University of Ljubljana, Faculty of Computer and Information Science

Recurrent and convolutional neural networks for text

Prof Dr Marko Robnik-Šikonja

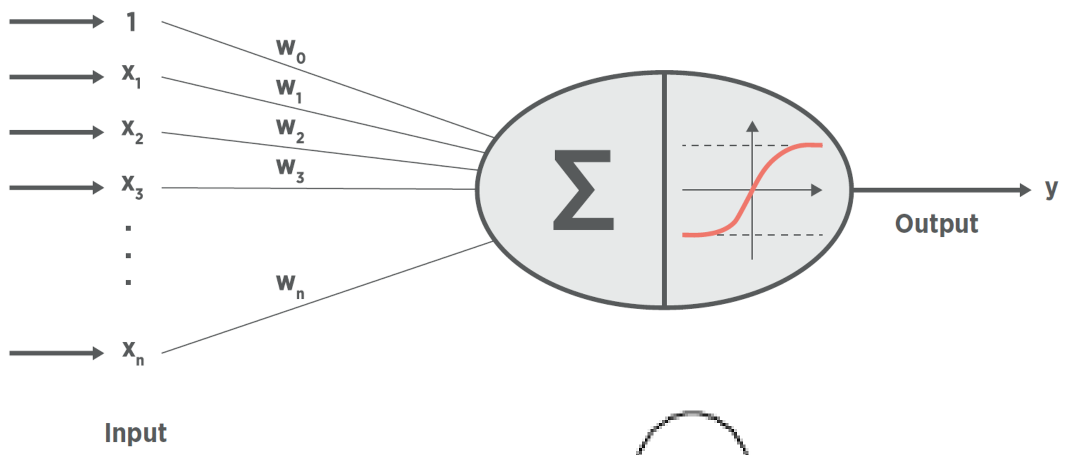
Natural Language Processing, Edition 2025

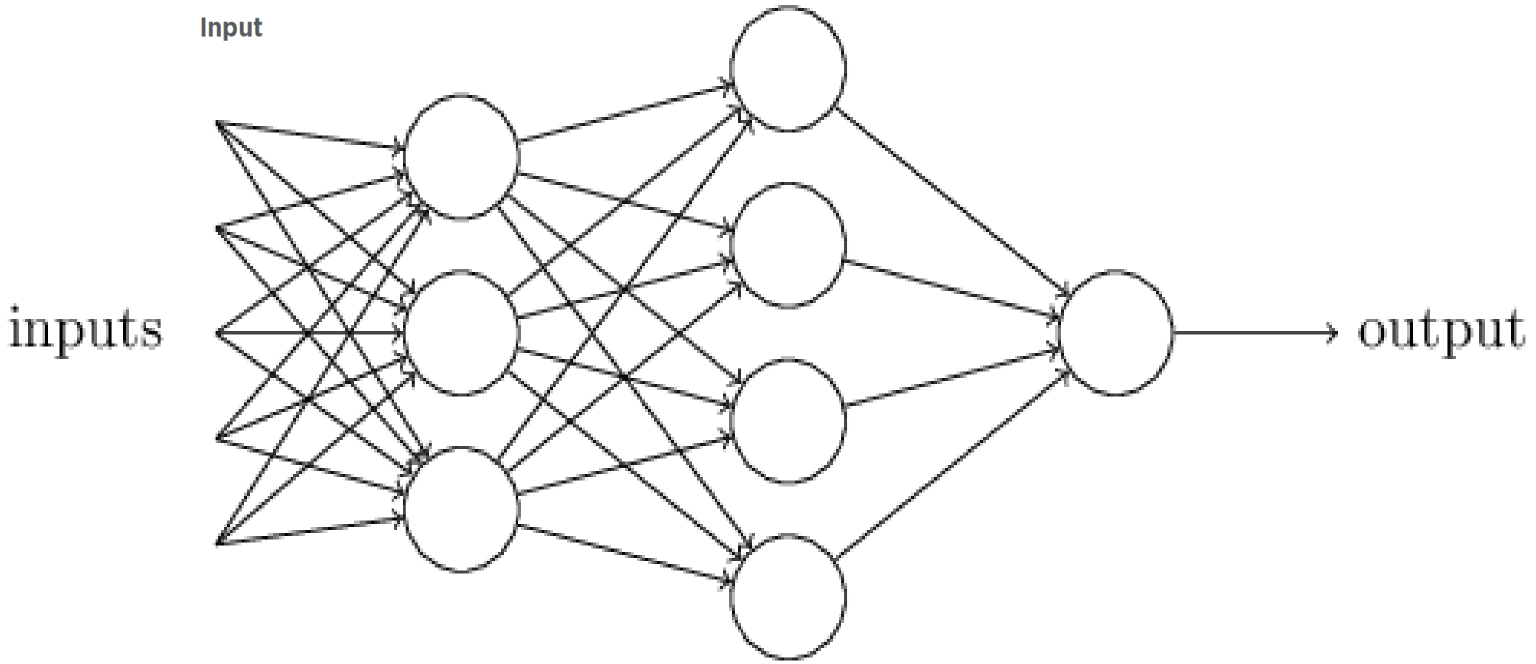
Contents

- recurrent neural networks
- LSTM networks
- attention mechanism
- convolutional neural networks

read Chapter 9 in Jurafsky & Martin, 3rd edition

Revision: artificial neural networks

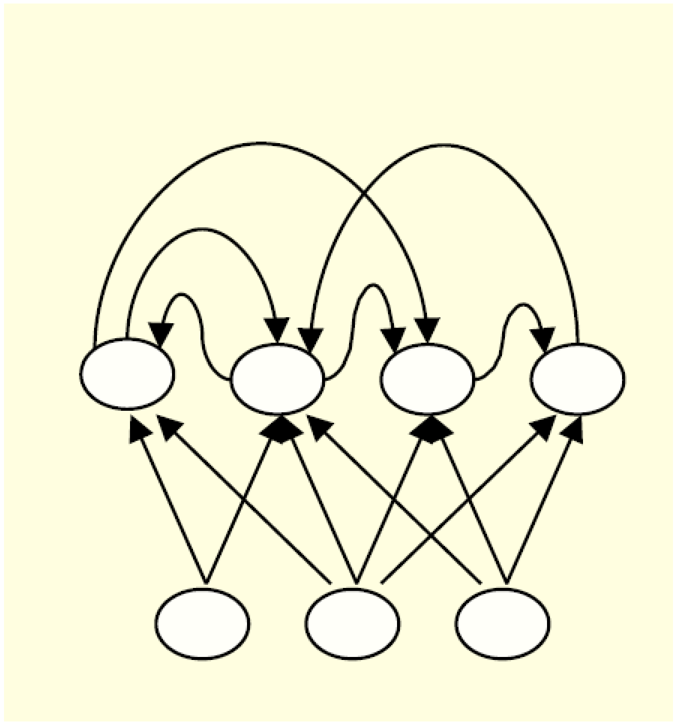




besides feed-forward networks, there are many other more complex network architectures

Recurrent network (RNN)

- back connections
- biologically more realistic
- store information from the past – back connections correspond to memory
- more difficult to learn



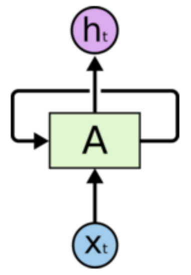
Recurrent Neural Networks

- The applications of standard Neural Networks (and also Convolutional Networks) in NLP are limited due to:
  - They only accepted a fixed-size vector as input (e.g., an image or a fixed number of words) and produce a fixed-size vector as output (e.g., probabilities of different classes).
  - These models use a fixed amount of computational steps (e.g. the number of layers in the model).

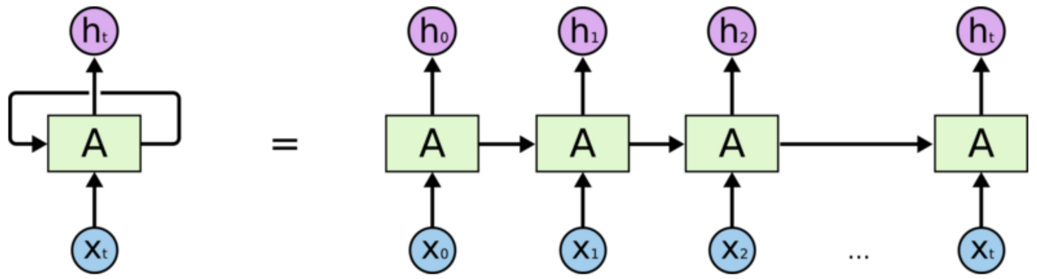
- Recurrent Neural Networks are unique as they allow us to operate over sequences of vectors.
  - Sequences in the input, the output, or in the most general case both

Recurrent Neural Networks

- Recurrent Neural Networks are networks with loops in them, allowing information to persist.



Recurrent Neural Networks have loops.



An unrolled recurrent neural network.

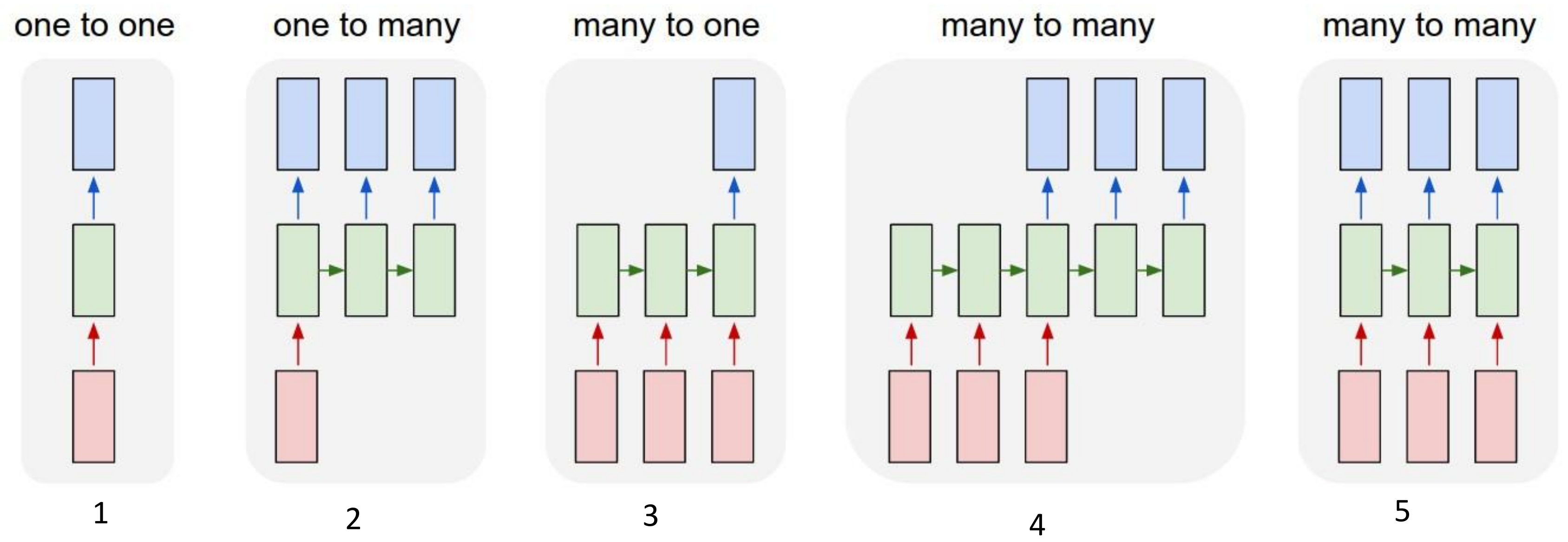
In the above diagram, a chunk of neural network, ***A***, looks at some input $\boldsymbol{x_t}$ and outputs a value $\boldsymbol{h_t}$.
A loop allows information to be passed from one step of the network to the next.

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.
The diagram above shows what happens if we unroll the loop.

Recurrent Neural Networks

- Intuition of Recurrent Neural Networks
  - Human thoughts have persistence; humans don’t start their thinking from scratch every second.
    - As you read this sentence, you understand each word based on your understanding of previous words.
  - One of the appeals of RNNs is the idea that they are able to connect previous information to the present task
    - E.g., using previous video frames to inform the understanding of the present frame.
    - E.g., a language model tries to predict the next word based on the previous ones.

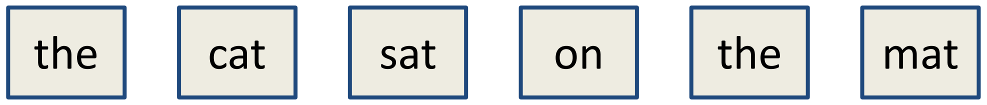
Examples of Recurrent Neural Networks



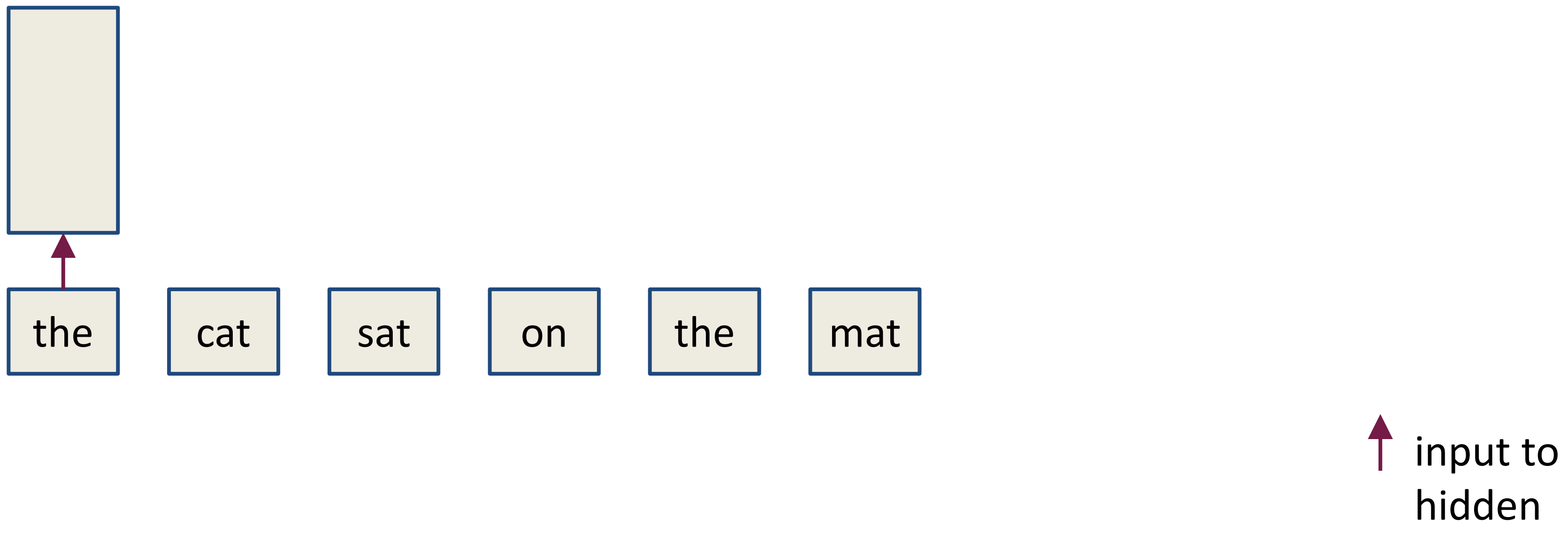
- Each rectangle is a vector and arrows represent functions (e.g. matrix multiply).
- Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state

1. Standard mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification).
2. Sequence output (e.g. image captioning takes an image and outputs a sentence of words).
3. Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).
4. Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).
5. Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

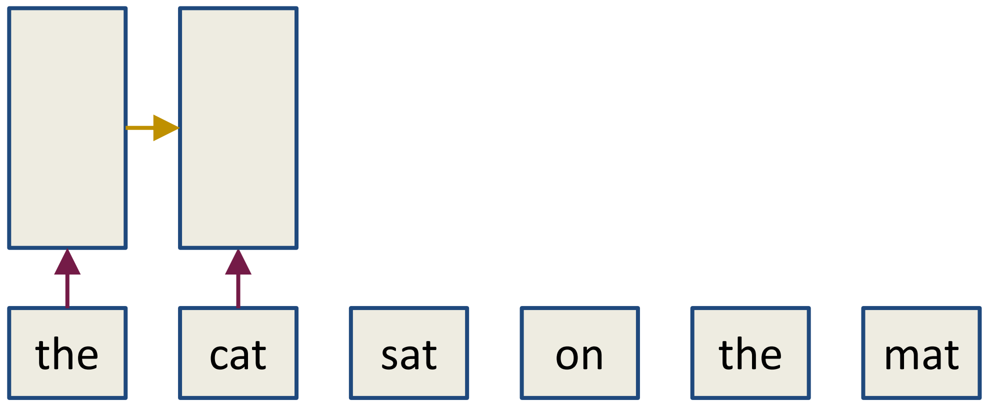
How an RNN works for text



How an RNN works



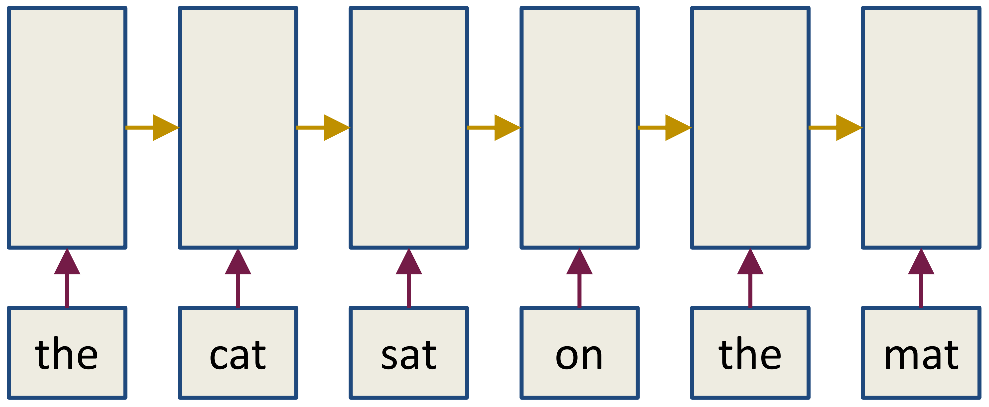
How an RNN works

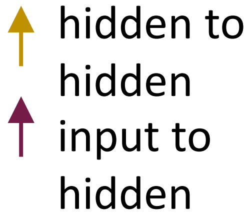


↑ hidden to hidden

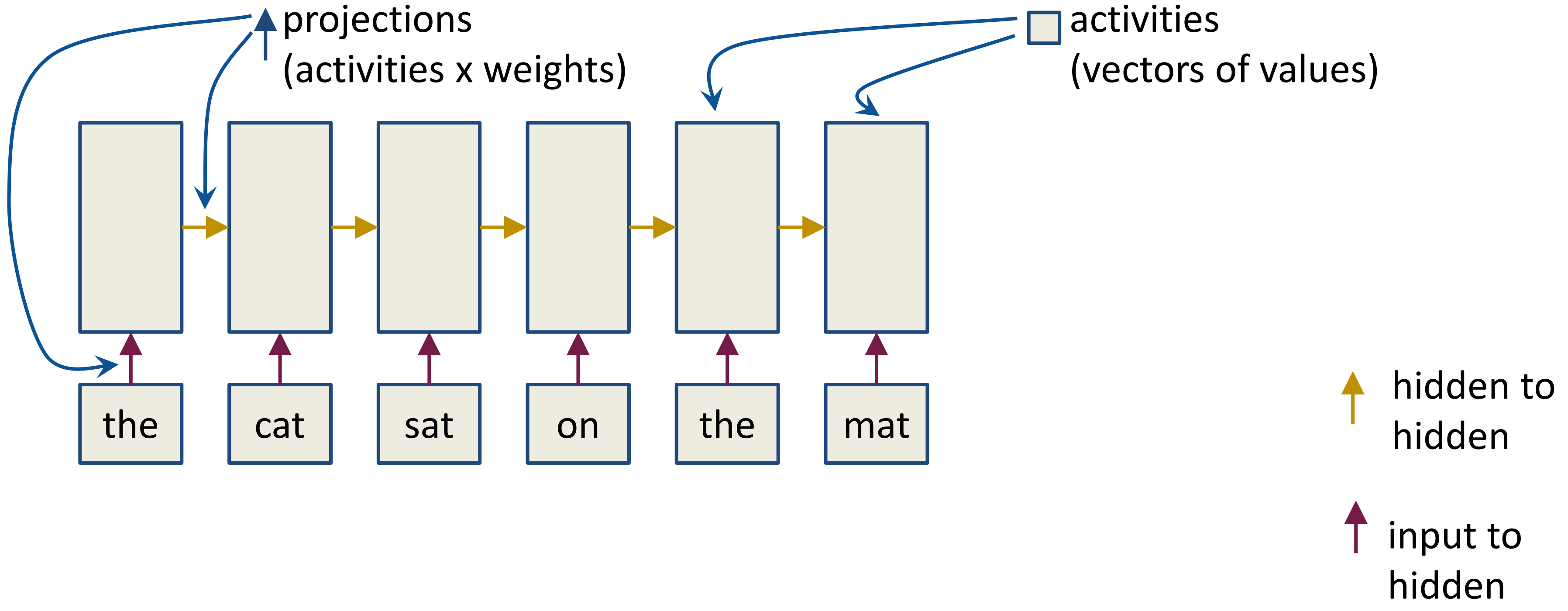
↑ input to hidden

How an RNN works

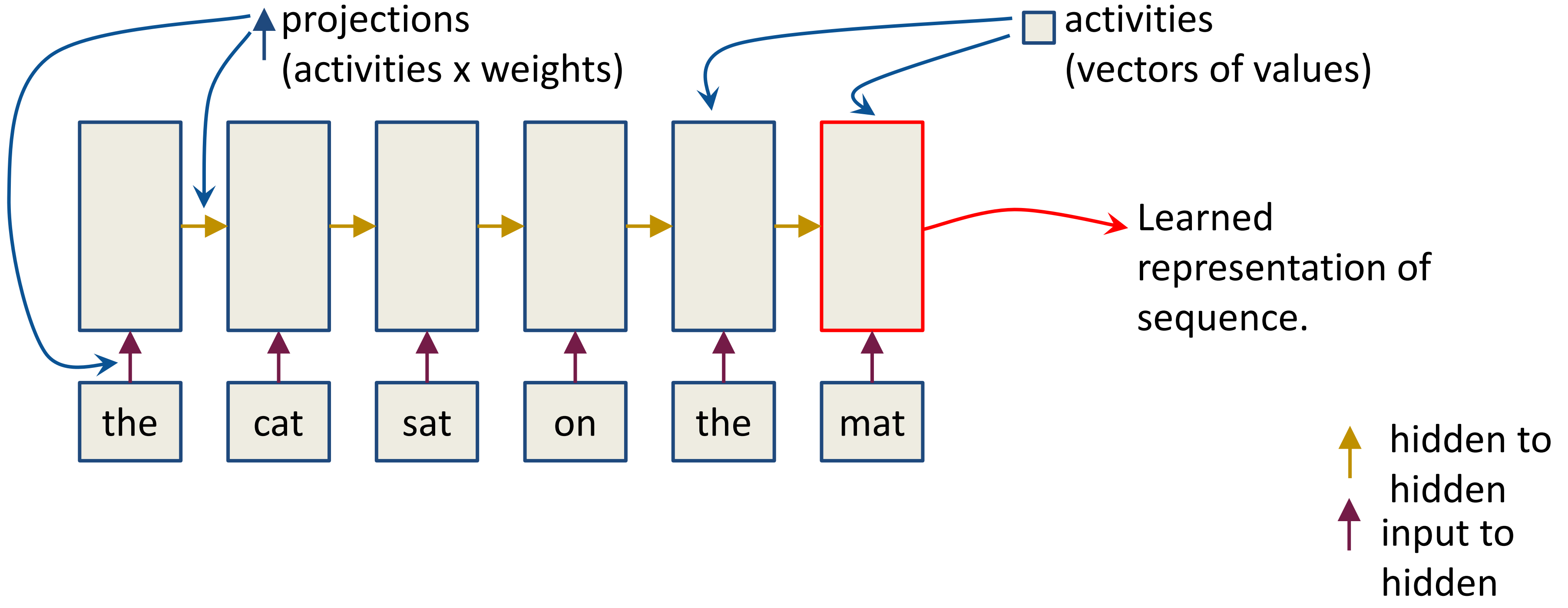




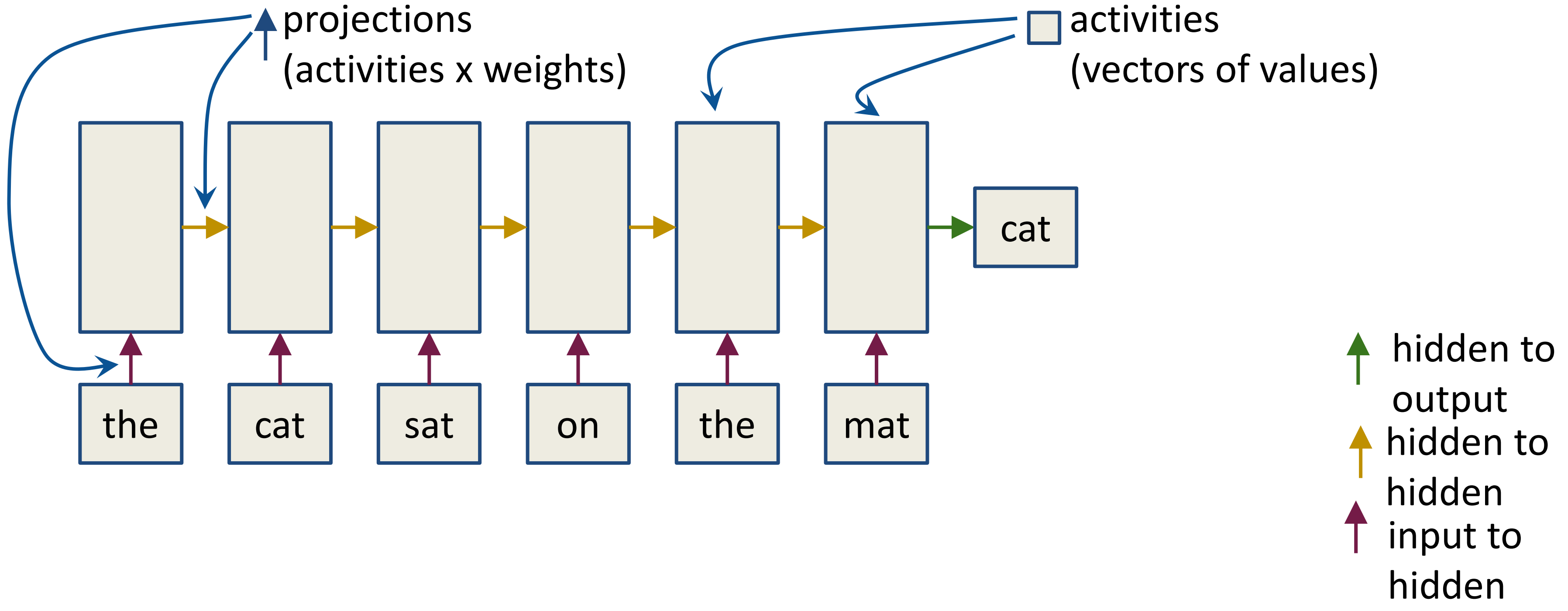
How an RNN works



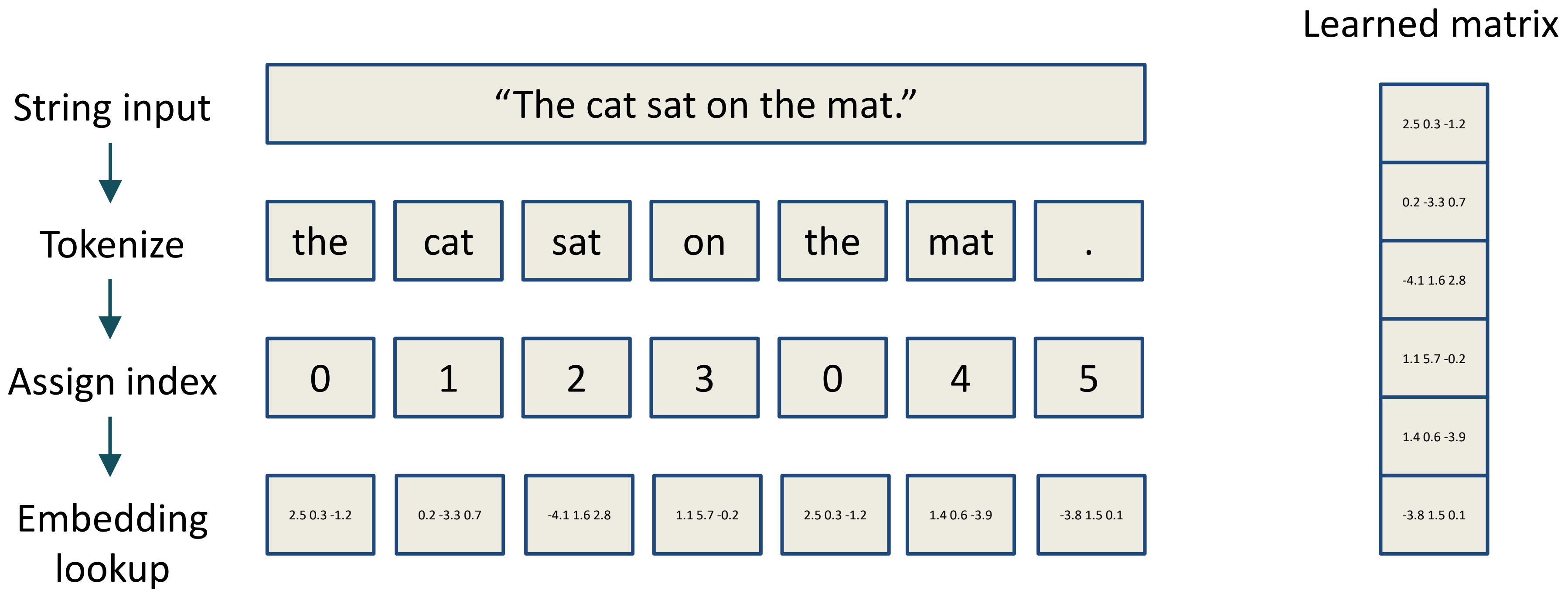
How an RNN works



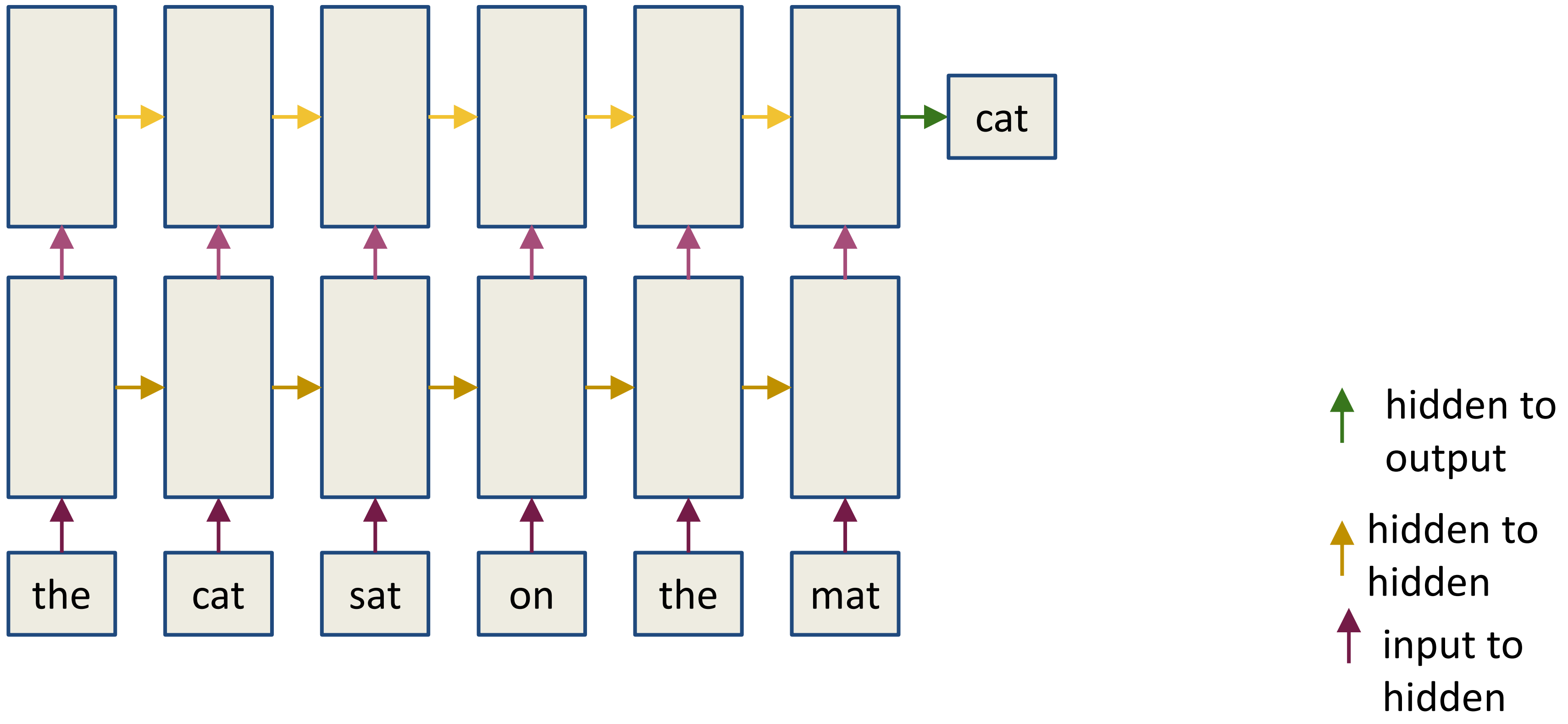
How an RNN works



From text to RNN input

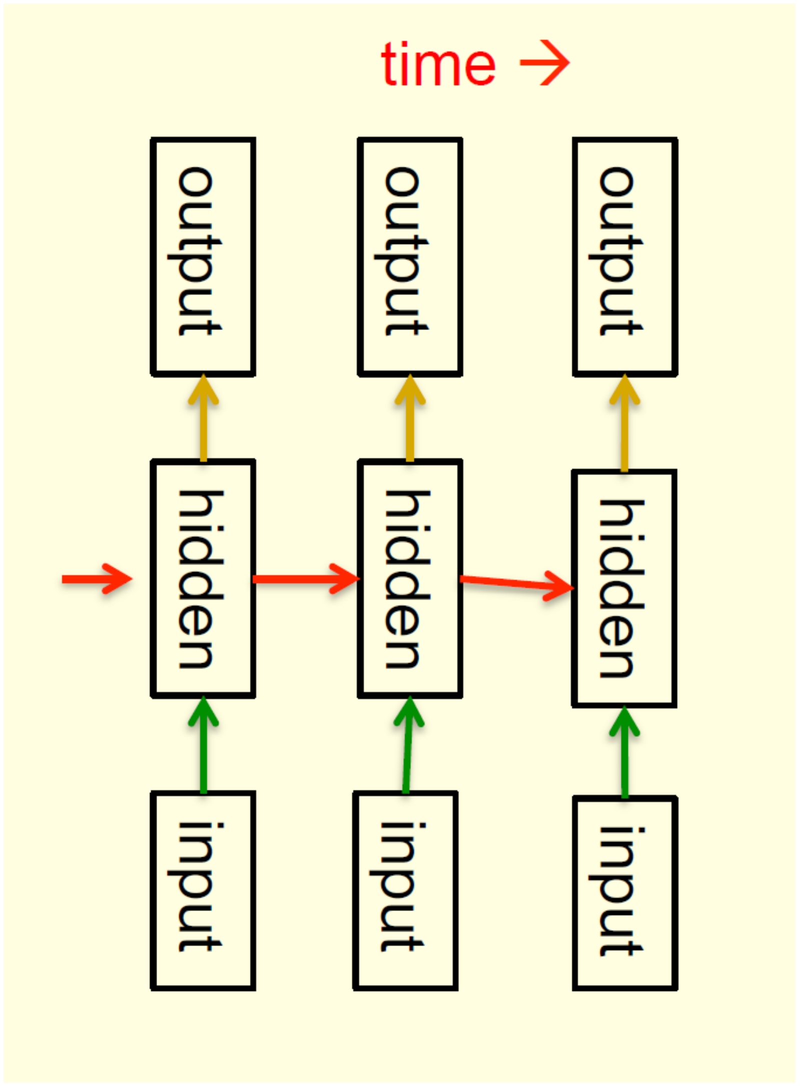


You can stack them too



Recurrent networks for sequence learning

- equivalent to deep networks with one hidden level per time slot
- but: hidden layers share weight (less parameters)



Example: RNN as a LM

- Ilya Sutskever (2011): RNN as a character-based language model – prediction of the next character
- Training set: half a billion characters from English Wikipedia
- Used for text generation:
  - predict probability distribution for the next character
  - sample from that distribution

RNN of Ilya Sutskever - sample output

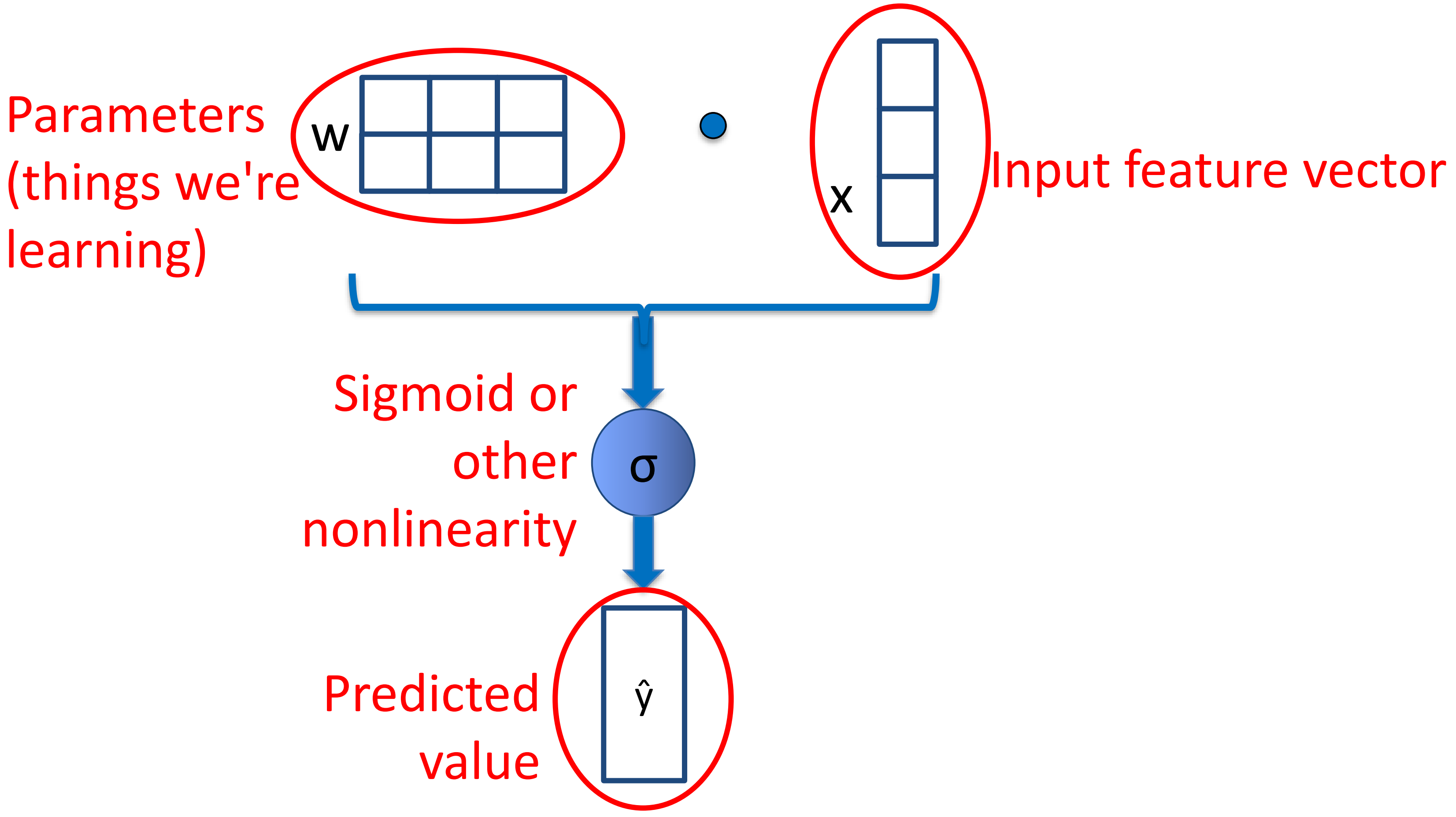
In 1974 Northern Denver had been overshadowed by CNL, and several Irish intelligence agencies in the Mediterranean region. However, on the Victoria, Kings Hebrew stated that Charles decided to escape during an alliance. The mansion house was completed in 1882, the second in its bridge are omitted, while closing is the proton reticulum composed below it aims, such that it is the blurring of appearing on any well-paid type of box printer.

Next word in speech recognition

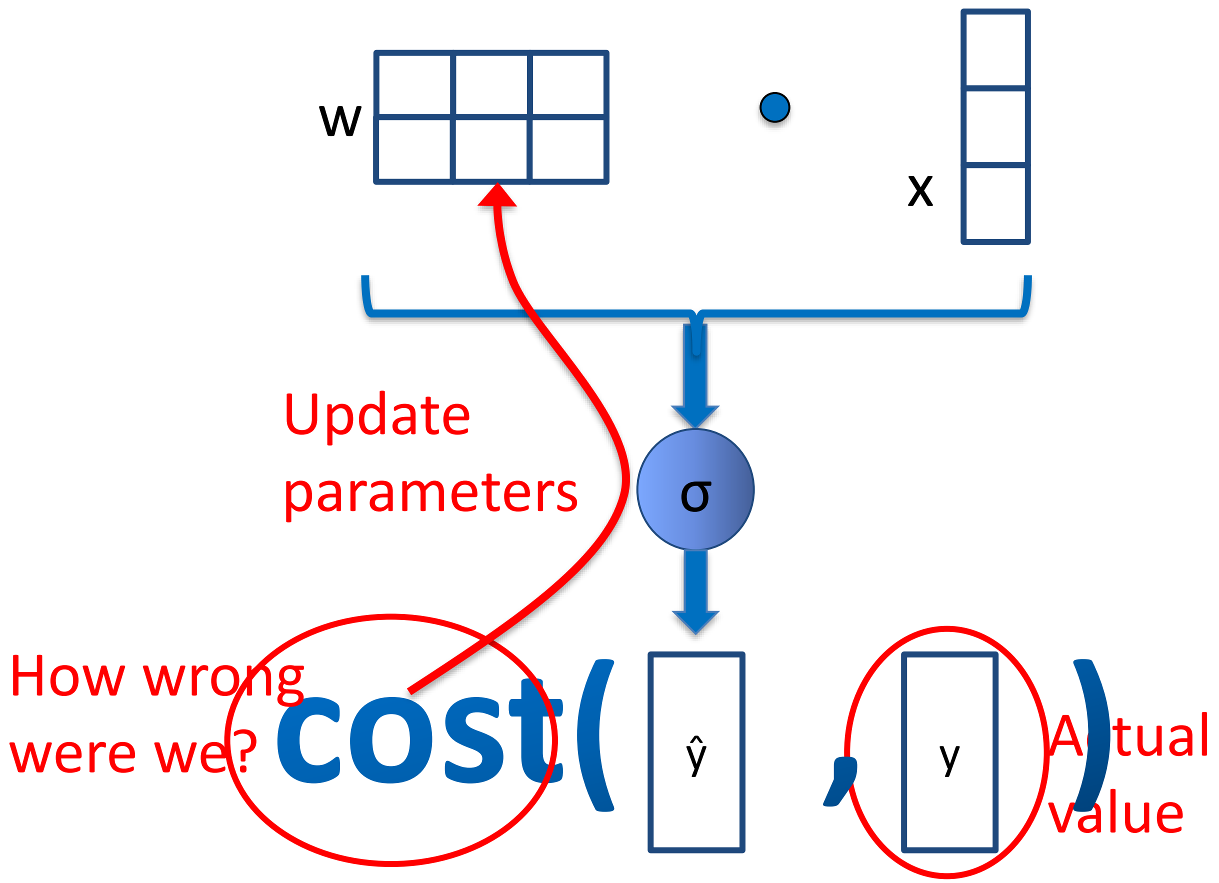
- difficult recognition in noisy environment
- many words with similar acoustic signal
- humans use semantics, context, and prediction to recognize the correct right word
- wreck a nice beach.
- recognize a speech

- automatic speech recognizers (ASR) need information about sensible next words
- classical approach was the trigram LM

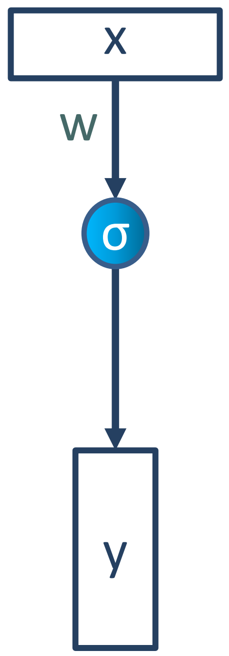
Schematic view of NNs


Parameters (things we're learning)
w
x
Input feature vector
Sigmoid or other nonlinearity
σ
ŷ
Predicted value

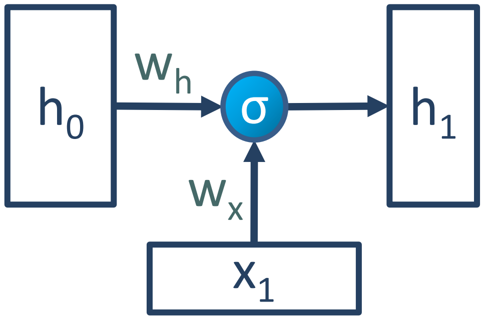
Schematic view of NNs



A Simplified Diagram

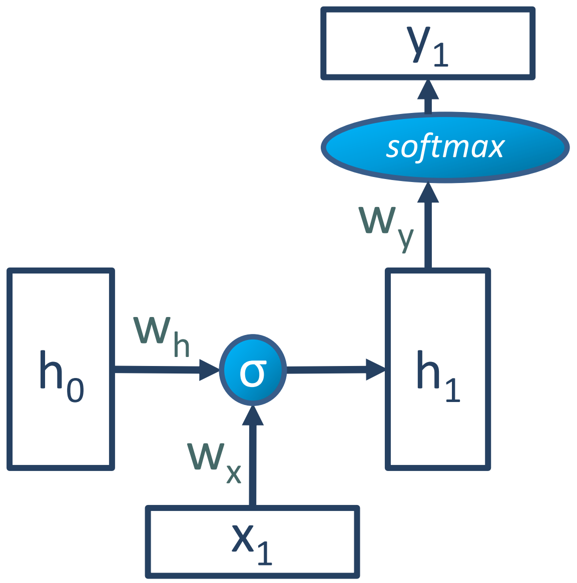


Recurrent Neural Networks (RNNs)



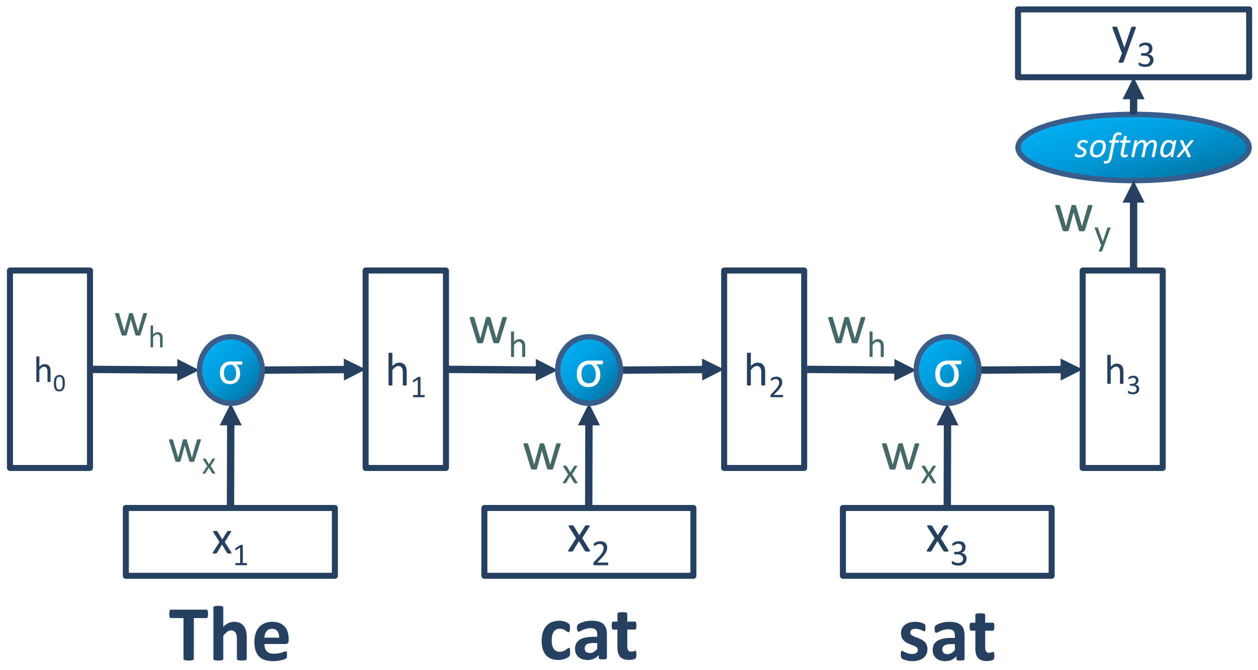
$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$

RNN

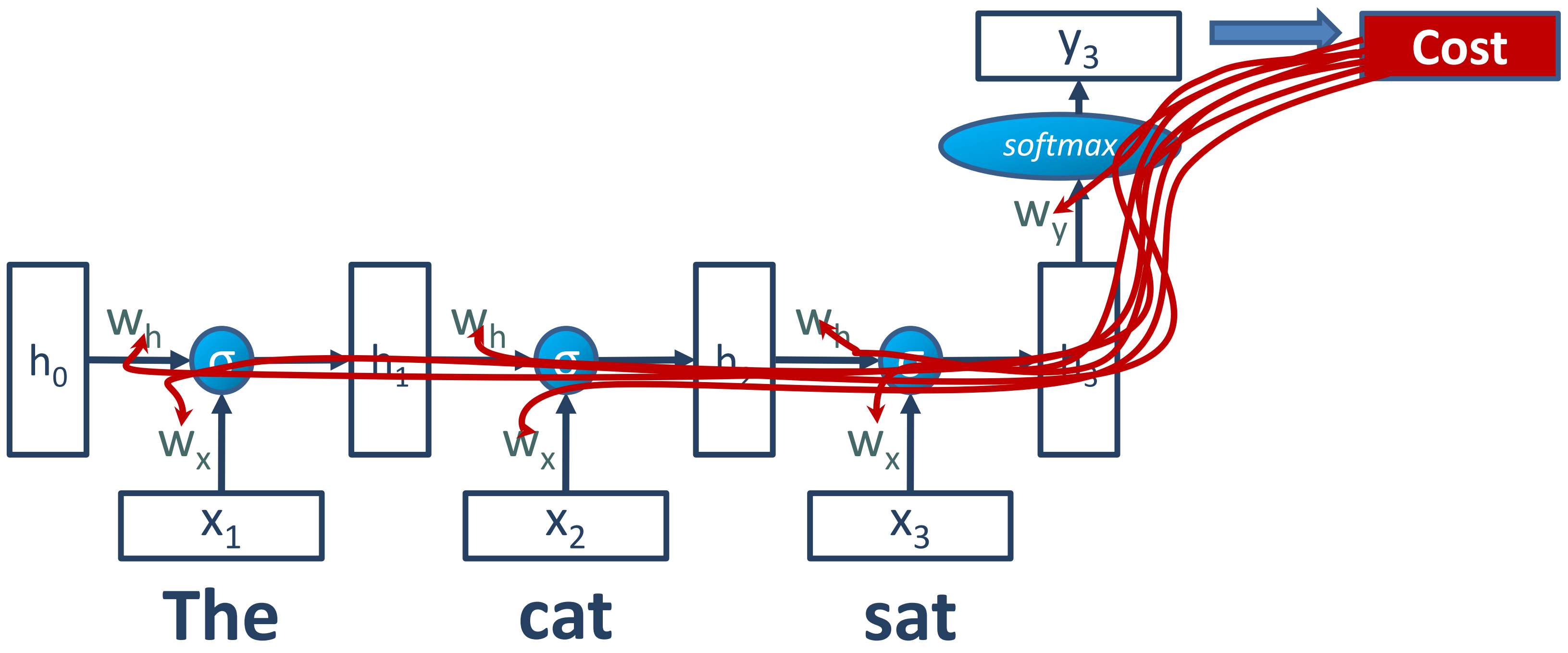


$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$
$$y_t = softmax(W_y h_t)$$

RNN

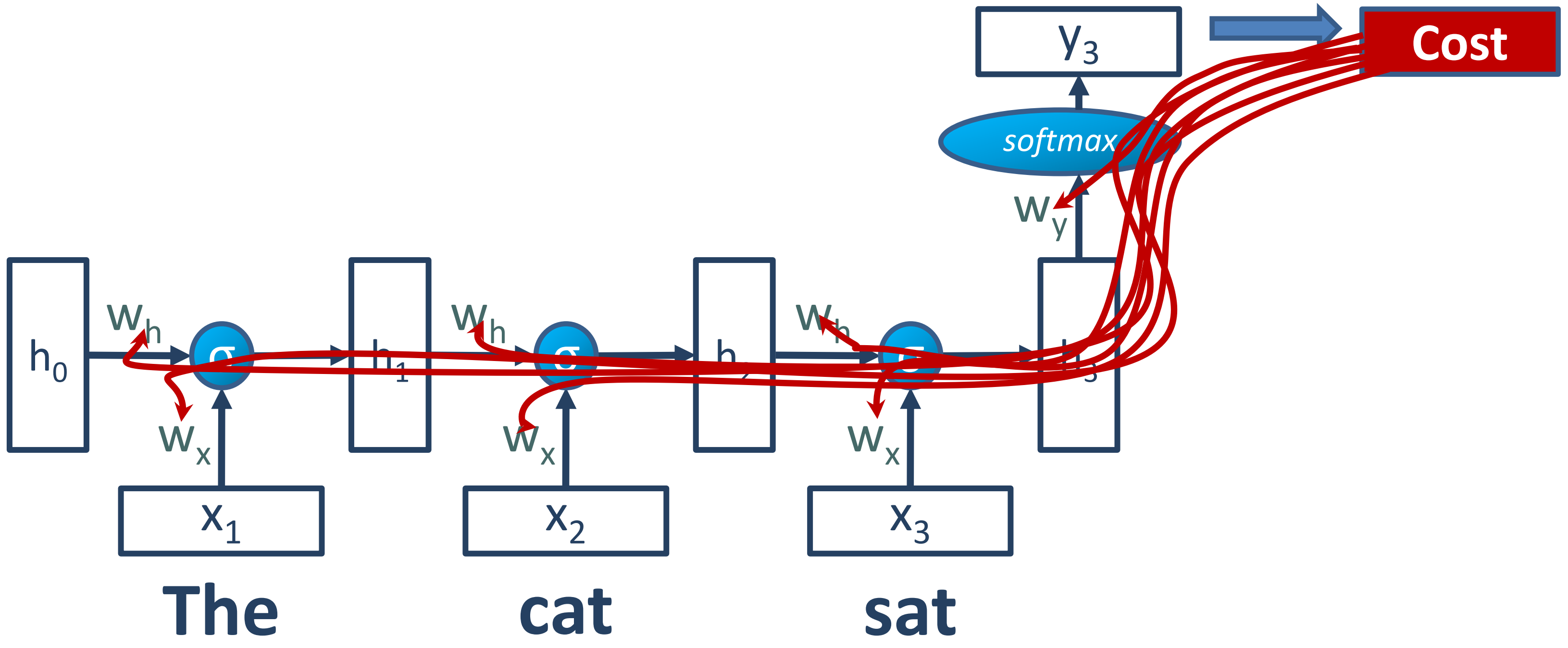


Updating Parameters of an RNN



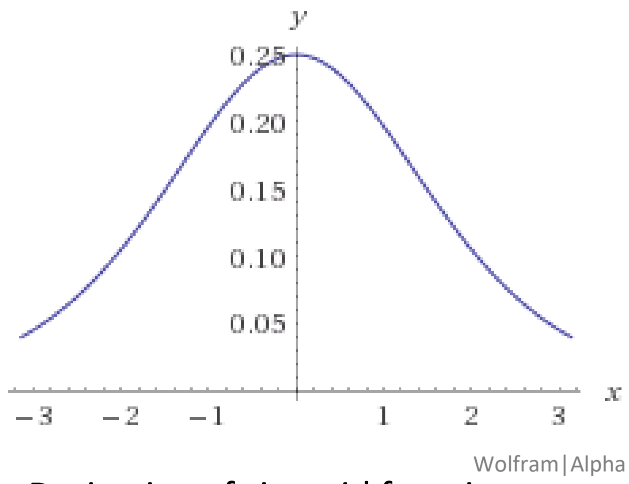
LSTM Motivation

Remember how we update RNN?



The Vanishing Gradient Problem

- Deep neural networks use backpropagation.
- Back propagation uses the chain rule.
- The chain rule multiplies derivatives.
- Often these derivatives are between 0 and 1.
- As the chain gets longer, products get smaller
- until they disappear.



Derivative of sigmoid function

Or do they explode?

- With gradients larger than 1,
- you encounter the opposite problem
- with products becoming larger and larger
- as the chain becomes longer and longer,
- causing overlarge updates to parameters.
- This is the exploding gradient problem.

Vanishing/Exploding Gradients Are Bad.

- If we cannot backpropagate very far through the network, the network cannot learn long-term dependencies.

  - My dog [chase/chases] squirrels. ✓

  vs.

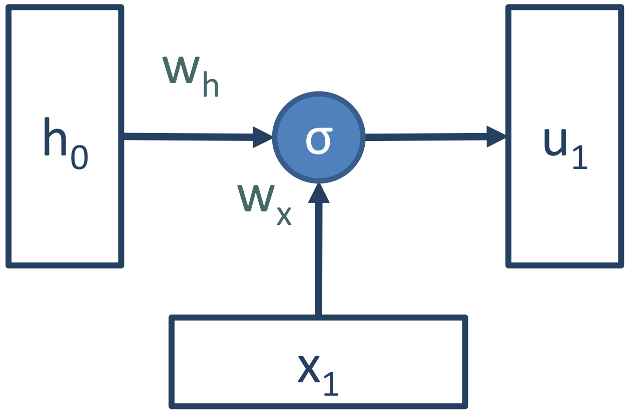
  - My dog, whom I adopted in 2009, [chase/chases] squirrels. ✗

LSTM Solution

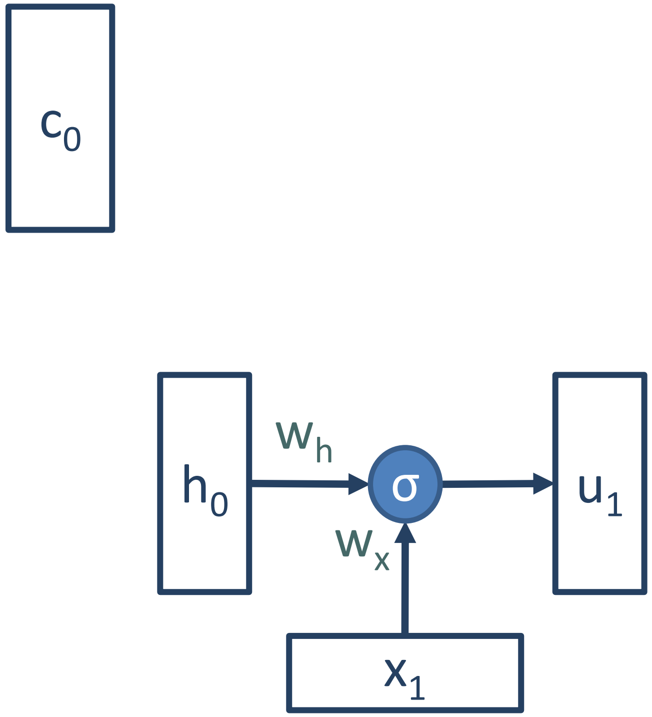
- Use memory cell to store information at each time step.
- Use “gates” to control the flow of information through the network.
  - Input gate: protect the current step from irrelevant inputs
  - Output gate: prevent the current step from passing irrelevant outputs to later steps
  - Forget gate: limit information passed from one cell to the next

Transforming RNN to LSTM

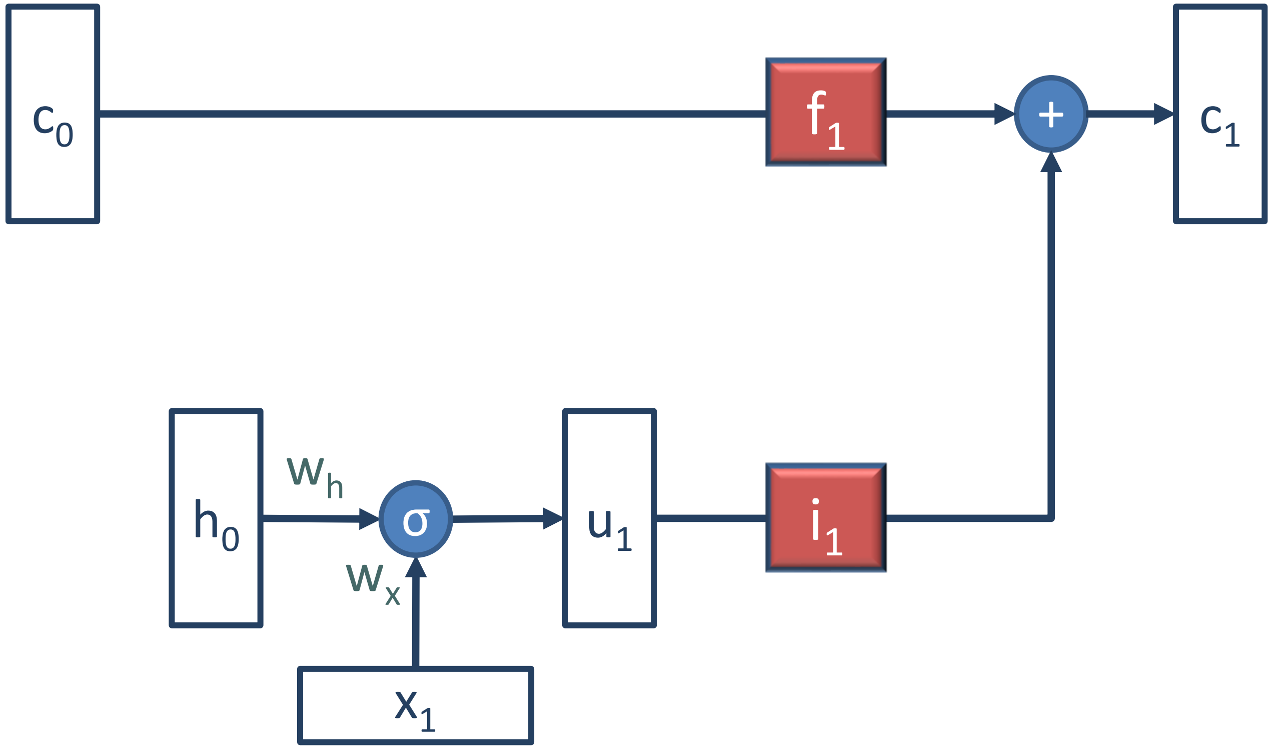
$$u_t = \sigma(W_h h_{t-1} + W_x x_t)$$



Transforming RNN to LSTM

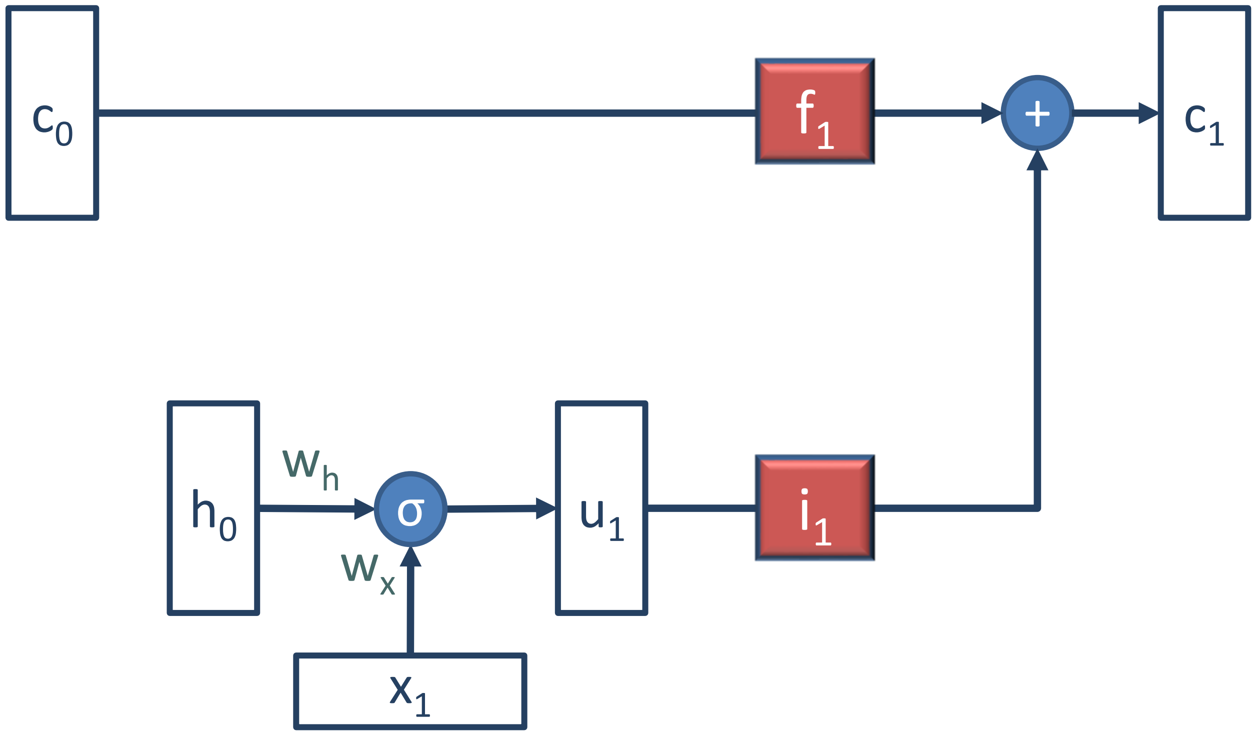


Transforming RNN to LSTM



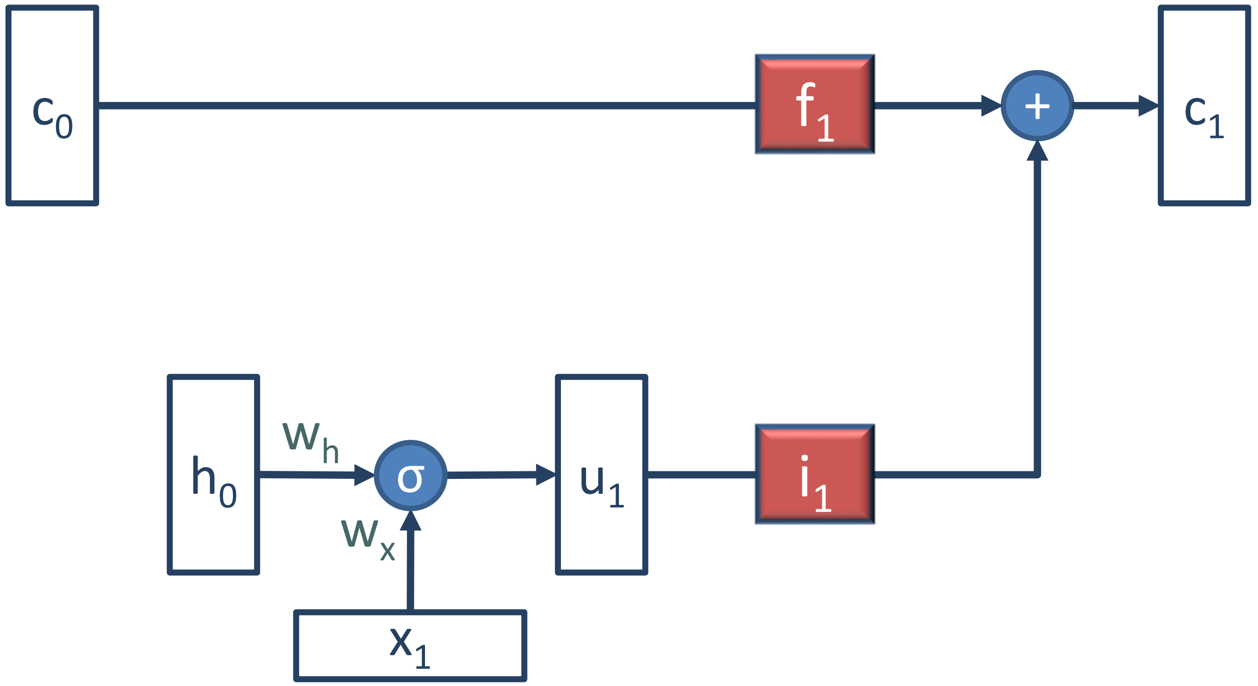
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

Transforming RNN to LSTM



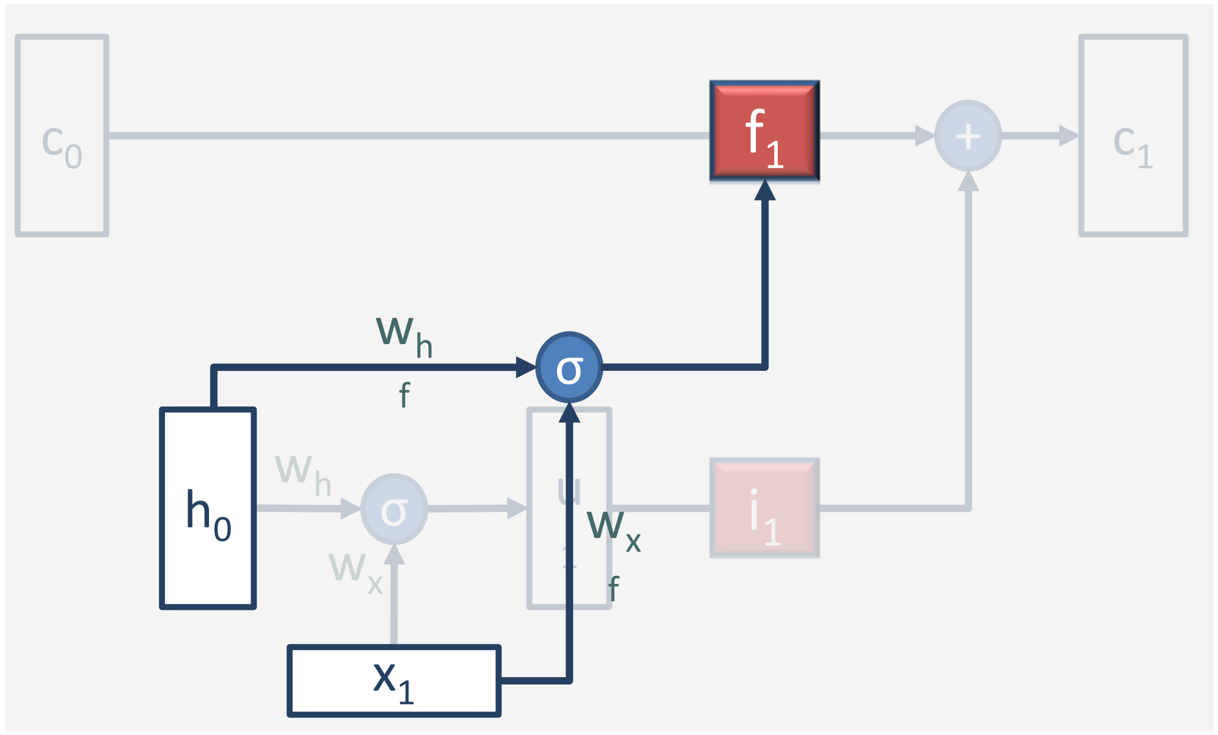
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

Transforming RNN to LSTM



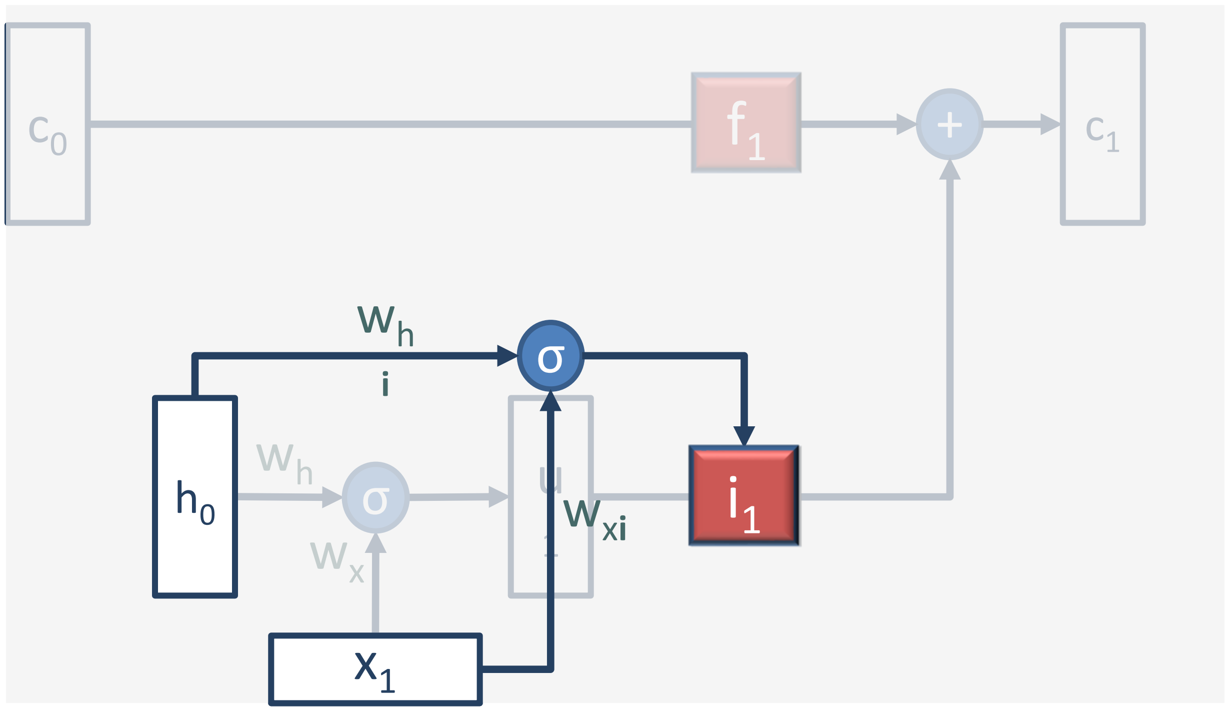
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

Transforming RNN to LSTM



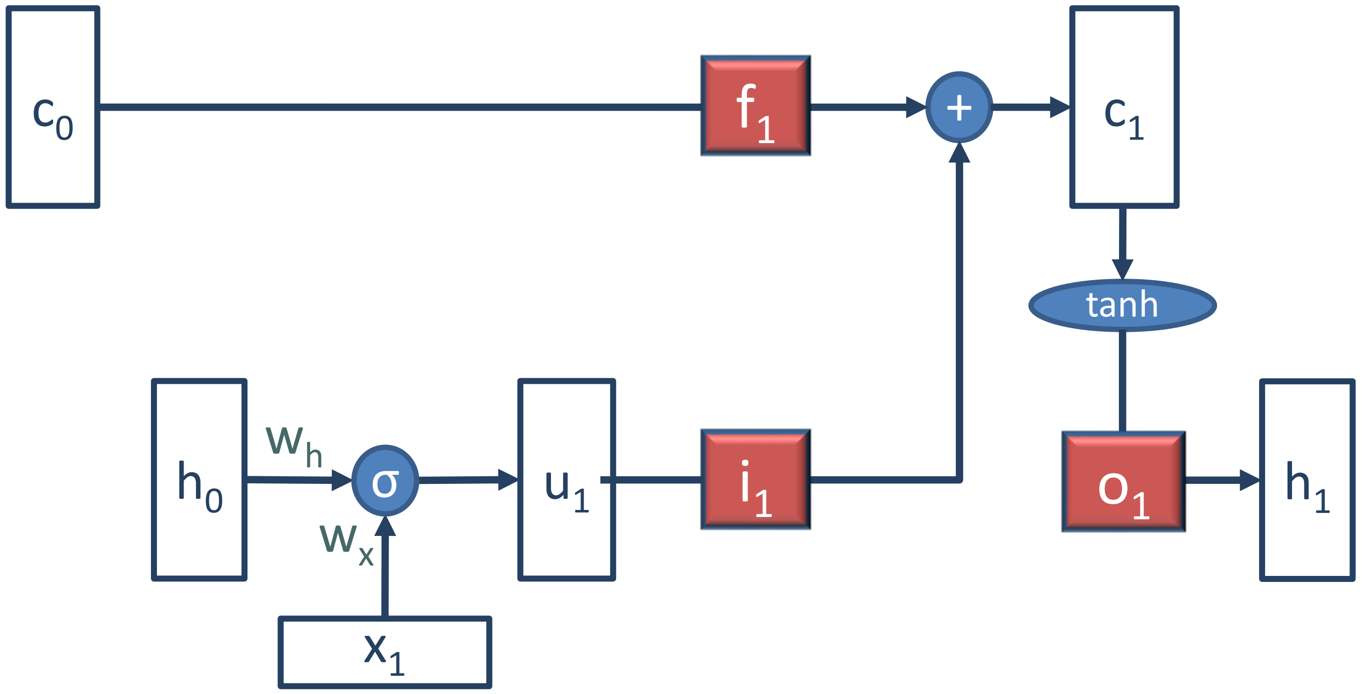
$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t)$$

Transforming RNN to LSTM



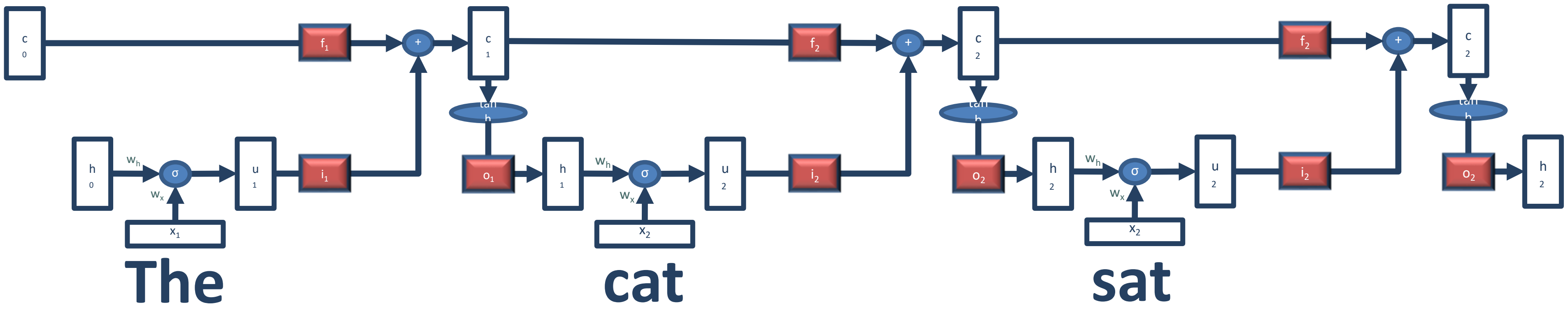
$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t)$$

Transforming RNN to LSTM



$$h_t = o_t \odot \tanh c_t$$

LSTM for Sequences



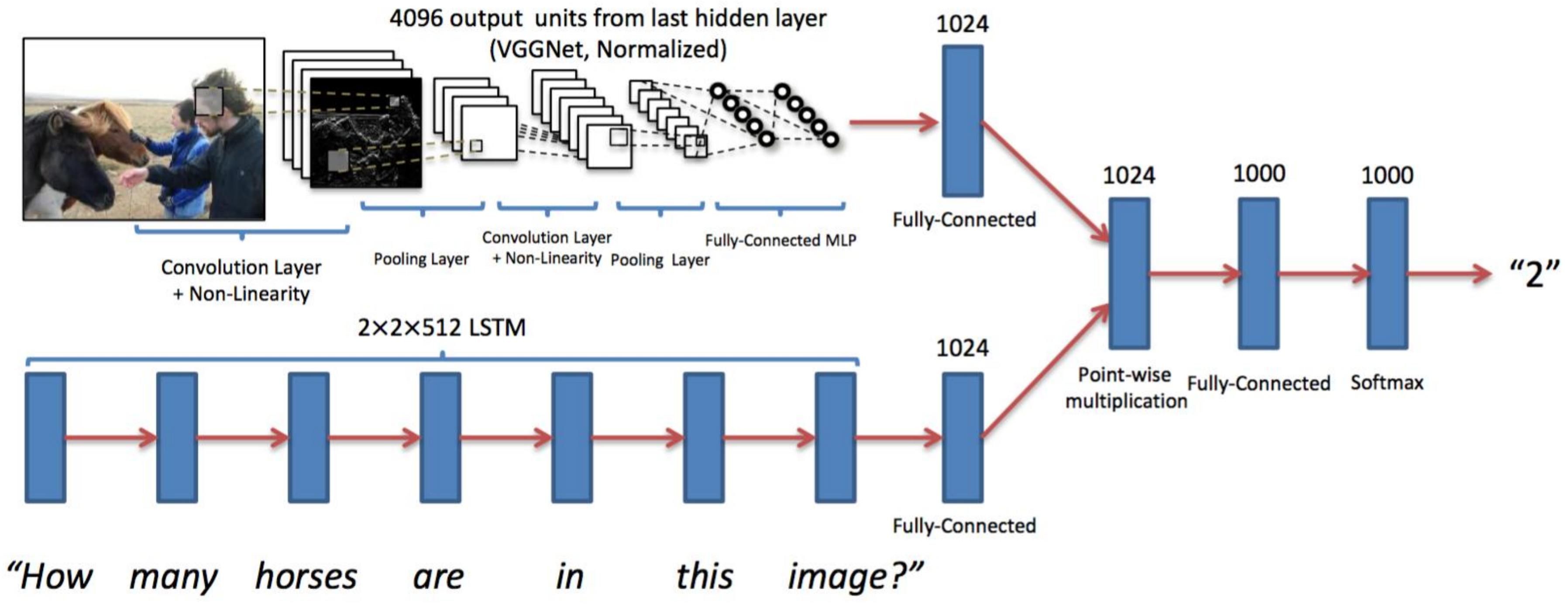
LSTM Applications

Handwriting generation

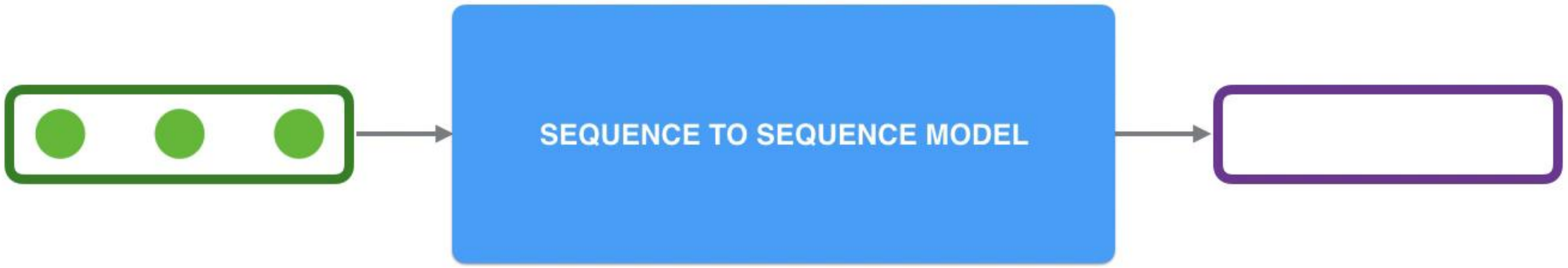
- Language identification (Gonzalez-Dominguez et al., 2014)
- Paraphrase detection (Cheng & Kartsaklis, 2015)
- Speech recognition (Graves, Abdel-Rahman, & Hinton, 2013)
- Handwriting recognition (Graves & Schmidhuber, 2009)
- Music composition (Eck & Schmidhuber, 2002) and lyric generation (Potash, Romanov, & Rumshisky, 2015)
- Robot control (Mayer et al., 2008)
- Natural language generation (Wen et al. 2015)
- Named entity recognition (Hammerton, 2003)

- And many more, but transformers mostly replaced them

Visual Question Answering

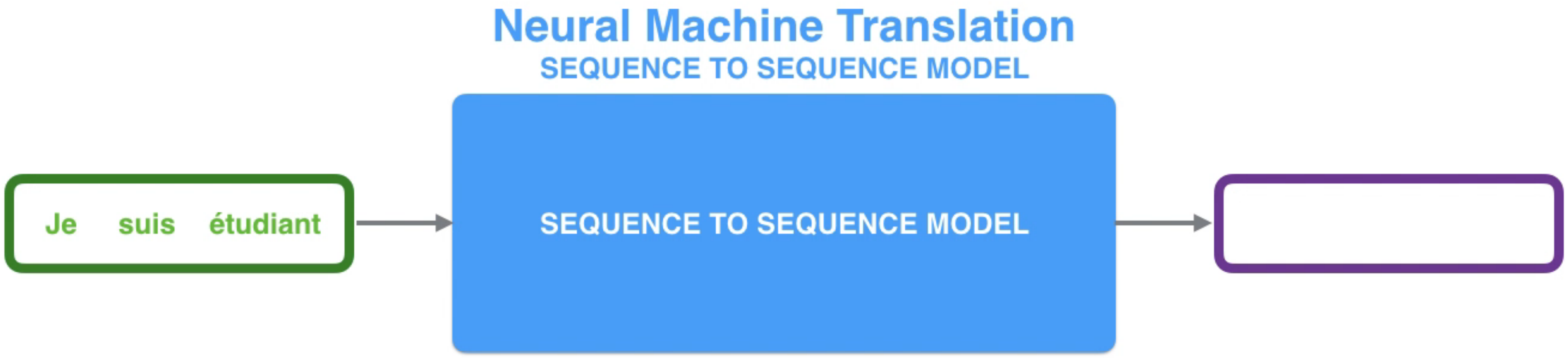


Seq2Seq model



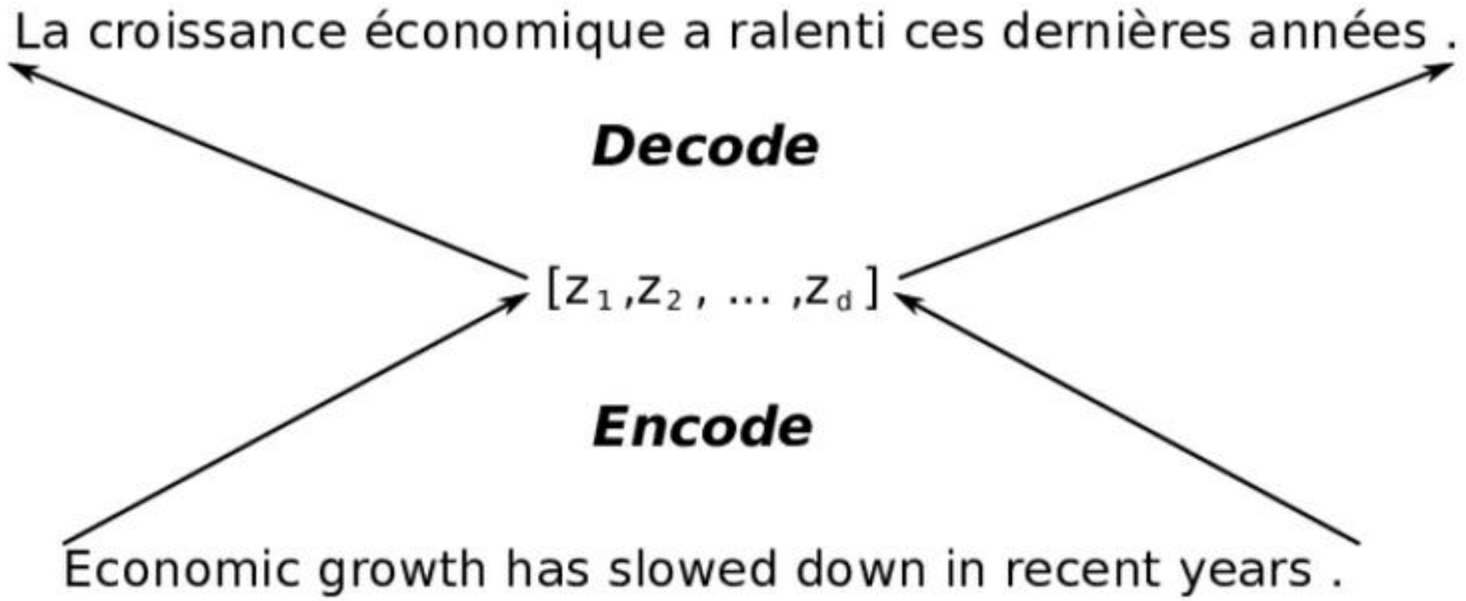
Videos by Jay Alammar: Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention), 2018

Seq2Seq for NMT

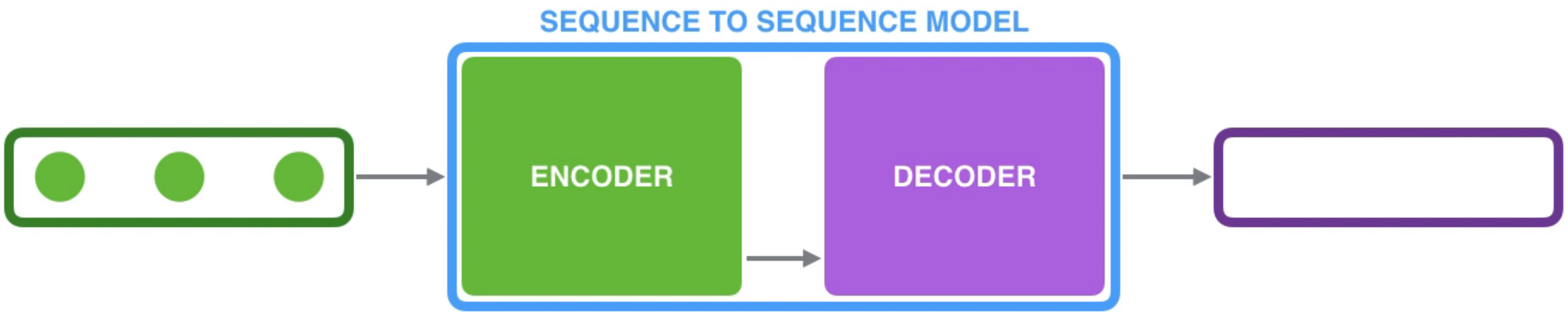

Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL
Je suis étudiant
SEQUENCE TO SEQUENCE MODEL

Encoder-Decoder model

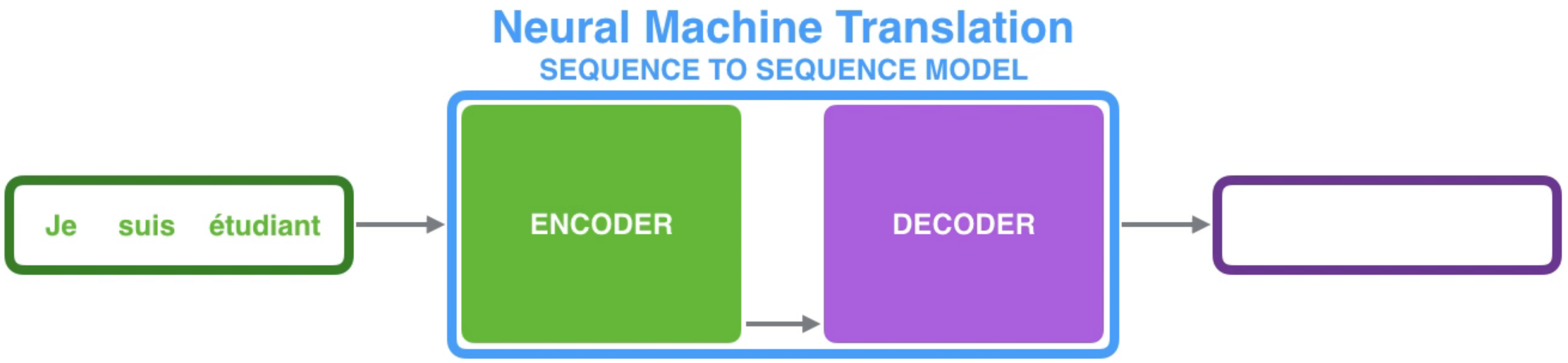
- encode into a latent space

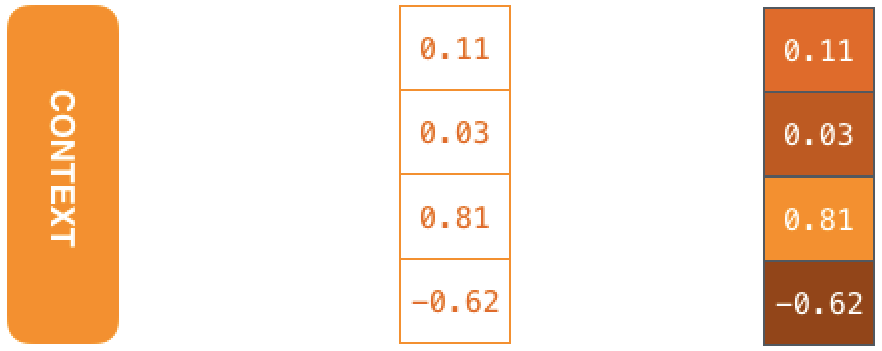

La croissance économique a ralenti ces dernières années .
Decode
$[z_1, z_2, \ldots, z_d]$
Encode
Economic growth has slowed down in recent years .

Encoder-decoder for sequences

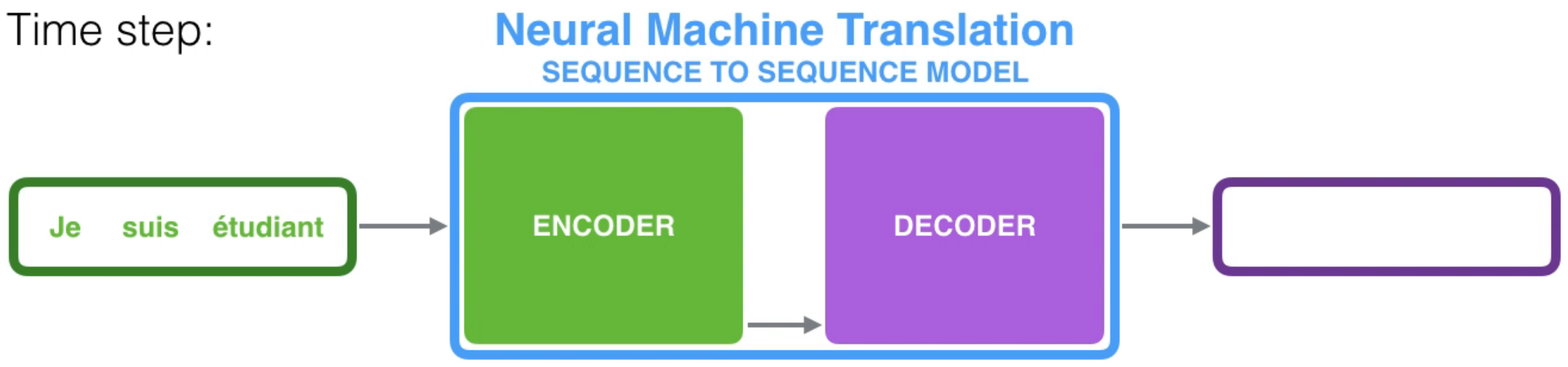


Encoder-decoder for NMT

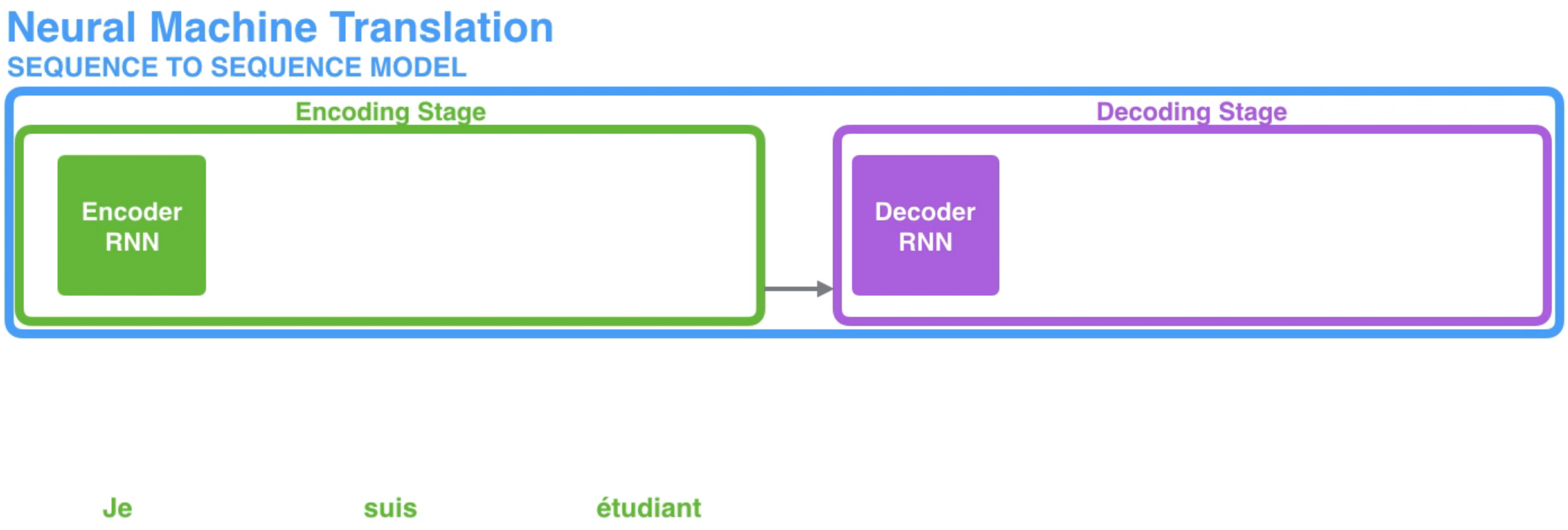




Encoder-decoder hidden states



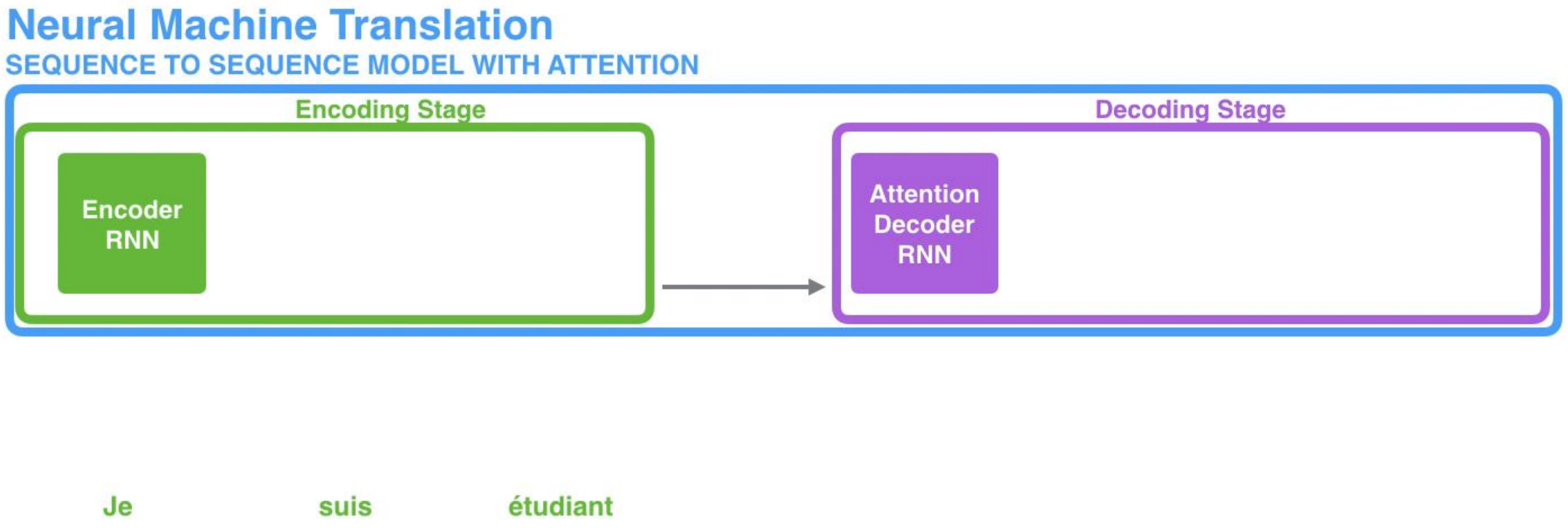
Unrolled encoder-decoder



Problems of encoder-decoder models

- long dependencies that would require larger networks and many more training data
- the information of different length sentences is stored in the fixed length hidden layer (migh be too long or to short)
- solution: attention mechanism

NMT with attention



Attention mechanism implementation for RNNs 1/2

- for all input words, we store their hidden layer weights
- during decoding, we add these vectors to the decoder input
- we use bidirectional encoding (forward and backward LM) and concatenate both weight vectors
- vectors are stored into a matrix

$$\overrightarrow{\boldsymbol{h}}_j^{(f)} = \mathrm{RNN}(\mathrm{embed}(f_j), \overrightarrow{\boldsymbol{h}}_{j-1}^{(f)})$$
$$\overleftarrow{\boldsymbol{h}}_j^{(f)} = \mathrm{RNN}(\mathrm{embed}(f_j), \overleftarrow{\boldsymbol{h}}_{j+1}^{(f)}).$$

$$\boldsymbol{h}_j^{(f)} = [\overleftarrow{\boldsymbol{h}}_j^{(f)}; \overrightarrow{\boldsymbol{h}}_j^{(f)}].$$

$$H^{(f)} = \mathrm{concat\_col}(\boldsymbol{h}_1^{(f)}, \ldots, \boldsymbol{h}_{|F|}^{(f)}).$$

Attention mechanism implementation for RNNs 2/2

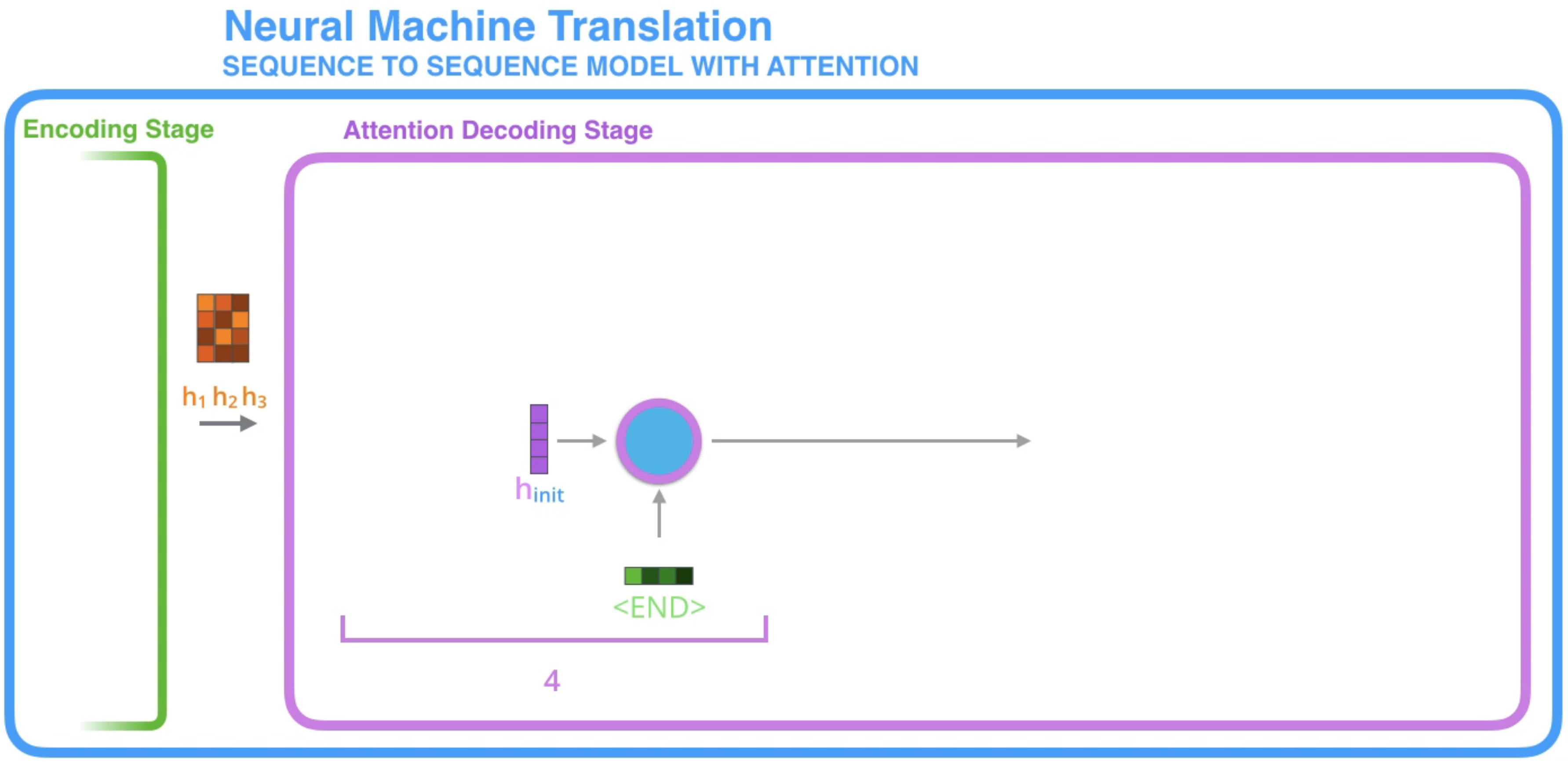
- we train the attention – which stored vectors are more or less important for decoding certain words
- the importance is determined with the attention vector α_t (between 0 and 1, sums to 1), applied to stored hidden weights and given as additional input to the decoder

$$\boldsymbol{c}_t = H^{(f)} \boldsymbol{\alpha}_t$$

Illustration of attention

Attention at time step 4

Decoder with attention


Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL WITH ATTENTION
Encoding Stage
Attention Decoding Stage
h1 h2 h3
hinit
<END>
4

Attention produces alignments

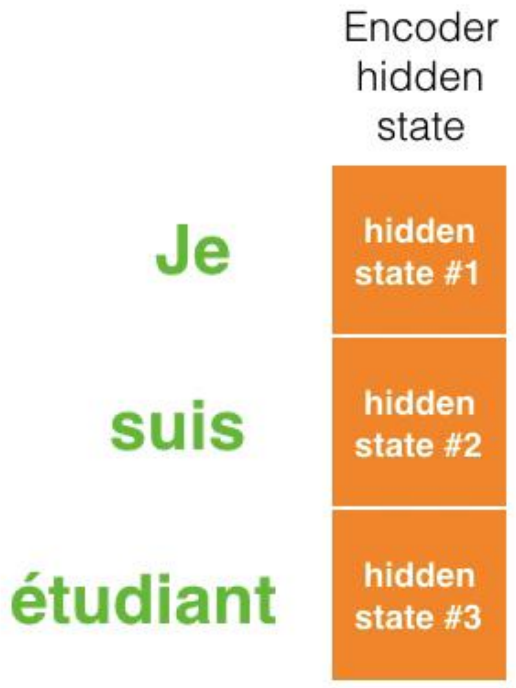
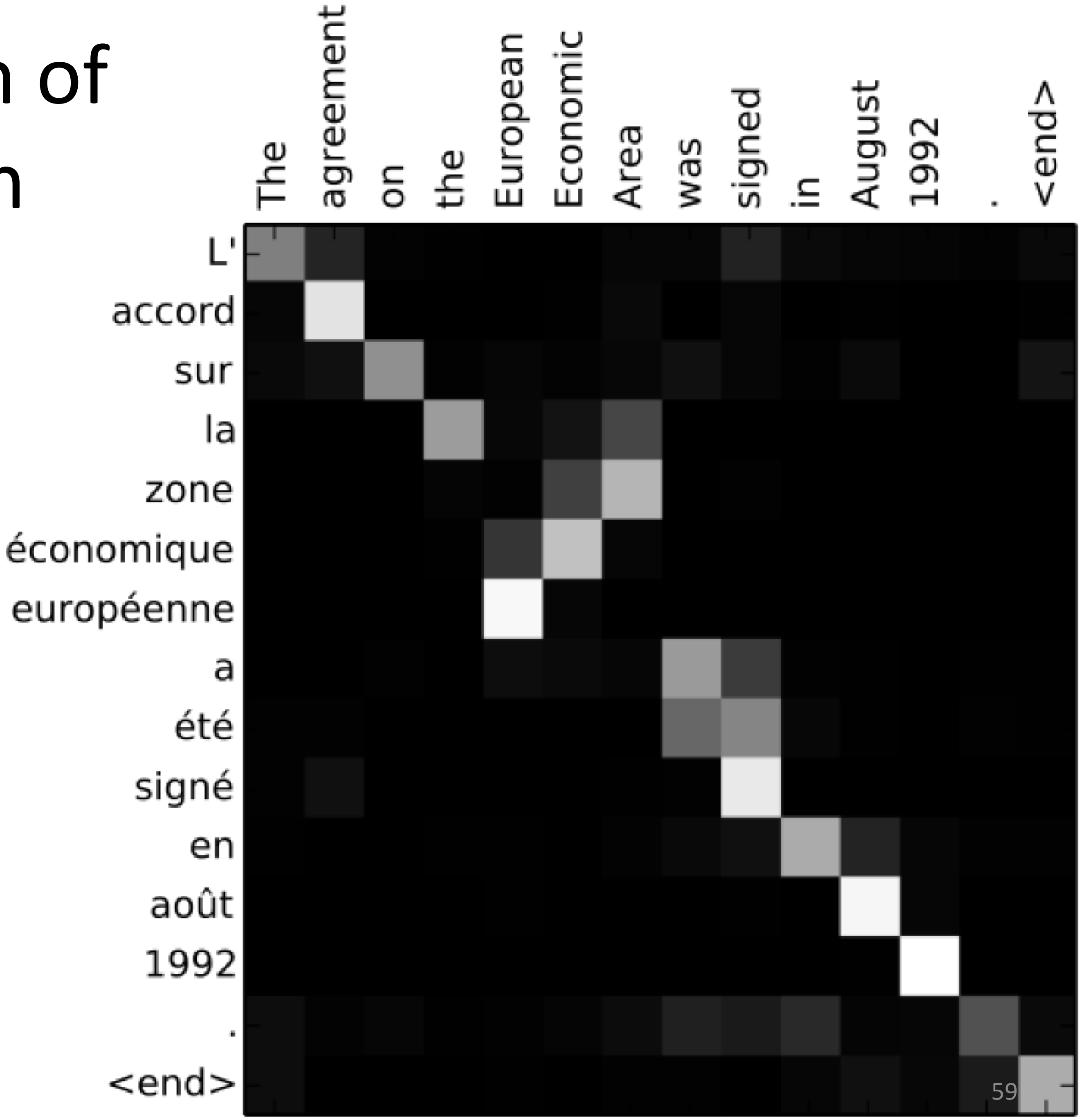
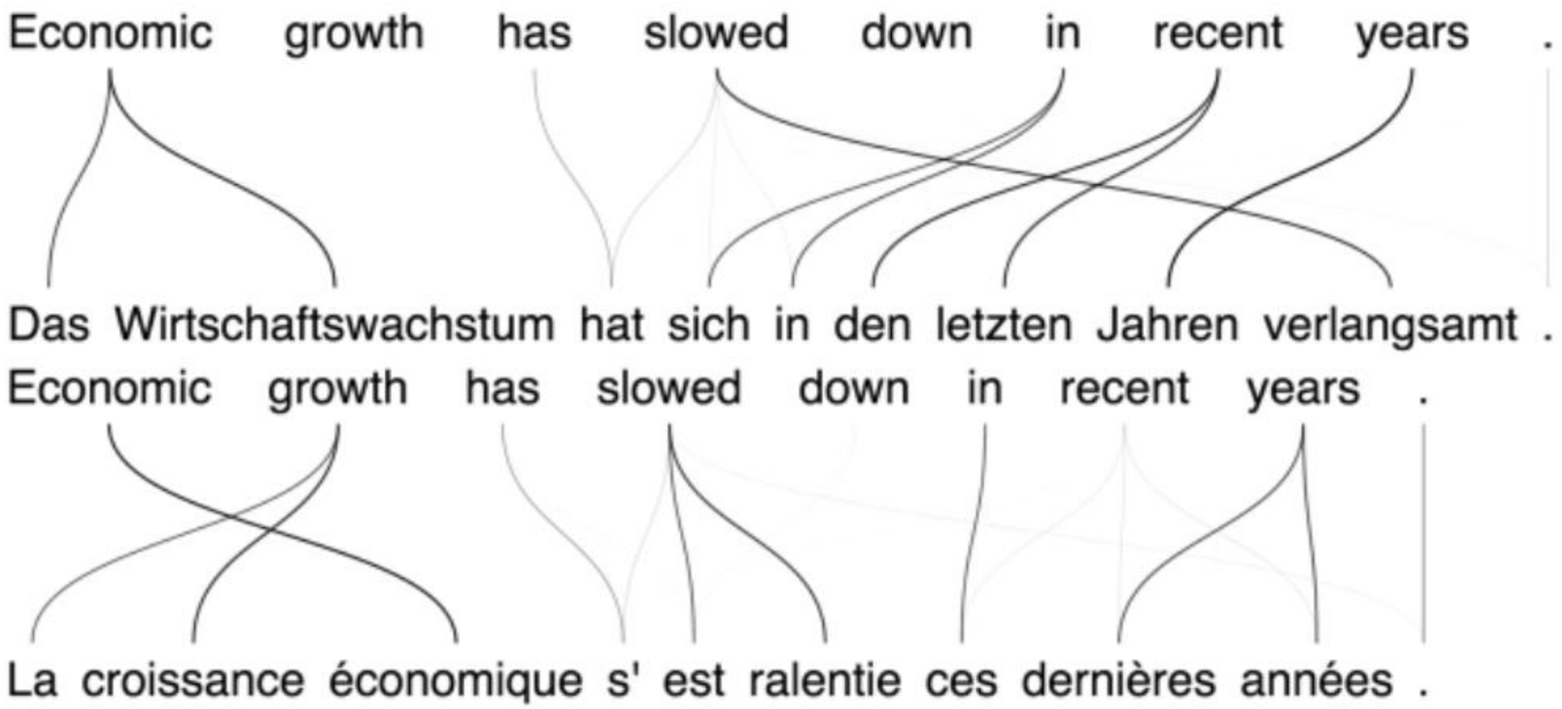


Illustration of attention



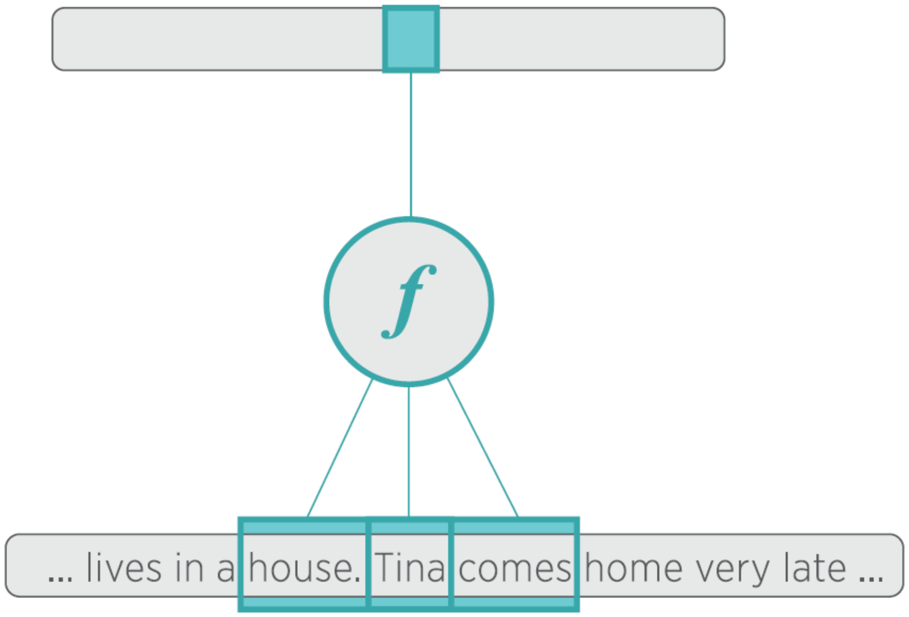
Attention illustration


Economic growth has slowed down in recent years .
Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt .
Economic growth has slowed down in recent years .
La croissance économique s' est ralentie ces dernières années .

Problems with RNNs

- We want parallelization, but RNNs are inherently sequential
- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length between states grows with sequence
- If attention gives us access to any state… maybe we can just use attention and don’t need the RNN?
- We will deal with this idea in transformers.

Convolutional neural networks



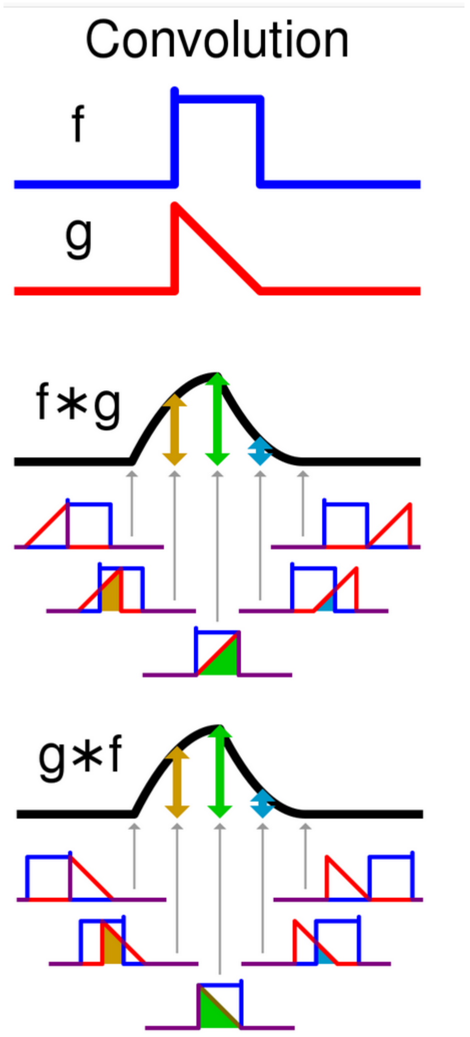
Convolution

- an operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other.

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau.$$

- for discrete functions

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

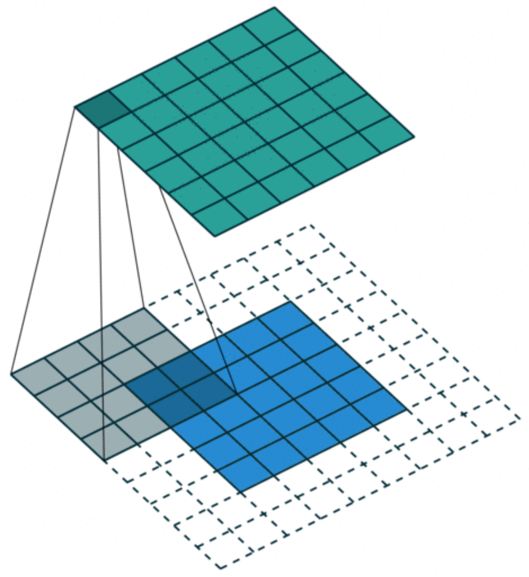


Convolutional Neural Network (CNN)

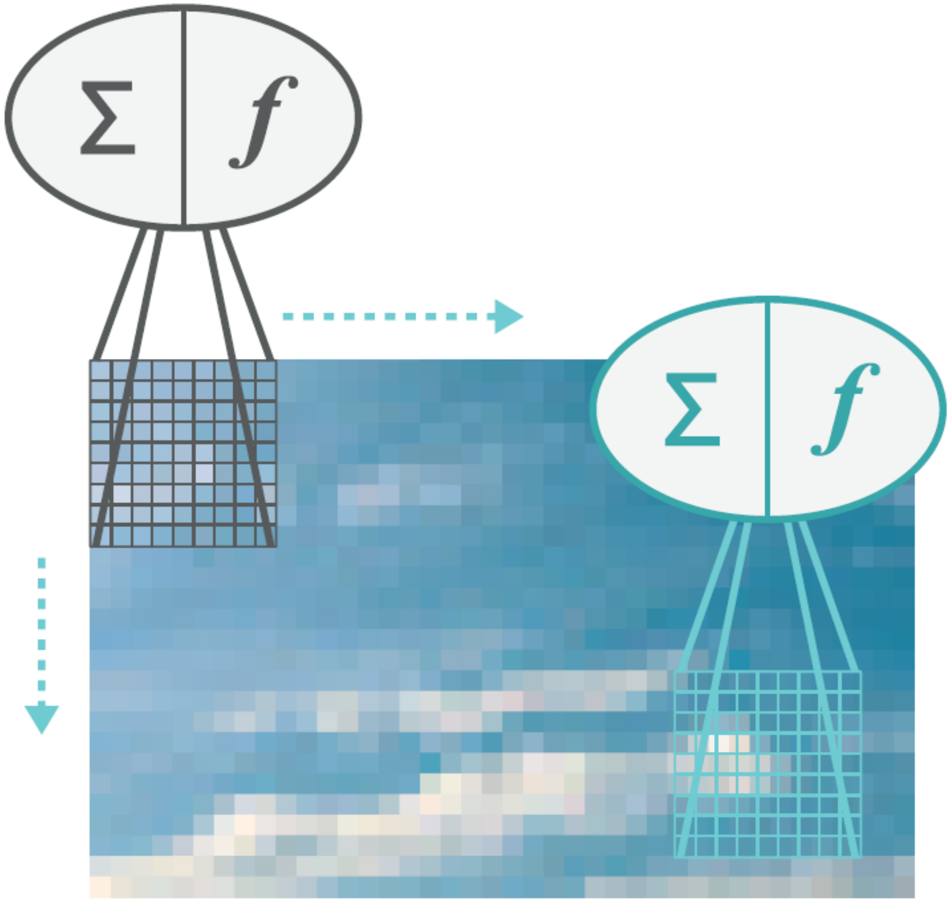
- Convolutional Neural Networks are inspired by mammalian visual cortex.
- The visual cortex contains a complex arrangement of cells, which are sensitive to small sub-regions of the visual field, called a receptive field. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images.
- Two basic cell types:
    - Simple cells respond maximally to specific edge-like patterns within their receptive field.
    - Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.

Convolutional neural networks (CNN)

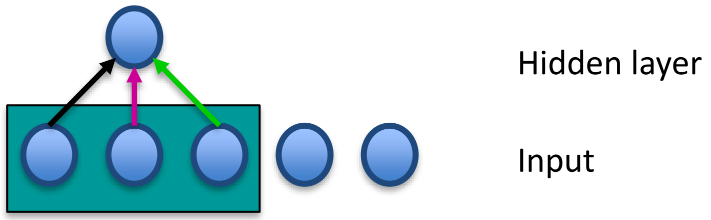
- currently, the most successful approach in image analysis, successful in language
- idea: many copies of small detectors used all over the image, recursively combined,
- detectors are learned, combination are learned



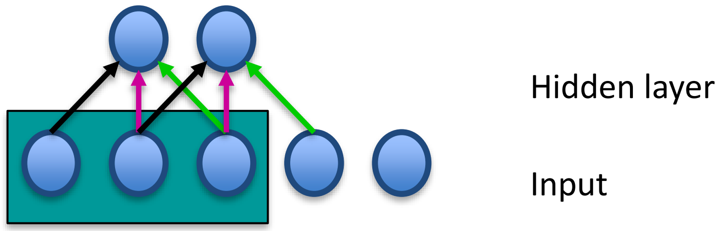
2d convolution for images



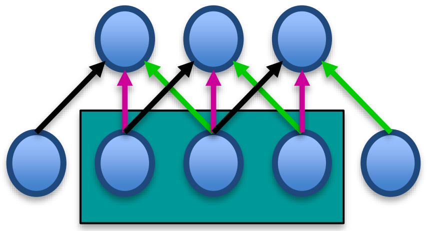
Basic Idea of CNNs



Basic Idea of CNNs



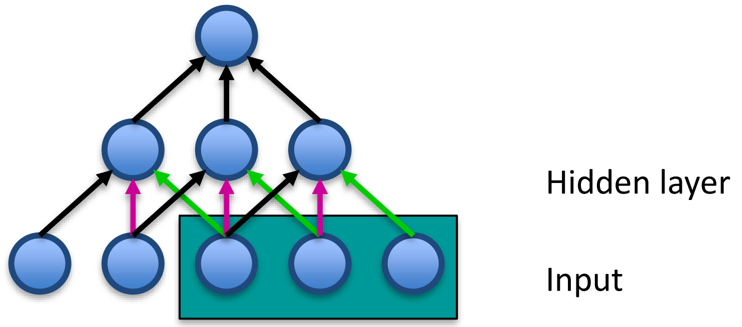
Basic Idea of CNNs



Hidden layer

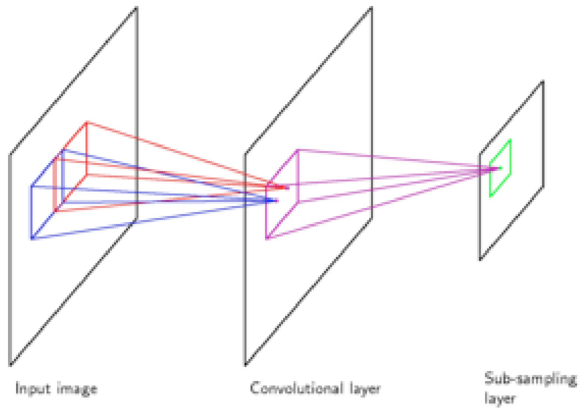
Input

Basic Idea of CNNs



Convolutional Network

- The network is not fully connected.
- Different nodes are responsible for different regions of the image.
- This allows for robustness to transformations.
- Convolution is combined with subsampling.



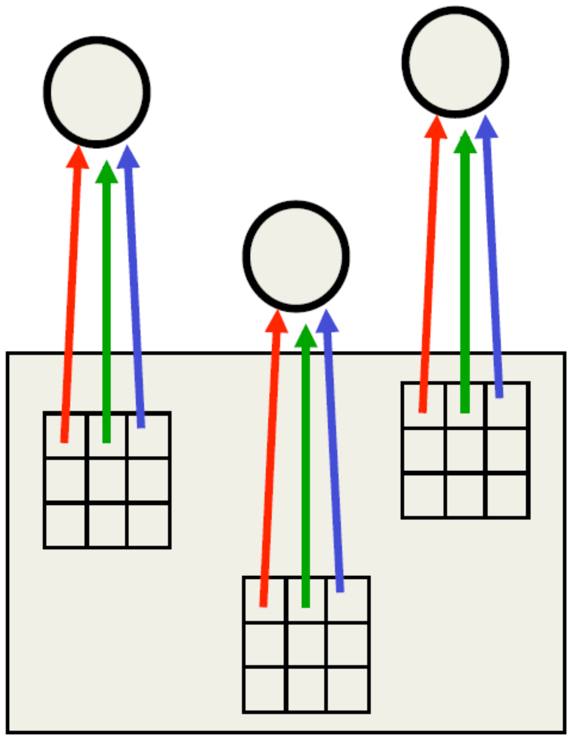
CNN Architecture: Convolutional Layer

- The core layer of CNNs
- The convolutional layer consists of a set of filters.
  - Each filter covers a spatially small portion of the input data.
- Each filter is convolved across the dimensions of the input data, producing a multidimensional feature map.
  - As we convolve the filter, we are computing the dot product between the parameters of the filter and the input.
- Intuition: the network will learn filters that activate when they see some specific type of feature at some spatial position in the input.
- The key architectural characteristics of the convolutional layer is local connectivity and shared weights.

Neural implementation of convolution

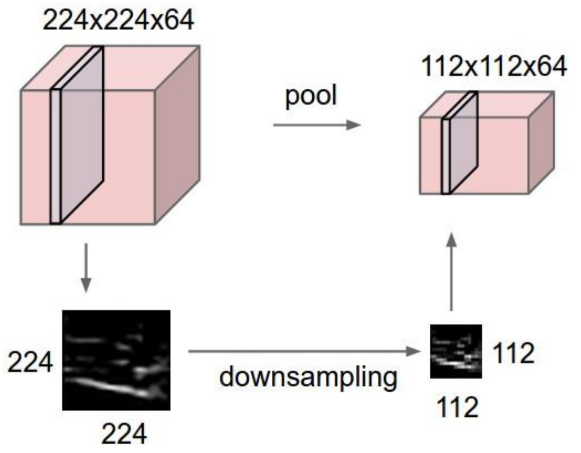
- weights of the same colors have equal weights
- adapted backpropagation

- images: 2d convolution
- languages: 1d convolution

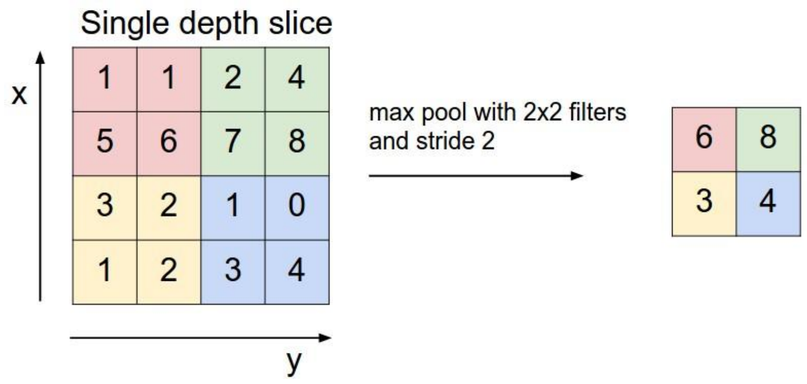


CNN Architecture: Pooling Layer

- Intuition: to progressively reduce the spatial size of the representation, to reduce the amount of parameters and computation in the network, and hence to also control overfitting
- Pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value of the features in that region.



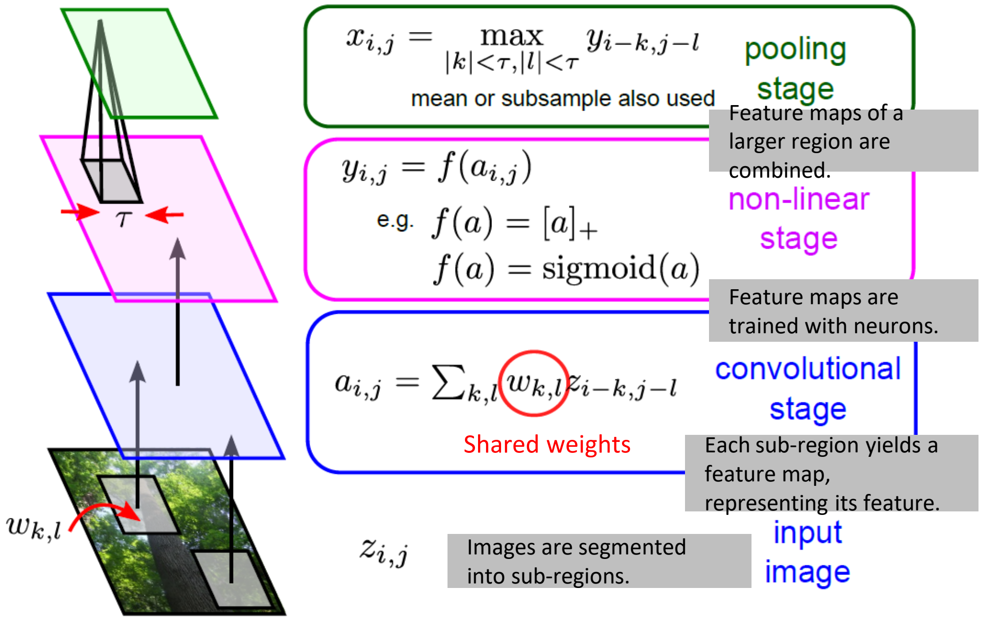
Input



CNN: pooling

- reduces the number of connections to the next layer (prevents the excessive number of parameters, speeds-up learning, reduces overfitting)
- max-pooling, min-pooling, average-pooling
- the problem: after several layers of pooling we lose the information about the exact location of the recognized pattern and about spatial relations between different patterns and features, e.g., a nose on a forehead

Building-blocks for CNN’s


$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$
mean or subsample also used
pooling stage
$y_{i,j} = f(a_{i,j})$
e.g. $f(a) = [a]_+$
$f(a) = \text{sigmoid}(a)$
non-linear stage
$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$
Shared weights
convolutional stage
$z_{i,j}$
input image
τ
$w_{k,l}$

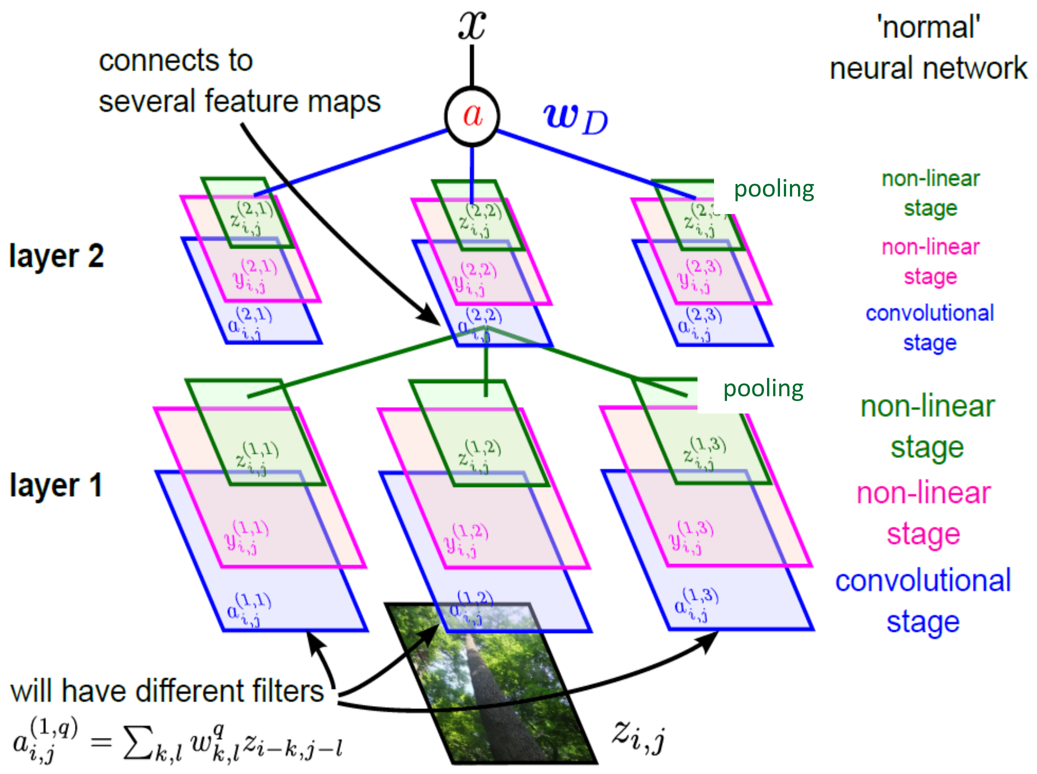
Feature maps of a larger region are combined.

Feature maps are trained with neurons.

Each sub-region yields a feature map, representing its feature.

Images are segmented into sub-regions.

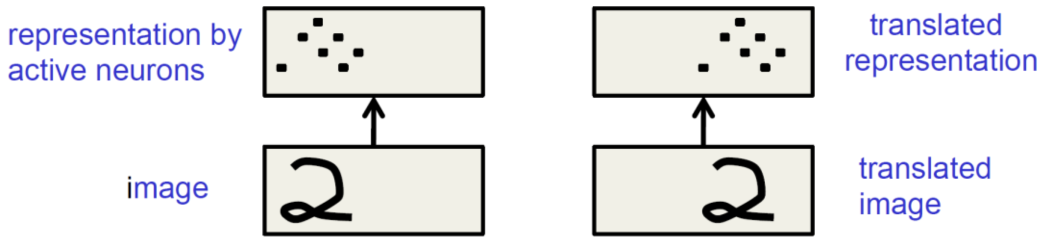
Full CNN



x
a
w_D
'normal' neural network
connects to several feature maps
pooling
non-linear stage
non-linear stage
convolutional stage
layer 2
$z_{i,j}^{(2,1)}$
$y_{i,j}^{(2,1)}$
$a_{i,j}^{(2,1)}$
$z_{i,j}^{(2,2)}$
$y_{i,j}^{(2,2)}$
$a_{i,j}^{(2,2)}$
$y_{i,j}^{(2,3)}$
$a_{i,j}^{(2,3)}$
pooling
non-linear stage
non-linear stage
convolutional stage
layer 1
$z_{i,j}^{(1,1)}$
$y_{i,j}^{(1,1)}$
$a_{i,j}^{(1,1)}$
$z_{i,j}^{(1,2)}$
$y_{i,j}^{(1,2)}$
$a_{i,j}^{(1,2)}$
$z_{i,j}^{(1,3)}$
$y_{i,j}^{(1,3)}$
$a_{i,j}^{(1,3)}$
will have different filters
$a_{i,j}^{(1,q)} = \sum_{k,l} w_{k,l}^{q} z_{i-k,j-l}$
$z_{i,j}$

Convolutional networks: illustration on image recognition

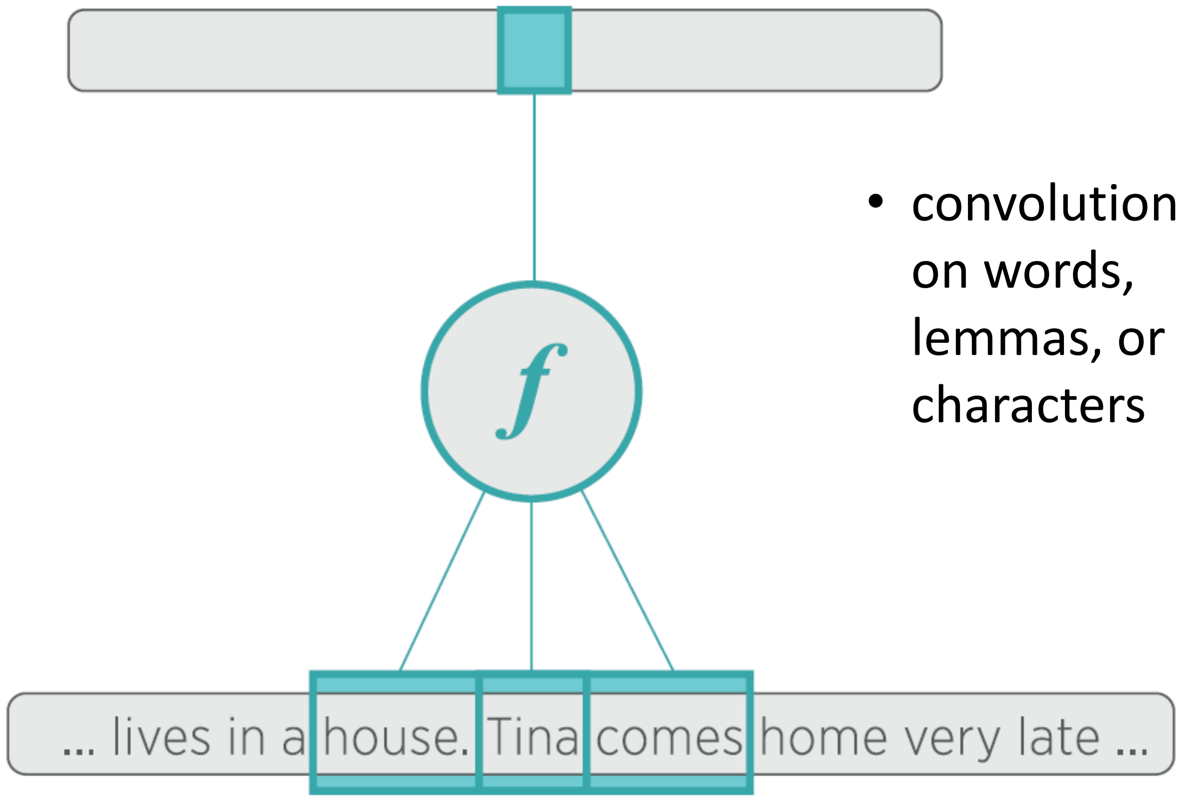
- a useful feature is learned and used on several positions
- prevents dimension hopping
- max-pooling



Benefits of CNNs

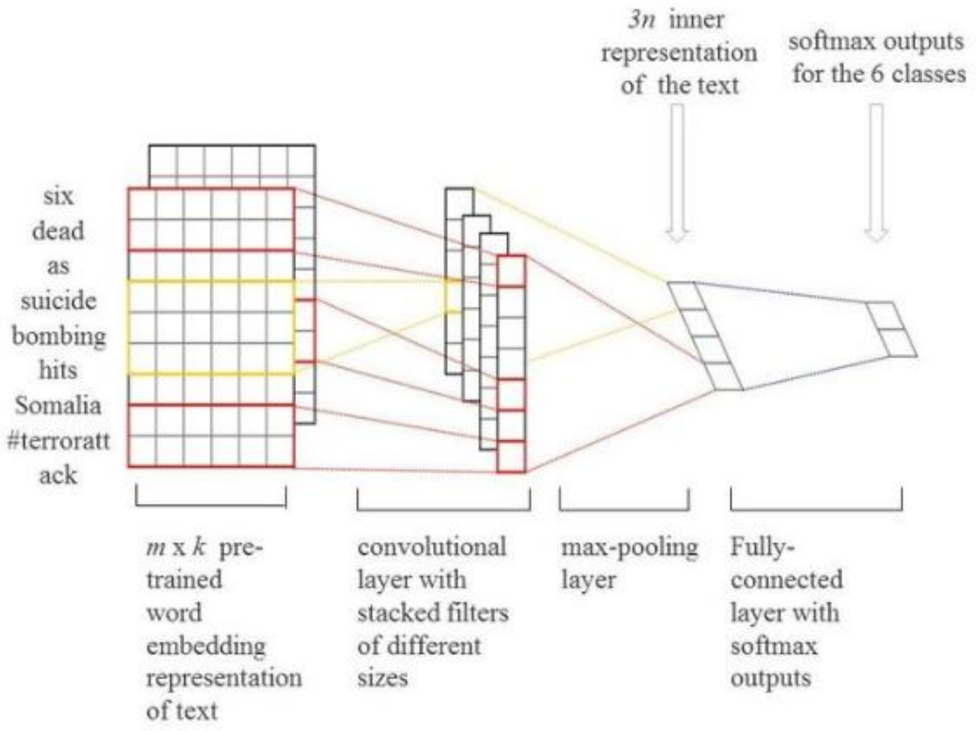
- The number of weights can be much less than for a fully connected layer.
- The small number of weights can use different parts of the image as training data. Thus, we have several orders of magnitude more data to train the fewer number of weights.
- We get translation invariance for free.
- Fewer parameters take less memory and thus all the computations can be carried out in memory in a GPU or across multiple processors.

1d convolution for text



- convolution on words, lemmas, or characters

1d CNN architecture

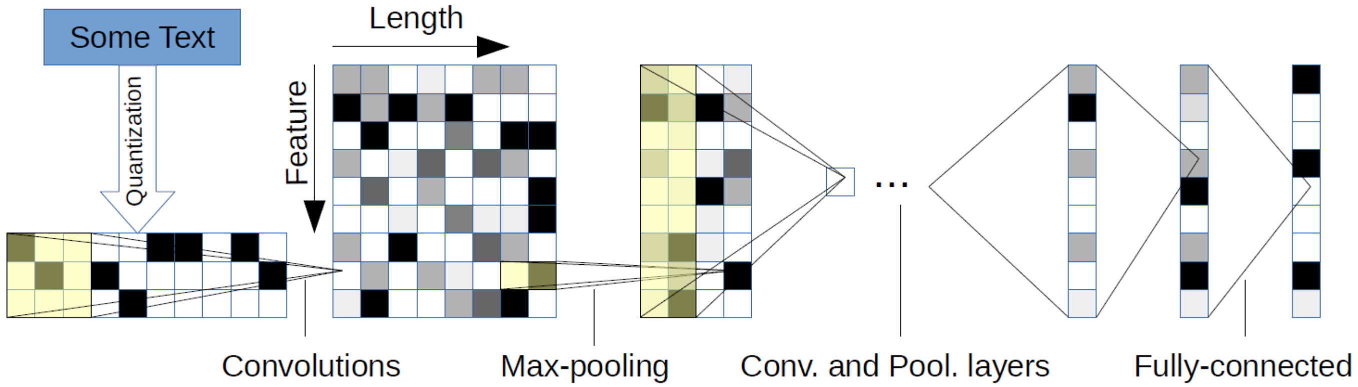


3n inner representation of the text
softmax outputs for the 6 classes
six
dead
as
suicide
bombing
hits
Somalia
#terroratt ack
m x k pre-trained word embedding representation of text
convolutional layer with stacked filters of different sizes
max-pooling layer
Fully-connected layer with softmax outputs

2d convolution on characters

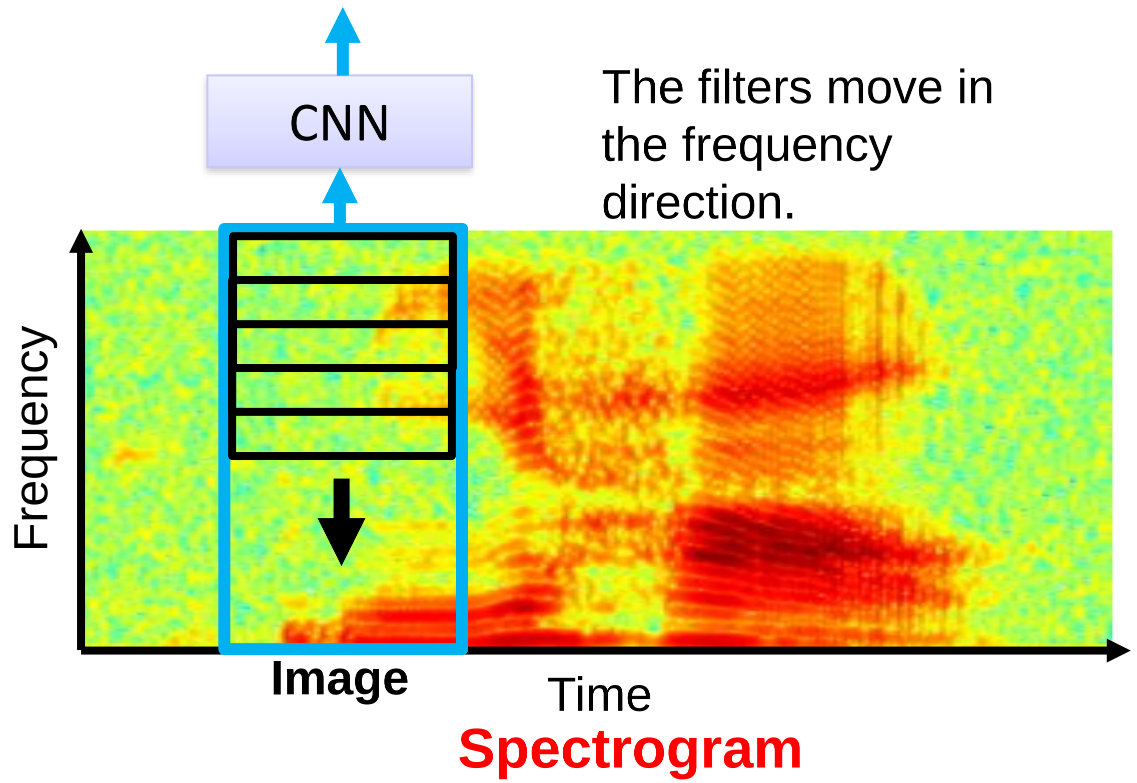
- text classification
- Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." Advances in Neural Information Processing Systems. 2015.
- convolution, max-pooling
- ReLU activation h(x) = max(0, x)
- backpropagation with momentum 0.9
- minibatch =128, starting step 0.01 is halved every 3 epoch
- character quantization, alphabet of 70 characters
  abcdefghijklmnopqrstuvwxyz0123456789
  -,;.!?:’’’/\|_@#$%ˆ&*˜‘+-=<>()[]{}
- one-hot encoding of characters

2d text convolution network architecture



- blocks of 1014 characters (i.e. 1014 x 70)
- 6 convolutional layers with filter lengths 7, 7, 3, 3, 3, 3
- stride = 1
- between fully connected layers: dropout with p=0.5
- good results on very large datasets >10^6
- for smaller datasets bag of n-grams is very competitive

CNN in speech recognition



The filters move in the frequency direction.