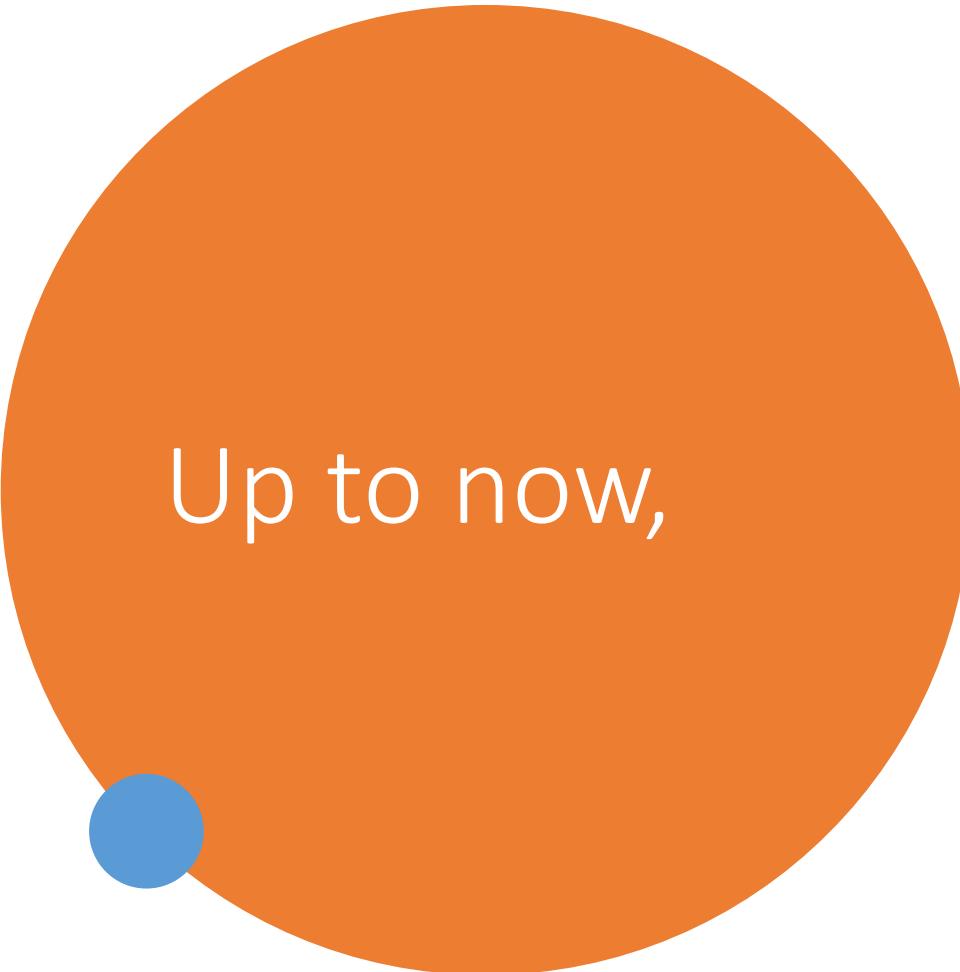


# Lecture 17 -- Deep Learning: Functional View and Features

Prof Xiaowei Huang

<https://cgi.csc.liv.ac.uk/~xiaowei/>

(Attendance Code: **491874**)



Up to now,

- Traditional Machine Learning Algorithms
- Adversarial Attack and Defence
- Deep learning
  - Introduction to Deep Learning (history of deep learning, including perceptron, multi-layer perceptron, why now?, etc)



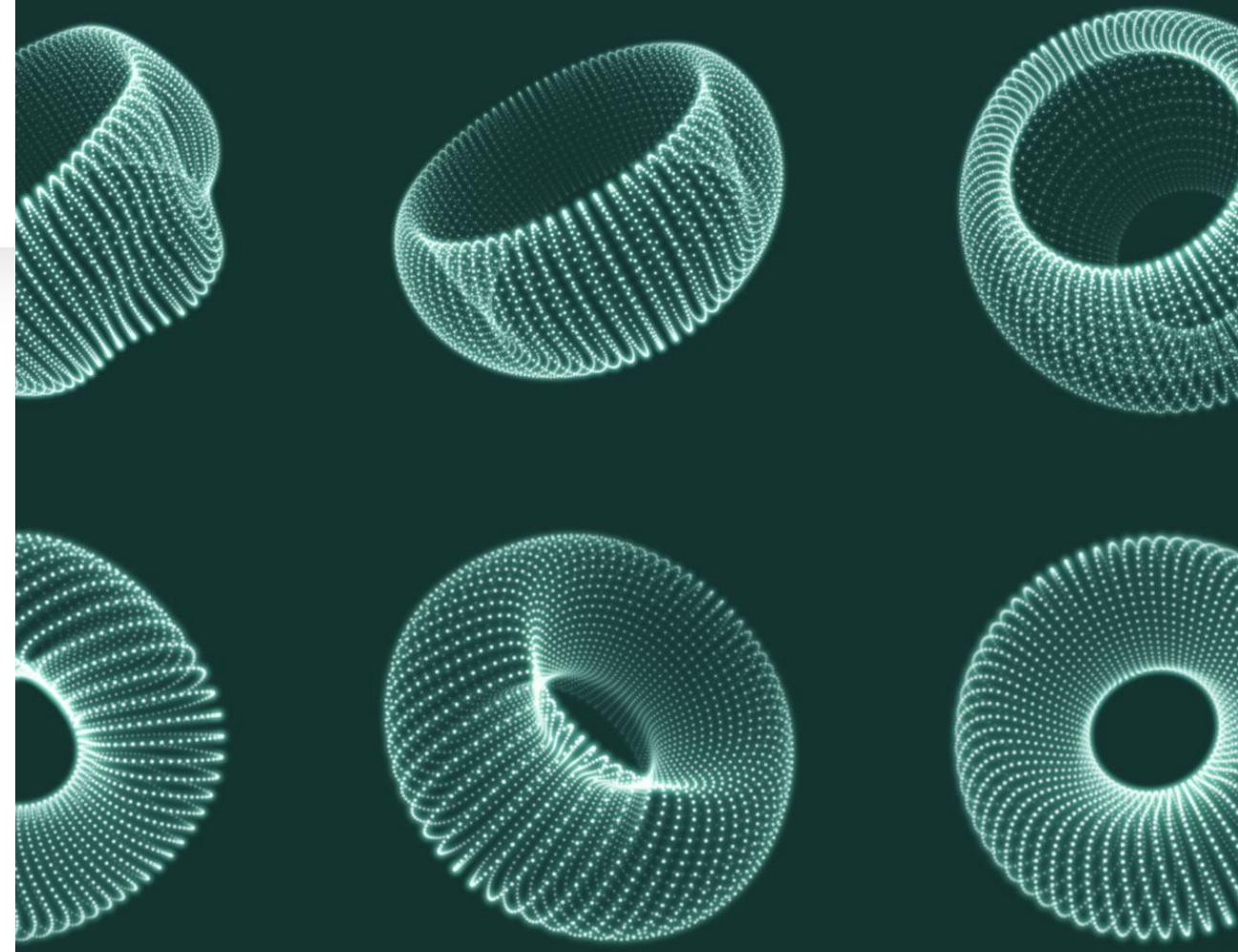
# Today's Topics

Functional View of DNNs

Learning Representations & Features

Explainable AI

# Functional View of DNNs



# What is a function?

- In programming, a named section of a program that performs a specific task. In this sense, a function is a type of procedure or routine.
- The term *function* is also used synonymously with *operation* and *command*. For example, you execute the delete *function* to erase a word.

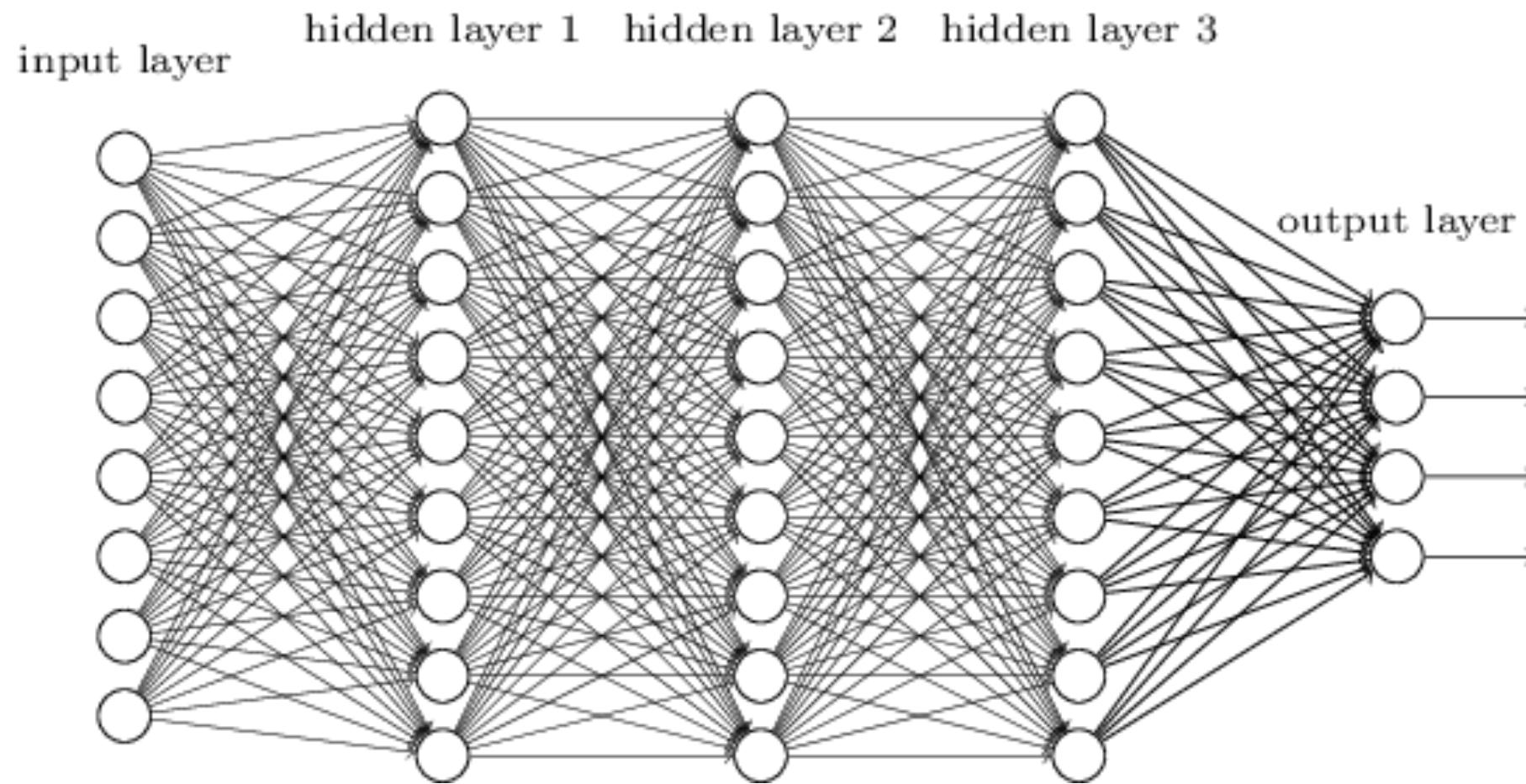


# What is a nested function?

- a **nested function** (or **nested procedure** or **subroutine**) is a function which is defined within another function, the *enclosing function*.

```
float E(float x)
{
    float F(float y)
    {
        return x + y;
    }
    return F(3) + F(4);
}
```

A **composition** of a function  $f \circ f$  with itself gives a nested function  $f(f(x))$ ,  $f \circ f \circ f$  which gives  $f(f(f(x)))$ , etc.



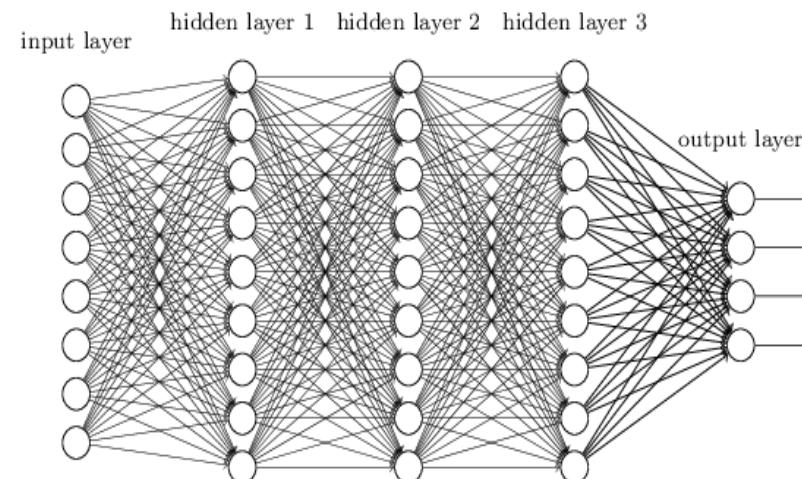
# Functional View of DNNs

---

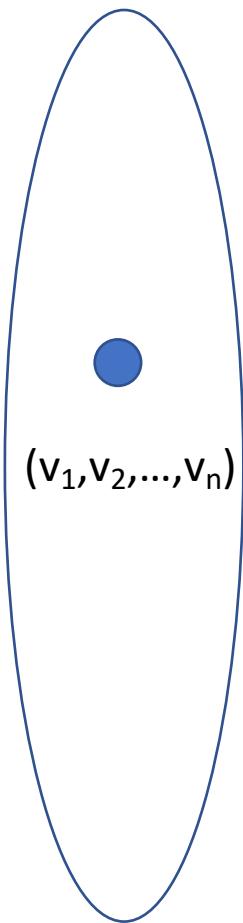
- A family of parametric, non-linear and hierarchical representation learning functions, which are massively optimized with stochastic gradient descent to encode domain knowledge, i.e. domain invariances, stationarity.

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

$x$ : input,  $\theta_l$ : parameters for layer  $l$ ,  $a_l = h_l(x, \theta_l)$ : (non-)linear function

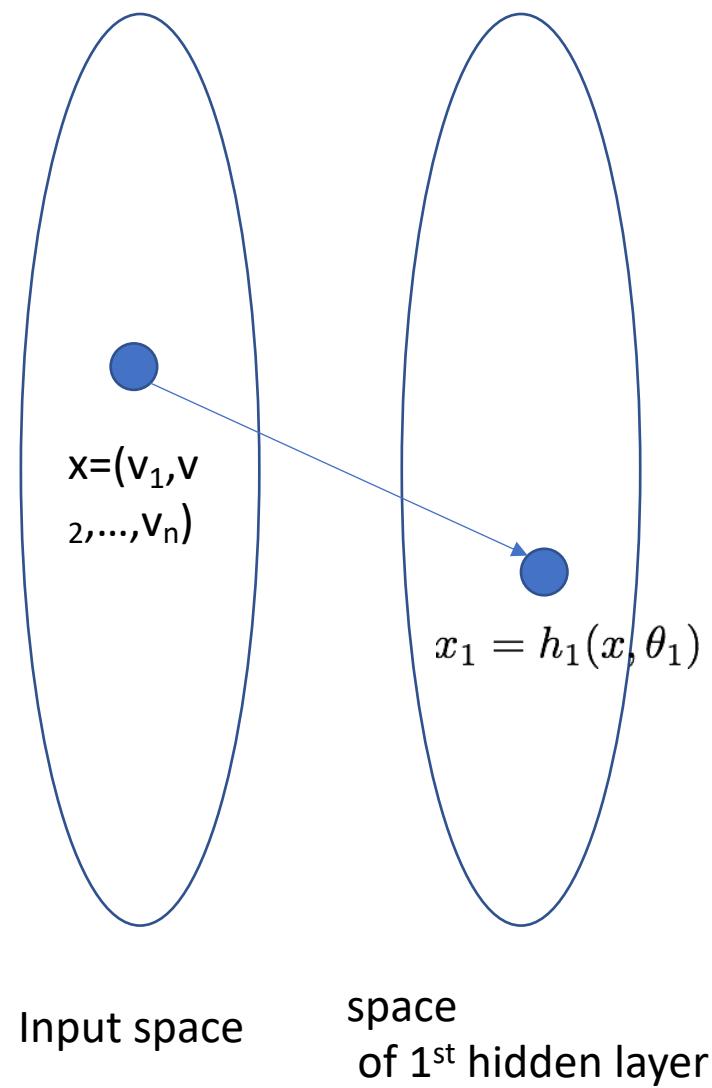


# Illustration of DNN function

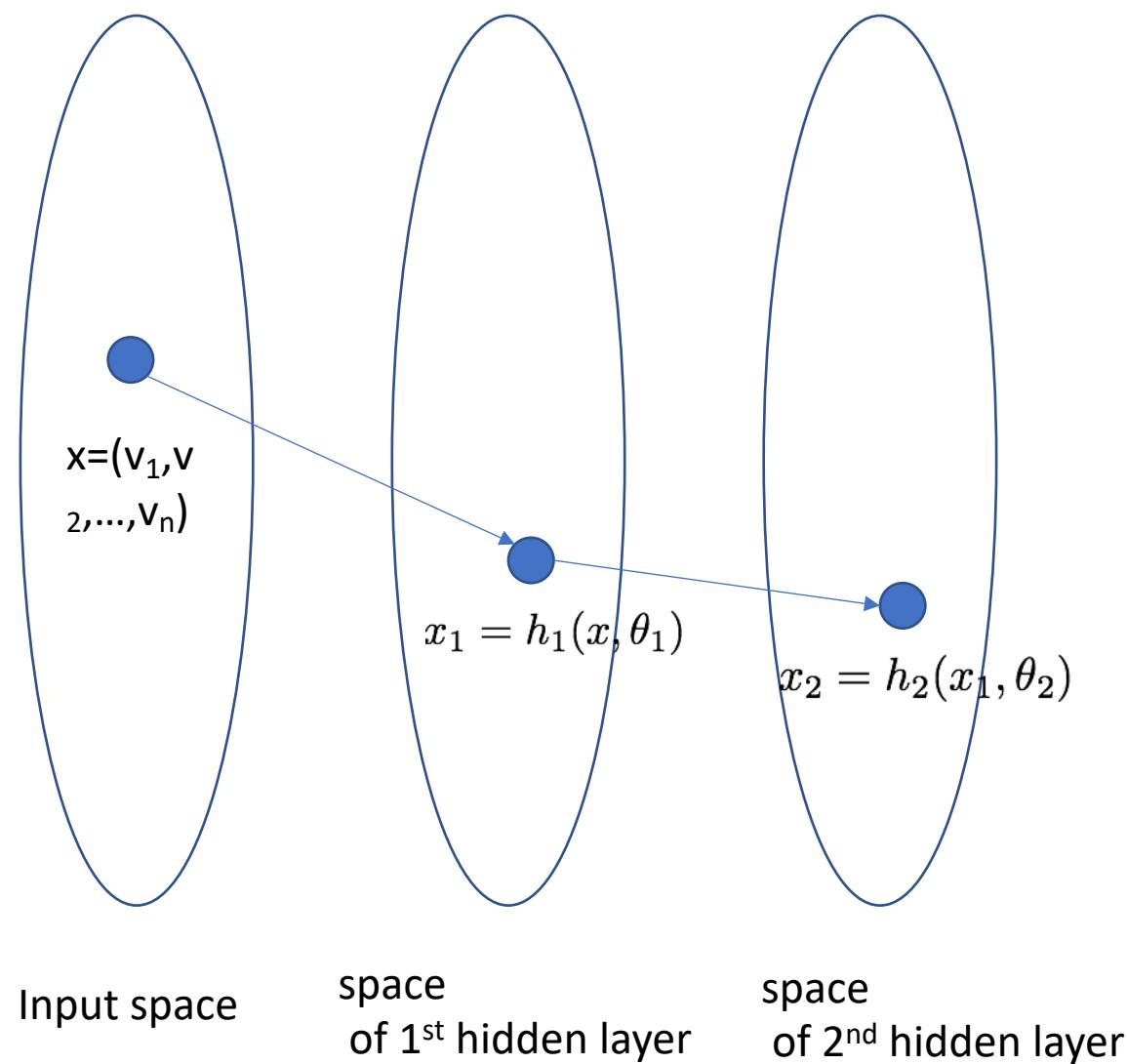


Input space

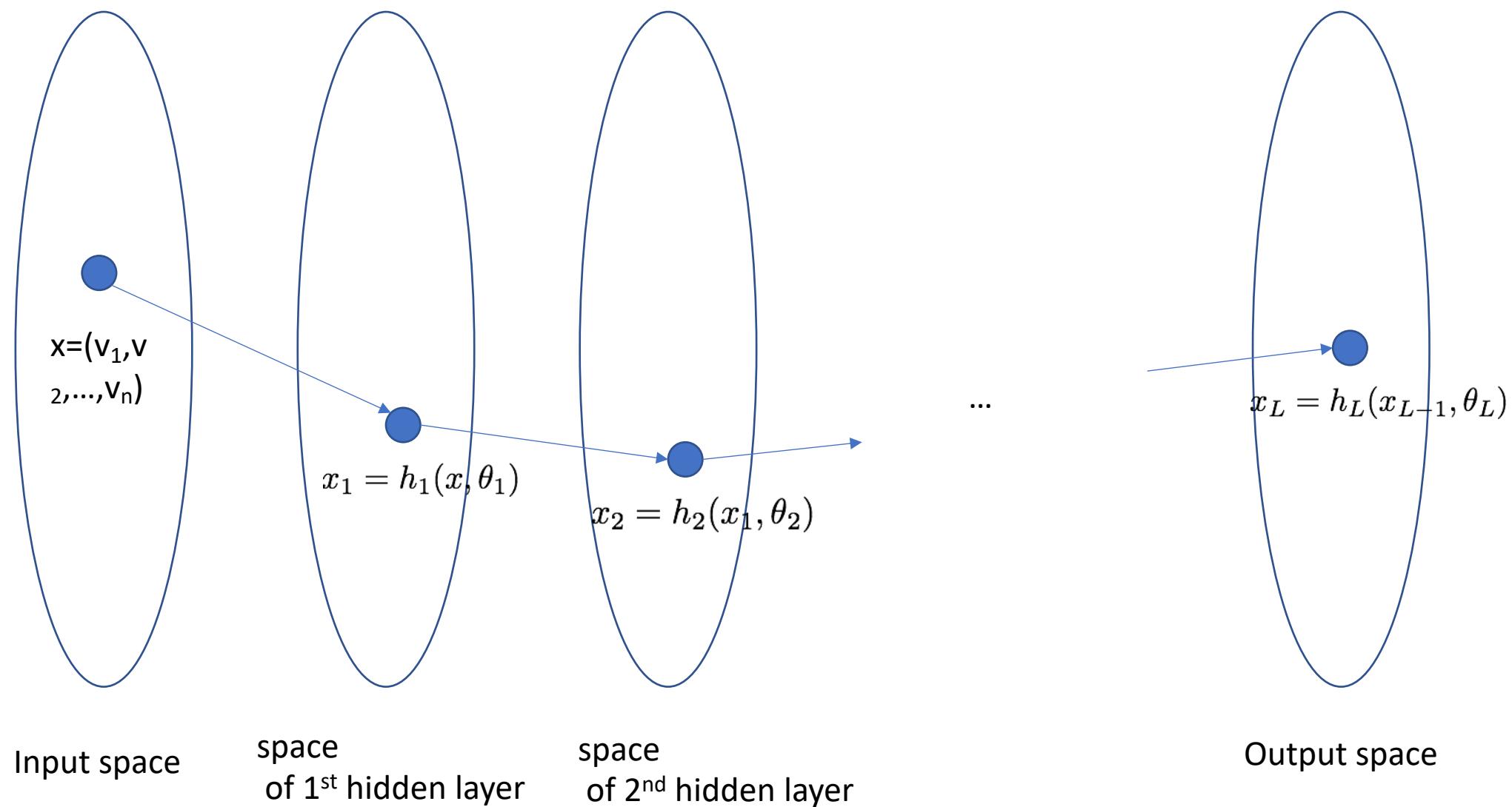
# Illustration of DNN function



# Illustration of DNN function



# Illustration of DNN function



## Note:

- Functions  $h_1, h_2, \dots, h_L$ , are usually given.
- Parameters  $\theta_1, \dots, \theta_L$  are obtained by learning algorithm



# Training Objective

Given training corpus  $\{X, Y\}$  find optimal parameters

Find an optimal model  
parameterised over  $\theta$

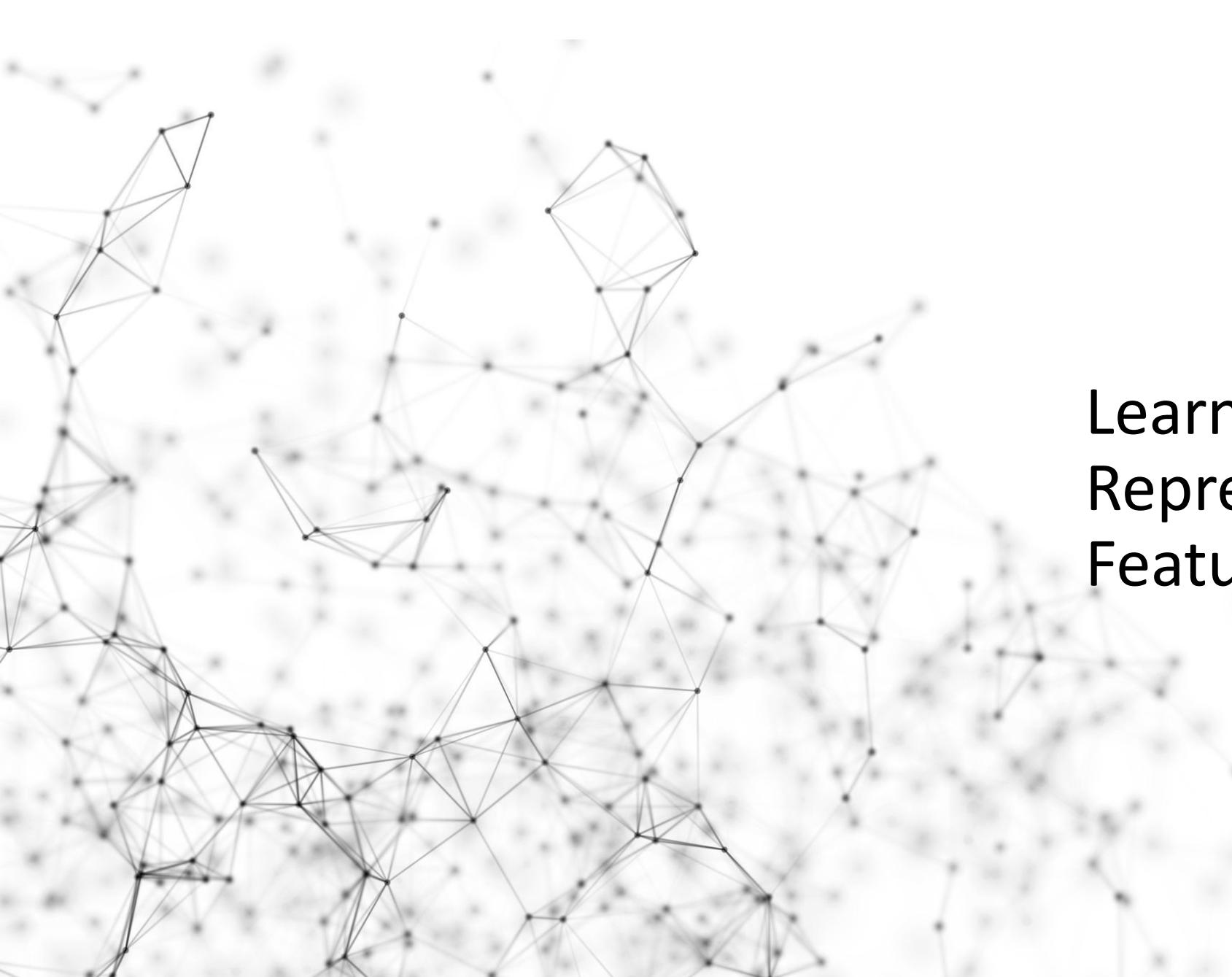
$$\theta^* \leftarrow \arg \min_{\theta} \sum_{(x,y) \subseteq (X,Y)} \ell(y, a_L(x; \theta_{1,\dots,L}))$$

accumulated loss

Ground truth

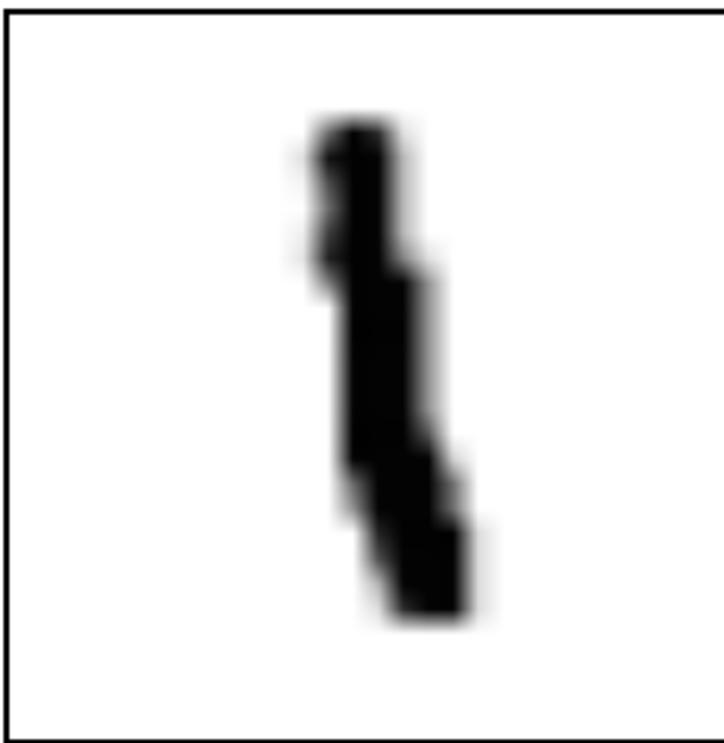
Loss function

Prediction

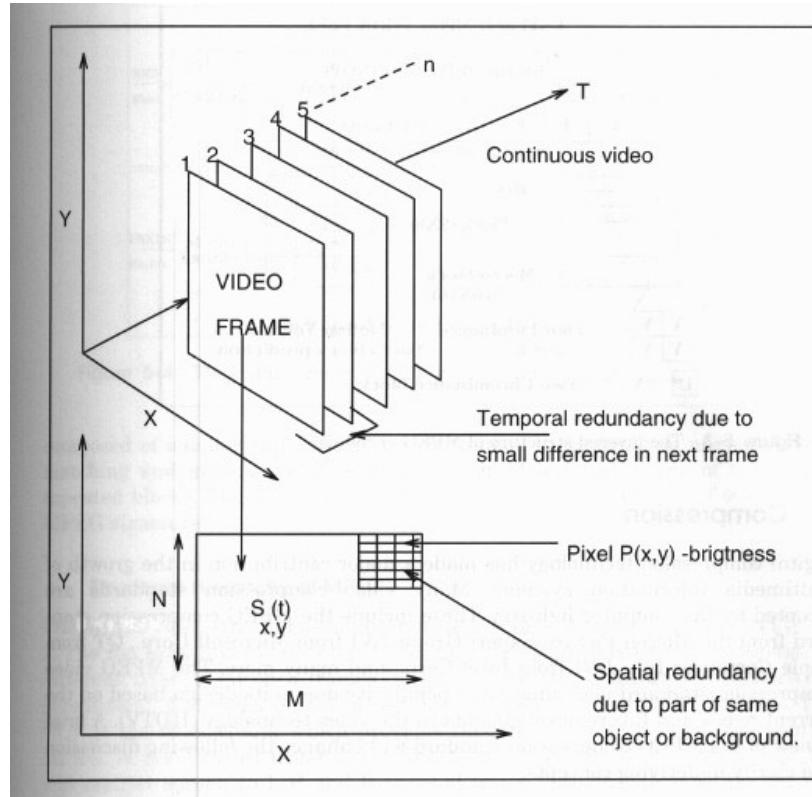
A complex, abstract network graph composed of numerous small, semi-transparent black dots connected by thin gray lines. The graph is highly interconnected, forming a dense web of triangles and larger polygons across the left half of the slide.

# Learning Representations & Features

# Raw digital representation -- Image



# Raw digital representation -- Video

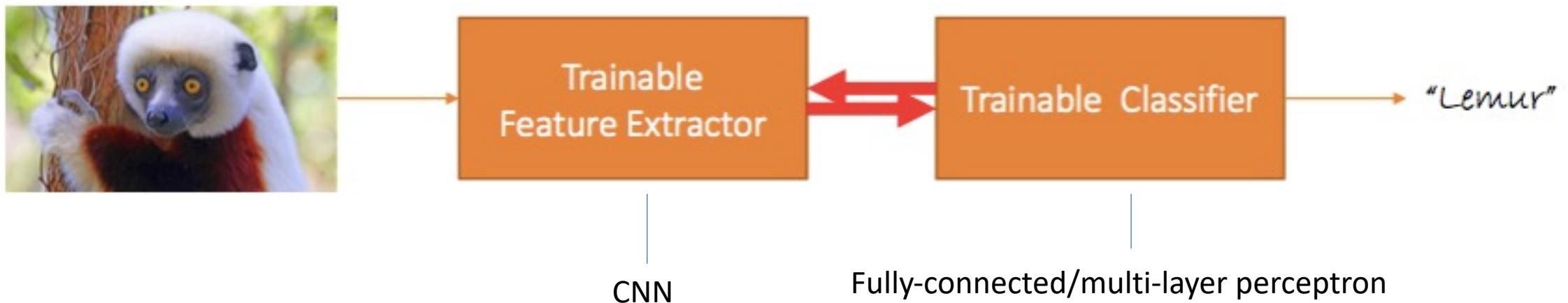


# Learning Representations & Features

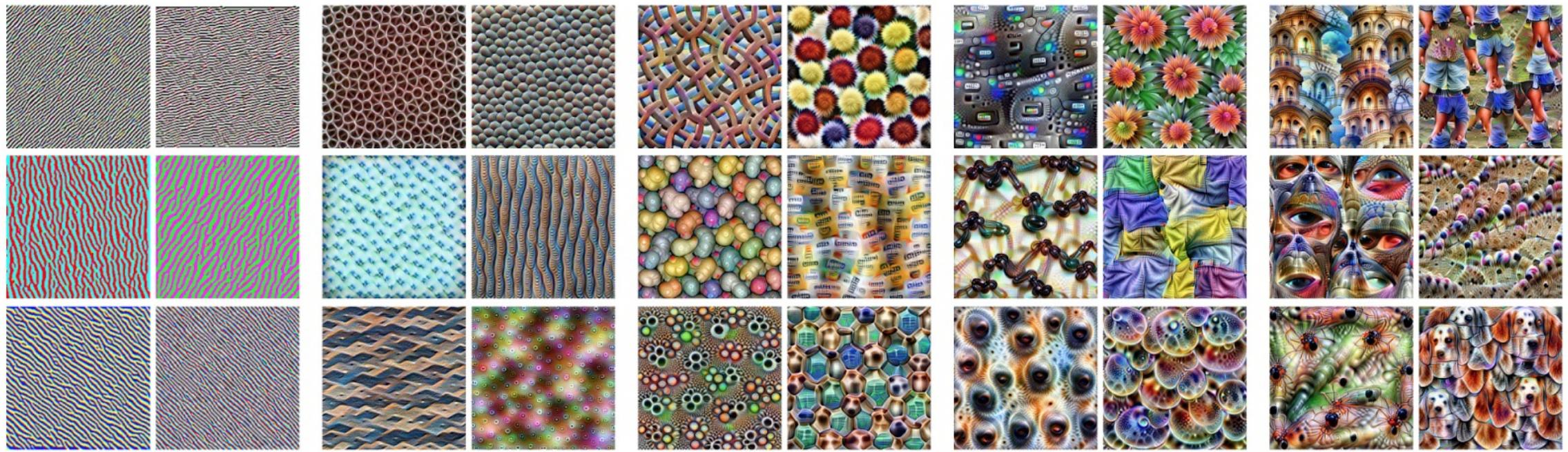
- Traditional pattern recognition



- End-to-end learning Features are also learned from data

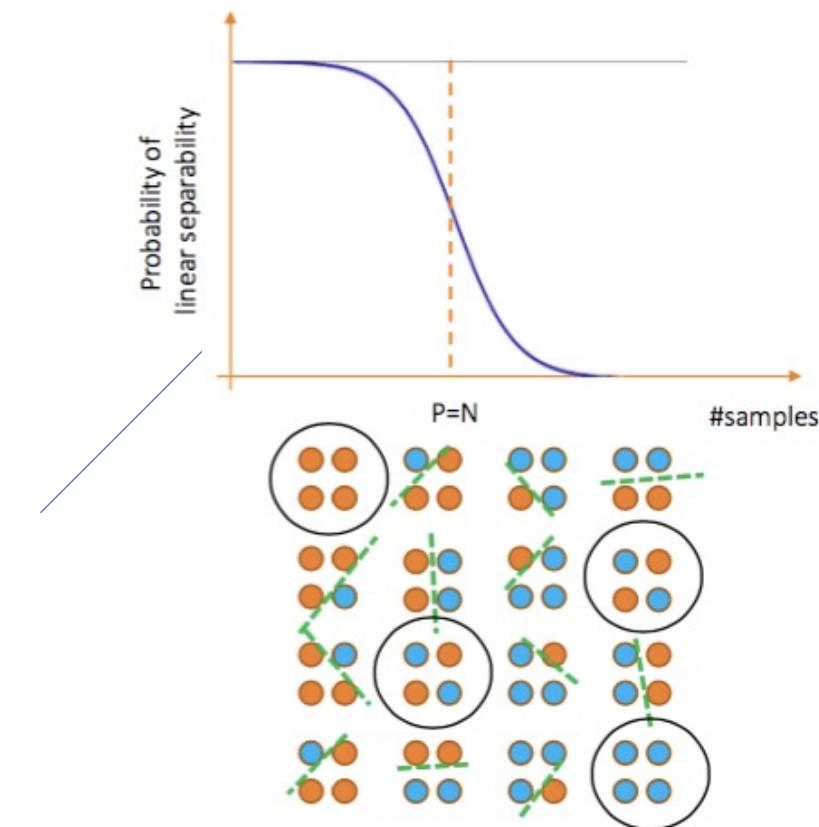


# Feature visualization by CNN



# Non-separability of linear machines

- $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^d$
- Given the  $n$  points there are in total  $2^n$  dichotomies
- Only about  $d$  are linearly separable
- With  $n > d$  the probability  $X$  is linearly separable converges to 0 very fast

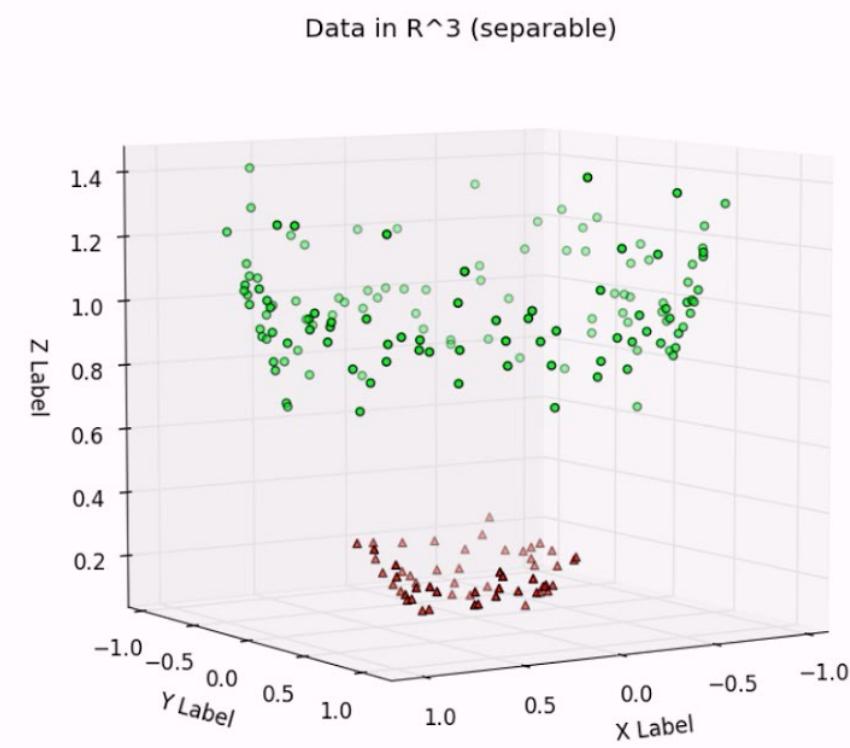
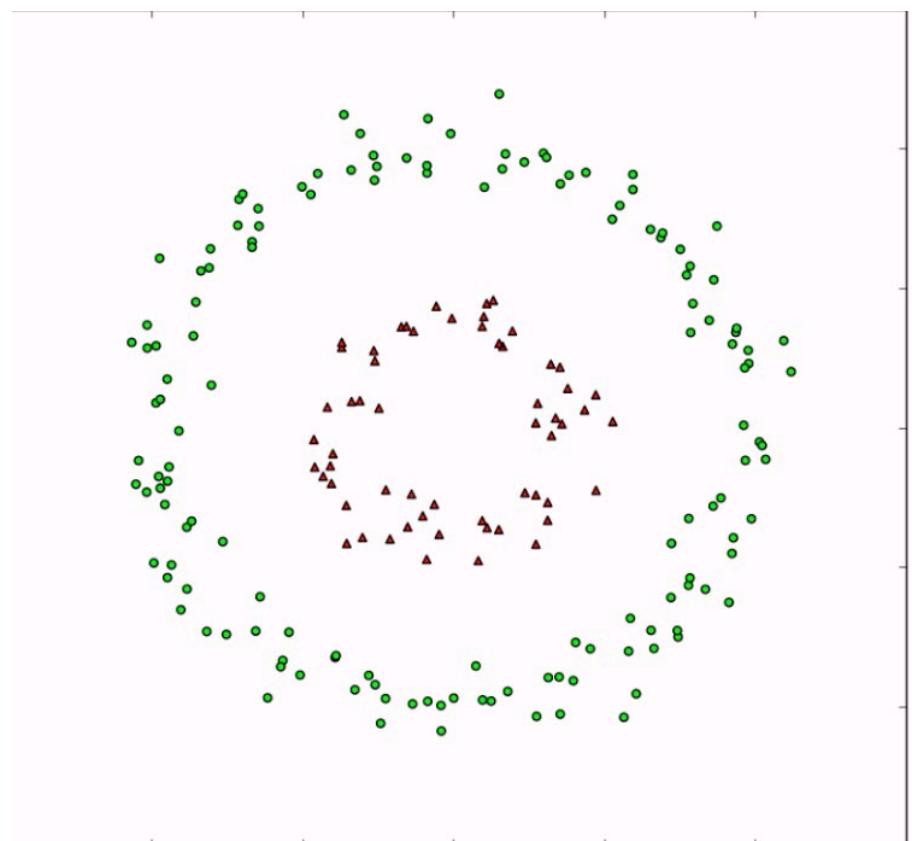


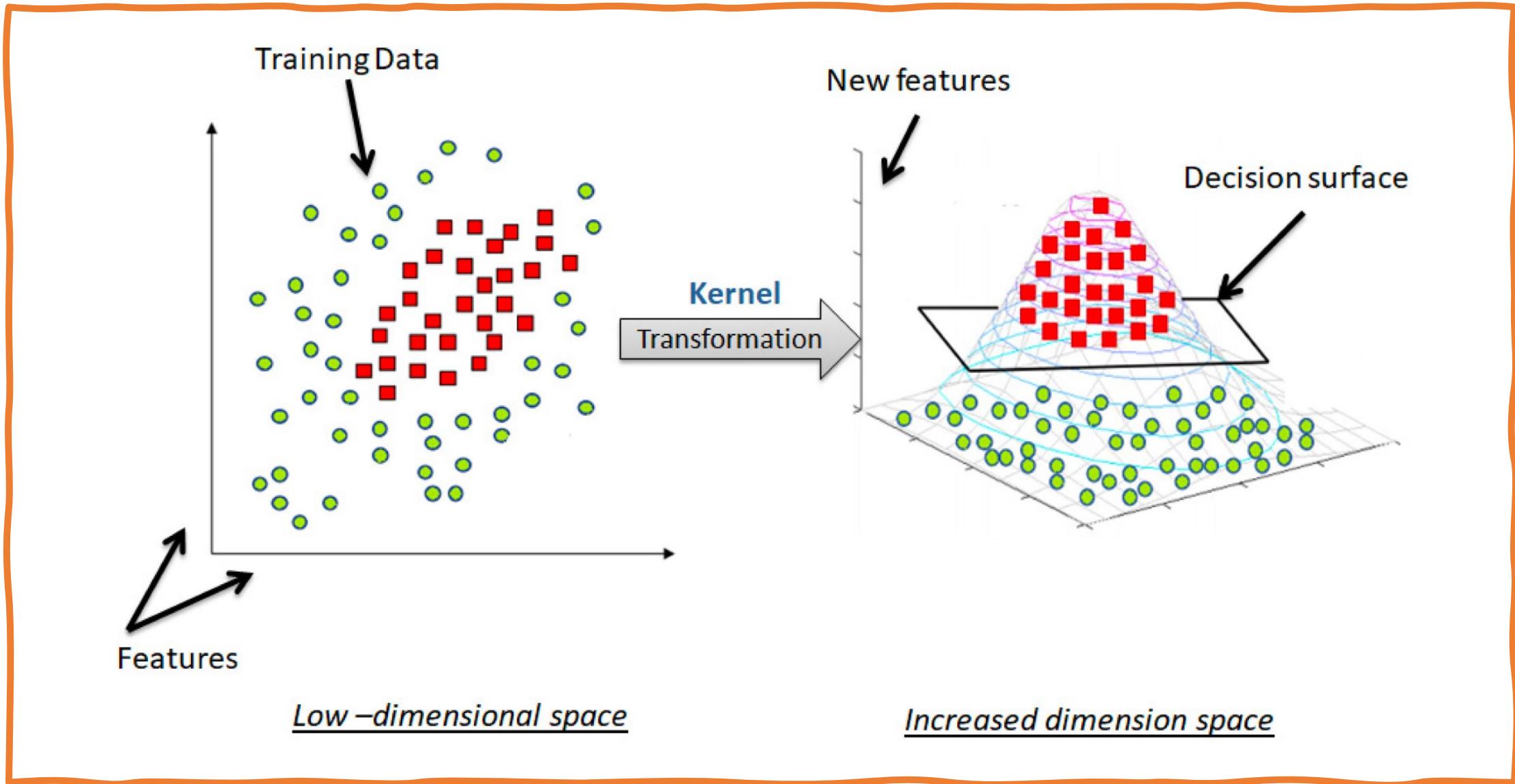
# Non-linearizing linear machines

---

- Most data distributions and tasks are non-linear
- A linear assumption is often convenient, but not necessarily truthful
- Problem: How to get non-linear machines without too much effort?
- Solution: Make features non-linear
- What is a good non-linear feature?
  - Non-linear kernels, e.g., polynomial, RBF, etc
  - Explicit design of features (SIFT, HOG)?

# Kernel: low dimension $\rightarrow$ high dimension







SIFT



# Good features

Invariant

- But not too invariant

Repeatable

- But not bursty

Discriminative

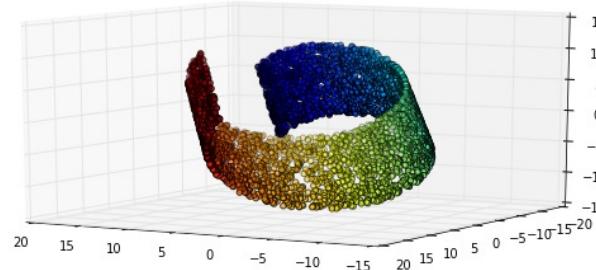
- But not too class-specific

Robust

- But sensitive enough

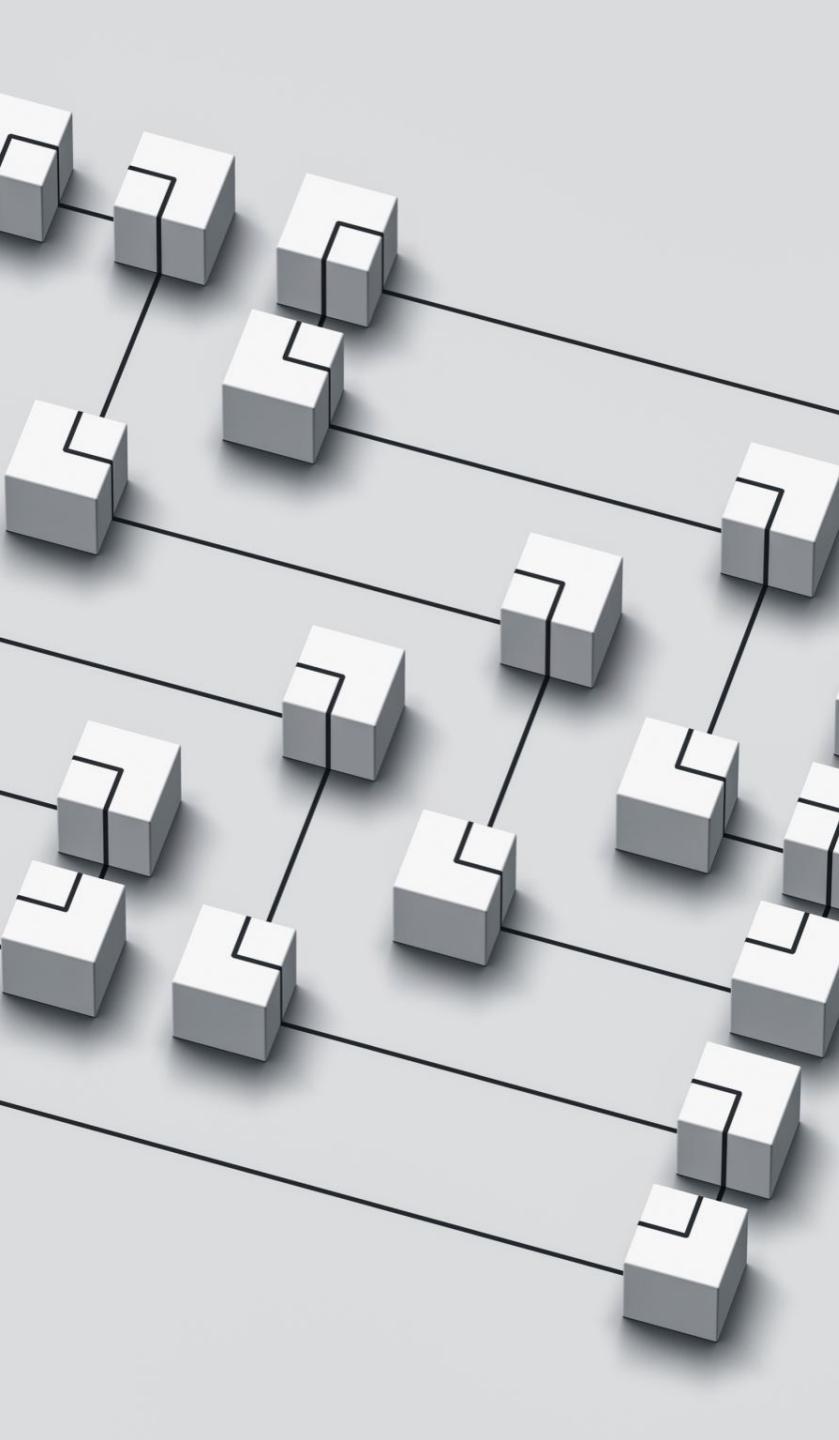
# Data manifold

- High-dimensional data (e.g. faces) lie on lower dimensional manifolds



Every point  
represents an  
input sample.

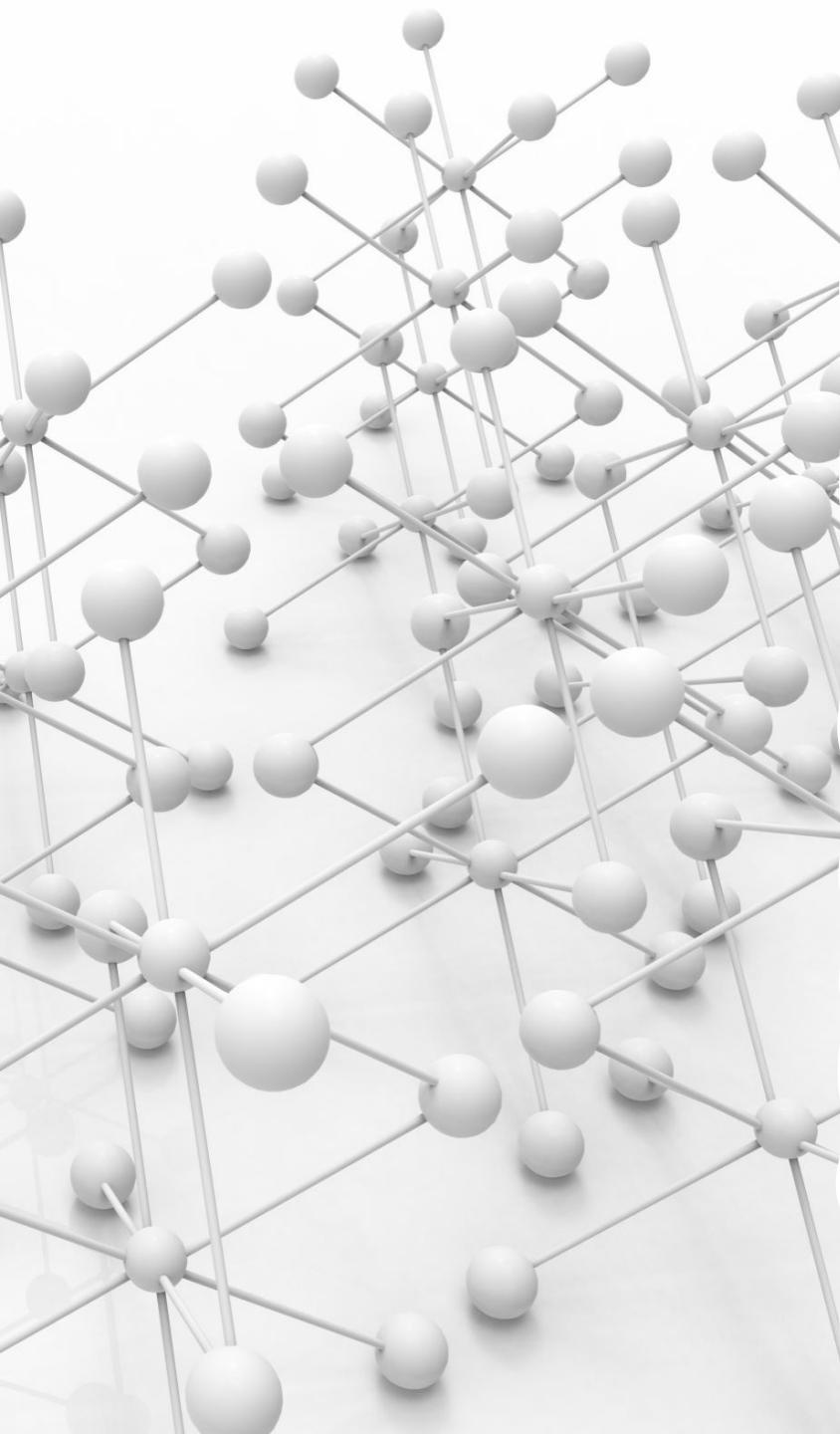
- This is so-called "swiss roll". The data points are in 3d, but they all lie on 2d manifold, so the dimensionality of the manifold is 2, while the dimensionality of the input space is 3.



## Data manifold is usually lower dimensional than the original

---

- High-dimensional data (e.g. faces) lie on lower dimensional manifolds
- Although the data points may consist of thousands of features, they may be described as a function of only a few underlying parameters.
  - That is, the data points are actually samples from a low-dimensional manifold that is embedded in a high-dimensional space.
- Goal: discover these lower dimensional manifolds
  - These manifolds are most probably highly non-linear



# Hypothesis

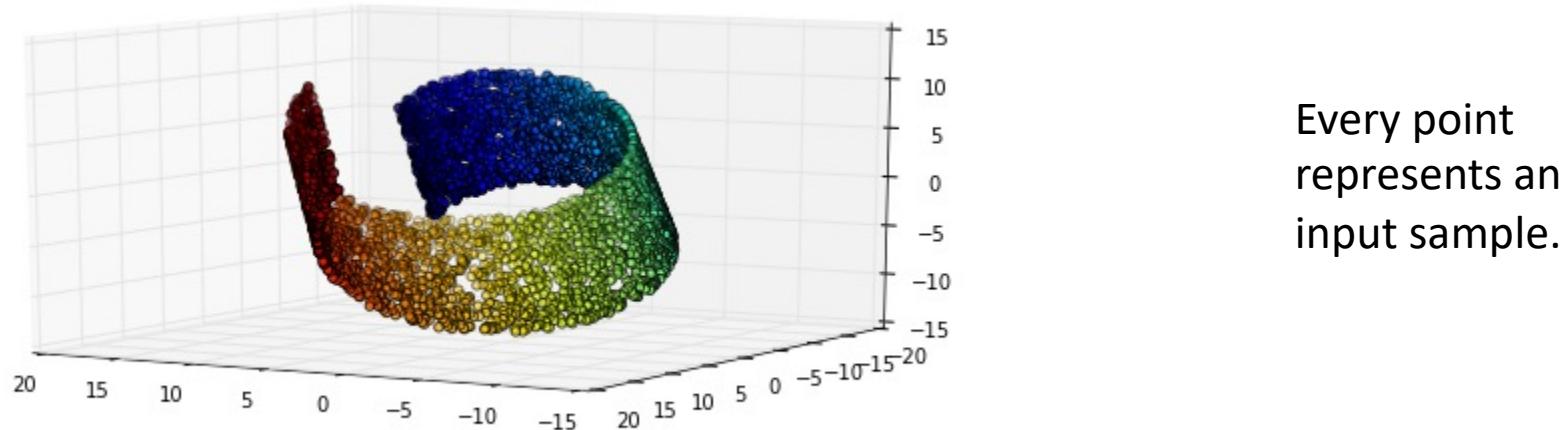
---

- High-dimensional data (e.g. faces) lie on lower dimensional manifolds
  - Goal: discover these lower dimensional manifolds
  - These manifolds are most probably highly non-linear
- Hypothesis (1): Compute the coordinates of the input (e.g. a face image) to this non-linear manifold -> data become separable
- Hypothesis (2): Semantically similar things lie closer together than semantically dissimilar things

# Hypothesis (1) -> existence of functional mapping

---

- High-dimensional data (e.g. faces) lie in lower dimensional manifolds

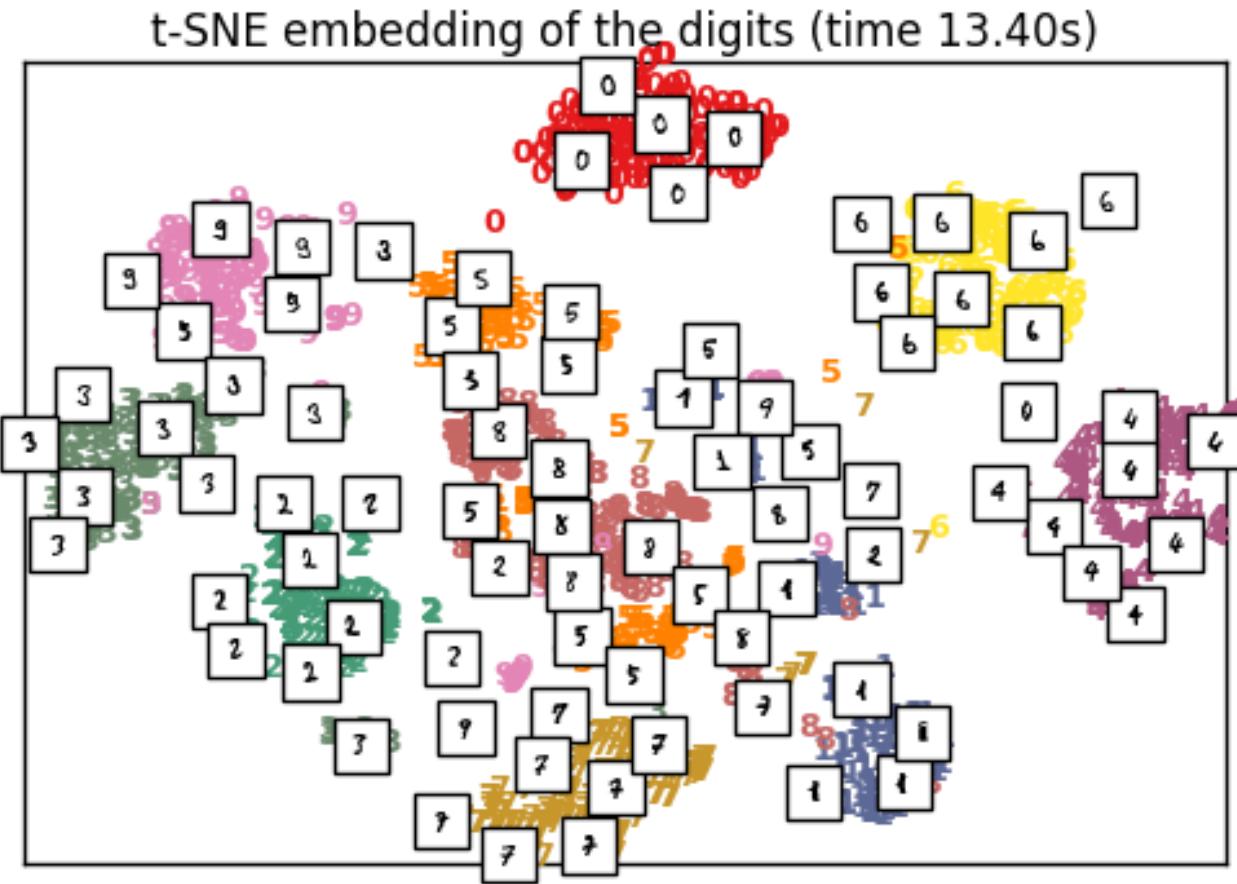


- So there should be a (non-linear) function mapping from 3d space to 2d space, on which the data can be linearly separable.

Hypothesis (2) -> some existing dimensional reduction methods

---

- It is not linear, but can be largely separated with a dimensionality much less than 28\*28



Hypothesis (2) -  
> some existing  
dimensional  
reduction  
methods

---

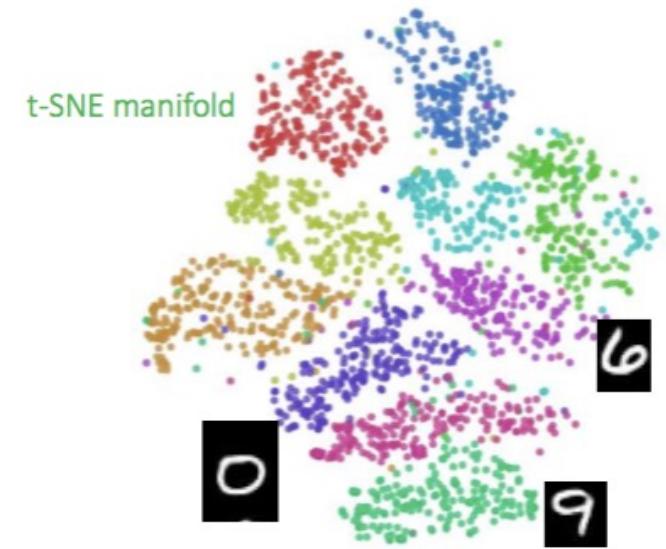
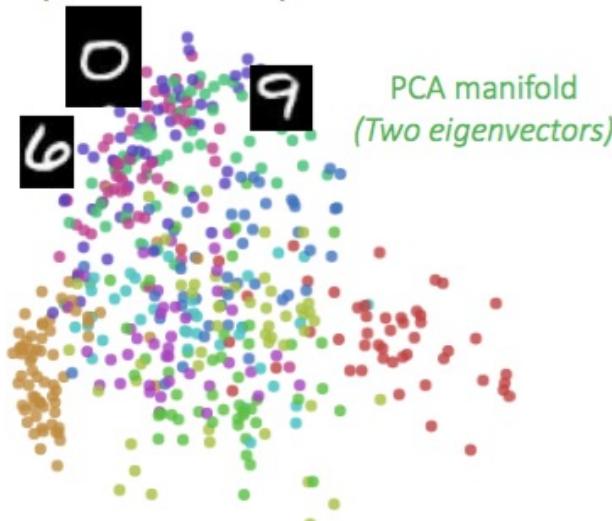
PCA (Principle Component  
Analysis), 2-dimensional

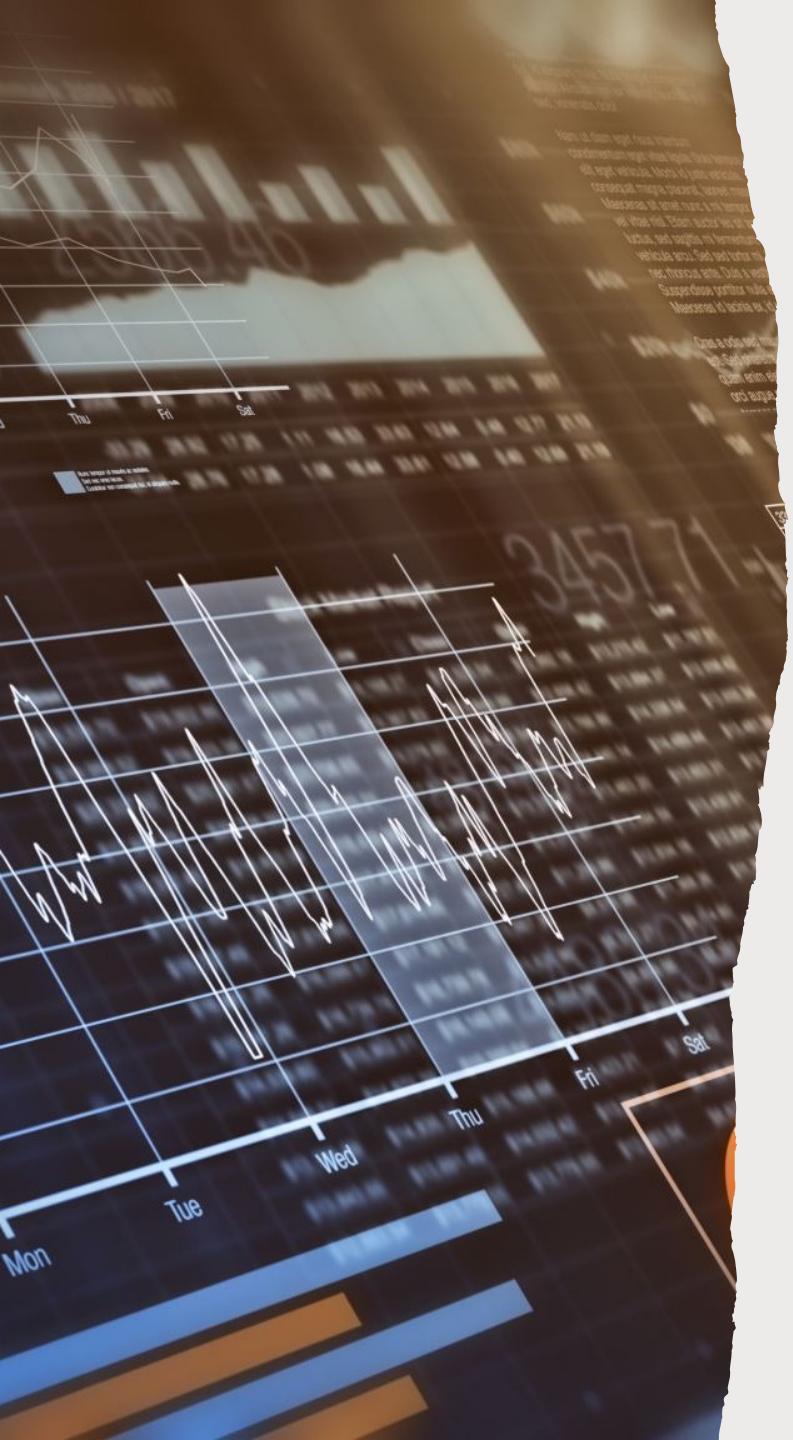


# The digits manifolds

---

- There are good features and bad features, good manifold representations and bad manifold representations
- $28 \text{ pixels} \times 28 \text{ pixels} = 784$  dimensions



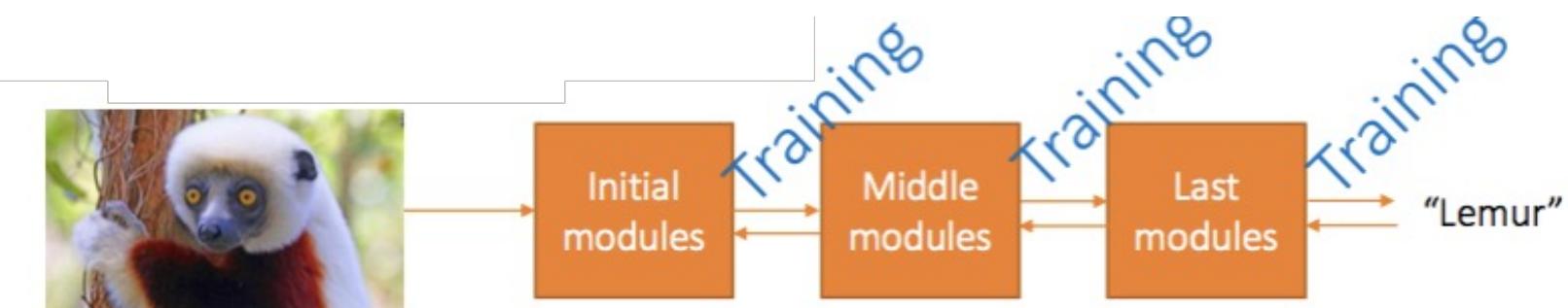


# Difficulties of simply using dimensionality reduction or kernel

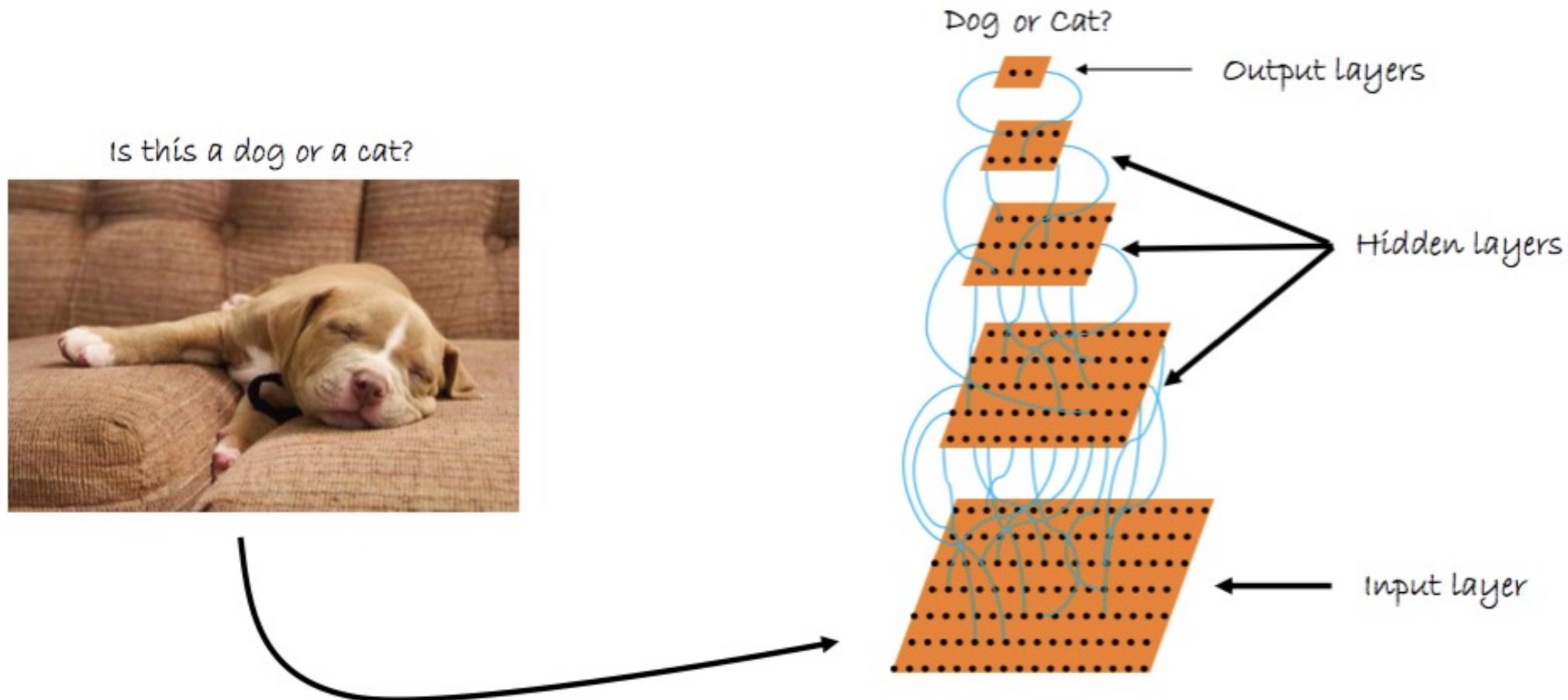
- Raw data live in huge dimensionalities
- Semantically meaningful raw data prefer lower dimensional manifolds
  - Which still live in the same huge dimensionalities
- Can we discover this manifold to embed our data on?

# End-to-end learning of feature hierarchies

- A pipeline of successive modules
- Each module's output is the input for the next module
- Modules produce features of higher and higher abstractions
  - Initial modules capture low-level features (e.g. edges or corners)
  - Middle modules capture mid-level features (e.g. circles, squares, textures)
  - Last modules capture high level, class specific features (e.g. face detector)
- Preferably, input as raw as possible
  - Pixels for computer vision, words for NLP



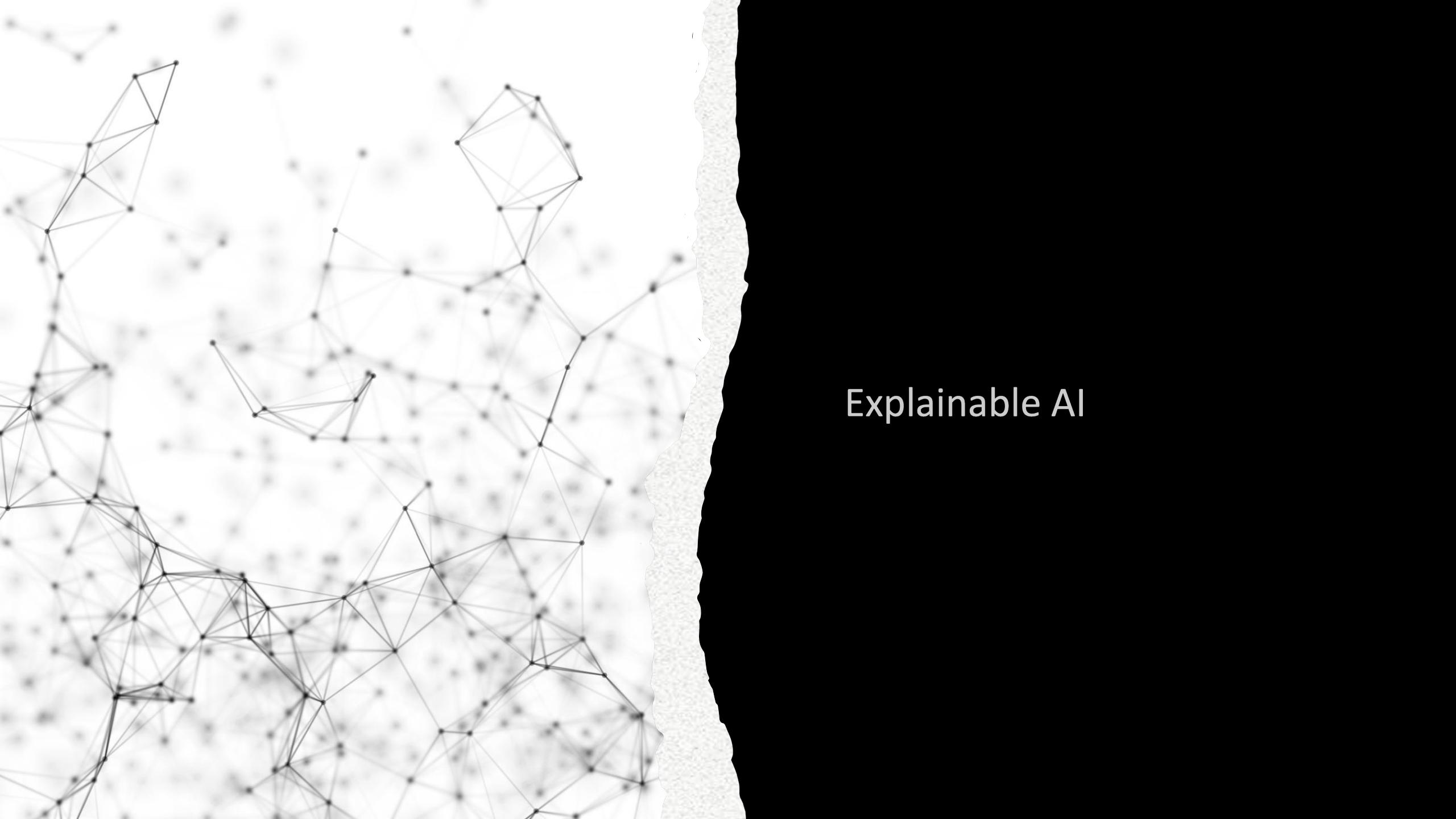
# Convolutional networks in a nutshell



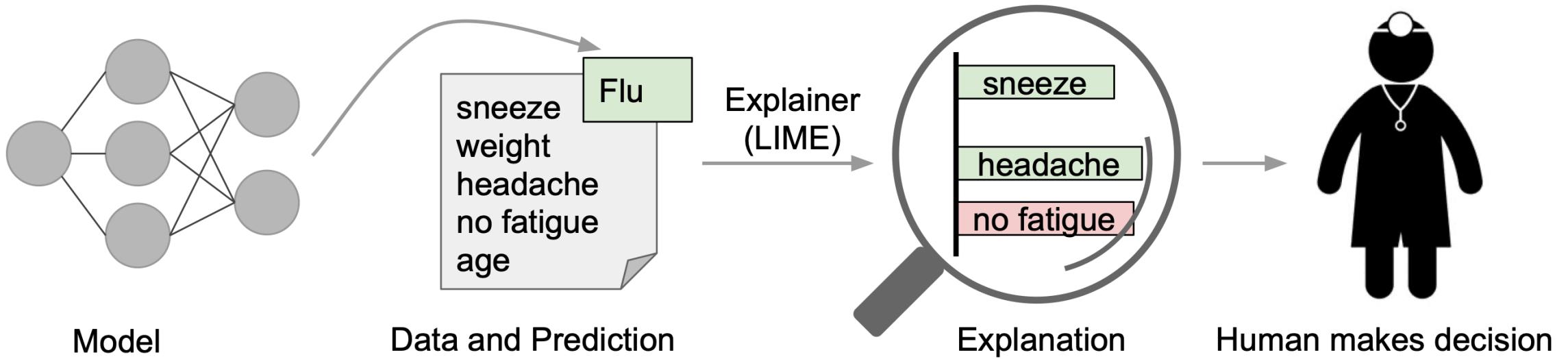
# Why learn the features?

---

- Manually designed features
  - Often take a lot of time to come up with and implement
  - Often take a lot of time to validate
  - Often they are incomplete, as one cannot know if they are optimal for the task
- Learned features
  - Are easy to adapt
  - Very compact and specific to the task at hand
  - Given a basic architecture in mind, it is relatively easy and fast to optimize
- Time spent for designing features now spent for designing architectures



Explainable AI



# LIME

- Explaining individual predictions.
- A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient's history that led to the prediction.
- Sneeze and headache are portrayed as contributing to the “flu” prediction, while “no fatigue” is evidence against it.
- With these, a doctor can make an informed decision about whether to trust the model’s prediction.



# LIME

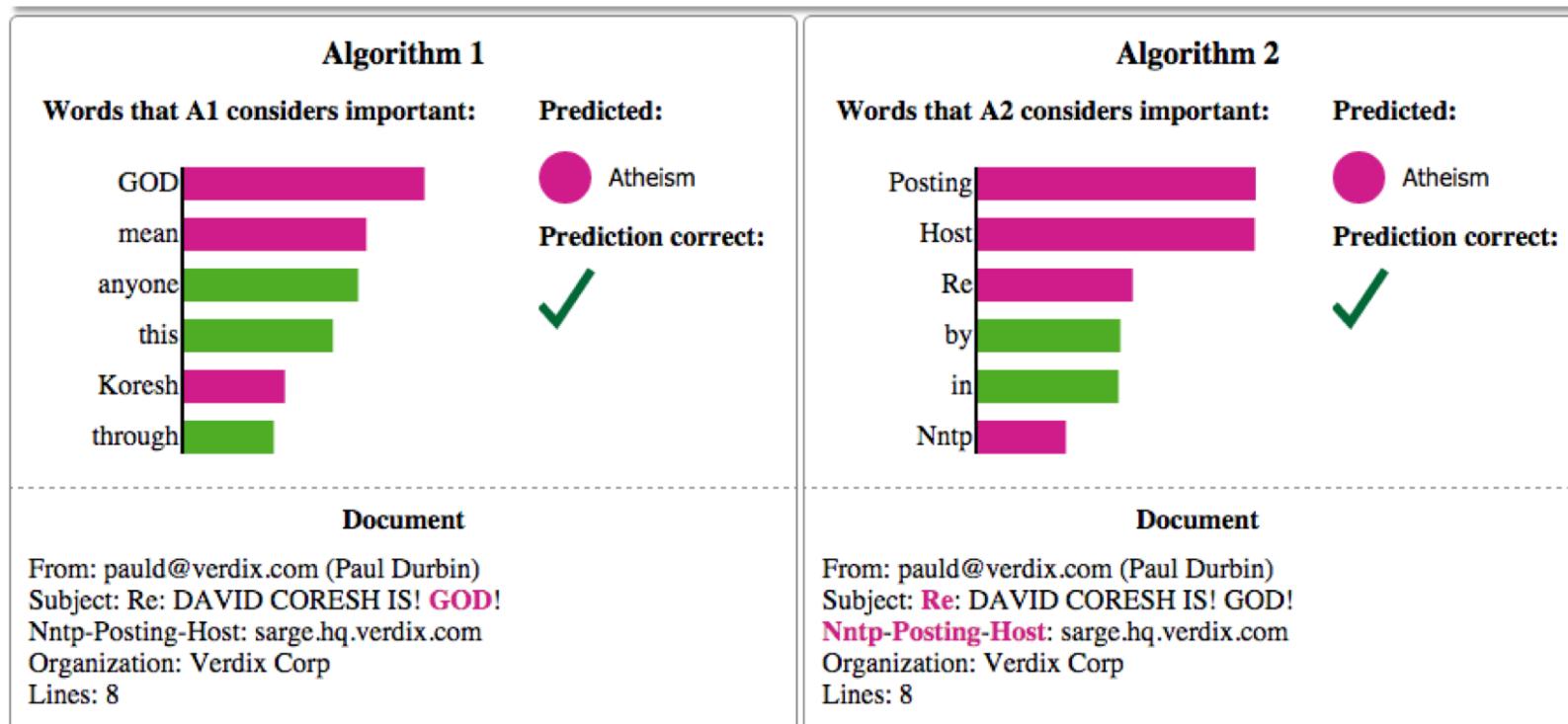
Example #3 of 6

True Class:  Atheism

[Instructions](#)

[Previous](#)

[Next](#)



- Explaining individual predictions of competing classifiers trying to determine if a document is about “Christianity” or “Atheism”.
- The bar chart represents the importance given to the most relevant words, also highlighted in the text.
- Color indicates which class the word contributes to (green for “Christianity”, magenta for “Atheism”).

# LIME



(a) Original Image



(b) Explaining *Electric guitar*



(c) Explaining *Acoustic guitar*



(d) Explaining *Labrador*

- Explaining an image classification prediction made by Google's Inception neural network.
- The top 3 classes predicted are "Electric Guitar" ( $p = 0.32$ ), "Acoustic guitar" ( $p = 0.24$ ) and "Labrador" ( $p = 0.21$ )