EURO

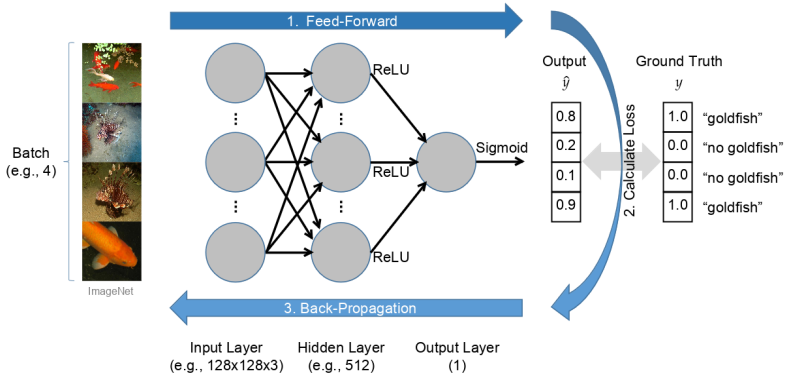
Deep Neural Network Building Blocks

Georg Zitzlsberger, IT4Innovations, 11-09-2023

Agenda

- Deep Neural Networks
- Building Blocks
  - Convolutional Neural Network Layer
  - Recurrent Neural Network Layer
  - Graph Neural Network Layer
  - Self Attention Layer
- Collateral Layers
- Summary

Deep Neural Networks



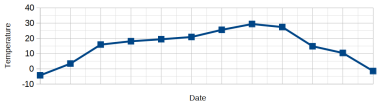
Shown here is a fully connected/dense hidden and output layer.
This is also known as feed-forward network, or Multilayer Perceptron (MLP).

Building Blocks

- Architecture defining layers:
  - Convolution Layer
  - Recurrent Neural Network Layer
  - Graph Neural Network Layer
  - Self-Attention Layer
- Collateral layers:
  - Pooling
  - Batch Normalization
  - Dropout
  - Embedding
  - ...

Convolutional Neural Network Layer

- Apply spatial filters called kernels
- Different dimensions:
  - 1D: Time series of measurements
  - 2D: Images, Matrices
  - 3D: Volumetric data
- Same dimension of input/kernel
- The kernels are learned
- Special case of fully connected
- Goal: Extract patterns from data with filters



ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

Bionerd, CC BY 3.0, via Wikimedia Commons

CNN Layer - How it works

0.20	0.10	0.50	0.80	0.10
0.40	0.60	0.20	0.80	0.50
0.20	0.40	0.30	0.90	0.20
0.50	0.80	0.10	0.20	0.70
0.80	0.70	0.90	0.50	0.50

Input

*

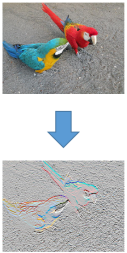
1	1	1
0	0	0
-1	-1	-1

Kernel
(e.g., 3x3)

=

-0.01	-0.02	0.00
-0.02	0.06	0.06
-0.17	-0.06	-0.06

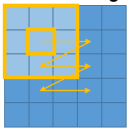
Output



Computation of first pixel...

(0.20*1 + 0.10*1 + 0.50*1 +
0.40*0 + 0.60*0 + 0.20*0 +
0.20*-1 + 0.40*-1 + 0.30*-1) / (3x3) = -0.01

...and the following ones:



CNN Layer - Pros/Cons

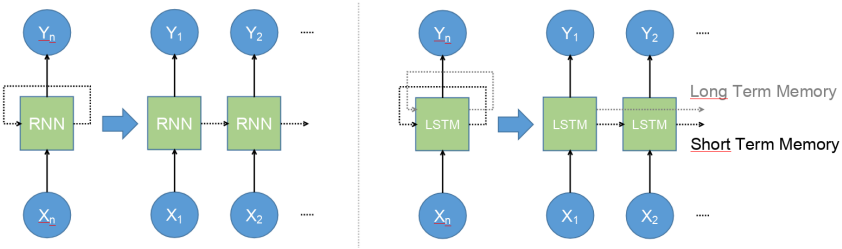
- Pro:
  - Fast and parallelizable
  - Ideal for dense data
- Con:
  - Kernel grows quadratic/cubic
  - Spatially constraint - requires pooling or dense/fully connected layers to connect

Note:
Fully convolutional networks allow the change of spatial input dimensions at all times. E.g., train with patches of 32×32 pixels and run inference with larger patches of 256×256 pixels.

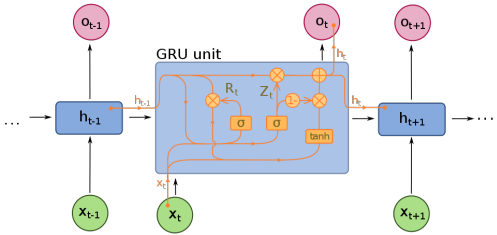
Recurrent Neural Network Layer

- Provide a sequence as input (e.g., text, video frames)
- A Recurrent Neural Network Layer (RNN) has states
- Different variants exist:
  - (Vanilla) RNN
  - Gated Recurrent Unit (GRU)
  - Long Short-Term Memory (LSTM)
- Combination with Convolution also exists (ConvLSTM)
- Goal: Learn relationships across sequences

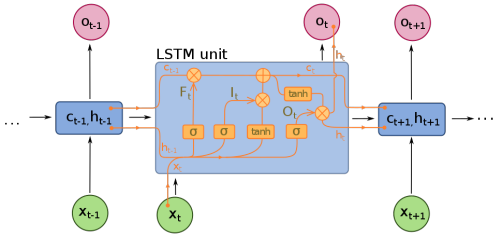
RNN Layer - How it works



RNN Layer - How it works



fdeloche, CC BY-SA 4.0, via Wikimedia Commons



fdeloche, CC BY-SA 4.0, via Wikimedia Commons fdeloche

RNN Layer - Pros/Cons

- Pro:
  - Easy to use
  - Still a go-to solution for sequenced data
- Con:
  - Slow, limited parallelization
  - Still problematic for large sequences (e.g., texts)
  - Spatial information is lost unless ConvLSTM is used

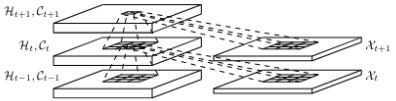


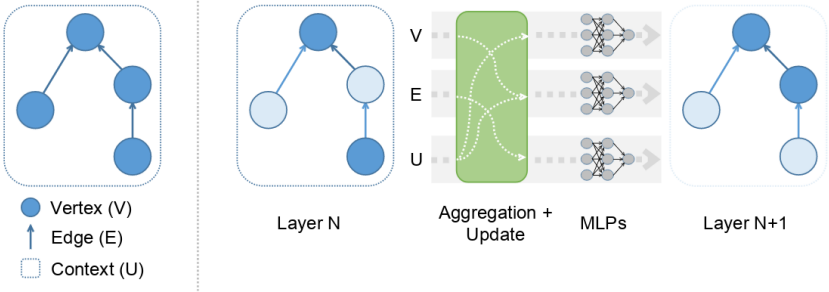
Figure 2: Inner structure of ConvLSTM

Shi et al., Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting

Graph Neural Network Layer

- Suitable for problems expressed as a graph
- Graph information is learned in:
  - Nodes/vertices (V)
  - Edges (E)
  - Graph context (U)
- Information can be exchanged between the Graph components:
  1. Aggregation:
     Select which and how component information should be combined
  2. Update:
     Combined information updates the current states of the (selected) components
  3. Learning:
     Train a network (e.g, MLP) with the updated information

GNN Layer - How it works



GNN Layer - Pros/Cons

- Pro:
  - Most problems can be expressed as graphs
  - More control over the learning process when a fixed structure exists
  - Good candidate for physics informed methods
- Con:
  - Can be inefficient (e.g., convolutions as graphs)
  - Graph isomorphism test problem (graph structure)
  - Higher vulnerability to noise

Self Attention Layer

- Relate different positions of input sequence to a representation
- Sparse patterns and long range complex correlations
- Common is stacking of Self Attention layers with multi-head attention
- Identify where to put attention
- Self Attention components:
  - Key: Input representation for search
  - Query: How to search across all keys
  - Value: How to map input with scoring

} result: scores

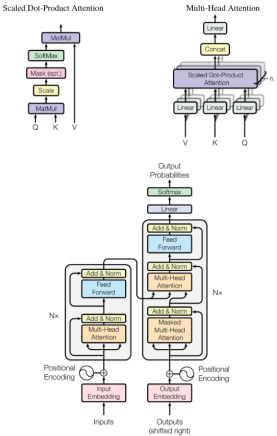
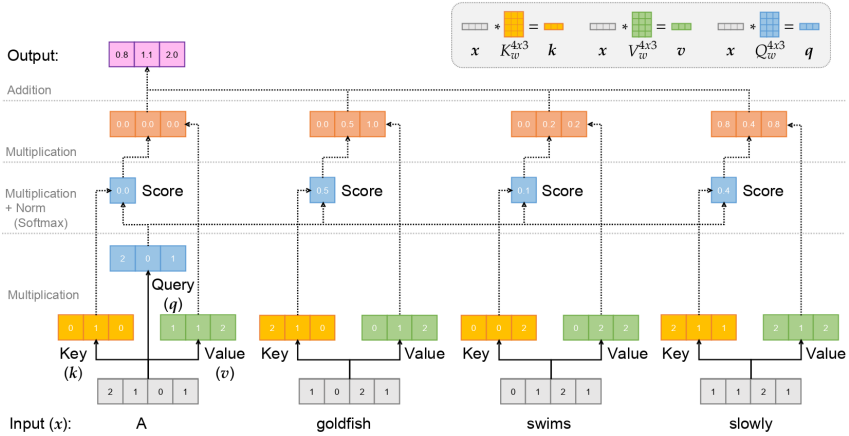


Figure 1: The Transformer - model architecture.

Vaswani et al., Attention Is All You Need

Self Attention Layer - How it works



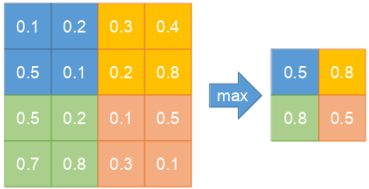
Self Attention Layer - Pros/Cons

- Pro:
  - Works well if structure/patterns are complex (non-sparse, unknown recurrence)
  - Only consists of linear operations and scales well
  - Less problematic to long range relations
- Con:
  - Large number of parameters
  - From *Theoretical Limitations of Self-Attention in Neural Sequence Models* (Hahn):

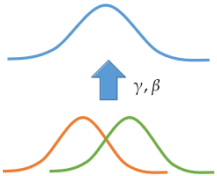
> *. . . it [self-attention] cannot model periodic finite-state languages, nor hierarchical structure, unless the number of layers or heads increases with input length. These limitations seem surprising given the practical success of self-attention and the prominent role assigned to hierarchical structure in linguistics, suggesting that natural language can be approximated well with models that are too weak for the formal languages typically assumed in theoretical linguistics.*

Collateral Layers

- Pooling:
  - Downsample/-scale
  - Reduce activations
  - Select a sampling operation (max, min, avg)

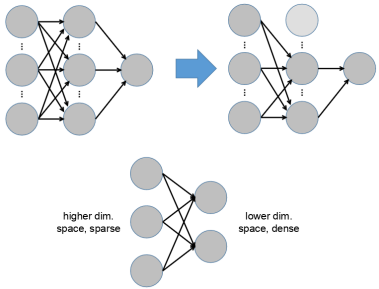


- Batch Normalization:
  - Scale and shift activations
  - Results in more reliable range/distribution of activation for following layers
  - Faster training, better generalization (due to implicit regularization)



Collateral Layers - cont'd

- Dropout:
  - Randomly remove activations during training (specified as dropout rate)
  - Adds regularization
  - Minimizes specializations of neurons
- Embedding:
  - Map higher dimensional space to lower one
  - Map inputs to numerical/vector representation
  - Input can be sparse, output is dense
  - Retain only needed context or spatial information for a specific task



Summary

- The same problems can be solved with different building blocks
- There is no *one-size-fits-all* layer/architecture
- Trade-off pros and cons for a given task and objective
- CNNs, RNNs and GNNs are quite common nowadays
  $\Rightarrow$ We'll shed more light on Self Attention (and Transformers) in the next days...

EURO

Thank you!

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia.