UNIVERSITÄT
ZU KÖLN

DEEP LEARNING – SESSION 1

WiSe 2024/2025

Janis Pagel

Institut für Digital Humanities

2024-10-10

01

ORGANISATION

Course Language

- Language spoken in course: German
- Language on slides: English
- Always feel free to ask questions in English

Janis Pagel

- Background in German Studies, Linguistics, Computational Linguistics and Digital Humanities
- PhD in Computational Linguistics
- Research Interests
  - Computational Literary Studies (CLS)
    - Character Types
    - Character Knowledge
    - Usage of Large Language Models in CLS
  - Annotation, Operationalization
  - Coreference Resolution
  - Compositionality

Module

- Module “Grundlagen der Computerlinguistik”
  - Seminar “Computerlinguistische Grundlagen”
  - Lecture & Exercise “Sprachverarbeitung”
- Module “Anwendungen der Computerlinguistik”
  - **Colloquium “Deep Learning”**
  - Hauptseminar “Anwendungen der Computerlinguistik”
  - Module Examination: In Hauptseminar
- This course only offers a “Studienleistung”
- In order to get the Studienleistung
  - Submit 10 out of 14 exercises before their respective deadline
  - You can choose which of the exercises you want to submit

Course Structure

- (Discussion of previous exercise)
- Lecture/slides on new topic
- Presentation of exercise sheet
- Lab session: Working on exercise sheet, alone or in small groups
- Finish exercise until deadline by uploading on GitHub
- Solution uploaded on GitHub after deadline

Course Organization

- Website:
  `https://lehre.idh.uni-koeln.de/lehrveranstaltungen/wintersemester-2024-2025/deep-learning`
- GitHub: `https://github.com/IDH-Cologne-Deep-Learning-2024`
- Python-Server: `http://compute.spinfo.uni-koeln.de`
- Email: `janis.pagel@uni-koeln.de`

Course Overview

- Introduction to git and GitHub
- Introduction to Python
- Introduction to Deep Learning
  - Linear and logistic regression
  - Loss functions
  - Gradient descent
  - Feed-forward neural networks (FFNN)
  - Recurrent neural networks (RNN)
  - Long Short-Term Memory (LSTM)
  - Overfitting, Dropout, Regularization
  - Word Embeddings
  - Encoder-Decoder
  - Attention
  - Transformer
  - Large Language Models

02

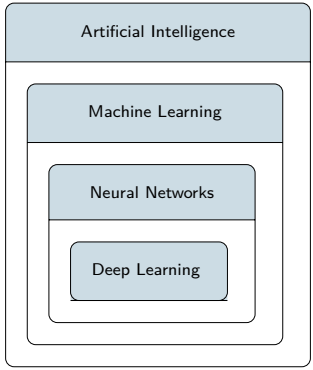
INTRODUCTION TO DEEP LEARNING

Deep Learning Definition

Definition

Deep learning is a subset of machine learning that uses multilayered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain. Some form of deep learning powers most of the artificial intelligence (AI) applications in our lives today.

Source: `https://www.ibm.com/topics/deep-learning`, last access: 2024-10-09T16:00:00



Neuronale Netze

- Mapping input data to desired output
- Stacking hidden layers that learn parameters based on input and desired output
- Classify new data using learned parameters

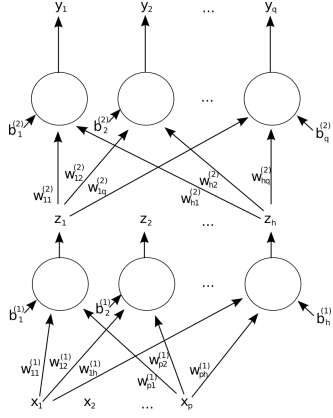


Figure: Source: `https://en.wikipedia.org/wiki/Neural_network`

Use Cases for Deep Learning

- In public applications
  - ChatGPT
  - Google Translate
  - Deepfake
  - Voice Synthecizer
  - And countless others
- In Digital Humanities / Language Processing
  - Automatic transcriptions
  - Named entity recognition
  - Coreference resolution
  - And countless others

Everything else

- Everything else to know about deep learning during the next weeks
- Reading on deep learning
  - Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd ed. Online manuscript released August 20, 2024. 2024. URL: https://web.stanford.edu/~jurafsky/slp3, chapters 5–12
  - Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing mit Transformern*. Trans. by Marcus Fraaß. Heidelberg: O'Reilly, 2023. ISBN: 978-3-96009-202-5

03

VERSION CONTROL

Version Control

- Keep track of all versions of source code
- Easily compare different versions with each other
- Work on several versions in parallel
- Work on source code in a team in parallel

Conflicts in Source Code

user-a.txt

```
1  This is the first line of code.
2  This is the second line of code.
3  Here, user A changed something.
4  This is the fourth line of code.
```

user-b.txt

```
1  This is the first line of code.
2  This is the second line of code.
3  Here, user B changed to something else from user A.
4  This is the fourth line of code.
```

remote.txt

```
1  This is the first line of code.
2  This is the second line of code.
3  This is the third line of code.
4  This is the fourth line of code.
```

- Source code located in remote repository
- People in teams working locally on same source code
- Change same line
- Merge conflict!
- Version control tools solve this problem

What to put under version control?

What to put

- Plain text files
  - Source code (Python, Java, R, C++, ...)
  - Text (Plain text, LaTeX, markdown, ...)
  - Data (CSV, XML, JSON, YAML, ...)
    - There often is no good support for very large files (i.e. multiple GB)
  - Vector graphics (SVG, ...)

What not to put

- Binary files
  - Microsoft Office documents (docx, xlsx, ...)
  - PDF files
  - Images (jpg, png, ...)
  - Compiled code (executables)
- Jupyter notebooks (ipynb)

Available Version Control Systems

- CVS (Concurrent Versions System) from 1990
- SVN (subversion) from 2000
- Mercurial from 2005
- **git** from 2005
- and many more

04

GIT

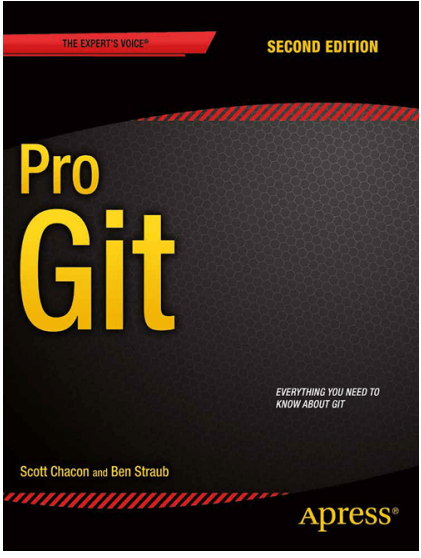
git

- Created by Linus Torvalds (Linux inventor)
- `https://git-scm.com`
- Distributed
  - Each user have their own local version of source code
  - Central repository not necessary, but often used
- Version assurance
  - Checksums for identifying revisions
- The most popular version control system (at the moment)

Hubs that support git

- Several websites that support working with git repositories
  - **GitHub**
    - Owned by Microsoft
    - Services like wikis, issue tracker, automatic integration, etc.
    - `https://github.com`
  - BitBucket
    - Owned by Atlassian
    - Also provides wiki, issue tracker, etc.
    - `https://bitbucket.org`
  - GitLab
    - Free to setup by everyone on their own server
    - Also provides wiki, issue tracker, etc.
    - `https://about.gitlab.com`
  - Many other services

Reading



- Scott Chacon and Ben Straub. *Pro Git*. 2nd ed. Apress, 2014. ISBN: 978-1484200773. URL: https://git-scm.com/book/en/v2
- Slightly outdated now
- Still good for learning basics

Ways to use git

- **Command line**
- GUI
  - GitKraken
    - Available for Windows, Mac and Linux
    - `https://gitkraken.com`
  - SourceGit
    - Available for Windows, Mac and Linux
    - `https://sourcegit-scm.github.io/`
  - Many more: `https://git-scm.com/downloads/guis`
- Code editor integration
  - Eclipse Plugin
    - `https://projects.eclipse.org/projects/technology.egit`
  - VSCode
    - `https://code.visualstudio.com/docs/sourcecontrol/overview`
  - Support available for many other editors

Command line

- Open terminal in Linux or Mac
- Open PowerShell in Windows
- Enter command in prompt
- Basic commands
  - `ls`
    - List files and folders in current directory
  - `cd <path/to/directory>`
    - Change directory

```
$ ls
Desktop MyComputer Games
$ cd Desktop/important-files
$ ls
importantfile1.txt importantfile2.txt
```

- You can usually also go to desired directory in file browser and choose option “Open in Terminal” (or similar) via right clicking

Repository

- Stores files for code project and git data files
- Go to directory in command line you want to make git repo

```
$ cd code-directory
$ ls -a
. .. file1.py file2.py
$ git init
Initialized empty Git repository in code-directory/.git/
$ ls -a
. .. .git file1.py file2.py
```

- Initially, all files are untracked

See Current Status

- Command `git status`
- Useful for seeing current situation of files and changes

```
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      file1.py
      file2.py

nothing added to commit but untracked files present (use "git add" to track)
```

Lifecycle of Files in git

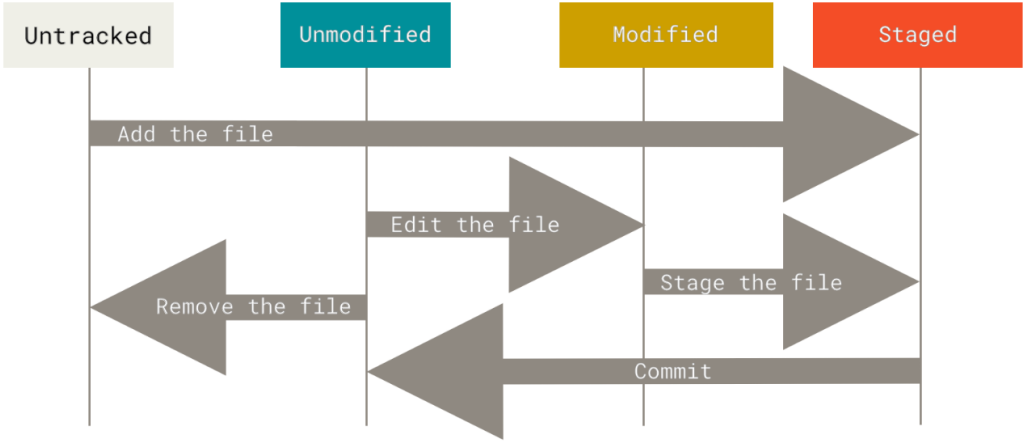


Figure: Source: Chacon and Straub [1, p. 28]

Adding files

- Add files to the “memory” of git
- git will check if changes have been made to the file

```
$ git add file1.py
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
      new file: file1.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      file2.py
```

- Use “`git add .`” or “`git add -A`” to add all files in a directory and recursively

Commit

- Commiting bundles all changes you made to tracked files under a referenceable checksum
- Needs to have commit message that describes changes

```
$ git commit -m "Add first python file"
[main (root-commit) 3d29873] Add first python file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.py
```

Log

- Git enables to see log of all commits

```
$ git log
commit 3d29873b5e32feee7fb0edaca39636dab7e309d7 (HEAD -> main)
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Wed Oct 9 23:59:20 2024 +0200

    Add first python file
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.py

nothing added to commit but untracked files present (use "git add" to track)
```

Log

```
$ git add file2.py
$ git commit -m "Add second python file"
[main 4d11534] Add second python file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.py
$ git log
commit 4d1153472b3236f2d2790a3bcf4f36301abaa253 (HEAD -> main)
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Thu Oct 10 00:05:59 2024 +0200

    Add second python file

commit 3d29873b5e32feee7fb0edaca39636dab7e309d7
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Wed Oct 9 23:59:20 2024 +0200

    Add first python file
```

Branches

- Branches are diverging versions of the same code base
- Useful to test experimental changes or to work on new features while main code stays untouched
- By default, git names the default branch “main” (used to be “master”)

```
$ git branch
* main
$ git branch newfeature
$ git branch
* main
  newfeature
$ git switch newfeature
Switched to branch 'newfeature'
$ git branch
  main
* newfeature
```

Branches

- Making changes to files in branch doesn't change files in all other branches

```
$ touch file3.py
$ ls
file1.py file2.py file3.py
$ git add file3.py
$ git commit -m "Add third python file"
[newfeature 5bcb322] Add third python file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file3.py
$ git log
commit 5bcb32292de865329b6265dfc8d31549bf15b56b (HEAD -> newfeature)
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Thu Oct 10 00:29:21 2024 +0200

    Add third python file

commit 4d1153472b3236f2d2790a3bcf4f36301abaa253 (main)
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Thu Oct 10 00:05:59 2024 +0200

    Add second python file
```

Branches

```
$ git switch main
Switched to branch 'main'
$ ls
file1.py file2.py
$ git log
commit 4d1153472b3236f2d2790a3bcf4f36301abaa253 (HEAD -> main)
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Thu Oct 10 00:05:59 2024 +0200

    Add second python file

commit 3d29873b5e32feee7fb0edaca39636dab7e309d7
Author: Janis Pagel <janis.pagel@uni-koeln.de>
Date: Wed Oct 9 23:59:20 2024 +0200

    Add first python file
```

Multiple Branches

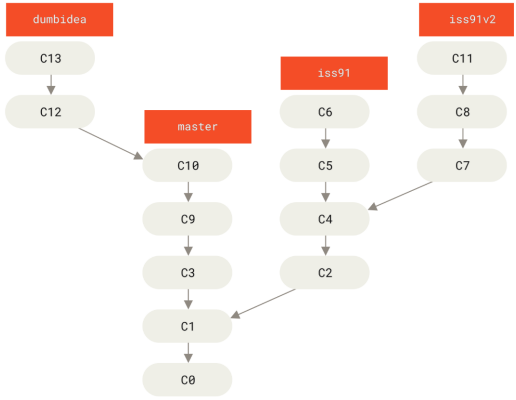


Figure: Source: Chacon and Straub [1, p. 84]

- All these branches can also be merged together again (next session)

Remote repository

- Local repos can be connected to remote repositories (e.g. on GitHub)

```
$ git remote add origin git@github.com:IDH-Cologne-Deep-Learning-2024/Exercise-1.git
$ git remote -v
origin git@github.com:IDH-Cologne-Deep-Learning-2024/Exercise-1.git (fetch)
origin git@github.com:IDH-Cologne-Deep-Learning-2024/Exercise-1.git (push)
```

- Calling the main remote repo “origin” is convention
- But any number of remote repositories under any name can be added

Push and Pull

- Push local commits to remote server via "`git push <remote-name> <branch-name>`"

```
$ git push origin main
```

- Pull changes from remote server made by some else (or by you from a different machine) via "`git pull <remote-name> <branch-name>`"

```
$ git pull origin main
```

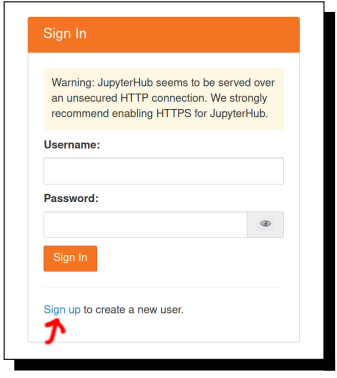
- In order to push to a remote GitHub repository, you need to create an SSH key and connect it to your GitHub account:
  - `https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent`
  - `https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account`

05

JUPYTER

Jupyter Login

- `http://compute.spinfo.uni-koeln.de`
  - This is only reachable from the University of Cologne Network
  - `https://rrzk.uni-koeln.de/internetzugang-web/netzzugang/vpn` for VPN access



- Sign up with a new username and password
- Go back to “Sign In” and sign in with your chosen username and password

06

EXERCISE 1

Exercise 1

- Every exercise can be found as a README file in the exercise repository of `https://github.com/IDH-Cologne-Deep-Learning-2024`
- The first exercise can be found in `https://github.com/IDH-Cologne-Deep-Learning-2024/Exercise-1/blob/main/README.md`
- Deadline for Exercise 1: October 17, 2024, 08:00:00 CEST
- Every future exercise also needs to be submitted via GitHub

UNIVERSITY
OF COLOGNE

Janis Pagel
Institut für Digital Humanities

eMail janis.pagel@uni-koeln.de
Website https://janispagel.de
Phone +49 221 470 5749

References

[1] Scott Chacon and Ben Straub. *Pro Git*. 2nd ed. Apress, 2014. ISBN: 978-1484200773. URL: https://git-scm.com/book/en/v2.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd ed. Online manuscript released August 20, 2024. 2024. URL: https://web.stanford.edu/~jurafsky/slp3.

[3] Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing mit Transformern*. Trans. by Marcus Fraaß. Heidelberg: O'Reilly, 2023. ISBN: 978-3-96009-202-5.