Seminar in Deep Neural Networks

Topics and Resources

What should a good presentation include?

1. Motivate the topic - why are people even doing research in this area?
2. Make sure everybody gets the basics - there is no point in explaining the details of an algorithm to someone who did not get the idea
3. Draw novel, thought provoking connections - Does the topic you are presenting relate to something that the author did not think about? In which way? What do you see as outcome of this connection?
4. Teach something new - the lecture is a success if everybody learned something interesting. Given the huge amount of papers in Deep Learning no one in the room will be aware of all papers in your topic - find something interesting and present it. Make sure however to not miss point 2.

23.2. - Introduction

In the first seminar we will have a broad overview over the topics we will discuss during the seminar:
We will discuss the background of Natural Language Processing (NLP) and have a short introduction into what deep reinforcement learning (DRL) is.
We will talk about the message passing framework used in graph neural networks (GNNs) as well as their application area. We further will highlight the topics on algorithm learning which we will discuss in the seminar.

2.3. - Deep Learning and Neural Architecture

This lecture should give an introduction into function approximation using deep learning. It should cover basic motivations for using deep learning such as the universal function approximation theorem and newer insights such as the deep double descent phenomena. The lecture should also give an overview over basic neural network architectures, including

- Convolutional Neural Networks
- Recurrent Neural Networks (LSTMs)

The lecture might also provide an overview of deep learning frameworks, there Pro and Cons (Tensorflow vs. PyTorch vs. JAX).

The second part of the lecture should focus a bit on the limitations of deep learning. Specifically, the no-free-lunch theorem, the extrapolation problem as well as more practical considerations like exploding and vanishing gradients and the memory requirements of backpropagation. The latter topic can be discussed alongside proposed solutions:

- Residual Networks
- Implicit Layers

9.3. - NLP: Benchmarks/Tasks/Metrics

This lecture should introduce the area Natural Language Processing (NLP), benchmarks (see GLUE and SuperGLUE) with the tasks therein, common metrics (BLEU and ROUGE scores) as well as some examples of what can be done with todays state-of-the-art models as well as examples of things that cannot be done yet.

9.3. - NLP: Embeddings

This lecture should cover different strategies to encode text. First, word embeddings like word2vec and Glove, then embeddings of groups of words (skip-gram) and finally sentence embeddings, e.g. Universal Sentence Encoder.

- Word2vec: https://arxiv.org/abs/1301.3781
- GloVE: https://www.aclweb.org/anthology/D14-1162/
- Skip-gram: https://papers.nips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html
- USE: https://arxiv.org/abs/1803.11175
- etc.

16.3. - NLP: Attention/BERT/GPT

This lecture is centered around the Transformer architecture. It should start by introducing the first paper using attention for NLP, and then the original Transformer paper (Attention is all you need).

- Original attention paper: https://arxiv.org/abs/1409.0473
- Attention is all you need (Transformer): https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

This talk should continue with the bi-directional encoding scheme and pretraining objectives in BERT and derived models as well as decoder-only transformer (GPT-2), the language modeling objective and the derived model GPT-3

- BERT: https://arxiv.org/abs/1810.04805
- BART: https://arxiv.org/abs/1910.13461
- T5: https://arxiv.org/abs/1910.10683
- GPT-2: http://www.persagen.com/files/misc/radford2019language.pdf
- GPT-3: https://arxiv.org/pdf/2005.14165.pdf

23.3. - RL: Stochastic planning in games

This lecture should cover two areas: Planning given a perfect model, foremost Monte Carlo Tree Search, and Self-Play, where agents learn by playing against themselves - often approximated by fictitious play. Papers for these subjects are:

- A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play
- Deep Reinforcement Learning from Self-Play in Imperfect-Information Games
- A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning
- Emergent Tool Use from Multi-Agent Interaction

An interesting case study might also be the AlphaStar algorithm.

30.3. - RL: Model based vs. model free DRL

While most original DRL algorithms (DQN, A3C, PPO) are model free, this lecture should introduce the idea of explicitly learning a model of the environment for planning (either Model Predictive Control or Monte Carlo Tree Search[1], see e.g., Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model) or learning in a “dream” (World Models, Model Based Reinforcement Learning for Atari). The lecture should also introduce successor features, a middle ground between model based and model free DRL. A paper that might also be interesting is Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees.

[1] MCTS should be introduced in the lecture on games the week before

13.4. - RL: Hierarchical DRL

This lecture should cover the idea of abstraction over time. That is, as humans we have skills/options - a prolonged set of atomic actions - that we use over and over again. Can a learning algorithm find/use such a temporal hierarchy? Papers in this direction include:

- The option-critic architecture
- Data-Efficient Hierarchical Reinforcement Learning
- FeUdal Networks for Hierarchical Reinforcement Learning
- Hierarchical Reinforcement Learning with Advantage-Based Auxiliary Rewards
- DAC: The Double Actor-Critic Architecture for Learning Options

20.4. - RL: Meta-Learning

The lecture on Meta-Learning should cover the learning to learn paradigm - learn on many tasks in order to learn faster on a new task. Papers include:

- RL2: Fast Reinforcement Learning via Slow Reinforcement Learning
- Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks
- Learning to learn by gradient descent by gradient descent
- Meta-Gradient Reinforcement Learning with an Objective Discovered Online

A good reference is also the ICML tutorial on meta learning.

27.4. - GNN: Architectures

This lecture introduces the general message passing framework of GNNs and several variants. You should give a good overview by clustering the main approaches. You should also relate this to the tasks at hand.

- Useful overview of GNNs in this paper: https://arxiv.org/pdf/1810.00826.pdf (bottom of page 2)
- Comprehensive Survey (don't introduce all of this!): https://arxiv.org/pdf/1901.00596.pdf
- Lecture Notes: https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_mar29.pdf
- Methods and Applications: https://arxiv.org/pdf/1812.08434.pdf
- Pointer network & Deepsets are a nice starting point
- GCN (vs CNN), GIN, Gated GNNs etc.

27.4. - GNN: Algorithmic Alignment / Necessity

This lecture looks at when to use GNNs and when not to use then.

- Neural Execution of GNNs: https://openreview.net/forum?id=SkgKO0EtvS
- Algorithmic Alignment: https://arxiv.org/abs/1905.13211
- Extrapolation: https://arxiv.org/pdf/2009.11848.pdf
- Pointer Graph Networks: https://arxiv.org/pdf/2006.06380.pdf
- GNN and termination: https://arxiv.org/abs/2010.13547

4.5. - GNN: Theoretical Limitations

This lecture focuses on the theoretical limitations of GNNs.

- Relation to WL tests (GIN): https://arxiv.org/pdf/1810.00826.pdf
- Relation to distributed computing models (Loukas): https://openreview.net/forum?id=B1l2bp4YwS
- Distinguishing graph (Loukas): https://arxiv.org/pdf/2005.06649.pdf
- Approximation ratios of GNN: https://arxiv.org/pdf/1905.10261.pdf

4.5. - GNN: Oversmoothing

This lecture should talk about one of the major problems with current GNN Architectures - oversmoothing - and how it might be countered. Many of these approaches are based on sparsifying the number of messages in each round.

- https://openreview.net/forum?id=S1ldO2EFPr
- measuring and relieving the over-smoothing problem
- DropEdge
- Simple and Deep Graph Convolutional Network
- Robust Graph Representation Learning via Neural Sparsification
- Bayesian Graph Neural Networks with Adaptive Connection Sampling

11.5. - GNN: Graph Generation

This lecture should cover how graphs can be generated using Neural Networks. Traditional graph generators are quite limited, as they can only capture some graph metrics and do not model graphs coming from realistic distributions well. NNs are great at capturing intricate patterns in the data, so they can be much more successful in generating graphs from complex distributions. One very promising application of such methods is molecule generation.
Some relevant references:

- GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders
- GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models
- Efficient Graph Generation with Graph Recurrent Attention Networks
- MolGAN: An implicit generative model for small molecular graphs

11.5. - GNN: Simulation using GNNs

This lecture should cover how Neural Networks, in particular Graph Neural Networks, can be used to learn and predict physical processes. Graph Neural Networks are particularly good at this, because they align well with the structure of physical interactions. What are the limitations of such neural simulators and what are the different problems they can be applied on?
Some relevant references:

- Learning Mesh-Based Simulation with Graph Networks
- Discovering Symbolic Models from Deep Learning with Inductive Biases
- Learning to Simulate Complex Physics with Graph Networks
- Scalable Graph Networks for Particle Simulations
- Hamiltonian Neural Networks

18.5. - ALG: Combining Algorithms and NNs

This lecture should talk about how NNs can be implanted into algorithms to improve accuracy or efficiency, as well as how algorithms can be implanted into NNs to help with the inherent deficiencies of NNs.

- Black Box Combinatorial Solver: https://arxiv.org/abs/1912.02175
- Can we do better than binary search? https://arxiv.org/pdf/1712.01208.pdf
- NN branching for MILP: https://arxiv.org/abs/1906.01629

18.5. - ALG: Math

This lecture should talk about how we can design NNs to do simple arithmetic. It should address why it is so difficult for NNs to do this, in particular the challenging issue of extrapolation, and how proposed architectures deal with the challenges.

- Neural Arithmetic Logic Unit
- Neural Arithmetic Unit
- Neural Power Unit
- Neural GPUs
- Neural Status Register
- Logical Neural Networks

25.5. - ALG: NN architectures for Algorithm Learning (Memory)

This lecture is about implementing equivalent forms of computer memory in Neural Networks. LSTMs offer a kind of “implicit” memory, but here we look at more explicit and exact memory that can be used for computations.

- Memory Networks
- Longformer is an example if the use of memory in NLP
- Learning to transduce with unbounded memory (the NN equivalent of Data structures)
- End-to-end memory networks (the NN equivalent of RAM)
- Neural turing machine
- Neural Stored-program Memory (~Self-modifying RAM)

25.5. - ALG: Non-differentiable optimization

A neat feature of RL is that the algorithms are designed to optimize a score (maximize the reward), which leaves a lot of room for the engineer to set the reward. More specifically, if one wishes to minimize a non-differentiable function f(x), one can train a DRL algorithm to alter x with a reward of -f(x). As such, RL opens the possibility to use deep learning in many new applications, including:

- Generation of Discrete Token Sequences - see SeqGAN
- Neural Architecture Search
- Graph Problems - see Attention, Learn to Solve Routing Problems!

The lecture should also contrast RL to the other big family for non-differentiable optimization - evolutionary algorithms.

1.6. - Review Seminar

In the last seminar, we will summarize the different insights we have seen, connecting them back to the overview of the research field from the first seminar. We will discuss unsolved problems and potential future work.
Also, we will discuss your results on the coding challenge.