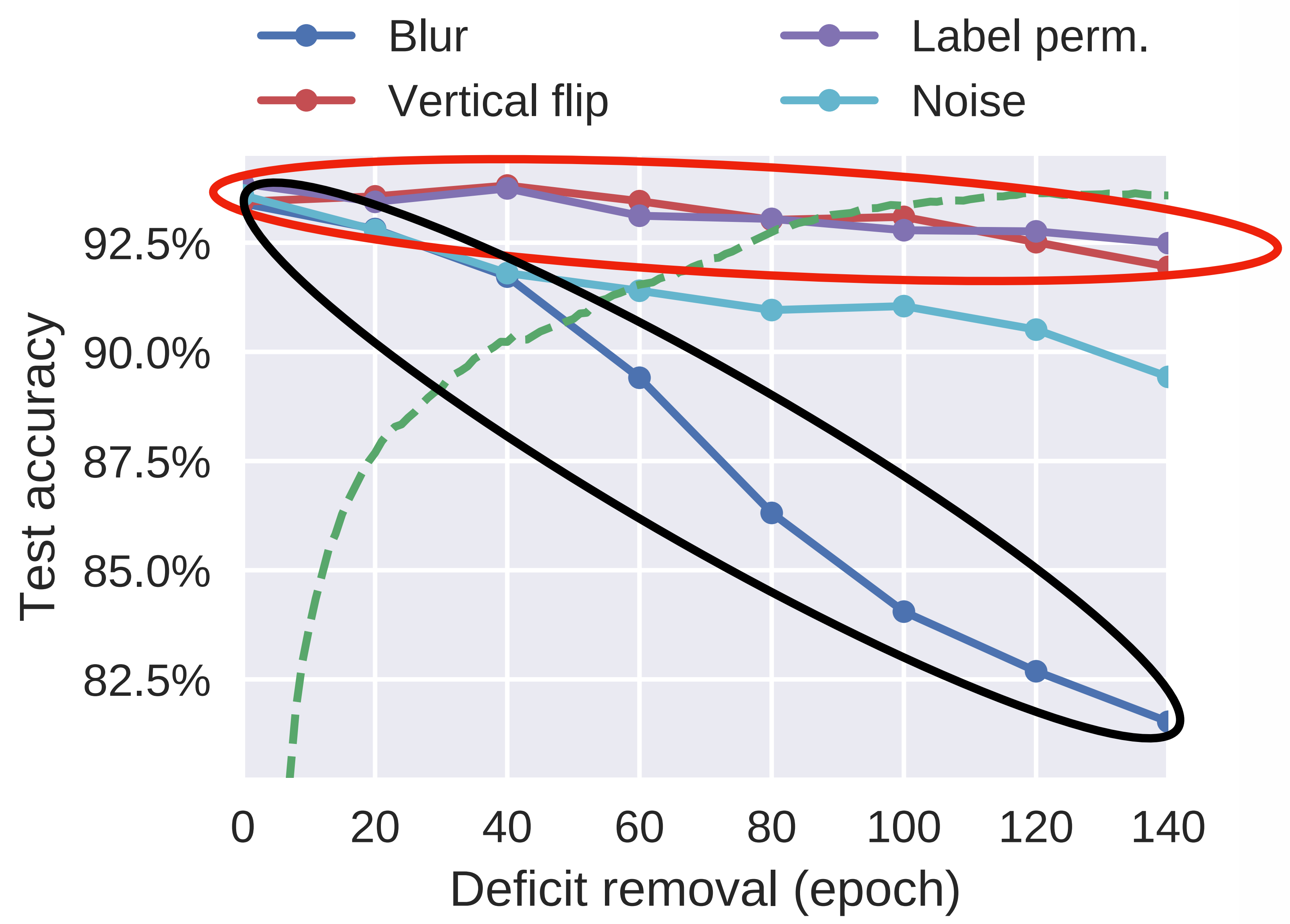






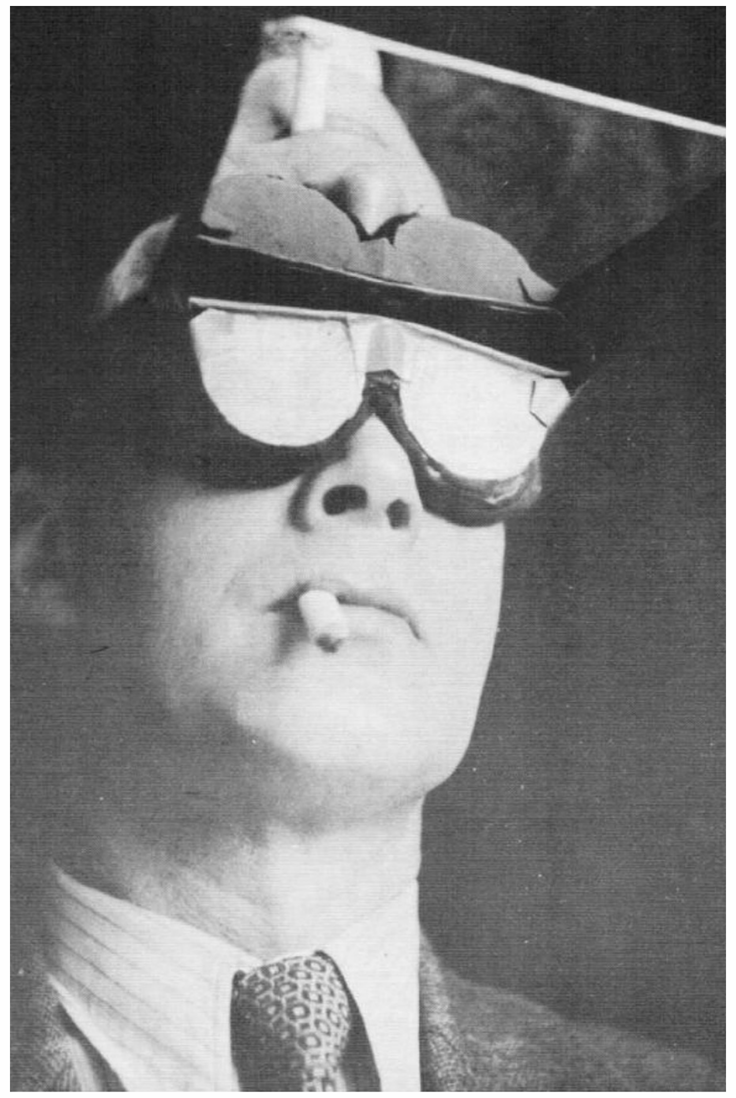
High-level deficits do not have a critical period

Deficits that only change high-level statistics of the data do not show a critical period.



High-level deficits do not exhibit a critical period

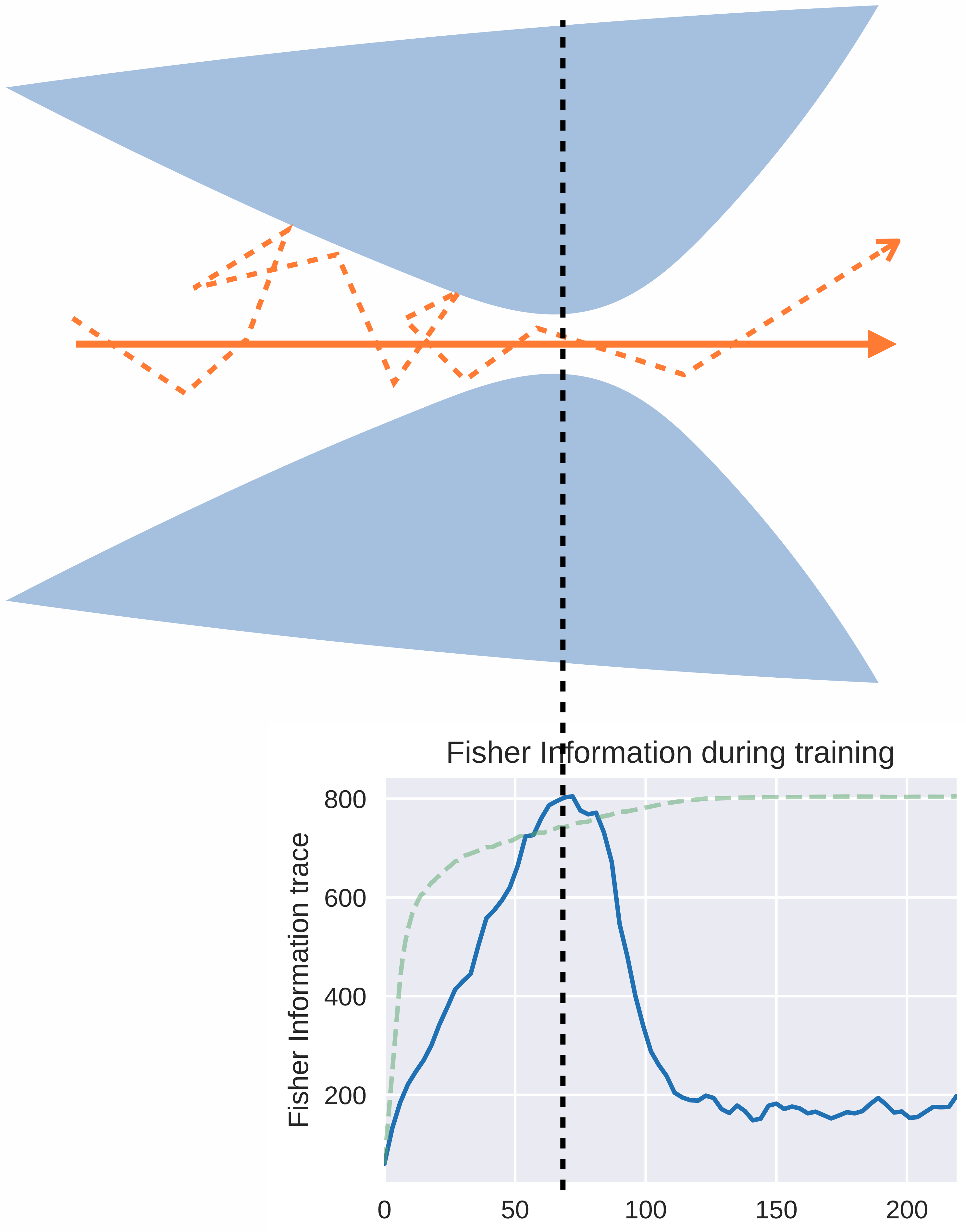
Low-level deficit exhibit a critical period



Achille, Rovere, Soatto, Critical Learning Periods in Deep Networks, 2018
Picture from “The world is upside down” — The Innsbruck Goggle Experiments of Theodor Erismann and Ivo Kohler, Sachse et al.

Information is physical

How can the Fisher Information affect the learning dynamics?

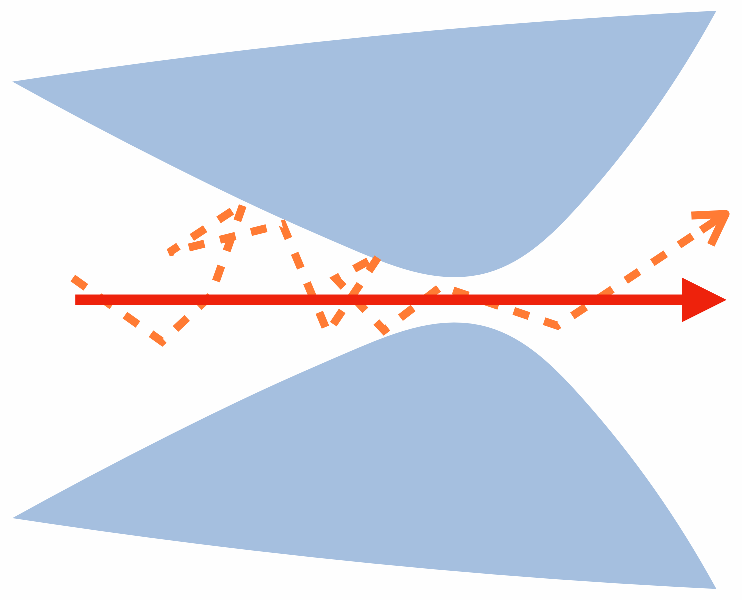


Idea: When using SGD, the Fisher Information adds a drag term controlled by the batch size

$$V_{\text{eff}} = \underbrace{U}_{\text{Real loss}} + \underbrace{\frac{k}{B} \log |F|}_{\text{Drag term}}$$

SGD MINIMIZES THE FISHER INFORMATION OF THE WIGHTS (INDUCTIVE BIAS OF SGD)

Path Integral Approximation and Task Reachability



SGD EFFECTIVELY MINIMIZES THE IBL FOR THE WEIGHTS

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

Reachability

Static part

Dynamic part

Information Lagrangian

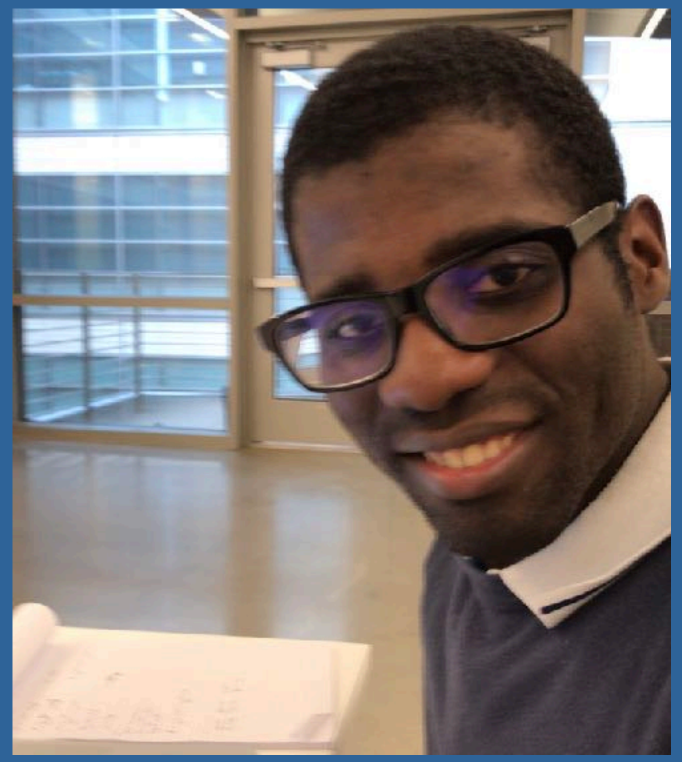
Critical Periods

Achille, Mbeng, Soatto, The Dynamic Distance Between Tasks, NeurIPS Workshop 2018

THANKS!

Matteo Rovere

Giovanni Paolini



Glen Mbeng

Stefano Soatto