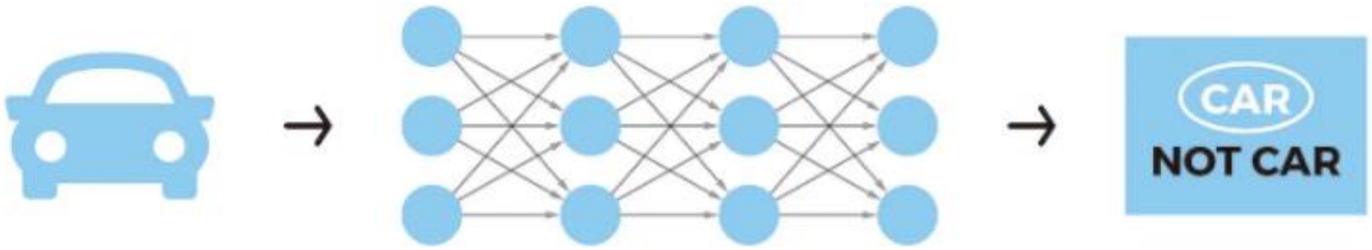
Understanding Deep Learning via Deep Learning? A Phenomenological Approach

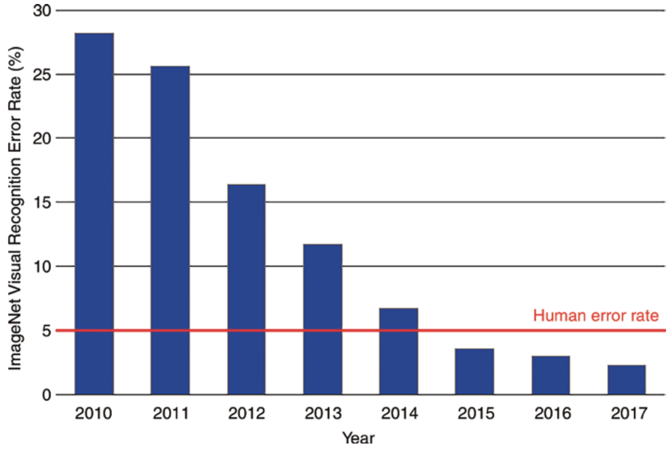
Weijie Su

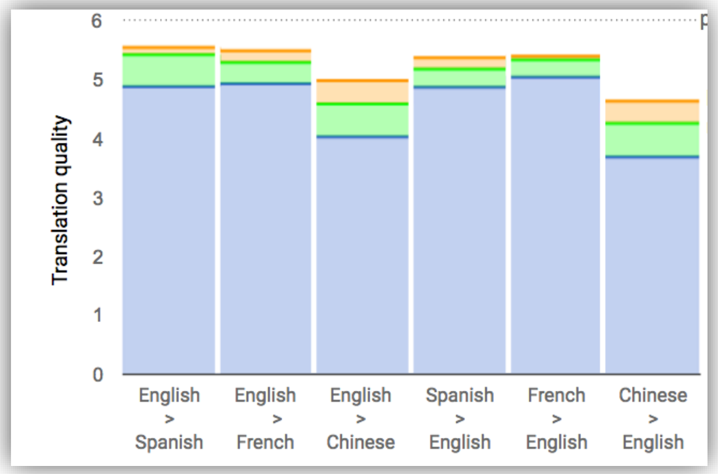
Wharton
University of Pennsylvania

Penn Engineering
University of Pennsylvania

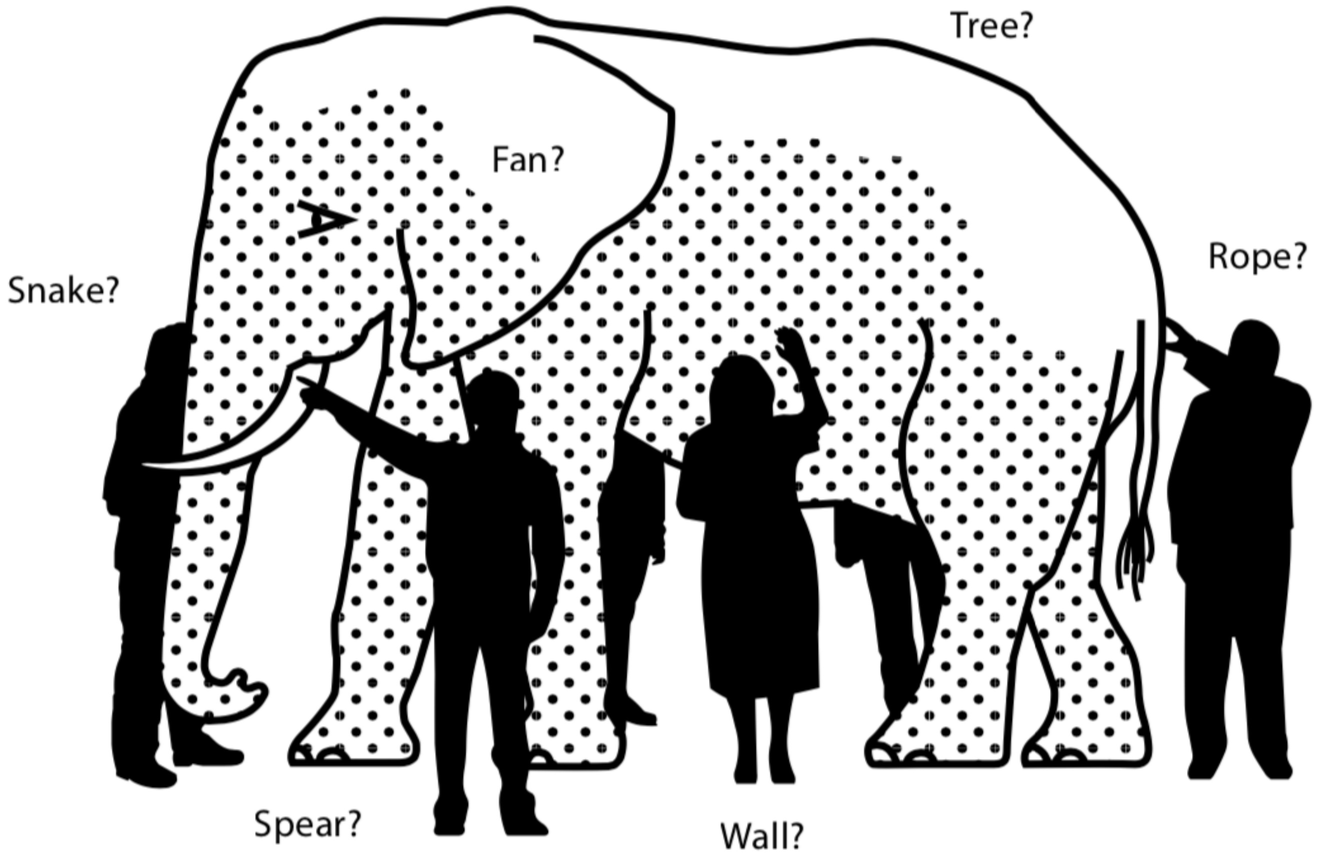
Deep learning







Elephant in the room, from a theoretical viewpoint…

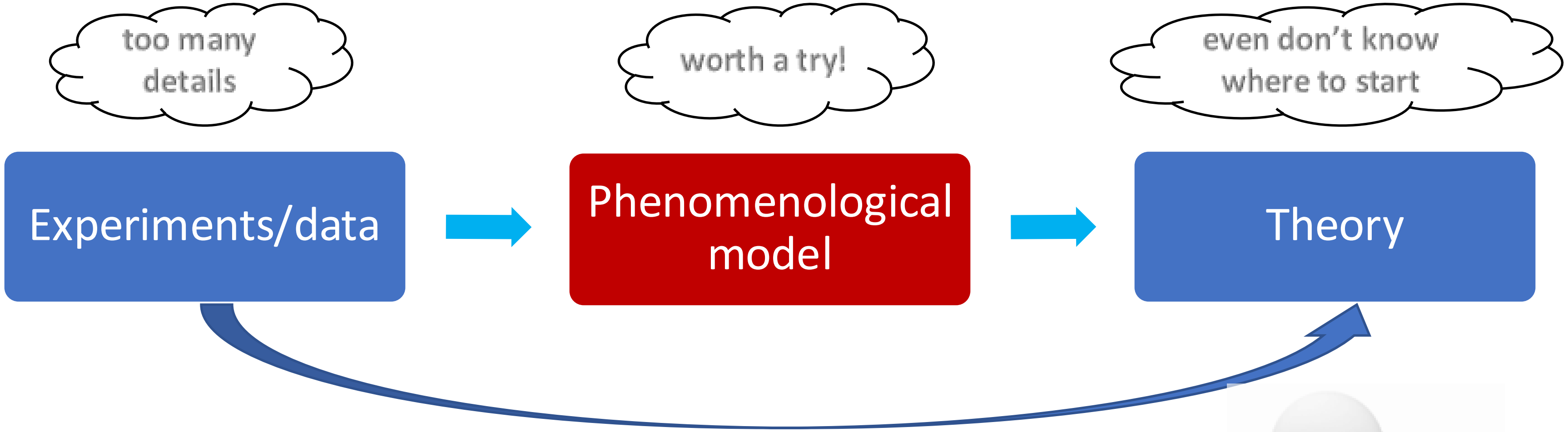


The fundamental questions in deep learning

Leo Breiman

- Why don’t heavily parameterized neural networks overfit the data?
- What is the effective number of parameters?
- Why doesn’t backpropagation get stuck in poor local minima with low value of the loss function, yet bad test error?

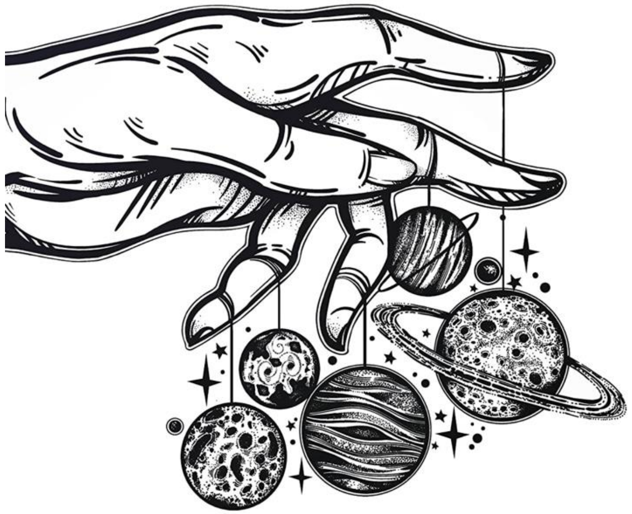
A *phenomenological* approach for deep learning?

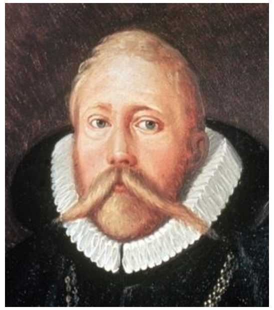


Ideally, we want

- Big picture instead of complex details
- Intuitive, though may not be rigorous
- Guides future research

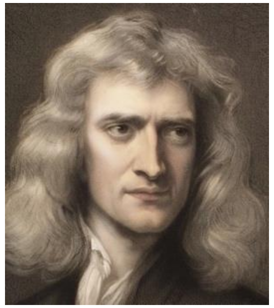
Examples of phenomenological models



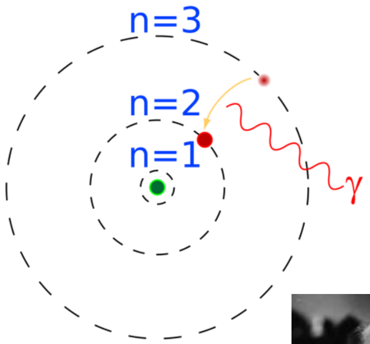


Tycho Brahe

Johannes Kepler

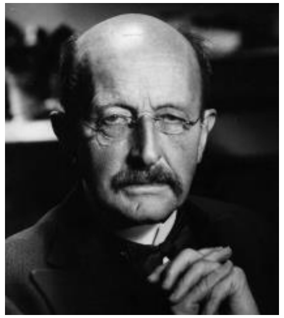


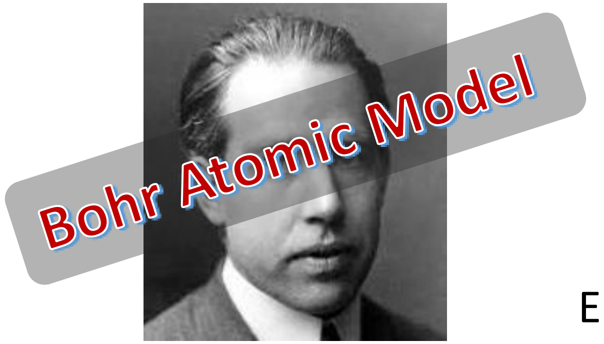
Isaac Newton



- Simple, though not rigorous
- Offers a big picture
- Guides future research

Max Planck





Niels Bohr

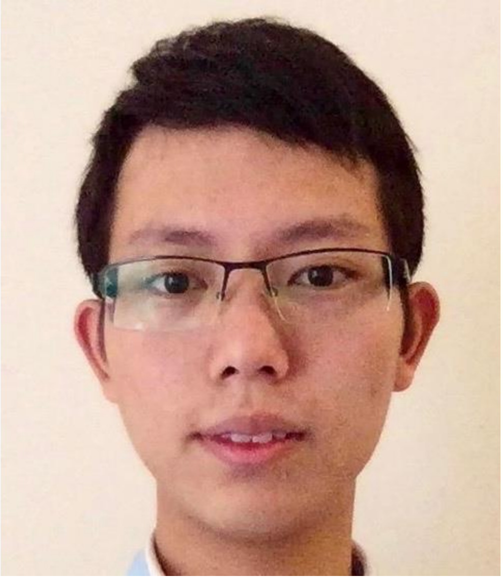
Erwin Schrödinger

Overview of the talk

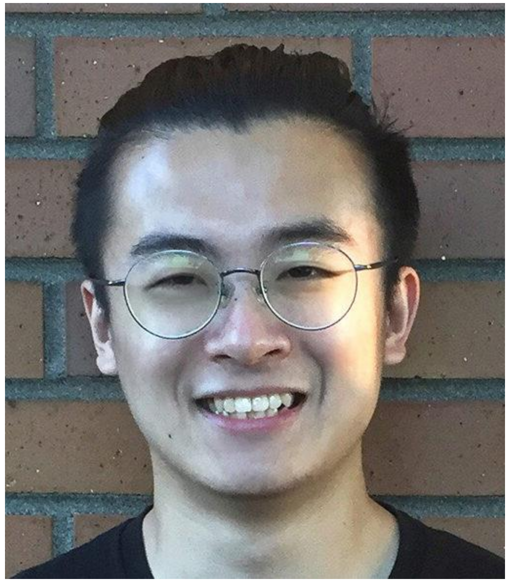
1. Introducing Local Elasticity
2. Evidence of Local Elasticity
3. Neurashed: the Origin of Local Elasticity?

Team

Hangfeng He (Penn CS)



Shuxiao Chen (Wharton Stats)

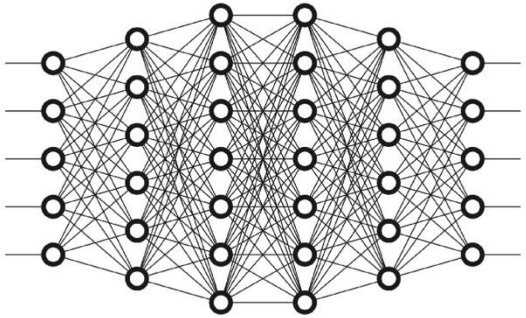


Zhun Deng (Harvard CS)

Part 1

Introducing Local Elasticity

They are similar, though both complex



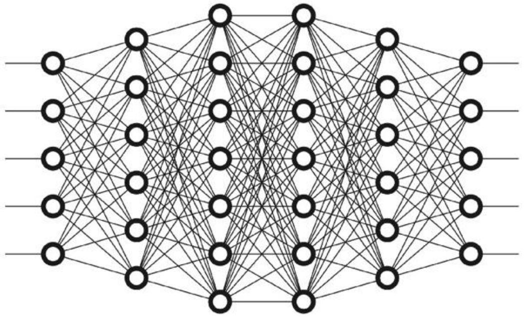
- 10^{14} synapses and lives for 10^{9} seconds (Hinton)
- Memorization yet w/ innovation
- Iterative learning
- Knowledge distillation

- 10^{7} parameters, trained on 10^{5} images
- Zero training error yet w/ generalization
- Iterative learning
- Compression

Learn by analogy

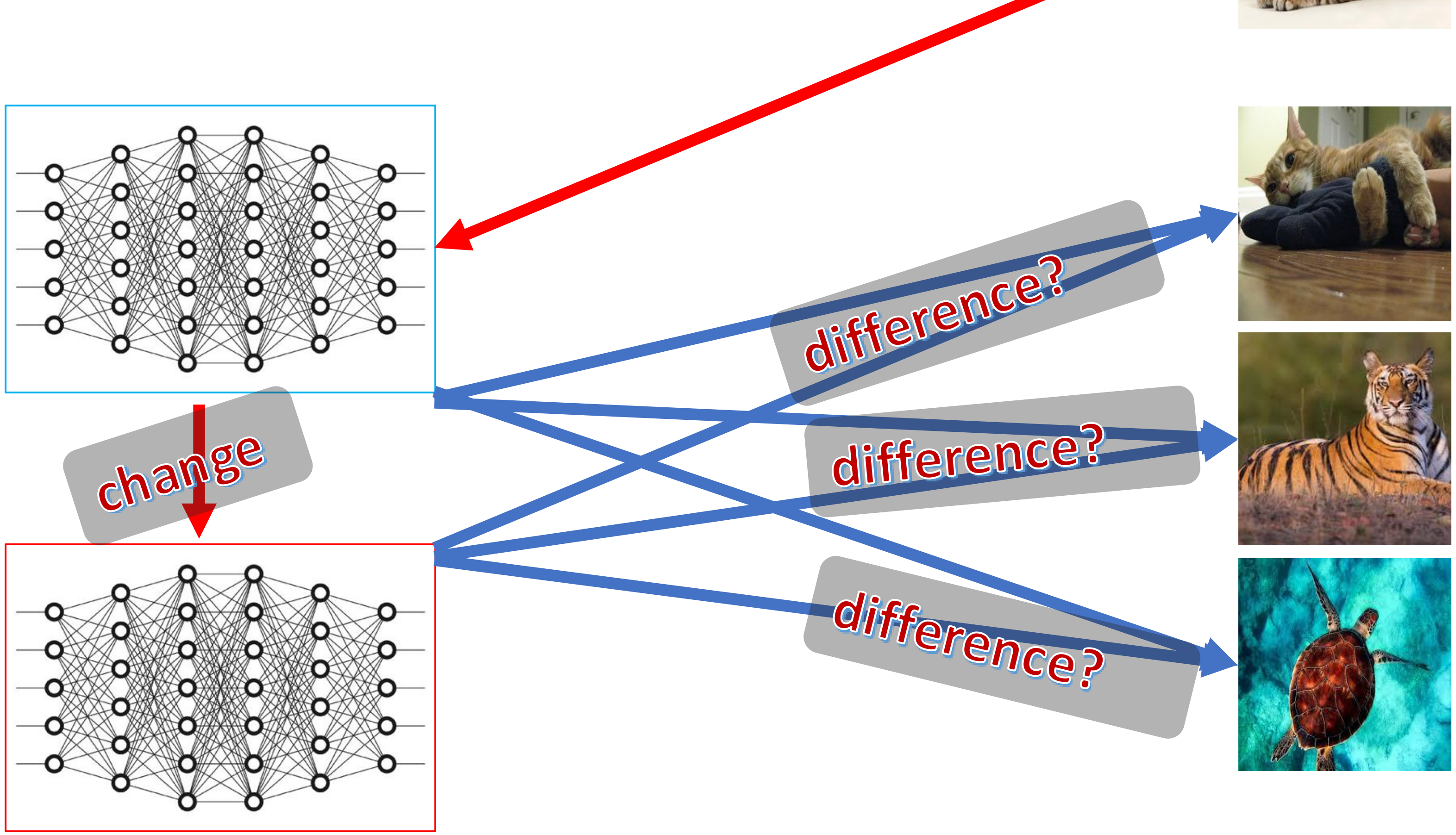
- We humans improve our understanding of things *related* to what we see early
- Learning French might affect English, but not math

How about neural networks?



Motivating question

How does the update of weights using SGD at an image of cat impact the prediction at another image?



A measure of the prediction change

- Let $f(\boldsymbol{x}, \boldsymbol{w})$ be the prediction of neural networks with weights $\boldsymbol{w}$

- Use SGD to update $\boldsymbol{w}$ with example $(\boldsymbol{x}, y)$ and loss function $\mathcal{L}(f, y)$:

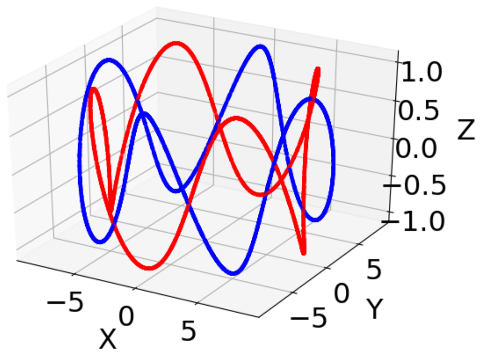
$$\boldsymbol{w}^{+} = \boldsymbol{w} - \gamma \frac{d\mathcal{L}(f(\boldsymbol{x}, \ \boldsymbol{w}), \ y)}{d\boldsymbol{w}} = \mathbf{w} - \gamma \frac{\partial\mathcal{L}(f(\boldsymbol{x}, \ \boldsymbol{w}), \ y)}{\partial f} \cdot \frac{\partial f(\boldsymbol{x}, \ \boldsymbol{w})}{\partial \boldsymbol{w}}$$

- Define the relative change ratio

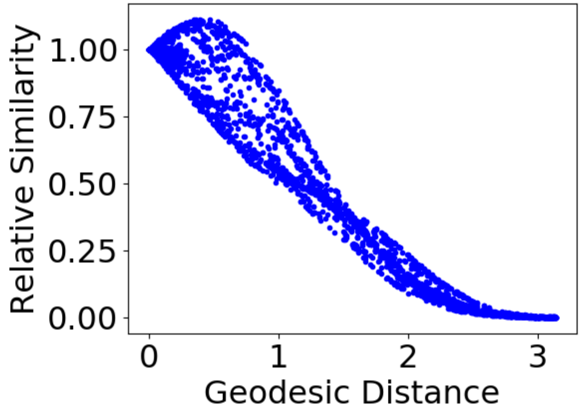
$$S_{rel}(\boldsymbol{x}, \boldsymbol{x}') := \frac{|f(\boldsymbol{x}', \ \boldsymbol{w}^{+}) - f(\boldsymbol{x}', \ \boldsymbol{w})|}{|f(\boldsymbol{x}, \ \boldsymbol{w}^{+}) - f(\boldsymbol{x}, \ \boldsymbol{w})|}$$

- Near optimal $\boldsymbol{w}$

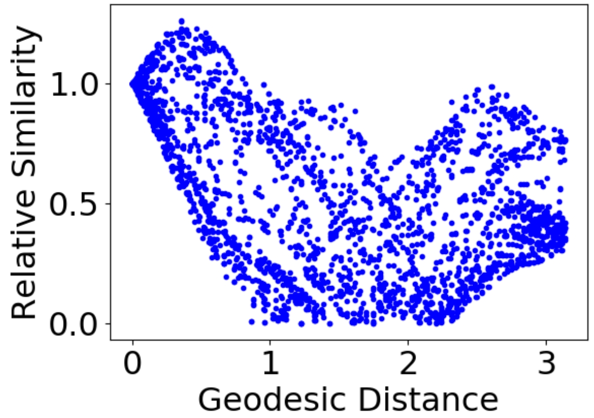
Toy manifolds



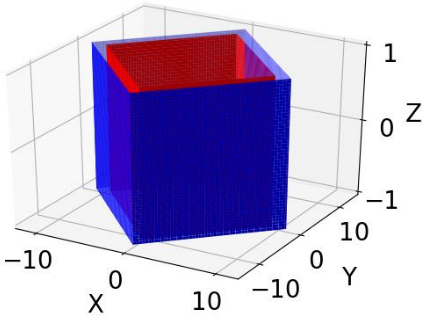
Double helix



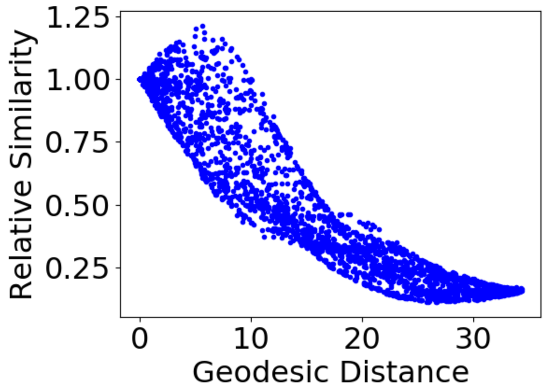
Two-layer neural nets fitting helix



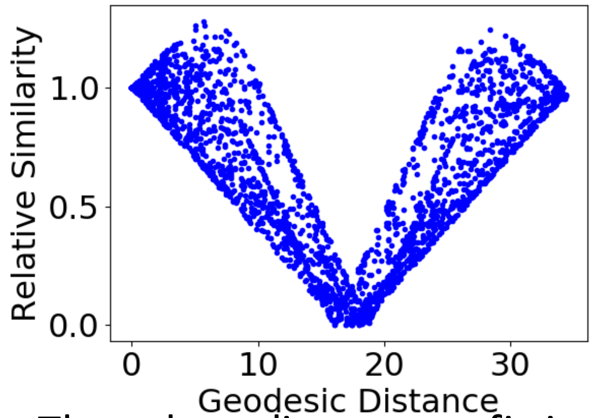
Two-layer linear nets fitting helix



Two folded boxes

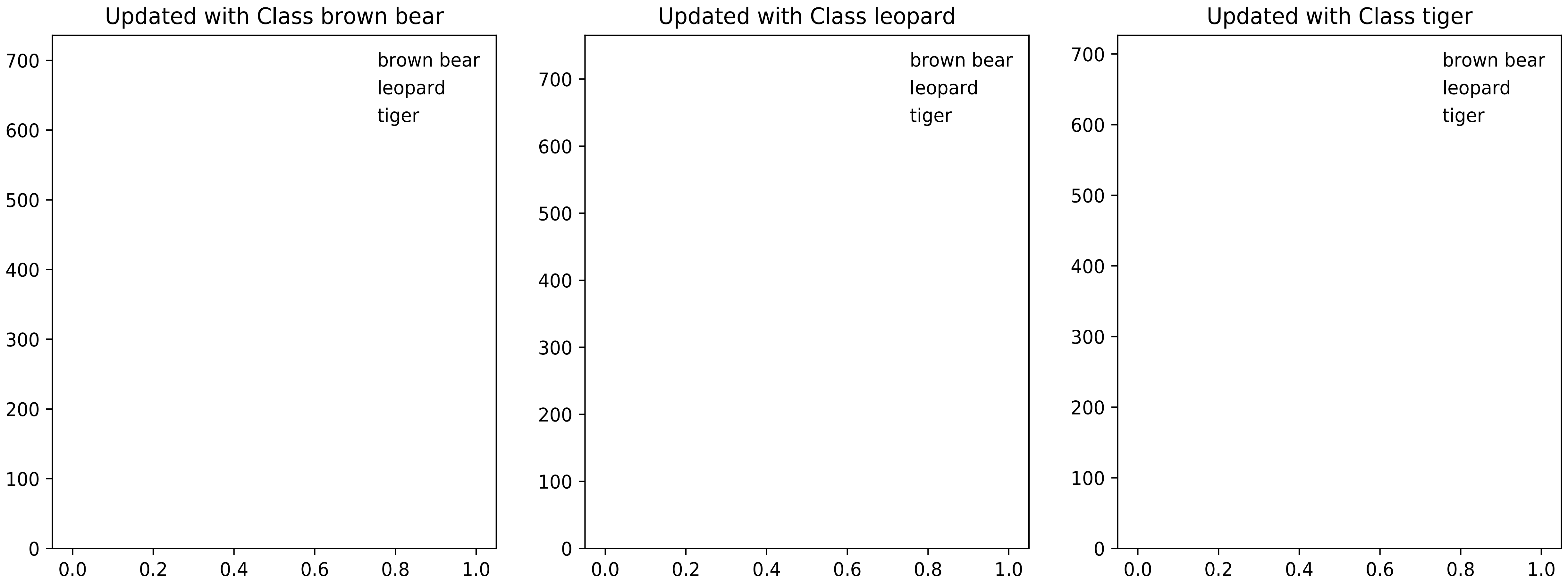


Three-layer neural nets fitting boxes



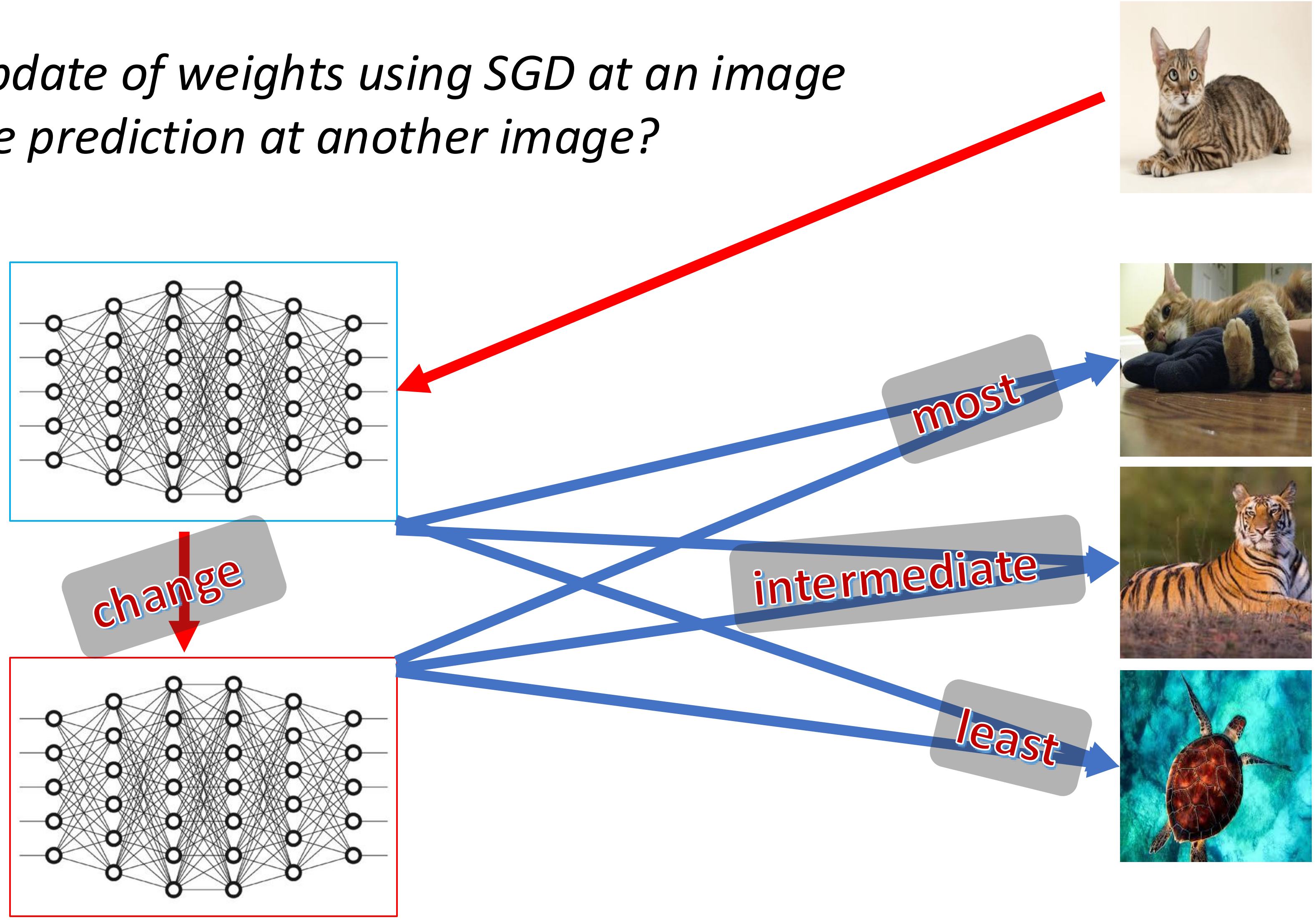
Three-layer linear nets fitting boxes

Experiments on VGG19

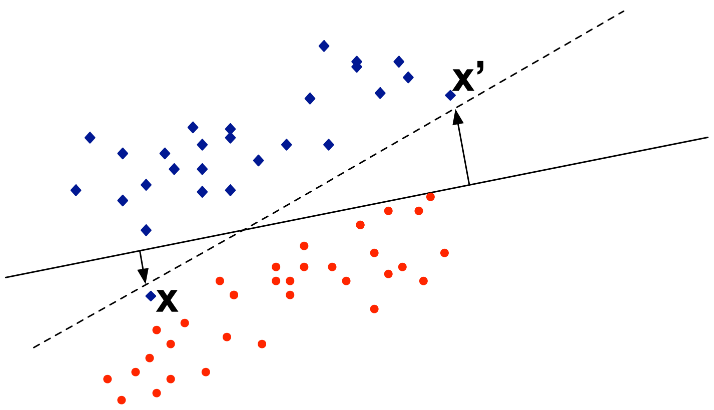


Return to the motivating question

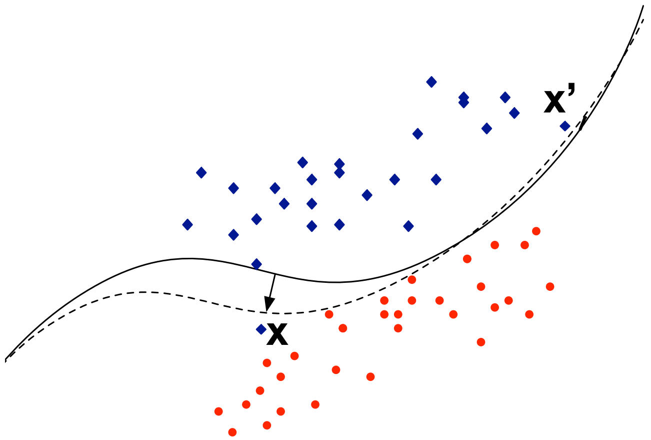
How does the update of weights using SGD at an image of cat impact the prediction at another image?



Hypothesis of *local elasticity* in neural networks

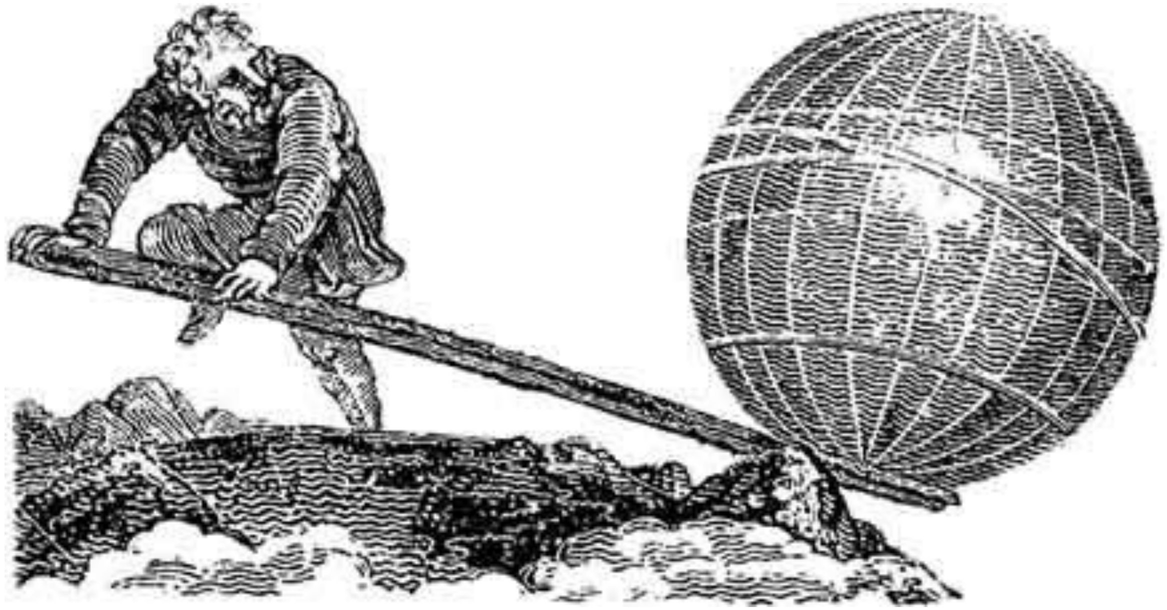


Linear classifier updated by SGD



Neural networks updated by SGD

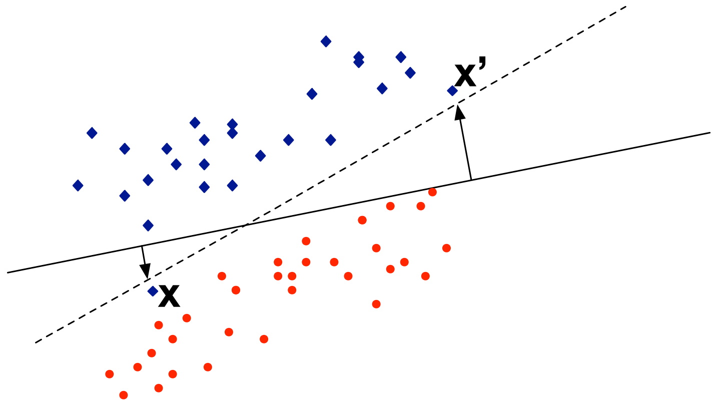
Why no local elasticity in linear classifiers?



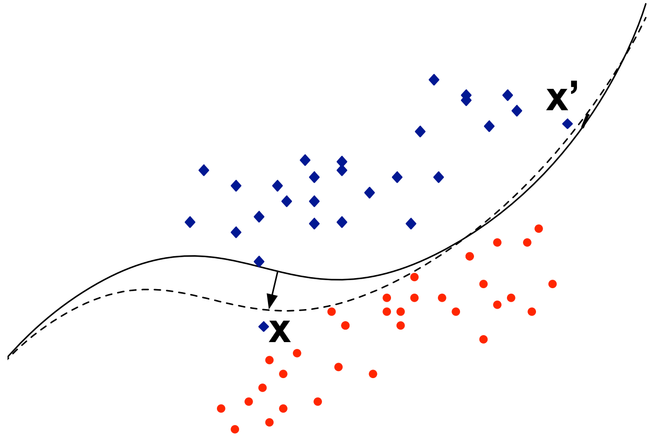
Give me a place to stand and I shall move the earth

--- Archimedes

Hypothesis of *local elasticity* in neural networks



Linear classifier updated by SGD



Neural networks updated by SGD

- ***Locality***: relative change is large when $\boldsymbol{x}$ and $\boldsymbol{x}'$ are close/similar (akin to the nearest neighbor)
- ***Elasticity***: relative change decreases *gradually* and *smoothly* (as opposed to abruptly) when $\boldsymbol{x}'$ moves away from $\boldsymbol{x}$

- Kicks in the late phase of training (neural collapse, Papyan, Han, and Donoho, 2020)
- Related to influence function (Koh and Liang, 2017)

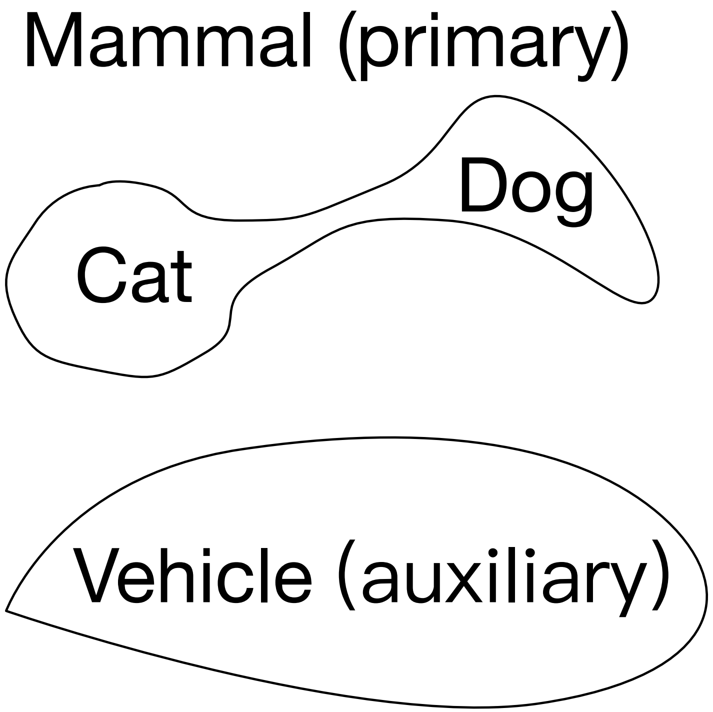
Any other locally elastic classifier?

Part 2

Evidence of Local Elasticity

Semi-supervised learning via local elasticity

- Clustering via local elasticity
  - Primary dataset $\mathcal{P} = \{x_i\}_{i=1}^{n}$
  - Auxiliary dataset $\mathcal{A} = \{\tilde{x}_j\}_{j=1}^{m}$
  - Classifier $f(\boldsymbol{x}, \boldsymbol{w})$, loss function $\mathcal{L}(f, y)$
  - Initial weights $\boldsymbol{w}_0$, learning rate η_t



Use relative change or something else
*as **proxy for the similarity** of two images!*

The algorithm

Algorithm 1 The Local Elasticity Based Clustering Algorithm

combine $\mathcal{D} = \{(\boldsymbol{x}_i, y_i = 1) \text{ for } \boldsymbol{x}_i \in \mathcal{P}\} \bigcup \{(\boldsymbol{x}_i, y_i = -1) \text{ for } \boldsymbol{x}_i \in \mathcal{A}\}$
set S to $n \times n$ matrix of all zeros
for $t = 1$ to $n + m$ **do**
    sample $(\boldsymbol{x}, y)$ from $\mathcal{D}$ w/o replacement
    $\boldsymbol{w_t} = \texttt{SGD}(\boldsymbol{w}_{t-1}, \boldsymbol{x}, y, f, \mathcal{L}, \eta_t)$
    if $y = 1$ **then**
        $\boldsymbol{p}_t = \texttt{Predict}(\boldsymbol{w}_t, \mathcal{P}, f)$
        find $1 \leqslant i \leqslant n$ such that $\boldsymbol{x} = \boldsymbol{x}_i \in \mathcal{P}$
        if $o = \texttt{relative}$ **then**
            $\boldsymbol{s}_t = \frac{|\boldsymbol{p}_t - \boldsymbol{p}_{t-1}|}{|\boldsymbol{p}_t(i) - \boldsymbol{p}_{t-1}(i)|}$
        else
            $g_t = \texttt{GetGradient}(\boldsymbol{w}_{t-1}, \boldsymbol{x}, y, f, \mathcal{L})$
            $\boldsymbol{s}_t = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-1}}{-\eta_t \times g_t}$
        end if
    end if
    set the ith row $S(i, :) = \boldsymbol{s}_t$
end for
$S_{\mathrm{symm}} = \frac{1}{2}(S + S^\top)$
$\boldsymbol{y}_{\text{subclass}} = \text{Clustering}(S_{\mathrm{symm}})$
return $\boldsymbol{y}_{\text{subclass}}$

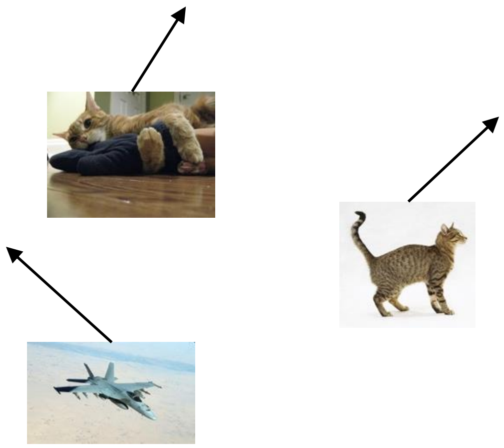
Results on MNIST

Primary Examples	5 vs 8	4 vs 9	7 vs 9	5 vs 9	3 vs 5	3 vs 8
Auxiliary Examples	3, 9	5, 7	4, 5	4, 8	8, 9	5
K-means	50.4	54.5	55.5	56.3	69.0	76.5
PCA + K-means	50.4	54.5	55.7	56.5	70.7	76.4
ℓ_2-relative(linear)	51.0	54.6	51.3	58.8	58.3	58.7
ℓ_2-kernelized (linear)	50.1	55.5	55.5	56.3	69.3	76.1
ℓ_2-relative (FNN)	**75.9**	55.6	62.5	89.3	50.3	74.7
ℓ_2-kernelized (FNN)	71.0	63.8	64.6	67.8	71.5	78.8
ℓ_2-relative (CNN)	54.2	53.7	89.1	50.1	50.1	83.0
ℓ_2-kernelized (CNN)	64.1	**69.5**	**91.3**	**97.6**	**75.3**	**87.4**
ℓ_2-relative (ResNet)	50.7	55.0	55.5	78.3	52.3	52.3
ℓ_2-kernelized (ResNet)	50.2	60.4	54.8	76.3	66.9	68.8

Yet another measure for local elasticity

- $$f(\boldsymbol{x}', \boldsymbol{w}^+) - f(\boldsymbol{x}', \boldsymbol{w}) = f\left(\boldsymbol{x}', \boldsymbol{w} - \gamma \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}\right) - f(\boldsymbol{x}', \boldsymbol{w})$$
$$\approx f(\boldsymbol{x}', \boldsymbol{w}) - \left\langle \frac{\partial f(\boldsymbol{x}', \boldsymbol{w})}{\partial \boldsymbol{w}}, \gamma \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}} \right\rangle - f(\boldsymbol{x}', \boldsymbol{w})$$
$$= -\gamma \frac{\partial \mathcal{L}}{\partial f} \left\langle \frac{\partial f(\boldsymbol{x}', \boldsymbol{w})}{\partial \boldsymbol{w}}, \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}} \right\rangle$$

- Kernelized change
$$S_{ker}(\boldsymbol{x}, \boldsymbol{x}') := \frac{f(\boldsymbol{x}', \boldsymbol{w}^+) - f(\boldsymbol{x}', \boldsymbol{w})}{-\gamma \frac{\partial \mathcal{L}(f(\boldsymbol{x}, \boldsymbol{w}), y)}{\partial f}} \approx \left\langle \frac{\partial f(\boldsymbol{x}', \boldsymbol{w})}{\partial \boldsymbol{w}}, \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}} \right\rangle$$



- In late stages, large inner product of two cats: learning at the tabby cat leads to improvement at the tiger cat
- Small inner product of the tabby cat and the warplane: learning at the tabby cat does not affect the warplane much

Connection with neural tangent kernel

$$\text{NTK}(\boldsymbol{x},\ \boldsymbol{x}') = \left\langle \frac{\partial f(\boldsymbol{x}',\boldsymbol{w})}{\partial \boldsymbol{w}},\ \frac{\partial f(\boldsymbol{x},\boldsymbol{w})}{\partial \boldsymbol{w}} \right\rangle, w \sim Gaussian$$

Jacot et al, 2018

Training neural networks using GD ≈ kernel regression

- Infinite width, at very large width
- Special scaling of the weights
- GD instead of SGD

Separation between kernel method and deep learning: Wei et al, 2018; Allen-Zhu and Li, 2020…

Fixed kernel! NTK doesn't adapt to the semantics/labels, as opposed to local elasticity

Label-aware neural tangent kernel

Incorporate label info into NTK!

$$\mathrm{LANTK}(\boldsymbol{x},\ \boldsymbol{x}') = \mathrm{NTK}(\boldsymbol{x},\ \boldsymbol{x}') + \boldsymbol{Z}(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{S})$$

- $\boldsymbol{Z}(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{S})$ is an estimator of (label of x) × (label of x')
- Can be obtained by regressing $y_i y_j$ on (x_i, x_j)

Experiments for label-aware neural tangent kernel

Train-train	frog vs ship	frog vs truck	deer vs ship	dog vs truck	bird vs truck	deer vs truck
NN-init	58.37	55.07	57.50	54.75	52.93	54.86
NN-trained	**71.99** ↑	**68.36** ↑	**69.98** ↑	**66.35** ↑	**63.99** ↑	**65.96** ↑
NTK	63.83	58.31	62.43	58.05	55.01	58.02
LANTK	**66.62** ↑	**60.57** ↑	**64.90** ↑	**59.75** ↑	**55.94** ↑	**59.59** ↑
Test-train	**frog vs ship**	**frog vs truck**	**deer vs ship**	**dog vs truck**	**bird vs truck**	**deer vs truck**
NN-init	58.31	55.06	57.64	54.62	52.93	54.94
NN-trained	**71.45** ↑	**67.91** ↑	**69.73** ↑	**65.80** ↑	**63.58** ↑	**65.53** ↑
NTK	63.76	58.30	62.67	57.84	55.00	58.14
LANTK	**66.53** ↑	**60.08** ↑	**65.20** ↑	**59.54** ↑	**55.97** ↑	**59.77** ↑

Table 3: Strength of local elasticity in binary classification tasks on CIFAR-10. The training makes NNs more locally elastic, and LANTK successfully simulates this behavior.

Stability and generalization

Journal of Machine Learning Research 2 (2002) 499-526 Submitted 7/01; Published 3/02

Stability and Generalization

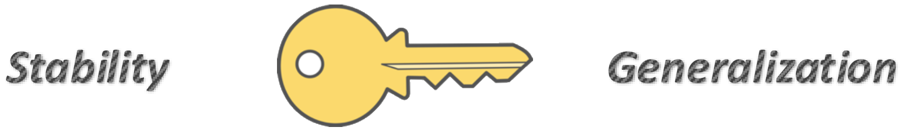
Olivier Bousquet BOUSQUET@CMAPX.POLYTECHNIQUE.FR
CMAP, Ecole Polytechnique
F-91128 Palaiseau, FRANCE

André Elisseeff ANDRE.ELISSEEFF@BIOWULF.COM
BIOwulf Technologies
305 Broadway,
New-York, NY 10007

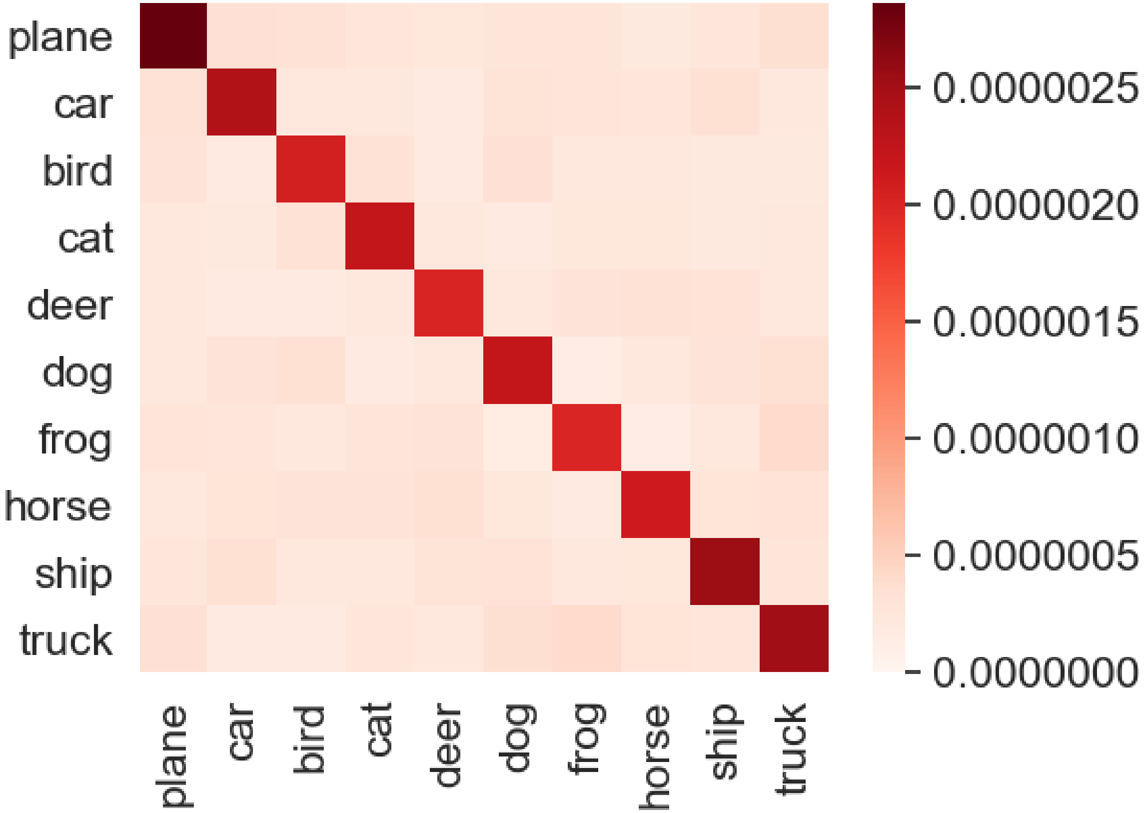
Editor: Dana Ron

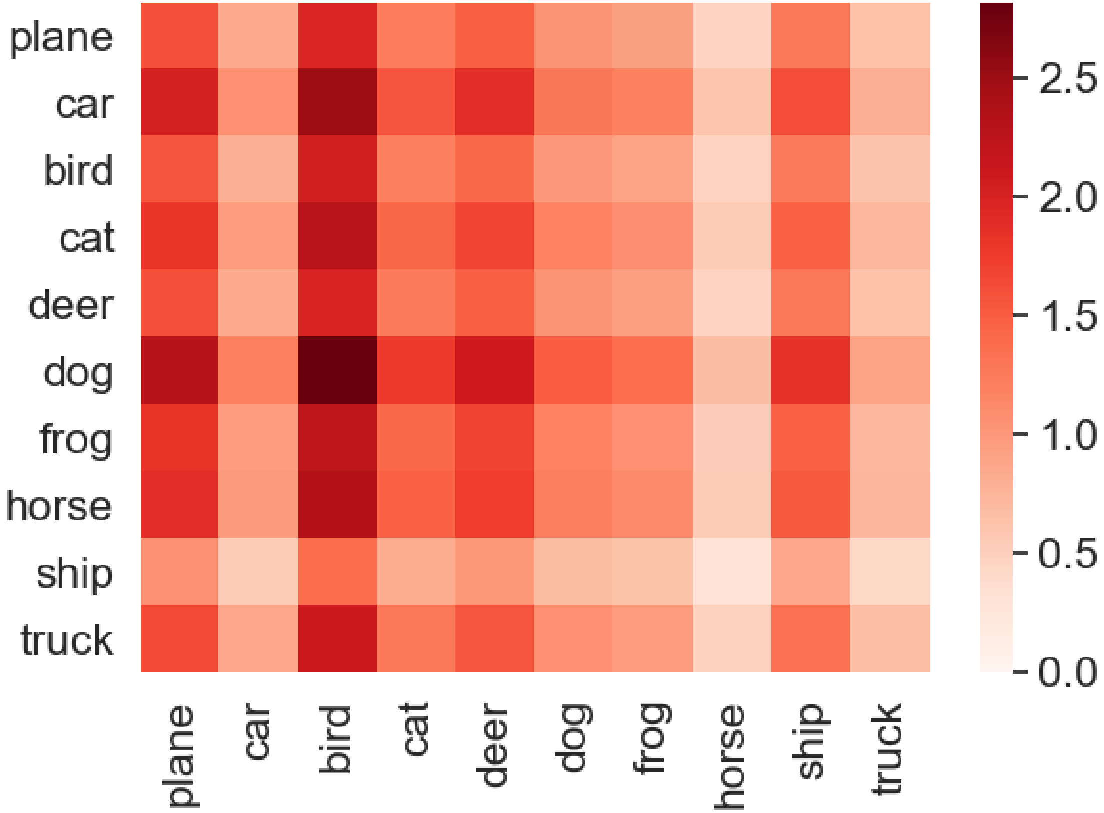
Abstract

We define notions of stability for learning algorithms and show how to use these notions to derive generalization error bounds based on the empirical error and the leave-one-out error. The methods we use can be applied in the regression framework as well as in the classification one when the classifier is obtained by thresholding a real-valued function. We study the stability properties of large classes of learning algorithms such as regularization based algorithms. In particular we focus on Hilbert space regularization and Kullback-Leibler regularization. We demonstrate how to apply the results to SVM for regression and classification.



Uniform stability too pessimistic





Related to the long tail theory of Feldman, 2019

Locally elastic stability

Definition 2.1 (Local Elasticity). *For a certain (abstract) distance or divergence $\mathscr{D}$, an algorithm A satisfies Local Elasticity with respect to loss function l and $\mathscr{D}$ if there exists a function $\beta_m(\cdot)$ that is dependent on sample size m, such that*

$$\forall S \in \mathcal{Z}^m,\ \forall i \in [m], z \in \mathcal{Z},\ |l(A_S, z) - l(A_{S^{\setminus i}}, z)| \leqslant \beta_m(\mathscr{D}(z_i, z)).$$

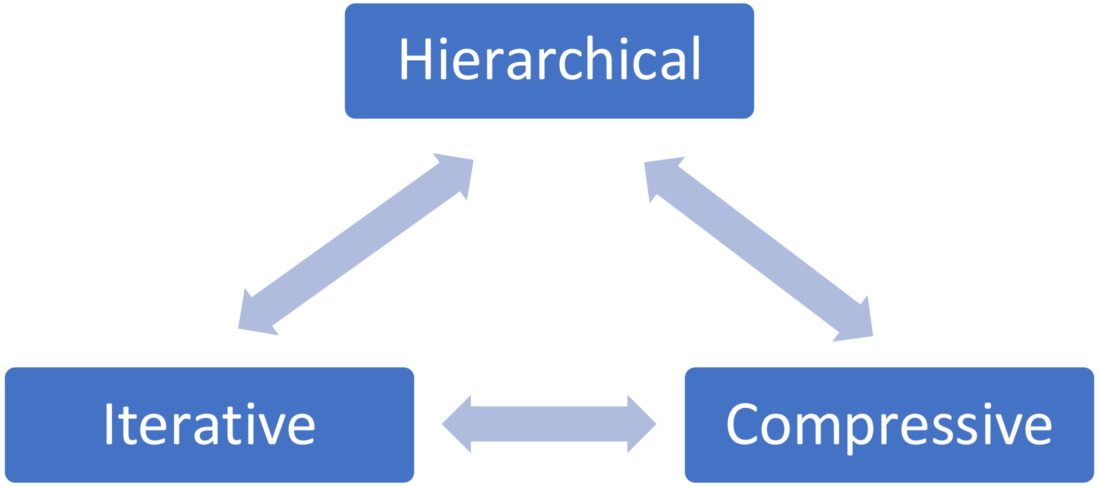
$$\mathbb{P}\Big(\mathbb{E}_z[l(\mathcal{A}_S, z)] \geqslant \frac{1}{m}\sum_{j=1}^{m} l(\mathcal{A}_S, z_j) + \frac{2\sup_{z\in\mathcal{Z}} \mathbb{E}_{z_j}\beta(z, z_j)}{m} + \varepsilon\Big) \leqslant 2\exp\Big(-\frac{m\varepsilon^2}{8\tilde{M}^2}\Big)$$

$E\ \beta(z, z_j)$ is expected to be much smaller than sup $\beta(z, z_j)$

Part 3

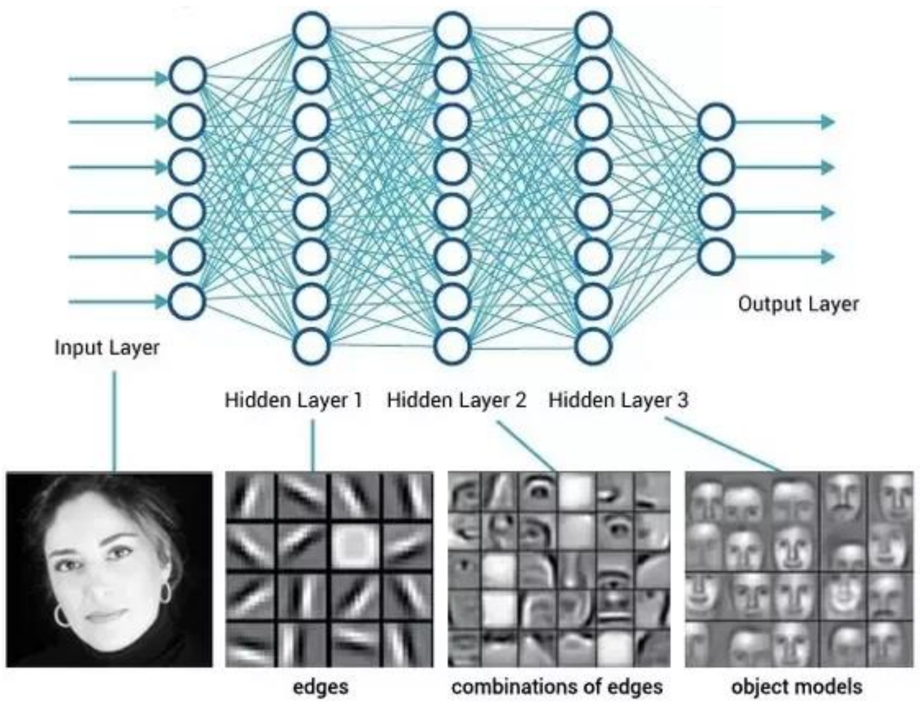
Neurashed: the Origin of Local Elasticity?

Three characteristics of deep learning



- Network architectures and featur es are ***hierarchically*** represented
- SGD and Adam are ***iterative***
- Information is ***compressed*** during training (local elasticity, implicit re gularization, information bottlene ck)

Deep learning is hierarchical feature learning



Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features.

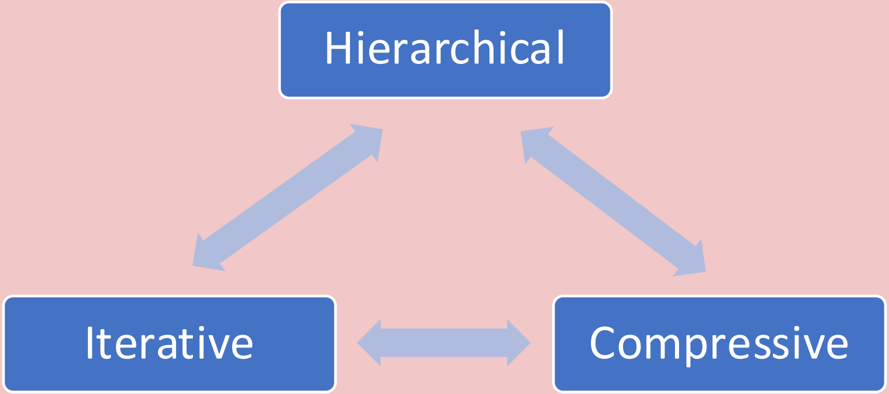
Yoshua Bengio

Credit: https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/

Hypotheses for a phenomenological model

Goal: develop a model showing local elasticity

Guideline:

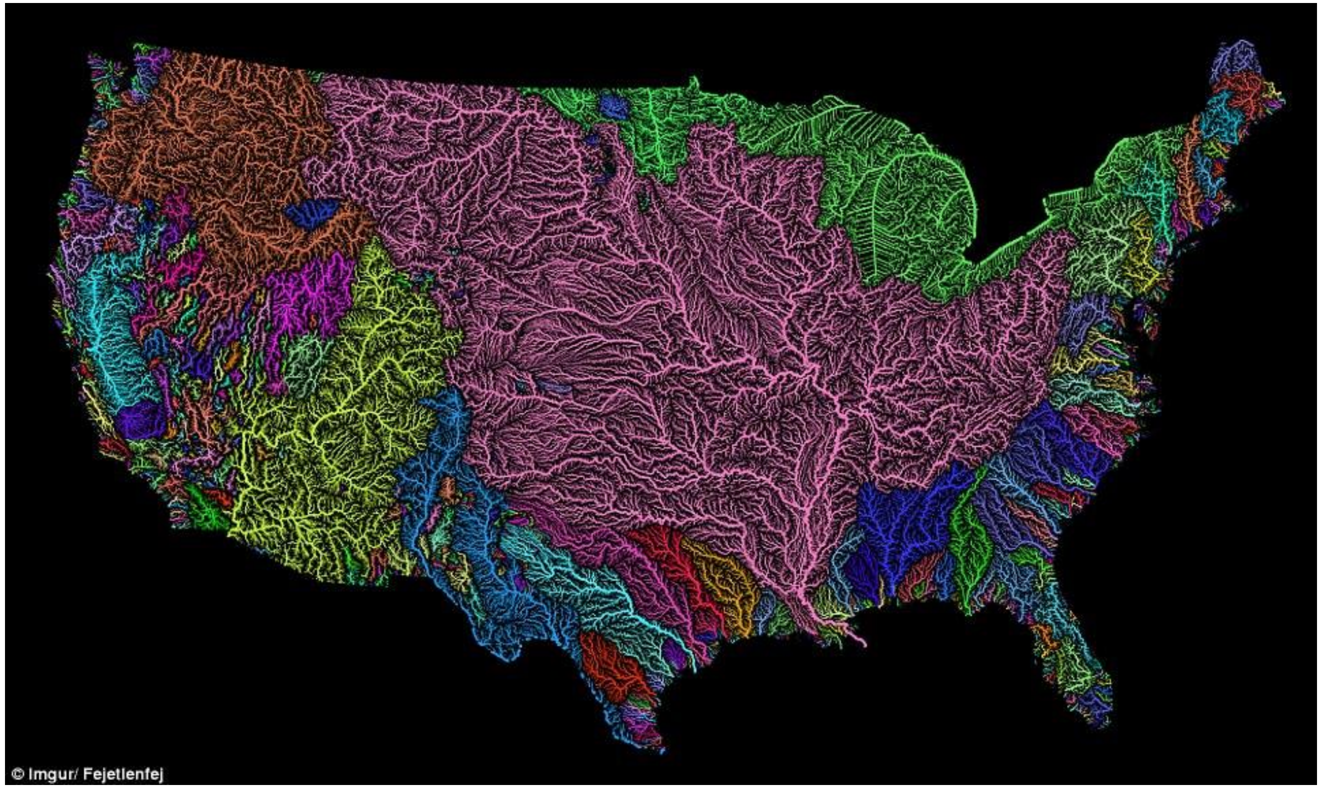


What I cannot create, I do not understand

Richard Feynman

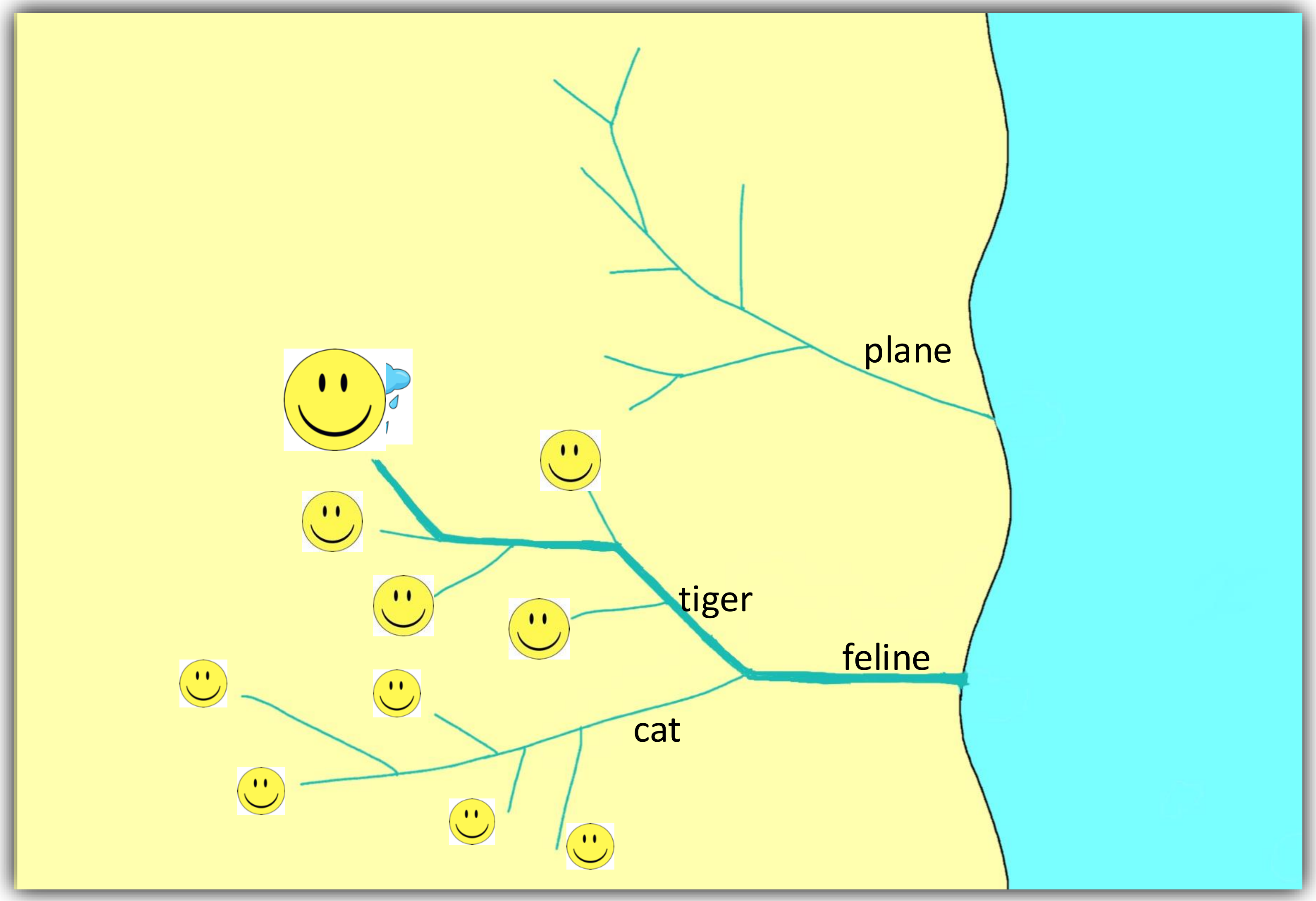
Inspirations: watershed

The watershed map of US



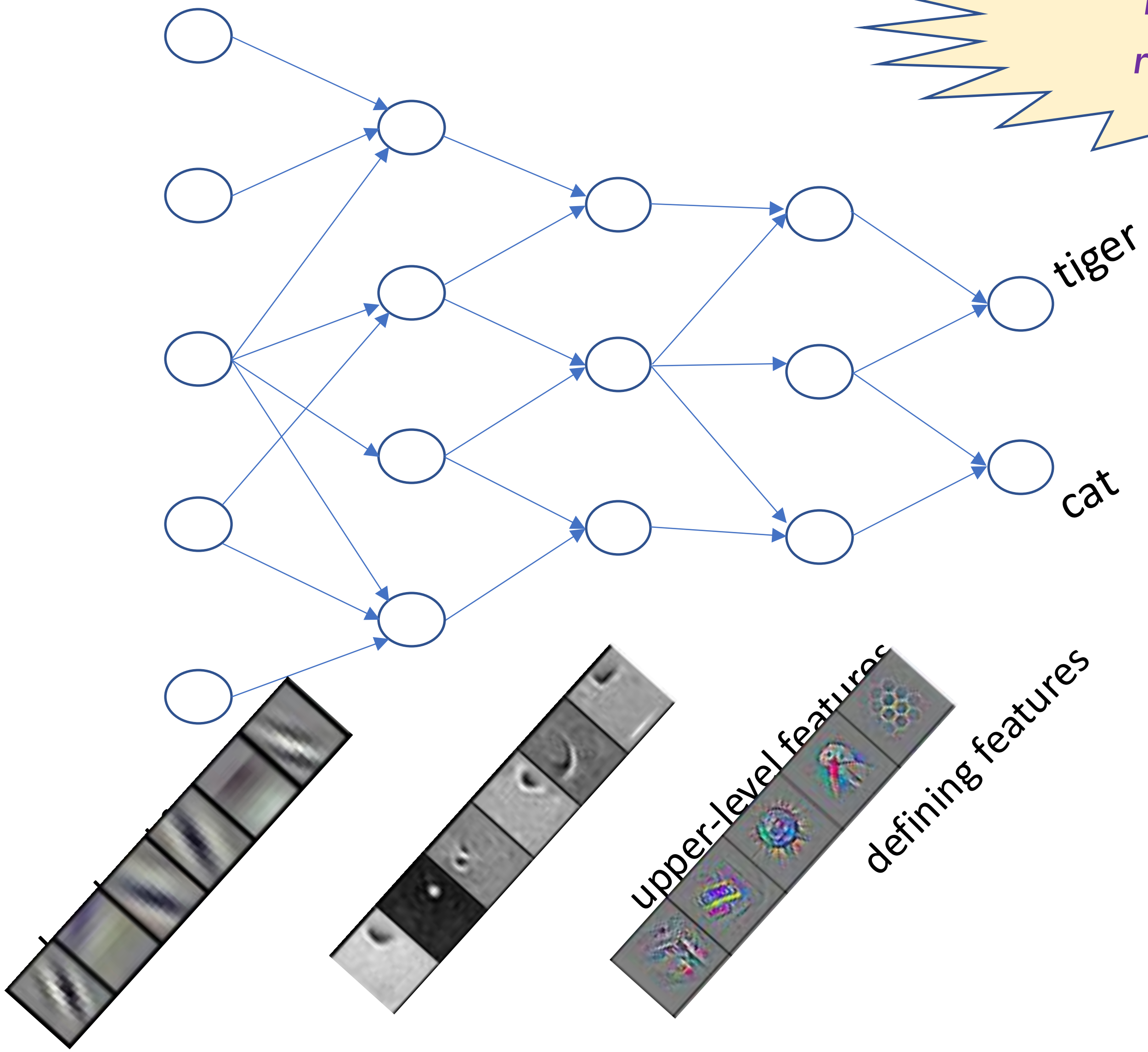
Credit: Szűcs Róbert

Watershed is locally elastic!

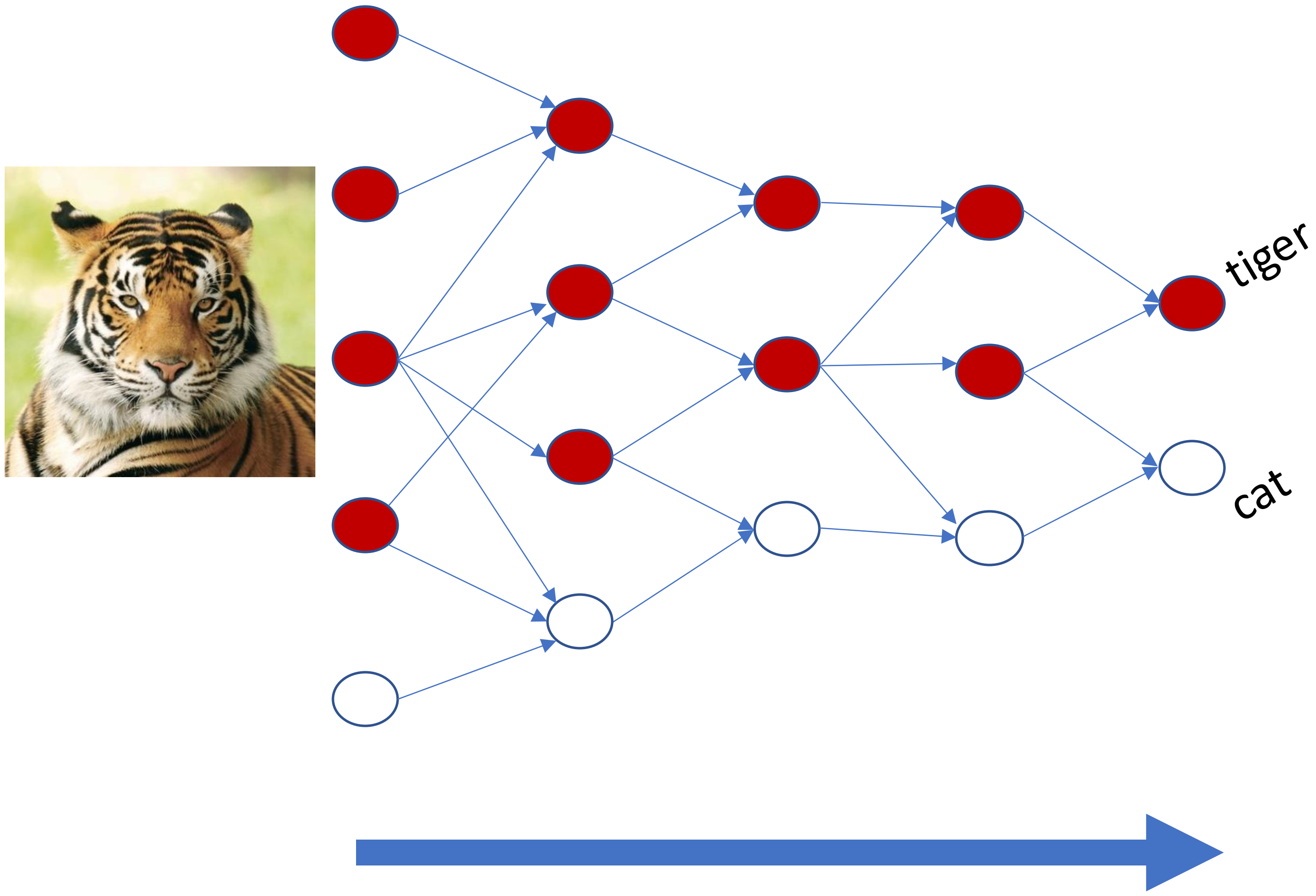


The *neurashed* model

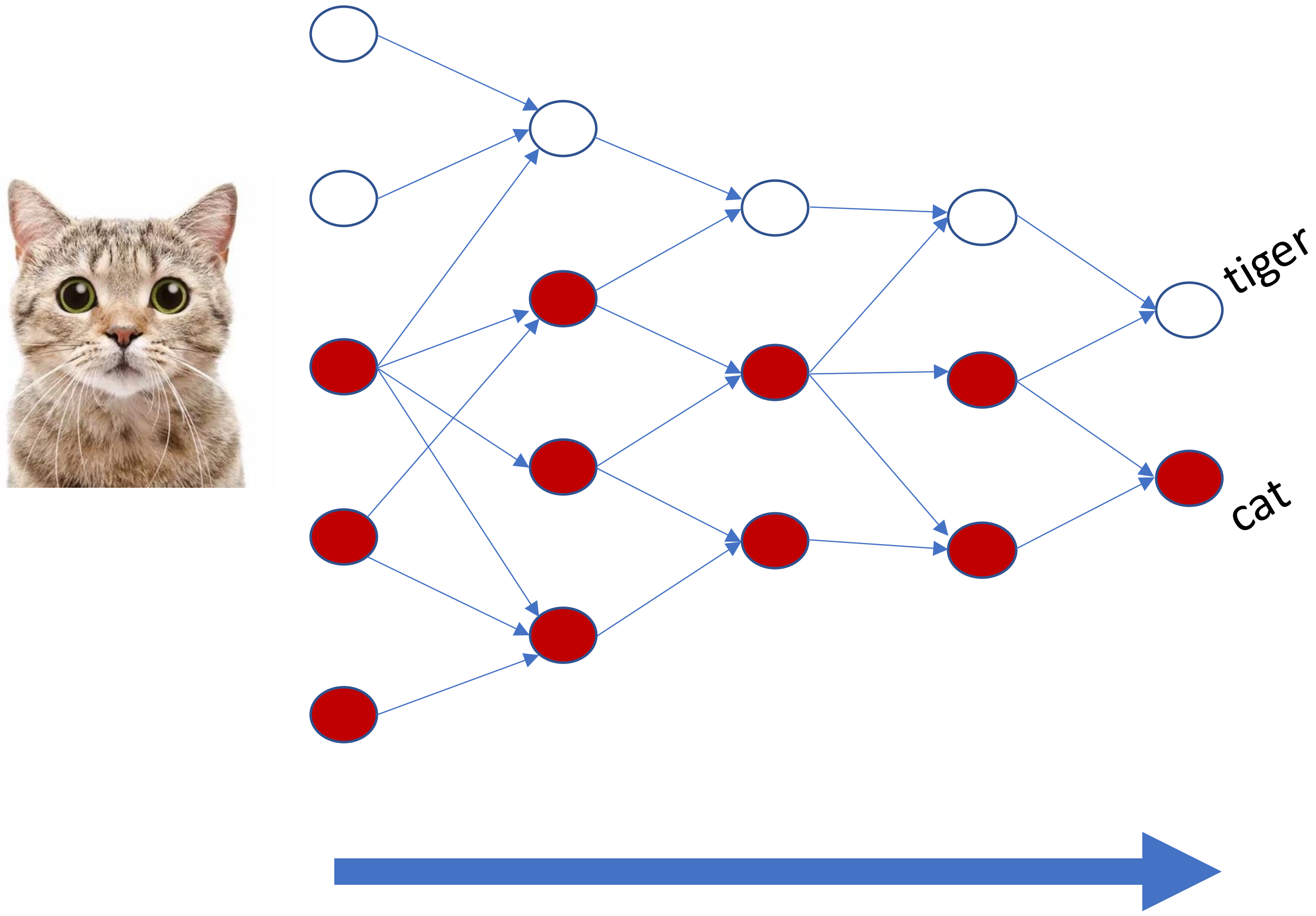
Nodes are features rather than neurons!



The neurashed model for prediction



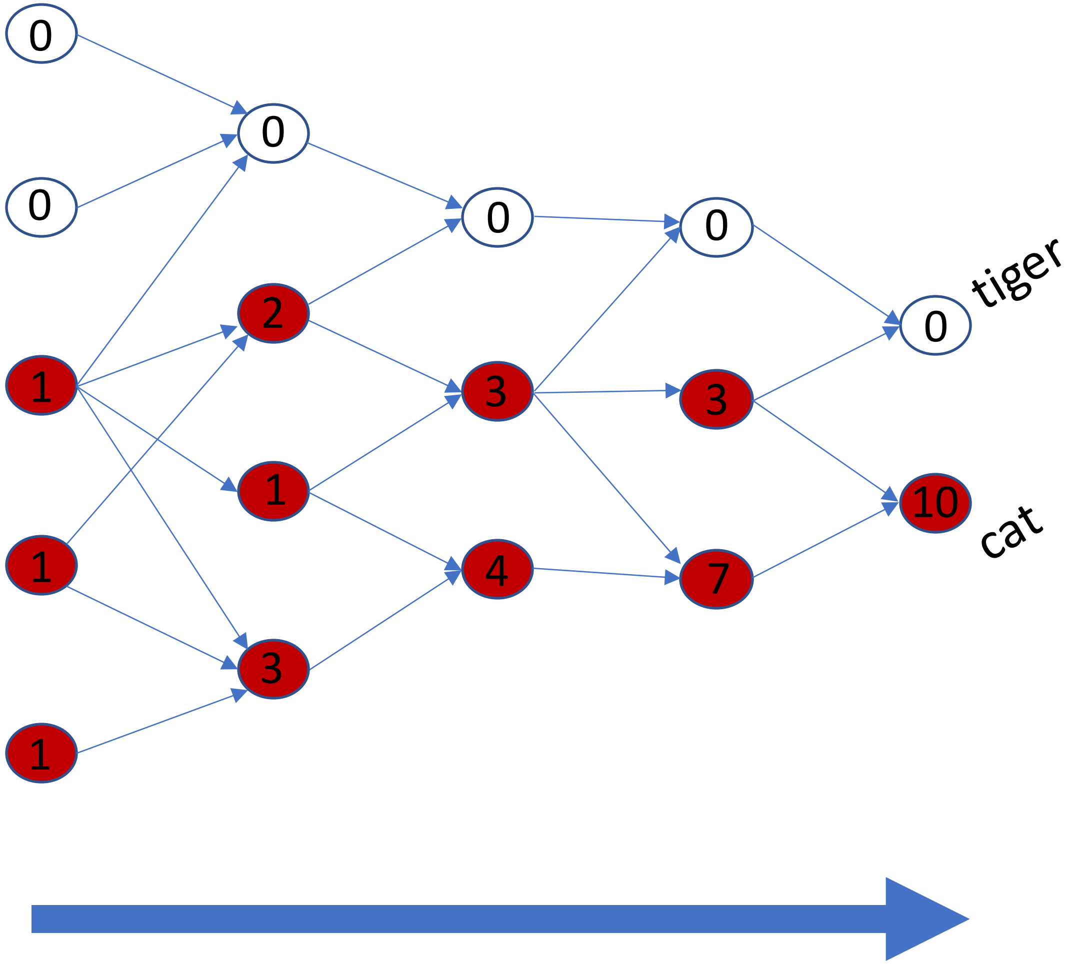
The neurashed model for prediction



Prediction

1. Red nodes denote activated features
2. Nodes in the first layer take 1 or 0 depending on being activated or not
3. An activated node F_j^l has value v_j^l that sums over its children, multiplied by its amplifying factor λ_j^l
4. Prediction probabilities $p_c = e^{v_c} / \sum e^{v_c}$

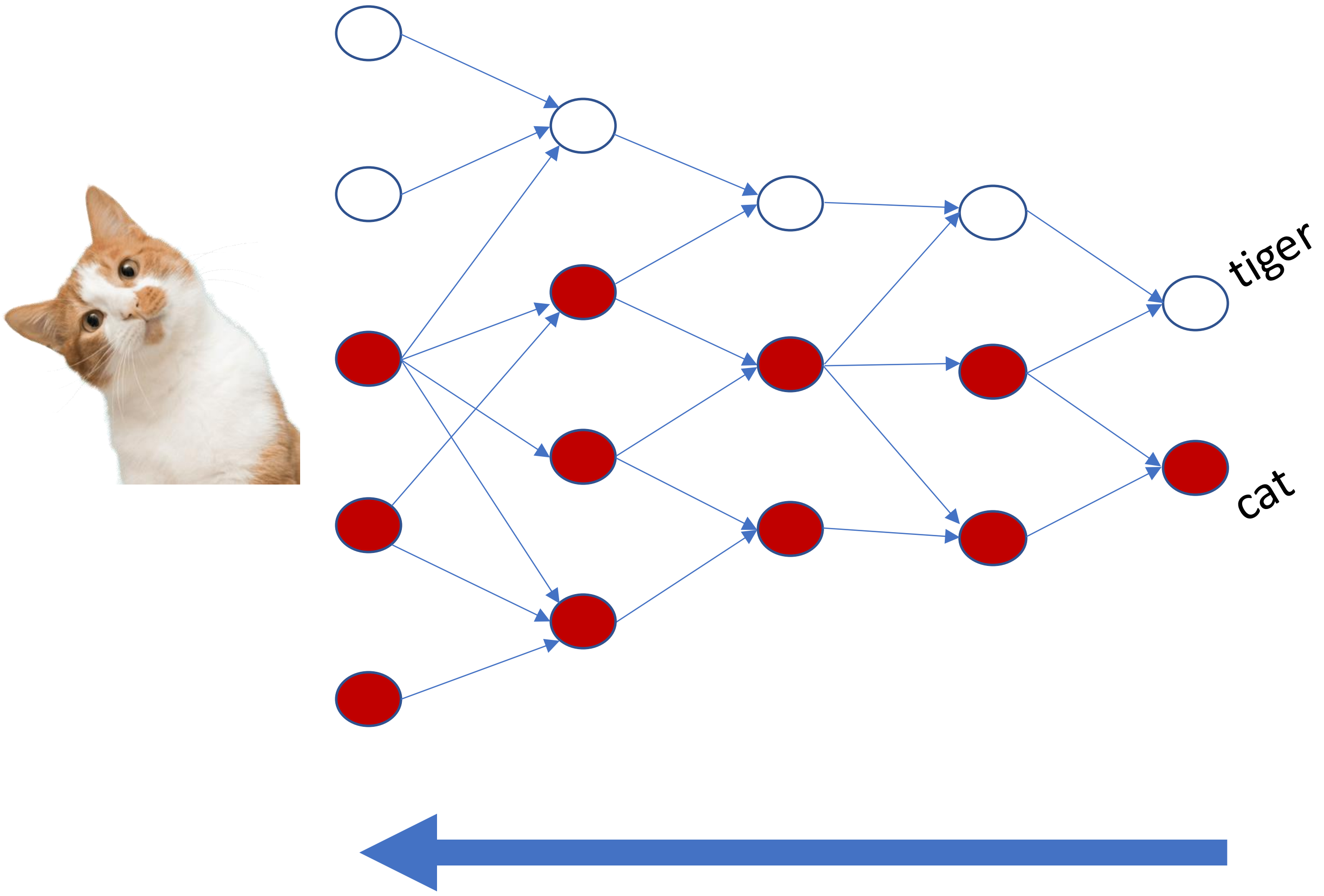
The neurashed model for prediction



Prediction

1. Red nodes denote activated features
2. Nodes in the first layer take 1 or 0 depending on being activated or not
3. An activated node F_j^l has value v_j^l that sums over its children, multiplied by its amplifying factor λ_j^l
4. Prediction probabilities $p_c = e^{v_c}/\sum e^{v_c}$

The neurashed model during training



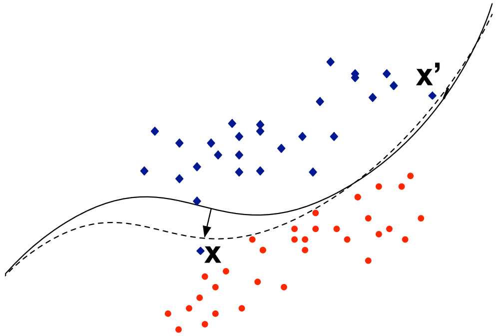
Training

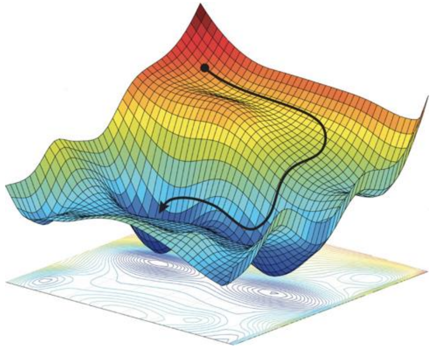
1. Initialize $\lambda_j^l = 0$ for all feature nodes
2. If a feature node is *activated*, then make the multiplier λ_j^l *larger*
3. If a feature node is *deactivated*, then make the multiplier λ_j^l smaller

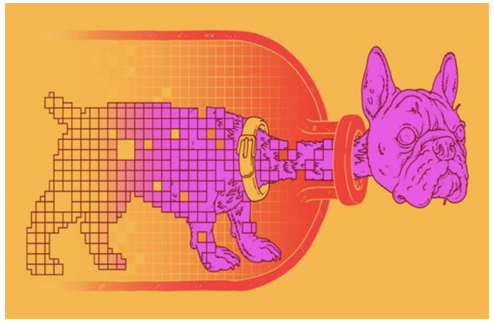
- *Perfection through practice*
- *Cells that fire together, wire together* (Hebbian theory)

Let's solve puzzles via Neurashed!

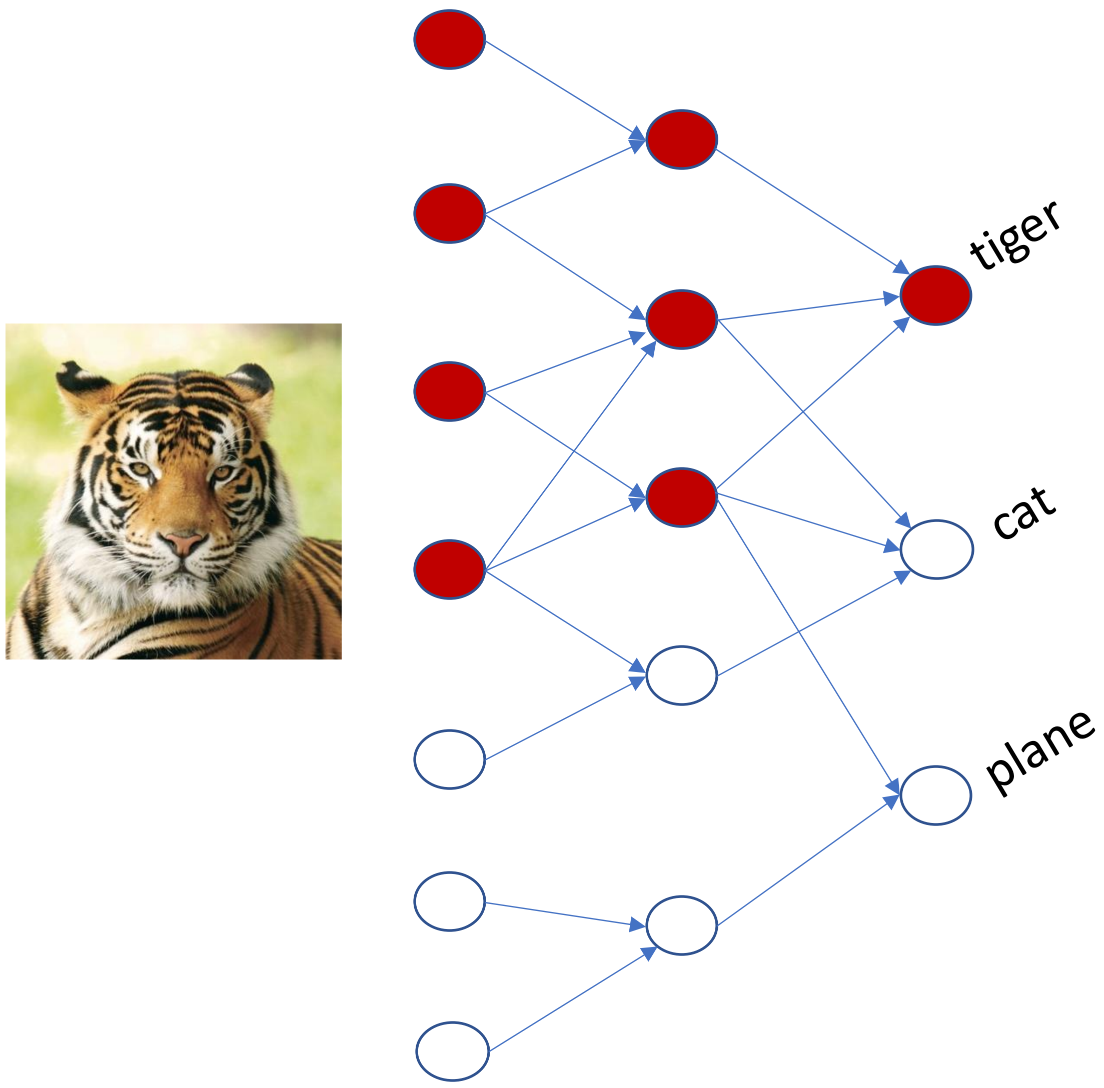
Local elasticity **Implicit regularization** **Information bottleneck**



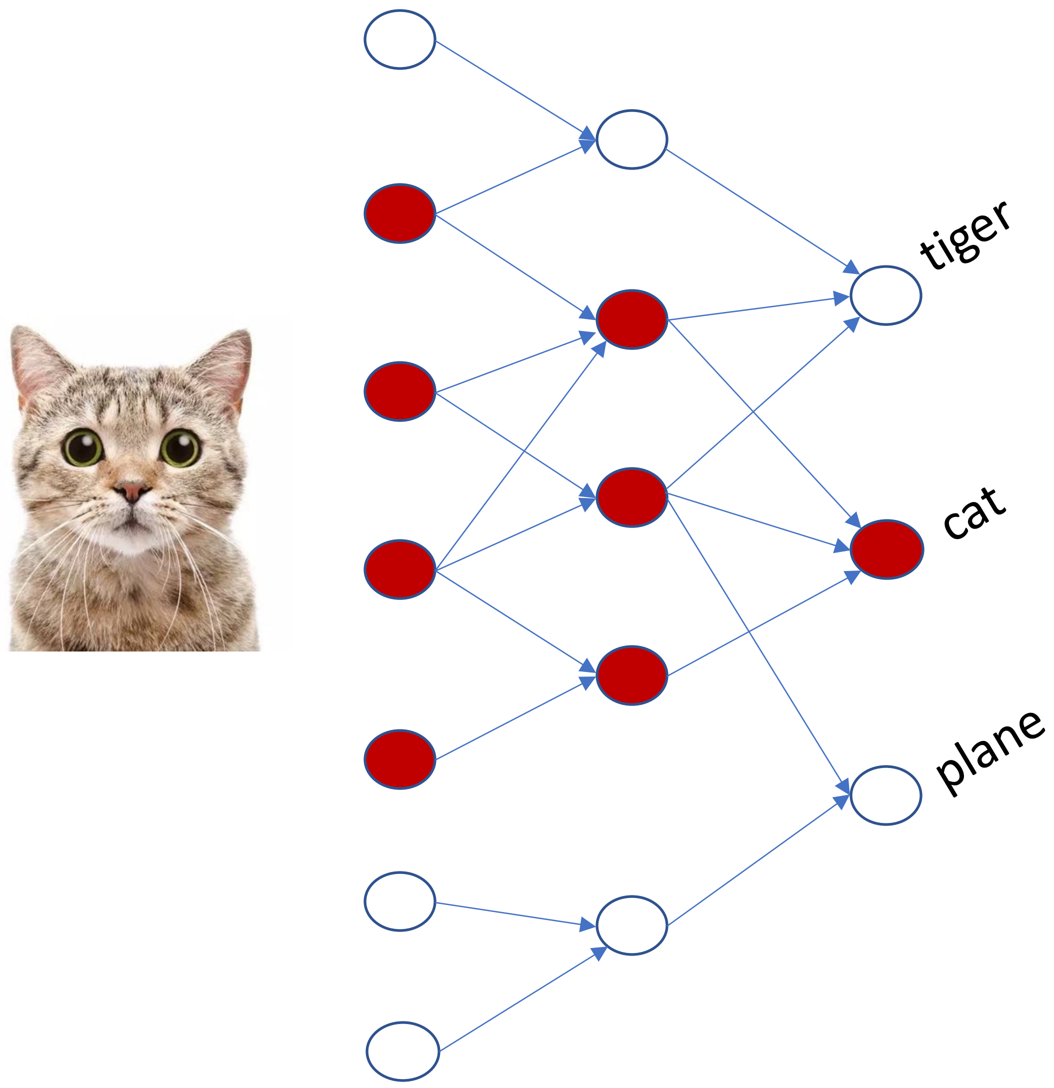




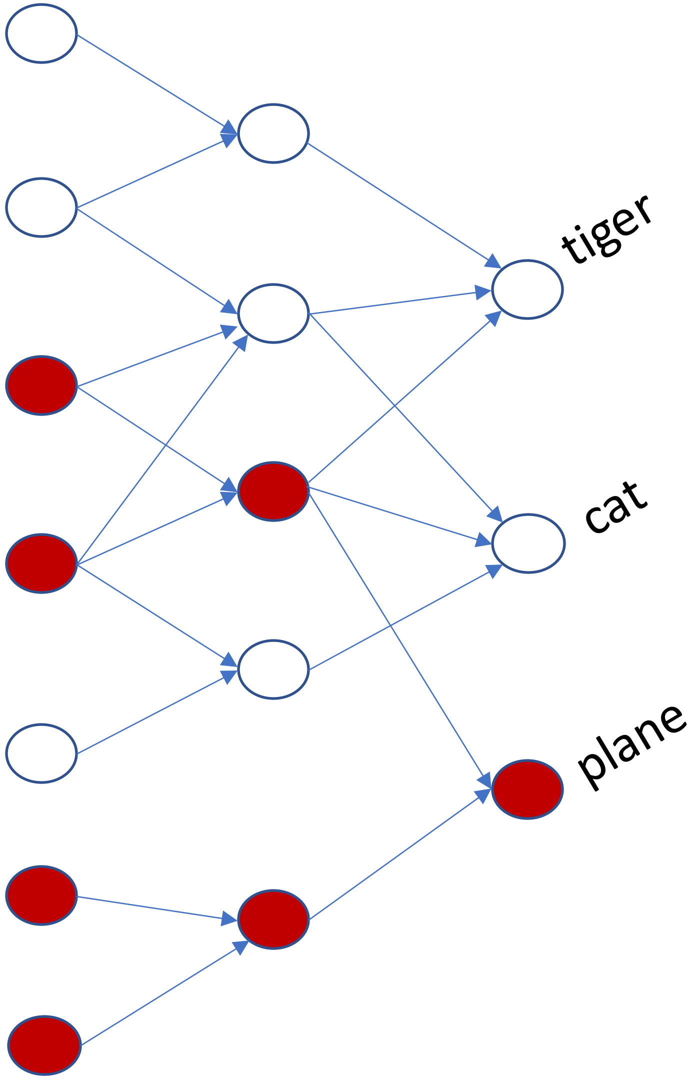
Neurashed implies local elasticity



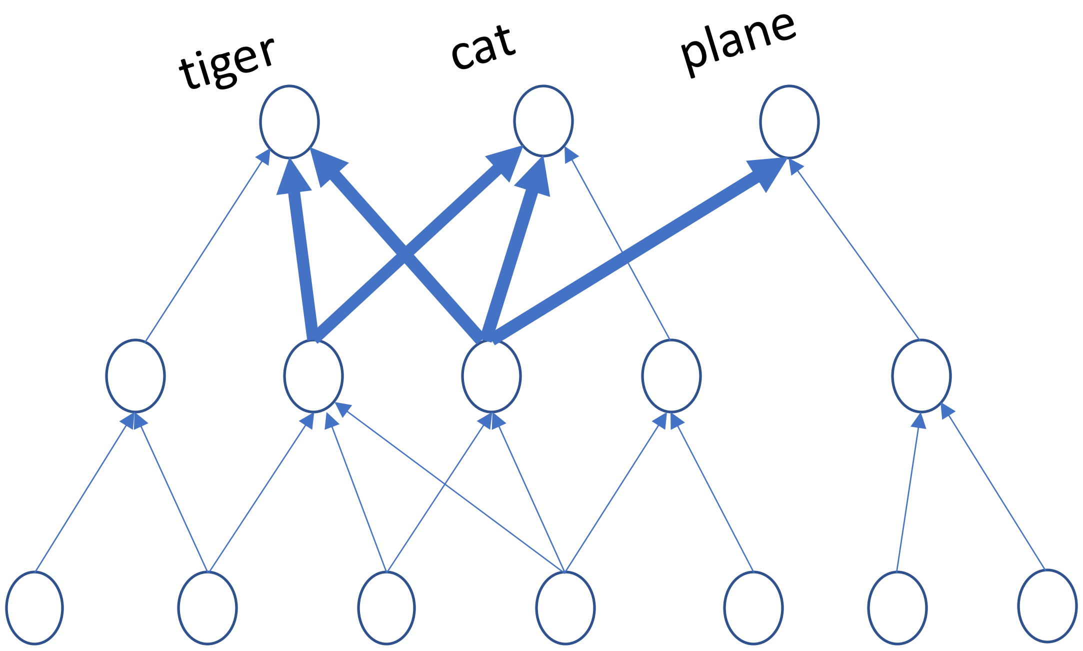
Neurashed implies local elasticity



Neurashed implies local elasticity



Neurashed implies local elasticity

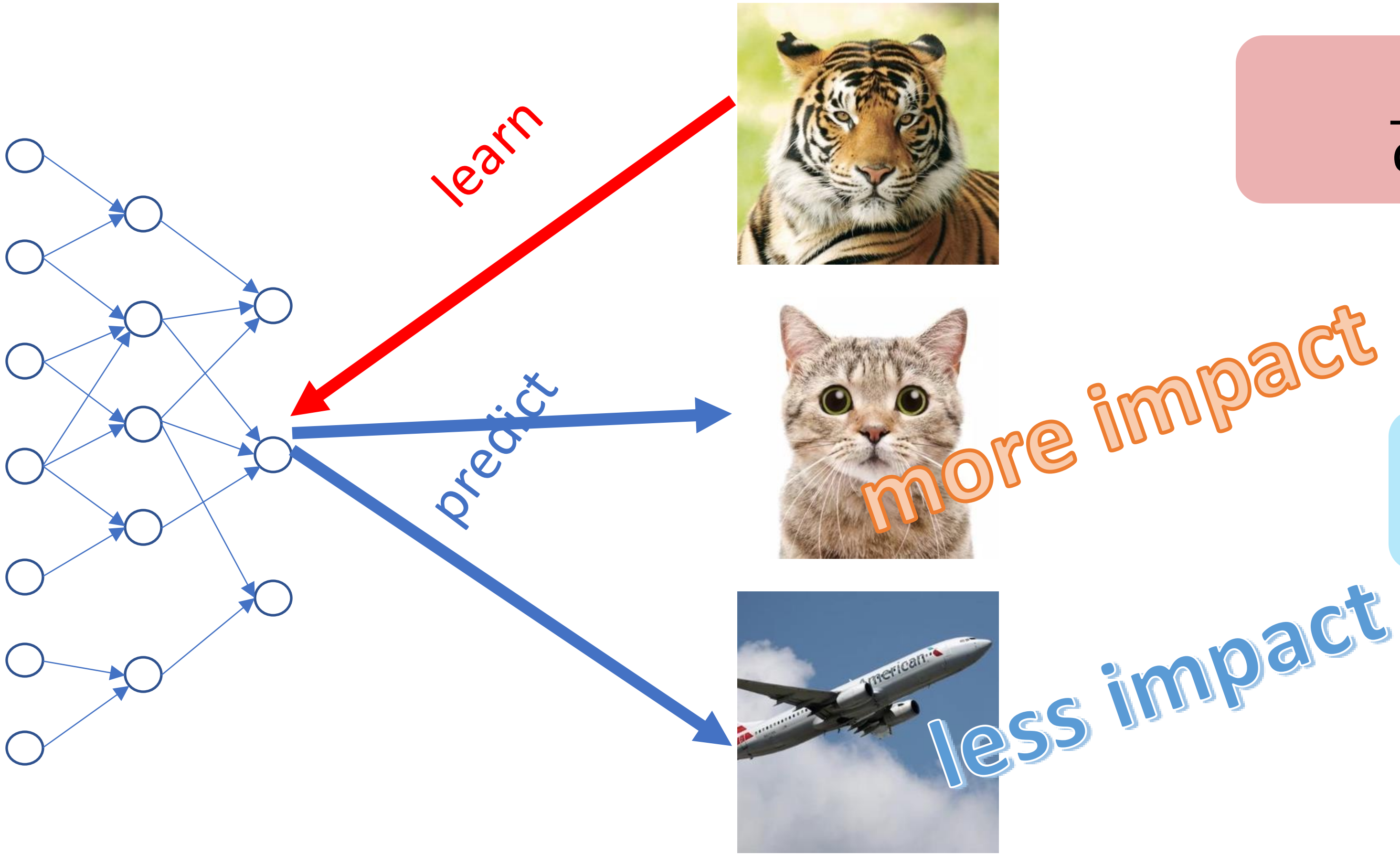


The more the feature is shared, the faster it grows

Let $N_t = N_c = N_p$. Then an update on a tiger leads to

$$\frac{\text{change in logit of cat}}{\text{change in logit of plane}} = \frac{5}{2}$$

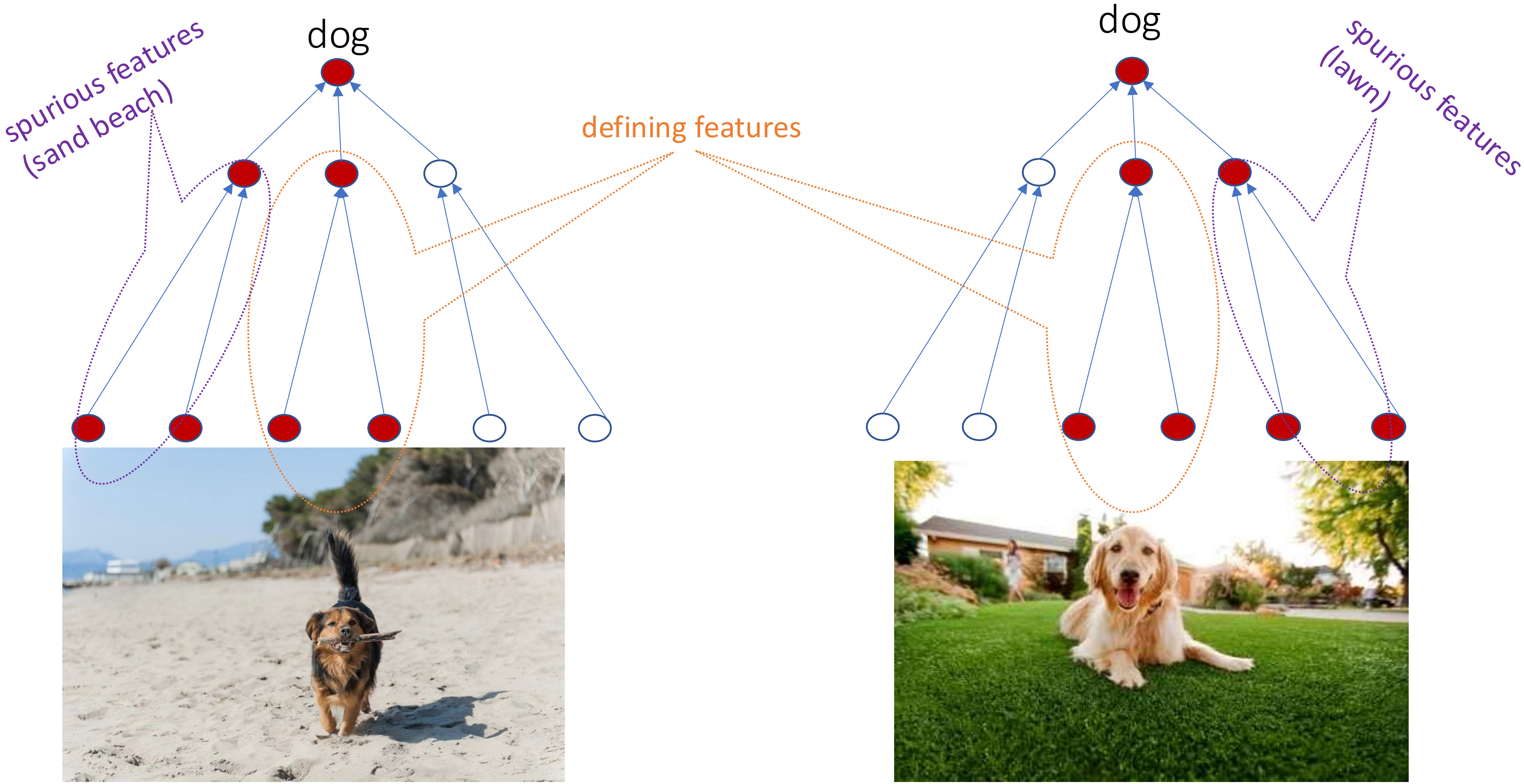
Neurashed implies local elasticity



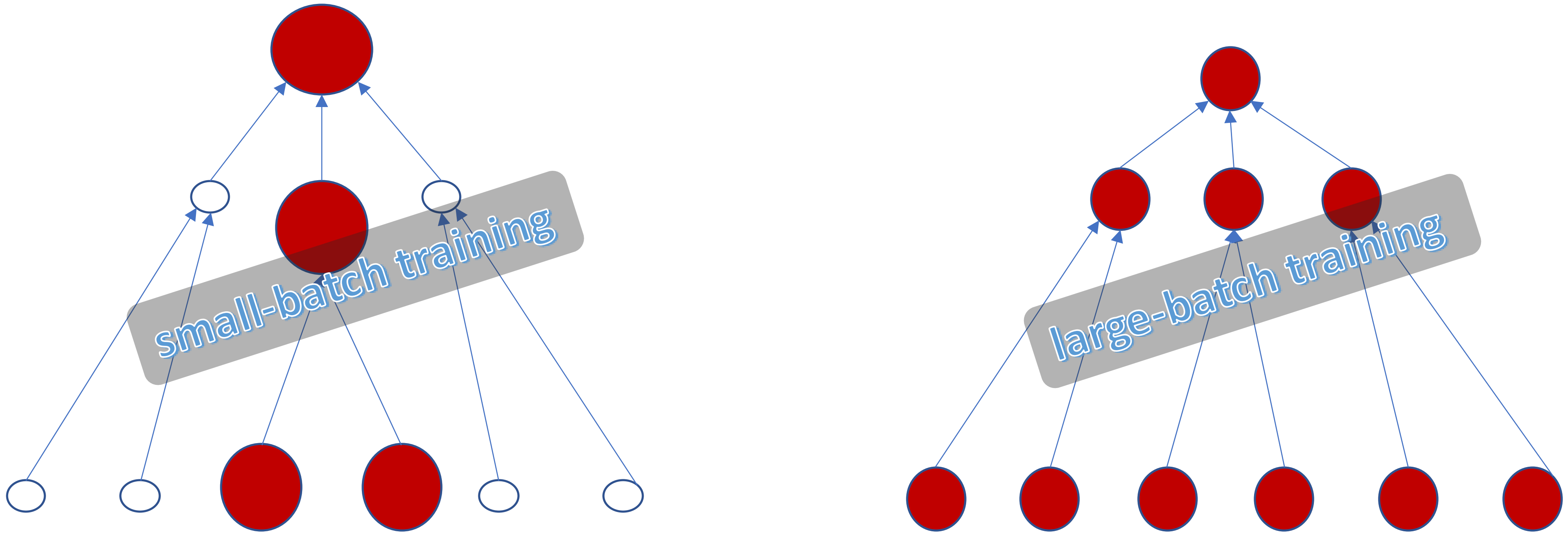
$$\frac{\text{change in logit of cat}}{\text{change in logit of plane}} = \frac{5}{2}$$

The difference would be more significant with more depth

Insights into implicit regularization



(Small-batch) training gives implicit regularization

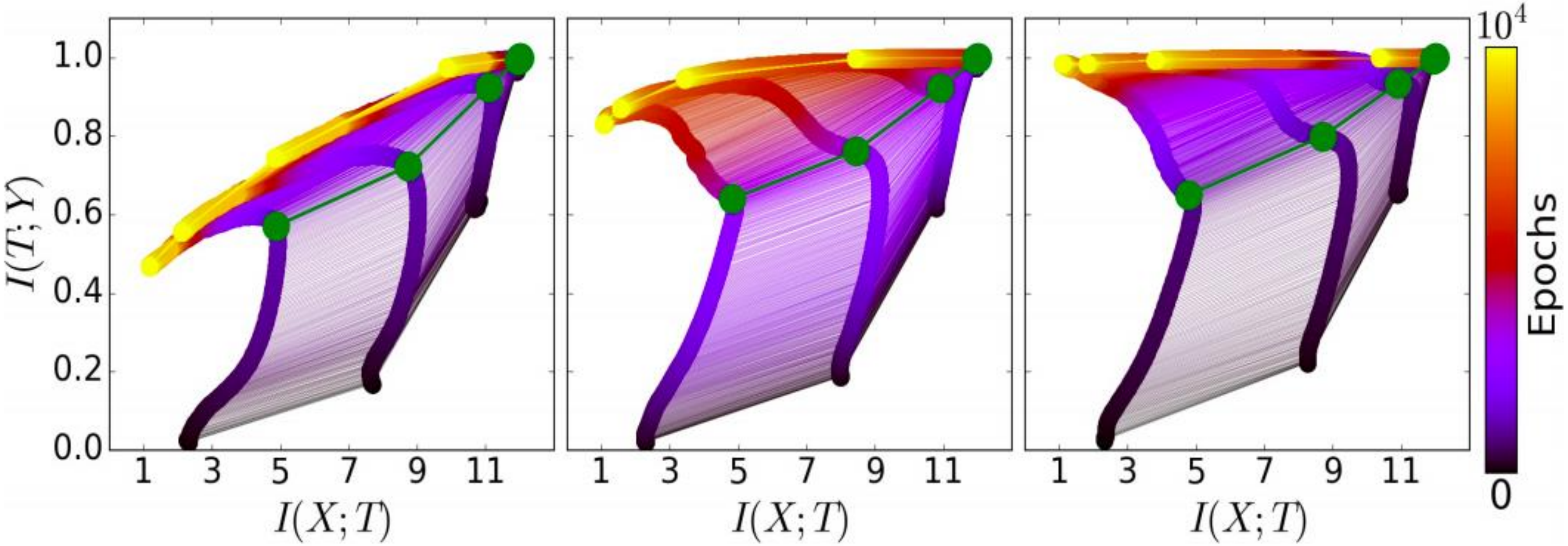


- Common features grow faster than rare features. A notion of sparsity?
- Common features generalize better (Ilyas et al, 2019)
- Fundamental difference between GD and SGD (Smith et al, 2020). Implications on NTK?

Information bottleneck (Tishby and Zaslavsky, 2015)

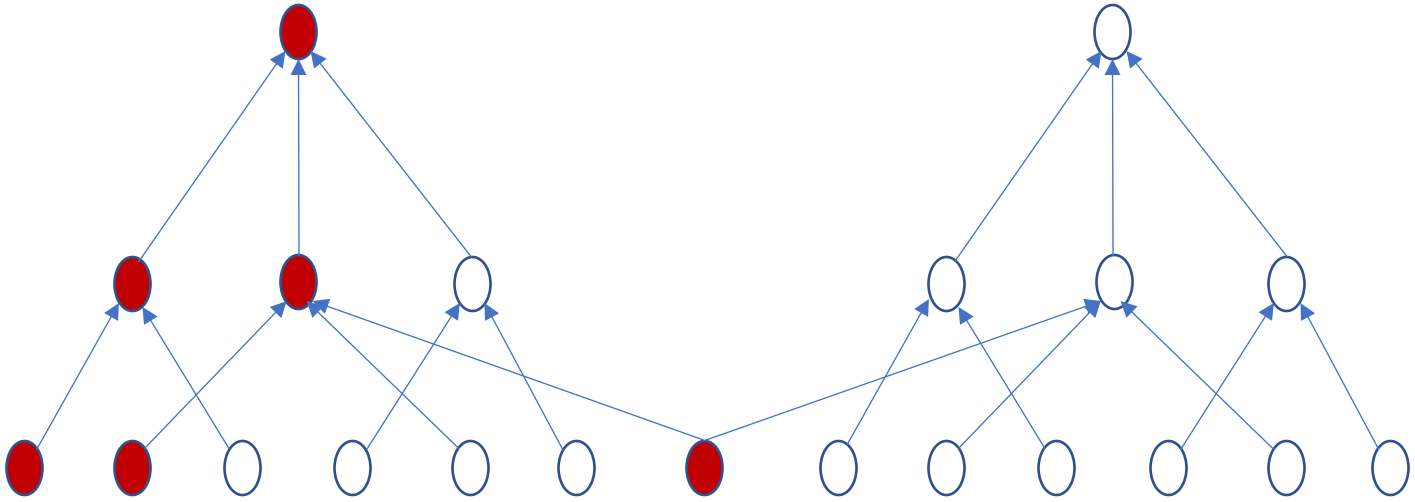
- In the initial phase, neural networks seek to fit both the input and output

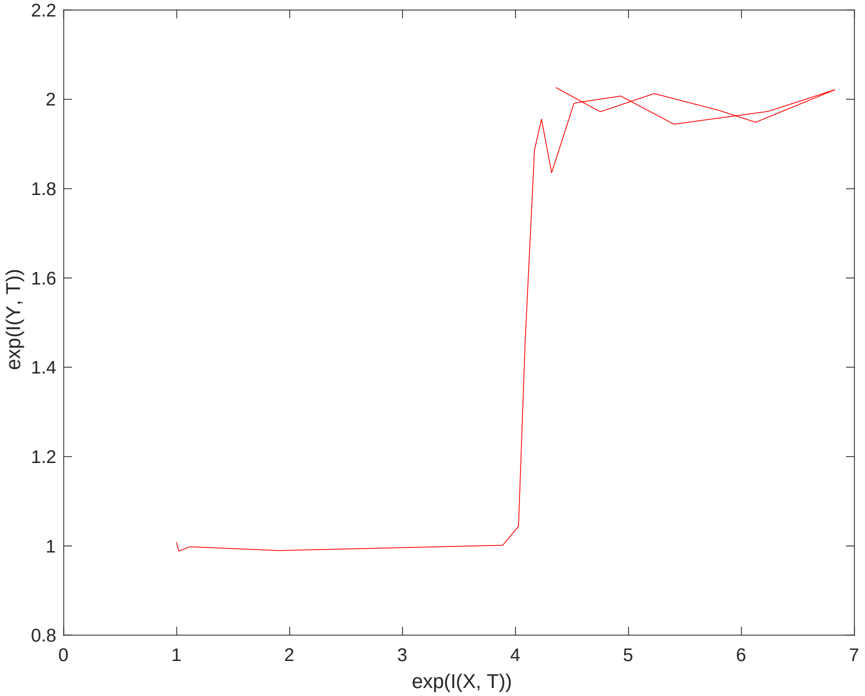
- In the second phase, the networks compress all irrelevant information of the input



Credit: Schwartz-Ziv and Tishby, 2017

Neurashed's read on information bottleneck

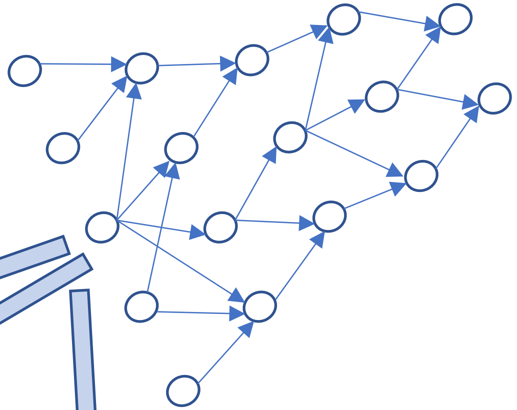




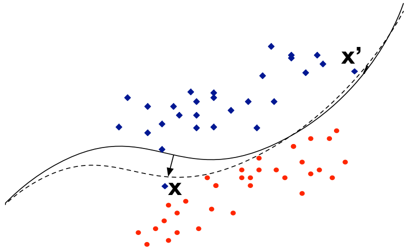
- 2 classes, each with 4 different types
- The bottom level: initially 3 bits, later 2 bits
- The middle level: initially 2 bits, later 1 bit

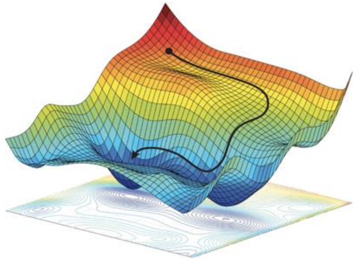
Summary of the neurashed model

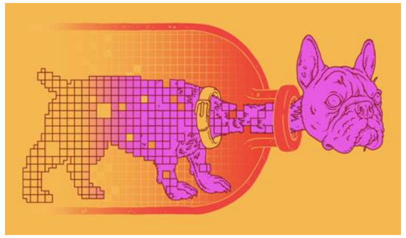
- Features are represented as their low-level features in a hierarchical manner
- More related classes share more features
- Features grow as they get activated in training



Local elasticity *Implicit regularization* *Information bottleneck*

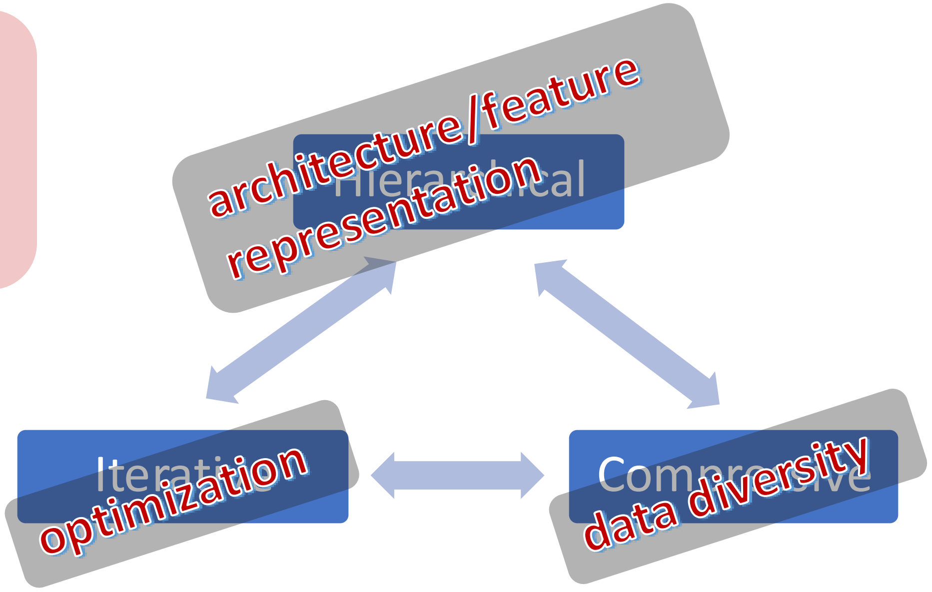






Future work motivated by local elasticity/neurashed

An 'ultimate' deep learning theory should be a ***neural network*** *itself, having hierarchical, iterative and compressive natures*



- How to (mathematically) define features?
- How do features relate to weights of the neural networks? Duality?
- How to model real-life data?
- How does the iterative nature of SGD/Adam relate to the training of neurashed
- How to quantify the information flow over layers?

Because it’s there…

References

1. The Local Elasticity of Neural Networks. w/ Hangfeng He. *ICLR 2020 (arXiv:1910.06943)*

2. Label-Aware Neural Tangent Kernel: Toward Better Generalization and Local Elasticity. w/ Shuxiao Chen and Hangfeng He. *NeurIPS 2020 (arXiv:2010.11775)*

3. Toward Better Generalization Bounds with Locally Elastic Stability. w/ Zhun Deng and Hangfeng He. *ICML 2021 (arXiv:2010.13988)*

4. Neurashed: A Phenomenological Model for Imitating Deep Learning Training *(arXiv:2112.09741)*

5. Imitating Deep Learning Dynamics via Locally Elastic Stochastic Differential Equations. w/ Jiayao Zhang and Hua Wang. NeurIPS 2021 *(arXiv:2110.05960)*

Acknowledgement: NSF CAREER award, Sloan research fellowship, NSF TRIPODS, and Wharton Dean's funding

@weijie444