

# Recurrent and convolutional networks

## Statistical Methods in NLP 2

### ISCL-BA-08

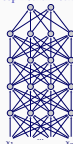
Çağrı Çöltekin  
ccoltekin@uni-tuebingen.de

University of Tübingen  
Seminar für Sprachwissenschaft

Summer Semester 2026

version: 16/07/2021 - 10:22:30 AM '21

# Deep neural networks



- Deep neural networks have recently been successful in many tasks
- They often use sparse connectivity and shared weights
- We will focus on two important architectures: recurrent and convolutional networks

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 1 / 28

## Why deep networks?

- We saw that a feed-forward network with a single hidden layer is a *universal approximator*
- However, this is a theoretical result – it is not clear how many units one may need for the approximation
- Successive layers may learn different representations
- Deeper architectures have been found to be useful in many tasks

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 2 / 28

## Why now?

- Increased computational power, especially advances in graphical processing unit (GPU) hardware
- Availability of large amounts of data
  - mainly unlabeled data (more on this later)
  - but also labeled data through ‘crowd sourcing’ and other sources
- Some new developments in theory and applications

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 3 / 28

## Recurrent neural networks

- Feed forward networks
  - can only learn associations
  - do not have memory of earlier inputs
  - if used for sequences, learning strongly depends on location of items
- Recurrent neural networks model sequences
- This is achieved by ‘recurrent’ connections in the network

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 4 / 28

## Recurrent neural networks



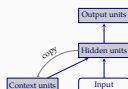
- Recurrent neural networks are similar to the standard feed-forward networks
- They include ‘loops’ that use previous output (of the hidden layers) as well as the input
- Forward calculation is straightforward, learning becomes somewhat tricky

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 5 / 28

## A simple version: SRNs

Elman (1990)



- The network keeps previous hidden states (context units)
- The rest is just like a feed-forward network
- Training is simple, but cannot learn long-distance dependencies

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 6 / 28

## Processing sequences with RNNs

- RNNs process sequences one unit at a time
- The earlier inputs affect the output through recurrent links



Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 7 / 28

## Learning in recurrent networks



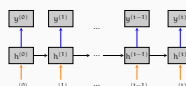
- We need to learn three sets of weights:  $W_0$ ,  $W_1$  and  $W_2$
- Backpropagation in RNNs are at first not that obvious
- The main difficulty is in propagating the error through the recurrent connections

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 8 / 28

## Unrolling a recurrent network

Back propagation through time (BPTT)



Note: the weights with the same color are shared.

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 9 / 28

## Unstable gradients

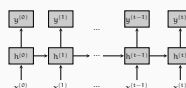
- A common problem in deep networks is *unstable gradients*
- The partial derivatives with respect to weights in the early layers calculated using the chain rule
- A long chain of multiplications may result in
  - *vanishing gradients* if the values are in range  $[-1, 1]$
  - *exploding gradients* if absolute values larger than 1
- A practical solution for exploding gradients is called *gradient clipping*
- The solution to vanishing gradients is more involved (coming soon)

Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 10 / 28

## RNN architectures

Many-to-many (e.g., POS tagging)

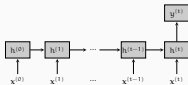


Ç. Çöltekin, SR | University of Tübingen

Summer Semester 2026 11 / 28

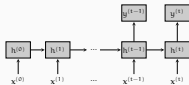
## RNN architectures

Many-to-one (e.g., document classification)

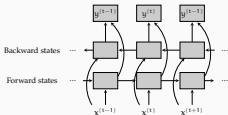


## RNN architectures

Many-to-many with a delay (e.g., machine translation)



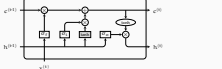
## Bidirectional RNNs



## Unstable gradients revisited

- We noted earlier that the gradients may *vanish* or *explode* during backpropagation in deep networks
- This is especially problematic for RNNs since the effective depth of the network can be extremely large
- Although RNNs can theoretically learn long-distance dependencies, this is affected by unstable gradients problem
- The most popular solution is to use *gated* recurrent networks

## Gated recurrent networks



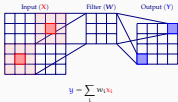
- Most modern RNN architectures are 'gated'
- The main idea is learning a mask that controls what to remember (or forget) from previous hidden layers
- Two popular architectures are
  - Long short term memory (LSTM) networks (above)
  - Gated recurrent units (GRU)

## Convolutional networks

- Convolutional networks are particularly popular in image processing applications
- They have also been used with success some NLP tasks
- Unlike feed-forward networks we have discussed so far,
  - CNNs are not fully connected
  - The hidden layer(s) receive input from only a set of neighboring units
  - Some weights are shared
- A CNN learns features that are *location invariant*
- CNNs are also computationally less expensive compared to fully connected networks

## Convolution in image processing

- Convolution is a common operation in image processing for effects like edge detection, blurring, sharpening, ...
- The idea is to transform each pixel with a function of the local neighborhood



## Example convolutions

- Blurring

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

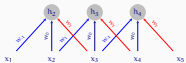
- Edge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

## Learning convolutions

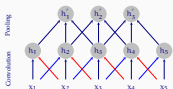
- Some filters produce features that are useful for classification (e.g., of images, or sentences)
- In machine learning we want to *learn* the convolutions
- Typically, we learn multiple convolutions, each resulting in a different feature map
- Repeated application of convolutions allow learning higher level features
- The last layer is typically a standard fully-connected classifier

## Convolution in neural networks



- Each hidden unit corresponds to a local window in the input
- Weights are shared: each convolution detects the same type of features

## Pooling



- Convolution is combined with *pooling*
- Pooling 'layer' simply calculates a statistic (e.g., max) over the convolution layer
- Location invariance comes from pooling

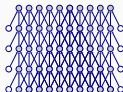
## Pooling and location invariance



- Note that the numbers at the pooling layer are stable in comparison to the convolution layer

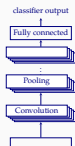
## Padding in CNNs

- With successive layers of convolution and pooling, the later layers shrink
- One way to avoid this is *padding* the input and hidden layers with enough number of zeros



## CNNs: the bigger picture

- At each convolution/pooling step, we often want to learn multiple feature maps
- After a (long) chain of hierarchical feature maps, the final layer is typically a fully-connected layer (e.g., softmax for classification)



## Real-world examples are complex



- The real-world CNNs tend to be complex
- Many layers (sometimes with repetition)
  - Large amount of branching

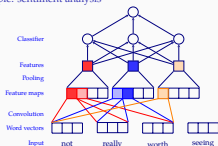
\* Diagram describes an image classification network, GoogleNet (Szegedy et al. 2016).

## CNNs in natural language processing

- The use of CNNs in image applications is rather intuitive
  - the first convolutional layer learns local features, e.g., edges
  - successive layers learn more complex features that are combinations of these features
- In NLP, it is a bit less straight-forward
  - CNNs are typically used in combination with word vectors
  - The convolutions of different sizes correspond to (word) n-grams of different sizes
  - Pooling picks important 'n-grams' as features for classification



## An example: sentiment analysis



## Some (important) architectures we did not cover (yet)

- It is common to use RNNs (and other networks) in combination with *attention*
  - Instead of relying on the (final) representation built for the whole input sequence, selectively relevant intermediate representations in the decoder
- Transformers: no recurrent networks, rely mainly on attention mechanism
  - These networks became the standard for the last couple of years, mainly because they can be trained on large amounts of data in parallel (using many GPUs)

## Summary

RNN models of sequences with (short term) memory  
CNN shared feed-forward weights, location invariance

- Reading: Jurafsky and Martin (2026, Chapter 8)

Next:

- Introduction to language models
- Reading: Jurafsky and Martin, 2026, Chapter 3

## References & further reading

- Jurafsky, Daniel and James H. Martin (2026). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released January 6, 2026. URL: <https://web.stanford.edu/~jurafsky/slp3/>.