Deep learning theory and application for astrophysics

From machine learning basics to deep learning

Hypotheses and models

Objective

Build a smart thermostat that regulates heating/cooling, predicting what the temperature will be on a given day.

Hypotheses and models

Objective

Build a smart thermostat that regulates heating/cooling, predicting what the temperature will be on a given day.

Challenge

Predicting the temperature is complex:

- Physical laws
- Influencing variables (e.g., external temperature)

Alternative approach

- Historical temperature measurements
- Formulation of a **hypothesis**
- Estimate a mathematical function

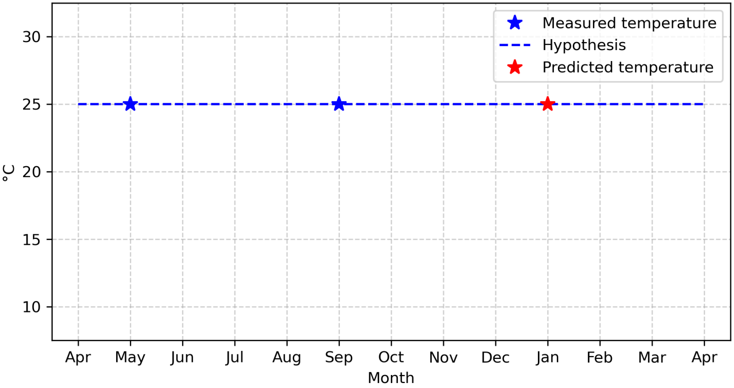
Alternative approach

- Historical temperature measurements
- Formulation of a **hypothesis**
- Estimate a mathematical function

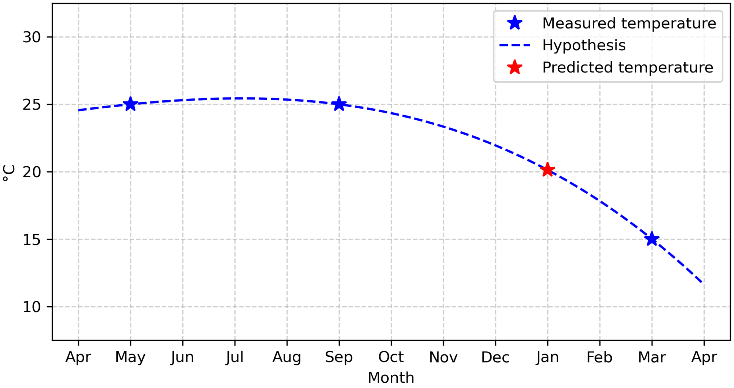
Machine Learning

Science that recognizes **patterns** from limited examples.

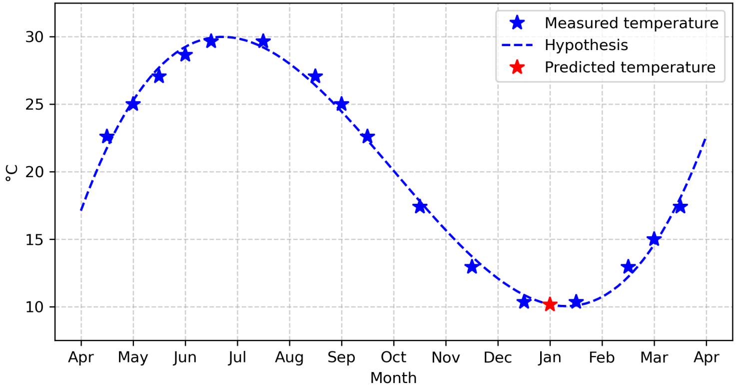
Importance of examples



Importance of examples



Importance of examples



Supervised machine learning

- **Dataset** $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- **Feature** x_i: data for prediction
- **Target** y_i: what we want to predict

Supervised machine learning

- **Dataset** $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- **Feature** x_i: data for prediction
- **Target** y_i: what we want to predict

Objective

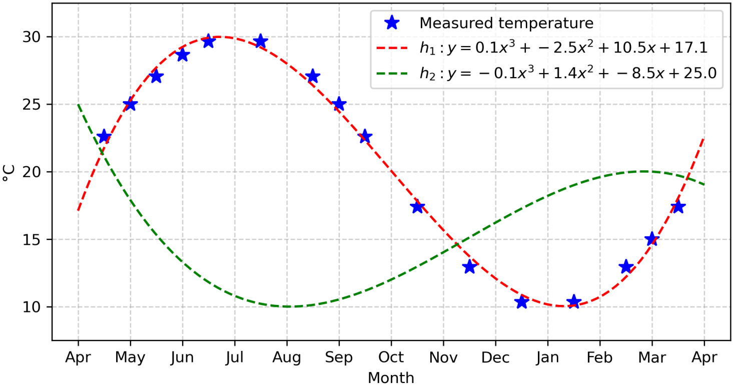
Find the hypothesis h:

$$h(x) \approx f(x), \forall x$$

Models and hypotheses

- Subset of hypotheses: **model**
- Family of hypotheses varies with **parameters**

Models and hypotheses



Model training

Model training

Loss function

Measures the error of the model

Model training

Loss function

Measures the error of the model

- **Training objective**:
  - Find parameters $\boldsymbol{\theta}^*$ that minimize the loss function $\mathcal{L}$
- Formally:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}\right)$$

Solving a machine learning problem

Solving a machine learning problem

1. **Model selection**
    - Suitable model for the problem
    - Risks of incorrect choice

Solving a machine learning problem

1. **Model selection**
   - Suitable model for the problem
   - Risks of incorrect choice
2. **Parameter optimization**
   - Find parameters for the best hypothesis
   - Exploration of the parameter space $\mathbb{R}^d$

Generalization

Loss optimization

- It is not enough for the model to perform well on training data
- We want it to perform well on data **never seen** during training

Generalization

Loss optimization

- It is not enough for the model to perform well on training data
- We want it to perform well on data **never seen** during training

Generalization capability

Correct predictions on **test** data

Generalization

Loss optimization

- It is not enough for the model to perform well on training data
- We want it to perform well on data **never seen** during training

Generalization capability

Correct predictions on **test** data

Dataset split

- **Training set** $\mathcal{D}_{\text{train}}$: training and optimization
- **Test set** $\mathcal{D}_{\text{test}}$: performance evaluation

Training and test loss

- **Training loss** $\mathcal{L}(\mathcal{D}_{\text{train}}, \boldsymbol{\theta})$
- **Test loss** $\mathcal{L}(\mathcal{D}_{\text{test}}, \boldsymbol{\theta})$

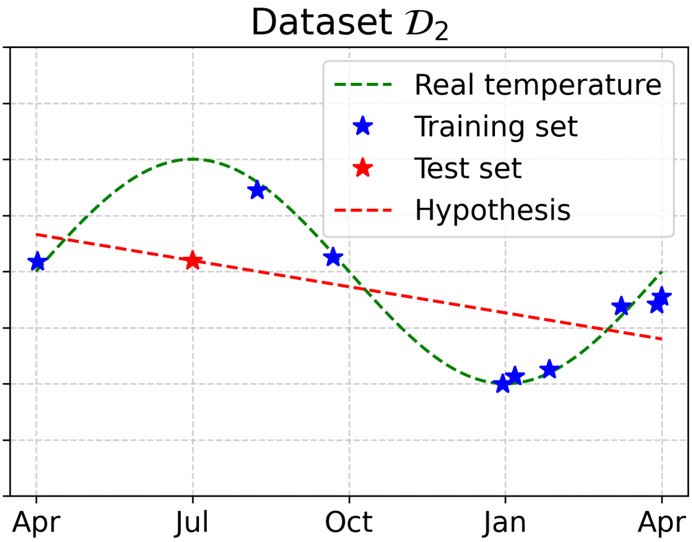
Training and test loss

- **Training loss** $\mathcal{L}(\mathcal{D}_{\text{train}}, \boldsymbol{\theta})$
- **Test loss** $\mathcal{L}(\mathcal{D}_{\text{test}}, \boldsymbol{\theta})$

Discrepancy

A model that performs well in training does not necessarily perform well in testing

Underfitting



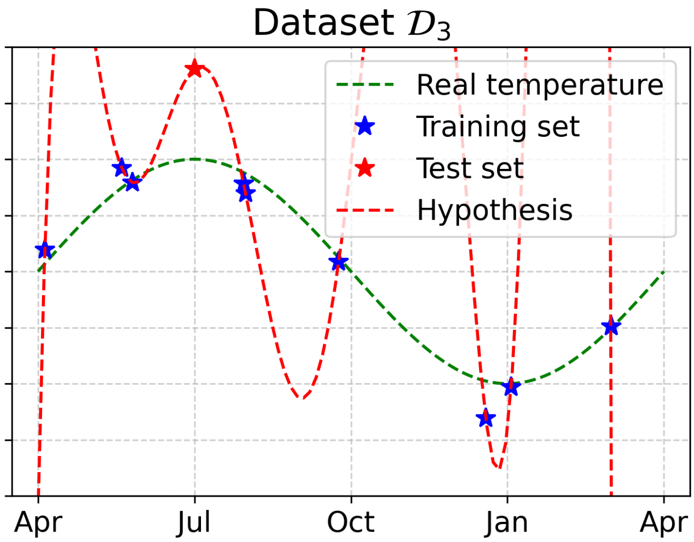
Underfitting

Definition

Inability to approximate the training data

- High errors on both training and test sets
- Limitation due to the simplicity of the model

Overfitting



Overfitting

Definition

The model learns the training data too well

- $\mathcal{L}(\mathcal{D}_{\text{train}}, \boldsymbol{\theta}) \approx 0$
- $\mathcal{L}(\mathcal{D}_{\text{test}}, \boldsymbol{\theta}) \gg 0$

Model choice

Possible approach

- Train different models on $\mathcal{D}_{\text{train}}$
- Evaluate performance on $\mathcal{D}_{\text{test}}$
- Choose the model with the best performance on $\mathcal{D}_{\text{test}}$

Model choice

Possible approach

- Train different models on $\mathcal{D}_{\text{train}}$
- Evaluate performance on $\mathcal{D}_{\text{test}}$
- Choose the model with the best performance on $\mathcal{D}_{\text{test}}$

Problem

- Test set used for model selection
- "Contaminated" its role in generalization

Validation set

Validation set

Estimate of generalization performance during training, without using the test set

Validation set

Validation set

Estimate of generalization performance during training, without using the test set

New division of the dataset

- **Training set**: training the parameters
- **Validation set**: model selection
- **Test set**: final evaluation

Validation set

Validation set

Estimate of generalization performance during training, without using the test set

New division of the dataset

- **Training set**: training the parameters
- **Validation set**: model selection
- **Test set**: final evaluation

Correct approach

- Divide $\mathcal{D}$ into $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{test}}$
- Train models on $\mathcal{D}_{\text{train}}$
- Evaluate on $\mathcal{D}_{\text{val}}$
- Select the model with the best performance on $\mathcal{D}_{\text{val}}$
- Evaluate the chosen model on $\mathcal{D}_{\text{test}}$

Introduction to artificial neural networks

History

- Peaks of popularity: 1950s-60s and 80s
- Decline: design and training complexity
- Revival: 2000s thanks to technological and algorithmic advances

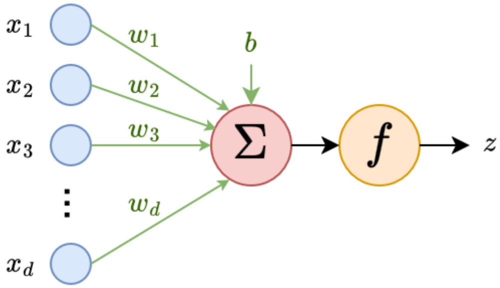
Artificial neuron

Definition

- Observation: $\mathbf{x} = [x_1, \ldots, x_d]^\intercal$
- Parameters:
  - **weights** $\mathbf{w} = [w_1, \ldots, w_d]^\intercal \in \mathbb{R}^d$
  - **bias** $b \in \mathbb{R}$
- f: **activation function**

Neuron output

$$z = f\left(\mathbf{x}^\intercal \mathbf{w} + b\right)$$



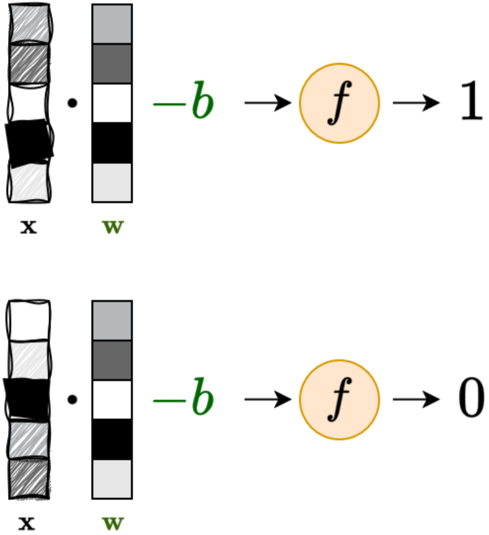
Interpretations of the artificial neuron

Neuron output

$$z = f\left(\mathbf{x}^\intercal \mathbf{w} + b\right)$$

Prototypical interpretation

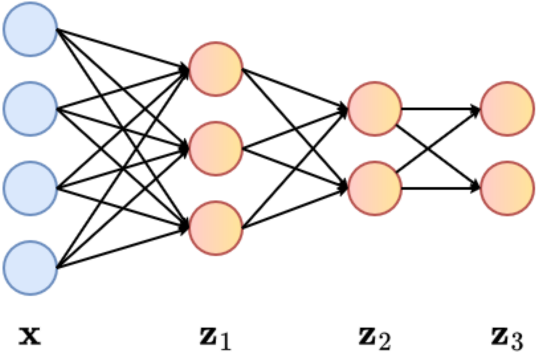
- $\mathbf{w}$: feature prototype
- $\mathbf{x}^\intercal \mathbf{w}$: measure of similarity
- Similarity threshold: $-b$



Multi-layer perceptron (MLP)

- Model equipped with **multiple layers** to approximate non-linear functions
- **Concatenation of layers**: output of one layer becomes input to the next

(Example)



Training a neural network

Objective

Minimize loss function $\mathcal{L}$

Training a neural network

Objective

Minimize loss function $\mathcal{L}$

Gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}$$

- $\boldsymbol{\theta}$: vector of network parameters
- $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \left[\frac{\partial \mathcal{L}}{\partial \theta_1}, \frac{\partial \mathcal{L}}{\partial \theta_2}, \ldots, \frac{\partial \mathcal{L}}{\partial \theta_p} \right]^\mathsf{T}$: gradient of the loss with respect to the parameters

Backpropagation

Definition

- Algorithm for computing gradients of a neural network
- Efficient and suitable for computer implementation
- Requires that layers are **differentiable**

Role of the learning rate

- Key hyperparameter in the gradient descent algorithm
- Determines the size of update steps

Choosing the learning rate η

- η too small: slow progress, difficulty escaping local minima
- η too large: risk of oscillations or divergence

Deep neural networks

Complex data

- Hierarchical and compositional nature
  - Images: pixels $\rightarrow$ edges $\rightarrow$ shapes $\rightarrow$ objects $\rightarrow$ scenes
  - Text: letters $\rightarrow$ words $\rightarrow$ sentences $\rightarrow$ complex meaning

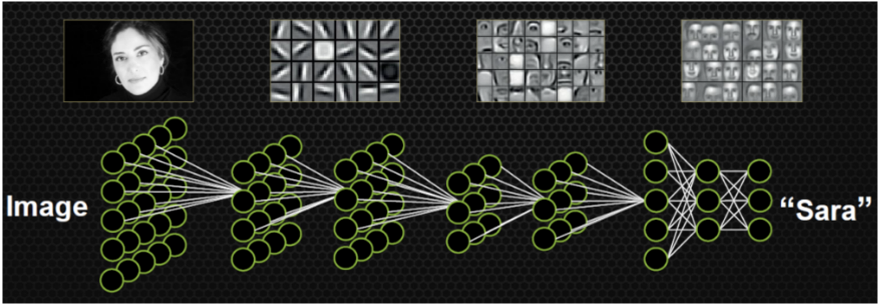
Deep neural networks

- Hierarchical representations
- Initial layers: low-level patterns
- Final layers: abstract and complex representations

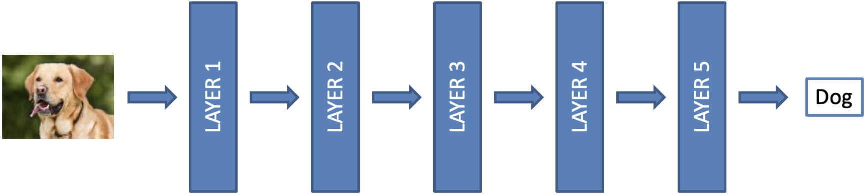
Convolutional neural networks

Suitable for 2D/3D data analysis

Network layers apply image processing filtering **convolutions**)



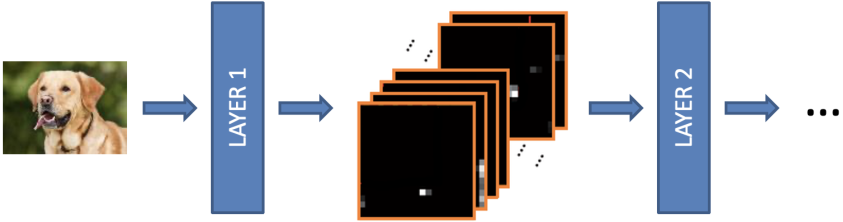
Convolutional neural networks



Convolutional neural networks

Representation

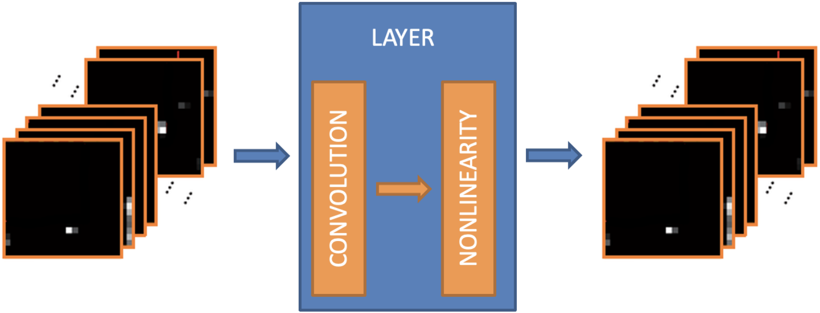
Each layer extracts a set of **feature maps**



Convolutional neural networks

Minimal layer configuration

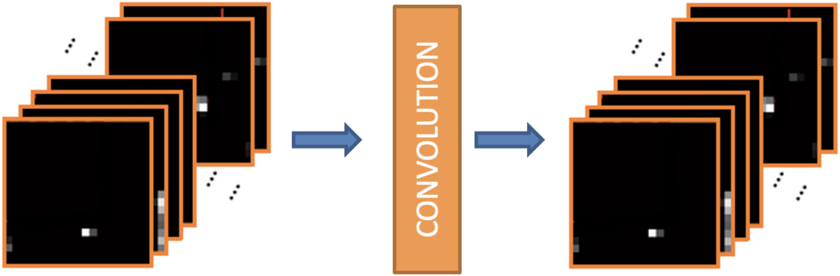
- Convolution operator
- Nonlinear activation



Convolutional neural networks

Convolution

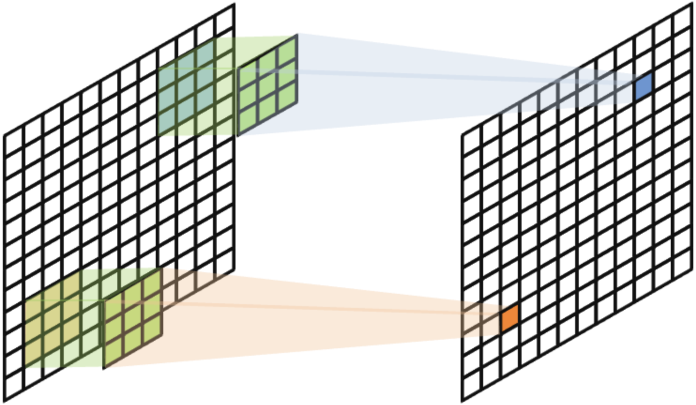
- Each feature map emphasizes specific visual characteristics
- Computed by aggregating information from previous layers



Convolutional neural networks

Convolution

- Each "neuron" is computed by filtering a window from the previous feature maps using a **kernel** matrix



Convolutional neural networks

Main parameters

- **Kernel size**
  - Larger size $\rightarrow$ more context, but more parameters
  - Must be tuned to input data
- **Stride**
  - Stride 1: process every pixel
  - Stride N: process one pixel every N
  - Suitable at high resolution

Convolutional neural networks

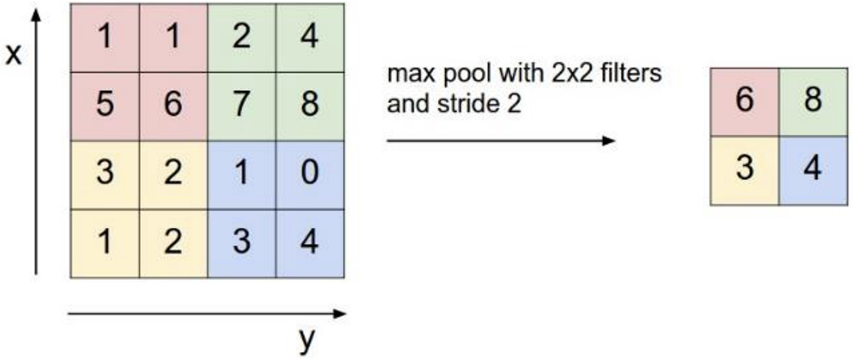
Feature map resolution

- Deep layers encode more and more aggregate information
- No need to keep original resolution

Convolutional neural networks

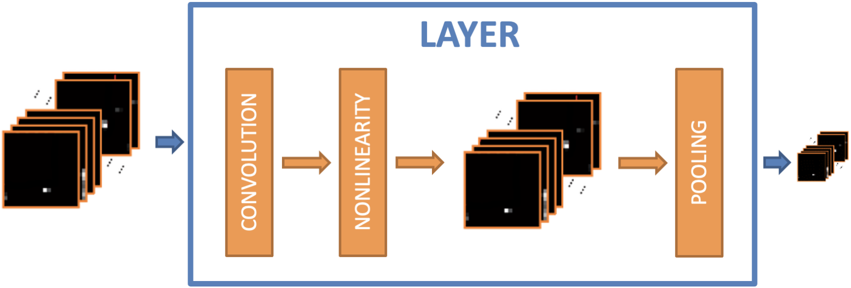
Feature map resolution

- Deep layers encode more and more aggregate information
- No need to keep original resolution
- **Max pooling**



Convolutional neural networks

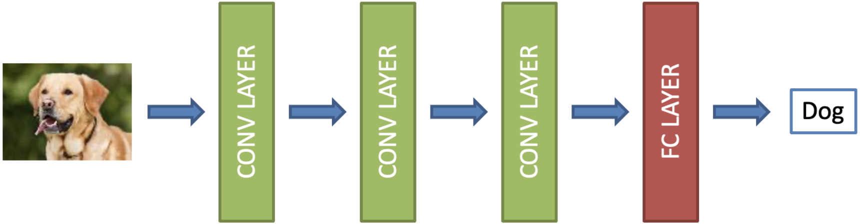
Basic layer implementation



Convolutional neural networks

CNN architecture

- Initial **feature extraction** stage
- Final **fully-connected** stage (traditional MLP)
- Task-specific output



Improve CNN training

Improve CNN training

Dropout

- At training time, randomly disable neurons in fully-connected layers
- Reduces model complexity, help preventing overfitting
- At test time, use full model

Improve CNN training

Dropout

- At training time, randomly disable neurons in fully-connected layers
- Reduces model complexity, help preventing overfitting
- At test time, use full model

Batch normalization

- Feed **batches** of samples at a time
- At each layer, standardize feature maps based on batch statistics
- Reduces variations of intermediate data distributions

Improve CNN training

Dropout

- At training time, randomly disable neurons in fully-connected layers
- Reduces model complexity, help preventing overfitting
- At test time, use full model

Batch normalization

- Feed **batches** of samples at a time
- At each layer, standardize feature maps based on batch statistics
- Reduces variations of intermediate data distributions

Data augmentation

- Create "variations" of input samples
- Increase dataset variability, help preventing overfitting
- Examples: random crop, flip, color jitter

Convolutional neural networks

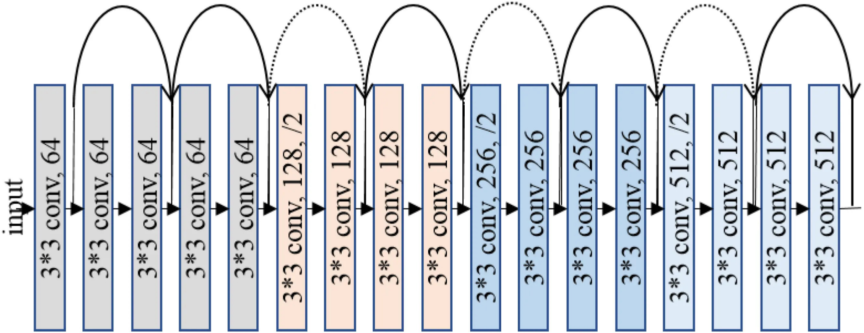
Pretrained models

- Adapt public trained models, rather than train from scratch

Convolutional neural networks

Pretrained models

- Adapt public trained models, rather than train from scratch
- Example: **ResNet** models



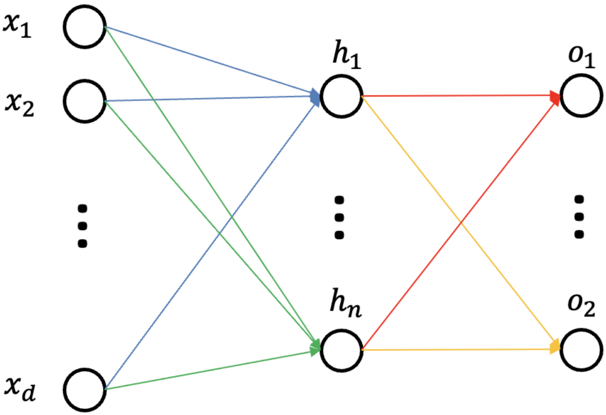
Recurrent neural networks

How to process sequential data?

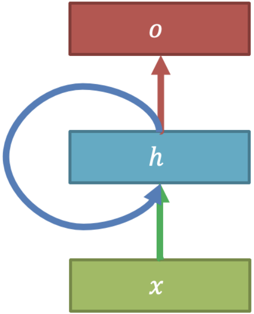
- Video
- Text
- Audio
- Time series

Recurrent neural networks

Feed-forward neural network



Recurrent neural networks



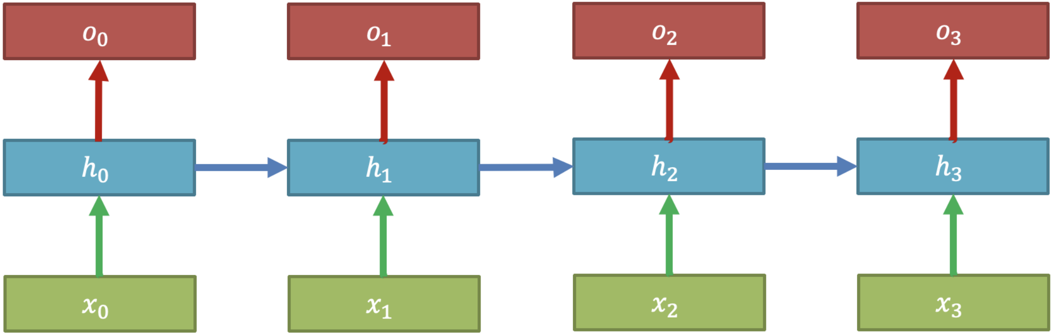
Output layer (e.g. classification)

Recurrent layer (nonlinear)

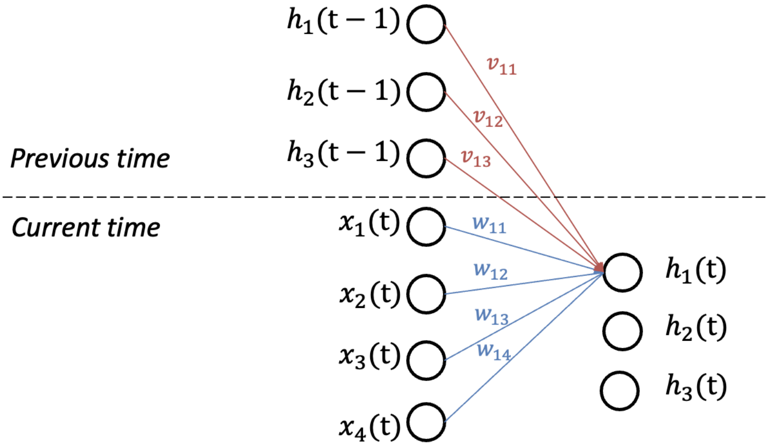
Input layer

Recurrent neural networks

"Unrolled" visualization



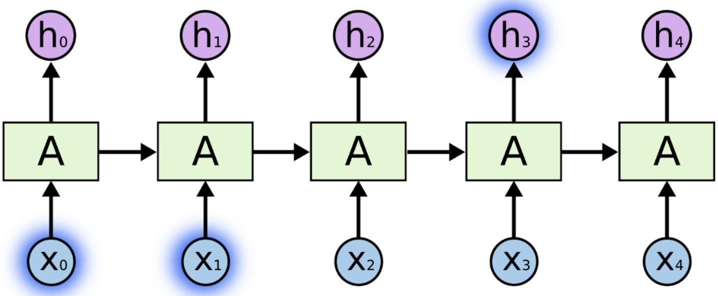
Recurrent neural networks



Recurrent neural networks

Long-term dependencies

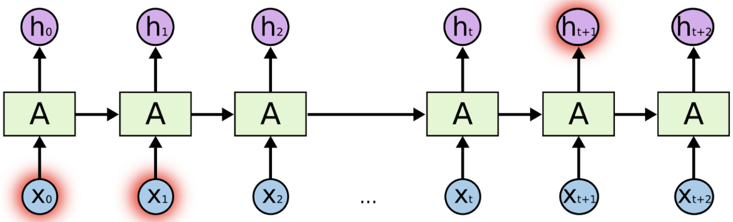
- RNNs store a representation of **context**
- Based on past information, make predictions on the future



Recurrent neural networks

Long-term dependencies

- **Recent** history affects state more than old history
- Old samples have little impact on future predictions



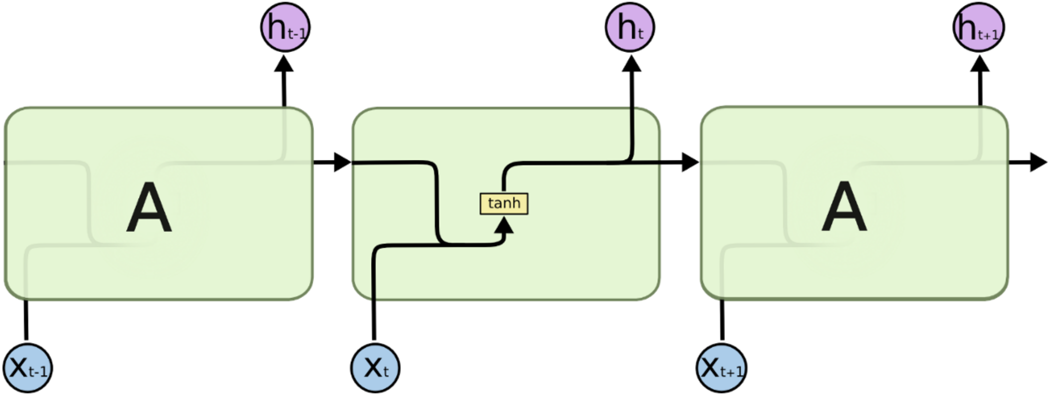
LSTM

Long short-term memory

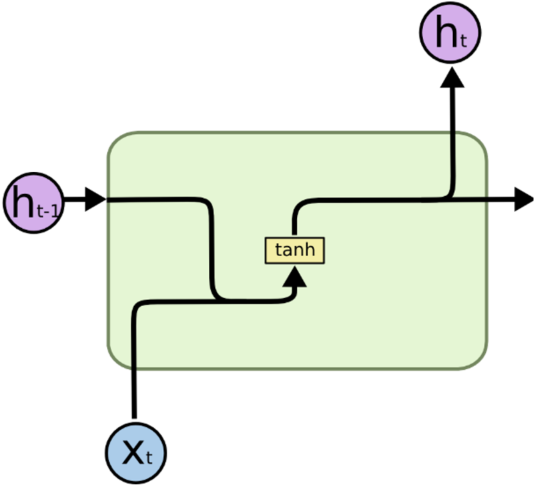
- Introduce a **memory** mechanism in the cell
- **Long**: memory enables to retain context information of a long time
- **Short-term**: stored information are dynamically selected based on the current input
- **Gates** control information flow
- Technical insight: improve backpropagation by preventing decreases in loss gradients

LSTM

Standard RNN layer

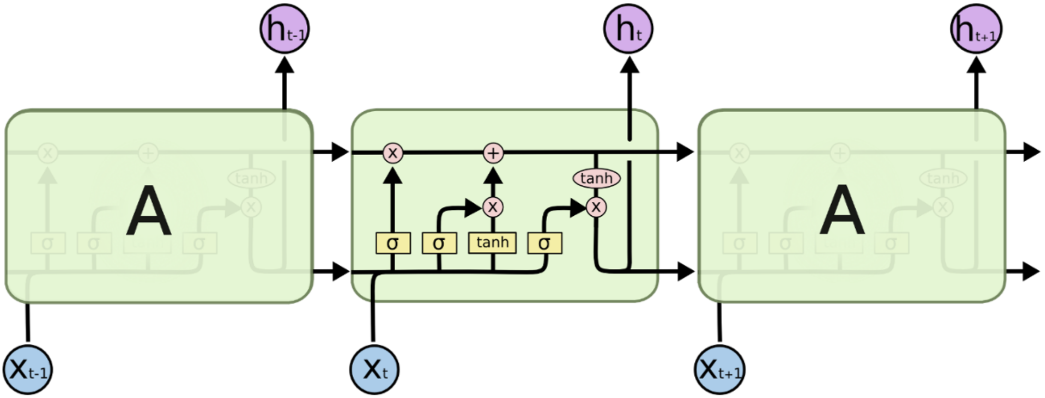


LSTM

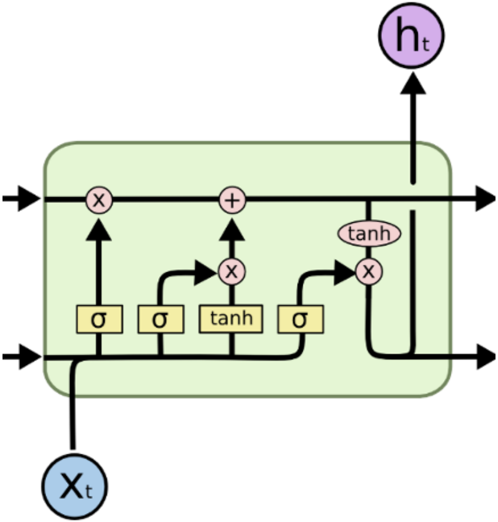


LSTM

LSTM layer



LSTM



LSTM

Long short-term memory

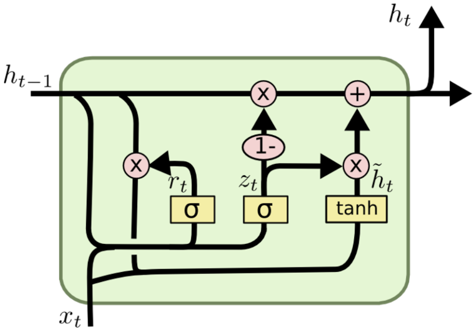
The cell internally controls:

- when information should be stored in memory
- when information should be removed
- when information should be provided as output

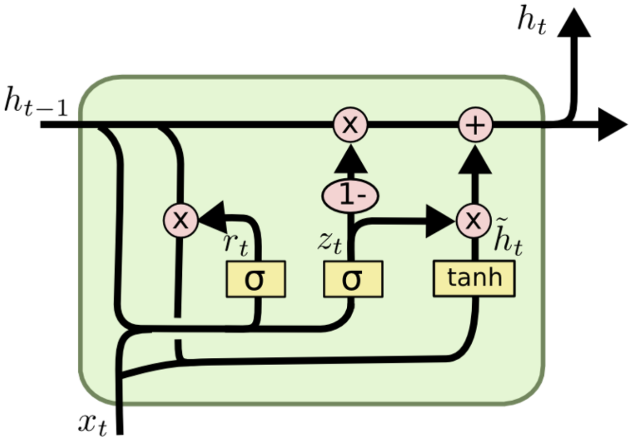
GRU

Gated recurrent unit

- Single cell state/output
- Single input/forget gate $\rightarrow$ **update** gate



GRU



Conclusions

Conclusions

Recap

- Fundamental ML concepts
- Common deep architectures

Conclusions

Recap

- Fundamental ML concepts
- Common deep architectures

New frontiers in astrophysics

- **Scalability:** tackle massive datasets (e.g., LSST, SKA).
- **Knowledge discovery:** identify subtle patterns missed by traditional methods.
- **Surrogate modeling:** accelerate computationally expensive simulations.

Conclusions

Recap

- Fundamental ML concepts
- Common deep architectures

New frontiers in astrophysics

- **Scalability:** tackle massive datasets (e.g., LSST, SKA).
- **Knowledge discovery:** identify subtle patterns missed by traditional methods.
- **Surrogate modeling:** accelerate computationally expensive simulations.

Challenges

- **Interpretability:** uncover the "black box"
- **Data scarcity/bias:** handling rare phenomena and observational biases.
- **Computational resources:** training large models can be demanding.