Introduction to Machine Learning: Deep Learning

Stefania Sarno

January 31, 2024

Outline

1. Convolutional Neural Networks (CNNs)
2. Autoencoders
3. Recurrent Neural Networks (RNNs)
4. Long-Short Term Memory (LSTM)

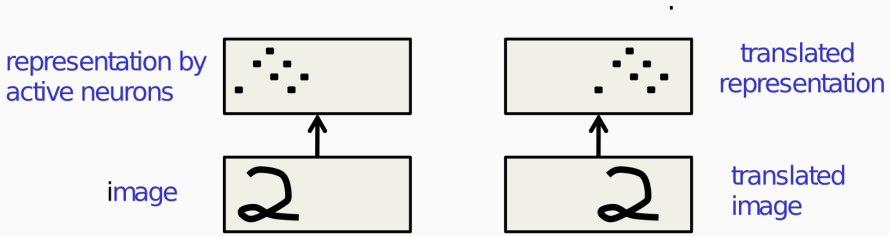
Convolutional Neural Networks

What are CNNs?

- CNNs are a class of deep neural network that represents the deep learning framework for computer vision.
- They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on shared-weights architecture and translation invariance characteristics.
- They are powerful tools for image segmentation, object detection, object recognition

Why Object Recognition is difficult?

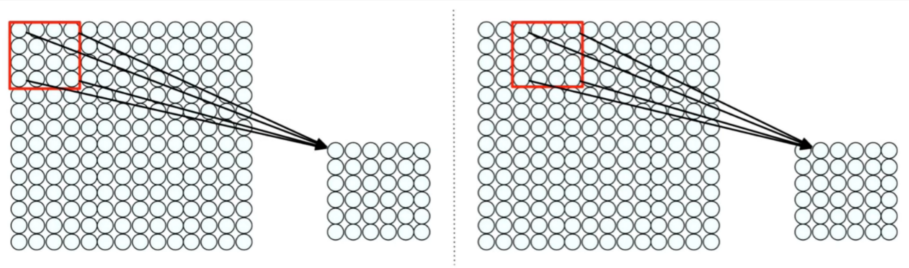
- Segmentation, Lighting, Deformation, Affordances
- Viewpoint (translation): changing the viewpoint would completely change the activation of the input layer in a multilayer perceptron



- Fully connected networks do not have spatial information and have plenty of parameters

How can we preserve spatial information?

Connect patches of inputs to neurons in hidden layers



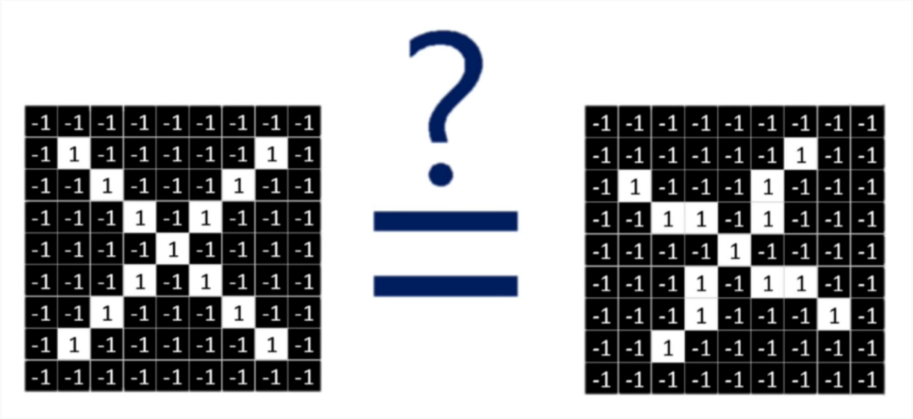
We want to have patches that are able to detect particular aspects of the image

Feature Extraction with Convolution

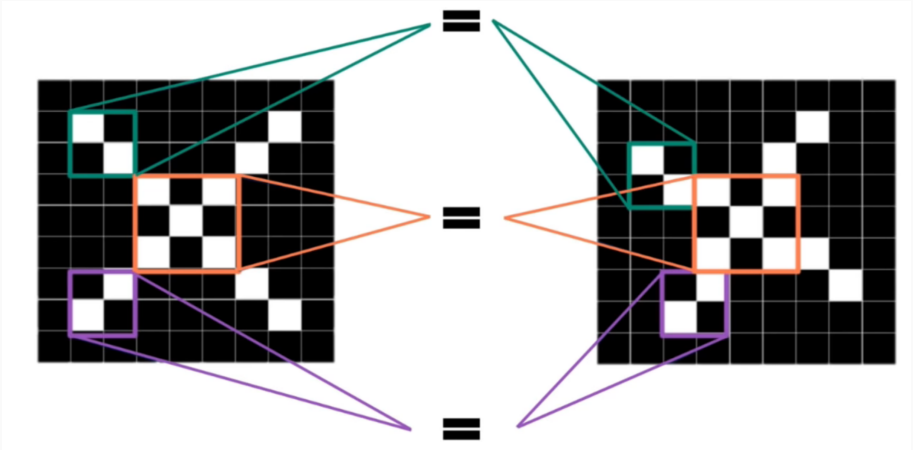
We want the following

1. Apply a filter to extract spatial information
2. Apply different filters in each convolutional layer
3. Learn each filter using shared parameters

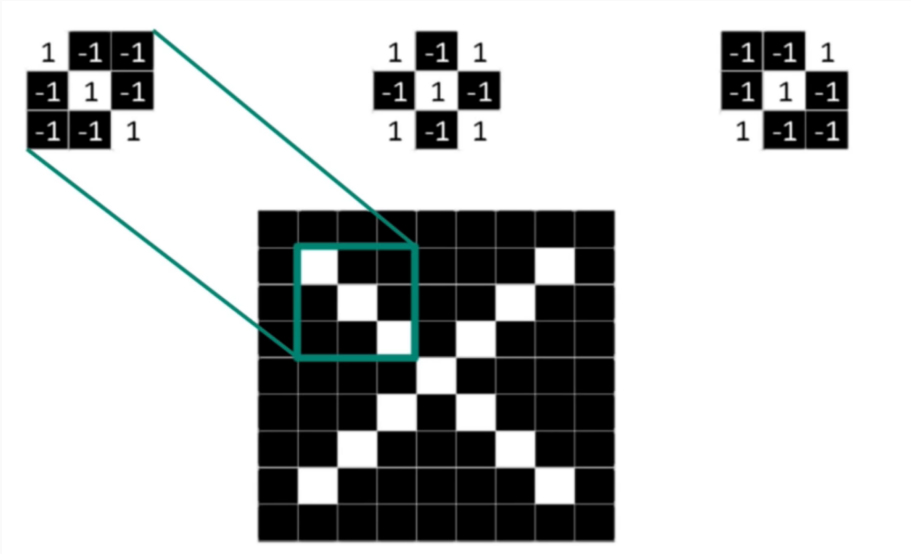
Relevant Features of a X



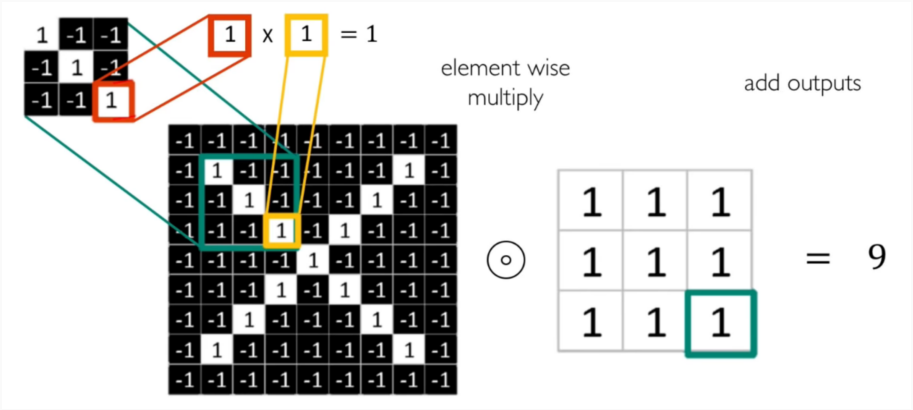
Relevant Features of a X



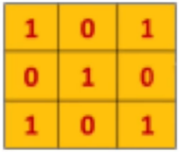
Relevant Features of a X



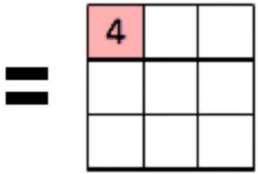
The Convolution Operation



The Convolution Operation

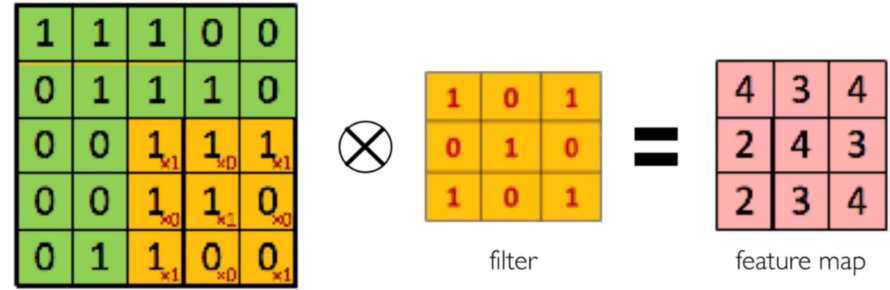


filter



feature map

The Convolution Operation

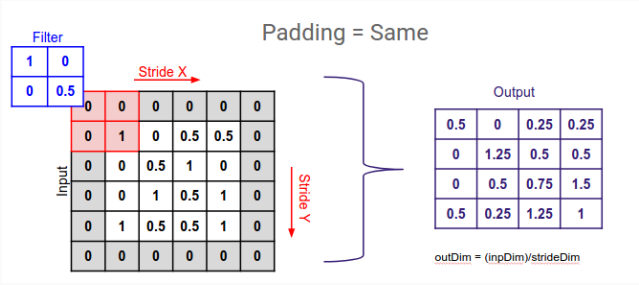


Padding before convolution

After convolution the dimension of the output O image (either height or width is):

$$O = \frac{W - K + 2P}{S} + 1 \tag{1}$$

where W = input dimension, K = filter size, P = padding, S = stride.



Without padding

1. Shrinking outputs
2. Loosing information on corners of the image

The Feature Maps

Original

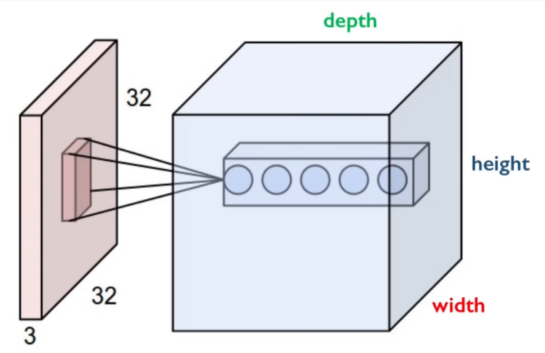
Sharpen

Edge Detect

"Strong" Edge Detect

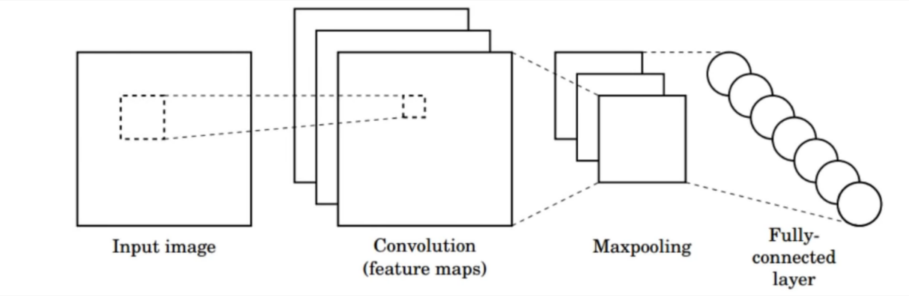
Spatial Arrangement of the Output Volume

- width and height depend on the input volume and on the filter size
- depth represents the number of weights



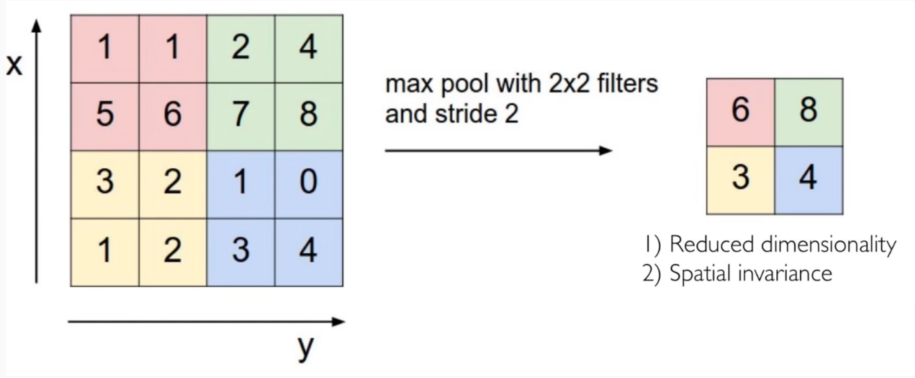
CNN for Classification

1. Multiple Convolutional Layers:
   - Apply a convolutional filter
   - Apply a non-linear function (ReLU)
   - Perform Maxpooling
2. Multiple Fullyconnected Layers
3. A softmax output

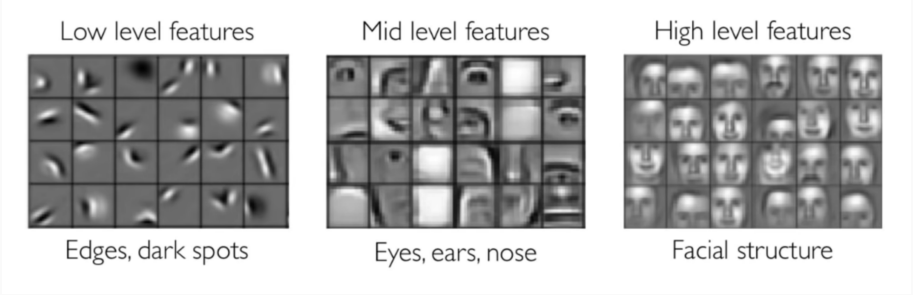


Pooling

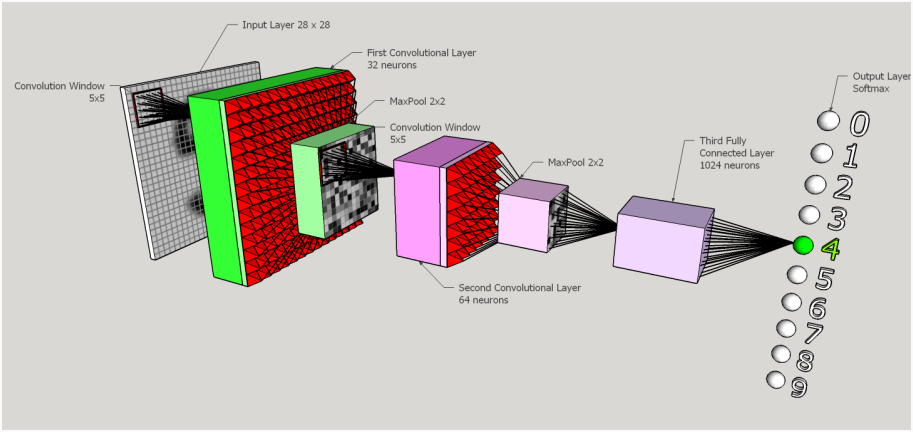
- Reduce dimensionality
- Preserve spatial invariance



Feature Map in CNNs



A CNN for the MNIST dataset



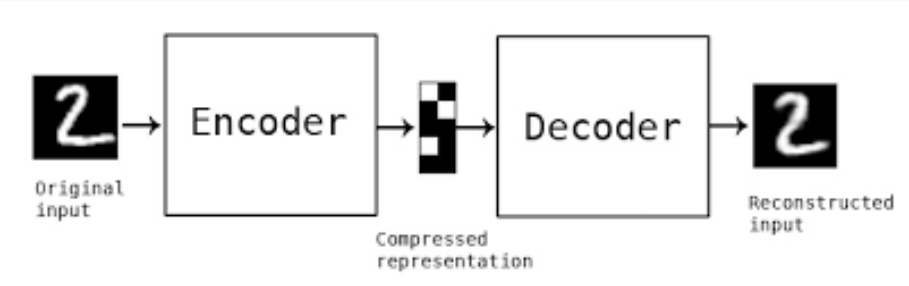
Autoencoders

What is an autoencoder?

An autoencoder is a type of artificial neural network used to learn data encodings in an unsupervised manner:

- They learn a lower-dimensional representation (encoding) for a higher-dimensional data
- They are often used for dimensionality reduction
- They consist of a network trained to capture the most important parts of the input.

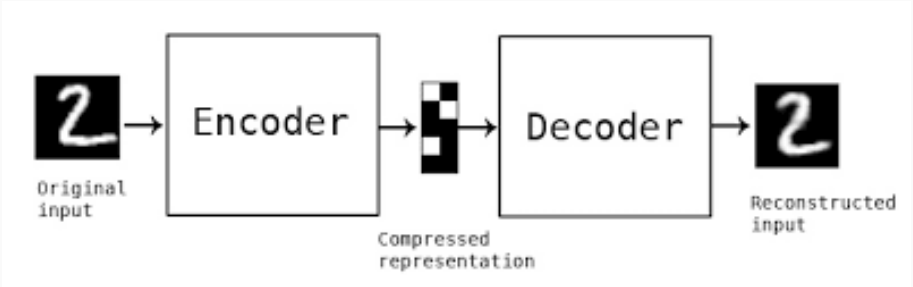
Typical architecture of autoencodes



Autoencoders consist of 3 parts:

1. Encoder: module that compresses the input data into an encoded representation (typically several orders of magnitude smaller than the input data).
2. Bottleneck: module that contains the compressed knowledge representations and is therefore the most important part of the network.
3. Decoder: module that helps the network to “decompress” the knowledge representations and reconstructs the data back from its encoded form.

Undercomplete autoencoders



Undercomplete autoencoders

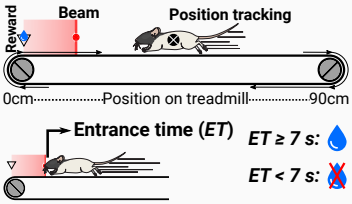
- Reconstruction loss function for input $\mathbf{x}$ and output $\hat{\mathbf{x}}$:

$$L = MSE(\mathbf{x}, \hat{\mathbf{x}})$$

- They are truly unsupervised as they do not take any form of label
- The loss function has no explicit regularisation term, good fit depends on using appropriate network architecture (number of hidden layers, size of the bottleneck)

Example: treadmill task

The treadmill moves backward and rats can approach the reward area after 7s:

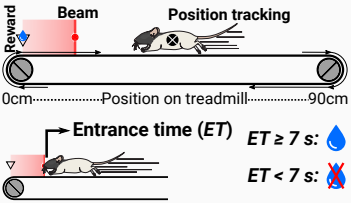


- Early entrance: no reward, punishment (increased running time)
- Correct entrance: reward, no punishment

[Rueda-Orozco and Robbe 2015, Safaie et al 2020]

Example: treadmill task

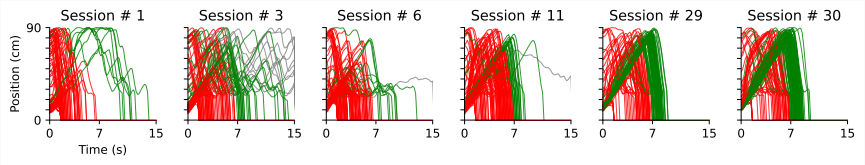
The treadmill moves backward and rats can approach the reward area after 7s:



- Early entrance: no reward, punishment (increased running time)
- Correct entrance: reward, no punishment

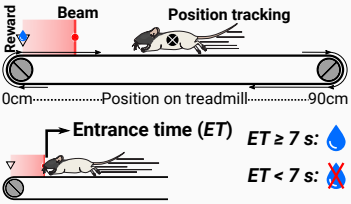
[Rueda-Orozco and Robbe 2015, Safaie et al 2020]

Most of the rats develop a "wait-run" routine



Example: treadmill task

The treadmill moves backward and rats can approach the reward area after 7s:

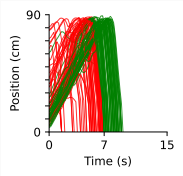


- Early entrance: no reward, punishment (increased running time)
- Correct entrance: reward, no punishment

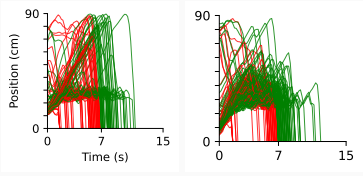
[Rueda-Orozco and Robbe 2015, Safaie et al 2020]

Not all rats converge to the same stereotyped strategy:

Stereotyped

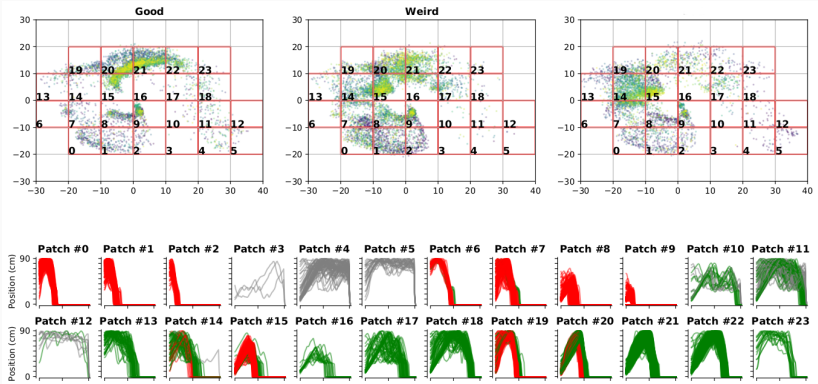


Weird



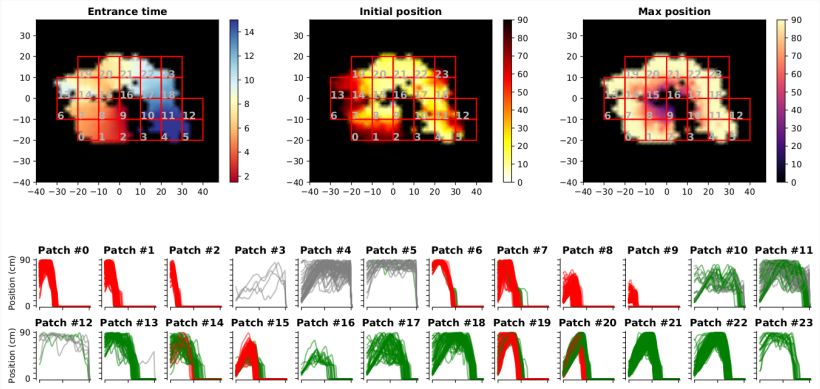
Example: treadmill task

We use an autoencoder to visualize trajectories of different rats in 2D

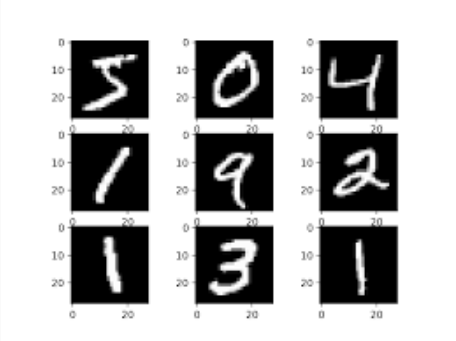


Example: treadmill task

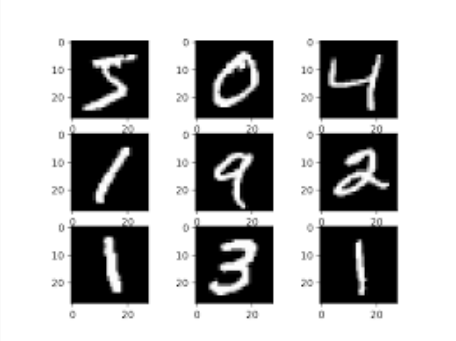
We can visualize how relevant features of the trajectories are mapped in 2D

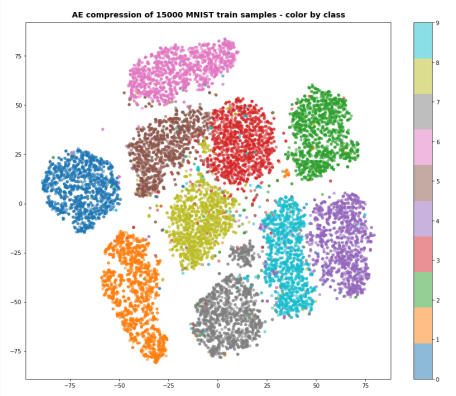


Example: MNIST

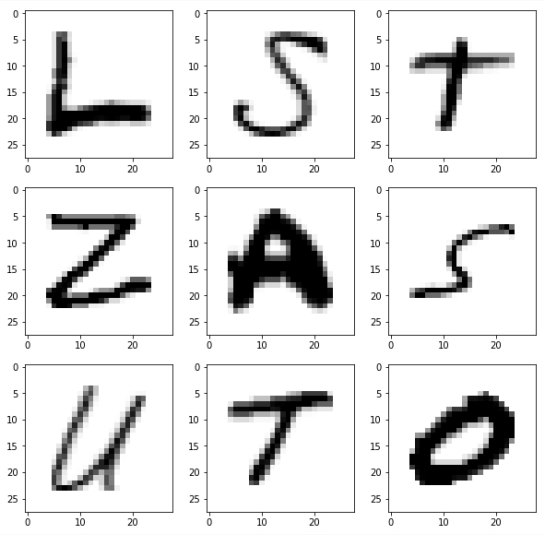


Example: MNIST





Project: autoencoder for hand-written characters

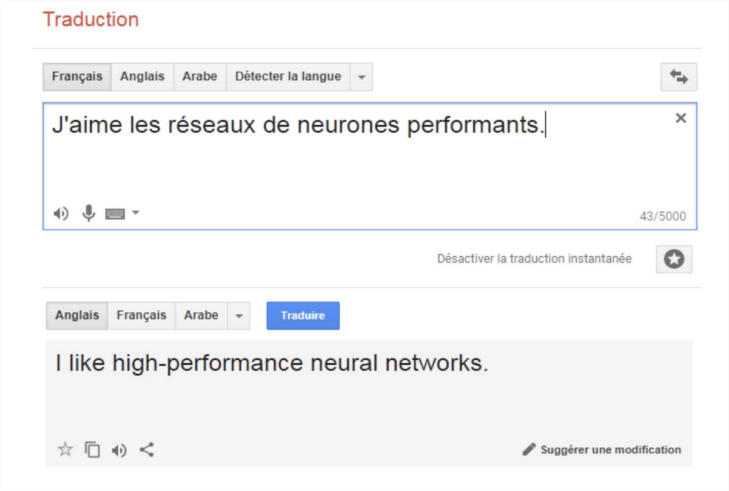


Recurrent Neural Networks

Why do we need RNNs?

To model sequential data (language model).

- Machine translation (text to text)



Why do we need RNNs?

To model sequential data (language model).

- Image captioning (image to text)

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

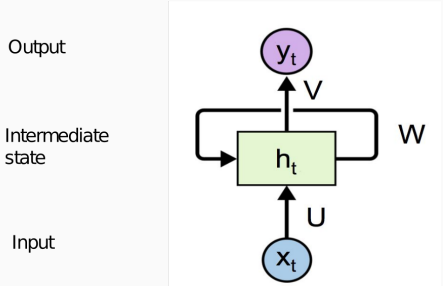
"young girl in pink shirt is swinging on swing."

What are RNNs?

Vanilla Recurrent Networks

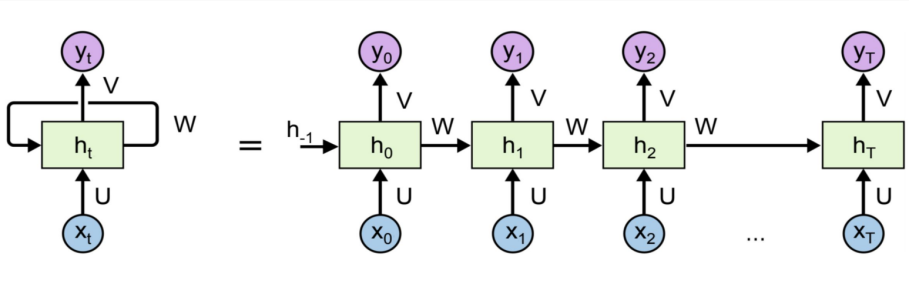
- The network is define by three sets of parameters: U, W, V
- The parameters are shared across time

$$h_t = \tanh(Ux_t + Wh_{t-1})$$
$$y_t = f(Vh_t)$$



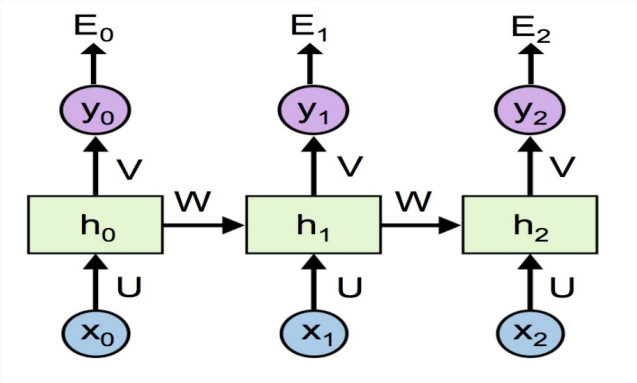
The unrolled network

Vanilla Recurrent Networks



Training RNNs

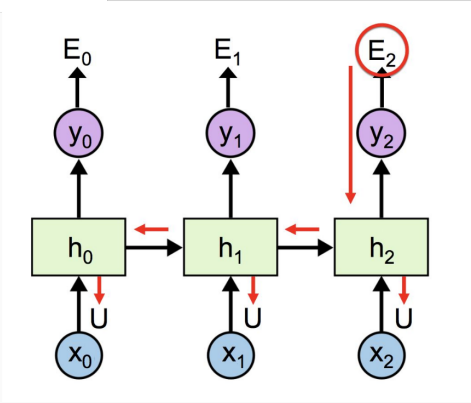
- The loss is: $E = \sum_{t=0}^{T} E_t$
- Do normal backpropagation using the loss E



Backpropagation through time (BPTT)

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2}\left(x_2^T + \frac{\partial h_2}{\partial h_1}\left(x_1^T + \frac{\partial h_1}{\partial h_0}x_0^T\right)\right)$$

mage from Christopher Olah's blog



Vanishing/Exploding gradient problem

$$\mathbf{h}_t = \boldsymbol{\theta}\boldsymbol{\phi}(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$
$$\mathbf{y}_t = \boldsymbol{\theta}_y \boldsymbol{\phi}(\mathbf{h}_t)$$

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial E_t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}}$$

[Yoshua Bengio et al]

Vanishing/Exploding gradient problem

$$\frac{\partial E_t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \boldsymbol{\theta}^T \mathrm{diag}[\boldsymbol{\phi}'(\mathbf{h}_{i-1})]$$

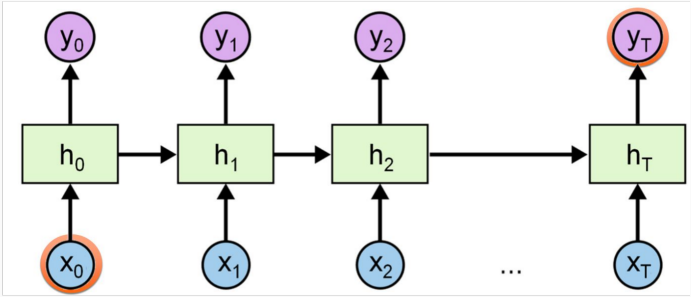
$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\boldsymbol{\theta}^T\| \|\mathrm{diag}[\boldsymbol{\phi}'(\mathbf{h}_{i-1})]\| \leq \gamma_\theta \gamma_\phi$$

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| \leq (\gamma_\theta \gamma_\phi)^{t-k}$$

The gradient problem is problematic

Long term dependencies

Who went to Paris?

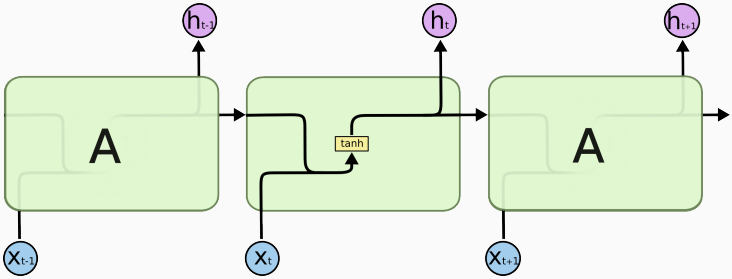


Vanishing/Exploding gradient problem

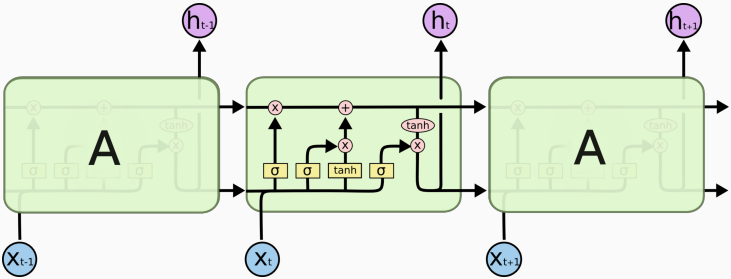
- One solution to this problem to deal with the vanishing gradient problem was LSTM
- It provides a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory

Long Short-Term Memory Cells

Long short-term memory (LSTM)

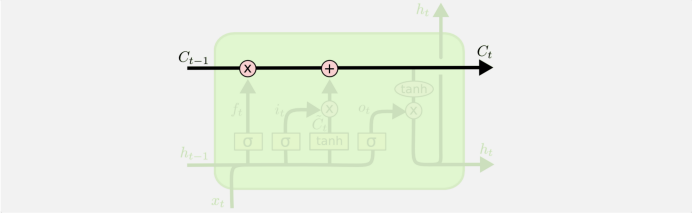


Long short-term memory (LSTM)



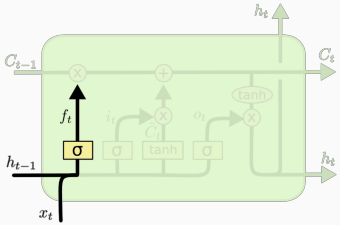
Long short-term memory (LSTM)

Introduce a cell state



Long short-term memory (LSTM)

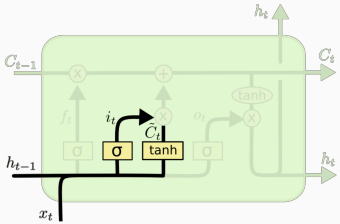
Introduce a forget gate (how much to keep in the cell state)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Long short-term memory (LSTM)

Introduce an input gate (how much of the input is added to the cell state)

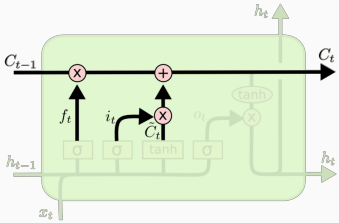


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Long short-term memory (LSTM)

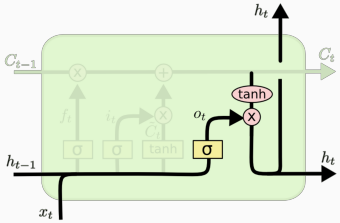
Update the cell state



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Long short-term memory (LSTM)

Produce an output



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution
- Attention mechanism like transformers has revolutionized the NLP landscape

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution
- Attention mechanism like transformers has revolutionized the NLP landscape
- Both BERT (Bidirectional Encoder Representations from Transformers), large language models (LLMs) are typically built with a transformer-based architecture

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution
- Attention mechanism like transformers has revolutionized the NLP landscape
- Both BERT (Bidirectional Encoder Representations from Transformers), large language models (LLMs) are typically built with a transformer-based architecture
- Despite the emergence of transformers the RNN/LSTM architecture still serves as an exceptional baseline for comparison and experimentation.

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution
- Attention mechanism like transformers has revolutionized the NLP landscape
- Both BERT (Bidirectional Encoder Representations from Transformers), large language models (LLMs) are typically built with a transformer-based architecture
- Despite the emergence of transformers the RNN/LSTM architecture still serves as an exceptional baseline for comparison and experimentation.
- Significant advantages of LSTM are computational efficiency and light model footprint

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution
- Attention mechanism like transformers has revolutionized the NLP landscape
- Both BERT (Bidirectional Encoder Representations from Transformers), large language models (LLMs) are typically built with a transformer-based architecture
- Despite the emergence of transformers the RNN/LSTM architecture still serves as an exceptional baseline for comparison and experimentation.
- Significant advantages of LSTM are computational efficiency and light model footprint
- More sophisticated algorithms are difficult to fine tune and more prone to overfitting when the size of the training data is small

Long short-term memory (LSTM)

- In recent years, the field of natural language processing (NLP) has witnessed a remarkable revolution
- Attention mechanism like transformers has revolutionized the NLP landscape
- Both BERT (Bidirectional Encoder Representations from Transformers), large language models (LLMs) are typically built with a transformer-based architecture
- Despite the emergence of transformers the RNN/LSTM architecture still serves as an exceptional baseline for comparison and experimentation.
- Significant advantages of LSTM are computational efficiency and light model footprint
- More sophisticated algorithms are difficult to fine tune and more prone to overfitting when the size of the training data is small
- Indeed LSTMs are still largely used in time series forecasting