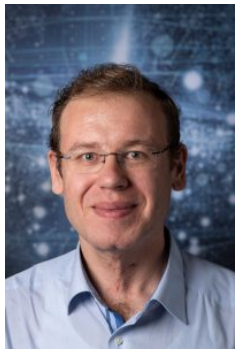
Numpy Tutorial

K. Breininger, V. Christlein, Z. Yang, L. Rist, M. Nau, S. Jaganathan, C. Liu, N. Maul, L. Folle, M. Zinnen, K. Packhäuser

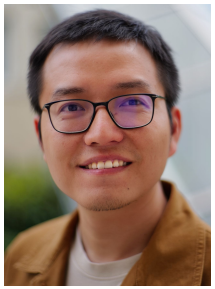
Pattern Recognition Lab, Friedrich-Alexander University of Erlangen-Nürnberg

October 1, 2024

Who are we? - Lab Members



Andreas
Maier



Chang
Liu

Alexander
Barnhill

Zijin
Yang

Leonhard
Rist

Merlin
Nau

Noah
Maul

Mathias
Zinnen

Who are we? - Student Tutors

Lukas
Hüttner

Majid
Sharghi

Mohannad
Barakat

Christian
Wielenberg

Haiting
Huang

Julian
Greil

Thakkar
Rahul
Jayantilal

Mohapatra
Maitreya

Organisation

Contact us ...

- in StudOn
- via the tutors mailing list: `cs5-deep-tutors@lists.fau.de`
- in MS Teams

Important: Don't hesitate to ask questions/give comments!

Online teaching

- Exercise will be completely online
- **Follow the Guide in StudOn**
  - You can also find all information under Organisational_concepts in StudOn homepage
- We will use MS Teams
  - Team activation in IDM required!
  - "General" channel for general questions and comments
  - "Private" channel for each exercise day
  - Direct support during exercise hours can be requested in resp. channel

Important: Feedback and suggestions are most welcome!

Semester plan

- Five exercises:
  0. Python + Numpy Recap and Data Generation
  1. Fully Connected Networks
  2. CNNs and Optimization
  3. Regularization and Recurrent Neural Networks
  4. Image Classification with PyTorch
- Platform: MS Teams

Semester plan

- Five exercises:
  0. Python + Numpy Recap and Data Generation
  1. Fully Connected Networks
  2. CNNs and Optimization
  3. Regularization and Recurrent Neural Networks
  4. Image Classification with PyTorch
- Platform: MS Teams
- Materials available in StudOn
- Each exercise takes 2-4 weeks $\rightarrow$ start early, submit early
- Bonus points for exam up to 10% (6 points). Unittests coverage determines your bonus points.
- Written exam (mock exam available in StudOn)

Submission

- Group submission possible - pairs of two
  → “Finding Group Partners" channel in **MS Teams**
- Personal submission only (max group of 2 people)
- **Required to get the bonus points**
- We prioritize the requests, have a look into our online course guide at StudOn
  → Organisational_concepts/03 – MS Teams
- Mind the deadlines !!!
  → please use the provided script (*dispatch.py*) to prepare your upload
- Upload your code to StudOn
- Explain your code (next week after uploading) (screen sharing)

No Plagiarism!

- One plagiarized case result in no bonus points for all exercises!
- We will check that with plagiarism software!
- Using code generated by ChatGPT can result in substantial overlap with other groups
- If you don't code yourself, you won't learn!

Verteilung - Exercise 4: AlexNet and ResNet in TF / AlexNet and ResNet

90% - 100%	1	#
80% - 90%	2	#
70% - 80%	13	##
60% - 70%	61	##########
50% - 60%	172	##############################
40% - 50%	245	##
30% - 40%	421	##
20% - 30%	314	##
10% - 20%	46	########
0% - 10%	0	.

Gruppierte Übereinstimmungen (90% - 100%)

[redacted]	[redacted]	97,23%

Exercise Setup

First part:

Build a neural network from scratch using test based development

- Implementation task is defined by **description** and **unit tests**
- No skeletons
- Every function and structure is built as a layer
  $\rightarrow$ As a class in its own file
  $\rightarrow$ Mandatory functions **__init__()**, **forward()**, **backward()**
- Unit tests help to expose bugs and errors
  $\rightarrow$ Tested and debugged with python3

Second part:

Build some common neural networks with PyTorch

- Some functionality provided
- No exhaustive unit tests

PyTorch

Schedule

Date	Event
21.10.2024	Handout all exercises
08.11.2024	Deadline Ex.0
22.11.2024	Deadline Ex.1
13.12.2024	Deadline Ex.2
19.01.2025	Deadline Ex.3
09.02.2025	Deadline Ex.4

A detailed time table could be found in StudOn with lecture suggestions

$\rightarrow$ Organisational_concepts/05 – Time table

- These deadlines are for code uploading. You have until the end of next week of each deadline to present your code

Bonus Points System

- Exercises contribute to max. 10% of bonus for the final exam
- Unittest coverage corresponds to bonus points
  → Each exercise consists of several TestCases
  → TestCases subdivide the bonus points
  → Each TestCase consist of several unittests
  → **All** unittests of one TestCase must pass to get the respective bonus
- The percentages correlate to effort and difficulty → but also the small bonuses add up
- The unittest files can compute the points for you → have a look in the description
- Be aware, some TestCases depend on others
  → It's impossible to test a neural network without having its layers first
- You only get the bonus if you submit in time! **Mind the deadlines!**

Bonus Points - Ex0

- Ex0 (1% in exam = 0.6 point from 60 points)
  - (10%) TestCircle
  - (10%) TestSpectrum
  - (10%) TestChecker
  - (70%) TestGen
- Percentage next to Ex0 represents the bonus points in the exam (here 1%)
- E.g., TestCircle contributes with 10% to the 1% (resulting in 0.1% for the exam)
- TestGen seems important with 70% and hence causes the most effort
- TestGen contains also more unittests than the others.

Bonus Points Distribution

- Ex1 (1.5% in exam = 0.9 point from 60 points)
  - (45%) TestFullyConnected1
  - (5%) TestReLU
  - (10%) TestSoftMax
  - (10%) TestCrossEntropy
  - (5%) TestOptimizers1
  - (25%) TestNeuralNetwork1 [1]
- Ex2 (3% in exam = 1.8 point from 60 points)
  - (45%) TestConv [2,3,4]
  - (15%) TestPooling [3]
  - (2%) TestFlatten
  - (5%) TestInitialization
  - (8%) TestOptimizers2 [0]
  - (2%) TestFullyConnected2 [0]
  - (23%) TestNeuralNetwork2 [0, 1]

Bonus Points Distribution

- Ex0 (1% in exam = 0.6 point from 60 points)
- Ex1 (1.5% in exam = 0.9 point from 60 points)
- Ex2 (3% in exam = 1.8 point from 60 points)
- Ex3 (3% in exam = 1.8 point from 60 points)
  - (2.5%) TestSigmoid
  - (2.5%) TestTanH
  - (5%) TestConstraints [2, 5]
  - (5%) TestDropout
  - (25%) TestBatchNorm [2, 3, 4, 6]
  - (40%) TestRNN [2, 4, 6]
  - (20%) TestNeuralNetwork3 [0, 1]
- Ex4 (1.5% in exam = 0.9 point from 60 points)
  - All Tests + Leaderboard f1 mean 0.6

Bonus Points Table

```
OK
..........
----------------------------------------------------------------------
Ran 10 tests in 6.853s

OK
========================== Statistics ==============================
| Pos   | Test            | Result   | Percent in Exercise   | Percent in Exam   |
|-------|-----------------|----------|-----------------------|-------------------|
| 0     | TestCheckers    | OK       | 10 / 10 (%)           | 0.100 / 10 (%)    |
| 1     | TestCircle      | OK       | 10 / 10 (%)           | 0.100 / 10 (%)    |
| 2     | TestSpectrum    | OK       | 10 / 10 (%)           | 0.100 / 10 (%)    |
| 3     | TestGen         | OK       | 70 / 70 (%)           | 0.700 / 10 (%)    |
|       |                 |          |                       |                   |
| Ex0   | Total Achieved  |          | 100 / 100 (%)         | 1.000 / 10 (%)    |
```

Example of bonus table for ex0. Tutors will assign your bonus points base on this table. See Description.pdf how to run this table.

Bonus Points - Dependencies

- [0] requires its predecessor (e.g. TestOptimizers1 requires TestOptimizers1)
- [1] requires all tests of current exercise
- [2] requires TestOptimizers1
- [3] requires TestFlatten
- [4] requires TestInitialization
- [5] requires TestOptimizers2
- [6] requires TestFullyConnected1

Bonus Points - Recommendation

Our recommmendation:

Do them all! Why?

1. You do not get confused by dependencies (it's easier to keep an overview if you do them all)

Bonus Points - Recommendation

Our recommmendation:

Do them all! Why?

1. You do not get confused by dependencies (it's easier to keep an overview if you do them all)
2. It is easier to get used to the framework if you do everything

Bonus Points - Recommendation

Our recommmendation:

Do them all! Why?

1. You do not get confused by dependencies (it's easier to keep an overview if you do them all)
2. It is easier to get used to the framework if you do everything
3. You get the maximum of bonus points - 10% (tentatively two grades).

Bonus Points - Recommendation

Our recommmendation:

Do them all! Why?

1. You do not get confused by dependencies (it's easier to keep an overview if you do them all)
2. It is easier to get used to the framework if you do everything
3. You get the maximum of bonus points - 10% (tentatively two grades).
4. We will cover content for the exam - thus highly relevant!

Bonus Points - Recommendation

Our recommmendation:

Do them all! Why?

1. You do not get confused by dependencies (it's easier to keep an overview if you do them all)
2. It is easier to get used to the framework if you do everything
3. You get the maximum of bonus points - 10% (tentatively two grades).
4. We will cover content for the exam - thus highly relevant!
5. Perform way better in the exam

Bonus Points - Recommendation (cont.)

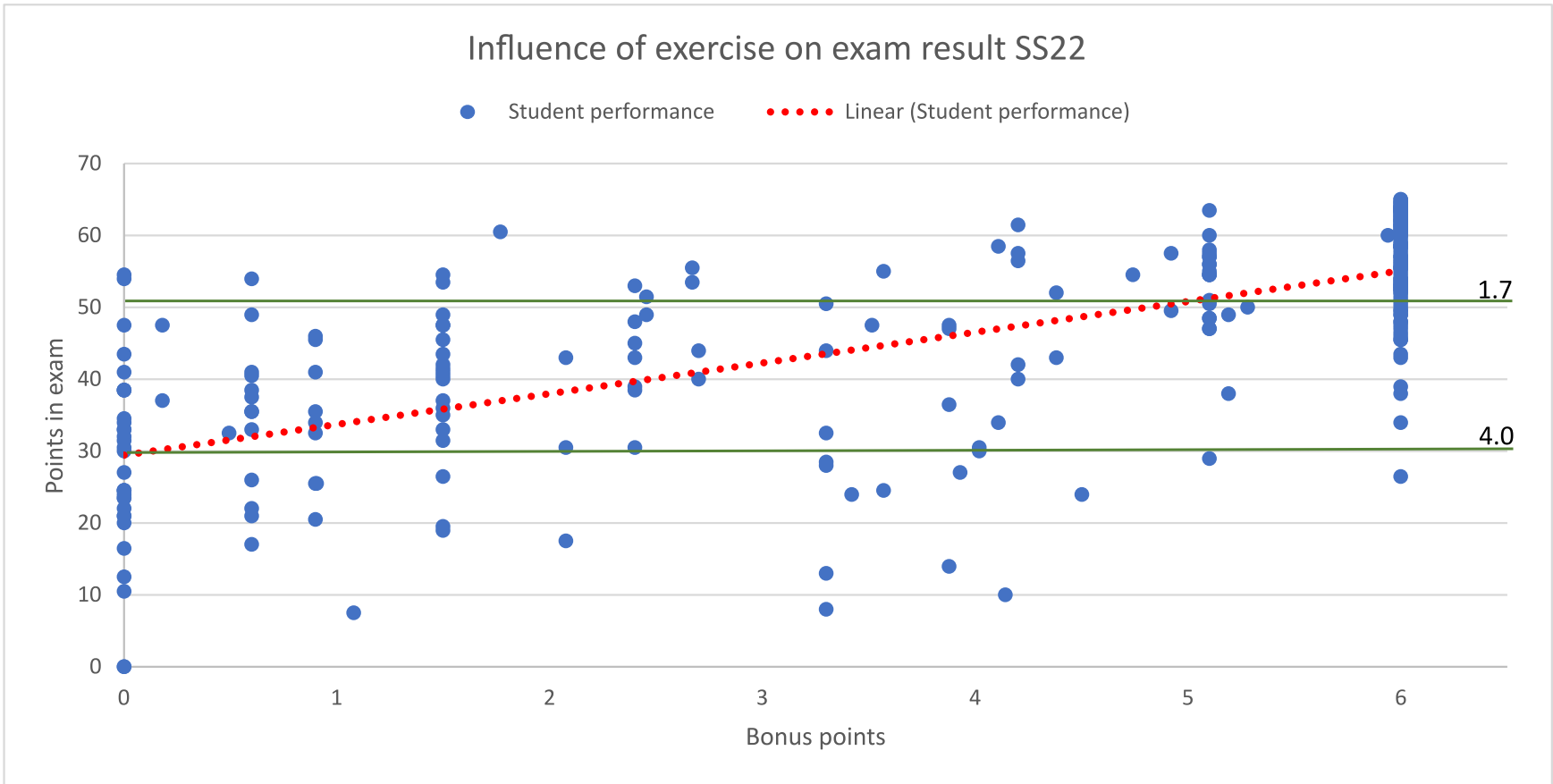


Figure: performance of 267 students from SS22. Students who completed all exercises perform much better than students who haven't completed the exercises

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Python Overview

About Python...

- Programming language with good readability
- Interpreted scripting language
  $\rightarrow$ Relies on the call of libraries written in lower-level programming languages
  $\rightarrow$ Basic programming semantics exist but are very inefficient
- Huge amount of libraries for all sorts of applications

About Numpy...

- Essential python package
- Central object: Numpy array
  $\rightarrow$ Acts like a matrix/vector
  $\rightarrow$ Enables all sorts of mathematical operations
  $\rightarrow$ Optimised for speed
- A cheat sheet with handy functions for this exercise can be found in the StudOn group

About Scipy...

- Python package closely linked to numpy
- Provides additional functionality
  $\rightarrow$ Signal processing
  $\rightarrow$ Statistical operations

Recommendations

Package Manager (not needed in CIPs)

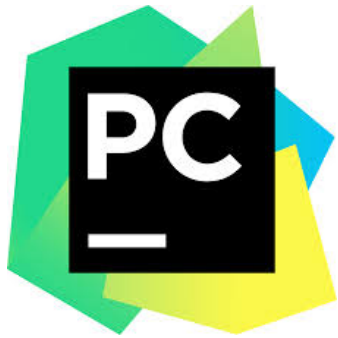
We recommend **Anaconda** (Windows)

- Open source
- One click installation
- Also installs python
- Easy handling of virtual environments

IDE

We recommend **PyCharm**

- Open source
- Easy package handling
- Debugging possibilities
- Free licenses for professional version for students



One alternative: Visual Studio Code with Live Share Plugin (allows remote pair programming)

Version Control

We recommend using GitLab!

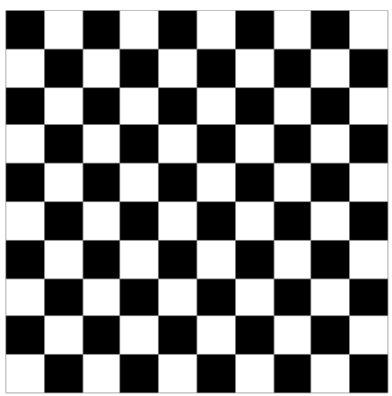
- Please use the university's gitlab server: https://gitlab.cs.fau.de/
- Perfect for co-working
- Compare your code with old versions
- Please use **private projects**! You can add your study partner as additional developer.

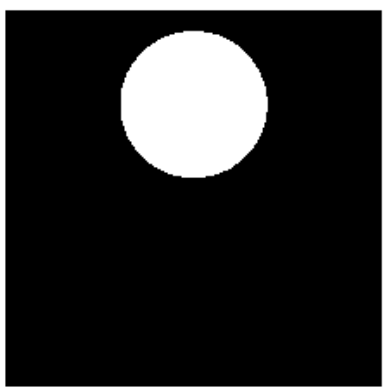
Today's Exercise

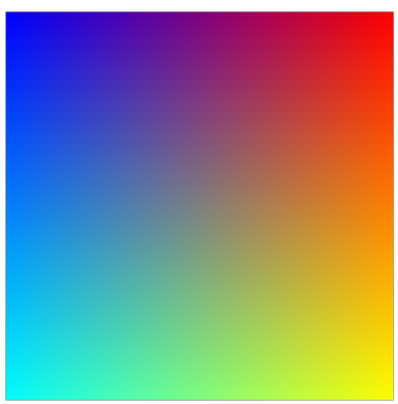
1st Task

Use basic numpy functions to create

- A binary checkerboard pattern
- A binary circle
- An RGB color spectrum







2nd Task

Often it is not possible or desired to train your neural network on the whole data set at once. $\rightarrow$ Divide your data into smaller portions.
Use numpy to implement an image generator class which also enables data augmentation.

- The generator yields so called batches (subsets of the training data) in an iterative manner.
- Batch in this context means a set of images, which are returned at once (by calling "next").
- These batches of images must be returned together with their corresponding labels.
- It returns batches until no training samples are left. One pass through the whole data set is also known as one epoch.

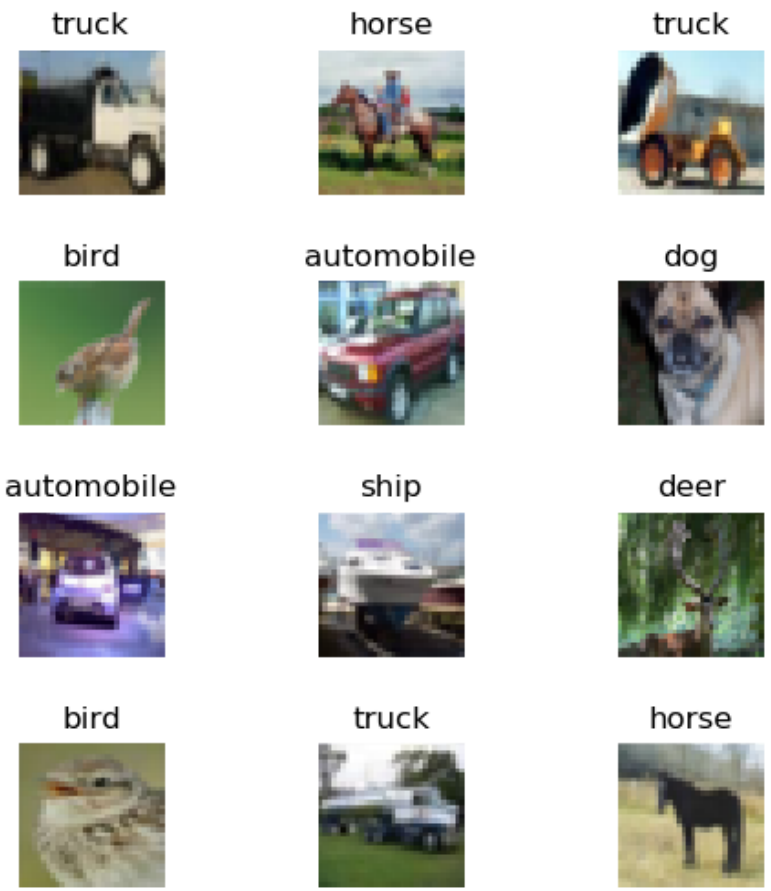


Figure: Example image generator output.

Get Started

- Open the IDE of your choice
- If you want to use PyCharm in the CIP:
  type **module load pycharm-community** into the console and open it by typing **pycharm**
- Follow the instructions of the exercise sheet
- Implement the tasks

Thanks for listening.

Any questions?