

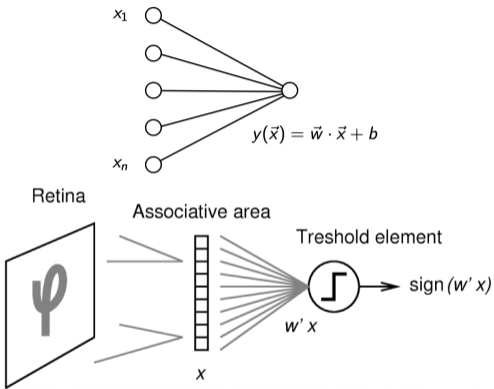
Introduction to Data Analysis and Machine Learning in Physics:

5. Convolutional Neural Networks and Graph Neural Networks

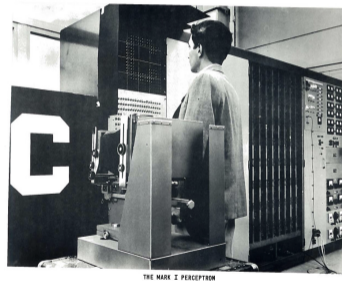
Jörg Marks, Klaus Reyers

Studierendentage, 11-14 April 2023

Historical perspective: Perceptron (1)



$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$



Mark 1 Perceptron. Frank Rosenblatt (1961)

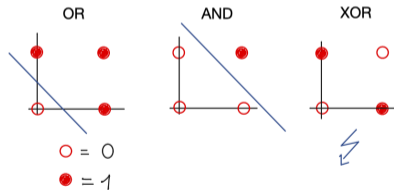
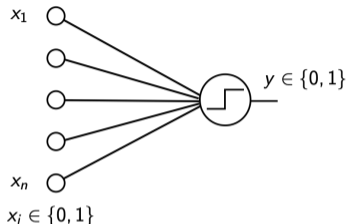
The perceptron was designed for image recognition. It was first implemented in hardware (400 photocells, weights = potentiometer settings).

Historical perspective: Perceptron (2)

- McCulloch–Pitts (MCP) neuron (1943)
 - ▶ First mathematical model of a biological neuron
 - ▶ Boolean input
 - ▶ Equal weights for all inputs
 - ▶ Threshold hardcoded
- Improvements by Rosenblatt
 - ▶ Different weights for inputs
 - ▶ Algorithm to update weights and threshold given labeled training data

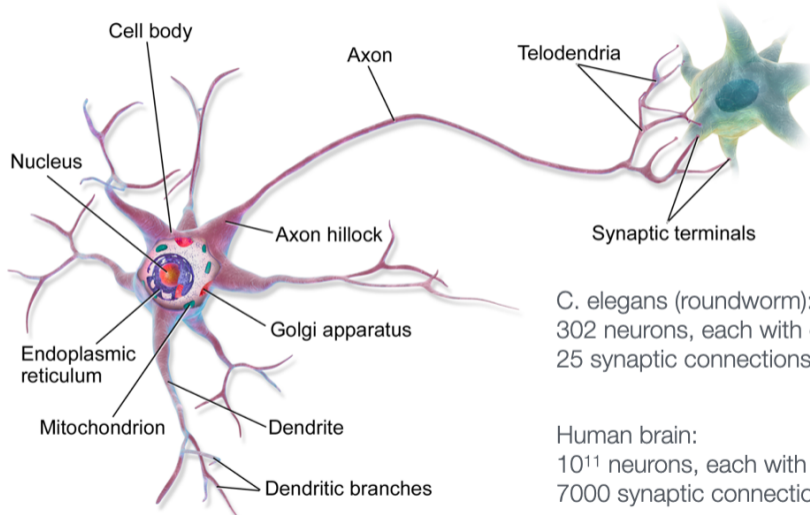
Shortcoming of the perceptron:
it cannot learn the XOR function

Minsky, Papert, 1969



XOR: not linearly separable

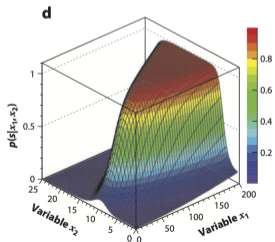
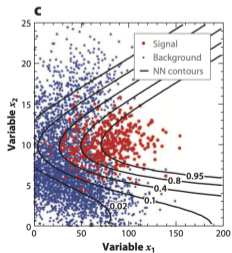
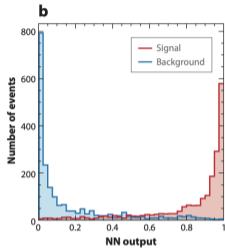
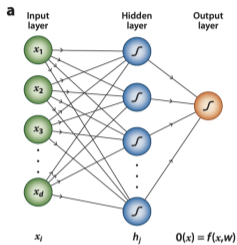
The biological inspiration: the neuron



C. elegans (roundworm):
302 neurons, each with on average
25 synaptic connections

Human brain:
 10^{11} neurons, each with on average
7000 synaptic connections

Neural network output and decision boundaries



P. Bhat, Multivariate Analysis Methods in Particle Physics, [inspire-hep.net/record/879273](https://inspirehep.net/record/879273)

Recap: Backpropagation

Backpropagation summary

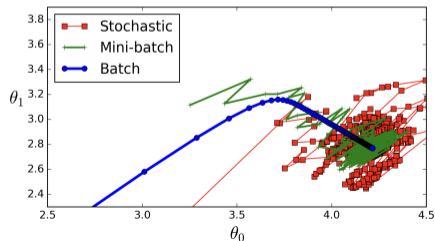
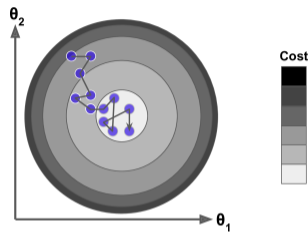
- Make prediction for a given training instance (forward pass)
- Calculate error (value of loss function)
- Go backwards and determine the contribution of each weight (reverse pass)
- Adjust the weights to reduce the error

Practical considerations:

- Nowadays, people will implements neural networks with frameworks like Keras or TensorFlow
- No need to implement backpropagation yourself
- TensorFlow efficiently calculates gradient function efficiently ('autodiff')

More on gradient descent

- Stochastic gradient descent
 - ▶ just uses one training event at a time
 - ▶ fast, but quite irregular approach to the minimum
 - ▶ can help escape local minima
 - ▶ one can decrease learning rate to settle at the minimum (“simulated annealing”)
- Batch gradient descent
 - ▶ use entire training sample to calculate gradient of loss function
 - ▶ computationally expensive
- Mini-batch gradient descent
 - ▶ calculate gradient for a random sub-sample of the training set

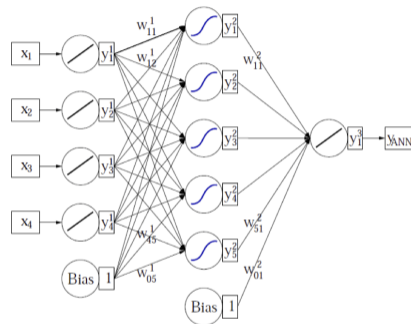


Universal approximation theorem

“A feed-forward network with a single hidden layer containing a finite number of neurons (i.e., a multilayer perceptron), can approximate continuous functions on compact subsets of \mathbb{R}^n .”

One of the first versions of the theorem was proved by George Cybenko in 1989 for sigmoid activation functions

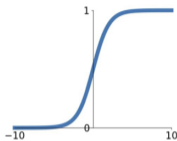
The theorem does not touch upon the algorithmic learnability of those parameters



Recap: Activation functions

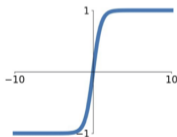
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



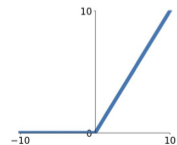
tanh

$$\tanh(x)$$



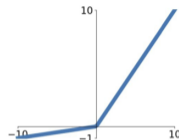
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

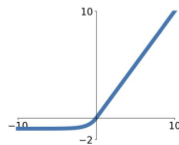


Maxout

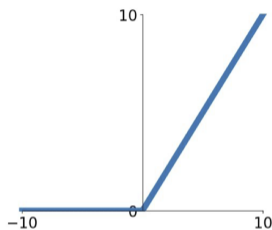
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



ReLU



ReLU (Rectified Linear Unit)

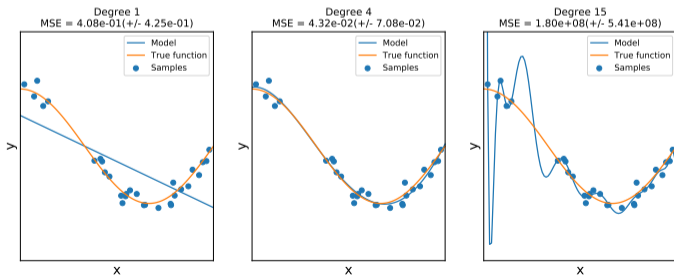
$$f(x) = \max(0, x)$$

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid tanh in practice
- Actually more biologically plausible than sigmoid
- But: gradient vanishes for $x < 0$

Bias-variance tradeoff

Goal: generalization of training data

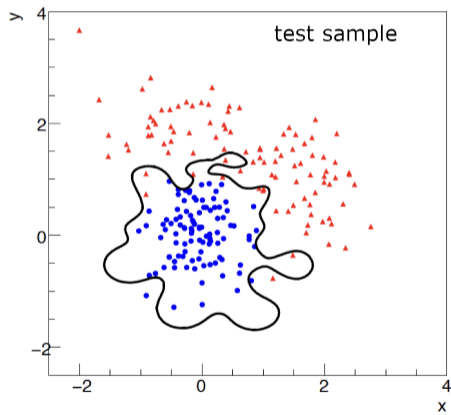
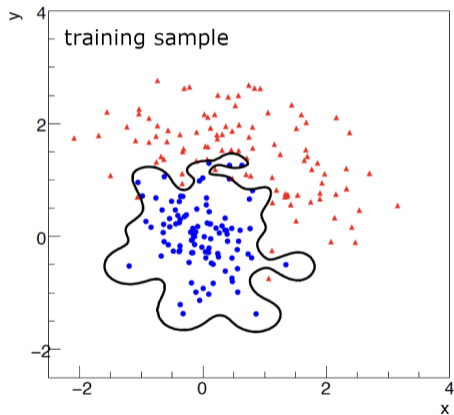
- Simple models (few parameters): danger of bias
 - ▶ Classifiers with a small number of degrees of freedom are less prone to statistical fluctuations: different training samples would result in similar classification boundaries ("small variance")
- Complex models (many parameters): danger of overfitting
 - ▶ large variance of decision boundaries for different training samples



Recap: Overtraining

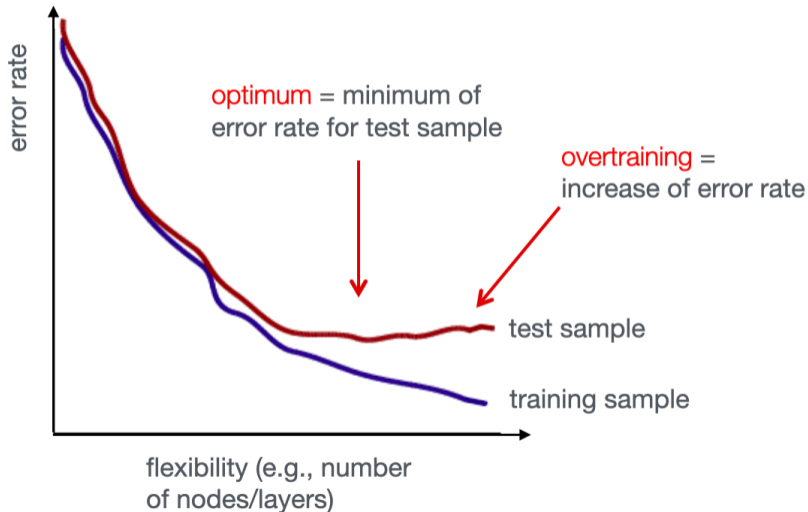
Too many neurons/layers make a neural network too flexible

→ overtraining



Monitoring overtraining

Monitor fraction of misclassified events (or loss function:)



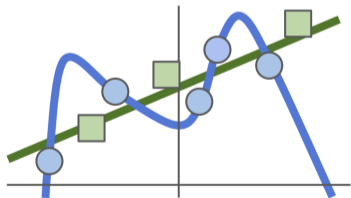
Regularization: Avoid overfitting

<http://cs231n.stanford.edu/slides>

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Model should be “simple”, so it works on test data



Occam's Razor:
“Among competing hypotheses, the simplest is the best”
William of Ockham, 1285 - 1347

L_1 regularization: $R(W) = \sum_k |W_k|$, L_2 regularization: $R(W) = \sum_k W_k^2$

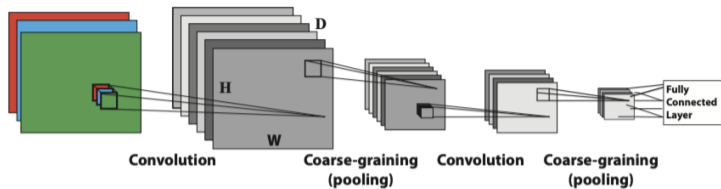
Exercise 1: Hyperparameter optimization

05_neural_networks_ex_1_hyperparameter_optimization.ipynb

The multi-layer perceptron did not perform well on the superconductivity dataset. Can you find better hyperparameters (number of hidden layers, neurons per layer, loss function, learning rate...)?

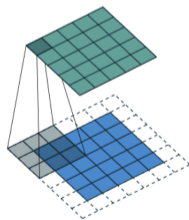
Use `sklearn.model_selection.GridSearchCV` to find optimal parameters.

Convolutional neural networks (CNNs): Overview



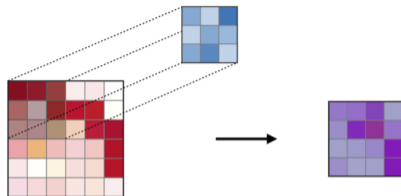
- CNNs emerged from the study of the visual cortex
- Behind many deep learning successes (e.g. in image recognition)
- Partially connected layers
 - ▶ Fully connected layers impractical for large images (too many neurons, overfitting)
- Key component: Convolutional layers
 - ▶ Set of learnable filters
 - ▶ Low-level features at the first layers; high-level features at the end

Sliding 3×3 filter

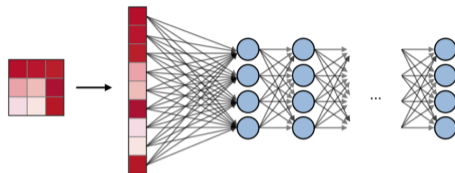


Different types of layers in a CNN

1. Convolutional layers



3. Fully connected layers



2. Pooling layers

	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	<ul style="list-style-type: none">- Preserves detected features- Most commonly used	<ul style="list-style-type: none">- Downsamples feature map- Used in LeNet

Afshine Amidi, Shervine Amidi

Convolutional Neural Networks cheatsheet

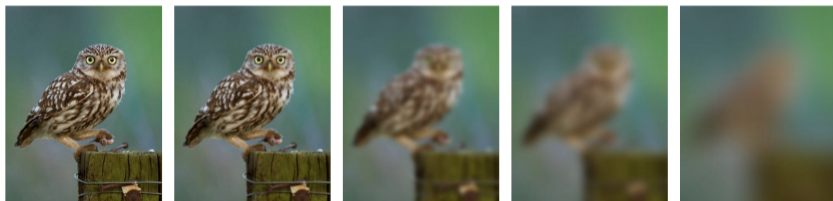
Convolution

Convolution of a function f with a kernel or filter function g :

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

Practical example: blurring of an image with a Gaussian filter

https://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec02_filter.pdf



$\sigma = 1$ pixel



$\sigma = 5$ pixels



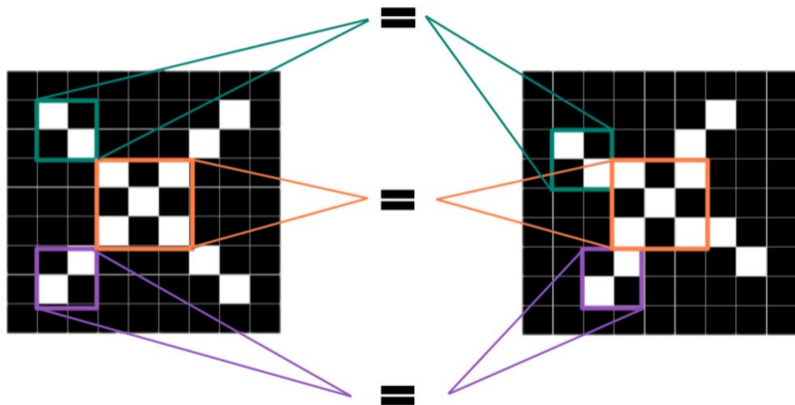
$\sigma = 10$ pixels



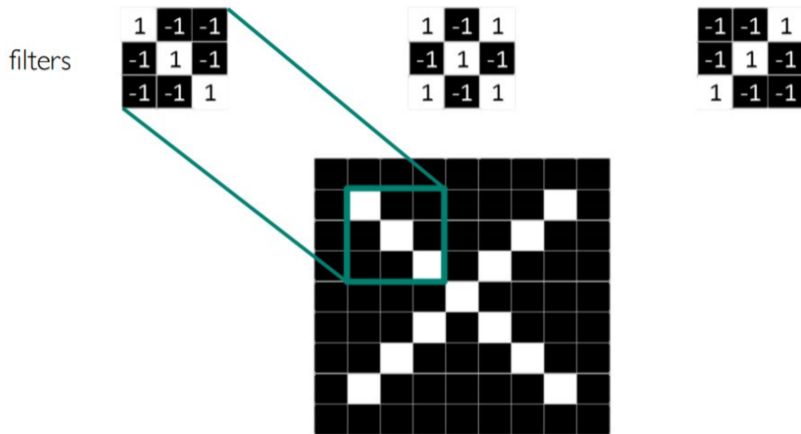
$\sigma = 30$ pixels

Filters can detect structures/features in an image (1)

Filters to detect the "X" structures:

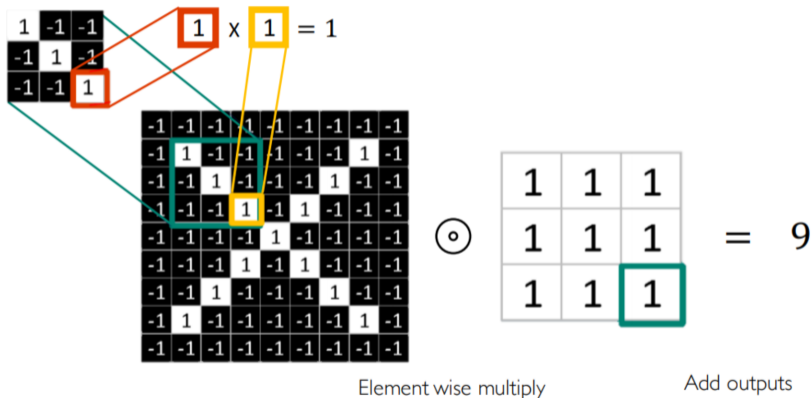


Filters can detect structures/features in an image (2)



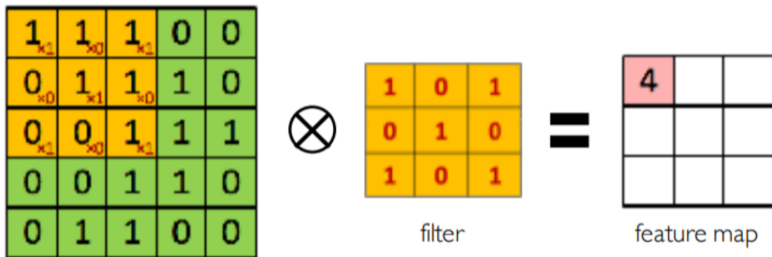
Filters can detect structures/features in an image (3)

The convolutional operation

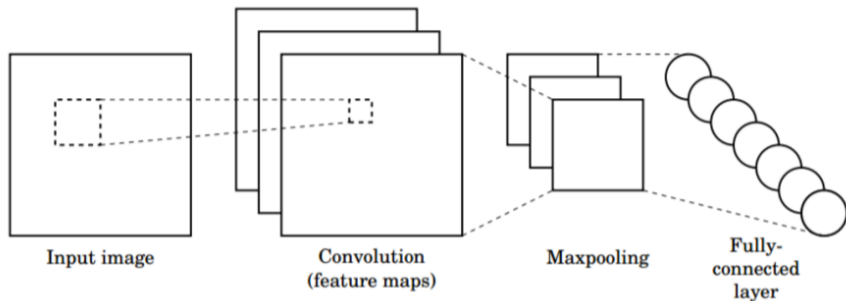


Filters can detect structures/features in an image (4)

Extracting a feature map



CNNs for classification



1. **Convolution:** Apply filters with learned weights to generate feature maps.
2. **Non-linearity:** Often ReLU.
3. **Pooling:** Downsampling operation on each feature map.

**Train model with image data.
Learn weights of filters in convolutional layers.**

AlexNet

“ImageNet Classification with Deep Convolutional Neural Networks”,
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012

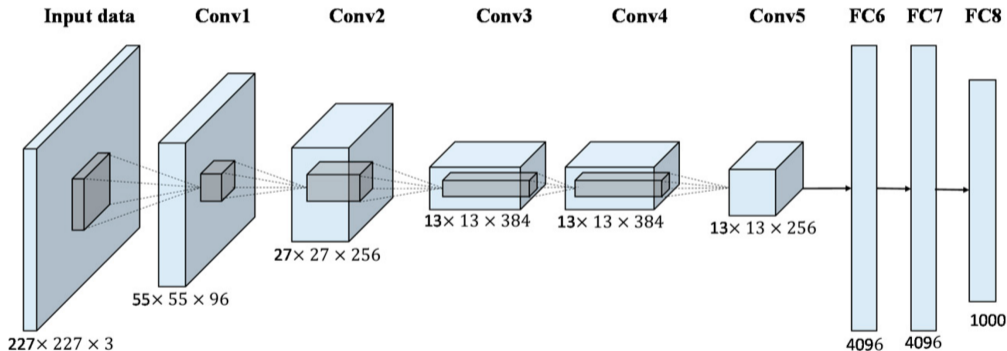


figure: <https://www.mdpi.com/2072-4292/9/8/848>

Inductive bias of a CNNs

Can you summarize the inductive bias that is associated with a convolutional neural network?

ChatGPT Mar 23 Version:

“Yes, of course! The inductive bias associated with a convolutional neural network (CNN) is that it assumes that the input data has a grid-like topology, such as images, audio spectrograms or video frames.

CNNs are designed to learn local and translation-invariant features by convolving small filters over the input data. This allows them to identify patterns in the input that are useful for classification or other tasks. The filters can be learned through backpropagation, and the output of each convolutional layer is usually downsampled (e.g., through max pooling) to reduce the dimensionality of the data.

The inductive bias of CNNs allows them to perform well on tasks involving images, audio, and video data, where the local structure of the data is important. However, it may not be as effective for other types of data that do not have a grid-like topology, such as text or graphs. In these cases, other types of neural networks or machine learning algorithms may be more appropriate.”

MNIST classification with a CNN in Keras

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D, Conv2D, Input

# conv layer with 8 3x3 filters
model = Sequential(
    [
        Input(shape=input_shape),
        Conv2D(8, kernel_size=(3, 3), activation="relu"),
        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(16, activation="relu"),
        Dense(num_classes, activation="softmax"),
    ]
)

model.summary()
```

Defining the CNN in Keras (2)

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 8)	80
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 8)	0
flatten_1 (Flatten)	(None, 1352)	0
dense_2 (Dense)	(None, 16)	21648
dense_3 (Dense)	(None, 10)	170

Total params: 21,898

Trainable params: 21,898

Non-trainable params: 0

Model definition

Using Keras, you have to compile a model, which means adding the loss function, the optimizer algorithm and validation metrics to your training setup.

```
model.compile(loss="categorical_crossentropy",  
              optimizer="adam",  
              metrics=["accuracy"])
```

Model training

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint(
    filepath="mnist_keras_model.h5",
    save_best_only=True,
    verbose=1)
early_stopping = EarlyStopping(patience=2)

history = model.fit(x_train, y_train, # Training data
    batch_size=200, # Batch size
    epochs=50, # Maximum number of training epochs
    validation_split=0.5, # Use 50% of the train dataset for validation
    callbacks=[checkpoint, early_stopping]) # Register callbacks
```

Exercise 2: Training a digit-classification neural network on the MNIST dataset using Keras

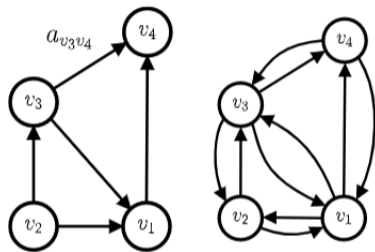
05_neural_networks_ex_2_mnist_keras_train.ipynb

- a) Plot training and validation loss as well as training and validation accuracy as a function of the number of epochs
- b) Determine the accuracy of the fully trained model.
- c) Create a second notebook that reads the trained model (`mnist_keras_model.h5`). Read `your_own_digit.png` and classify it. Create your own 28×28 pixel digits with a program like gimp and check how the model performs.
- d) Try to improve the performance of the network by increasing the number of filters and by adding a second convolutional layer.

Graph Neural Networks

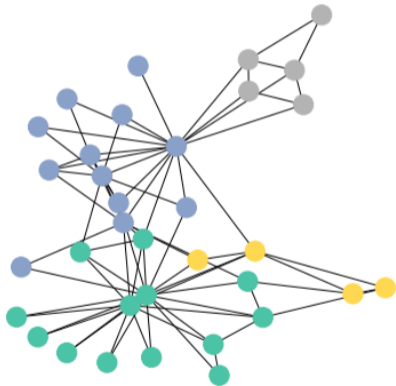
slides on GNNs by Martin Kroesen

- Graph Neural Networks (GNNs): Neural Networks that operate on graph structured data
- Graph: consists of nodes that can be connected by edges, edges can be directed or undirected
- no grid structure as given for CNNs
- node features and edge features possible
- relation often represented by adjacency matrix:
 $A_{ij} = 1$ if there is a link between node i and j , else 0
- tasks on node level, edge level and graph level
- full lecture:
<https://web.stanford.edu/class/cs224w/>

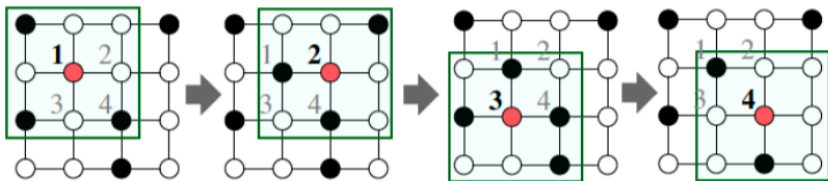


Simple Example: Zachary's karate club

- link: https://en.wikipedia.org/wiki/Zachary's_karate_club
- 34 nodes: each node represents a member of the karate club
- 4 classes: a community each member belongs to
- task: classify the nodes
- many real world problems for GNNs exist, e.g. social networks, molecules, recommender systems, particle tracks



From CNN to GNN



(from Stanford GNN lecture)

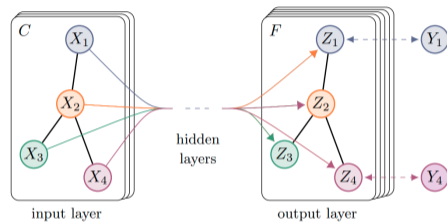
- GNN: Generalization of convolutional neural network
- No grid structure, arbitrary number of neighbors defined by adjacency matrix
- Operations pass information from neighborhood

Architecture: Graph Convolutional Network

- Message passing from connected nodes
- The graph convolution is defined as:

$$H^{(l+1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

- The adjacency matrix A including self-connections is given by \tilde{A}
- The degree matrix of the corrected adjacency matrix is given by $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
- The weights of the given layer are called $W^{(l)}$
- $H^{(l)}$ is the matrix for activations in layer l



<https://arxiv.org/abs/1609.02907>

Architecture: Graph Attention Network

- Calculate the attention coefficients e_{ij} from the features \mathbf{h} for each node i with its neighbors j

$$e_{ij} = a(W\mathbf{h}_i, W\mathbf{h}_j)$$

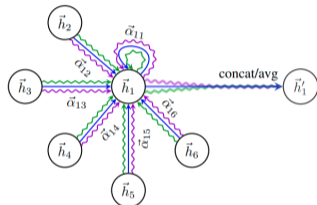
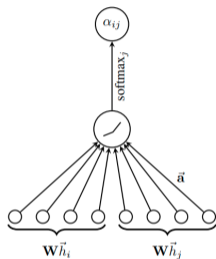
a : learnable weight vector

- Normalize attention coefficients

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

- Calculate node features

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum \alpha_{ij} W\mathbf{h}_j^l \right)$$

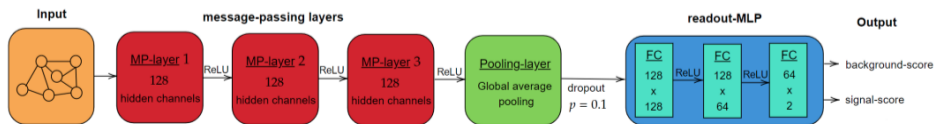
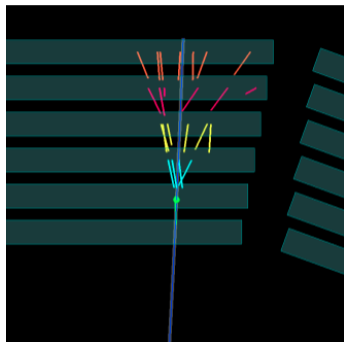


<https://arxiv.org/abs/1710.10903>

Example: Identification of inelastic interactions in TRD

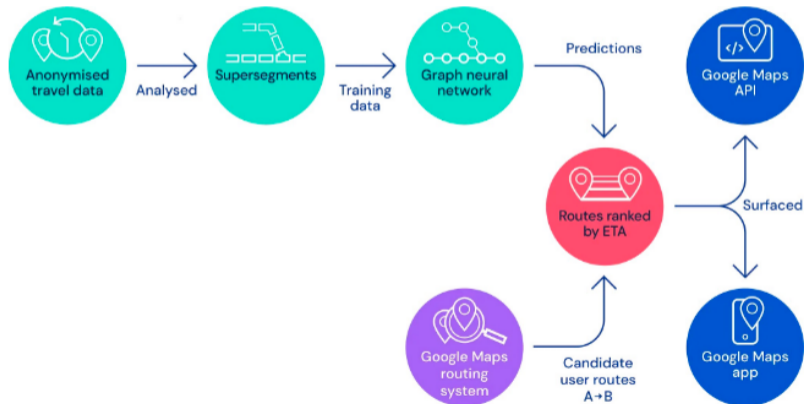
- Identification of inelastic interactions of light antinuclei in the Transition Radiation Detector in ALICE
- Thesis:
https://www.physi.uni-heidelberg.de/Publications/Bachelor_Thesis_Maximilian_Hammermann.pdf
- Construct nearest neighbor graph from signals in detector
- Use global pooling for graph classification

interaction of antideuteron:



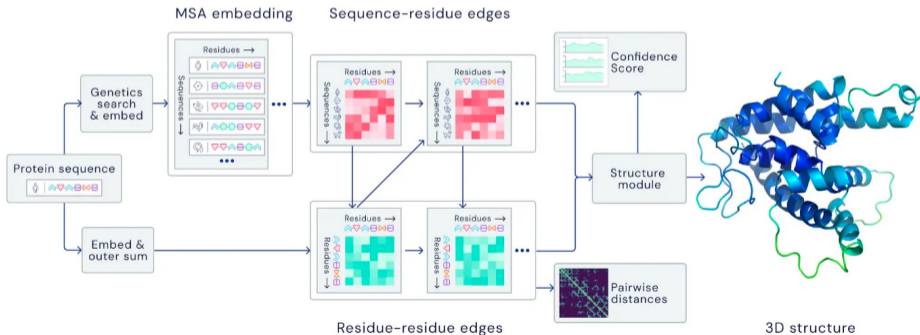
Example: Google Maps

- link: <https://www.deepmind.com/blog/traffic-prediction-with-advanced-graph-neural-networks>
- GNNs are used for traffic predictions and estimated times of arrival (ETAs)



Example: Alpha Fold

- link: <https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology>
- “A folded protein can be thought of as a ‘spatial graph’, where residues are the nodes and edges connect the residues in close proximity”



Exercise 3: Illustration of Graphs and Graph Neural Networks

On the PyTorch Geometric webpage, you can find official examples for the application of Graph Neural Networks:

https://pytorch-geometric.readthedocs.io/en/latest/get_started/colabs.html

The first introduction notebook shows the functionality of graphs with the example of the Karate Club. Follow and reproduce the first [notebook](#). Study and understand the data format.

At the end, the separation power of Graph Convolutional Networks (GCN) are shown via the node embeddings. You can replace the GCN with a Graph Attention Layers and compare the results.

Exercise 4: Classifying molecules

The PyTorch Geometric webpage also provides an example of the classification of molecules:

https://colab.research.google.com/drive/1I8a0DfQ3fl7Njc62__mVXUlcAleUclnb?usp=sharing

Study this notebook and then modify it to apply it to the PROTEINS dataset.

Practical advice – Which algorithm to choose?

From Kaggle competitions:

Structured data: “High level” features that have meaning:

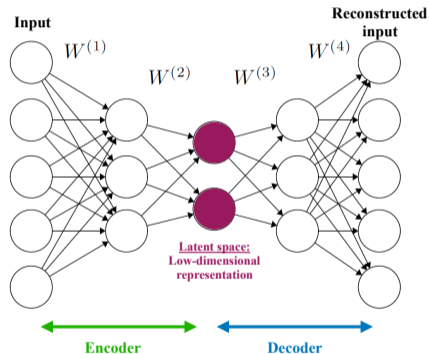
- feature engineering + decision trees
- Random forests
- XGBoost

Unstructured data: “Low level” features, no individual meaning:

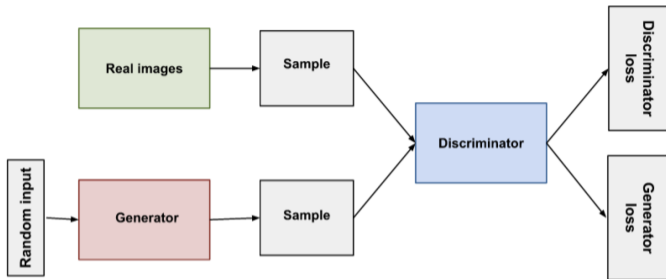
- deep neural networks
- e.g. image classification: convolutional NN

Outlook: Autoencoders

- Unsupervised method based on neural networks to learn a representation of the input data
- Autoencoders learn to copy the input to the output layer
 - ▶ low dimensional coding of the input in the central layer
- The decoder generates data based on the coding (*generative model*)
- Applications
 - ▶ Dimensionality reduction
 - ▶ Denoising of data
 - ▶ Machine translation



Outlook: Generative adversarial network (GANs)



https://developers.google.com/machine-learning/gan/gan_structure

- Discriminator's classification provides a signal that the generator uses to update its weights
- Application in particle physics: fast detector simulation
- Full GEANT simulation usually very CPU intensive

The future

“Das Interessante an unserer Intelligenz ist, dass wir Go spielen können und dann vom Tisch aufstehen und Essen machen können, was eine Maschine nicht kann.”

Bernhard Schölkopf, Max-Planck-Institut für intelligente Systeme (Interview FAZ)

“My view is throw it all away and start again”

Geoffrey Hinton (DNN pioneer) on deep neural networks and backpropagation (Interview, 2017)