Algorithms

Deep learning

Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

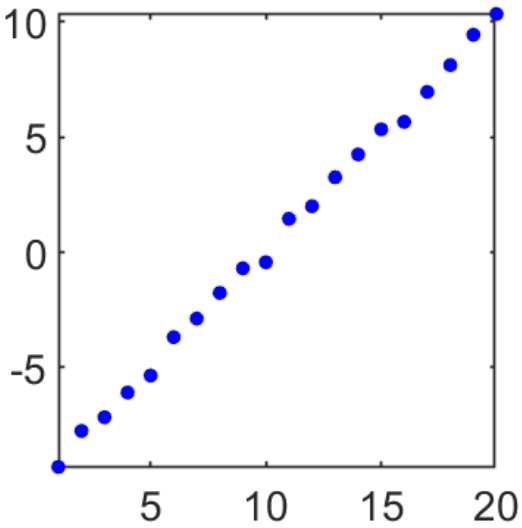
2nd semester a.y. 2020/2021 · April 29, 2021

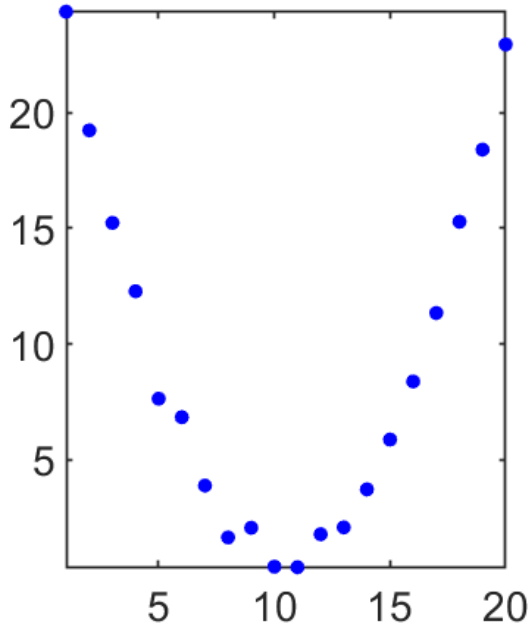
Models for describing the data

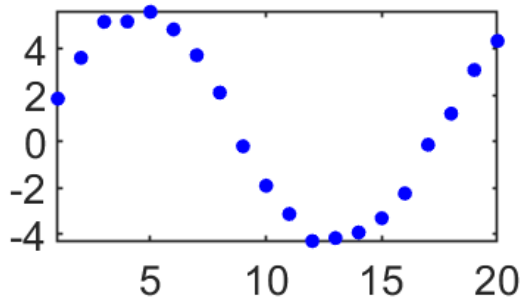
Learning is about describing data, or more specifically, describing the process, or model, that yields a given output from a given input.

Models for describing the data

Learning is about describing data, or more specifically, describing the process, or model, that yields a given output from a given input.

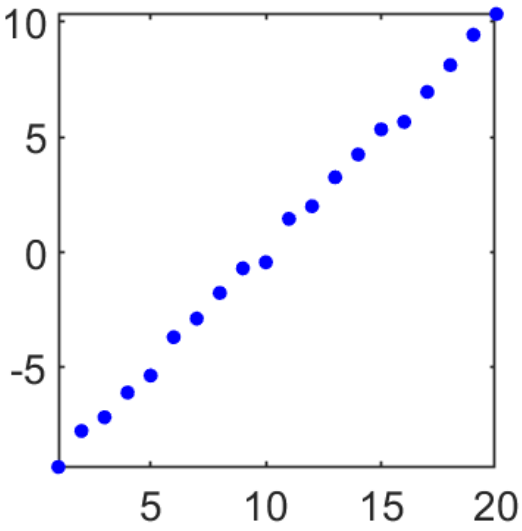




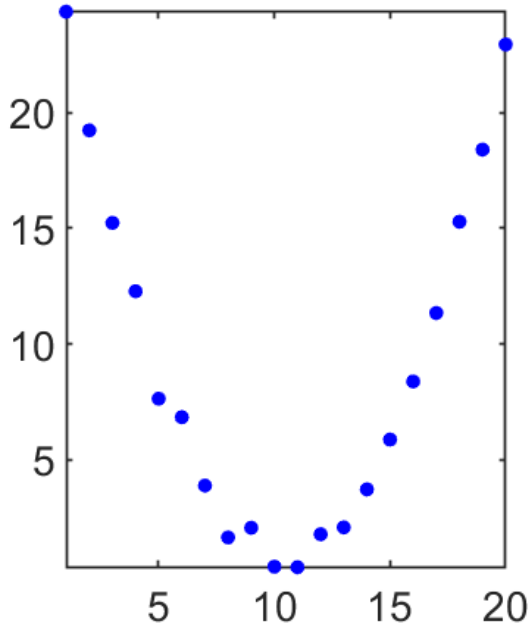


Models for describing the data

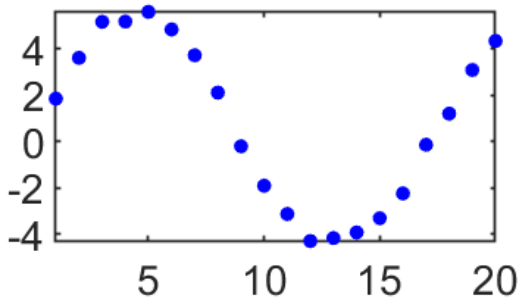
Learning is about describing data, or more specifically, describing the process, or model, that yields a given output from a given input.



$y = ax + b$



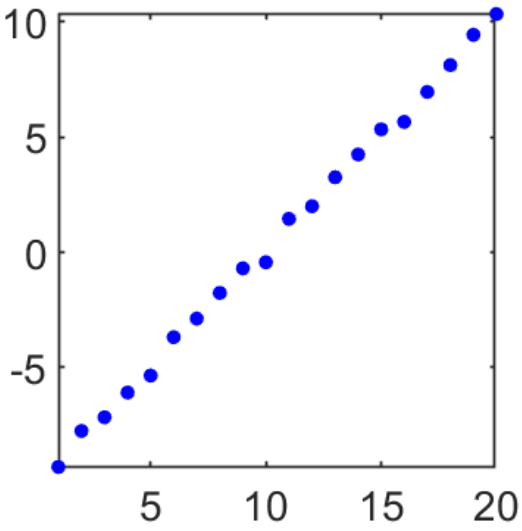
$y = ax^2 + bx + c$

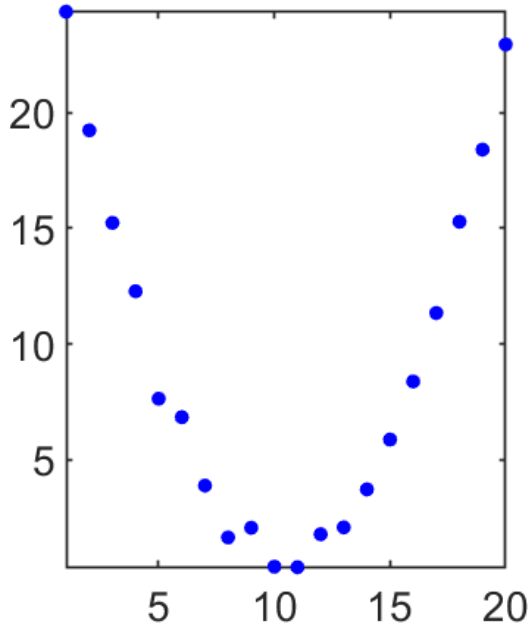


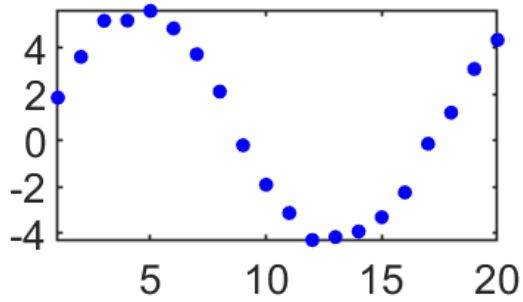
$y = ax^3 + bx^2 + cx + d$

Models for describing the data

Learning is about describing data, or more specifically, describing the process, or model, that yields a given output from a given input.







$y = ax + b$ $y = ax^2 + bx + c$ $y = a\sin(x) + bx + c$

Our model might use prior knowledge on the data.

For example, in the third plot, we might know *a priori* that the data actually comes from a periodic process.

Modeling prior knowledge

Main idea: Look at the world, identify what knowledge we have about it, and use this knowledge to construct our model.

Modeling prior knowledge

Main idea: Look at the world, identify what knowledge we have about it, and use this knowledge to construct our model.

Some forms of prior knowledge:

- Data distribution
- Energy function
- Constraints
- Invariances
- Input-output examples (data prior)

All these encode, to different extents, some expected behavior.

Explaining the data

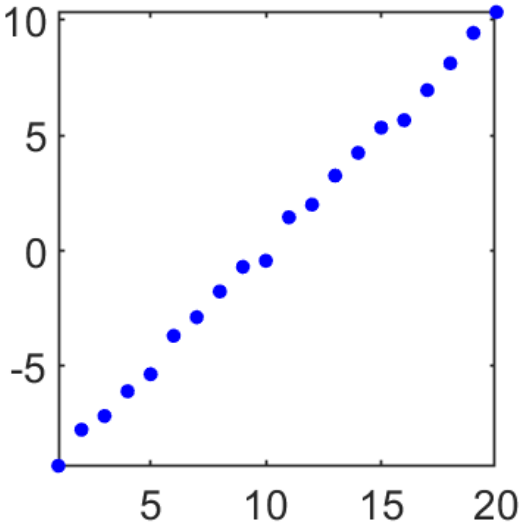
Learning is about discovering a **map** from input to output.

Finding a model explaining the data means determining the map.

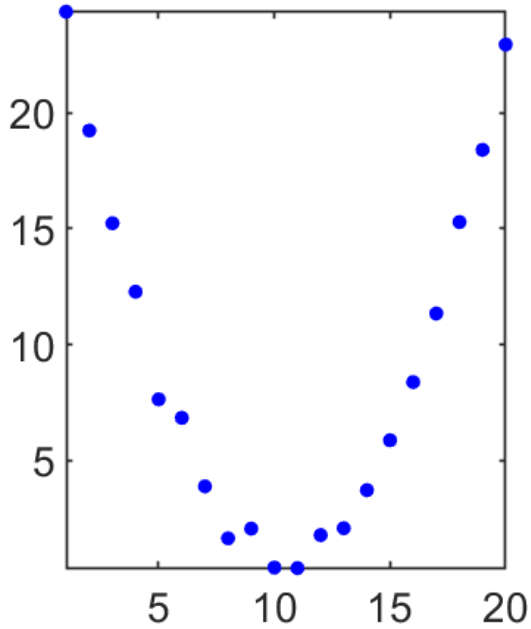
Explaining the data

Learning is about discovering a **map** from **input** to **output**.

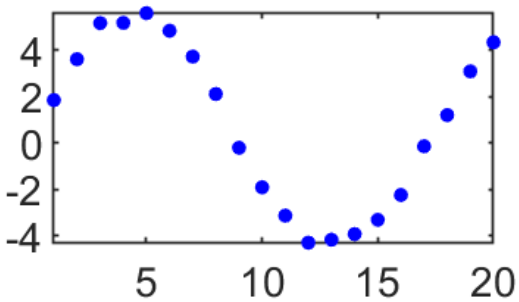
Finding a model explaining the data means determining the map.



$y = ax + b$



$y = ax^2 + bx + c$



$y = a\sin(x) + bx + c$

Explaining the data

Learning is about discovering a **map** from input to output.

Finding a model explaining the data means determining the map.

Key assumption: the data has an underlying structure.

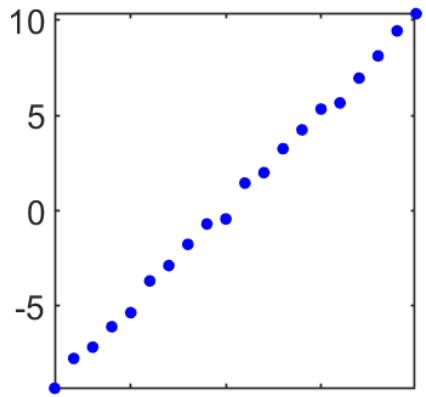
Explaining the data

Learning is about discovering a **map** from input to output.

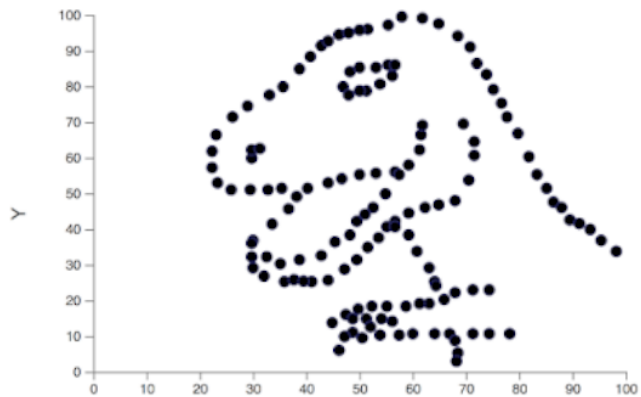
Finding a model explaining the data means determining the map.

Key assumption: the data has an underlying structure.

This structure is almost never captured by a simple expression.



$y = ax + b$



$?$

The curse of dimensionality

Of course, data can have more than 1 or 2 dimensions.

The curse of dimensionality

Of course, data can have more than 1 or 2 dimensions.

For example, a $w \times h$ image has wh dimensions, i.e., it is a point in a wh-dimensional space. A dataset of such images is a point cloud in $\mathbb{R}^{wh}$.

$\in \mathbb{R}^{w \times h} \cong \mathbb{R}^{wh}$

Example: $\sim$1 megapixel photo (grayscale) has $\sim 10^6$ dimensions.

The curse of dimensionality

Of course, data can have more than 1 or 2 dimensions.

For example, a $w \times h$ image has wh dimensions, i.e., it is a point in a wh-dimensional space. A dataset of such images is a point cloud in $\mathbb{R}^{wh}$.

$\in \mathbb{R}^{w \times h} \cong \mathbb{R}^{wh}$

Example: $\sim$1 megapixel photo (grayscale) has $\sim 10^6$ dimensions.

Are all those dimensions significant?

The curse of dimensionality

For simplicity, consider 1×1 images, i.e., consisting of one single pixel.

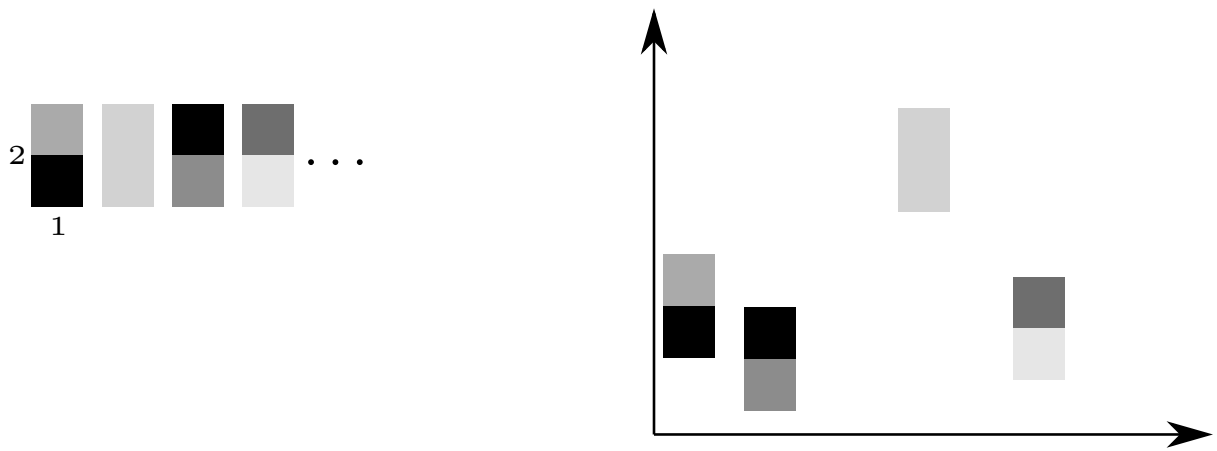
There is only one dimension; each image is a point along one axis.

The curse of dimensionality

For simplicity, consider 1×1 images, i.e., consisting of one single pixel.

There is only one dimension; each image is a point along one axis.

Similarly, with 2 pixels we get:

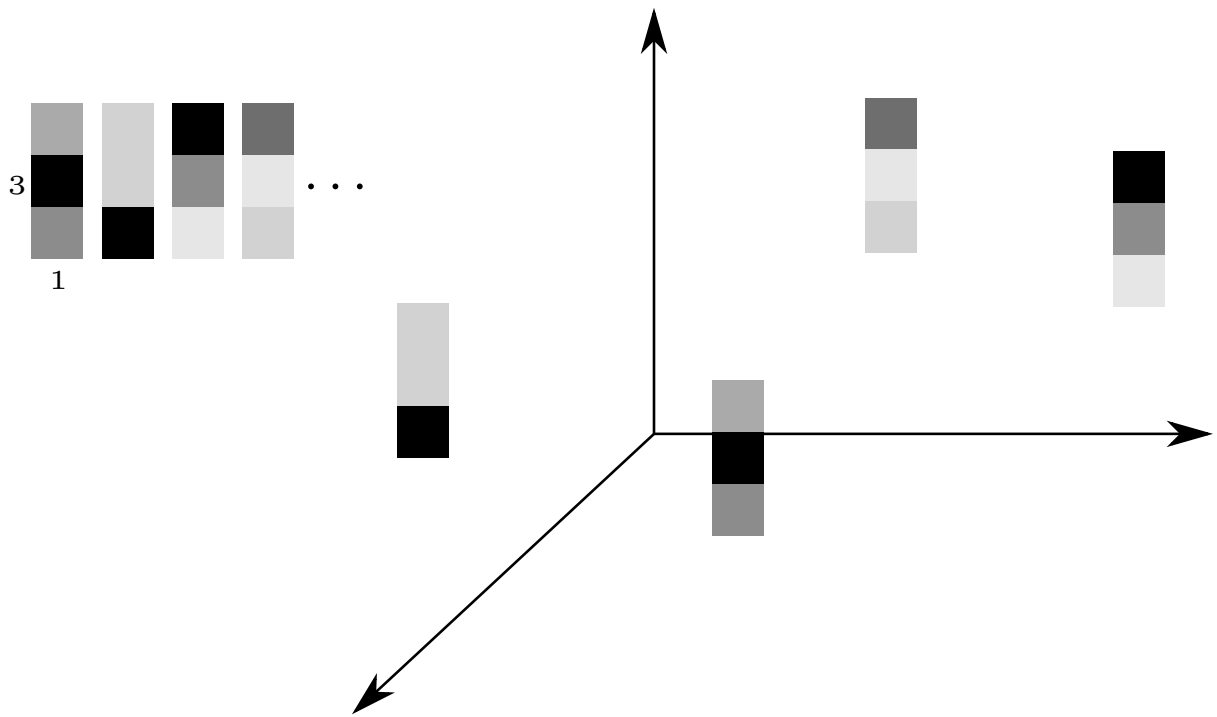


The curse of dimensionality

For simplicity, consider 1×1 images, i.e., consisting of one single pixel.

There is only one dimension; each image is a point along one axis.

Similarly, with 3 pixels we get:

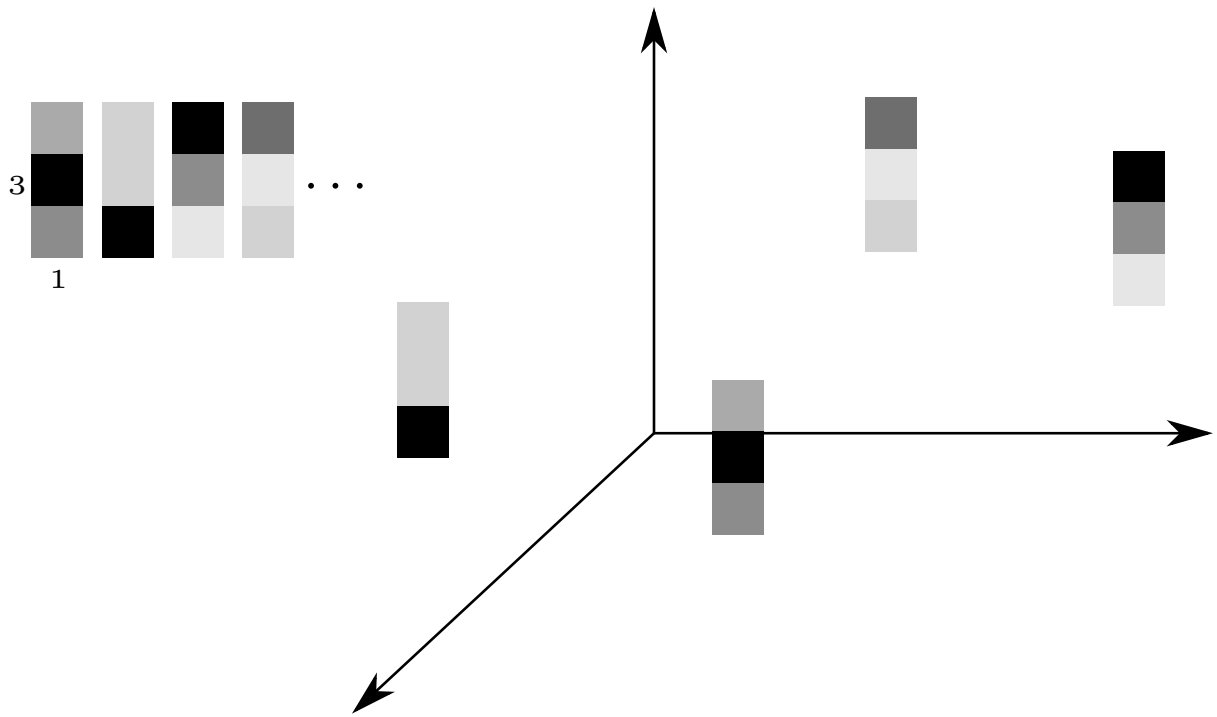


The curse of dimensionality

For simplicity, consider 1×1 images, i.e., consisting of one single pixel.

There is only one dimension; each image is a point along one axis.

Similarly, with 3 pixels we get:



Each new dimension increases sparsity of the point cloud.

The curse of dimensionality

A dataset of natural images will be extremely sparse in $\mathbb{R}^{w \times h}$, since each region of space is observed very infrequently.

New samples are less likely to fall close to the previous ones.

The curse of dimensionality

A dataset of natural images will be extremely sparse in $\mathbb{R}^{w \times h}$, since each region of space is observed very infrequently.

New samples are less likely to fall close to the previous ones.

As a consequence, all images will approximately be equally spaced

$\Rightarrow$ no meaningful structure will emerge from the dataset.

The curse of dimensionality

A dataset of natural images will be extremely sparse in $\mathbb{R}^{w \times h}$, since each region of space is observed very infrequently.

New samples are less likely to fall close to the previous ones.

As a consequence, all images will approximately be equally spaced

$\Rightarrow$ no meaningful structure will emerge from the dataset.

We would need exponentially many observations as we have dimensions!

If n data points cover well the space of 1-dimensional images,
then n^d data points are required for d-dimensional images.

More data points make interesting structures emerge

The curse of dimensionality

A dataset of natural images will be extremely sparse in $\mathbb{R}^{w \times h}$, since each region of space is observed very infrequently.

As a consequence, all images will approximately be equally spaced

$\Rightarrow$ no meaningful structure will emerge from the dataset.

We would need exponentially many observations as we have dimensions!

If n data points cover well the space of 1-dimensional images,
then n^d data points are required for d-dimensional images.

Two options:

1. **Increase** the dataset
2. **Decrease** the dimensions

Features

Assume each data point $x \in \mathcal{D} \subset \mathbb{R}^n$ is the result of a synthesis process:

$$\sigma : F \mapsto x$$

which takes a set of features F and composes them to form x.

Features

Assume each data point $x \in \mathcal{D} \subset \mathbb{R}^n$ is the result of a synthesis process:

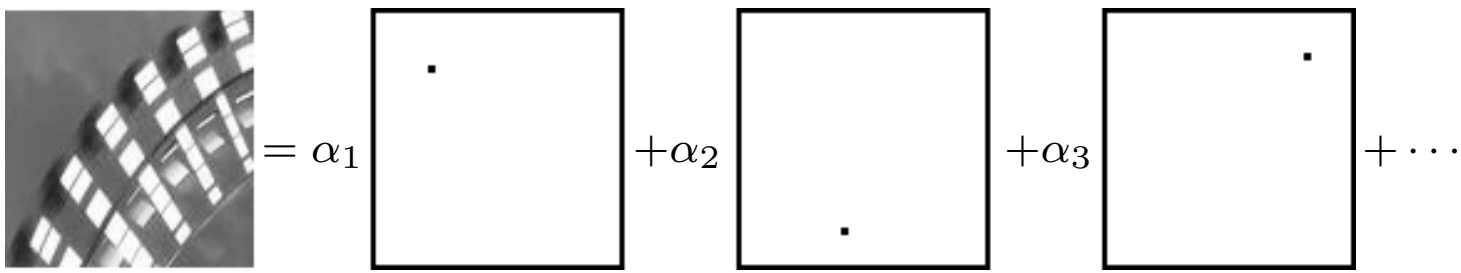
$$\sigma : F \mapsto x$$

which takes a set of features F and composes them to form x.

Example

An image $x \in \mathbb{R}^{w \times h}$ is composed by pixels.

If each pixel of x is a feature, then σ simply sums them up:



Features

Assume each data point $x \in \mathcal{D} \subset \mathbb{R}^n$ is the result of a synthesis process:

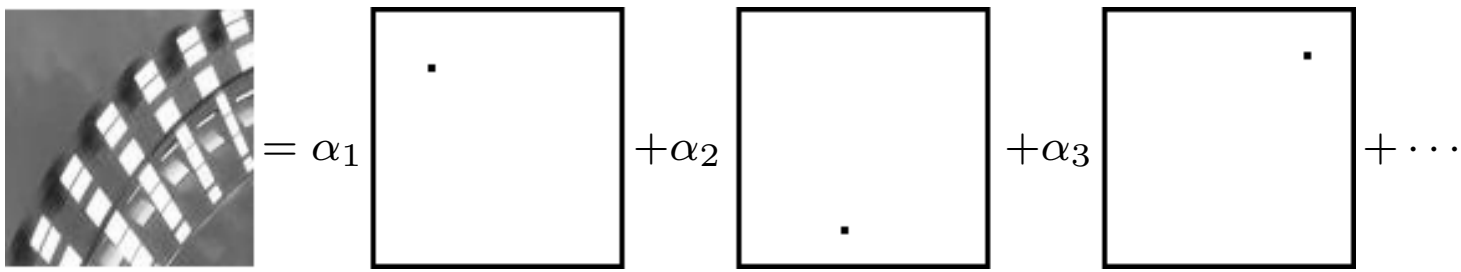
$$\sigma : F \mapsto x$$

which takes a set of features F and composes them to form x.

Example

An image $x \in \mathbb{R}^{w \times h}$ is composed by pixels.

If each pixel of x is a feature, then σ simply sums them up:



In this case, the feature space F is spanned by individual pixels.

Each feature (each pixel) represents a dimension.

Features

Assume each data point $x \in \mathcal{D} \subset \mathbb{R}^n$ is the result of a synthesis process:

$$\sigma : F \mapsto x$$

which takes a set of features F and composes them to form x.

Example

An image $x \in \mathbb{R}^{w \times h}$ is composed by pixels.
If each pixel of x is a feature, then σ simply sums them up:

$$x = \sigma(F) = \sum_{f_i \in F} \alpha_i \cdot f_i$$

α_i are the weights in the representation of x.

Features

Assume each data point $x \in \mathcal{D} \subset \mathbb{R}^n$ is the result of a synthesis process:

$$\sigma : F \mapsto x$$

which takes a set of features F and composes them to form x.

Example

An image $x \in \mathbb{R}^{w \times h}$ is composed by pixels.
If each pixel of x is a feature, then σ simply sums them up:

$$x = \sigma(F) = \sum_{f_i \in F} \alpha_i \cdot f_i$$

α_i are the weights in the representation of x.

In this **particular case**, the feature space is a vector space and σ is linear.

Features

Having one feature per pixel is extremely wasteful!

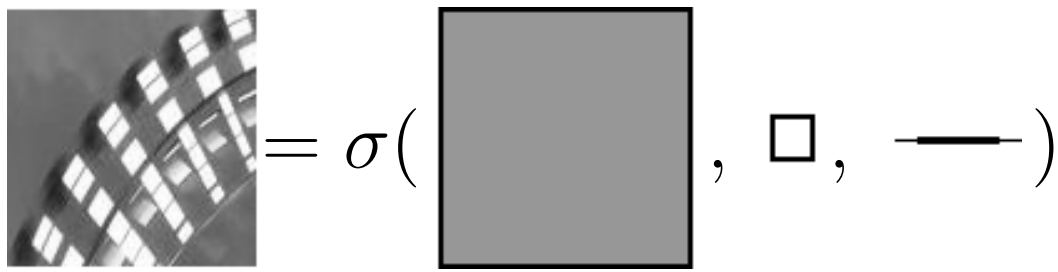
Curse of dimensionality: features $\gg$ observations

Features

Having one feature per pixel is extremely wasteful!

Curse of dimensionality: features $\gg$ observations

What does really characterize our image?



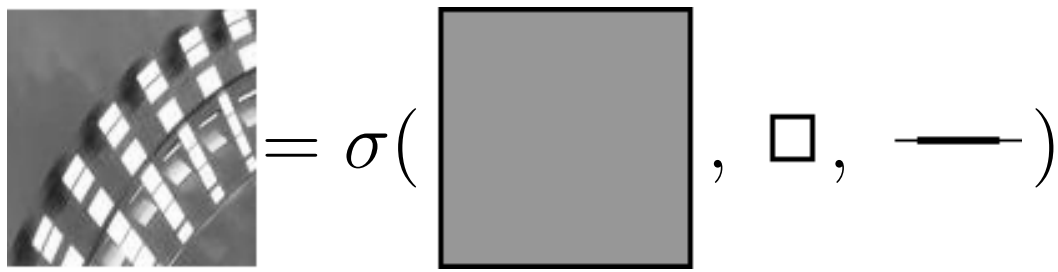
In general, the transformation σ acts nonlinearly on the features.

Features

Having one feature per pixel is extremely wasteful!

Curse of dimensionality: features $\gg$ observations

What does really characterize our image?



In general, the transformation σ acts nonlinearly on the features.

The output of σ is called an embedding of the data point.
For the data point $x \in \mathcal{D} \subset \mathbb{R}^n$, the embedding space is $\mathbb{R}^n$.

Intrinsic invariances

In general, a given data point admits many possible embeddings.

Intrinsic invariances

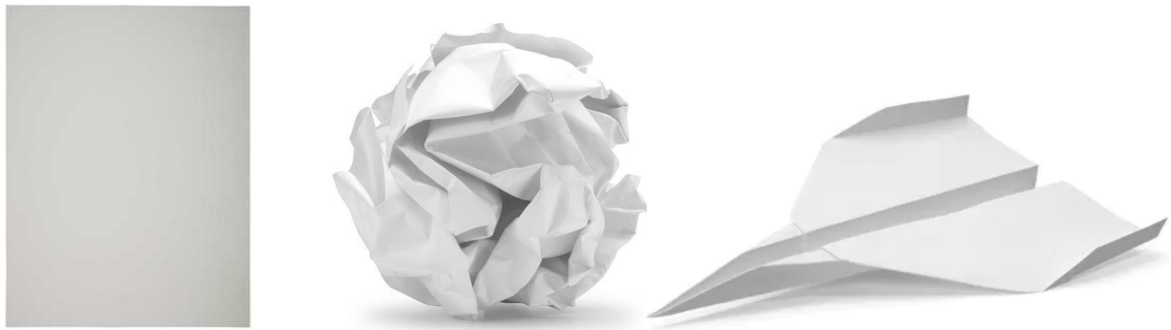
In general, a given data point admits many possible embeddings.

Example: A sheet lives naturally in $\mathbb{R}^2$

Intrinsic invariances

In general, a given data point admits many possible embeddings.

Example: A sheet lives naturally in $\mathbb{R}^2$, but is usually embedded in $\mathbb{R}^3$.

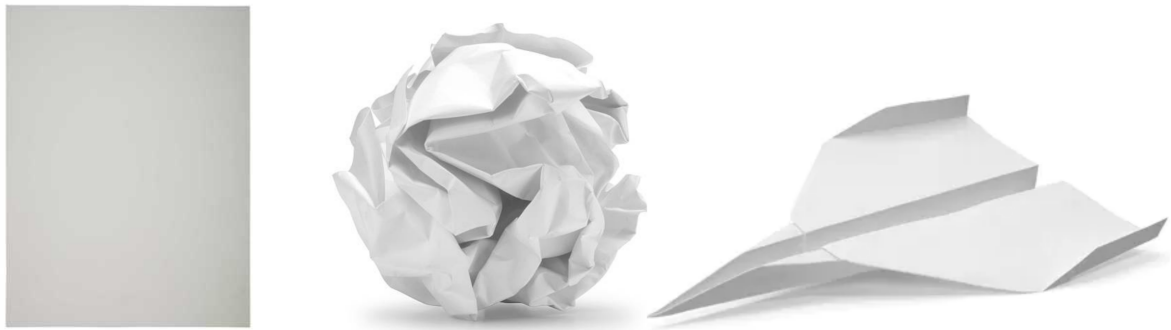


Three different embeddings of the same object

Intrinsic invariances

In general, a given data point admits many possible embeddings.

Example: A sheet lives naturally in $\mathbb{R}^2$, but is usually embedded in $\mathbb{R}^3$.



Three different embeddings of the same object

(distances are preserved in all the embeddings)

Intrinsic invariances

In general, a given data point admits many possible embeddings.

Example: A sheet lives naturally in $\mathbb{R}^2$, but is usually embedded in $\mathbb{R}^3$.

Three different embeddings of the same object
(distances are preserved in all the embeddings)

Challenge: discover what intrinsic properties are preserved; these properties characterize the data.

Latent features

In the general case:

- Features are not necessarily localized in space
- Features are not necessarily evident in the embedding

Latent features

In the general case:

- Features are not necessarily localized in space
- Features are not necessarily evident in the embedding

We talk about latent features. Direct access to the embedding only.

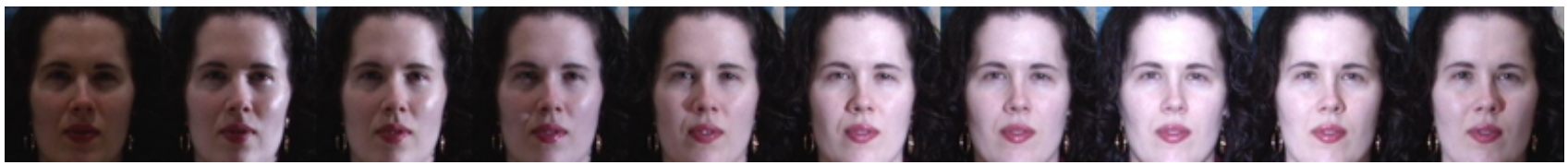
Latent features

In the general case:

- Features are not necessarily localized in space
- Features are not necessarily evident in the embedding

We talk about latent features. Direct access to the embedding only.

Example



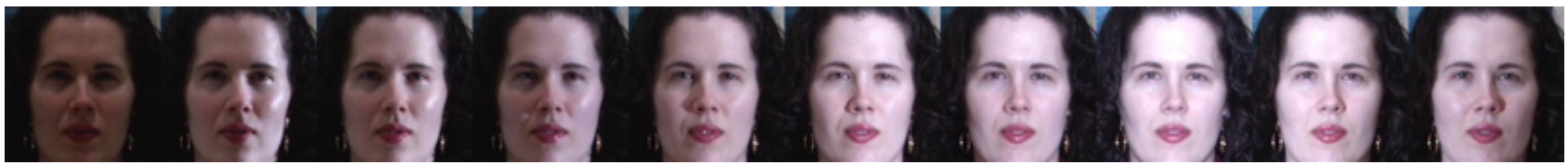
Latent features

In the general case:

- Features are not necessarily localized in space
- Features are not necessarily evident in the embedding

We talk about latent features. Direct access to the embedding only.

Example



Latent feature: directional illumination

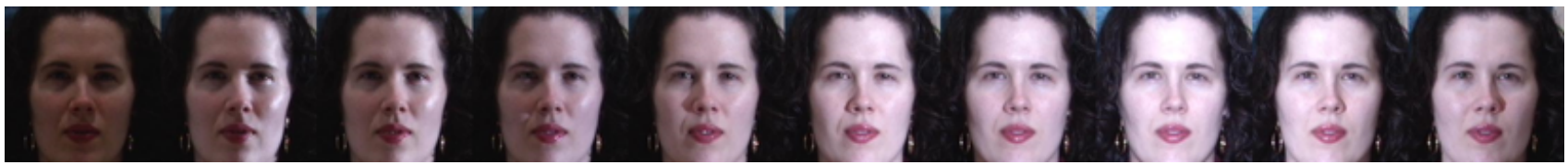
Latent features

In the general case:

- Features are not necessarily localized in space
- Features are not necessarily evident in the embedding

We talk about latent features. Direct access to the embedding only.

Example



Latent feature: directional illumination

3 params for the light source position + **1** param for light intensity

Latent features

In the general case:

- Features are not necessarily localized in space
- Features are not necessarily evident in the embedding

We talk about latent features. Direct access to the embedding only.

Discovering latent features involves discovering:

the "true" embedding space for the data
+
the transformation between the two spaces

We want to discard the non-informative dimensions from the data.

The manifold hypothesis

Deep learning assumes that the input data lives on some underlying non-Euclidean structure called a manifold.

The manifold hypothesis

Deep learning assumes that the input data lives on some underlying non-Euclidean structure called a manifold.

Features are task-driven

How are features extracted from given data?

Speaking about features only makes sense if we are given a task to solve!

Features are task-driven

How are features extracted from given data?

Speaking about features only makes sense if we are given a task to solve!

Is color important?

Features are task-driven

How are features extracted from given data?

Speaking about features only makes sense if we are given a task to solve!

Is color important?

Rank, suit, and color are generic features, but specific problems determine what features are important for that task.

What counts in spades, does not count in poker.

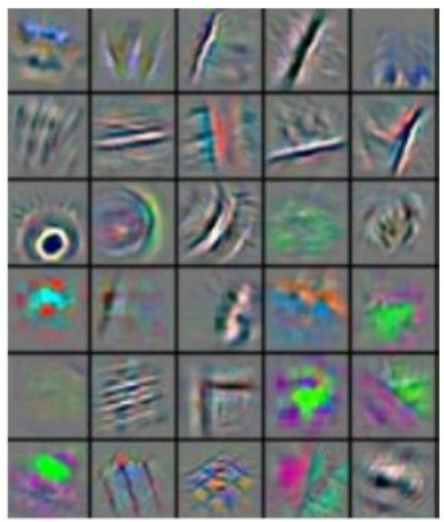
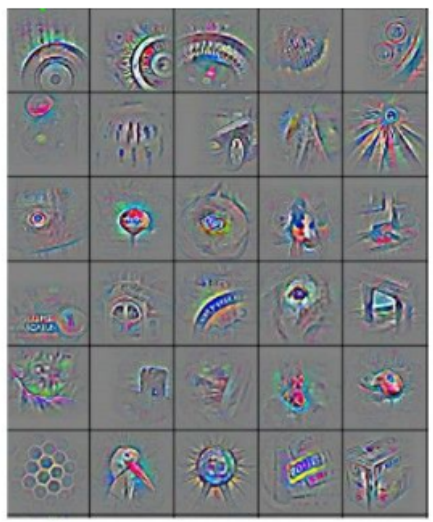
Deep learning is a task-driven paradigm to extract patterns and latent features from given observations

Deep learning is a task-driven paradigm to extract patterns and latent features from given observations

However, features are not always the focus of deep learning; rather, they are instrumental for the given task and drive the decision.

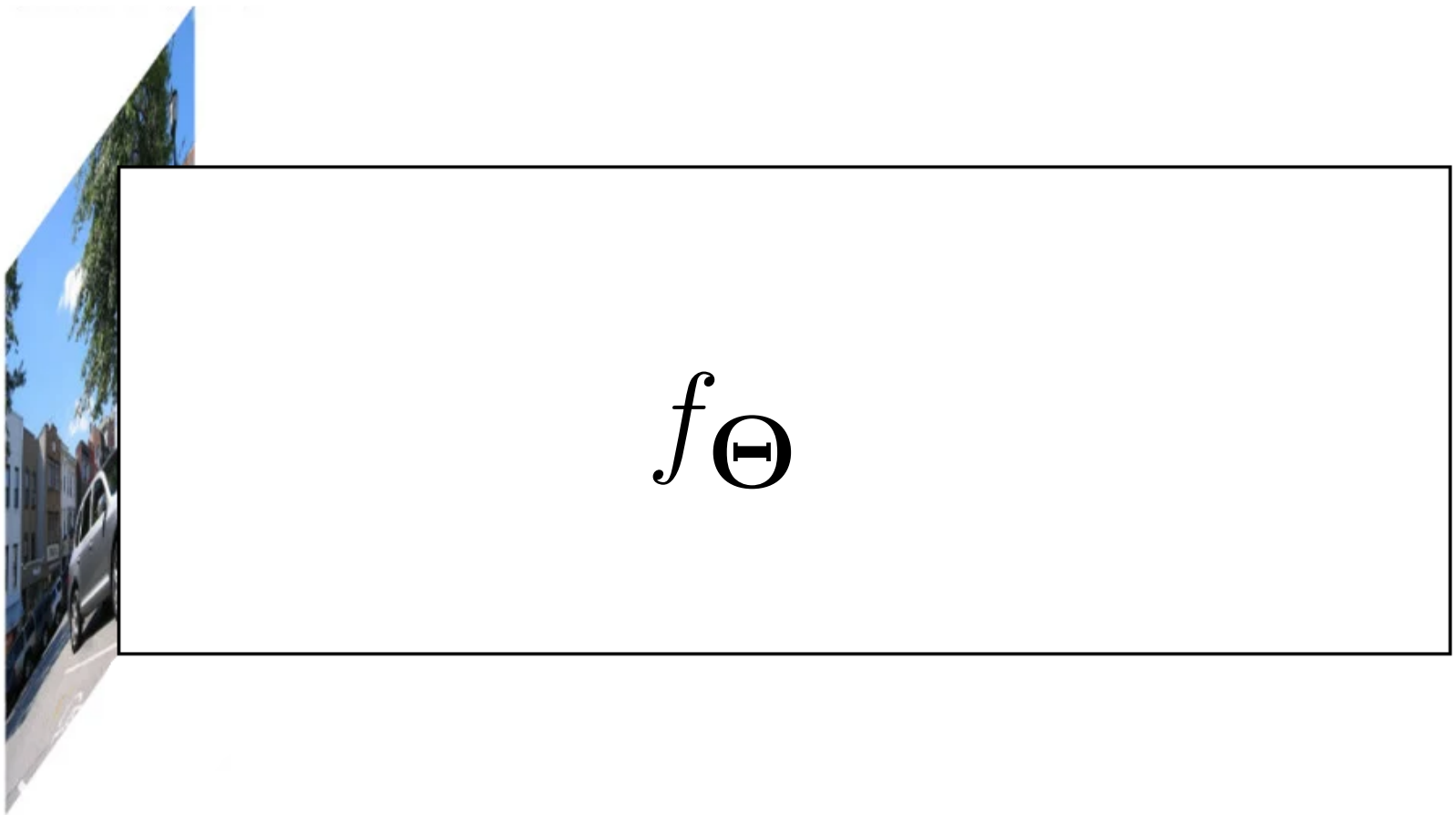
Example: Visual classification


→ low-level features → mid-level features → hi-level features → "car"

A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:



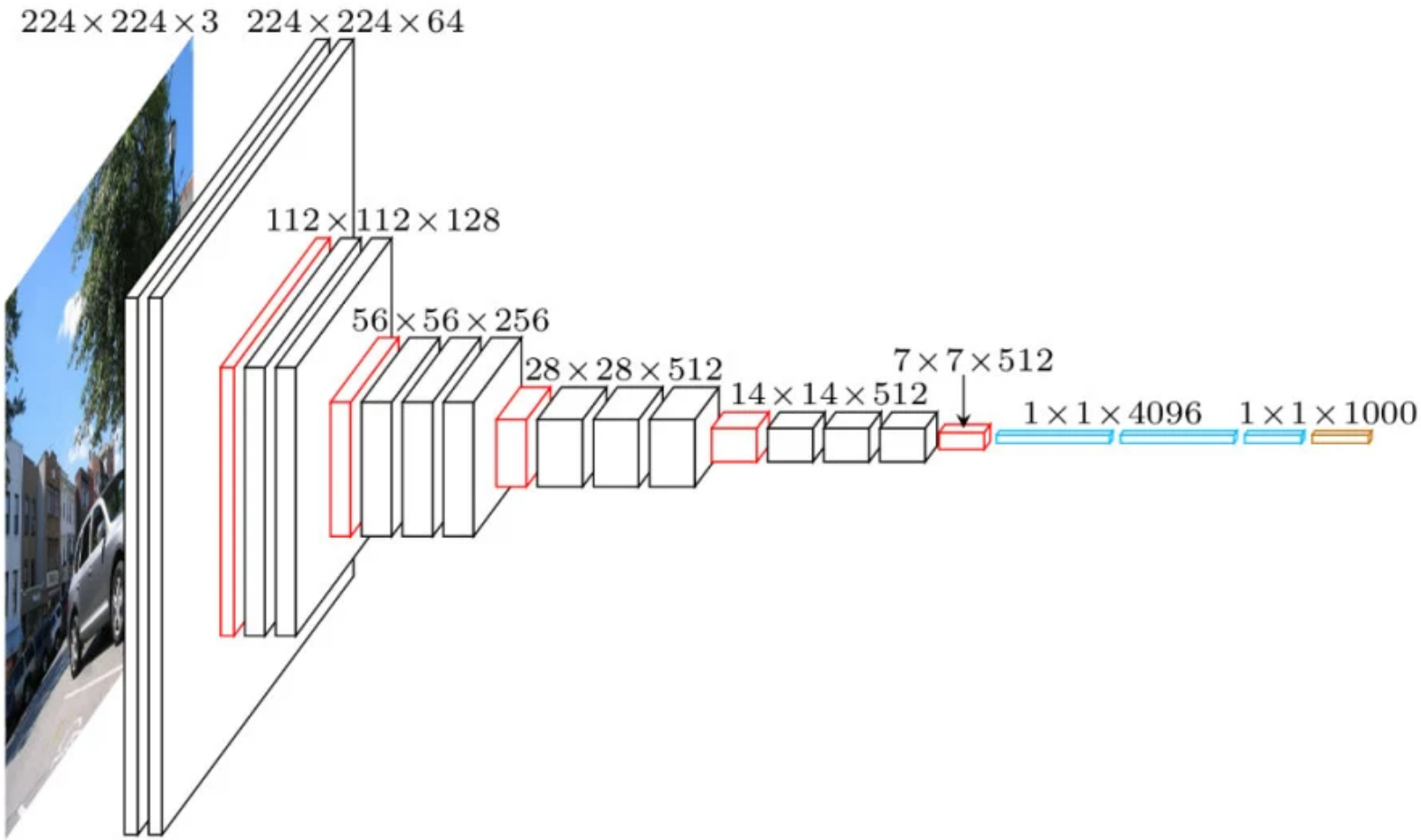
A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:

$$f_{\boldsymbol{\Theta}}(\mathbf{x}) = \mathbf{y}$$

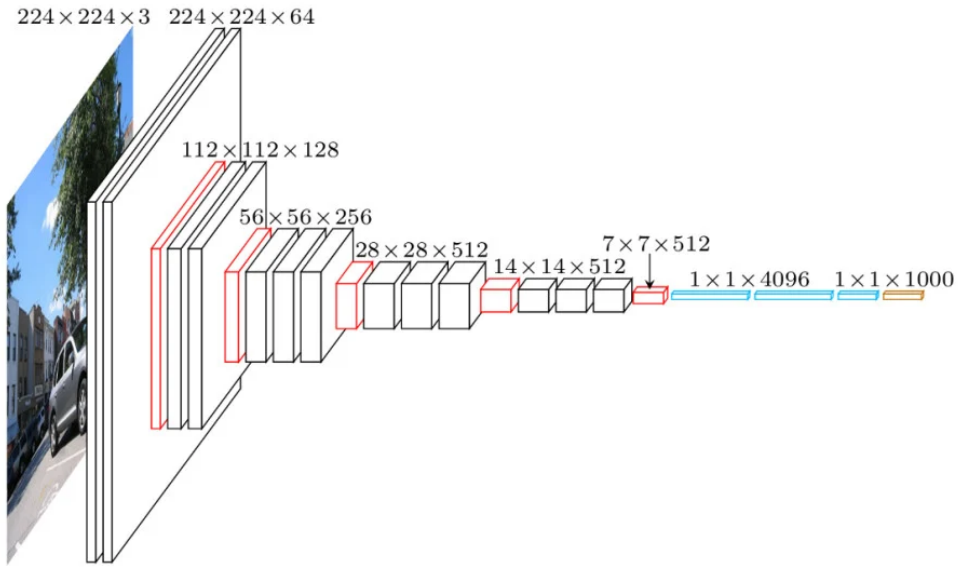
A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:



A glimpse into neural networks

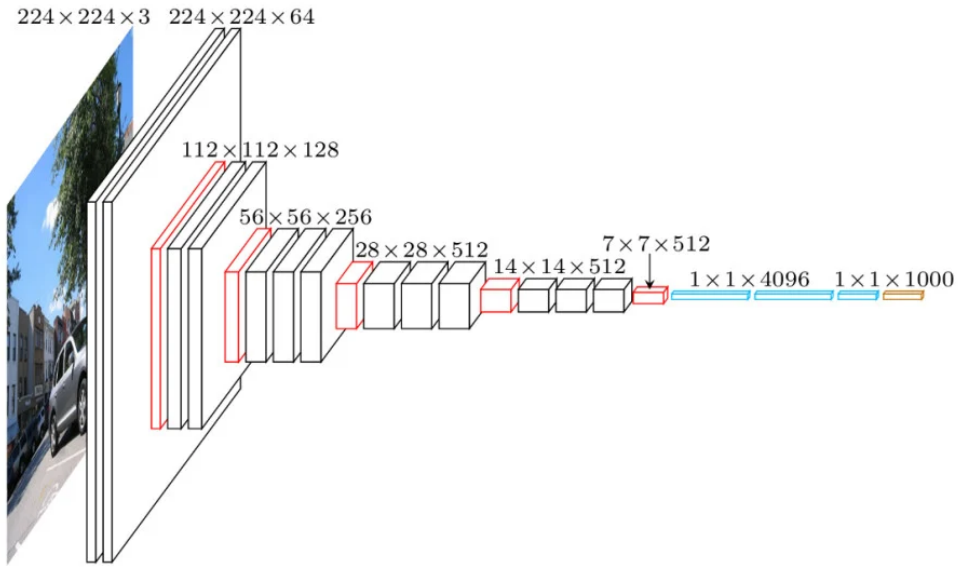
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)

A glimpse into neural networks

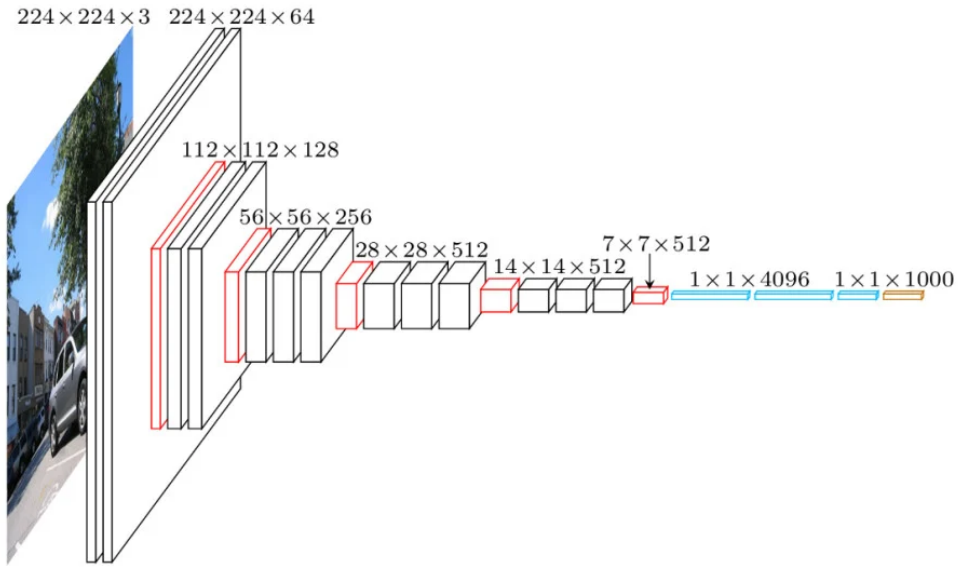
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ

A glimpse into neural networks

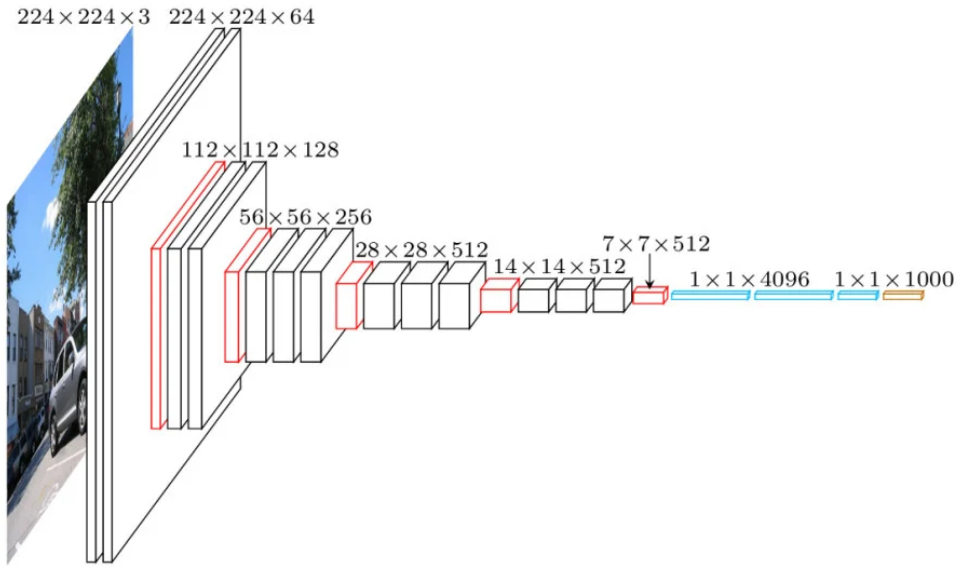
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ
- Finding the parameter values is called training...

A glimpse into neural networks

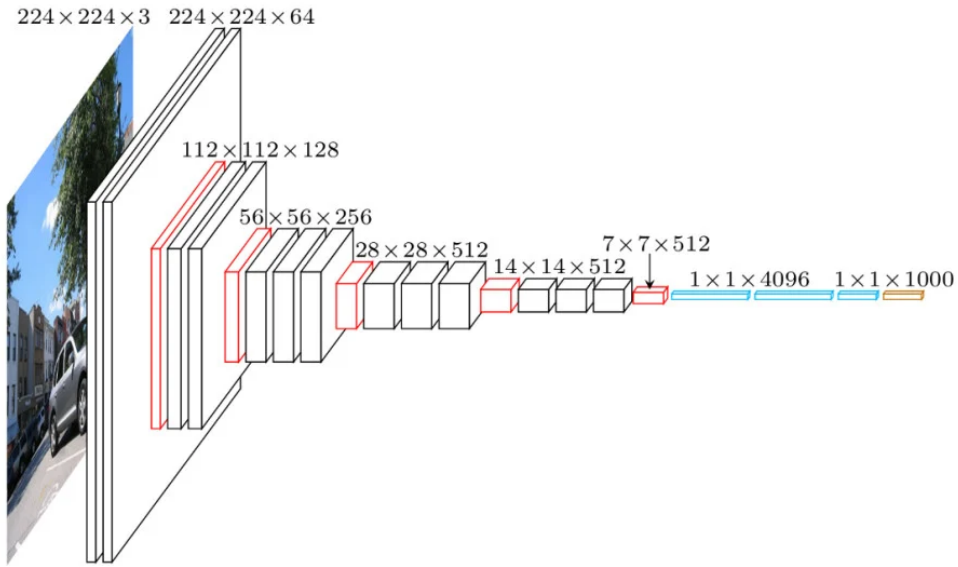
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ
- Finding the parameter values is called training...
- ...which is done by minimizing a function called loss

A glimpse into neural networks

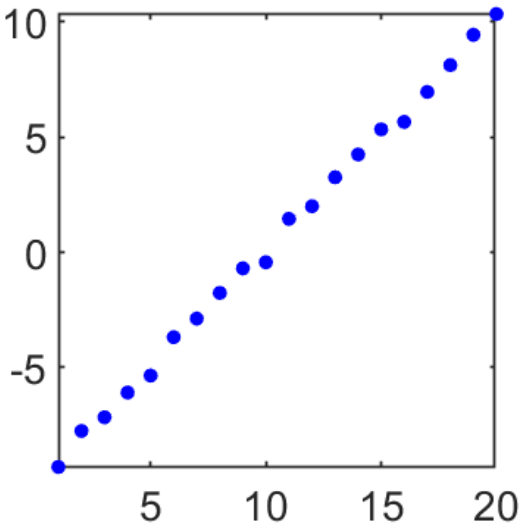
In deep learning, we deal with highly parametrized models called deep neural networks:



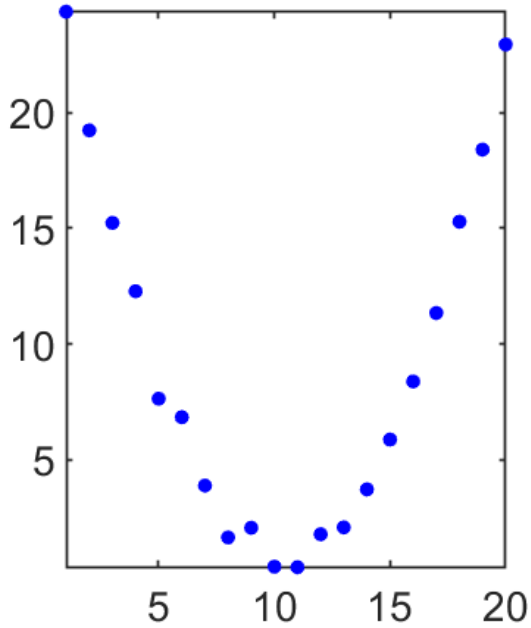
- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ
- Finding the parameter values is called training...
- ...which is done by minimizing a function called loss
- Minimization requires computing gradients, called backpropagation

Parametrized models

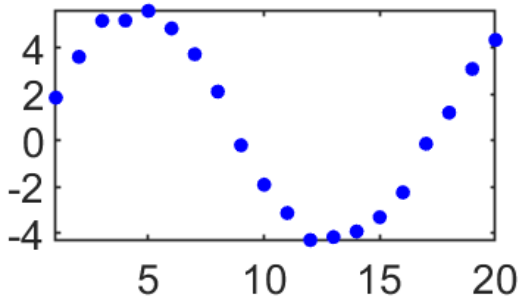
The parameters describe the behavior of the network, and must be solved for.



$y = ax + b$



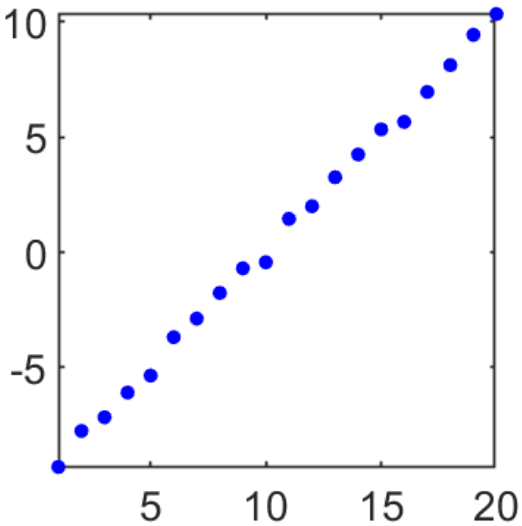
$y = ax^2 + bx + c$

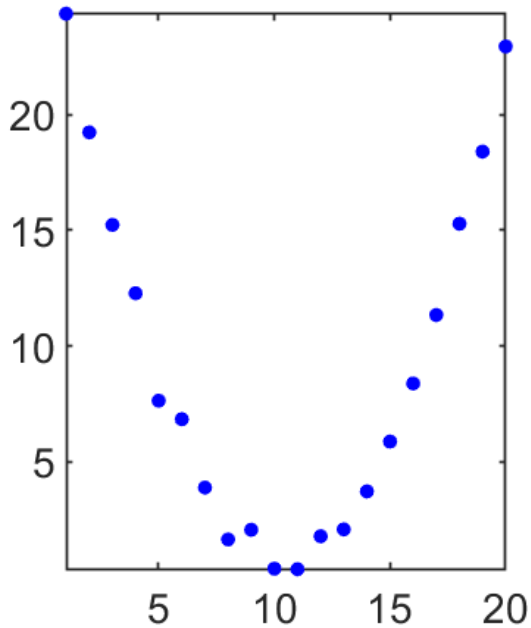


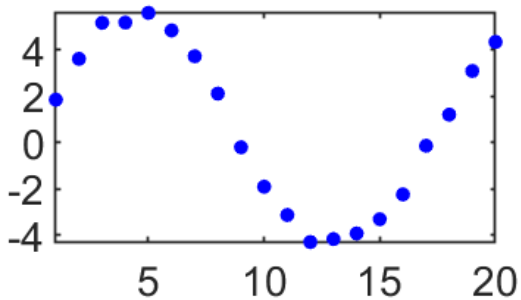
$y = a\sin(x) + bx + c$

Parametrized models

The parameters describe the behavior of the network, and must be solved for.



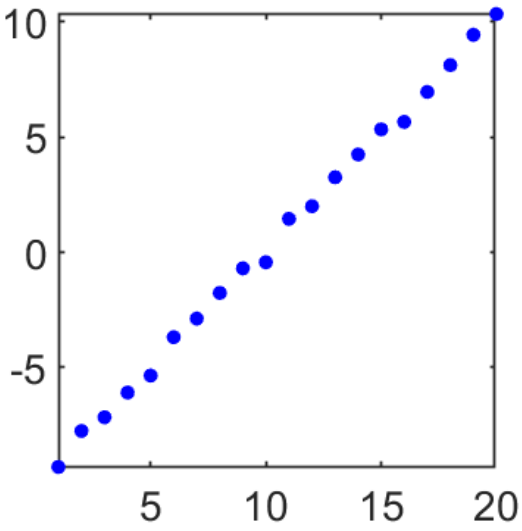




$$f_{a,b}(x) = ax + b$$

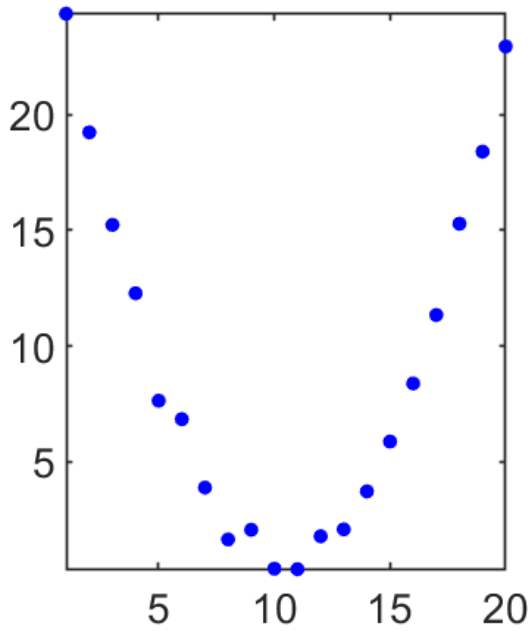
Parametrized models

The parameters describe the behavior of the network, and must be solved for.

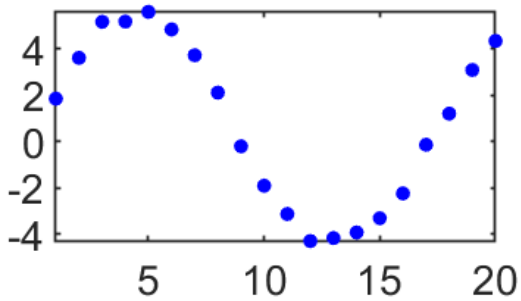


$f_{\Theta_f}(x)$

$\Theta_f \equiv \{a, b\}$



$g_{\Theta_g}(x)$

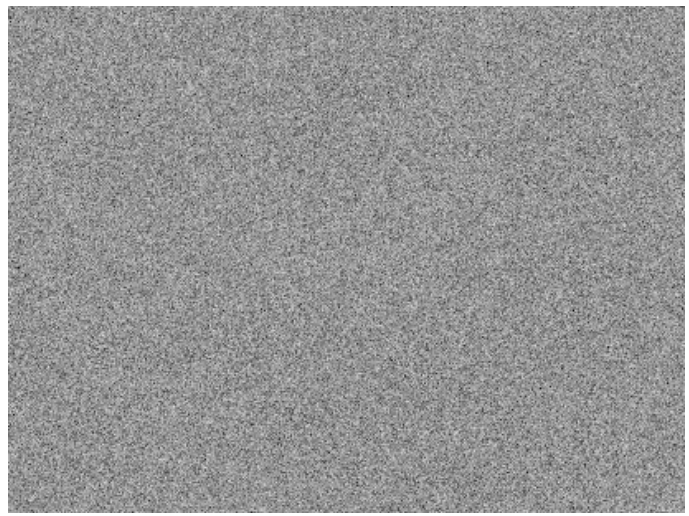


$h_{\Theta_h}(x)$

From a technical standpoint, our task is to determine the parameters Θ.

Structure as a strong prior

Key insight: Data often carries structural priors in terms of repeating patterns, compositionality, locality, ...

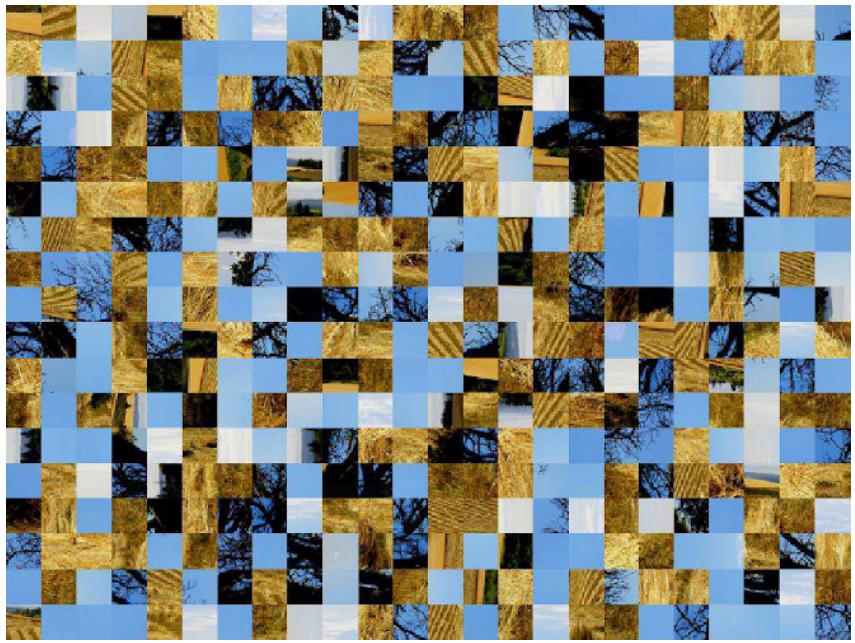


Structure as a strong prior

Key insight: Data often carries structural priors in terms of repeating patterns, compositionality, locality, ...

Structure as a strong prior

Key insight: Data often carries structural priors in terms of repeating patterns, compositionality, locality, ...



Son et al, "Solving Small-piece Jigsaw Puzzles by Growing Consensus", CVPR 2016

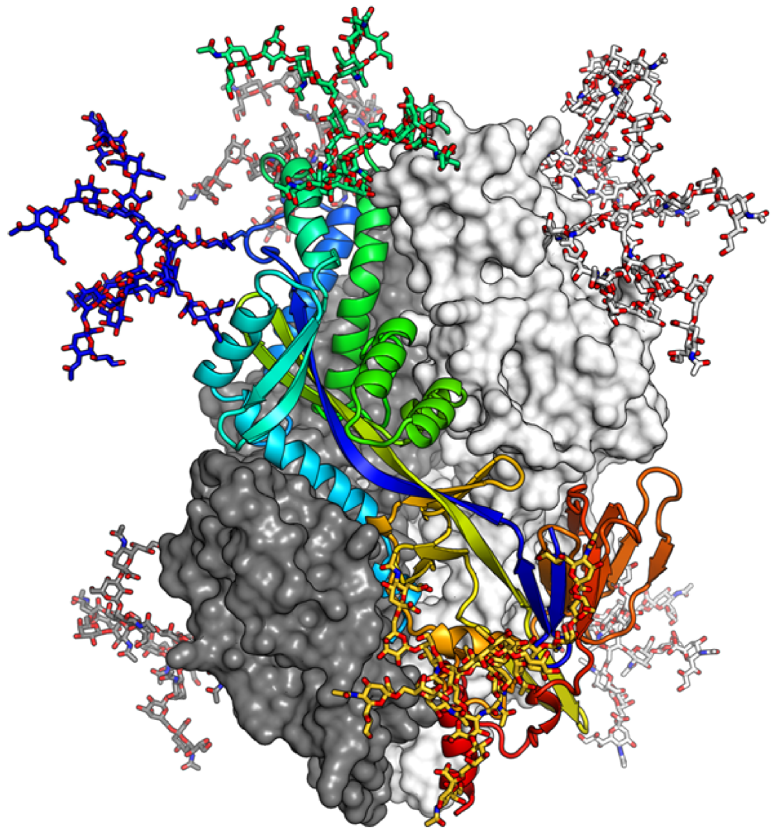
Structure as a strong prior

Key insight: Data often carries structural priors in terms of repeating patterns, compositionality, locality, ...

Son et al, "Solving Small-piece Jigsaw Puzzles by Growing Consensus", CVPR 2016

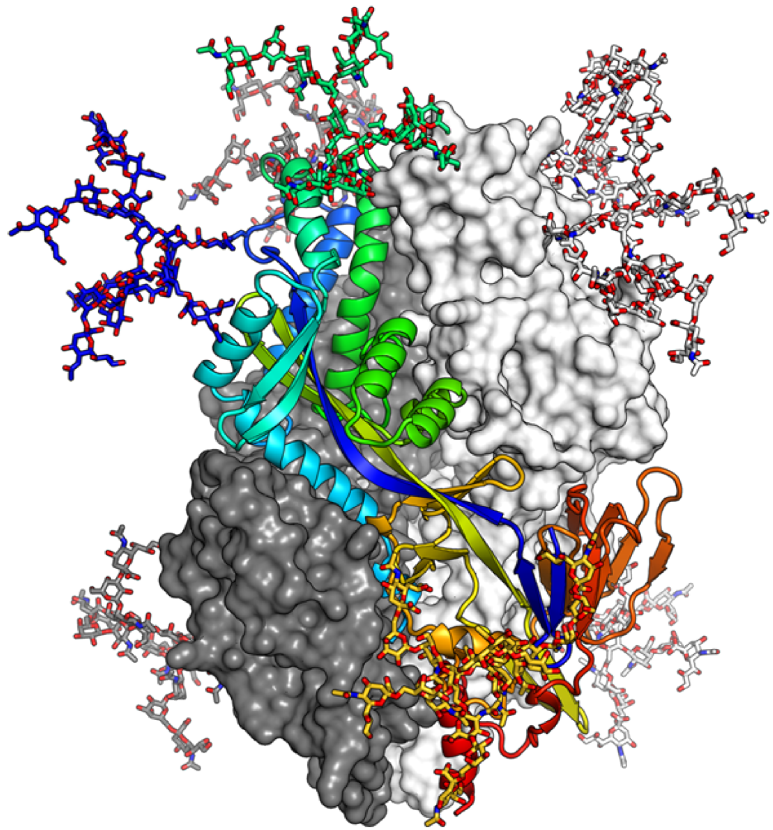
Structure as a strong prior

Key insight: Data often carries structural priors in terms of repeating patterns, compositionality, locality, ...



Structure as a strong prior

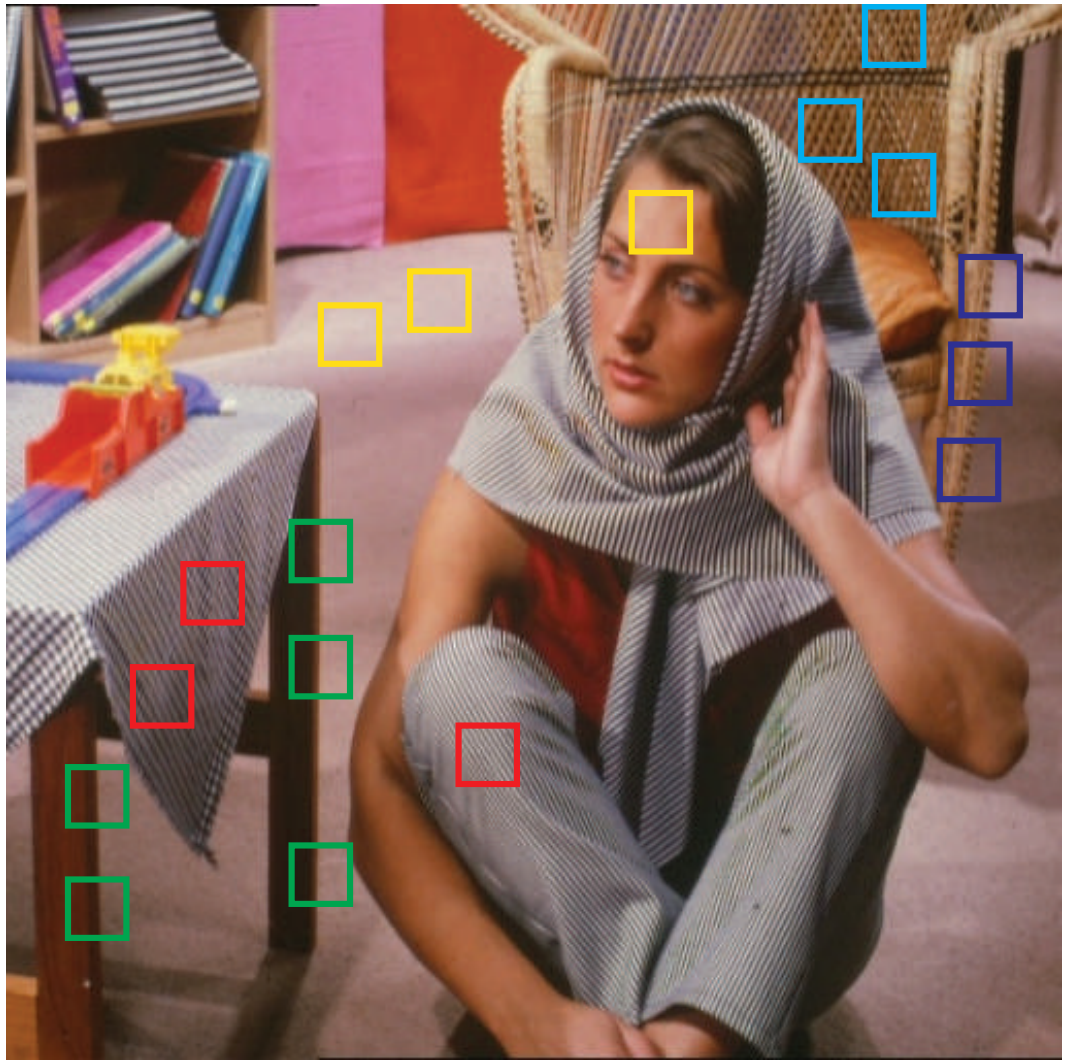
Key insight: Data often carries structural priors in terms of repeating patterns, compositionality, locality, ...



Take advantage of the structure of the data.

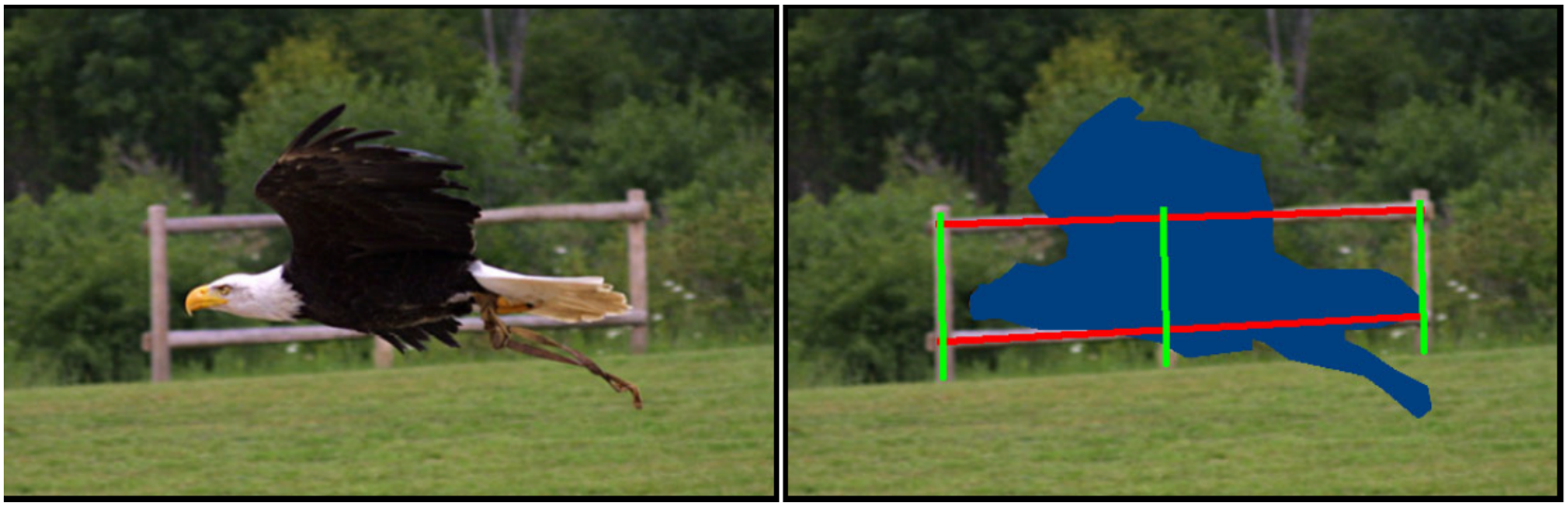
Self-similarity

Data tends to be self-similar across the domain:



Self-similarity

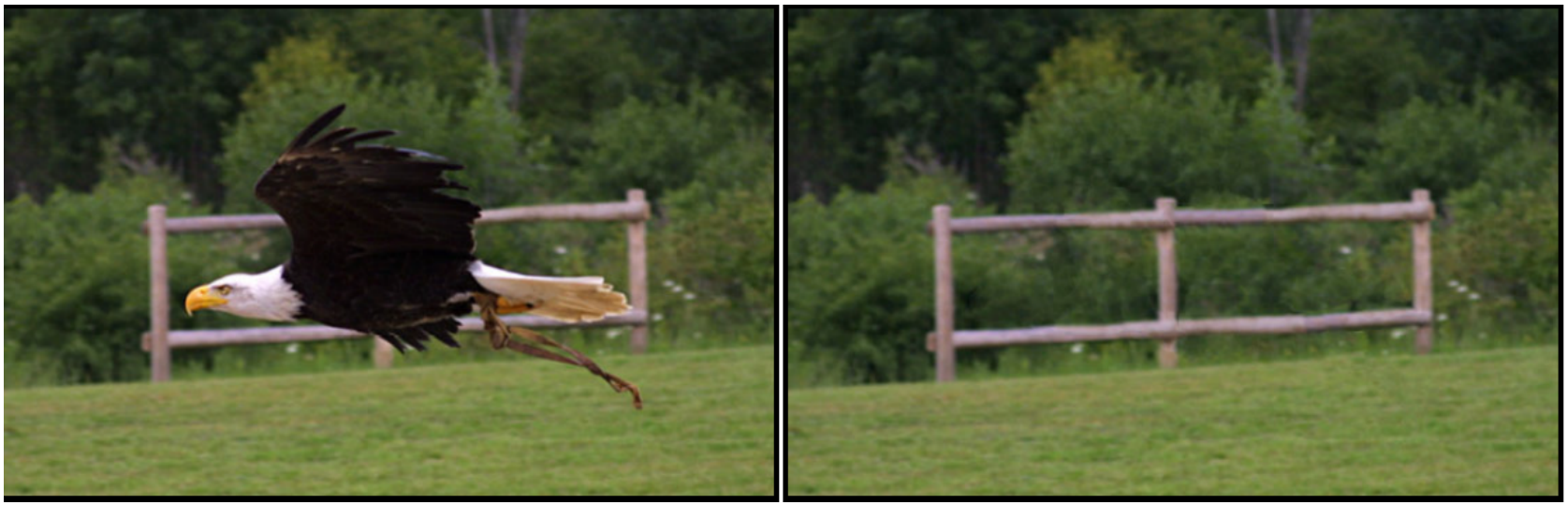
Data tends to be self-similar across the domain:



Barnes et al, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing", TOG 2009

Self-similarity

Data tends to be self-similar across the domain:



Barnes et al, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing", TOG 2009

Convolutional neural networks (CNN)

Data is often composed of hierarchical, local, shift-invariant patterns.

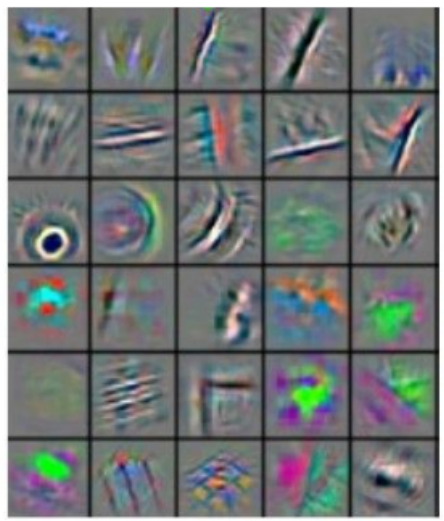
Convolutional neural networks (CNN)

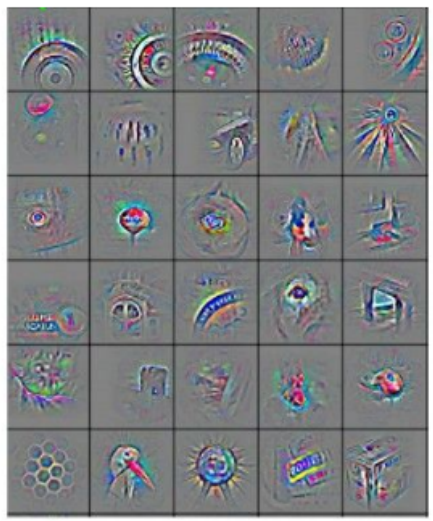
Data is often composed of hierarchical, local, shift-invariant patterns.

CNNs directly exploit this fact as a prior.

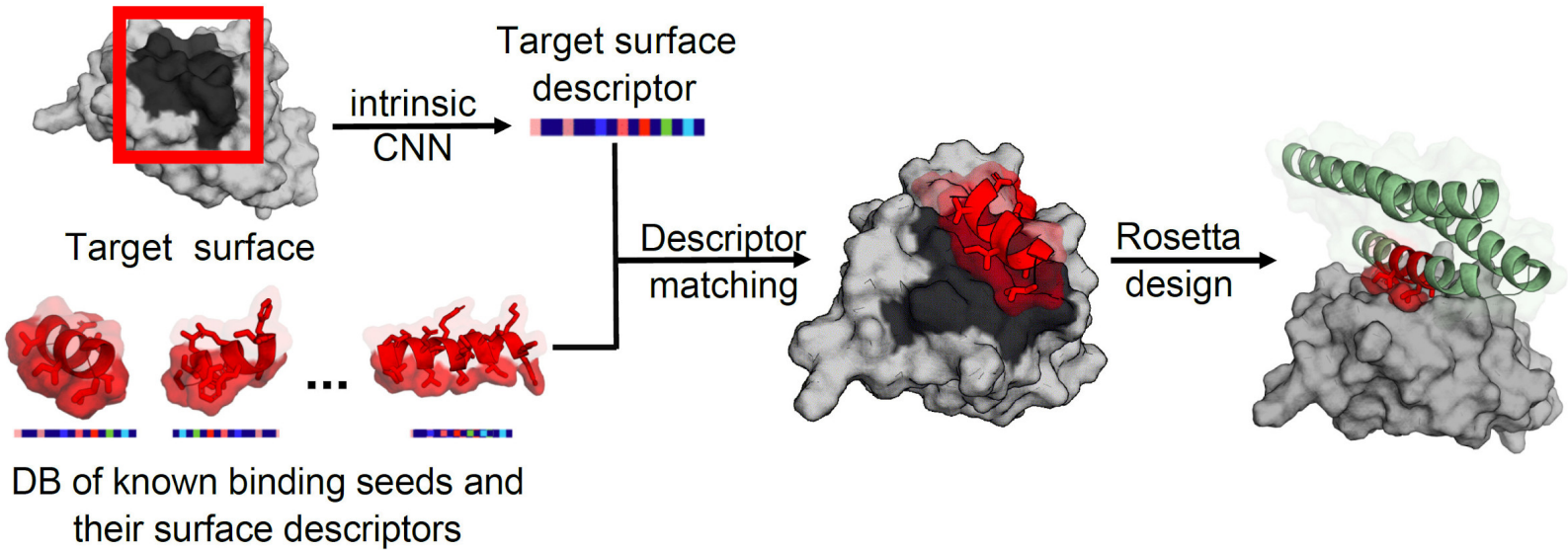
Learned features

→ low-level features → mid-level features → hi-level features → “car”





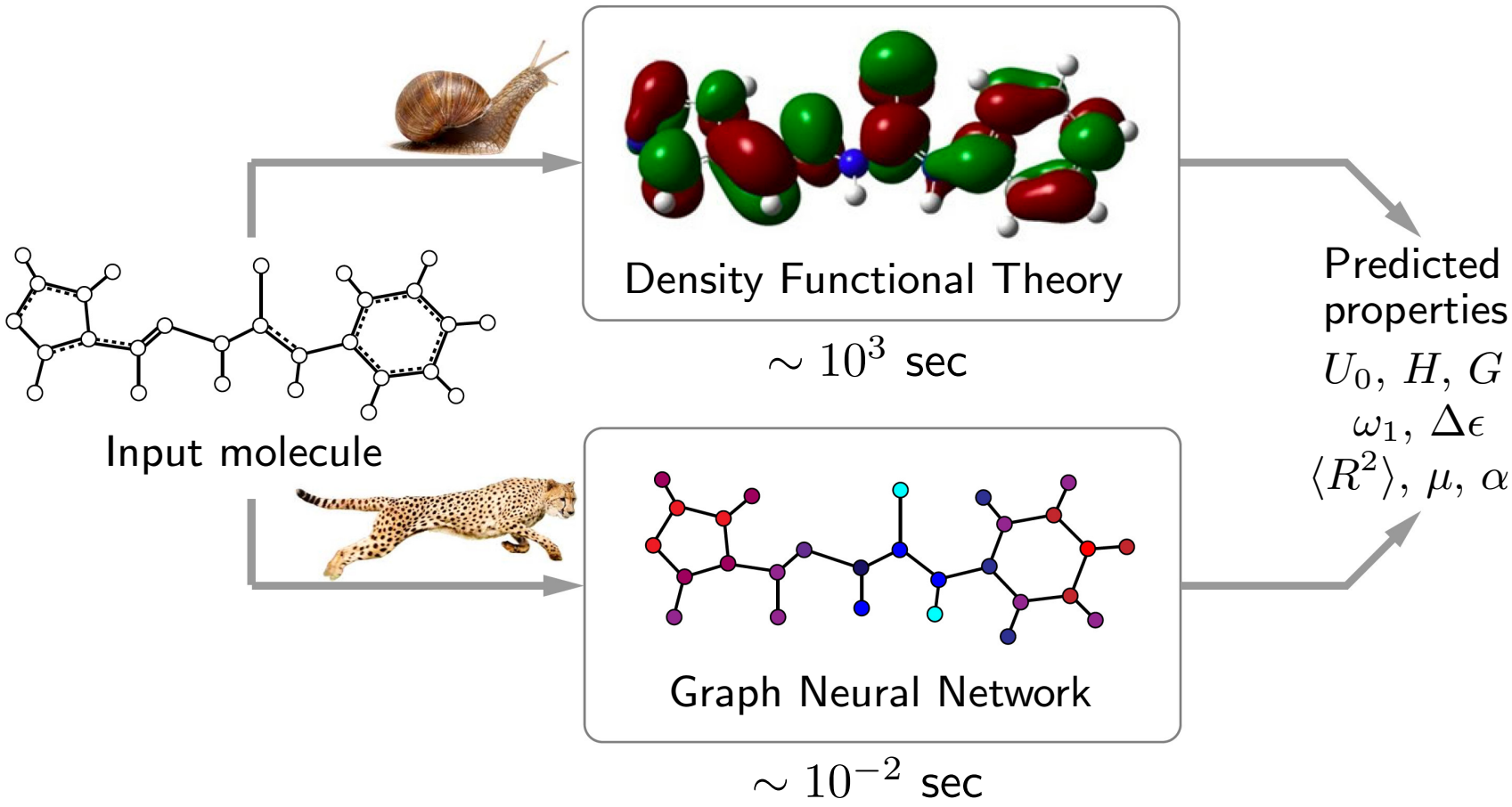
Application: Protein-Protein Interaction



Designing protein binder for the PD-L1 protein target

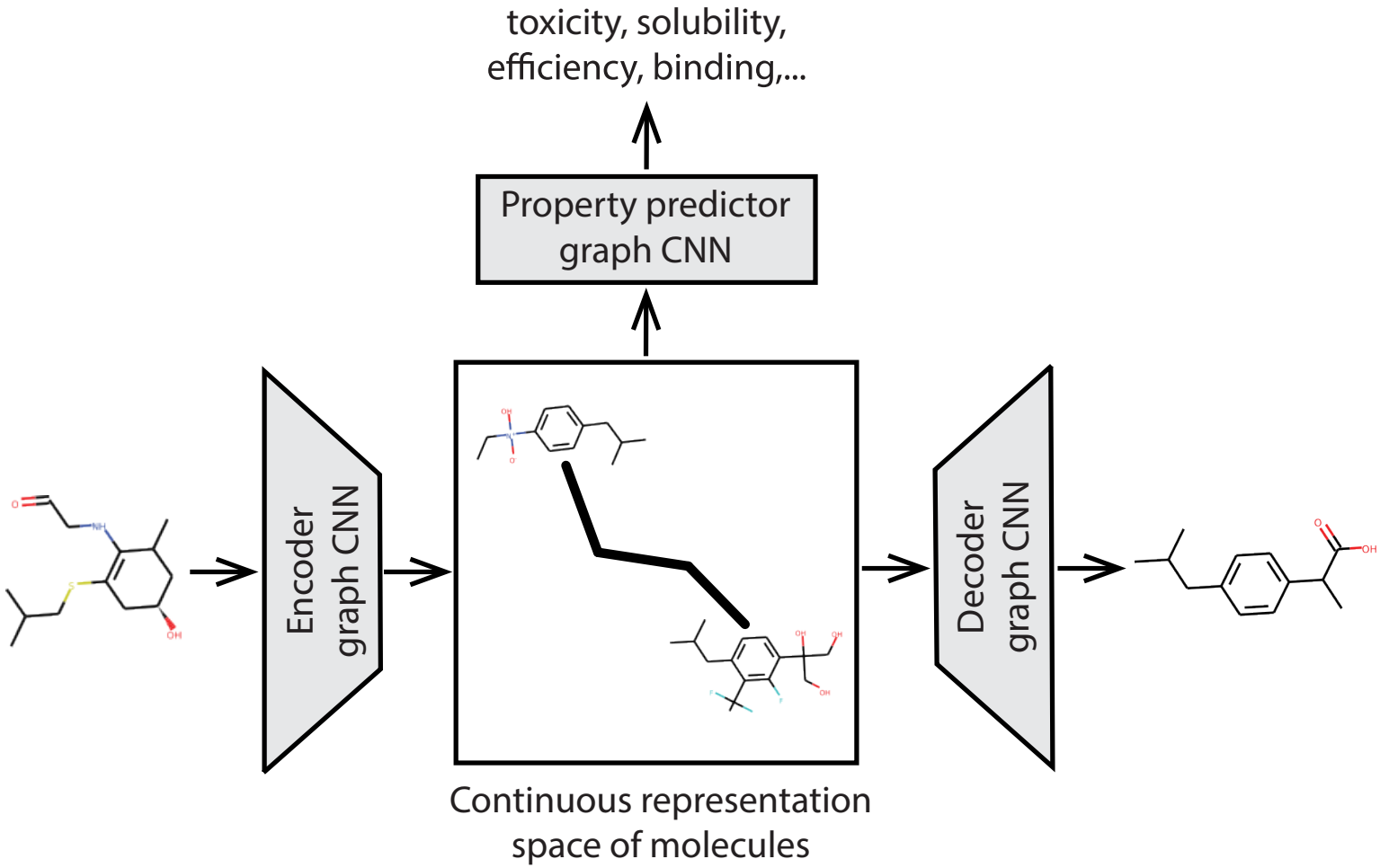
Gainza et al, "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning", Nature Methods 2020

Molecule property prediction

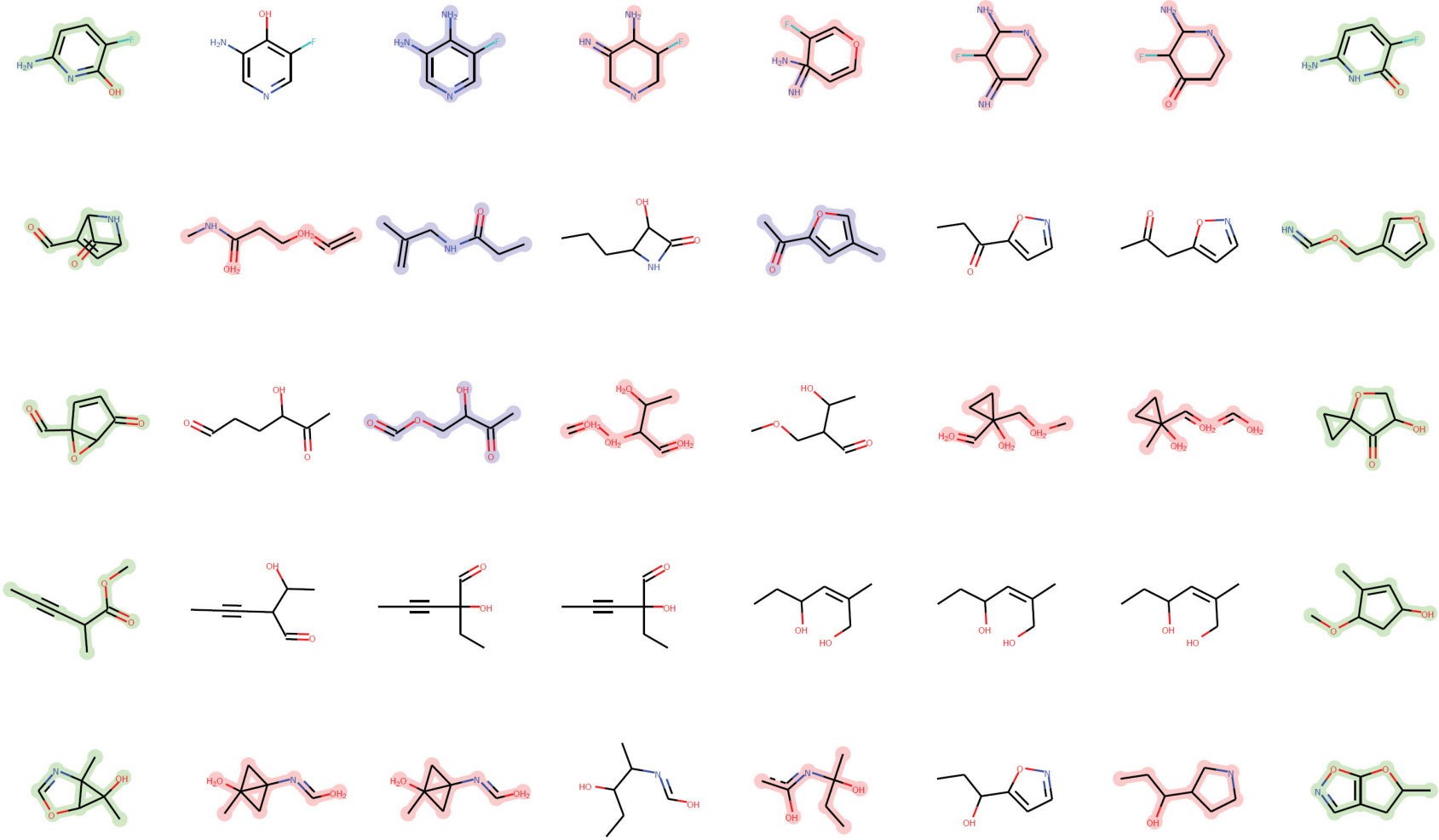


Duvenaud et al, "Convolutional Networks on Graphs for Learning Molecular Fingerprints", NIPS 2015; Gomez-Bombarelli et al, "Automatic chemical design using a data-driven continuous representation of molecules", ACS Cent. Sci. 2018

Generative models



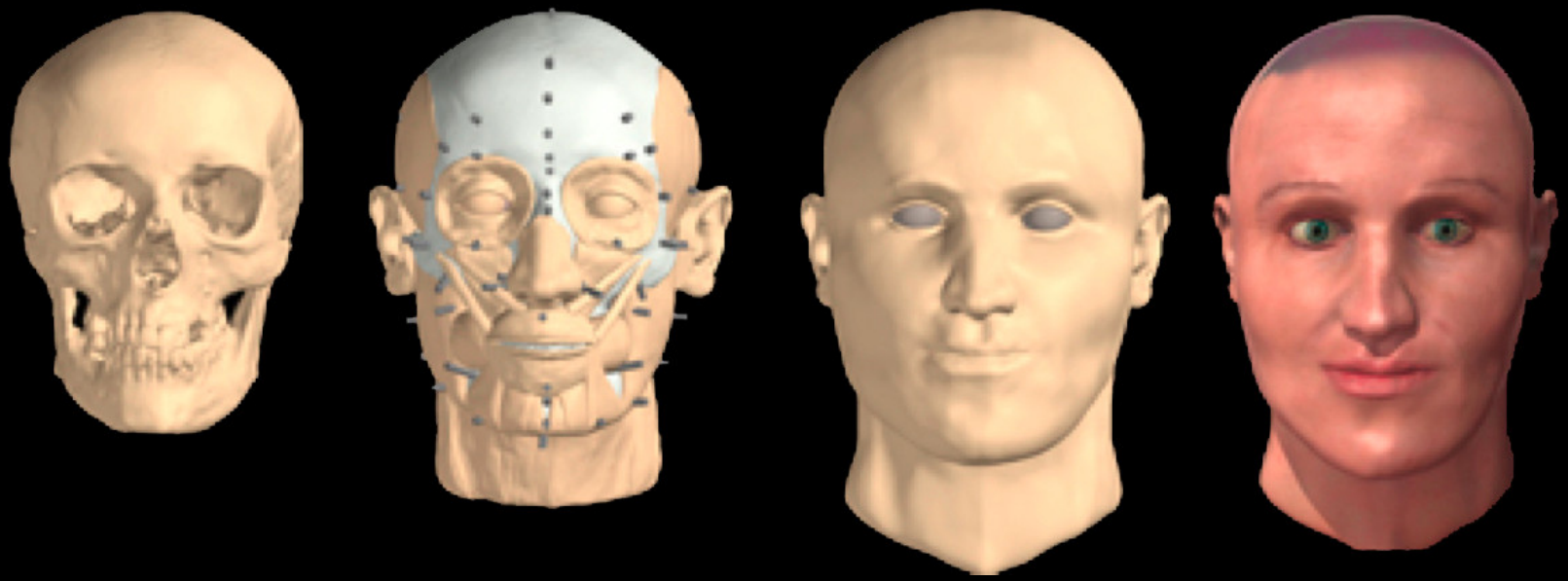
Molecule generation



Molecules generated with a graph VAE

Simonovsky and Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders", 2017; De Cao and Kipf, "MolGAN: An implicit generative model for small molecular graphs", 2018

Face from DNA



Claes et al, “Facial recognition from DNA using face-to-DNA classifiers”, Nature Communications 2019

Suggested reading

If you are curious, my course on Deep Learning (Computer Science):

`https://erodola.github.io/DLAI-s2-2021/`

`https://www.youtube.com/playlist?list=PL53MAHEGhXqLNzg-8ue8li2SU6ORE2rCl`