

OTTO-FRIEDRICH UNIVERSITY OF BAMBERG

CHAIR OF EXPLAINABLE MACHINE LEARNING

---

# Deep Learning Exercise

## Assignment 1

---



*Term:* WS 24/25

*Instructor:* Prof. Dr. Christian Ledig

*Teaching Assistants:* Sebastian Dörriech,  
Jonas Alle

*Start:* Monday, 14th of October 2024, 08:00am

*Deadline:* Sunday, 10th of November 2024, 11:59pm

*Discussions Theory:* [VC-Forum](#)

*Discussions Programming:* [VC-Forum](#)

---

## Guidelines

- We generally encourage you to collaborate with other students. You may talk to a friend or create a post in VC to discuss the questions and potential directions for solving them. However, this is not a group assignment. Hence, each student must write his/her/their own answers and explicitly mention any collaborators.
- We will apply anti-cheating software to check the submissions for plagiarism. Besides all submissions of the current semester, we further check for online resources as well as submissions of previous semesters. Anyone who is flagged by the software will automatically receive 0 points for the homework. If you feel that you were accused wrongly, please approach your TA immediately.
- Do NOT change the filenames and function definitions in the skeleton code provided. This will cause the test scripts to fail and results in you receiving 0 points for the respective task. You may also NOT change the import modules in each file or import additional modules.
- It is your responsibility to make sure that all code and other deliverables are in the correct format and that your submission compiles and runs. Failing to do so will result in the reduction of points.
- It is your responsibility to hand in the submission in time. Anyone failing to do so will automatically receive 0 points for the homework. If you were not able to submit your work due to illness or other reasons, please approach your TA immediately to clarify further action.
- We hope you enjoy your dive into the area of Deep Learning with its broad range of applications. Deep Learning is considered the driver of AI with its limitations having yet to be explored. Please have fun coding and solving the challenging questions, so you don't remain the dinosaur, Mark Cuban labeled you as back in 2017.



---

## Related Classes and Resources

- [Deep Learning, Georgia Institute of Technology](#)
- [Deep Learning, Eberhard Karls Universität Tübingen](#)
- [Deep Learning, FAU Erlangen-Nuremberg](#)
- [Deep Learning Specialization, DeepLearning.AI](#)
- [Deep Neural Networks with PyTorch, IBM](#)
- [Convolutional Neural Networks for Visual Recognition, Stanford](#)
- [Machine Learning, Oxford](#)
- [Deep Learning, New York University](#)
- [Deep Learning, CMU](#)
- [Deep Learning, University of Maryland](#)
- [Hugo Larochelle's Neural Networks class](#)



---

## Necessary Tools and Those That Make Your Life Easier

### PyCharm - The IDE for Python Programmers

If you have never coded in **Python** before, and even if you have, we recommend you to check out [PyCharm](#) (by [JetBrains](#)), a powerful IDE for programming in Python. [PyCharm](#) comes in a free community version as well as in billed professional one. The community version already includes all the features you might need. However, the professional version includes some additional ones which could come in handy. Therefore, you might want to check out the latter. The great deal about using software from JetBrains is that they offer [free educational licenses](#) for their products. Hence, as a student you can apply for such a licence and use all their billed products for free during your time of study.

[PyCharm](#) offers a bunch of features like allowing to look for the occurrences of a keyword anywhere in a project or allowing to split the tool window in order to maintain a good overview of the code. It also comes with a version control system and an in-built debugger to avoid the need for countless print statements.

### Jupyter Notebooks

Next to regular Python files, you will also be using [Jupyter Notebooks](#) throughout this class. Formerly known as IPython Notebooks, [Jupyter Notebooks](#) are an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and media. The combination of markdown and code cells allow such Notebooks to provide a clear guidance through entire coding projects while enabling the visualization of data and results next to their associated code. This allows for doing data science with Python for that IDEs like [PyCharm](#) are not ment for. For more details on [Jupyter Notebook](#), please see the official [Jupyter website](#) or this [beginner's guide](#) hosted on medium.

### DataSpell - The IDE for Data Scientists

For all of you who like writing code in a robust IDE like [PyCharm](#) which offers features such as autocompletion, spellchecker, version control, easy terminal access, database integrations, or a debugger, coding in [Jupyter Notebook](#) (which in fact lacks all of those) might seem like a nuisance to you. Hence, you might want to check out [DataSpell](#) (by JetBrains) and IDE for data scientists. In contrast to [PyCharm](#), unfortunately [DataSpell](#) does not come in a free community version but only in a payable one. However, since it is a product of JetBrains you can use it for free with your already obtained [free educational student license](#).

[DataSpell](#) offers great coding assistance (smart-code completion, error checking, etc.) and markdown support ("notebook style"). It has the ability to output visualizations without the need for extra installations and works well with interactive visualization libraries. It comes with a version control, terminal, database tools as well as a debugger and offers also the use of plugins.

---

## Google Colab

[Google Colab](#) is a free cloud service based on [Jupyter Notebooks](#) that supports the usage of free GPU resources provided by Google. This service enables anybody to develop deep learning applications without the need for purchasing expensive hardware. It has some limits with respect to the your session or file sizes but in this class we won't be using it to such an extent that you will have to worry about those. You can create notebooks in Colab, upload notebooks, store notebooks, share notebooks, and lots of other things. The only requirement is that you will need a free [Google account](#) which can be created easily in a couple of minutes since [Google Colab](#) is directly connected to and makes use of [Google Drive](#). If you have never worked with [Google Colab](#), please check out the following tutorials by [Towards Data Science](#) or [GeeksforGeeks](#) to understand how to set up a project in this framework.

We will be using [Google Colab](#) later in this class. But we still recommend to check it out as soon as possible and get familiar with its processes, so that you don't have to worry about it mid-semester when your assignments might start to pile up.

---

# Initializing Your Tools

## Python and Dependencies

In this assignment, we will work with **Python 3**. If you have never used Python before, please make yourself familiar with the language first by studying one of its numerous, free tutorials, e.g. the [Official Python Tutorial](#) or the [W3 Schools Python Tutorial](#). If you do not have a python distribution installed yet, we recommend installing [Miniconda](#) (or [Anaconda](#)) with Python 3. Refer to the [users' manual](#) for more details about the use of conda. We will provide you with the file `environment.yaml` which contains the libraries needed for this assignment. You can further use it to create an assignment-specific conda environment which will help you get rid off any potential import or package version errors. To do so, open the [Anaconda Prompt](#) within the folder for this assignment and run the following command:

```
$ conda env create -f environment.yaml
```

It is important to note that [PyTorch](#) which we will be using for training and testing of our networks is currently not working for Python versions > 3.9. To avoid package dependency issues when creating your conda environment we have specified the use of Python 3.9 explicitly in `environment.yaml`.

If you already have your own Python development environment, please refer to the environment file to find the necessary libraries for this assignment.

## PyCharm

To create a [PyCharm](#) project you have to execute the following steps:

1. If you have never opened [PyCharm](#) you will be forwarded to a "Welcome screen" after running the application for your first time. Click **New Project**.  
If you've already worked with [PyCharm](#) and you run the application, you will always be forwarded to your last project page. Here you have to select **File** → **New Project** from the main menu.
2. To initialize the already created conda environment as your interpreter you have to select **Pure Python** → **Previously configured interpreter**. If your conda environment is not shown in the dropdown menu, you have to select ... → **Conda Environment** →  
**Interpreter: C:\...\.conda\envs\xai-dl-a1\python.exe** and click **Ok**
3. Choose the project location. Enter the path to your local **Assignment 1** copy within **Location:** to specify the directory for your project and hit **Create** afterwards.
4. You will get a notification saying **Directory is Not Empty - Do you want to create project from existing sources?**. This is fine, you can simply click **Create from Existing Sources** and start your project in [PyCharm](#).

---

## Jupyter

If you used the `environment.yaml` file to create your development environment for this assignment, you have already installed **Jupyter Notebook** and you don't have to do anything else.

If you used your own environment and don't have **Jupyter Notebook** yet, install it using one of the following commands:

```
$ conda install -c anaconda jupyter
$ pip install jupyter
```

After installing **Jupyter Notebook** successfully, to run it in your browser, first you have to activate your conda environment within the **Anaconda Prompt**:

```
$ conda activate xai-dl-a1
```

This ensures that you run your **Jupyter Notebook** with the Python version and packages we need for the assignment. This helps to avoid import and package errors. Afterwards navigate to the directory of your local **Assignment 1** copy. Here you have to enter

```
$ jupyter notebook
```

within the **Anaconda Prompt** to start **Jupyter Notebook** automatically in your browser. **Jupyter Notebook** will always open within the folder you have specified before by navigating to the desired directory.

## DataSpell

To create a **DataSpell** project you have to execute the following steps:

1. After starting the application you have to select **File** → **Open** from the main menu and specify the path to your local **Assignment 1** copy.
2. You have to specify the interpreter for your **DataSpell** project. Hence, you have to select **<No Interpreter>** from the bottom-right and specify your conda environment. If your conda environment is not within the dropdown menu of **Python Interpreter**:, you have to click **Add Interpreter** → **Add Local Interpreter...** and specify the path to your **xai-dl-a1** environment.

## Google Colab

The most important commands are:

1. Mount your drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

- 
2. Test whether your GPU is running:

```
!nvidia-smi
```

3. Run external Python files with command line parameters:

```
!python "/content/drive/.../file.py" --para1 "..."
```



---

## Submission Instructions

Besides checking your submissions by hand, we will be using automated tests, too. Please read the following instructions for submitting your solutions carefully, since failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failing to follow the instructions.

All exercises or questions marked with **[Bonus]** do not have to be done to achieve the maximum amount of points for their respective assignment. However, these tasks can be used to replace another task of the same assignment or guarantee that you achieve the maximum amount of points possible. All bonus tasks can only be used for their respective assignment and can not be used to improve the results in a later assignment. This means one can at max get 100 Points for each assignment.

## Theory

For the theory part of the homework you have to upload one single PDF containing the answers to all the questions. Please note that the solution to each problem/sub-problem must be on a separate page. This will help us to grade your submissions in less time. Please use the provided template to create your submission.

Please also state any collaborators you have exchanged your thoughts and ideas with under the section **Colaborators**. If you have worked completely on your own just enter **None** in that section.

After finishing the theory part, generate the resulting PDF and submit it via VC.

However, if you are tasked with running experiments in the programming section of this assignment, you will need to create the resulting report, add it at the end of your theory submission PDF to create a single PDF and submit this in the end via VC.

For the report you will use the provided template to collect and summarize your experimental results and findings. Simply go through all of the provided questions within the template to do so. For whichever questions asking for plots, please include plots from all your experiments.

Explanations should go into why things work the way they do with proper deep learning theory. For example, with hyperparameter tuning you should explain the reasoning behind your choices and what behavior you expected. If you need more than one slide for a question, you are free to create new slides **right after the given one**.

In the end you will export your report in PDF format and combine it with your submission of the theory questions to a single PDF before submitting the resulting PDF via VC.

---

## Coding

For your convenience and to assure that your code can be processed by our test cases you are only allowed to write code between the `# YOUR CODE HERE` comments:

```
#####
#                               START OF YOUR CODE                               #
# TODO:                                                                    #
#     ...                                                                    #
#####
.
.
.
#####
#                               END OF YOUR CODE                               #
#####
```

For Jupyter Notebooks or within DataSpell, after writing your code, you can run the cell by either pressing SHIFT+ENTER or by clicking on "Run Cell" (denoted by a play symbol) in the upper bar of the notebook.

*It is important to note that after you modified the code within a cell you always have to run that cell again before the changes are recognized*

After passing all provided test, make sure you don't forget to upload your submission to VC. For that you will need to submit a zip file containing all your codes. For your convenience, we provide a handy script for you that does that directly. Use the "collect\_submission"-script, which checks all files for completeness and zips the files you need for the upload.

Try running:

```
$ bash collect_submission.sh
```

or if you are using a Windows machine, try running:

```
$ C:\...\assignmentfolder> collect_submission.bat
```

Before submitting your assignment, please make sure you are not doing the following:

1. You have not added any *extra* print statement(s) in the assignment.
2. You have not added any *extra* code cell(s) in the assignment.
3. You have not changed any of the function parameters.
4. You are not using any global variables inside your graded exercises. Unless specifically instructed to do so, please refrain from it and use the local variables instead.
5. You are not changing the assignment code where it is not required, like creating extra variables.

If you do any of the mentioned, our test scripts will fail and as a result you will receive **0 points** for the respective task.

---

## Motivation

The purpose of this exercise is to get you familiar with Python, NumPy, PyTorch as well as Jupyter Notebooks or refresh your knowledge about those. Even if you have used them before, this exercise will help you build an understanding of the functions and tools we are going to need during this class.

## Theory [1 point]

This assignment does not include any theory questions. Nonetheless, we can practice the submission process here, so you don't run into any issues later in the course. Hence, please fill out the provided template `writeup.tex` with your name and BA-number. Furthermore, please list any collaborators you have exchanged your thoughts and ideas with. Afterward, generate the respective PDF and submit it via VC.

## Code [99 points]

### Overview

In this assignment you will first get an introduction to the [Numpy](#) package provided by Python. Afterward you will explore how to use the previously learned [Numpy](#) functions on a real task as well as learn how to create and work with class objects. The third part of the assignment gives you an introduction to the basics of [Pytorch](#) tensor operations. Whereas the fourth and last part gives you an introduction on how to compute derivatives and gradients for these [Pytorch](#) tensors.

For all assignment parts, unit tests are available which will help you to assess whether your implementation is correct or not. For all [Jupyter Notebooks](#), we have implemented test cells directly beneath the code snippets you are going to implement. You can just run these test cells to test your implementations. Besides that many IDEs (e.g., [PyCharm](#)) offer the option to run unit tests directly. Additionally, you can run via the command line. However, passing all local tests neither means your code is free of bugs nor guarantees that you will receive full credits for the coding section. Your code will be tested via additional tests which are not present in your local unit tests.

#### ToDo:

1. Implement the Jupyter Notebook [numpy\\_basics.ipynb](#).
2. Implement the Jupyter Notebook [numpy\\_applied.ipynb](#).
3. Implement the Jupyter Notebook [torch\\_tensors\\_basics.ipynb](#).
4. Implement the Jupyter Notebook [derivatives\\_and\\_graphs.ipynb](#).

# 1 Numpy Basics [24 points]

**Numpy** is the main, fundamental package for scientific computing in Python. It is well-optimized, fast, easy to use and provides lots of useful mathematical functions as well as linear algebra routines. Thanks to its fast and versatile vectorization, indexing and broadcasting concepts it poses the standard for array computing in Python. You will get to know some of the most important functions and learn how to use these which in the end will help you succeed in future assignments.

ToDo:

1. Follow the instructions in `numpy_basics.ipynb` to finish this exercise.

## 1.1 Basic Numpy Functions [24 points]

### 1.1.1 General Array Properties [1 point]

Follow the instructions in the method `extract_array_dimensions` to extract the dimensions of a given Numpy-array.

### 1.1.2 Creating Arrays [2 points]

Follow the instructions in the method `create_numpy_arrays` to create numpy arrays via various ways.

### 1.1.3 Array Handling [4 points]

Follow the instructions in the method `handle_numpy_arrays` to handle and combine numpy arrays.

### 1.1.4 Calculating with Arrays [5 points]

Follow the instructions in the method `calculations_with_numpy_arrays` to use numpy arrays for calculations.

### 1.1.5 Statistics with Arrays [1 point]

Follow the instructions in the method `statistics_with_numpy_arrays` to calculate statistics on numpy arrays.

### 1.1.6 Array Slicing 1D [5 points]

Follow the instructions in the method `slicing_of_numpy_arrays_1d` to apply smart indexing and slicing to extract specific values from 1D-arrays, like the one presented in Figure 1.

1	5	6	9	13	4	-5	17	-3	2	3	7	-5
---	---	---	---	----	---	----	----	----	---	---	---	----

**Figure 1:** Example for a 1D numpy array.

### 1.1.7 Array Slicing 2D [6 points]

Follow the instructions in the method [slicing\\_of\\_numpy\\_arrays\\_2d](#) to apply smart indexing and slicing to extract specific values from 2D-arrays, like the one presented in [Figure 2](#).

-1	15	4	6
5	-4	17	0
3	3	-8	6
7	-1	-10	2

**Figure 2:** Example for a 2D numpy array.

## 2 Numpy Applied [25 points + 10 points (Bonus)]

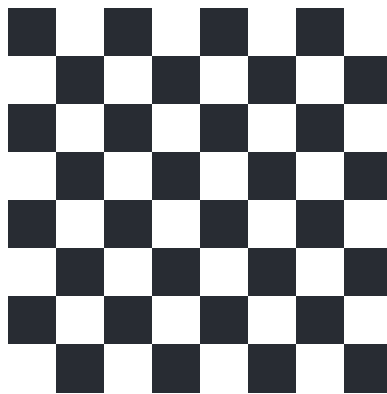
In this exercise you will apply the previously learned basics on real tasks.

ToDo:

1. Follow the instructions in `numpy_applied.ipynb` to finish this exercise.

### 2.1 Checkerboard Pattern [7.5 points]

In this first exercise you will be implementing a checkerboard pattern in the class `CheckerBoard` with an adaptable `size` and `tile_size`. An example of how the result shall look like can be seen in Figure 3:



**Figure 3:** Example for the checkerboard pattern.

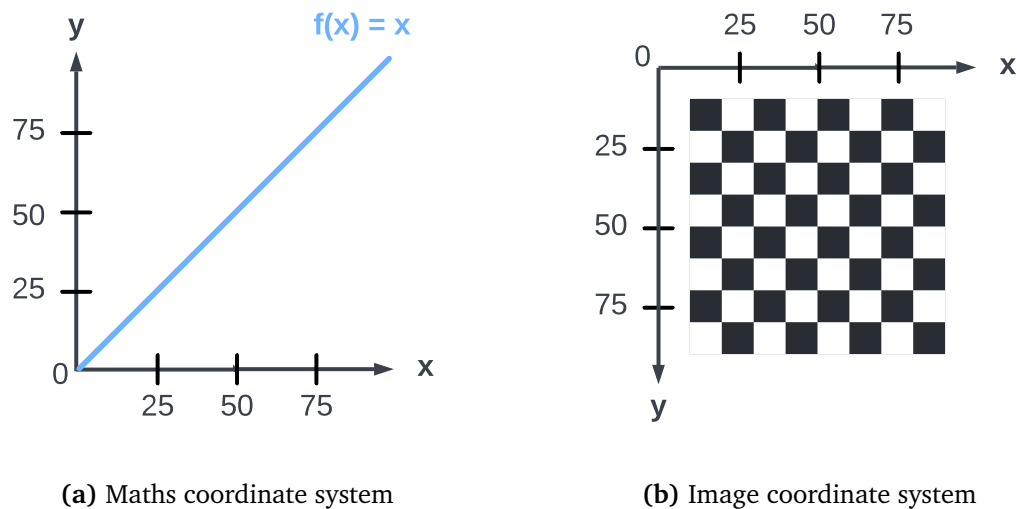
Please note that if we are talking about images, the coordinate system differs slightly from what you are used to from other (math) classes. In maths, if we talk about 2D functions within the Cartesian Coordinate System, the origin of the coordinate system is in the lower left corner with the x-axis ranging from left to right and the y-axis ranging from bottom to top.

However, for images the origin of the coordinate system is in the upper left corner instead with the x-axis ranging from left to right and the y-axis ranging from top to bottom. Furthermore, for images, a particular pair of (x-coordinate, y-coordinate) describes a single pixel within the image.

As a result, this means that when you have to deal with x- and y-coordinates of images, you have to treat them differently when dealing with x- and y-coordinates for functions. See figure 4 for reference.

Hints:

1. `__init__` represents the constructor of the class.
2. Following functions might be useful: `np.zeros`, `np.ones`, `np.tile`, `np.arange`, `np.concatenate`, `np.expand_dims`.



**Figure 4:** Visualization of the origin and axes differences of the two coordinate systems.

3. To simplify things we assume that the `size` is divisible by the `tile_size` without remainder.
4. Think about how the relation between `size` and `tile_size` has to look like to create a valid checkerboard pattern.

#### Attention:

If you are using the dark mode of your development environment instead of the standard light one, it will most likely happen that the appearance of the colors white and black are completely switched. This means everything written in black is shown in white and vice versa. This affects the appearance of text but might as well affect the appearance of plots you will generate throughout this exercise. This means a figure which should be rendered in the color black will be rendered in the color white and vice versa. However, this accounts only for figures you will create by running parts of the code below. Figures that were provided by us will still appear the way they are, regardless which appearance mode you are using within your development environment. Please keep that in mind if you want to use your development environment's dark mode!

#### 2.1.1 Initialize the Checkerboard Pattern [1 point]

Follow the instructions in the method `__init__` to implement the constructor.

##### Parameters:

- `size`: Defines the width and height of our checkerboard in pixels (int).
- `tile_size`: Defines the width and height of each individual tile in pixels (int).

##### ToDo:

## 2.1 Checkerboard Pattern [25 POINTS + 10 POINTS (BONUS)]

1. Implement the `constructor`.
2. Store each of the given parameter in a class variable of class `CheckerBoard`.
3. Create an additional class variable with the name `pattern` that will store the final checkerboard pattern.

### 2.1.2 Create the Checkerboard Pattern [5.5 points]

Follow the instructions in the method `draw` which constructs the checkerboard pattern as a numpy array.

Parameters: —

ToDo:

1. Implement the method `draw` which constructs the checkerboard pattern as a numpy array.
2. Black tiles shall have the value 0 and white tiles shall have the value 1.
3. Make sure that the tile in the top left is colored black.
4. To avoid truncated patterns, make sure your code only allows values for `size` that are evenly dividable by `2*tile_size`.
5. Store the created pattern in the variable `pattern`.

### 2.1.3 Visualize the Checkerboard Pattern [1 point]

Follow the instructions in the method `show` which will visualize our previously created checkerboard pattern.

Parameters: —

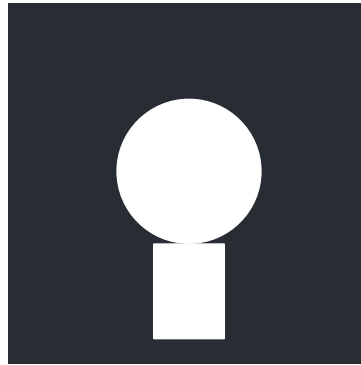
ToDo:

1. Implement the method `show` which shows the checkerboard pattern as an image.
2. Display the checkerboard pattern as a grayscale image by using the `imshow` function of Matplotlib with the respective parameter.



## 2.2 Shapes [17.5 points]

In this second exercise you will be implementing a binary pattern consisting of a white circle, and white rectangle in front of a black background. An example of how the result can look like can be seen in Figure 5:



**Figure 5:** Example for the shapes pattern.

### Hints:

1. For code-generated images and plots the axes' orientations differ from the ones you may know from math classes. Here, we use the pixel coordinate system. Conventionally the coordinate (0, 0) is located in the upper left corner of the image, with the x-axis ranging from left to right (along the columns of the image) and the y-axis ranging from top to bottom (along the rows of the image).
2. For the rectangle shape, take into account that the given x1 and y1 coordinates could be smaller than the respective x2 and y2 coordinates.
3. For the circular shape, think of a formula describing the circle with respect to pixel coordinates.

### Attention:

If you are using the dark mode of your development environment instead of the standard light one, it will most likely happen that the appearance of the colors white and black are completely switched. This means everything written in black is shown in white and vice versa. This affects the appearance of text but might as well affect the appearance of plots you will generate throughout this exercise. This means a figure which should be rendered in the color black will be rendered in the color white and vice versa. However, this accounts only for figures you will create by running parts of the code below. Figures that were provided by us will still appear the way they are, regardless which appearance mode you are using within your development environment. Please keep that in mind if you want to use your development environment's dark mode!

### 2.2.1 Create the Rectangle Shape [6 points]

As a first step we will be implementing the class `Rectangle` which creates a white rectangle in front of a black background. An example of how the result can look like can be seen in Figure 6:



Figure 6: Example for the rectangle pattern.

#### Parameters:

- `size`: Defines the width and height of our pattern in pixels (int).
- `x1y1`: Defines the x- and y-coordinate in pixels of one corner of the rectangle (tuple).
- `x2y2`: Defines the x- and y-coordinate in pixels of the opposing corner to the `x1y1`-specified corner of the rectangle (tuple).

#### ToDo:

1. Implement the `constructor`.
2. Implement the method `draw()` which constructs the pattern as a numpy array.
3. Implement the method `show()` which plots the pattern.

#### Hints:

1. Take into account that the given `x1` and `y1` coordinates could be smaller than the respective `x2` and `y2` coordinates.

### 2.2.2 Create the Circle Shape [7.5 points]

As the second step we will be implementing the class `Circle` which creates a white circle in front of a black background. An example of how the result can look like can be seen in Figure 7:

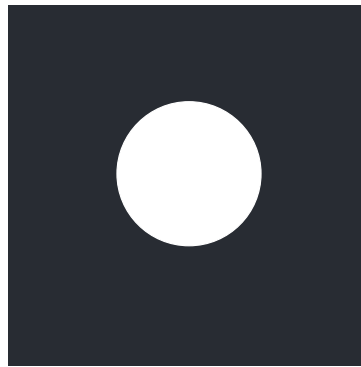


Figure 7: Example for the circle pattern.

#### Parameters:

- `size`: Defines the width and height of our pattern in pixels (int).
- `radius`: Defines the radius of the circle in pixels (int).
- `center`: Defines the x- and y-coordinate of the circle center in pixels (int).

#### ToDo:

1. Implement the `constructor`.
2. Implement the method `draw()` which constructs the pattern as a numpy array.
3. Implement the method `show()` which plots the pattern.

#### Hints:

1. Think of a formula describing the circle with respect to pixel coordinates.

## 2.2 Shapes [17.5 points] NUMPY APPLIED [25 POINTS + 10 POINTS (BONUS)]

### 2.2.3 Combine and Plot the Shapes [4 points]

As the last step we will call the previously created classes `Rectangle` and `Circle` to create the wanted shape pattern.

#### Parameters:

- `size`: Defines the width and height of our pattern in pixels (int).
- `x1y1`: Defines the x- and y-coordinate in pixels of one corner of the rectangle (tuple).
- `x2y2`: Defines the x- and y-coordinate in pixels of the opposing corner to the `x1y1`-specified corner of the rectangle (tuple).
- `radius`: Defines the radius of the circle in pixels (int).
- `center`: Defines the x- and y-coordinate of the circle center in pixels (int).

#### ToDo:

1. Create the individual patterns of `Rectangle` and `Circle`.
2. Combine the patterns in one image and plot the latter as a gray-scale image.

#### Hints:

1. Think about how to process overlapping shapes in one image to keep the binary style of the pattern.

## 2.3 Bonus: Color Spectrum [10 points]

In this third bonus exercise you will be implementing an RGB color spectrum in the class `Spectrum`. Since this exercise is optional it won't count towards your grade on this assignment. However, it might be quite interesting and fun to code, so feel free to do it! An example of how the result should look like can be seen in Figure 8:



**Figure 8:** Example for the spectrum pattern.

### Parameters:

- `size`: Defines the width and height of our pattern in pixels (int).

### ToDo:

1. Implement the `constructor`.
2. Implement the method `draw()` which constructs the pattern as a numpy array.
3. Implement the method `show()` which plots the pattern.

### Hints:

1. RGB images have 3 channels.
2. A spectrum consists of rising values across a specific dimension.
3. The intensity minimum and maximum should be 0.0 and 1.0, respectively, for each of the three color channels.
4. Check out the corners of the reference image to figure out the underlying distribution of channels.

### 3 Torch Tensors Basics [25 points]

This third part of the exercise gives you an introduction to the basics of tensor operations. Tensors are an essential part of PyTorch which we will use later in the course to build our neural networks. Hence, understanding the fundamentals now will help you succeed in later assignments.

ToDo:

1. Follow the instructions in `torch_tensors_basics.ipynb` to finish this exercise.

#### 3.1 Install Torch [1 point]

Please note that the provided `environment.yaml` file does not contain a [PyTorch Installation](#) for you. This is due to the fact that you may run the code on a **CPU** or **GPU**, or for a different **CUDA version**. Hence, as a first step, please refer to the [official documentation](#) of **PyTorch** to install the version that is suited best for you and your equipment.

#### 3.2 Tensor Basics [11 points]

**PyTorch** is an open source machine learning framework which is used for applications such as computer vision and natural language processing. It is a free and open-source software which provides the following two high-level features:

- Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU)
- Deep neural networks built on a tape-based automatic differentiation system

In this exercise you will learn some basic functions the torch package provides you with to create, define and index tensors.

##### 3.2.1 Create a Tensor [3.5 points]

In this exercise you will learn different ways of how to create a tensor.

Parameters:

- `shape`: Defines the shape of the tensor (tuple).
- `lst`: Defines an n-dimensional list of values (list).
- `a`: Defines an n-dimensional numpy array of values (`np.ndarray`).

ToDo:

1. Implement the method `create_tensor_with_specified_shape` which creates a tensor with a specified `shape`.
2. Implement the method `create_tensor_from_list` which creates a tensor from a given list `lst`.
3. Implement the method `create_tensor_from_numpy_array` which creates a tensor from a given numpy array `a`.
4. Implement the method `create_random_tensor` which creates a tensor of a specified shape `shape` with random values.
5. Implement the method `create_tensor_filled_with_zeros` which creates a tensor of a specified shape `shape` filled with zeros.
6. Implement the method `create_tensor_filled_with_ones` which creates a tensor of a specified shape `shape` filled with ones.
7. Implement the method `create_tensor_with_the_diagonal_filled_with_ones` which creates a 2D-tensor with a specified number of rows and columns `shape` for which its diagonal is filled with ones.

### 3.2.2 Extract Tensor Information [2.5 points]

In this exercise you will learn how to extract information from an existing tensor.

Parameters:

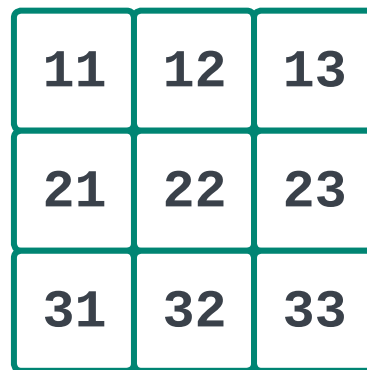
- `t`: Defines the tensor (`torch.tensor`).

ToDo:

1. Implement the method `extract_the_type_of_a_tensor` which extracts the type of a given tensor `t`.
2. Implement the method `extract_the_data_type_of_a_tensor` which extracts the data type of a given tensor `t`.
3. Implement the method `extract_the_dimensions_of_a_tensor` which extracts the dimension of a given tensor `t`.
4. Implement the method `extract_the_shape_of_a_tensor` which extracts the shape of a given tensor `t`.
5. Implement the method `extract_the_number_of_elements_of_a_tensor` which extracts the number of elements of a given tensor `t`.

### 3.2.3 Indexing and Slicing [5 points]

In this exercise you will learn how to access the different elements of a tensor. For this task, we will consider the following two-dimensional tensor which can be seen in Figure 9:



11	12	13
21	22	23
31	32	33

Figure 9: Used 2D tensor.

#### Parameters:

- `t`: Defines the tensor (`torch.tensor`).

#### ToDo:

1. Implement the method `slicing_01` to extract the first slice (specified within the notebook) of the given tensor `t`.
2. Implement the method `slicing_02` to extract the second slice (specified within the notebook) of the given tensor `t`.
3. Implement the method `slicing_03` to extract the third slice (specified within the notebook) of the given tensor `t`.
4. Implement the method `slicing_04` to extract the fourth slice (specified within the notebook) of the given tensor `t`.



### 3.3 Tensor Calculations [3 points]

We can also do calculations with tensors. In this exercise you will learn some of the basic calculations that can be done with tensors.

#### 3.3.1 Tensor Addition [0.5 points]

In this exercise you will learn how to add two tensors together.

Parameters:

- `t1`: Defines the first tensor (`torch.tensor`).
- `t2`: Defines the second tensor (`torch.tensor`).

ToDo:

1. Implement the method `tensor_summation` which calculates the sum of two given tensors `t1` and `t2`.

#### 3.3.2 Scalar Multiplication [0.5 points]

In this exercise you will learn how to multiply a scalar with a tensor.

Parameters:

- `t`: Defines a tensor (`torch.tensor`).
- `s`: Defines a scalar (`float`).

ToDo:

1. Implement the method `scalar_multiplication` which multiplies a given tensor `t` with a given scalar `s`.

#### 3.3.3 Hadamard Product [0.5 points]

In this exercise you will learn how to calculate the hadamard product of two tensors by multiplying them element-wise.

Parameters:

- `t1`: Defines the first tensor (`torch.tensor`).
- `t2`: Defines the second tensor (`torch.tensor`).

ToDo:

1. Implement the method `hadamard_product` which calculates the hadamard (element-wise) product of two given tensors `t1` and `t2`.

### 3.3.4 Matrix Multiplication [0.5 points]

In this exercise you will learn how to calculate the matrix product of two tensors.

Parameters:

- `t1`: Defines the first tensor (`torch.tensor`).
- `t2`: Defines the second tensor (`torch.tensor`).

ToDo:

1. Implement the method `matrix_product` which calculates the matrix product of two given tensors `t1` and `t2`.

### 3.3.5 Dot Product [0.5 points]

In this exercise you will learn how to calculate the dot product of two tensors.

Parameters:

- `t1`: Defines the first tensor (`torch.tensor`).
- `t2`: Defines the second tensor (`torch.tensor`).

ToDo:

1. Implement the method `dot_product` which calculates the dot product of two given 1D tensors `t1` and `t2`.

### 3.3.6 Square Tensor [0.5 points]

In this exercise you will learn how to square all elements in a tensor.

Parameters:

- `t`: Defines a tensor (`torch.tensor`).

ToDo:

1. Implement the method `square_tensor` which squares all elements of a given tensor `t`.

### 3.4 Tensor Statistics [2 points]

We can also do statistics with tensors. In this exercise you will learn some of the basic commands.

#### 3.4.1 Tensor Min [0.5 points]

In this exercise you will learn how to find the minimum value in a tensor.

Parameters:

- `t`: Defines a tensor (`torch.tensor`).

ToDo:

1. Implement the method `minimum` which finds the minimum value within a tensor `t` and returns it as another tensor.

#### 3.4.2 Tensor Max [0.5 points]

In this exercise you will learn how to find the maximum value in a tensor.

Parameters:

- `t`: Defines a tensor (`torch.tensor`).

ToDo:

1. Implement the method `maximum` which finds the maximum value within a tensor `t` and returns it as another tensor.

#### 3.4.3 Tensor Mean [0.5 points]

In this exercise you will learn how to find the mean value of a tensor.

Parameters:

- `t`: Defines a tensor (`torch.tensor`).

ToDo:

1. Implement the method `mean` which finds the mean value of a tensor `t` and returns it as another tensor.

Hints:

1. You may need to cast the given tensor to another datatype before the mean can be calculated...

#### 3.4.4 Tensor Standard Deviation [0.5 points]

In this exercise you will learn how to find the standard deviation of a tensor.

Parameters:

- `t`: Defines a tensor (`torch.tensor`).

ToDo:

1. Implement the method `standard_deviation` which finds the standard deviation of a tensor `t` and returns it as another tensor.

Hints:

1. You may need to cast the given tensor to another datatype before the mean can be calculated...

### 3.5 Tensor Operations [8 points]

We can further do additional operations with tensors which are needed to avoid shape or type errors during calculations. In this exercise you will learn some of the most important commands for tensors.

#### 3.5.1 Casting [3 points]

In this exercise you will learn how to apply casting on tensors.

Parameters:

- `t`: Defines a tensor (torch.tensor).

ToDo:

1. Implement the method `cast_to_list` which casts a tensor `t` to a python list having the same dimensions.
2. Implement the method `cast_to_numpy` which casts a tensor `t` to a numpy array having the same dimensions.
3. Implement the method `cast_to_double` which casts a tensor `t` containing integer values to a tensor containing double values.

#### 3.5.2 Transform a Tensor [4 points]

In this exercise you will learn how to transform a tensor to a different shape or a different arrangement of its elements. These transformations can be necessary to allow the combination of tensors or to enable the calculations within the neural networks we will be working with later on in the course.

Parameters:

- `t`: Defines a tensor (torch.tensor).
- `shape`: Defines the new shape (tuple).

ToDo:

1. Implement the method `reshape` which reshapes a tensor `t` to a new shape `shape`.

2. Implement the method `swap` which swaps the first axis (dimension) of a 3D-tensor `t` with its last axis (dimension).
3. Implement the method `trans` which transposes a 2D-tensor `t`.
4. Implement the method `pave` which paves an nD-tensor `t` to a 1D tensor.

### 3.5.3 Concatenate Tensors [1 point]

In this exercise you will learn how to concatenate tensors together to one single tensor.

#### Parameters:

- `t1`: Defines the first tensor (torch.tensor).
- `t2`: Defines the second tensor (torch.tensor).
- `ax`: Defines the concatenation axis (int).

#### ToDo:

1. Implement the method `concat` which concatenates two tensors `t1` and `t2` along the given axis `ax`.

#### Hints:

1. Some tensor combinations can not be concatenated together directly but require some transformations first (e.g transposing the second tensor to match the shape of the first one).
2. Some tensor combinations can never be concatenated together and should be dealt with accordingly (e.g. raise an error).

## 4 Derivatives and Graphs [25 points]

This fourth and last part of the assignment gives you an introduction on how to compute derivatives and gradients for Pytorch tensors. When applying deep neural networks, one always has to compute gradients of multiple different network parameters. Hence, understanding the fundamentals now will help you succeed in later assignments.

ToDo:

1. Follow the instructions in `derivatives_and_graphs.ipynb` to finish this exercise.

### 4.1 Derivatives [17.5 points]

In this exercise you will learn how to calculate the gradient (derivative) of basic functions which do calculations based on Pytorch tensors.

#### 4.1.1 Derivative of a Simple Function at a Specific Position [5 points]

In this exercise you will learn how to calculate the derivative of the simple function  $f(x)$  given as:

$$f(x) = x^2$$

at position  $x = 2$ .

Remember the derivative  $f'(x)$  of the function  $f(x)$  with respect to  $x$  is calculated as:

$$f'(x) = \frac{df(x)}{dx} = (x^2)' = 2x$$

which means that  $f'(x)$  for  $x = 2$  is:

$$f'(x = 2) = 4$$

Of course, we could hard code all of that and compute our derivative. However, we would do it only for a single function at a single evaluation position. This means that we would need to code everything again for a different function or a different position we want our derivative to be evaluated at. Hence, this is not quite efficient.

But since each one of you is a highly skilled programmer and probably wants to stay one, you want to use a more convenient and efficient way to compute derivatives of different functions. And for that Pytorch will give you the answer. In the next few steps, we will learn how to use a few simple commands provided by Pytorch to achieve that goal.

ToDo:

1. Implement the method `derivative_of_simple_function_at_position` which calculates the derivative of  $f(x)$  at  $x = 2$ .

Hints:

1. Check out the `requires_grad` option for Pytorch tensors.
2. Check out the function `backward()` provided by Pytorch for tensors.

#### 4.1.2 Derivative of a More Complicated Function at a Specific Position [5 points]

Let's try what we have learned above for a more complicated function  $g(x)$ . The function  $g(x)$  is given as:

$$g(x) = 2x^3 + x^2 + x$$

And we want to calculate its derivative  $g'(x)$  at position  $x = 5$ .

ToDo:

1. Implement the method `derivative_of_more_complicated_function_at_position` which calculates the derivative  $g'(x)$  of  $g(x)$  at  $x = 5$

#### 4.1.3 Derivative of an Entire Function [7.5 points]

Having the derivative at a specific position  $x^*$  is cool, but in general we are interested in the derivative of an entire function. Calculation of the latter can also be done quite fast by using the autograd functionality of Pytorch. Let's take a look at the Rectified Linear Unit (ReLU) function which is given as:

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

We want to calculate the derivative of the entire function at every point of  $x$ .

ToDo:

1. Implement the method `derivative_of_relu` which calculates the derivative  $f'(x)$  of  $f(x)$  at every point in  $x$ .

Hints:

1. Check out the `sum()`-trick provided by Pytorch to get the derivative of a vector valued function.



## 4.2 Partial Derivatives [7.5 points]

In this exercise you will learn how to calculate partial derivatives of functions which do calculations based on Pytorch tensors. Consider the function  $f(x)$  given as:

$$f(u, v) = v * u + u^2 + 3 * v$$

We want to calculate the derivatives  $f_u$  and  $f_v$  of  $f(u, v)$  with respect to  $u$  and  $v$  respectively.

ToDo:

1. Implement the method `partial_derivatives` which calculates the derivatives  $f_u$  and  $f_v$  of  $f(u, v)$  with respect to  $u$  and  $v$  respectively.