TÉCNICO+
FORMAÇÃO AVANÇADA

Representation Learning

Representations of Complex Data Structures using Deep Learning

DASH: Data Science e Análise Não Supervisionada

Rui Henriques, rmch@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa

Outline

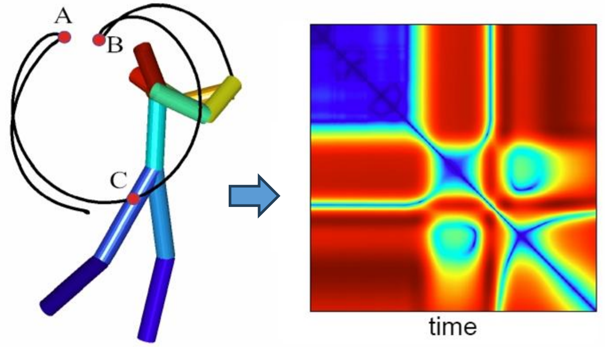
- Data Representations
- Neural Networks
- Unsupervised and Self-supervised Stances
- Handling Complex Data Structures
- Multimodal Data Representations

Representation Learning

- **Good features** essential for successful descriptive and predictive tasks
  - *feature extraction* and *engineering* estimated to entail 90% of the overall ML effort
  - principle: instead of learning $f : X \rightarrow Y$, learn a *representation* $g : X \rightarrow Z$, then learn the predictor/descriptor $f : Z \rightarrow Y$ on top of it

- "*A good representation is one that makes a subsequent learning task easier*"

  — *Deep Learning*, Goodfellow et al. 2016



Representation Learning

However… feature extraction is now always trivial

- *manual* extraction is laborious, often incomplete and prone to errors
  (e.g., features derived from EEG visual inspection)
- *automated* extraction pipelines not always good to handle complex data
  - statistics on a sliding window for *time series*, structure–texture–color cues for *images*, term frequencies for *text*, geometric features for *trajectories*…
  - … are often uninformative and susceptible to diverse idiosyncrasies

TÉCNICO+
FORMAÇÃO AVANÇADA

Representation Learning

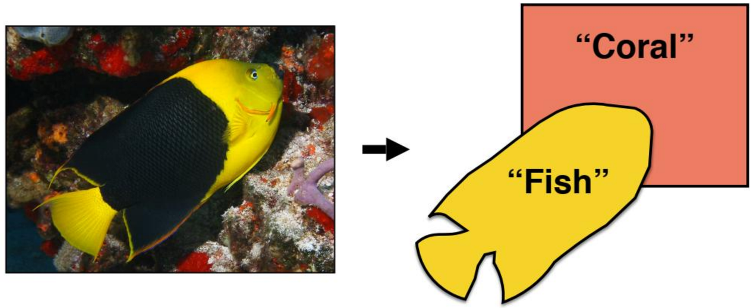
- Core **premises**
    - observed data is a causal result of **underlying factors**
    - **variation** in these factors explain the variation in the data
- *Example*: objects, rotation, lightning, etc. are underlying factors of images

- **Why** learning representations?
    - feature extraction from **complex data structures** (e.g., text, time series, image, events)
    - less dependence on feature engineering and domain knowledge
    - easy application of machine learning (**controlled dimensionality**, feature **independence**)
    - improved **descriptive** and **predictive capacity**
    - others: **denoising**, **summarization**, **visualization**…

Good representations

What makes a good representation? Good representations are:

1. compact (*minimal*)
2. explanatory (*sufficient*)
3. disentangled (*independent factors*)
4. interpretable
5. informative *(make subsequent problem solving easier)*



by Isola, Freeman, Torralba

Good representations

- **Smoothness:** if $x1 \approx x2$, then $g(x1) \approx g(x2)$
  - the most basic prior
  - similarity in the representation (somehow) reflects similarity of raw data
- **Invariances/equivariance/coherence:**
  small semantic changes should result in similar representations. Invariance is domain-specific:
  - image representations should be invariant under transformations like rotations, color jitter…
  - time series representations can be invariant to temporal shifts, rescales…
- **Less supervision:** unsupervised / semi-supervised / self-supervised representations
  - representations should capture raw content (easily lost when focusing on specific targets)
  - one representation may be used for multiple end goals

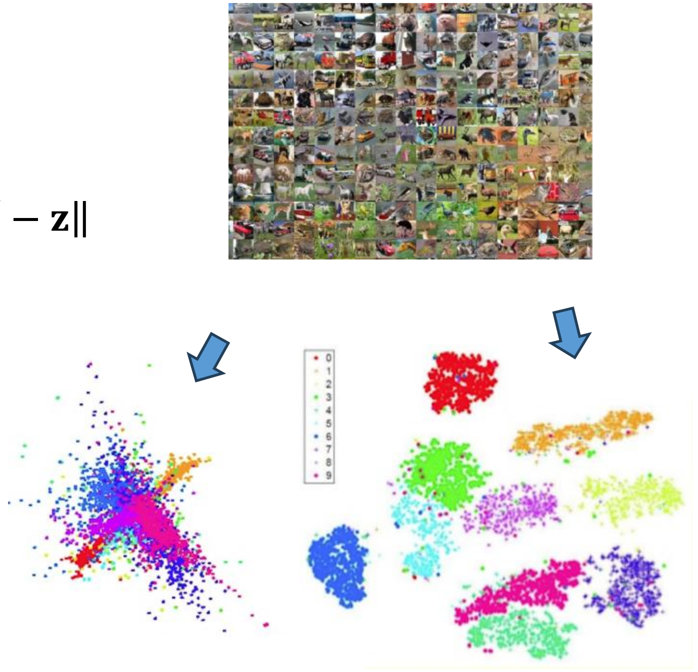
Good representations

- **Multiple explanatory factors**
  - recover different factors so it is useful for many tasks
- **Disentangled explanatory factors**
  - each dimension of the representation should capture a separate/meaningful aspect of the data
- **Hierarchical explanatory factors**
  - some underlying factors are more abstract than others
    and could be further defined in terms of less abstract ones
- **Loose factor dependencies**
  - factors should be related through simple, linear dependencies
  - leverage interpretability (e.g., physics) and support subsequent learning
- **Sparsity:** for any observation x, only some factors are
  - relevant $\Rightarrow$ most dimensions of $g(x)$ should be zero, or
  - invariant to small variations of x

Good representations

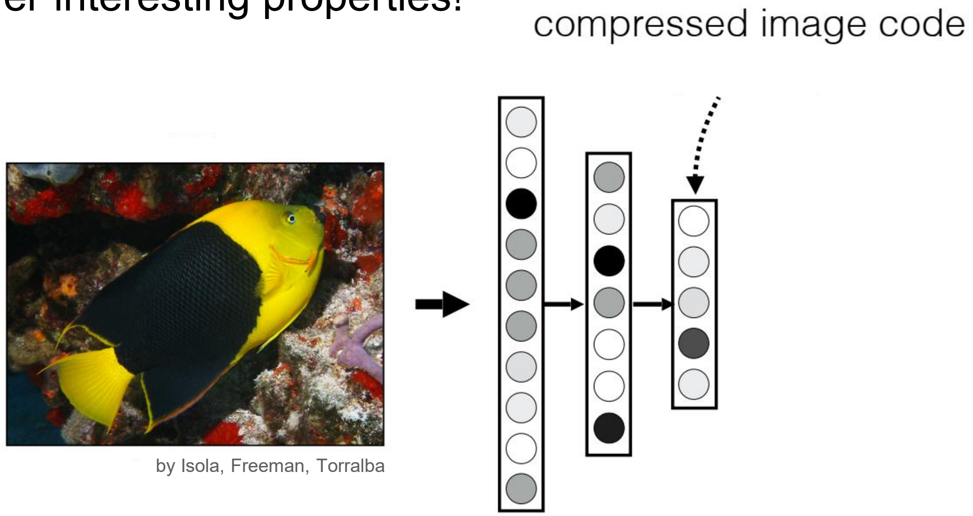
These properties can be easily tested

- smoothness: reconstruction capacity
  - given $\mathbf{z} = g(\mathbf{x})$, recover $\hat{\mathbf{x}} = g1(\mathbf{z})$ and assess $||\hat{\mathbf{x}} - \mathbf{x}||$
- invariance: inject non-relevant changes, $\mathbf{x}'$, and assess $||\mathbf{z}' - \mathbf{z}||$
- disentanglement: low dependencies in Z (recall descriptive statistics from previous class)
- sparsity: average #zeros per representation, $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_n\}$
- learning utility: predictive scores (accuracy, F1, errors…) and descriptive scores (clustering quality…)
- compactness: #features
- …



Numeric representations

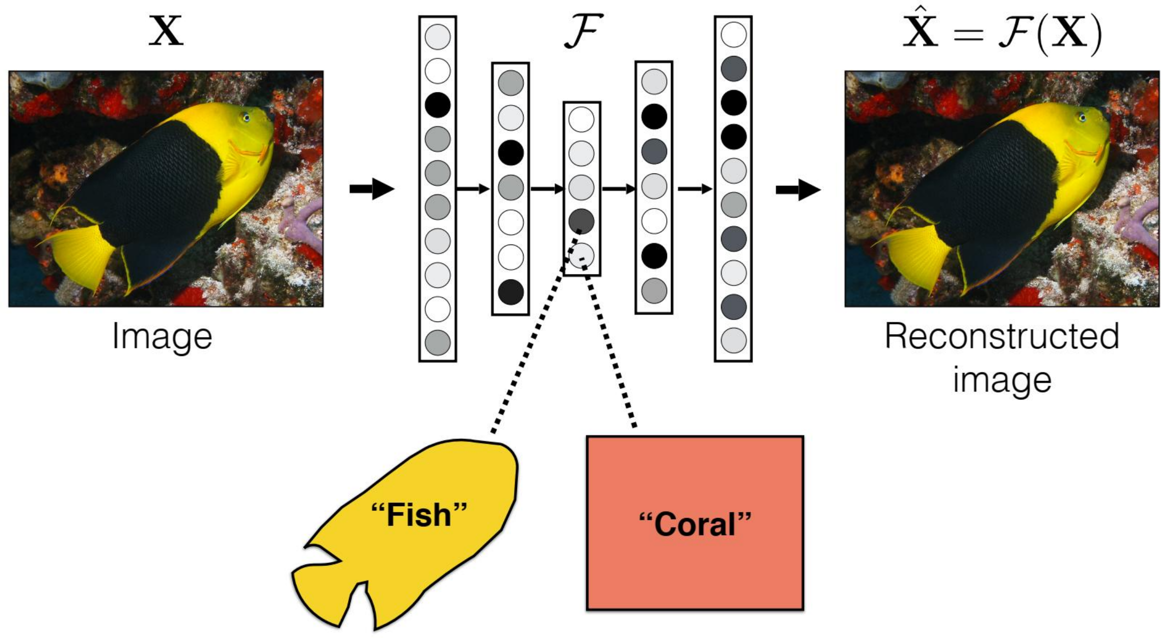
- **Goal**: capture underlying explanatory factors attending to their specific constraints
    - coverage, invariance, hierarchy, disentanglement, diversity, sparsity, smoothness…
- *Clue*: **numeric representations** may offer interesting properties!
- *Still*: some hard trade-offs
    - attaining *coverage* (preserving as much input information) together with nice properties (e.g., feature *independence*, *sparsity*, *invariance*)
- It seems a *hard task*! How to?



by Isola, Freeman, Torralba

Autoencoders

- How then? A classic approach is to learn an **autoencoder**



Autoencoders

- How then? A classic approach is to learn an **autoencoder**:
  - learn an encoder $g\colon X \to Z$ and the corresponding (g^{-1}) decoder $d\colon Z \to X$
  - the **goal** is to reconstruct the input, i.e. $\mathbf{x} \approx d(g(\mathbf{x}))$
  - we can learn the encoder and decoder functions from data by minimizing a loss
    - reconstruction loss, $Loss(\mathbf{x}) = Loss(\mathbf{x}, d(g(\mathbf{x})))$

      using, for instance, the mean squared error (MSE) on numeric features
- Yet… How to learn encoder-decoder functions without previous knowledge?
  - **Neural networks**!
  - and indeed… neural representations can attain some of the nice properties!

Outline

Neurons…

- Central unit of our brain: **neuron**
- Computational peer: *perceptron* (node)
  - given an input signal: set of features

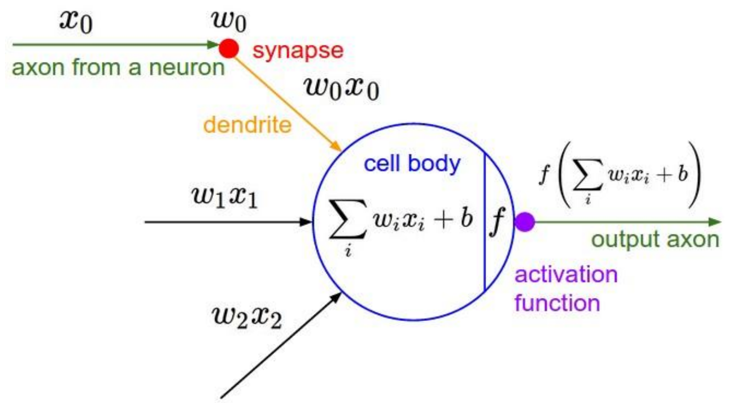
    $\mathbf{x} = (x_1, .., x_m)$
  - returns an output quantity: weighted sum

    $z = w_0 + \sum_{i=1}^{m} w_i x_i$

    (linear function of input features)
- Learning a perceptron $f: X \rightarrow Z$…
  - iteratively adjusting the parameters (weights)

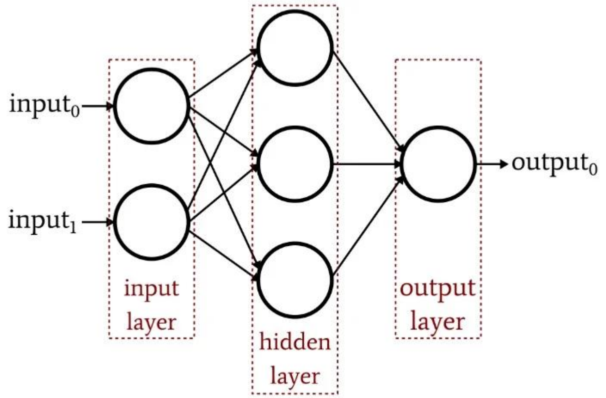
    $\mathbf{w} = (w_1, .., w_m)$

    from pairs $\{(\mathbf{x}_1, z_1), .., (\mathbf{x}_n, z_n)\}$



Neural networks

- Yet… most real-world predictive and descriptive problems not well described by linear functions
- And indeed… our brain is a complex **connectome**
  - 10^{40} neurons
  - 10^{4-5} connections per neuron
  - necessary for neuroplasticity: learning and memory
- In maths, we can compose ***linear*** functions, $f(x)$ ang $h(x)$, to form a ***non-linear*** function, $c(f(x), h(x))$ for modeling more complex behavior
- How to compose neurons (nodes)?
  - the composition can be **organized** in **layers**
  - the outputs of each node in one layer feeds the input of the nodes in the next layer



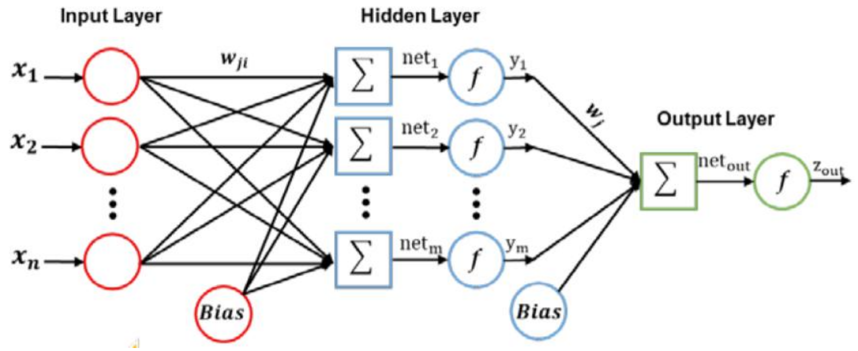
TÉCNICO+ FORMAÇÃO AVANÇADA

Non-linear models

Multiple layers of cascade linear units
still produce only linear functions

$$z_{out} = \sum w \left(\sum w \, x_i \right)$$

- *Solution*?



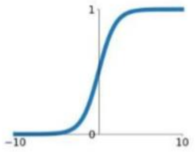
$$z_{out} = f\left(\sum w \, f\left(\sum w \, x_i \right) \right)$$

Activation functions

A large part of the expressiveness of networks are driven by the f functions, termed **activations**

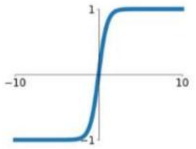
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



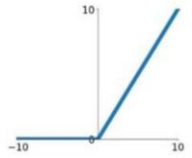
tanh

$\tanh(x)$



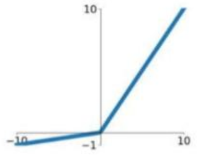
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

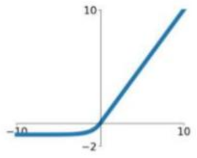


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



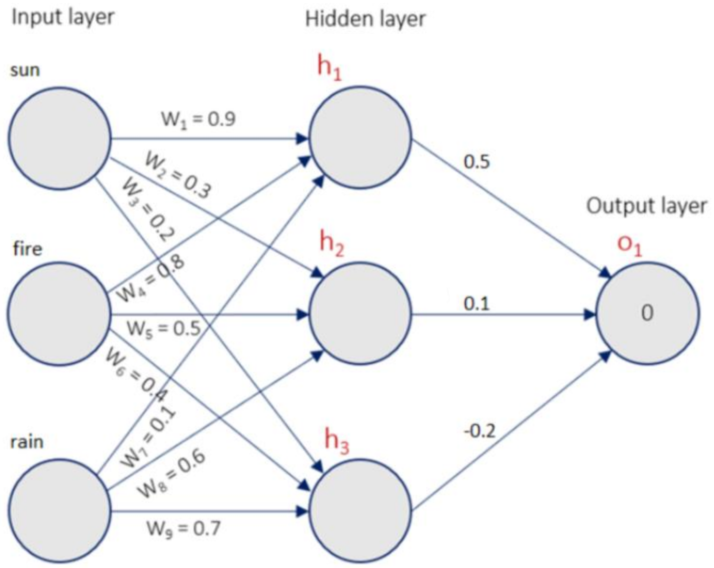
Exercise moment

Consider the following multilayer perceptron:

- trained on pairs (*document*, *graded relevance*)
- **inputs**: frequencies of specific terms in a document
- **output**: how likely a document covers natural catastrophes
- all biases as zero ($w_0 = 0$) and rectifier ($ReLU$) activation on the hidden and output nodes where $ReLU(x) = \max(0, x)$

Exercise: score document $\mathbf{x}$ knowing the frequencies of sun, fire and rain are 1, 0 and 2, i.e. $\mathbf{x} = (1,0,2)$

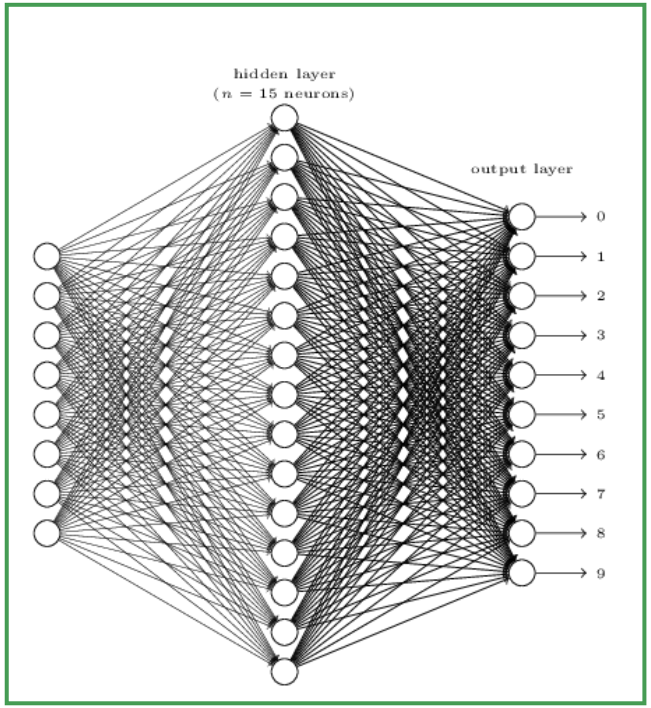
Solution: a simple propagation step is necessary



$$\begin{aligned}\hat{z} &= 0.5 \times h_1 + 0.1 \times h_2 - 0.2 \times h_3 \\ &= 0.5 \times (\max(0, 0.9 \times 1) + \max(0, 0.8 \times 0) + \max(0, 0.1 \times 4)) + 0.1 \times (\dots) - 0.2 \times (\dots) = 0.7\end{aligned}$$

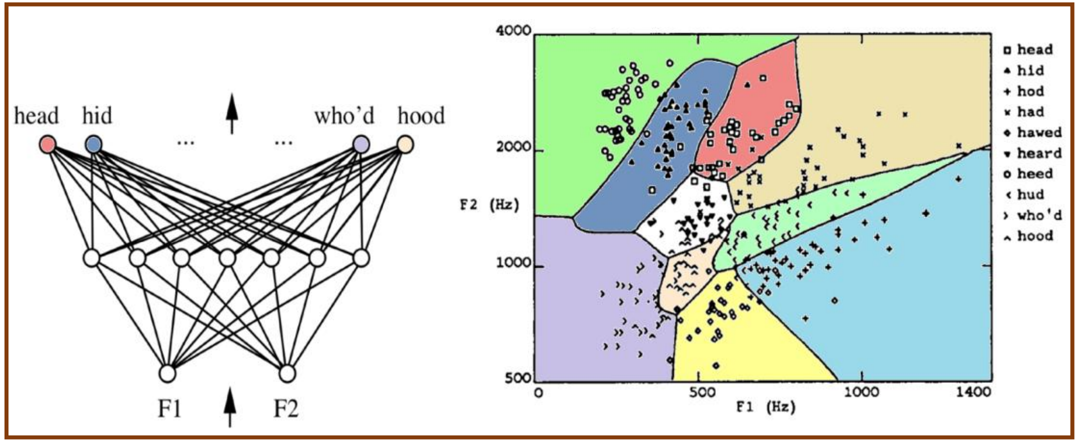
Neural networks for predictive tasks

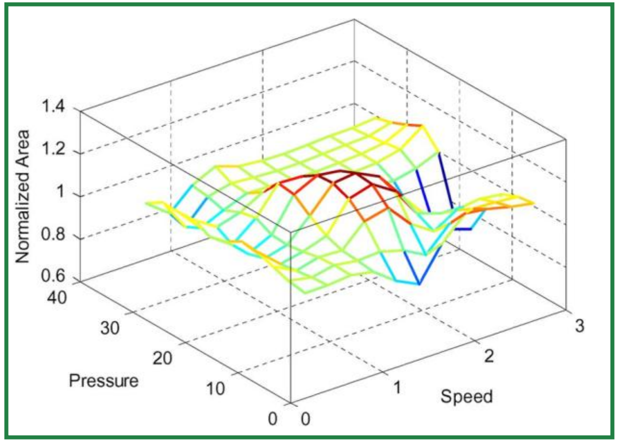
- The previous neural networks can be used for:
  - **single-output regression** (estimate quantity)
  - **binary classification** (output above a given threshold θ)
- Neural networks can have multiple output nodes:
  - each output is a category in **multi-class classification** (e.g., document categorization, $\boxed{\text{digit recognition)}}$ $\longrightarrow$
    - the output category is the node with highest value
  - **multiple-output prediction**, i.e. multiple targets (e.g., autonomous driving – speed and direction)



Expressive boundaries

- Non-linear (often non-convex) predictors
  - expressive hyperboundaries in classification problems (*left*)
  - expressive hypersurfaces in regression problems (*right*)





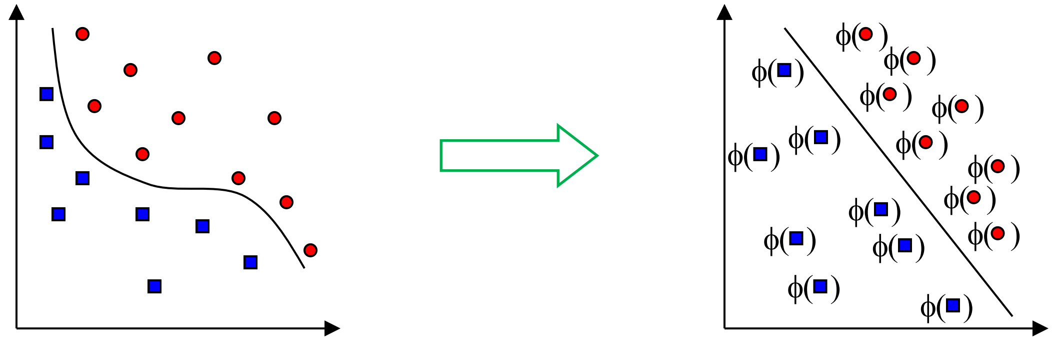
Learning neural networks

- **Goal**
  - given a training set of input-output pairs $(\mathbf{x}_i, \mathbf{z}_i)$, learn the **parameters** θ of the network
  - the *parameters* are essentially the weights from all layers
- **How?**
  - finding the weights that minimize a *loss* between the real and estimated outputs
    - a common loss is the squared error $L(\hat{\mathbf{z}}, \mathbf{z}) = ||\hat{\mathbf{z}} - \mathbf{z}||^2$
    - ***gradients*** are used to adjust weights in a direction that decreases the loss
      - the adjusting process is often termed *gradient descent* (GD) or *backpropagation*
  - randomly initialize weights and iteratively adjust throughout epochs using batches of pairs
    - the *learning rate* η and *batch size* controls the intensity and stability of weight updates

Neural processing layers as transformations

- A layer $[p]$ can be seen as a non-linear mapping, transforming the input signal $\mathbf{x}^{[p-1]}$ into a new signal $\mathbf{x}^{[p]}$

$$\mathbf{x}^{[p]} = \phi(\mathbf{W}^{[p]}\,\mathbf{x}^{[p-1]} + \mathbf{b}^{[p]})$$

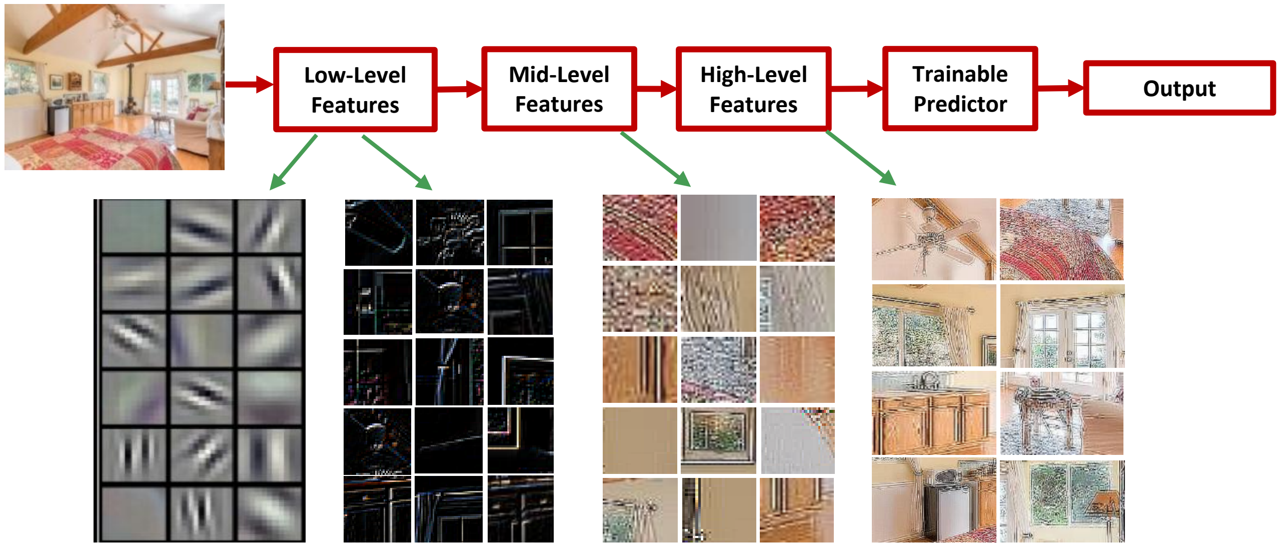


TÉCNICO+
FORMAÇÃO AVANÇADA

From single to multiple hidden layers

Multi-layer learning process to extract rich features (good data representations)

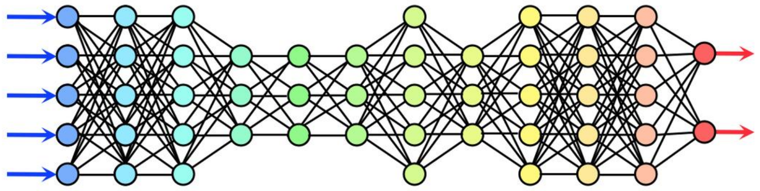
- **image**: pixels → edges → textures → motifs → parts → objects
- **text**: character → word → word group → clause → sentence → story



by Param Vir Singh

Deep Learning

- DL is a ML subfield dedicated to the learning of models with a high number of parameters
  - paradigmatic case: neural networks (*deep* in reference to the #layers and/or #parameters)
- Mathematically: shallow NNs (few layers, high #nodes per layer) should be as good as deeper NNs
  - the fact that deep NNs work better is empirical: better convergence and computational scalability
- Advantages of deep NNs
  - **outperform** other ML techniques in **many tasks**
  - expressivity to learn from high-dimensional data (including image, text, signal)
  - effective **end-to-end learning** system (can bypass the need for explicit feature extraction)



- *Challenges* of deep NNs: larger amounts of data required, prone to overfitting (regularization necessary)

Hyperparameterization

- How many layers? Some of the mentioned hyperparameters:
  - #layers, #neurons per layer, activations
  - loss function, learning rate, batch size
  - others: layering (*more to come*!), regularization (e.g. penalty, dropout rate), momentum, decay…
- The hyperparametric choices define the **architecture** of the neural network
- So many choices 😐 How do we select the best hyperparmeters?
  - <u>manual</u> optimization (rely on *intuition* yet discouraged as DL is not rocket science)
  - <u>automatic</u> optimization can be exhaustive (grid searches) or approximate (default option)
    - many good packages available for effective approximate optimization (e.g. *optuna*)

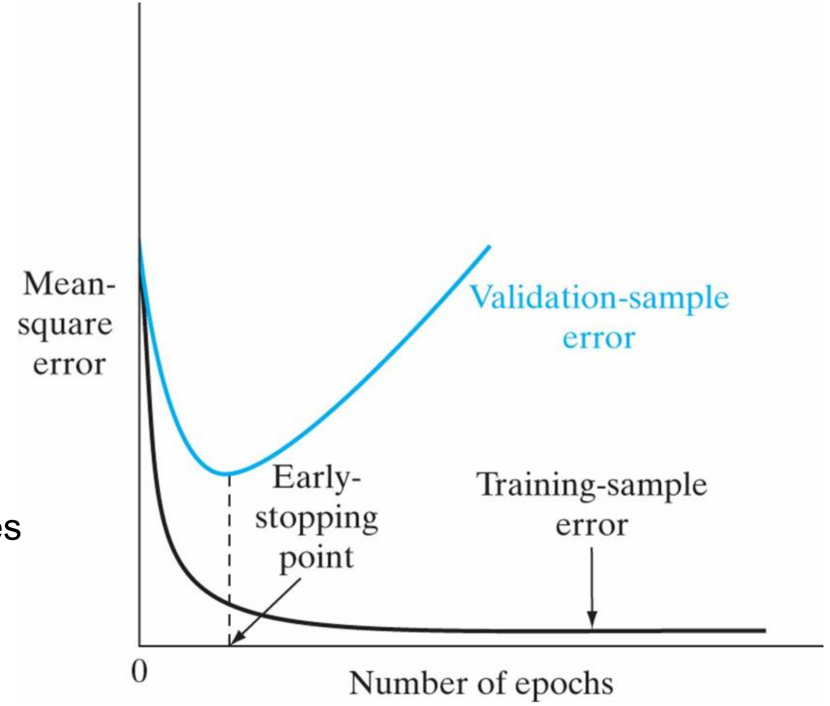
Convergence: early stopping

- We can keep optimizing the network weights…

  … until the network perfectly overfits the training data, hampering the ability to generalize to unseen data

- One possible solution? **Early stopping**
  - stop convergence before MLP overfits data
  - **how?**
    - optimize weights with training data, yet **assess the loss on a validation set**
    - stop learning when the validation error increases along few iterations (evidence of overfitting)

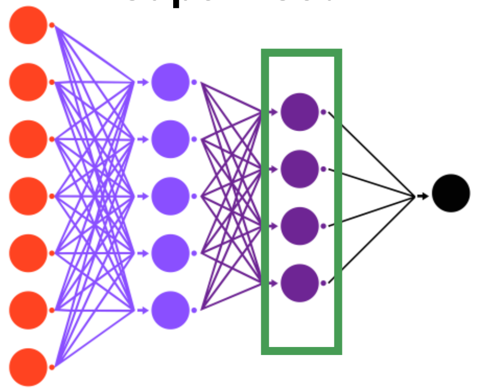
- Deep networks? Still provide a max number of epochs!



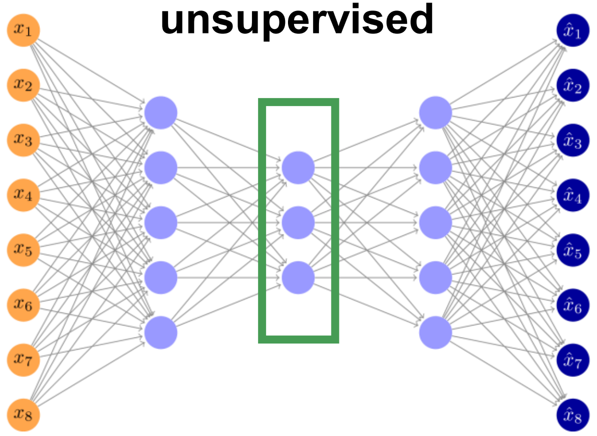
NNs for Representation Learning

- Two major schools for obtaining a **numeric representation** (termed ***embedding***):

supervised



unsupervised



- Hybrid variants: network with supervised and unsupervised paths

Supervised data representations

- The first school seems a natural choice…
  - extracts features in a latent space (hidden layer) with inherent predictive value for a selected task
- **Problems**
  - discriminative yet not necessarily descriptive
    - unable to represent/reconstruct input
    - no guarantees of smoothness, invariance
  - biased towards the target predictive task
- **Solutions**
  - multi-task learning stances
    - extend the number of outputs to capture multiple tasks
    - features become less biased and more informative towards varying ends
  - unsupervised learning stances ⇒ *next!*

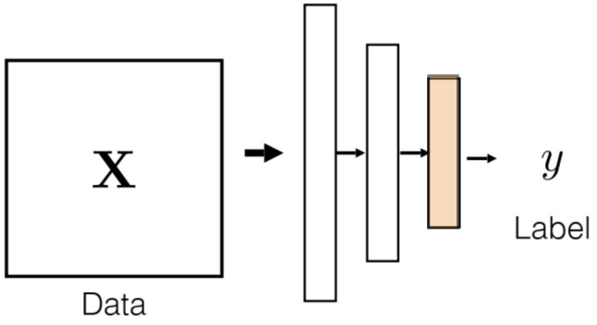
Outline

- Data Representations
- Neural Networks
- **Unsupervised and Self-supervised Stances**
- Handling Complex Data Structures
- Multimodal Data Representations

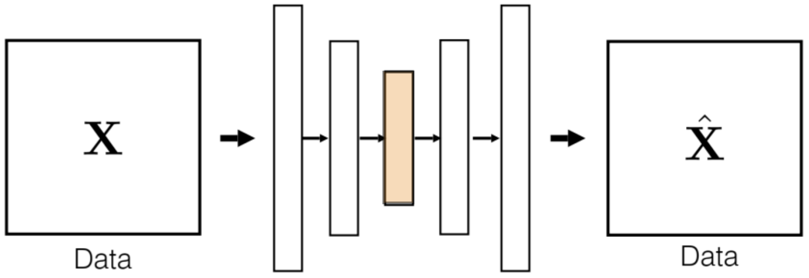
Unsupervised vs supervised stances

- *Recall* the two major paradigms for data representation

supervised



unsupervised



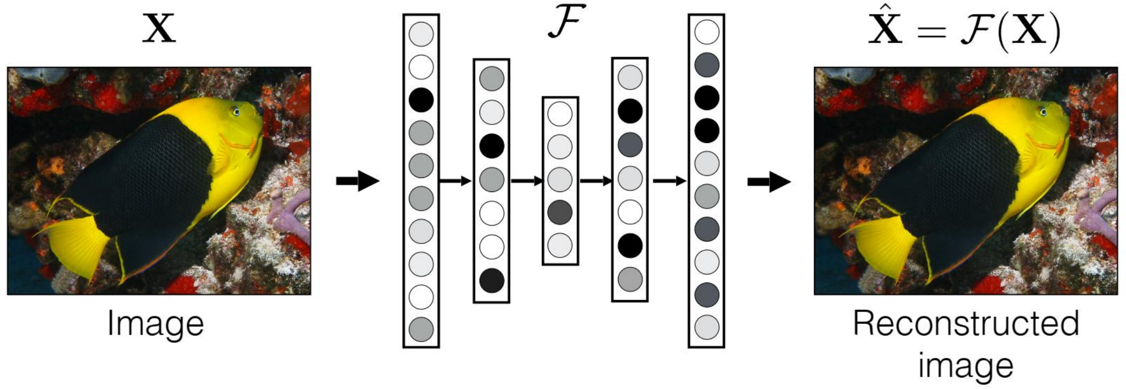
by Isola, Freeman, Torralba

TÉCNICO+
FORMAÇÃO AVANÇADA

Autoencoders

Recall: autoencoders as **unsupervised** approaches to representation learning

- **goal:** learn a compact representation (*embedding*) to reconstruct input, i.e. $\mathbf{x} \approx d(g(\mathbf{x}))$
  - learn the parameters of the encoder $g\colon X \rightarrow Z$ and decoder $d\colon Z \rightarrow X$ minimizing a reconstruction loss such as $||\hat{\mathbf{x}} - \mathbf{x}||^2$ where $\hat{\mathbf{x}} = d\big(g(\mathbf{x})\big)$

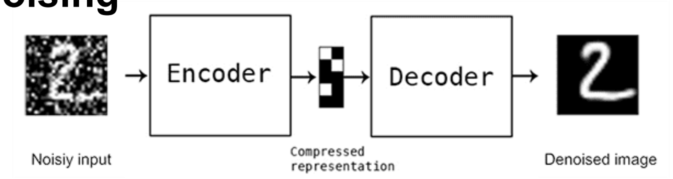


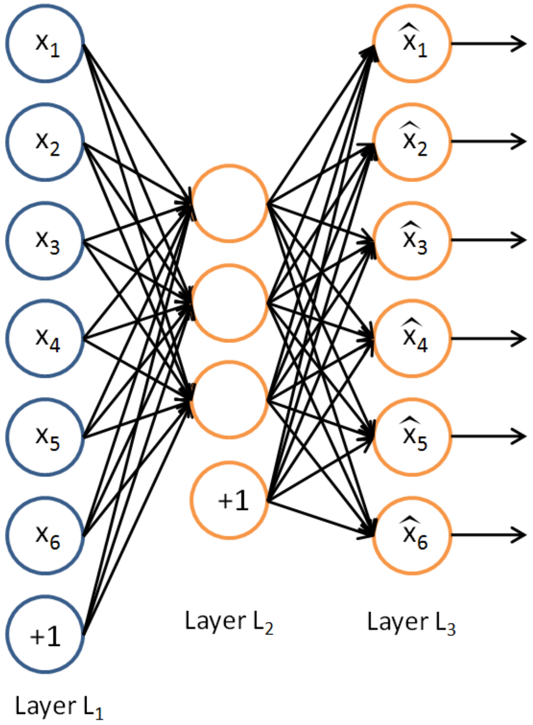
$$\arg\min_{\mathcal{F}} \mathbb{E}_{\mathbf{X}}[||\mathcal{F}(\mathbf{X}) - \mathbf{X}||]$$

50

Autoencoders

- Learning an **autoencoder** ≡ learning a network that learns to predict the input in the output
- Beyond representation…
  - **compression**
  - **visualization** (in lower spaces)
  - **manifold learning**
  - **denoising**

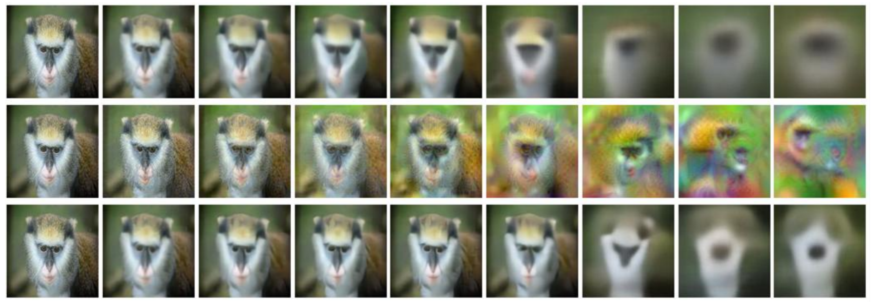




Explainability

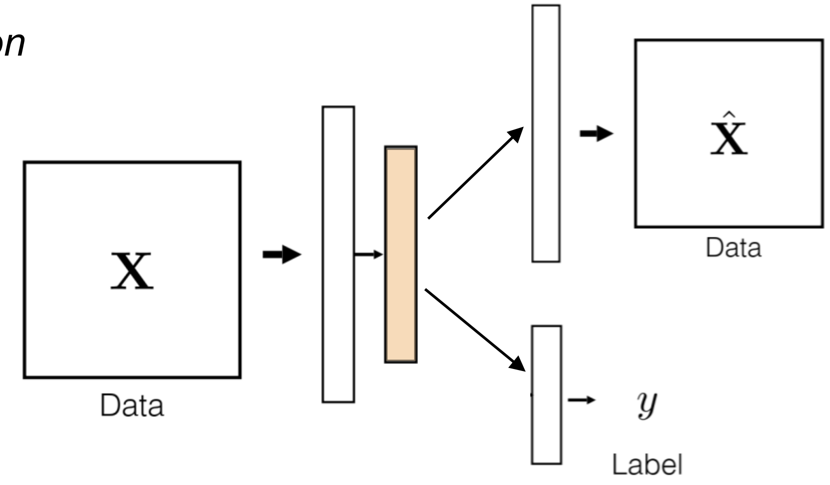
- Meaning of the *embedding* features
  - **saliency maps** in the original/raw inputs (check our *Explainability* class)
- **Inverting representations** (all *versus* subset of factors)

[81 20 84 64 58 39 17 54 72 15]



Hybrid architectures

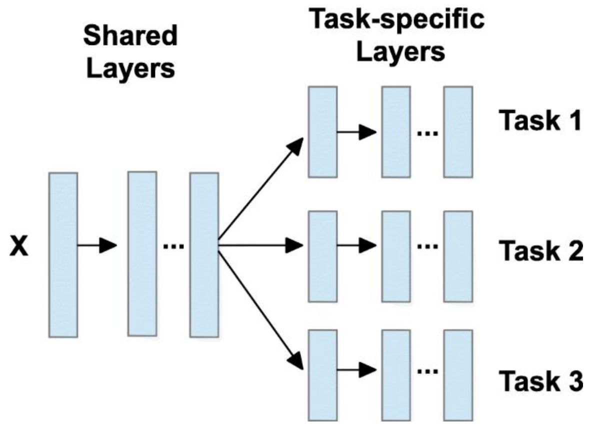
- **Downside** of purely unsupervised approaches
  - focus on reconstruction may not ensure that the extracted features yield the sufficient predictive power to effectively handle predictive tasks
- **Solution**: combining *supervision* and *unsupervision*
  - **single encoder**
  - **two decoders**
    - a dedicated path for the predictive task, other for reconstruction
    - loss with two components as well
    - parameters of decoders updated simultaneously or alternatively



Multi-task learning

Going beyond two dedicate paths…

- as many paths as relevant predictive/descriptive tasks
    - ensure representations are more expressive generalizable for multiple downstream ends



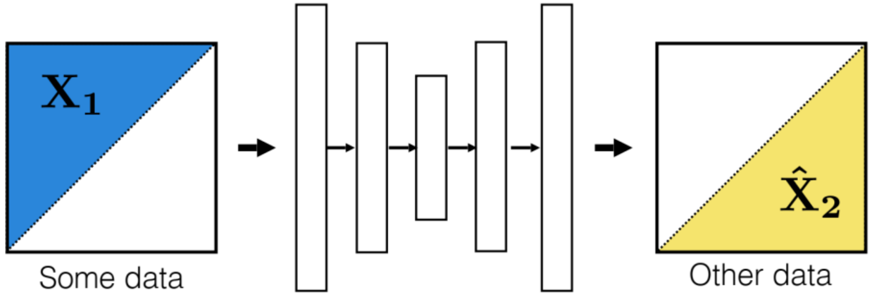
Examples:

- on time series: forecasting, anomaly detection, imputation (reconstructing incomplete observations)
- on text: tagging, syntactic parsing, sentiment analysis, text classification, translation

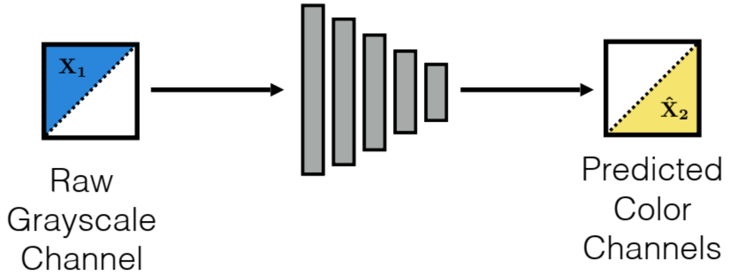
Self-supervision

In the absence of annotations (unsupervised settings), self-supervision can be pursued to emulate predictive tasks

- masking parts of data for reconstruction (e.g. word masking is at the basis of LLMs such as ChatGPT)

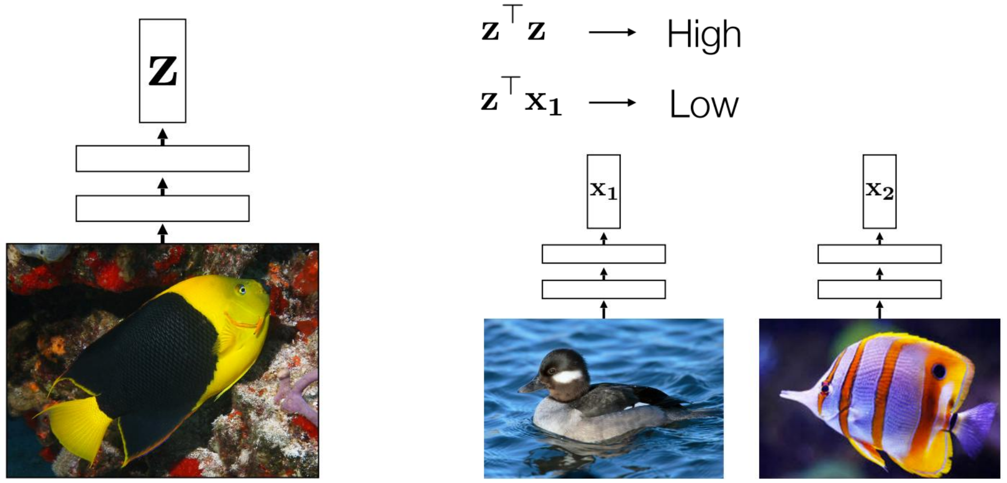


A supervised alternative for coloring tasks:



Contrastive learning

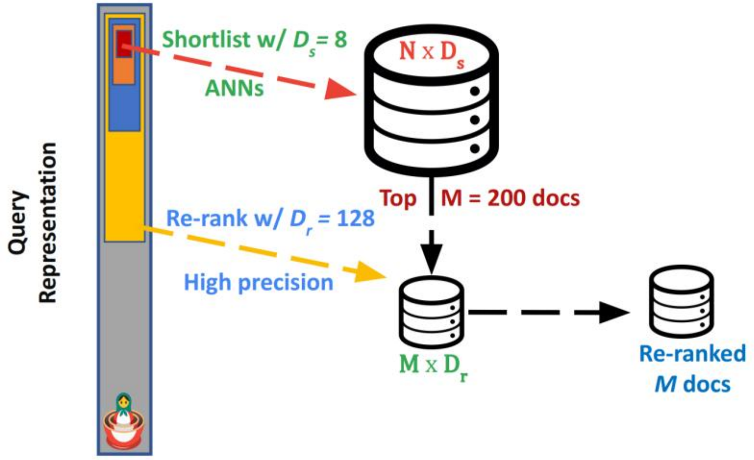
- Learn embeddings by comparing inputs rather than solely relying on reconstruction or prediction
- Form pairs of inputs and train models by
  - pulling representations of similar (positive) examples closer together
  - pushing representations of dissimilar (negative) examples farther


$\mathbf{z}$
$\mathbf{z}^\top \mathbf{z} \longrightarrow$ High
$\mathbf{z}^\top \mathbf{x}_1 \longrightarrow$ Low
$\mathbf{x}_1$
$\mathbf{x}_2$

by Isola, Freeman, Torralba

Multi-level embeddings

- *Challenge*: high-dimensional embeddings (e.g. thousands of dimensions for text) hamper tasks, such as retrieval and clustering, where you want to quickly find neighbors (closest records, images, signals, documents, webpages) for a given input (e.g., web search)
- *Solution*: learn multiple representations by training autoencoders with varying bottleneck sizes
  - small-sized embeddings used to find the nearest neighbor candidates (e.g., top 200) using cosine similarity
  - optimal-sized embedding of candidates to obtain a final ranking (e.g., ordered top 10)



Outline

- Data Representations
- Neural Networks
- Unsupervised and Self-supervised Stances
- **Handling Complex Data Structures**
- Multimodal Data Representations

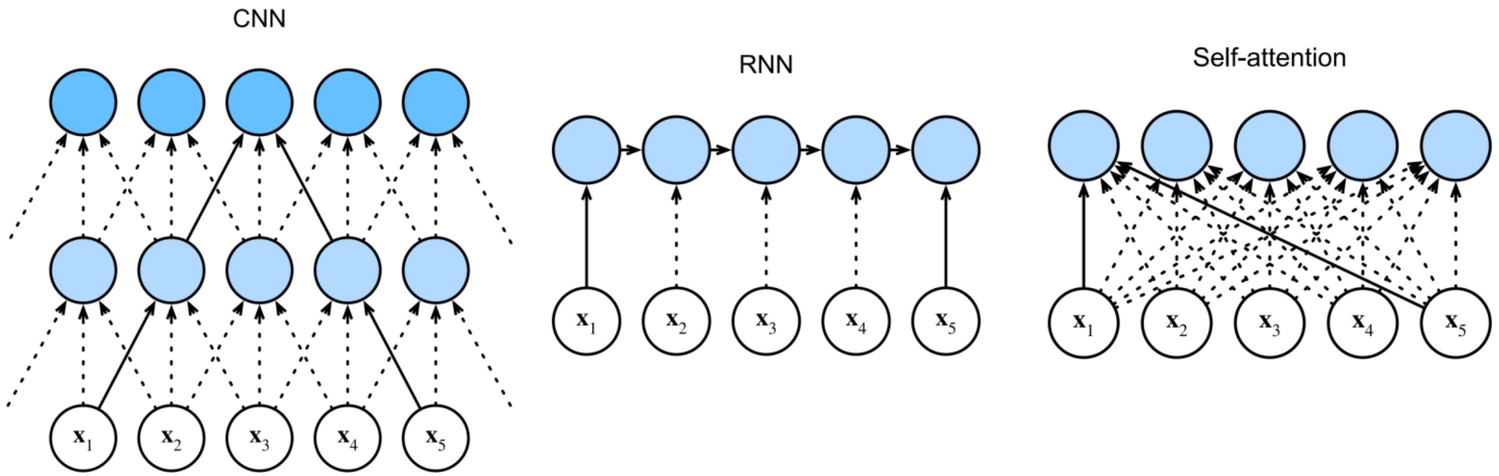
From MLPs to other neural architectures

MLPs originally proposed for the analysis of **simple multivariate data**, however…

- fully-connected layers are computationally expensive for high-dimensional data
  - given p units in the 1st layer: $m \times p$ parameters (high m, p in same magnitude) for this single layer
- other data structures are described by features with unique dependencies…
  - **image and video**: features are spatiotemporally correlated! **Spatial dependencies**
  - (multivariate) **time series**: features change along time! **Temporal dependencies**
  - **text and speech**: terms are situated (syntactical and semantical context)! **Sequential dependencies**
- we need to go beyond classic fully-connected layering…
  - ***convolutional*** layering (CNNs) for signal and image data
  - ***recurrent*** layering (RNNs) for time series data
  - ***self-attention*** layering for language data

CNNs, RNNs, self-attention

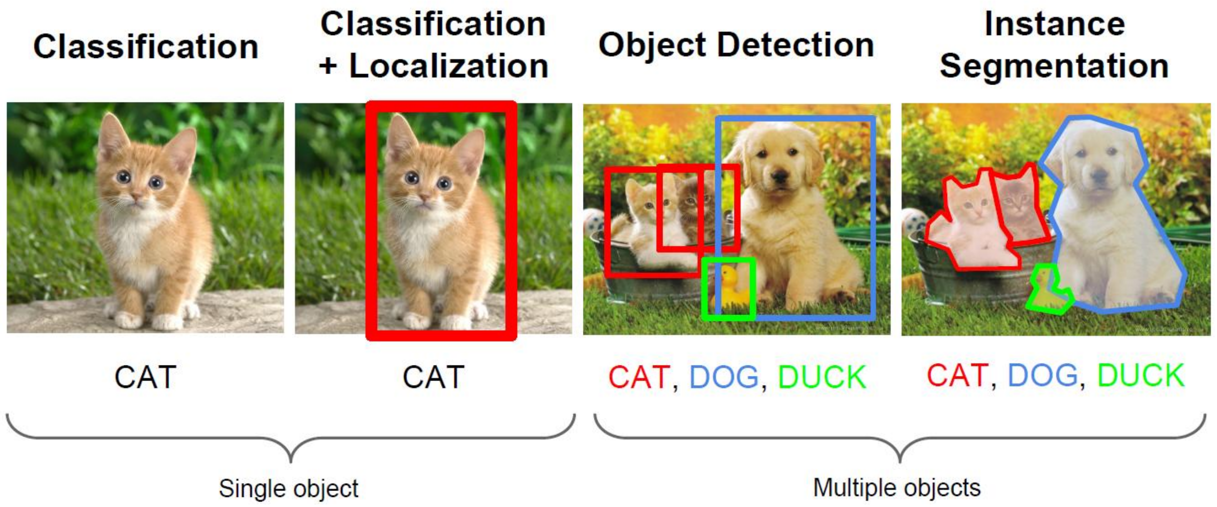
- **convolutions** and **recurrence** replace full connections with different forms of locality
- **self-attention** rely on flexible positional encodings (not requiring processing in any fixed order)



https://d2l.ai/chapter_attention-mechanisms-and-transformers/index.html

Computer vision tasks

- Computer vision has been a focal area of interest for ML
- Illustrative tasks: localization, object detection, instance segmentation

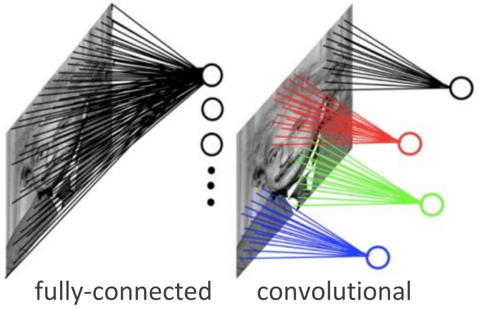


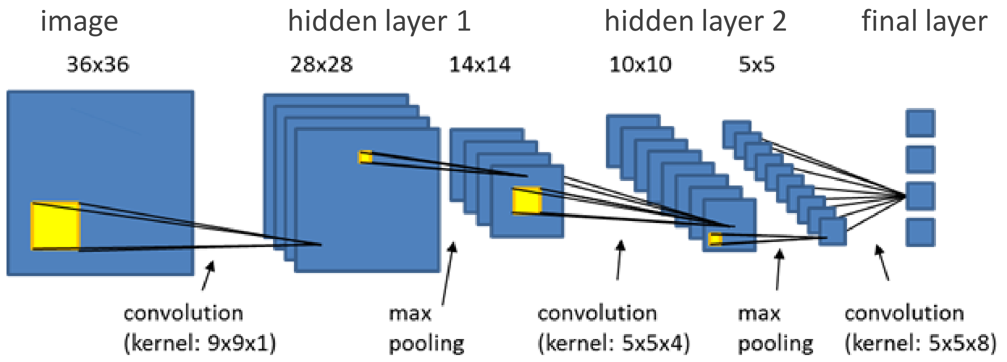
Picture from: Li, Karpathy, Johnson – Understanding and Visualizing CNNs

Convolutional NNs

Hidden units are connected to *local* receptive fields (*spatial dependencies*)

- much lower number of parameters than fully-connected layers!
- inspired by neurophysiological experiments [Hubel & Wiesel 1962]
- *convolution* and *pooling* operations form the basis of CNNs



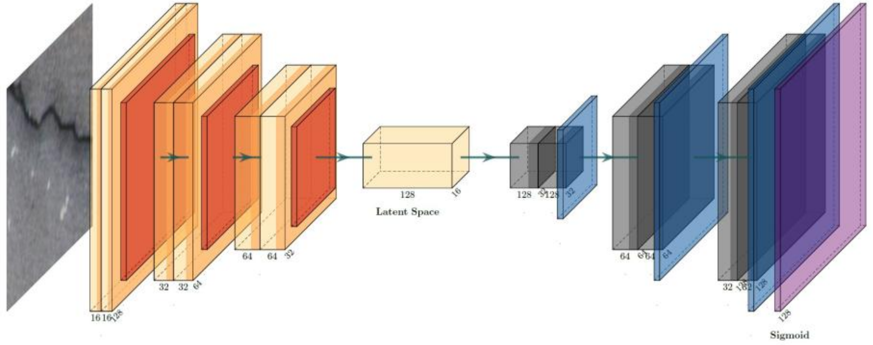


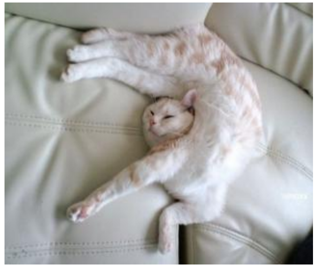
by eCognition

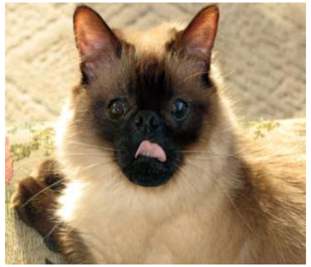
Convolutional autoencoders

Convolutions can be applied in autoencoder architectures to create image representations

- embedding size and layering should be sufficiently expressive to handle vision challenges…

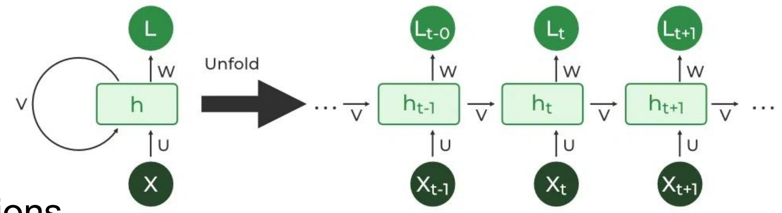






Recurrent NNs

- How to learn from *time series* data or *video* data?
  - CNNs can be used for these ends: 1D CNNs for time series, 3D CNNs for video data
  - however, the locality of convolution operations often neglect long-term temporal dependencies
- Recurrent NNs offer a possible way out
  - recurrent connections: a single unit to process features from one variable along time
    - memory of the previous inputs stored in the internal state to capture temporal associations

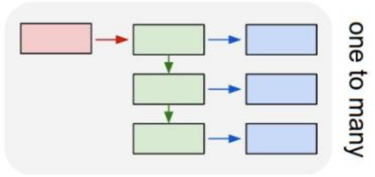
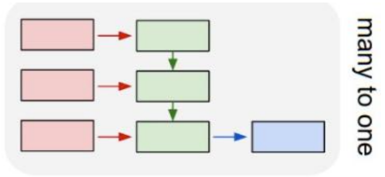
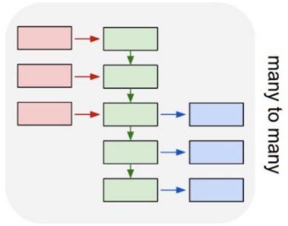


  - Long Short-Term Memory (LSTM) networks are a RNN variant with enhanced internal state
    - well-prepared to model long-term temporal dependencies
- Trade-off: RNNs are more sensitive to the vanishing gradient problem than CNNs, hence less stable

TÉCNICO+
FORMAÇÃO AVANÇADA

Recurrent NNs

RNNs traditionally applied over time series data… yet applications can go well beyond….

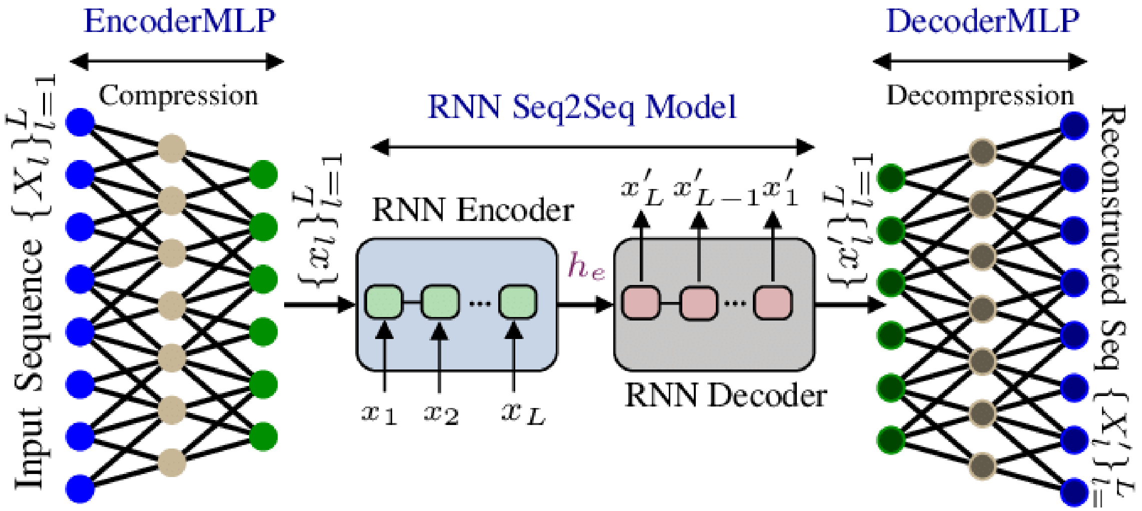
RNN	Application	Input	Output
	Video Captioning		A person riding a bike on dirt road
	Sentiment Analysis	Awesome movie. Highly recommended.	Positive
	Machine Translation	Happy Diwali	शुभ दीपावली

Recurrent autoencoders

… and, yes, recurrent layering can be also considered for **autoencoders**

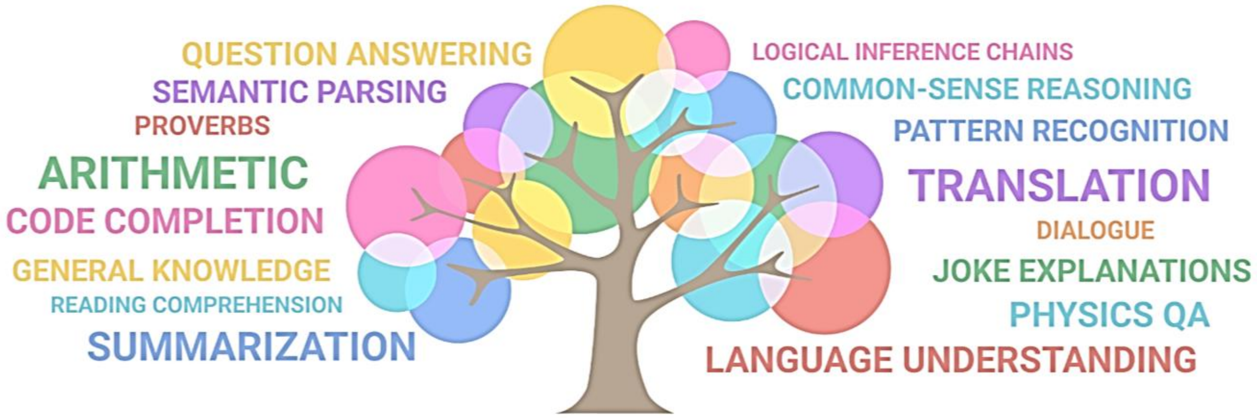
- find numeric representations of (multivariate) time series

Hybrid layering: combines different types of transformations



Large Language Models

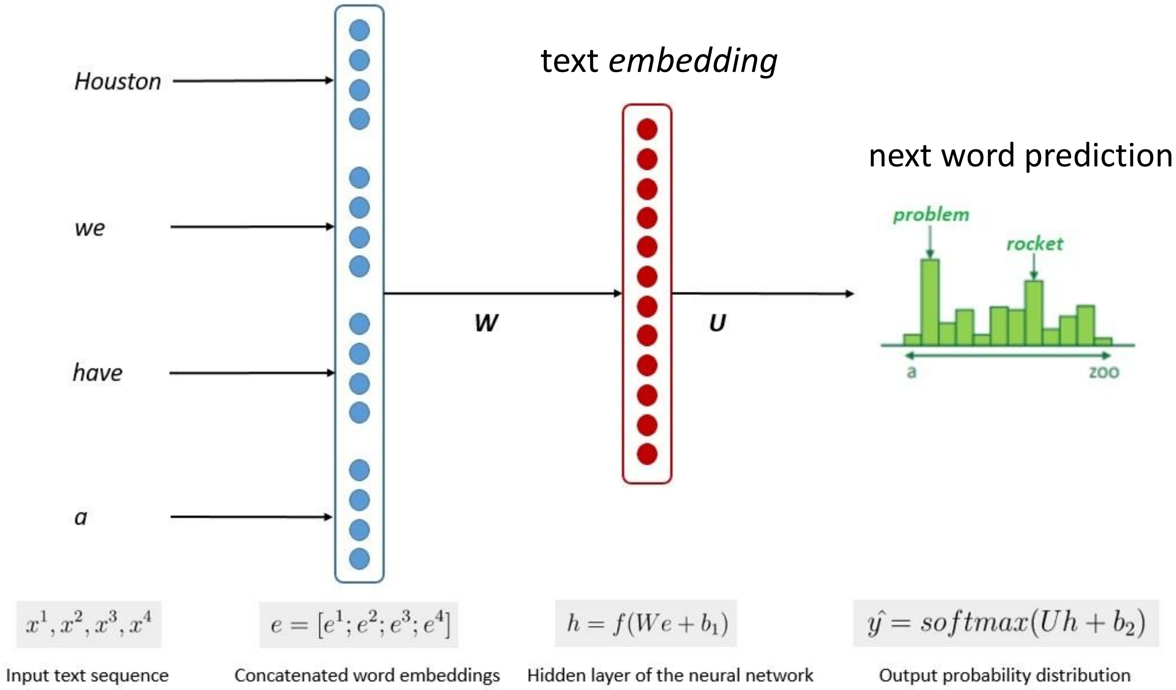
- Deep neural networks used for general-purpose language understanding and generation
- LLMs they embody the data representation principle!
  - expressive embeddings for high-efficacy downstream tasks
  - key given the diversity of downstream applications…



Large Language Models

Central principles:

- word embeddings
- positional embeddings
- self-supervision
- self-attention
- pre-training + fine-tuning

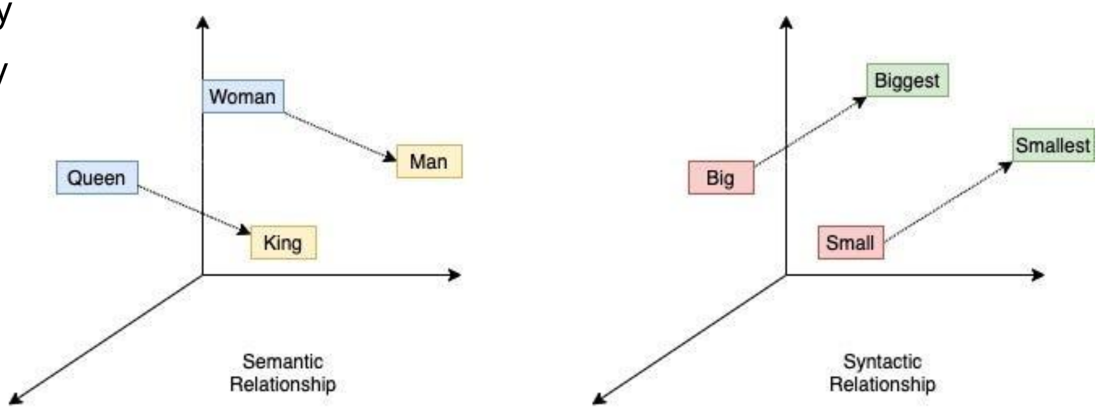

Houston
we
have
a
W
U
text *embedding*
next word prediction
problem
rocket
a
zoo
x^1, x^2, x^3, x^4
Input text sequence
$e = [e^1; e^2; e^3; e^4]$
Concatenated word embeddings
$h = f(We + b_1)$
Hidden layer of the neural network
$\hat{y} = softmax(Uh + b_2)$
Output probability distribution

https://www.baeldung.com/cs/large-language-models

LLMs: text encodings

Apply a dedicated NN to map **words** into **numeric vectors** capturing:

- semantic similarity
- syntactic similarity



Create a complementary numeric vector per word with its **positional encoding**

- aims at capturing relative positions of a word in text (e.g. “Mary loves John” ≈ “John is loved by Mary”)

LLMs: self-supervision

The sequence of word (and positional) embeddings in a text concatenated into a numeric **tensor**

- this tensor can be a text representation by itself, yet is high-dimensional and lacks expressivity
- padding/transforms entailed to ensure the tensor has uniform shape for texts with varying length

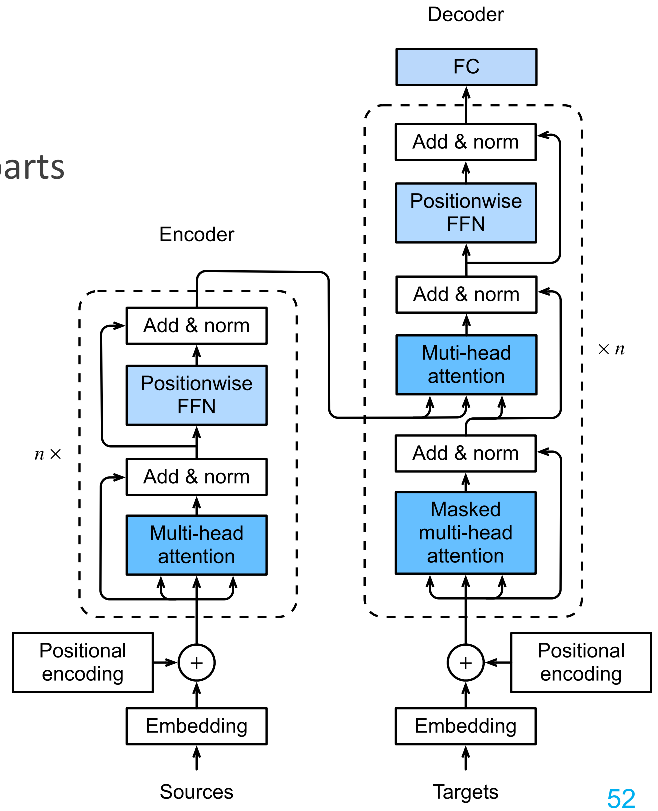
The tensor feeds a neural network with the aim of learning an expressive embedding

- autoencoders are rare since as text reconstruction is generally insufficient to capture text semantics
- supervised neural architectures are alternatively applied using a key principle
  - **self-supervision**
    - create fictious predictive tasks to learn embeddings
    - common: targets derived from unlabeled text by **masking words** to allow supervision
      - predictive task: classify the masked words (e.g., arbitrarily positioned word, *next* word)

LLMs: attention-based architecture

Architectural elements of an LLM

- **attention**: enhance parts of the signal while diminishing other parts
- **self-attention**: weighting the relevance of terms/tokens (from their embedding) to predict the masked term
- central component: attention-based **transformer** (*right*)
  - *encoder* layer encodes which parts of the input are relevant to each other
  - *decoder* takes all encodings to generate an output sequence
  - encoder-only (whole text modeling) and decoder-only (word prediction) variants are also common
- an LLM is a stacking of multiple transformer components with skip connections for multi-domain learning capacity

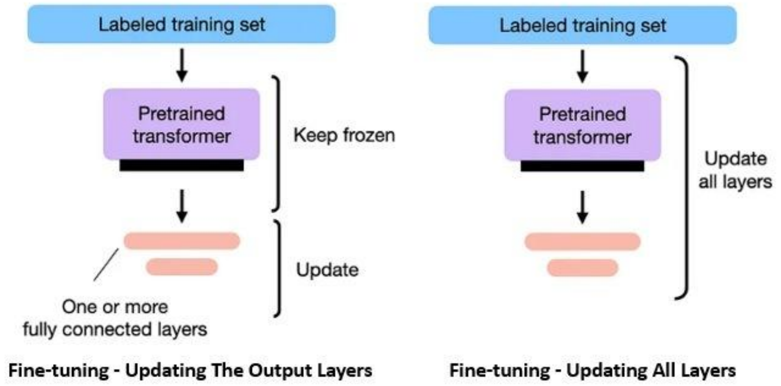


TÉCNICO+
FORMAÇÃO AVANÇADA

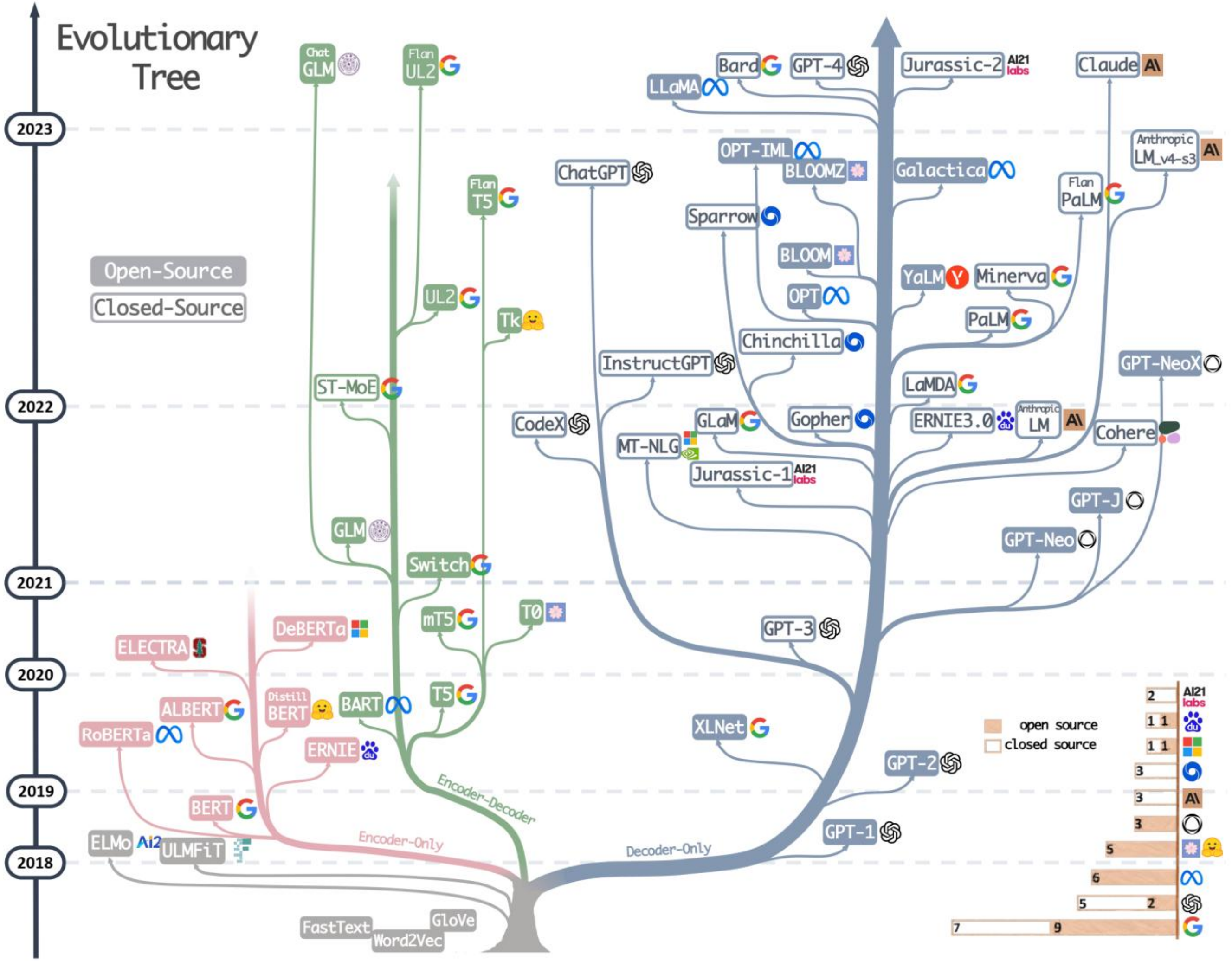
LLMs: pre-training and fine-tuning

- LLM are commonly **pre-trained** on different datasets and different tasks
  - going beyond single corpus and word prediction (e.g., semantic similarity between paired text)
  - the parameters of the first layers (i.e. transformers) are updated for the alternating tasks

- Pre-training followed by **fine tuning**
  - to orient the LLM for *specific tasks* and/or more *closely related data*
  - two major strategies
    - retrain all network (heavy)
    - retrain last layers only (preserve embeddings/intermediate signals)



https://www.baeldung.com/cs/large-language-models


Evolutionary Tree
2023
2022
2021
2020
2019
2018
Open-Source
Closed-Source
Encoder-Only
Encoder-Decoder
Decoder-Only
FastText
Word2Vec
GloVe
ELMo
AI2
ULMFiT
BERT
RoBERTa
ALBERT
ELECTRA
DeBERTa
Distill BERT
ERNIE
BART
T5
mT5
T0
Switch
GLM
ST-MoE
UL2
Tk
Flan T5
Chat GLM
Flan UL2
GPT-1
GPT-2
XLNet
GPT-3
GPT-Neo
GPT-J
GPT-NeoX
Cohere
Anthropic LM
ERNIE3.0
LaMDA
Gopher
GLaM
MT-NLG
Jurassic-1
AI21 labs
CodeX
InstructGPT
Chinchilla
PaLM
Minerva
YaLM
OPT
BLOOM
Sparrow
BLOOMZ
OPT-IML
ChatGPT
Galactica
Flan PaLM
Anthropic LM_v4-s3
LLaMA
Bard
GPT-4
Jurassic-2
AI21 labs
Claude
open source
closed source
2
1 1
1 1
3
3
3
5
6
5 2
7 9

Outline

- Data Representations
- Neural Networks
- Unsupervised and Self-supervised Stances
- Handling Complex Data Structures
- **Multimodal Data Representations**

Multimodal Representations

- Learning representations from **heterogeneous data** structures
  - combining features, text, vision, time series, events, trajectories, multiple omics…
- **Classic stance**:
  - extract features from each mode separately
    - *classic statistics* and/or *unimodal data representations*
  - concatenate the representations (larger numeric vector with mode-dedicated sections)
- **Problems**
  - informative features capturing cross-modal synergies are neglected
  - many of earlier representation principles no longer satisfied (e.g., succinctness, disentanglement)
    - dependencies between representations of each mode
  - what if one modality is not available for some training or testing observations?
- **Solution**: multimodal autoencoders and shared multi-task learning

Multimodal Representations

Multimodal LLMs are well-known class

- *embeddings* for coupled vision, audio, and text observations
  - captioning

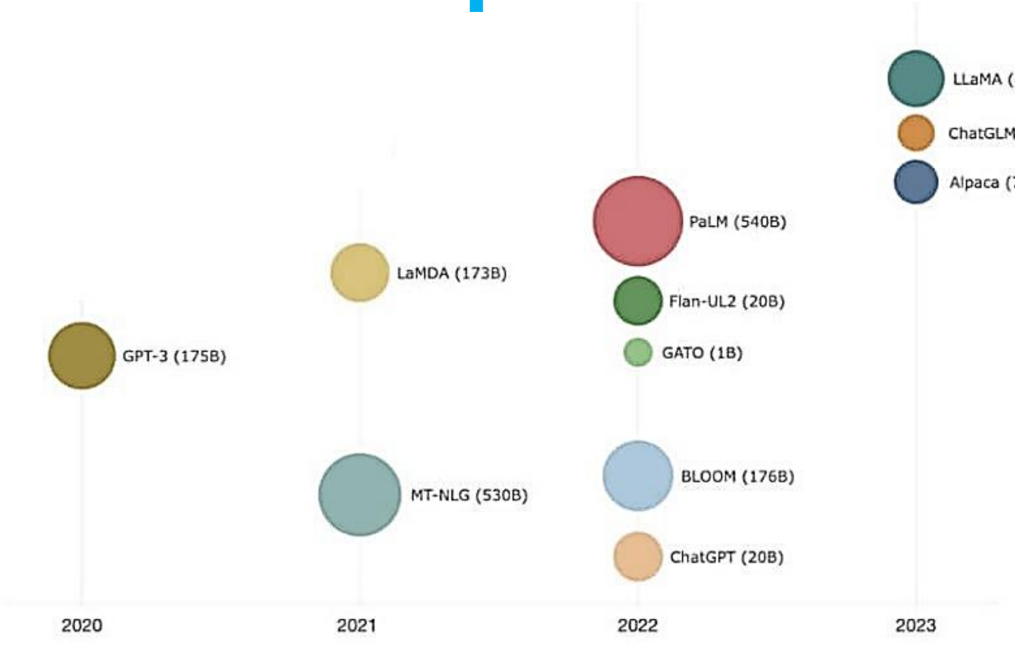
http://cs.stanford.edu/people/karpathy/deepimagesent

"two young girls are playing with lego toy." "boy is doing backflip on wakeboard." "construction worker in orange safety vest is working on road." "man in black shirt is playing guitar."

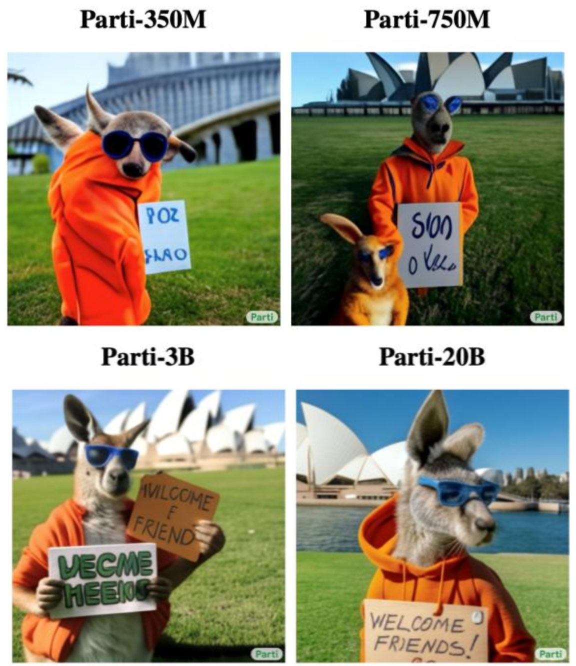
  - text-to-image generation
    - "teddy bear swimming at the Olympics"
    - "cute corgi lives inside sushi house"
    - "sloth holding treasure with bright golden light"

by Saharia et al. 2022

How deep?



https://vectara.com

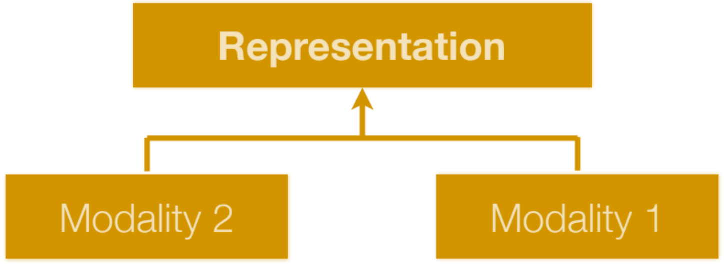


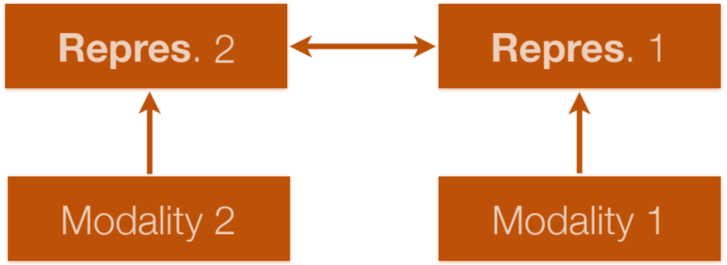
A portrait photo of a kangaroo wearing an orange hoodie with blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends! by Yu *et al.* 2022

TÉCNICO+
FORMAÇÃO AVANÇADA

Multimodal representations

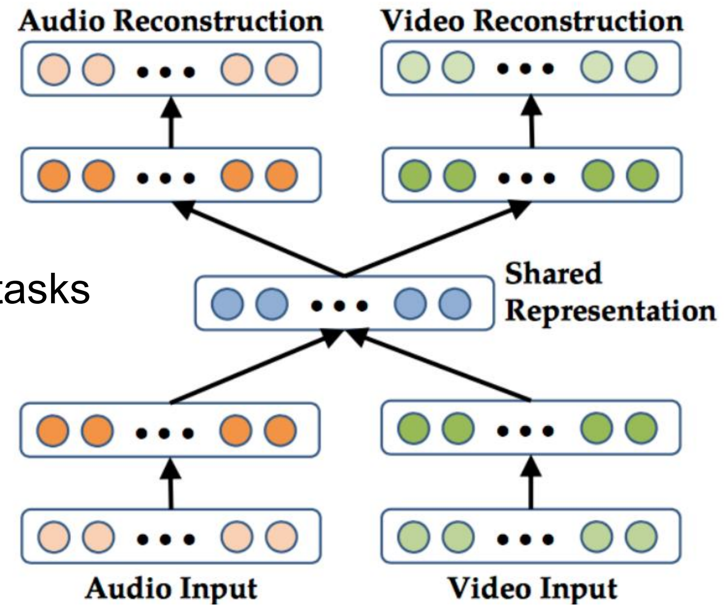
- Different strategies to learn multimodal representations, including:
  - **joint representation** (simplest version)
    - modality concatenation (early fusion)
    - unsupervised or supervised
    - similarity-based methods (e.g., cosine)
  - **coordinated representations**
    - constraints on structure (e.g., orthogonality, sparseness)





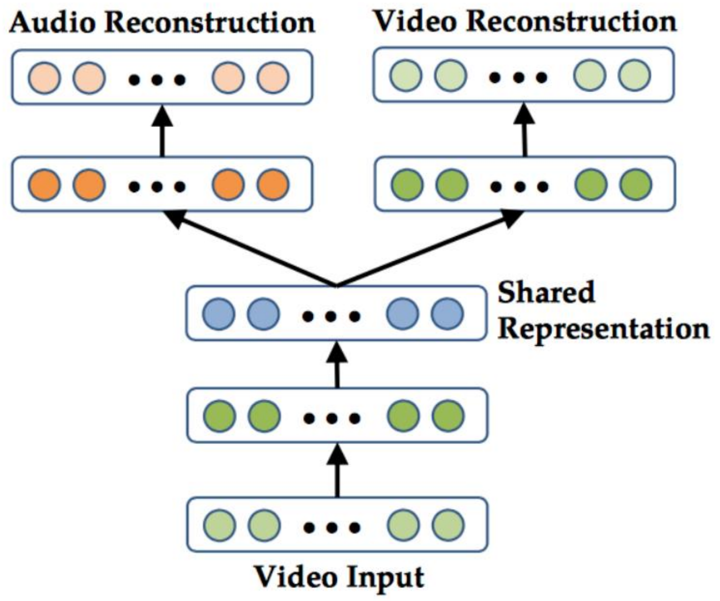
Joint representations

- Shared representations (joint *embedding*) capture cross-modal synergies
  - *efficiency*: lower dimensionality than the sum of individual representations
  - *expressivity*: higher efficacy for downstream tasks
- Each modality can be pre-trained (using modality-specific autoencoders) before adding the shared layer



Joint representations

- At *training* time:
  - all modalities can be inputted
  - a subset of modalities can be removed for some data instances
    - regarded as a form of regularization (dropout) that helps the training
- At *testing* time: if only some modalities are available, the remaining are generated



TÉCNICO+
FORMAÇÃO AVANÇADA

Outline

- Data Representations
- Neural Networks
- Unsupervised and Self-supervised Stances
- Handling Complex Data Structures
- Multimodal Data Representations

Thank you!

Rui Henriques

rmch@tecnico.ulisboa.pt