Graph Neural Networks in Biology: Lecture 2

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
April 22, 2022

Contents Today

- Graph Neural Networks: Definition and Simple Examples
- Convolutional Neural Networks

Graph Neural Networks: Definition

Graph Neural Networks: Definition

Definition [Graph Neural Network]:
A *graph neural network (GNN)* is an

- optimizable transformation on
- all attributes of the graph (nodes, eges, global) that
- preserves graph symmetries (permutation invariances)

In the following, we will build GNN's

- using the *message passing neural network* framework proposed by [Gilmer et al., 2017]
- using the *Graph Nets architecture* introduced by [Battaglia et al., 2018].

Graph Neural Networks: Definition

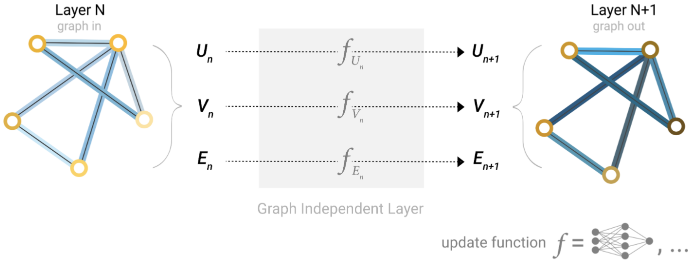
Definition [Graph Neural Network]:
A *graph neural network (GNN)* is an

- optimizable transformation on
- all attributes of the graph (nodes, eges, global) that
- preserves graph symmetries (permutation invariances)

- GNN's adopt a "graph-in, graph-out" architecture:
  - Graph loaded with information accepted as input
  - Embeddings are progressively transformed
  - Connectivity of input graph never changed

Simple Graph Neural Networks

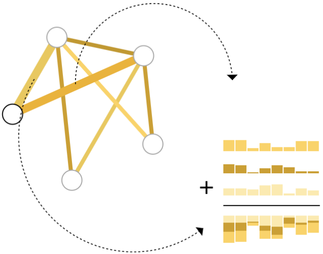
SIMPLE GNN I



U_n, V_n, E_n reflect global, vertex, edge information.
From `https://distill.pub/2021/gnn-intro/`

- Initial embeddings: U_0, V_0, E_0
- $U_n, V_n, E_n, n \geq 0$ iteratively updated to $U_{n+1}, V_{n+1}, E_{n+1}$ ...
- ... using multilayer perceptions (MLP's) $f_{U_n}, f_{V_n}, f_{E_n}$ until ...
- ... final layer is reached, where final embeddings are computed.

PREDICTIONS BY POOLING

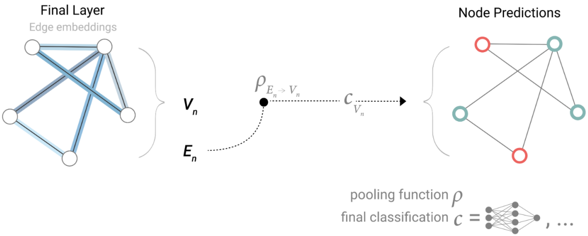


From `https://distill.pub/2021/gnn-intro/`

- May not always be so simple. For example:
  - Would like to raise predictions about nodes
  - But only edge embeddings available
- *Solution:* Aggregate (adjacent) edge embeddings using pooling function

$$\rho_{E_n \to V_n}$$

Predictions by Pooling II

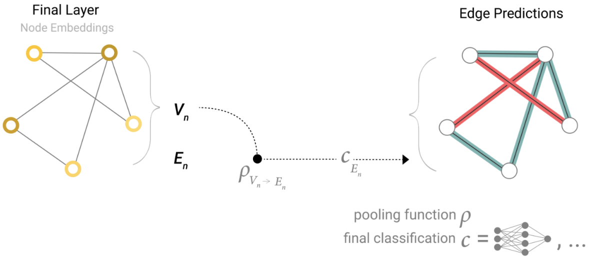


Aggregating edge embeddings for raising node predictions

From `https://distill.pub/2021/gnn-intro/`

- Pooling function $\rho_{E_n \to V_n}$ enables node predictions from edge embeddings

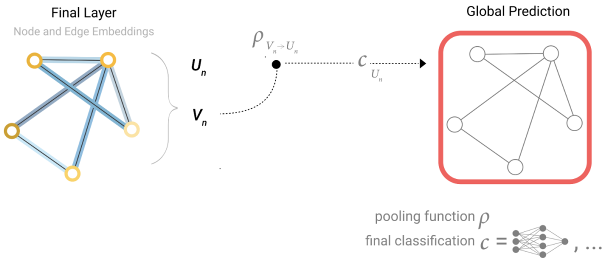
Predictions by Pooling III



Aggregating node embeddings for raising edge predictions
From `https://distill.pub/2021/gnn-intro/`

- $\rho_{V_n \rightarrow E_n}$ enables edge predictions from node embeddings
- *Example:* Predict neighboring nodes maintaining particular relationship

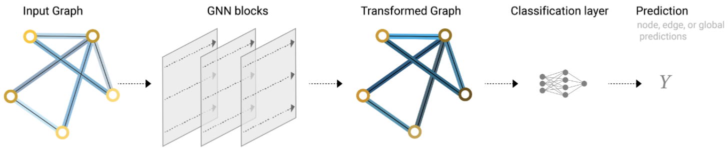
Predictions by Pooling IV



Aggregating node embeddings for raising global prediction
From `https://distill.pub/2021/gnn-intro/`

- $\rho_{V_n \to U_n}$ enables prediction about entire graph from node embeddings
- *Example:* Predict toxicity of molecule from information about atoms

PREDICTIONS BY POOLING V



GNN: End-to-end predcition task

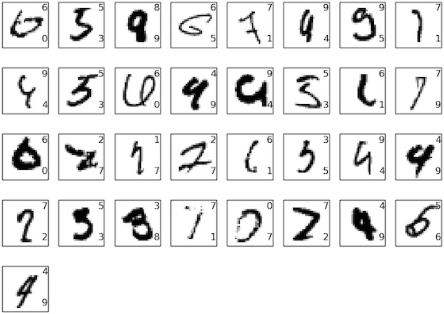
From `https://distill.pub/2021/gnn-intro/`

- Classification layer comprises pooling as well, if necessary
- *Remark:* Classification model can be any differentiable model
  - Models other than MLP's conceivable

Convolutional Neural Networks (CNNs)

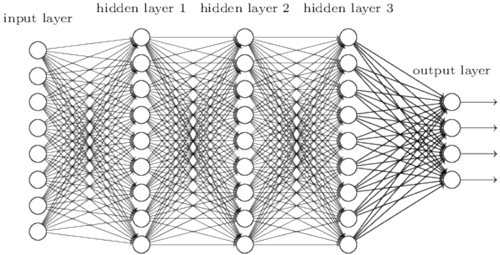
Goal

Setting up a neural network that correctly classifies 9967 out of 10 000 images; see below for the 33 misclassified ones.



33 misclassified images; correct/predicted classification upper/lower right corner

Fully Connected Networks



Fully connected neural network with 3 hidden layers

Issue: With fully connected NN's, we only reach about 98% accuracy in prediction.

Question: How to get to 99,67% accuracy?

Convolutional Neural Networks

Motivation

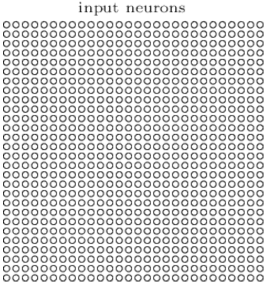
- Use that images have a spatial structure
  - ☞ Neighboring pixels are more likely to belong to the same structural elements
- Exploit this to speed up training, and reduce number of parameters (weights)

Basic Ideas

- Local receptive fields
- Shared weights
- Pooling

Convolutional Neural Networks

Local Receptive Fields



One image are 28 x 28 = 784 pixels

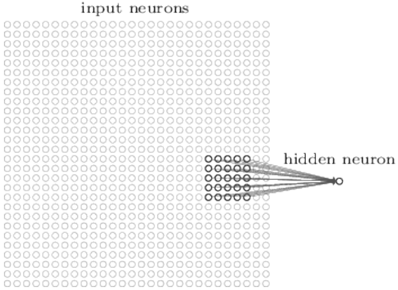
In a fully connected network

- Every node of the first hidden layer is connected to every input neuron (a.k.a pixel)
- Every node of the second layer is connected to every neuron in the first hidden layer

Convolutional Neural Networks
Local Receptive Fields

In a convolutional NN,

- Every node in the first hidden layer is connected to a rectangular subregion
- Here: subregion = square of 5x5=25 input neurons

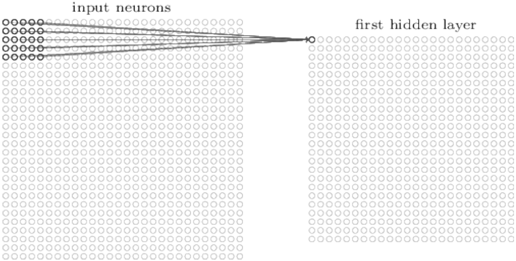


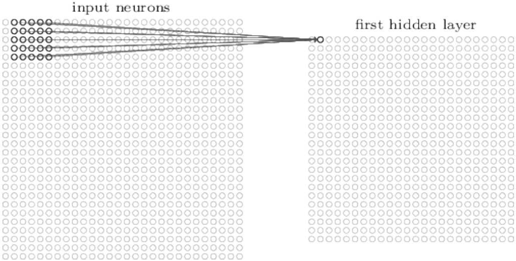
Convolutional filter of size 5 x 5

Definition
The region in the input images to which a hidden neuron is connected is called the *local receptive field (LRF)* of the hidden neuron.

Convolutional Neural Networks

Local Receptive Fields



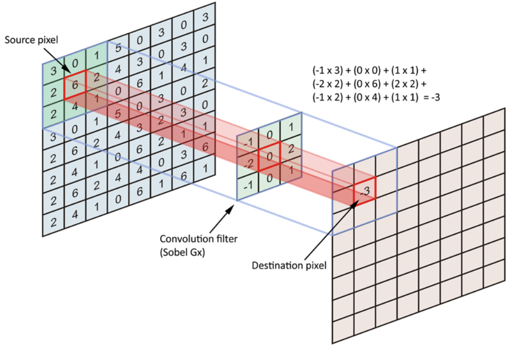


- One *hidden layer* is generated by one pass of the LRF
- Several hidden layers will be generated by several passes of the LRF
- The activation a_{jk}^{l+1} of the j,k-th hidden neuron within the layer, using a $M \times M$ LRF, is computed as (σ may represent activation function of choice)

$$a_{jk}^{(l+1)} = \sigma(b + \sum_{l=0}^{M} \sum_{m=0}^{M} w_{l,m} a_{j+l,k+m}^{l}) \tag{1}$$

Convolutional Neural Networks

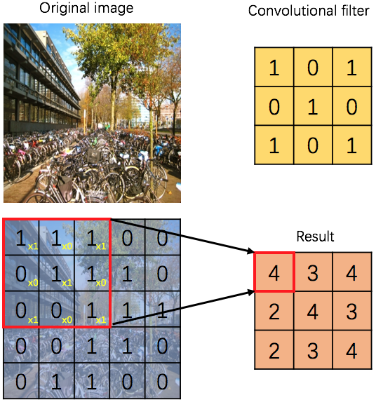
Convolutional Filters



For generating one hidden layer, identical parameters, together defining one convolutional filter, are used

Convolutional Neural Networks

Convolutional Filters

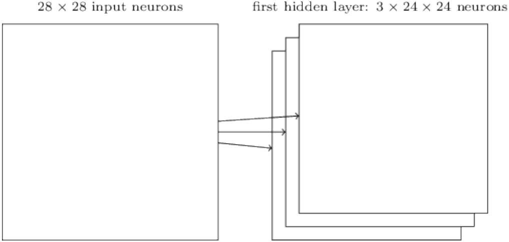


For generating one hidden layer, identical parameters, together defining one convolutional filter, are used

Convolutional Neural Networks

Convolutional Filters

Definition
A *feature map* is a mapping associated with one convolutional filter.



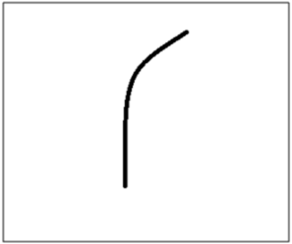
- A complete convolutional layer consists of several hidden sublayers
- Each sublayer is defined by one feature map

NEURAL NETWORKS

CONVOLUTION FILTERS

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



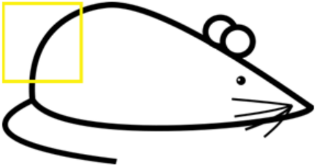
Visualization of a curve detector filter

Filter for recognizing a curve

Neural Networks

Convolution Filters

Original image



Visualization of the filter on the image

Neural Networks

Convolution Filters

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

$*$

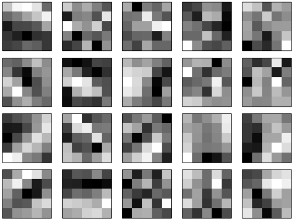
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Convolutional Neural Networks

Convolutional Filters Real World Example



MNIST example, 20 different filters

- The darker the more positive, the whiter the more negative
- In reality, convolutional filters are hard to interpret
- Literature: M.D. Zeiler, R. Fergus, “Visualizing and Understanding Convolutional Networks”, `https://arxiv.org/abs/1311.2901`

Convolutional Neural Networks

Shared Weights and Biases

- *Reminder*: The activation a_{jk}^{l+1} of the j, k-th hidden neuron within the layer, using a $M \times M$ LRF, is computed as (σ may represent activation function of choice)

$$a_{jk}^{l+1} = \sigma(b + \sum_{l=0}^{M} \sum_{m=0}^{M} w_{l,m} a_{j+l,k+m}^{l}) \tag{2}$$

- *Observation:* For each node in the same hidden layer, the same parameters $w_{l,m}, 1 \leq l, m \leq M$ are used
- That is, we only need $M \times M$ parameters to generate the entire hidden layer

MNIST example

:

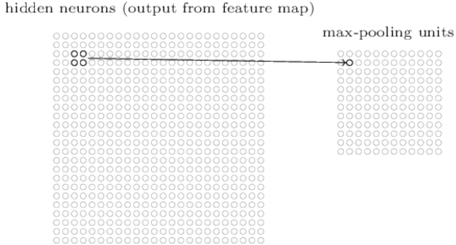
- Convolutional layer, 20 feature maps, each of size 5×5, roughly requires $20 \times 5 \times 5 = 500$ weights
- Fully connected network, connecting 784 input neurons with 30 hidden neurons requires $784 \times 30 = 23\,520$ weights
- **CNN requires roughly 40 times less parameters**

- *Remark*: Sometimes it helps to think of a convolutional layer, as a new type of image, where each sublayer refers to a different color.
- Note that colored pictures of size $N \times N$ come in 3 input layers of size $N \times N$, each of which refers to one of the 3 base colors red, green and blue.
- So, when using $M \times M$-filters, one applies a $3 \times M \times M$ sized *tensor* (and not an $M \times M$-sized matrix) to the input layer
- This principle can later be repeated: hence the name *tensor flow*.

- In addition to convolutional layers, CNN's make use of *pooling layers*.
- Pooling layers generate *condensed feature maps*: it takes a rectangle of neurons, and summarizes their values into one value
- This generates a considerably smaller layer

CONVOLUTIONAL NEURAL NETWORKS

POOLING LAYERS

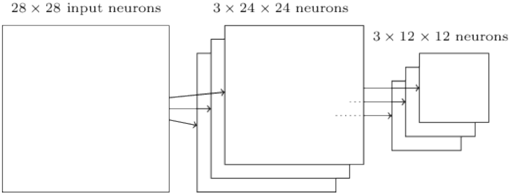


2×2 pooling

- *Max pooling*: Each $L \times L$ rectangle is mapped onto the maximum of its values
- *L2 pooling*: Each $L \times L$ rectangle is mapped to the rooted average of the squares of the values
- This overall yields a layer that is $L \times L$ times smaller
- Usually $L = 2$ is used

Convolutional Neural Networks

Combining Convolutional and Pooling Layers

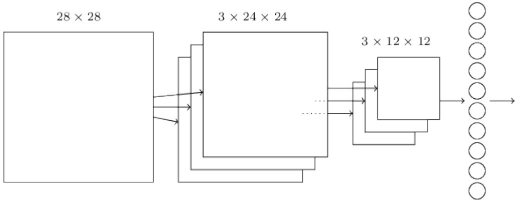


Convolutional layer followed by pooling layer

- Convolutional and pooling layers are used in combination
- Pooling layers usually follow convolutional layers
- *Intuition*:
  - The exact location of the occurrence of a feature is not important
  - Pooling helps to handle distortions and rotations

Convolutional Neural Networks

A Complete CNN



Convolution followed by pooling followed by fully connected output layer

- 10 output nodes, one for each digit
- Each output node is connected to *every* node of the pooling layer
- *Training*: Stochastic gradient descent plus backpropagation

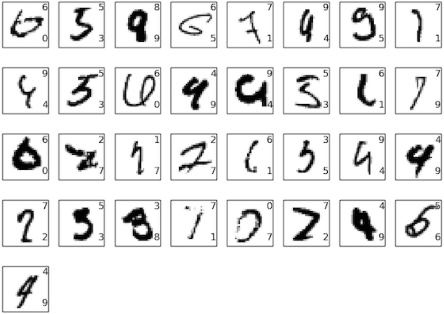
Ensemble of networks: Idea

- Train several different networks
- For example, employ repeated random initialization while always using the same architecture
- For classification, take the majority vote of the different networks
- While each network performs similarly, the majority vote may yield improvements
- Here: 5 randomly initialized network of the architecture o described in the slides before
- Accuracy: 99.67%
- That has been our goal!

CNNs in Practice

Ensemble of Networks

- Ensemble of 5 randomly initialized networks
- Architecture as described in the slides before
- Accuracy: 99.67% – that has been our goal!



33 misclassified images; correct/predicted classification upper/lower right corner

CNNs in Practice

References

- Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf` [Architecture: “LeNet-5”]

CNNs on MNIST

Further Improvements

- For further improvements on MNIST (and on famous datasets in general see `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`
- *Noteworthy*:
  - See D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber, "Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition", `https://arxiv.org/abs/1003.0358`
  - Fully connected network, without convolutional layers that achieves 99.65% accuracy.
  - Training for that non-convolutional network proceeds very slow, however.

OUTLOOK

- Message Passing
- Convolution on Graphs

Thanks for your attention!