# Deep Learning: The *What* and *Why*

UVA DEEP LEARNING COURSE
EFSTRATIOS GAVVES – 1

# Long story short

A family of parametric non-linear and hierarchical representation learning functions, which are massively optimized with stochastic gradient descent to encode domain knowledge, i.e. domain invariances, stationarity.

$$a_L(x; \theta_{1,\ldots,L}) = h_L(h_{L-1}(\ldots(h_1(x, \theta_1), \ldots), \theta_{L-1}), \theta_L) \Rightarrow$$
$$\Rightarrow a_L = h_L \circ h_{L-1} \circ \cdots \circ h_1(x)$$

- $x$:input, $\theta_l$: parameters for layer l, $a_l = h_l(x, \theta_l)$: (non-)linear function
- Given training corpus $\{X, Y\}$ find optimal parameters

$$\theta^* \leftarrow \arg\min_\theta \sum_{(x,y)\subseteq(X,Y)} \ell(y, a_L(x; \theta_{1,\ldots,L}))$$

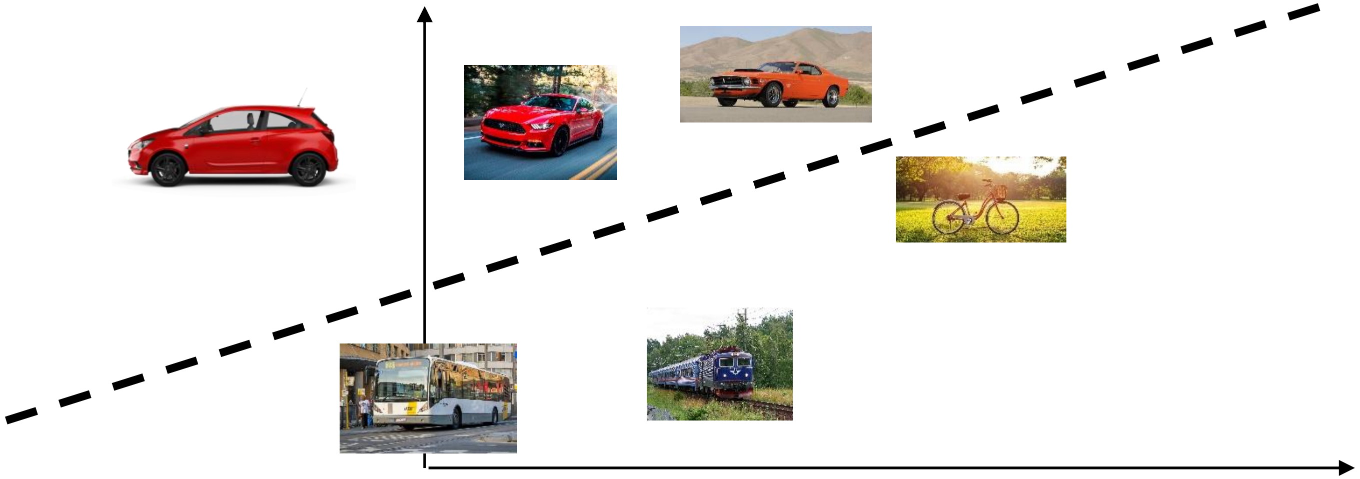
- But why all the trouble?

# Simplest case: Linear machines (classifiers)

- Think of an SVM, a logistic regression, or the original perceptron on raw data

$$y = \sum_j w_j \, x_j$$

- Say our data $(x, l)$ are images of either "cars" $(l = +1)$ or "not cars" $(l = -1)$
- Task: Find a line that must separate the +1's from the -1's

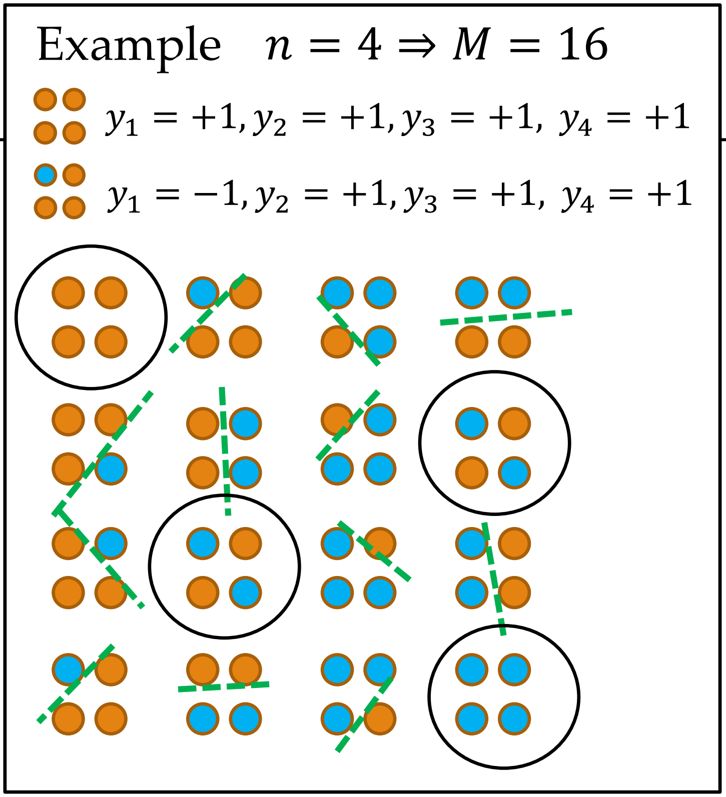


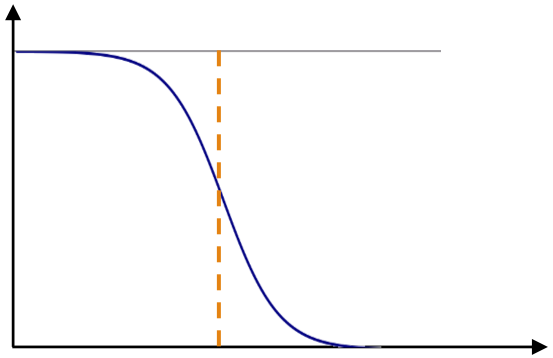
# Non-separability of linear machines

- Let's abstractify

$$(x, l)_n : x_i \in \mathbb{R}^d$$

- We have $M = 2^n$ "possible datasets"
- Only (about) $d$ out of $M$ are linearly separable
- With $n > d$ the probability $X$ is linearly separable converges to 0 very fast
- The chances that a dichotomy is linearly separable is very small



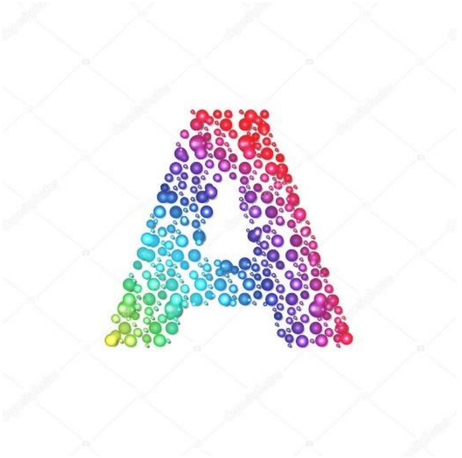
# Idea: Non-linear features, linear classifiers

- Most interesting problems are non-linear
  - image classification, speech generation, machine translation, tumour detection, …

- Idea: have non-linear features $x_j$, then linear machines are good enough
  - E.g., kernel trick

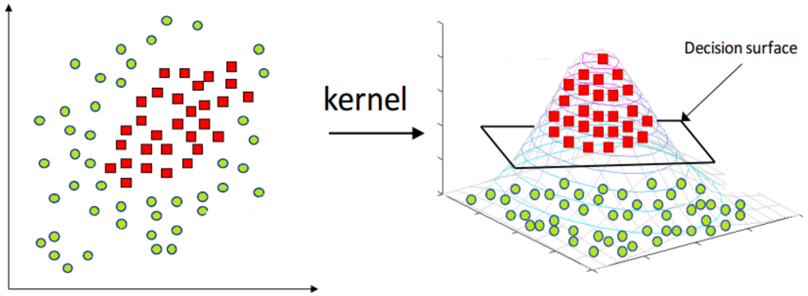
# What is a good feature?

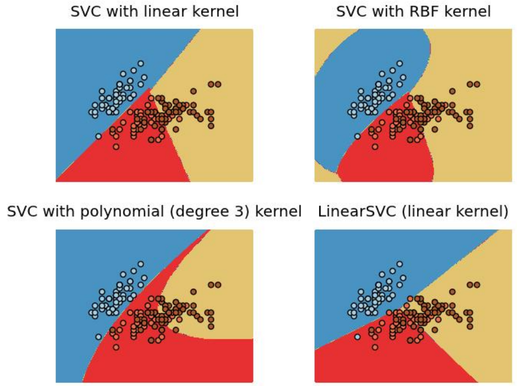
- Invariant … but not too invariant
- Repeatable … but not bursty
- Discriminative … but not too class-specific
- Robust … but sensitive enough



# How to get a good feature? Manual feature engineering

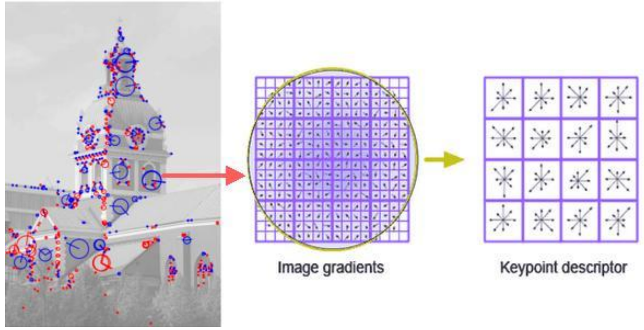
- Non-linear kernels, e.g., polynomial, RBF, etc



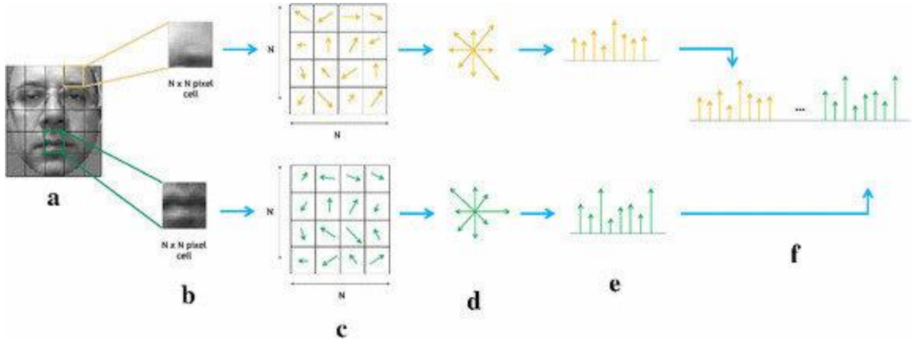




- Explicit design of features

SIFT

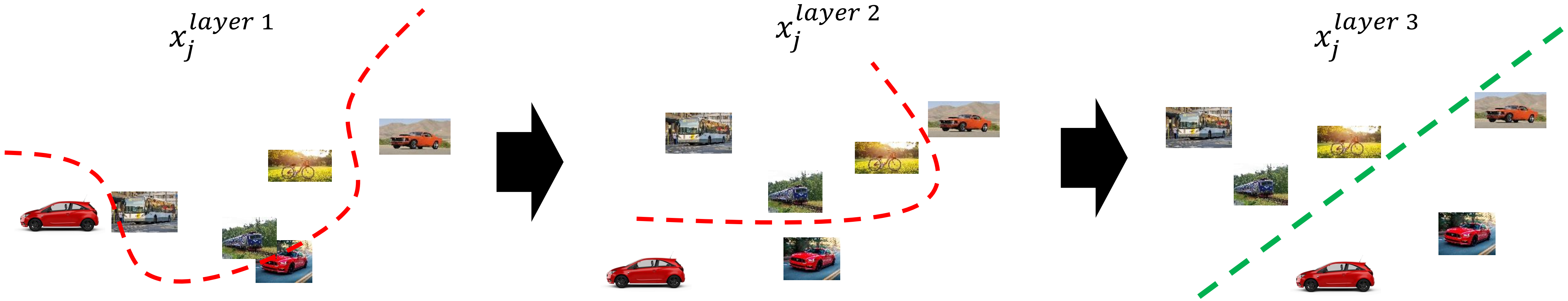



HOG



# Better: Learn non-linear features, linear classifiers

- Start from $x_j$ being raw data (*e.g.*, pixels)
- Transform them gradually till linear enough for classifiers
- Transformations learned from (raw) data for optimal separation
  - If not raw data, data already transformed and little we can do about it



# Why learn the features?

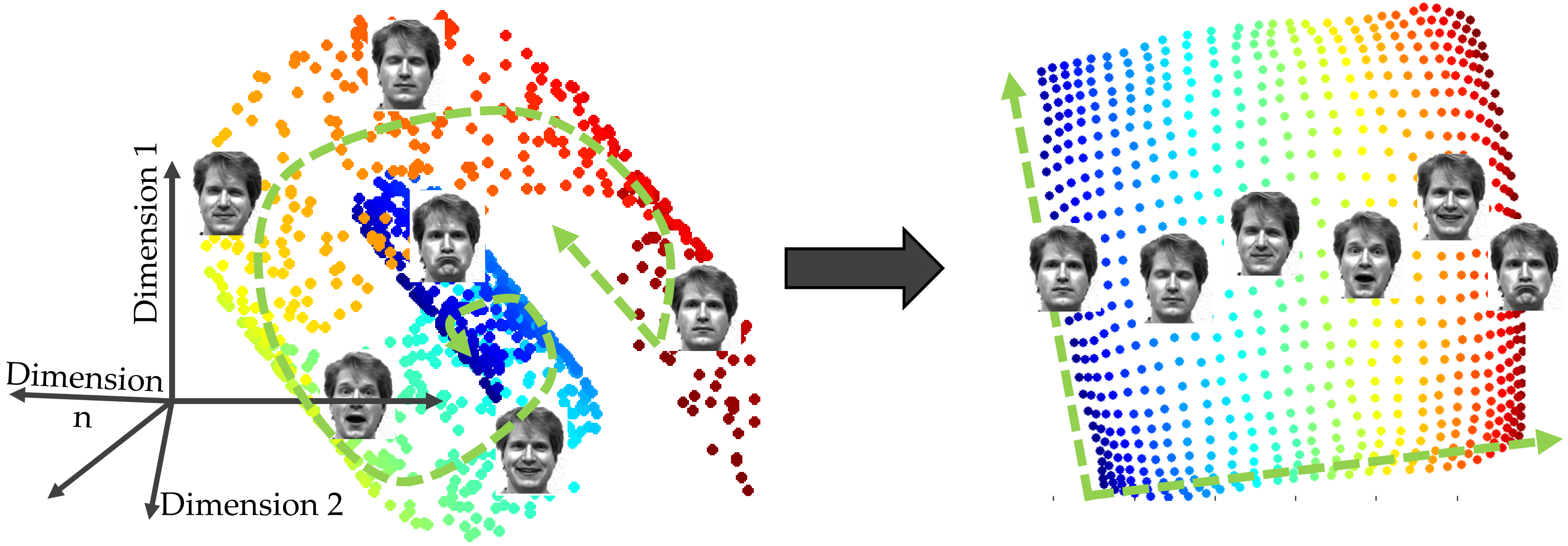
- Manually designed features
  - Expensive to research & validate
- Learned features
  - If data is enough, easy to learn, compact and specific
- Time spent for designing features now spent for designing architectures

# How to get good features?

- Goal: discover these lower dimensional manifolds
  - These manifolds are most probably highly non-linear
- First hypothesis: Semantically similar things lie closer together than semantically dissimilar things
- Second hypothesis: All images (e.g., face) lie on a high-dimensional hidden manifold (manifold hypothesis)
  - Each face (or whatever data) is a coordinate on that manifold
  - That coordinate is a good feature as it places the data in relation to all others
  - We must only discover what the manifold

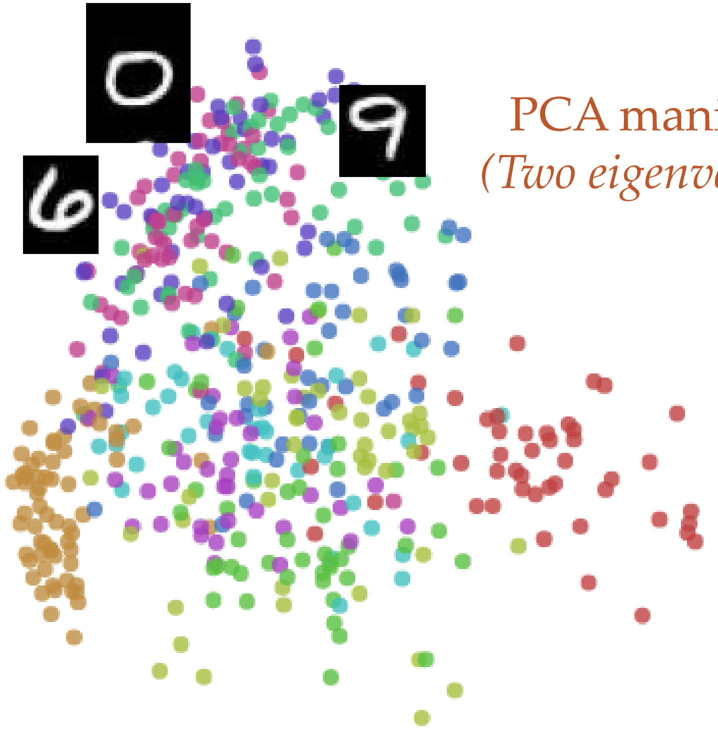
# Manifolds

- Learning transformations to discover "latent" manifolds
- Raw data live in huge dimensionalities
- But, effectively lie in lower dimensional manifolds



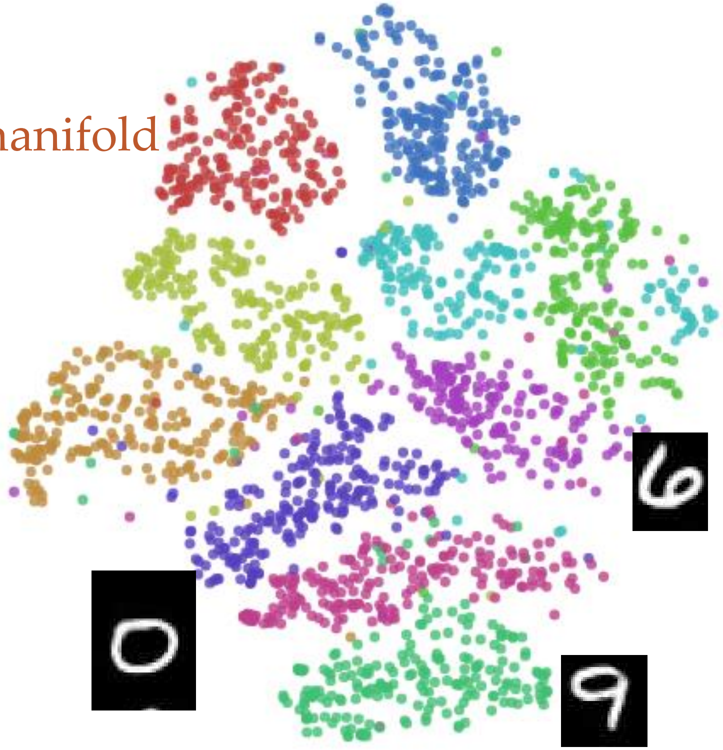
# The digits manifolds

- There are good features (manifolds) and bad features
- 28 pixels x 28 pixels = 784 dimensions



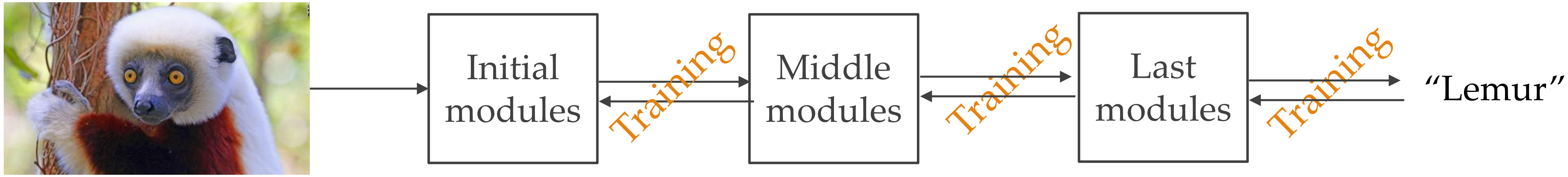
PCA manifold
*(Two eigenvectors)*

t-SNE manifold



# Deep learning ⇔ Learning Hierachical Representations

- A pipeline of successive, differentiable modules (transformations)
  - Each module's output is the input for the next module
- Each subsequent module produce higher abstraction features



# Deep Learning Approximation Theory

- Deep Networks are universal approximators

**Theorem** *Let* $\rho()$ *be a bounded, non-constant continuous function. Let* $I_m$ *denote the* $m$*-dimensional hypercube, and* $C(I_m)$ *denote the space of continuous functions on and* $\epsilon > 0$*, there exists N>0 and* $v_i, w_i, b_i, i = 1, \ldots, N$ *such that* $F(x) = \sum_{i \leq N} v_i \rho(w_i^T x + b_i)$ *satisfies* $\sup_{x \in I_m} |f(x) - F(x)| < \epsilon$.

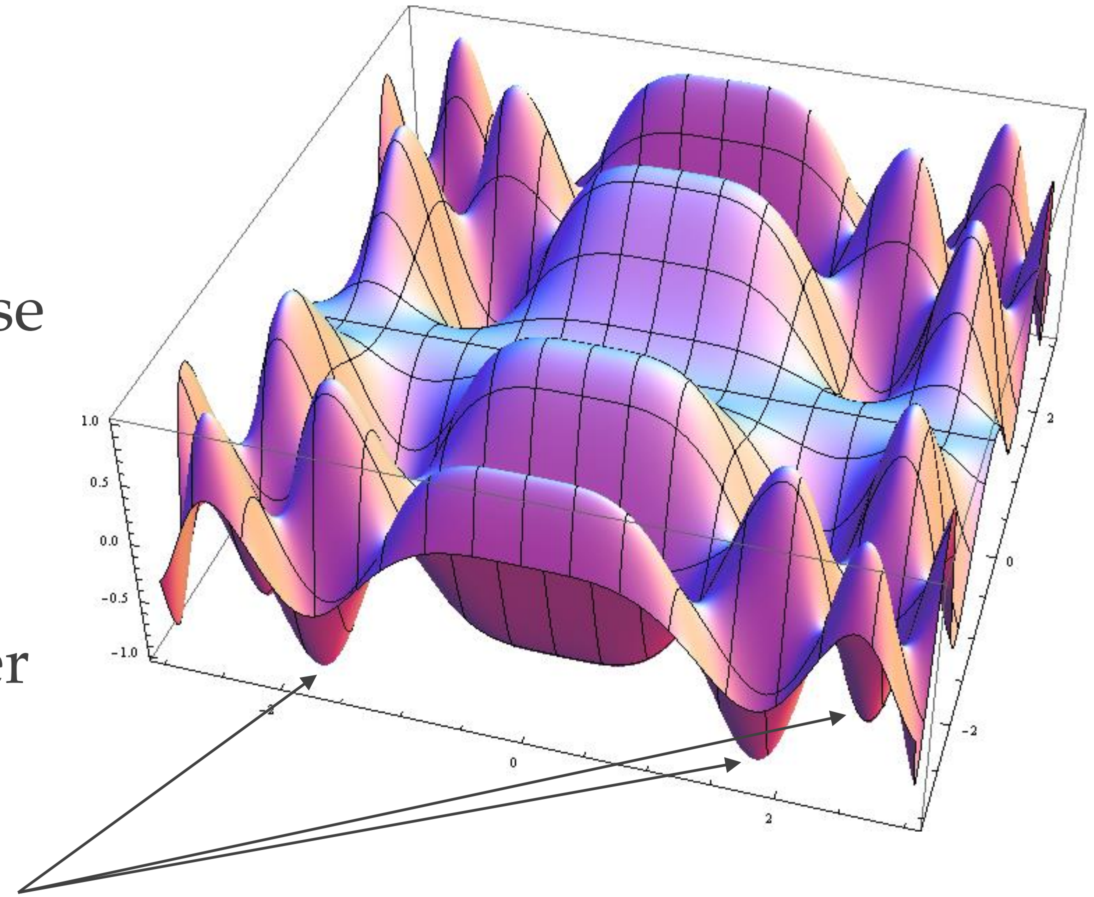
- Even a single hidden layer can approximate any function
  - and represent any locally linear boundary.
- But what is the precise architecture? And how to train?
  - The theorem odes not answer that

# So, why deep and not shallow?

- Deep architectures tend to be more data efficient
  - Better modelling capacity for the same number of parameters
- Otherwise, very wide, shallow networks
  - Wide and shallow are shown to also work pretty well
- Deep and narrow architectures show quite good generalization
  - Depth as a regularizer

# Depth → Non-convexity

- Highly non-convex. Yet, stable & accurate learning
- Current hypothesis: Most local minima close to global minimum and hence roughly equivalent
  - With many assumptions
- In practice, ensembles of models even better




Roughly equivalent.
Combine them to ensembles

# Deep learning is representation learning

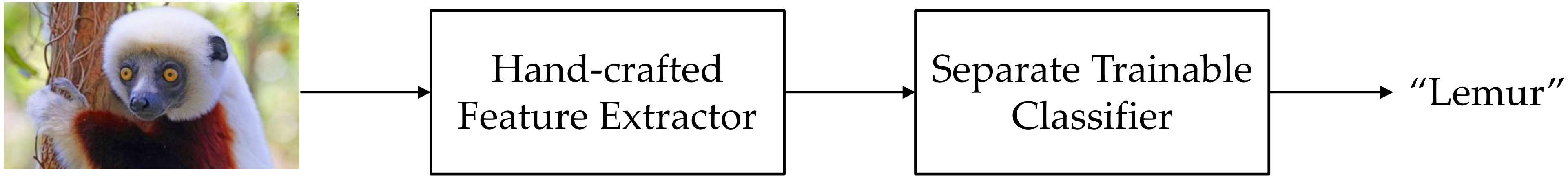
- Deep learning makes sense if "raw" data is uninterpretable
  - In that case, you can either create a representation or learn it
- If "raw" data is interpretable, you already got good representations
  - Deep learning might not add much
  - You must search for the right level of abstraction for deep learning

Examples

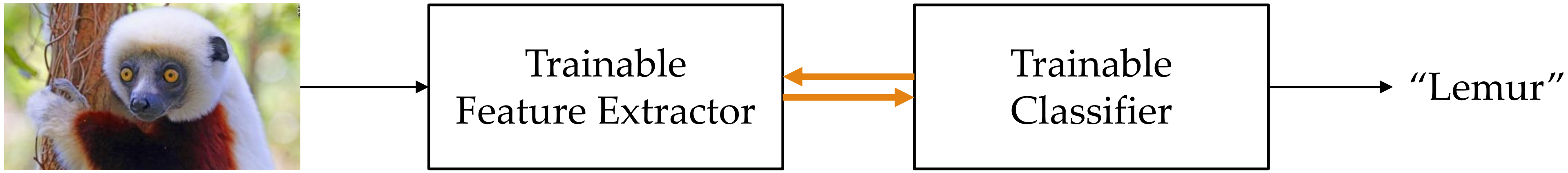
- In images raw data is pixels → Each pixel by itself means nothing → Representations must be learned → deep learning thrives
- In text words and letters are good representations already → deep learning may not immediately do better → go to higher abstraction, *e.g*, semantics?

# To conclude

- Traditional pattern recognition




- End-to-end learning → Features are also learned from data
  - If no raw data, deep learning makes no much sense



# Questions open?

- Unsupervised learning, reinforcement learning, world models, …
- Deep generative models
- Deep temporal learning
- Deep stochastic models
- Deep causality
- Deep private & federated learning
- The maths and physics of deep learning
- …
- Hopefully, some will answered by you in the near future ;)

# References

- http://www.deeplearningbook.org/
  - Chapter 1: Introduction, p.1-28

Extra reading for the interested reader

- A more comprehensive review of NN history
- A 'Brief' History of Neural Nets and Deep Learning, Part 1, 2, 3, 4
- Deep Learning in a Nutshell: History and Training
- The Brain vs Deep Learning

# Summary

- Course information
- A brief history of neural networks and perceptrons
- The arrival of deep learning
- Deep learning: The what and why

**Next lecture:** Deep modularity, backpropagation