Online Course: Machine Learning for Physicists 2020

Lecture Series by Florian Marquardt

This website contains the course material for the lectures and tutorials delivered in the summer of 2020.

Style of the lectures: 'Inverted classroom' style, one zoom online session per week, each Tuesday 6pm (Central European Time), running from April 21 to August 4, 2020. See below for videos, code, slides, notes, etc.!

> **[August 4] Course finished!**
> Today we finished the lecture series with student mini-project presentations, with a large variety of different projects from different areas.
>
> I will still keep checking the discussion group from time to time. It was fun teaching you all. For now, go and apply what you learned here to create something useful and exciting!

This is the website for the online lecture series by Florian Marquardt (April-August 2020). Here we collect the course overview and links to the forum, code, etc.

Most important info

The course is **inverted-classroom style**. This means you watch one of the pre-recorded video lectures (about 90min) on your own, and then we use a live zoom meeting to: discuss the lecture, do live tutorials, and discuss homework problems! The live meetings take place every week on Tuesday, at 6pm German time (CEST = Central European Summer Time).

The first meeting, with a quick overview of the course, took place Tuesday April 21st at 6pm. For the second meeting (April 28), you will then already watch the first recorded lecture. Meetings take place up until (and including) August 4. The full course schedule can be found below.

Map: Registered participants

Here is a Google Maps visualization of all the registered participants' cities:

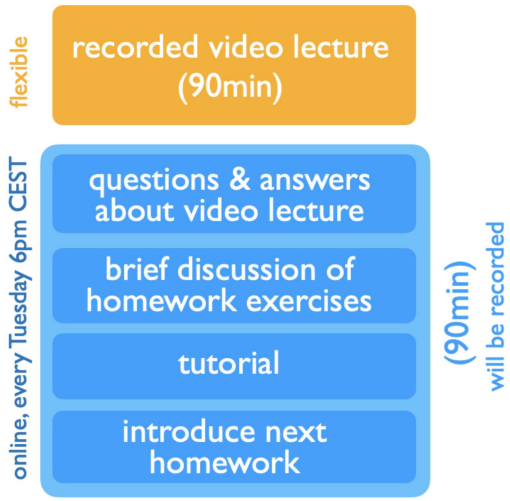
Map: Machine Learning 2020 Lecture Series (https://drive.google.com/open?id=1V-_5KUYL4h0zsAxgWBoO9qds2DIhROcS&usp=sharing)

Some of the questions we will address in this lecture series

- How do artificial neural networks look like?
- How do you train them efficiently?
- How do you recognize images?
- How do you learn a compact representation of data, without providing any extra information?
- How do you analyze time series and sentences?
- How do you discover strategies from scratch without a teacher?
- What are some modern applications in science/physics?

Structure of lessons

Hint: for a description of each of the lectures, please go to ‘course overview’. There you will also find the material to download, such as the jupyter notebooks we will use in the tutorials.



Lecture video

Before each online session, please watch the pre-recorded video.

Questions & Answers

In the beginning of each online session, I will answer questions about the material. Please 'raise your hand' in zoom and when it is your turn, I will unmute you and tell you to speak. (during the Q&A session, I will usually have muted everyone else, to prevent chaos)

Brief discussion of homework

This discusses the homework exercises that I suggested for you to do until this session.

Tutorials: Interactive group practice sessions using 'Breakout Rooms'

These practice sessions are using zoom's **breakout rooms**. After introducing some jupyter notebook and explaining the task, I will randomly distribute all of you into some dozens of these breakout rooms. Once I 'open' the rooms, you should get a message from zoom inviting you to join your breakout room. Please accept this, and you will appear inside this 'room' together with about ten (or so) fellow students. You can now **unmute yourself** (and also **switch on your video** if you like).

Please: everyone introduce themselves to the others at the start of the discussion! (say who you are and where you are from, maybe also your level, i.e. undergraduate/graduate student etc.)

Please start discussing the task! Someone has hopefully already downloaded the jupyter notebook (I put them up for download on this website the day before; you might need to refresh the website). This someone please open the notebook in jupyter and **share their screen**, so that others can see what is going on. Then discuss and experiment and explore together!

After some while (maybe 20min or so), I will call you back. I can broadcast messages to all breakout rooms. Inside the room, there is also some zoom option to 'call for help', which alerts me that I should go to that room and see whether there is someting I can help. (I have not yet found out whether there is some option for direct chat from within the rooms to me, i.e. to the host)

Brief introduction of next homework

At the end of each online session, I will briefly present to you the suggested homework exercises. These will also be listed on the website, under the heading for this particular lecture/online session, in the 'course overview' below.

Discussion forum

Use the forum to ask & answer questions about the course material and homework, as well as to organize yourself into small teams to work together:

https://groups.google.com/d/forum/machine-learning-for-physicists
(https://groups.google.com/d/forum/machine-learning-for-physicists)

Github code repository

All the python codes for tutorials, homework, and examples in the lectures (as jupyter notebooks and also as pure python code):

https://github.com/FlorianMarquardt/machine-learning-for-physicists
(https://github.com/FlorianMarquardt/machine-learning-for-physicists)

Individual links for the code, with descriptions, can also be found below, in the lecture content summaries (at the end of this web page).

Some extra info

This course has been delivered twice before, in the summers of 2017 and 2019, at the physics department (https://www.physics.nat.fau.eu) of the university in Erlangen (https://www.fau.eu) (Bavaria/Germany). Both versions have been recorded on video, and it is the 2019 recordings that we will use here (see below for the links). Please disregard any organizational announcements made on the recorded video, as they of course relate to the 2019 course. We will make any up-to-date announcements in the live sessions.

The original website for the course is https://machine-learning-for-physicists.org (https://machine-learning-for-physicists.org), but the detailed up-to-date materials for the 2020 online version will be found here!

I (Florian Marquardt) am a theoretical physicist working at the intersection of nanophysics and quantum optics, with a strong recent focus on applications of machine learning. My group is located at the Max Planck Institute for the Science of Light (https://mpl.mpg.de/divisions/marquardt-division/), and I am also affiliated with the university in Erlangen. On my low-volume twitter account FMarquardtGroup (https://twitter.com/FMarquardtGroup), you can find announcements of our most recent research papers as well as job openings (the latter also on the group website (https://mpl.mpg.de/divisions/marquardt-division/)).

If you want to reach me with questions regarding the course, it is most efficient to use the forum. My email account is usually overflowing with administrative stuff, so I would likely miss your email in my inbox.

Finally, I also would like to thank Thomas Fösel, Leopoldo Sarra, Riccardo Porotti, and Victor Lopez-Pastor for their help in the earlier lectures.

Lecture Notes

Although there are no extended lecture notes that cover all the material of this series, I have produced a condensed version (40 pages) of lecture notes, on the occasion of the Les Houches Summer School 2019. You can download an almost-finished draft here (including discussions of applications to quantum physics, and a bit about quantum machine learning):

Les Houches 2019 Machine Learning Lecture Notes
(https://owncloud.gwdg.de/index.php/s/KekM07rYhv4EZi8)

Once I have finished some yet incomplete parts in the quantum machine learning section, I will upload the draft to the arXiv; later these notes will become part of the official book with the lecture notes for this school. If you find these lecture notes useful for your research, please cite them, e.g. like this:

F. Marquardt, "Machine Learning and Quantum Devices", to appear in "Quantum Information Machines; Lecture Notes of the Les Houches Summer School 2019", eds. M. Devoret, B. Huard, and I. Pop (Oxford University Press)

Slides

Here are the slides used in the lectures, split up in three parts (first one goes up to and including t-SNE; third one covers some applications to science).

Machine Learning for Physicists, Slides, Part One (pdf, 25 MB)
(https://owncloud.gwdg.de/index.php/s/qetLJgXMW6u3FwC)
Machine Learning for Physicists, Slides, Part Two (pdf, 13 MB)
(https://owncloud.gwdg.de/index.php/s/Sc89k0zzFlaw0g9)
Machine Learning for Physicists, Slides, Part Three (pdf, 4 MB)
(https://owncloud.gwdg.de/index.php/s/iLpJmXvDI2I7zpz)

Software needed

First of all, you want to install the python (https://www.python.org) programming language. Also, I recommend JupyterLab (https://jupyter.org) as a convenient notebook interface for python programming. Depending on your taste and your system, you might want to download these individually or as part of a full distribution like Anaconda (https://www.anaconda.com).

An alternative, completely online solution is the Colaboratory (https://colab.research.google.com/notebooks/intro.ipynb) platform by Google. This is a web-based jupyter notebook interface to python that comes with tensorflow & keras pre-installed and allows you to run your code for free on their GPU servers.

Starting in the 3rd lecture, we will use keras (https://keras.io), the convenient high-level python neural-network software package. This is included automatically in every recent tensorflow (https://www.tensorflow.org) installation, so I recommend installing tensorflow (after having python) and then getting access to keras commands in the form

```
import tensorflow as tf
from tensorflow.keras.layers import Dense
```

Note: In the examples shown in the course, I assumed you have a keras installation (and, below that, as support for keras, there would be tensorflow or other similar packages), so the syntax for these imports looked slightly different. Adapt as needed.

If you have trouble installing the software (all of which is free!), you may use the forum (link above) to ask other students in the course who might help you.

If you have suggestions for useful software or platforms etc., please post them on the forum! This is a community effort (since I will not be able to provide individual help with installations).

Course overview

Session 0: Introduction

📅 Online session introducing the course: **April 21, 6pm** (all times CEST, i.e. time in Germany)

We will discuss the lecture series program, prerequisites, and the format of the online sessions. I will also say some words about the pre-history of neural networks. Please pre-register! (Link at the top of this website, under "most important info")

Slides for online session (PDF, 14 MB) (https://owncloud.gwdg.de/index.php/s/icQRtIwFtwPvXfh)

Thanks to Mahdi Aslani, we now also have a recording available:
Video recording for online session 0 (mp4 video) (https://owncloud.gwdg.de/index.php/s/xoC1OUE5IFy7g8U)

Suggested historical reading: Alan Turing's "Computing Machinery and Intelligence" (1950) (https://academic.oup.com/mind/article/LIX/236/433/986238), introducing what is now known as the "Turing test" for artificial intelligence. A wonderful (and very nontechnical) paper!

Further historical reading: Rosenblatt's concept of a "Perceptron" (1957) (http://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf)

Note on videos: Alternative to the links below, you can also check out the **Videos on iTunes University (https://podcasts.apple.com/us/podcast/id1490099216)** (no account required)

Lecture 1: Basic Structure of Neural Networks

📹 Recorded video: Lecture 1 (https://www.video.uni-erlangen.de/clip/id/10611) (slides: see above in section “Slides”)
📅 Online Q&A session and tutorials about this material: **April 28, 6pm** (please watch the lecture video before). See the online session recording (https://owncloud.gwdg.de/index.php/s/xEKan8JgaMtb1ba) (100 MB mp4) and the online session slides (https://owncloud.gwdg.de/index.php/s/sGX5FWcOIzdyFos)

Contents: Introduction (the power of deep neural networks in applications), brief discussion of the lecture outline, structure of a neural network and information processing steps, very brief introduction to python and jupyter, implementing a deep neural network efficiently in basic python (without any additional packages like tensorflow), illustration: complicated functions generated by deep neural networks

After this lecture, you will know the basic structure of a neural network and how to implement the ‘forward-pass’ in python, but you don’t yet know how to adapt the network weights (training).

Code (jupyter notebook): 01_MachineLearning_Basics_NeuralNetworksPython.ipynb (https://owncloud.gwdg.de/index.php/s/UnI2Yru1HsqwQNK)
(or download code as pure python script (https://owncloud.gwdg.de/index.php/s/WLfHIv2YXhq0Z29))

This notebook shows how to calculate the forward-pass through a neural network in pure python, and how to illustrate the results for randomly initialized deep neural networks (as shown in the lecture).

Notebooks for tutorials:
Tutorial: Network Visualization notebook (https://owncloud.gwdg.de/index.php/s/lugGF9eCxCIOk56) (also as pure python (https://owncloud.gwdg.de/index.php/s/SgogI8iVzi60owR))
Tutorial: Curve Fitting notebook (https://owncloud.gwdg.de/index.php/s/63ok2nUCuTvYwb7) (also as pure python (https://owncloud.gwdg.de/index.php/s/SgogI8iVzi60owR))

The “Network Visualization” notebook shows how to visualize arbitrary multilayer networks (with two inputs and 1 output neuron, so the result can be displayed in the plane), including a visualization of the network structure.

The “Curve Fitting” notebook visualizes nonlinear curve fitting for a 1D function with a few parameters, via stochastic gradient descent. This is useful to understand what is going on in the higher-dimensional case of neural networks, where essentially the same concept is applied. We did not get around to trying this out during the online tutorials, but please try it and ask questions or present your results on the forum!

Homework for Lecture 1
(The solutions will be briefly discussed in the online session for lecture 2. In order to follow the lecture series, please do at least two of these problems – I suggest the ones with the *. And of course, if you do more, you will get more practice and quickly become a master!)

(1)* Implement a network that computes XOR (arbitrary number of hidden layers); meaning: the output should be +1 for y1 y2<0 and 0 otherwise!
(2) Implement a network that approximately or exactly computes XOR, with just 1 hidden layer(!)
(3)* Visualize the results of intermediate layers in a multi-layer randomly initialized NN (meaning: take a fixed randomly initialized multi-layer network, and then throw away the layers above layer n; and directly connect layer n to the output layer; see how results change when you vary n; you can start from the notebook 01_MachineLearning_Basics_NeuralNetworksPython.ipynb (https://owncloud.gwdg.de/index.php/s/UnI2Yru1HsqwQNK))
(4) What happens when you change the spread of the random weights? Smart weight initialization is an important point for NN training.
(5) Explore cases of curve fitting where there are several (non-equivalent) local minima. Is sampling noise helpful (i.e. the noise that comes about because of the small number of x samples)?

Solutions for tutorials and homework:
Visualization Notebook with solutions for tutorials and for some of the homework (https://owncloud.gwdg.de/index.php/s/vuPytW60yS4IhCw)

Lecture 2: Training a Neural Network

Recorded video: Lecture 2 (https://www.video.uni-erlangen.de/clip/id/11034)
Online session: **May 5, 6pm** (video recording (https://owncloud.gwdg.de/index.php/s/XOkgYsCbRVs5dDt), 166 MB)

Contents: Batch processing of many input samples, efficient implementation in python, neural networks can generate arbitrary functions, training a network as (high-dimensional) nonlinear curve fitting, cost function, stochastic gradient descent, backpropagation algorithm, full implementation in python, relation to physics (path integral), summary: basic ingredients of neural networks (and hyperparameters)

After this lecture, you will in principle be able to train a deep neural network on arbitrary tasks (using pure python code that we provide in the lecture). But you don't yet know how to choose a smart representation of the data in more complicated cases, how best to choose the parameters during training, how to accelerate the gradient descent, etc.

Code shown in lecture:
Notebook: backpropagation with python (https://owncloud.gwdg.de/index.php/s/1xIsExzz03f0e5b) (also as pure python script (https://owncloud.gwdg.de/index.php/s/wLg15kOdNFrc4WF))

This notebook demonstrated how to implement backpropagation in python, in a small amount of code. For a more cleaned-up version of this implementation and much better visualization, see the tutorial notebook below!

Notebook for online tutorials:
Tutorial: Network Training Visualization notebook (https://owncloud.gwdg.de/index.php/s/ME4Gxe0XLskNPK1) (as pure python script (https://owncloud.gwdg.de/index.php/s/QFiiS41PIJFpw8V))

This notebook visualizes training of multilayer neural networks!

Homework for Lecture 2

(1)* Carefully study the backpropagation algorithm, on paper and in the program
(2)* Visualize the training of a multi-layer network for some interesting function! Explore reLU vs sigmoid!
(3) Analyze the evolution of the slope w during stochastic gradient descent on a cost function given by C=(1/2) sum_j (w x_j - wtilde x_j)^2, where x_j are the N samples drawn from a Gaussian distribution in a single training step. (this is an advanced exercise)

Lecture 3: Training and analyzing networks, Keras package, Image recognition

Recorded video: Lecture 3 (https://www.video.uni-erlangen.de/clip/id/11243)
Online session: **May 12, 6pm** (Recorded Video, 134MB (https://owncloud.gwdg.de/index.php/s/pSj2LyYC2fvfgyt))
Slides for online session (7 MB; including previous ones) (https://owncloud.gwdg.de/index.php/s/7uc5XCRYWcDfwD7)

Contents: extended review of previous material, training a 2D function (reproducing an arbitrary image), first steps towards analyzing and interpreting a neural network, influence of batch size and learning rate, brief introduction to the keras package and its basic neural network routines, image recognition (one-hot-encoding, softmax, cross-entropy), the danger of overfitting, training vs. validation vs. test data

After this lecture, you will have more experience in training, understand the dangers, and know what the most elementary hyperparameters such as batch size or learning rate will do. You will also have a first taste of keras (https://keras.io), the convenient high-level neural-network software package (included in every tensorflow (https://www.tensorflow.org) installation). You will know the basics of how image recognition works, which is one of the most important applications of neural networks. But you don't yet know about the more advanced network structures one can build, such as the convolutional neural networks important for image recognition.

Notebooks shown in lectures:
Notebook: Training on an Image and Interpreting Neurons (https://owncloud.gwdg.de/index.php/s/S6VfIxnaEuBGorv) (also as python script (https://owncloud.gwdg.de/index.php/s/S6VfIxnaEuBGorv)); you also need to download the smiley image (https://owncloud.gwdg.de/index.php/s/MNSDa0GcIF9i85h)!

This is the training on the smiley image and inspecting the network by switching off neurons in the last layer (slightly modified from the lecture). Of course, you can replace the smiley by an image of your choice!

Notebook: Handwritten Digits (MNIST) Recognition using keras (https://owncloud.gwdg.de/index.php/s/UfQRfNPtdmAYUuA)

This will require you to download the freely available MNIST images (Update: it seems these are included with keras installations nowadays). You can also adapt this notebook to work on other images, of course, including algorithmically generated ones.

Notebook for online tutorials:
Tutorial Notebook: Minimal Keras example (https://owncloud.gwdg.de/index.php/s/I6b9EILLngufWCF) (also as python script (https://owncloud.gwdg.de/index.php/s/GUhLTBaMiVArHWe))

This is really a minimal example, which shows how to build a neural network in a few lines using keras, and how to train and evaluate it (with one line each).

Further notebook:
Tutorial Notebook: Network Training Visualization using Keras (https://owncloud.gwdg.de/index.php/s/ehGzaEavwb7DDvr) (also as python script (https://owncloud.gwdg.de/index.php/s/I2DLevxQc8PCCkn))

This is essentially a keras-version of our visualization notebook from lecture 2!

Homework for Lecture 3
(will be discussed in online session for lecture 4; hint: use the forum to discuss and post your ideas and solutions!)

(1)* Train on some function or image and explore the role of various neurons (by switching weights on/off like in the lecture); use the keras training visualization notebook above!

(2)* Produce a simple network with keras that can distinguish a 1D plot of a random Gaussian from a 1D plot of a random Lorentzian! (or other classes of functions)

(3) Produce a simple network with keras that can distinguish a 2D picture of a randomly placed square from that of a randomly placed circle! What happens if you add noise to the images?

Lecture 4: Convolutional Neural Networks, Autoencoders, Principal Component Analysis

Recorded video: Lecture 4 (https://www.video.uni-erlangen.de/clip/id/11266)
Online session: **May 19, 6pm** (recorded video (https://owncloud.gwdg.de/index.php/s/Be5oFKLFoUCf4e6), 94 MB mp4) (Slides for online sessions up to and including session 4 (https://owncloud.gwdg.de/index.php/s/WFoWQJMAcO4c2Wn))

Contents: A bit more on image recognition, convolutional neural networks as an efficient way to process images and other data with translational invariance, (kernels, channels, and so on), autoencoders for unsupervised learning and information compression, principal component analysis as a simple linear way to extract the main (linear) features of data sets

After this lecture, you will know the workhorse behind image recognition networks, i.e. convolutional layers, you will understand the beautiful concept of autoencoders which automatically extract the essential features of data (and can turn those back into the data), and you will know about the widely used tool of principal component analysis.

But there is more to come!

Notebook with solutions for lecture 3 homework
(discussed in online session 4)

Notebook: Image Recognition (1D / 2D) (https://owncloud.gwdg.de/index.php/s/x8XRVXiXHsJpqDg) (also as pure python script (https://owncloud.gwdg.de/index.php/s/2xY1ywo8KygF24j))

Notebook for online tutorials:
Tutorial Notebook: Visualizing the training of convolutional autoencoders (https://owncloud.gwdg.de/index.php/s/WaTnoRBSF9UxWoN) (also as pure python script (https://owncloud.gwdg.de/index.php/s/GFe4zAy02vSiltM))

See how an autoencoder operates! And take the **Grand Autoencoder Challenge** in the homework problem afterwards!

The deadline for the **Grand Autoencoder Challenge** challenge is Tuesday May 26 at noon (CEST, i.e. time in Germany). Up until then, feel free to continuosly post your updated current cost values in the forum (https://groups.google.com/d/forum/machine-learning-for-physicists). After this deadline, you can post your networks in the forum, and I have a chance to present and discuss the winner networks in the online session!

Homework for Lecture 4
(will be discussed in online session for lecture 5; hint: use the forum to discuss and post your ideas and solutions!)

*(1) Compete in the ***Grand Autoencoder Challenge!*** This is described in the notebook above: try to optimize an autoencoder that can efficiently encode randomly placed circles (post your intermediate cost values in the forum (https://groups.google.com/d/forum/machine-learning-for-physicists); specify which version of the challenge this refers to: MEDIUM/HARD, NORMAL/LONG-TRAINING)

(2) Train a network to turn images that contain multiple random circles into squares of the same size and placement!

Lecture 5: Principal Component Analysis, t-SNE and unsupervised dimensionality reduction, Advanced Gradient Techniques, Introduction to Recurrent Neural Networks

Recorded video: Lecture 5 (https://www.video.uni-erlangen.de/clip/id/11487)
Online session: **May 26, 6pm** Recorded video, 123 MB
(https://owncloud.gwdg.de/index.php/s/ihFKDp3wqIURxZ0)

Contents: More about the principal component analysis, unsupervised dimensionality reduction techniques for clustering and other applications (t-SNE etc.), advanced gradient descent techniques (like 'adam' and its siblings), first introduction to recurrent neural networks

After this lecture, you will know how to take a high-dimensional data cloud and reduce it down to low dimensions without losing the important clusters of data points, you will know what's behind 'adam' and similar gradient descent techniques that are adaptive and can accelerate training, and you will know what's the motivation for recurrent networks which can be used to analyze time series and sentences.

However, you do not yet know exactly how efficient recurrent networks are implemented, which will be the subject of lecture 6!

Notebook for online tutorials:
Tutorial Notebook: t-SNE for dimensionality reduction
(https://owncloud.gwdg.de/index.php/s/6EwPmdqcHUWOkAt) (also as pure python script (https://owncloud.gwdg.de/index.php/s/dtsLRWKvkaCIjPP))

See how t-SNE works for producing 2D representations of high-dimensional data!

Notebook with solutions for lecture 4 homework
(discussed in online session 5)

Notebook: Image transformation - 'Transforming Circles To Squares' - using a convolutional network (https://owncloud.gwdg.de/index.php/s/OsZ1ZOXYGItCNLU) (also as pure python script (https://owncloud.gwdg.de/index.php/s/Hd6j6b1SX2P3C6I))

Homework for Lecture 5
(will be discussed in online session for lecture 6; hint: use the forum to discuss and post your ideas and solutions!)

*(1) Train an autoencoder and then visualize the neuron activations in the bottleneck layer via t-SNE! (i.e. for each of many input images extract the bottleneck activations, which defines a vector in a high-dimensional space. Plot all these vectors in a 2D projection produced by t-SNE) Can you observe any meaningful clusters?

Lecture 6: Recurrent Neural Networks (LSTM), Word Vectors, Introduction to Reinforcement Learning

Recorded video: Lecture 6 (https://www.video.uni-erlangen.de/clip/id/11537)
Online session: **June 2nd, 6pm** Recorded video 175 MB mp4
(https://owncloud.gwdg.de/index.php/s/mvjjZ28EOEUCECs)

Contents: The inner workings of a long short-term memory (LSTM) recurrent network, simple examples for recurrent network applications, word vector encodings, first introduction to reinforcement learning

After this lecture, you will know how to set up a recurrent network to process sequences, you will have learned about “word vectors” that encode conceptual similarities between words, and you will know what is the main goal of reinforcement learning.

But you do not yet know how reinforcement learning really works in practice!

Notebook with solutions for lecture 5 homework
(discussed in online session 6)

Notebook: Applying tSNE to the bottleneck neurons of an autoencoder (https://owncloud.gwdg.de/index.php/s/FOYEqZ9eRynfH0c) (also as pure python script (https://owncloud.gwdg.de/index.php/s/ieNTFFLLiaYYSde))

Notebook: Simple LSTM examples (https://owncloud.gwdg.de/index.php/s/4rwfuMpKZow6V6m) (also as pure python script (https://owncloud.gwdg.de/index.php/s/iuPohyfCrf8cgFG))

From the lecture: the recall network and the countdown network!

Notebook for online tutorials:
Tutorial Notebook: Recurrent neural network for predicting physical time evolution (https://owncloud.gwdg.de/index.php/s/ewRZOw8B1sRYaWJ) (also as pure python script (https://owncloud.gwdg.de/index.php/s/wZNzWTPeFzzasG5))

See how an LSTM network can help to predict the time evolution of a particle that is observed only indirectly through its effects on another particle. This task requires memory!

Homework for Lecture 6
(will be discussed in online session for lecture 7; hint: use the forum to discuss and post your ideas and solutions!)

*(1) Apply a recurrent network to predict the time-evolution of a physical system during t=T...2T after observing it during t=0...T (for random initial conditions); try to treat the case of 'partial' observation (i.e. observing only one of the particles).

Hint: Give as input a trajectory that is nonzero for 0...T (according to the evolution), and zero during T...2T. Give as the correct output the full trajectory for all times.

Lecture 7: Reinforcement Learning: Policy Gradient, Baseline, Simple Examples

Recorded video: Lecture 7 (https://www.video.uni-erlangen.de/clip/id/11571)
Online session: **June 9, 6pm** Recording 126 MB
(https://owncloud.gwdg.de/index.php/s/799ls5pi3cmmGaN)

Contents: Reinforcement learning (agent, environment, state, action, reward, policy), policy gradient, baseline, simple example: random walker (with analytical solution)

After this lecture, you will understand how the important "policy gradient" algorithm for reinforcement learning is derived, how a baseline for the rewards can help, and you will have seen in a very simple example how this all works (not yet with neural networks).

However, you still need to see other techniques of reinforcement learning, and how "deep reinforcement learning" with neural networks works.

Notebook with solutions for lecture 6 homework
(discussed in online session 7)

Notebook: LSTM for predicting time evolution during second half of an interval (https://owncloud.gwdg.de/index.php/s/nAJnfaEKuXPBkxN) (also as pure python script (https://owncloud.gwdg.de/index.php/s/reBBgeN95LCr5Yx))

Given the trajectory of a single particle (coupled to others!) during [0,T], predict its evolution during [T,2T]. This requires implicitly understanding the initial conditions for all the particles, as well as the coupled equations of motion!

Notebook for online tutorials:
Tutorial Notebook: Reinforcement Learning with Policy Gradient: Treasures in a Maze (https://owncloud.gwdg.de/index.php/s/TbJU0Qgw4r76MIt) (also as pure python script (https://owncloud.gwdg.de/index.php/s/fXI6ChrMLLTNnI3))

A little robot wanders through a maze, searching for treasure chests! Will it discover the optimal strategy to get the maximum sum of rewards? This shows table-based reinforcement learning.

Lecture 8: Reinforcement Learning: Policy Gradient with Neural Networks, Q Learning

Recorded video: Lecture 8 (https://www.video.uni-erlangen.de/clip/id/11621)
Online session: **June 16, 6pm** Recording 150 MB
(https://owncloud.gwdg.de/index.php/s/vDLOQmMjPbZFtTw)

Contents: more examples for policy gradient, neural-network-based policy gradient (“deep policy gradient”), AlphaGo, Q learning

Notebook for online tutorials:
Tutorial Notebook: Reinforcement Learning with Neural-network-based Policy Gradient (https://owncloud.gwdg.de/index.php/s/WIMyFn709fkxmXC) (also as pure python script (https://owncloud.gwdg.de/index.php/s/qj8gG9GznQODWwc))

This shows neural-network-based reinforcement learning, converting an input image (a map showing the location of the robot) into action probabilities. Note: Unfortunately, at present this does not yet successfully solve arbitrary mazes, although in principle it is set up to tackle this challenge. Possibly a question of hyperparameter optimization, if there is no bug left.

Lecture 9: Boltzmann machines

Recorded video: Lecture 9 (https://www.video.uni-erlangen.de/clip/id/11647)
Online session: **June 23, 6pm** Recorded video (138 MB)
(https://owncloud.gwdg.de/index.php/s/8KyTxuBS6CRUOVI)

Updated version of all tutorial slides (11 MB)
(https://owncloud.gwdg.de/index.php/s/7uc5XCRYWcDfwD7)

Contents: Boltzmann machines as a way to generate samples from an observed probability distribution (e.g. images that look like images that have been observed), connection to statistical physics and Monte Carlo sampling, applications in physics

Notebook with solutions for lecture 8 homework
(discussed in online session 9)

Notebook: Improved deep policy gradient for solving mazes (https://owncloud.gwdg.de/index.php/s/yZQQjmeeVK6hi74) (also as pure python script (https://owncloud.gwdg.de/index.php/s/Mo1J79RoeMRP5pu))

That now works very nicely.

Notebook for online tutorials:
Tutorial Notebook: Boltzmann Machine (https://owncloud.gwdg.de/index.php/s/om63z9vknk195Bi) (also as pure python script (https://owncloud.gwdg.de/index.php/s/qsZJxJFqPu5bLTD)))

Train a Boltzmann machine to sample from a probability distribution that it has observed via training samples!

(this is really just a few lines of code)

Homework for Lecture 9
(will be discussed in online session for lecture 10; hint: use the forum to discuss and post your ideas and solutions!)

*(1) Apply the Boltzmann machine to the MNIST images! (explore the role of learning rate, batchsize, number of hidden units, etc.)

(2) **Challenge**: Who can get the highest return in the maze reinforcement learning? [simulation parameters: 15x15 maze, 10 treasure chests, 40 time steps, chests deleted; like in the last example in the 'improved' deep policy gradient notebook above, see there for typical values for the return!]

You can try to experiment with the network, with the learning rate and procedure, with the batchsize, etc.

Lecture 10: Neural Networks in Science, Artificial Scientific Discovery

Recorded video: Lecture 10 (https://www.video.uni-erlangen.de/clip/id/11735)
Online session: **June 30, 6pm** Recording (https://owncloud.gwdg.de/index.php/s/euimhfklySpwDo1)

Contents: General considerations in applying neural networks to science, example applications in various domains (like chemistry and medicine) and especially in physics (e.g. statistical physics), artificial scientific discovery as a long-term goal: what would be needed?

Notebook with solutions for lecture 9 homework
(discussed in online session 10)

Notebook: Boltzmann Machine applied to MNIST images
(https://owncloud.gwdg.de/index.php/s/FsqAqiS8a4D15e0) (also as pure python script (https://owncloud.gwdg.de/index.php/s/2RjCREAfbFz6drx))

Learn to reproduce images of digits, by training a restricted Boltzmann machine on the MNIST images (supplied with every keras installation).

Lecture 11: Applications to Quantum Science and Technology, Reinforcement Learning for Quantum Error Correction

Recorded video: Lecture 11 (https://www.video.uni-erlangen.de/clip/id/11761)
Online session: **July 14, 6pm** Recording (https://owncloud.gwdg.de/index.php/s/h7JIwczC8wL0yjQ)

Overview of some applications of machine learning to challenges in quantum science and technology, detailed example from our own research: applying deep reinforcement learning to quantum error correction

Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt, "Reinforcement Learning with Neural Networks for Quantum Feedback", Physical Review X, 031084 (2018)
(https://journals.aps.org/prx/abstract/10.1103/PhysRevX.8.031084)

This 2018 article was actually the first to apply deep reinforcement learning anywhere in quantum physics, together with a qubit control paper from another team (August and Hernandez-Lobato) that appeared practically simultaneously.

Lecture 12: Some further interesting aspects of machine learning for physicists

no pre-recorded video
Online session: **July 21, 6pm** Watch Recording (183 MB)
(https://owncloud.gwdg.de/index.php/s/ZOwuFnrVY6bz6Gw)

I plan to discuss a few arbitrarily selected topics. Among them probably: Keras custom layers, "mutual information" as a measure of statistical dependence and as a tool for unsupervised feature extraction (in the context of artificial scientific discovery), some remarks on quantum machine learning, some prospects for new classical physical hardware for machine learning.

Homework for FAU students
Download the 2019 exam (as an illustrative example)
(https://owncloud.gwdg.de/index.php/s/9A1H9rPqHsLRg9l)
Before lecture 13, please do this exam for practice (2 hours time). We will discuss the solutions in lecture 13. This year's exam may add/substitute some more programming-related questions, but a part of it will be similar in spirit. If you want more practice, you can also try this 2017 example exam (https://owncloud.gwdg.de/index.php/s/Wj0MUP6GZyu66GF).

Notebook shown in discussion
Notebook: Keras Custom Layers (https://owncloud.gwdg.de/index.php/s/w29lCFP0UHbtdMr), also as python script (https://owncloud.gwdg.de/index.php/s/lwuPZusGAYDRTzS)
This shows how to implement custom layers in keras, in an example that demonstrates a 1D convolutional layer with periodic boundary conditions.

Lecture 13: Exam preparation

no pre-recorded video
Online session: **July 28, 6pm** Watch Recording (192 MB)
(https://owncloud.gwdg.de/index.php/s/BoPj7SJFc0RD8pL)

We will go through one typical set of exam questions (partly from previous years). This is in preparation for the written exam for FAU students (that takes place on August 1st).

Lecture 14: Mini-projects presentations

no pre-recorded video
Online session: **August 4, 6pm** Recording (250 MB)
(https://owncloud.gwdg.de/index.php/s/gLqQ1B3L7cZBetc)

External students gave brief presentations of their mini-project results, covering a large range of topics!

Exam and Solutions

The full exam plus the solutions with comments
(https://pad.gwdg.de/2020_MachineLearningPhysicists_Exam_Solution)