

Lecture 10

Accessible Multimodal Learning and Transfer Learning with PyKale



Haiping Lu

YouTube Playlist: <https://www.youtube.com/c/HaipingLu/playlists>

COM4059/6059: MLAI21@The University of Sheffield

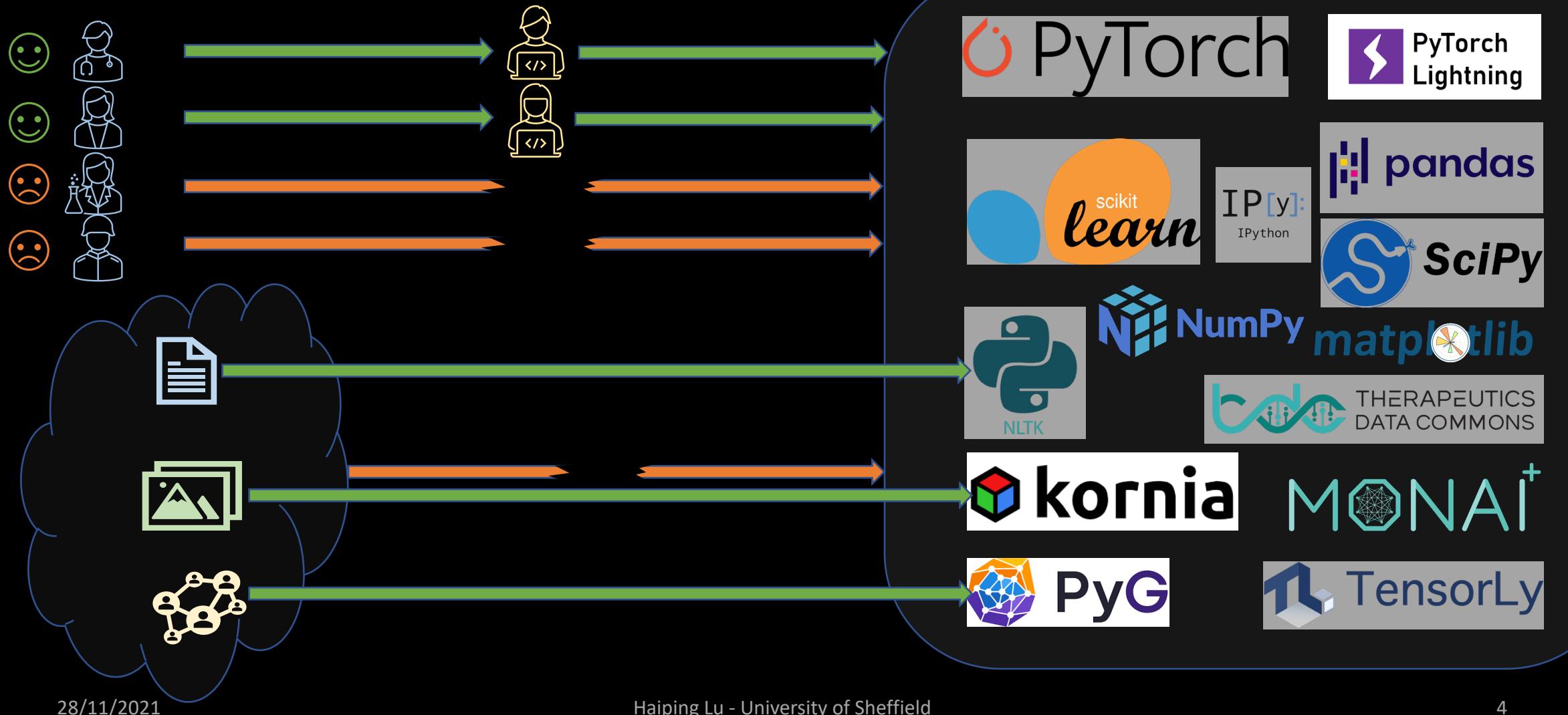
Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- What PyKale Looks Like
- Summary and Review

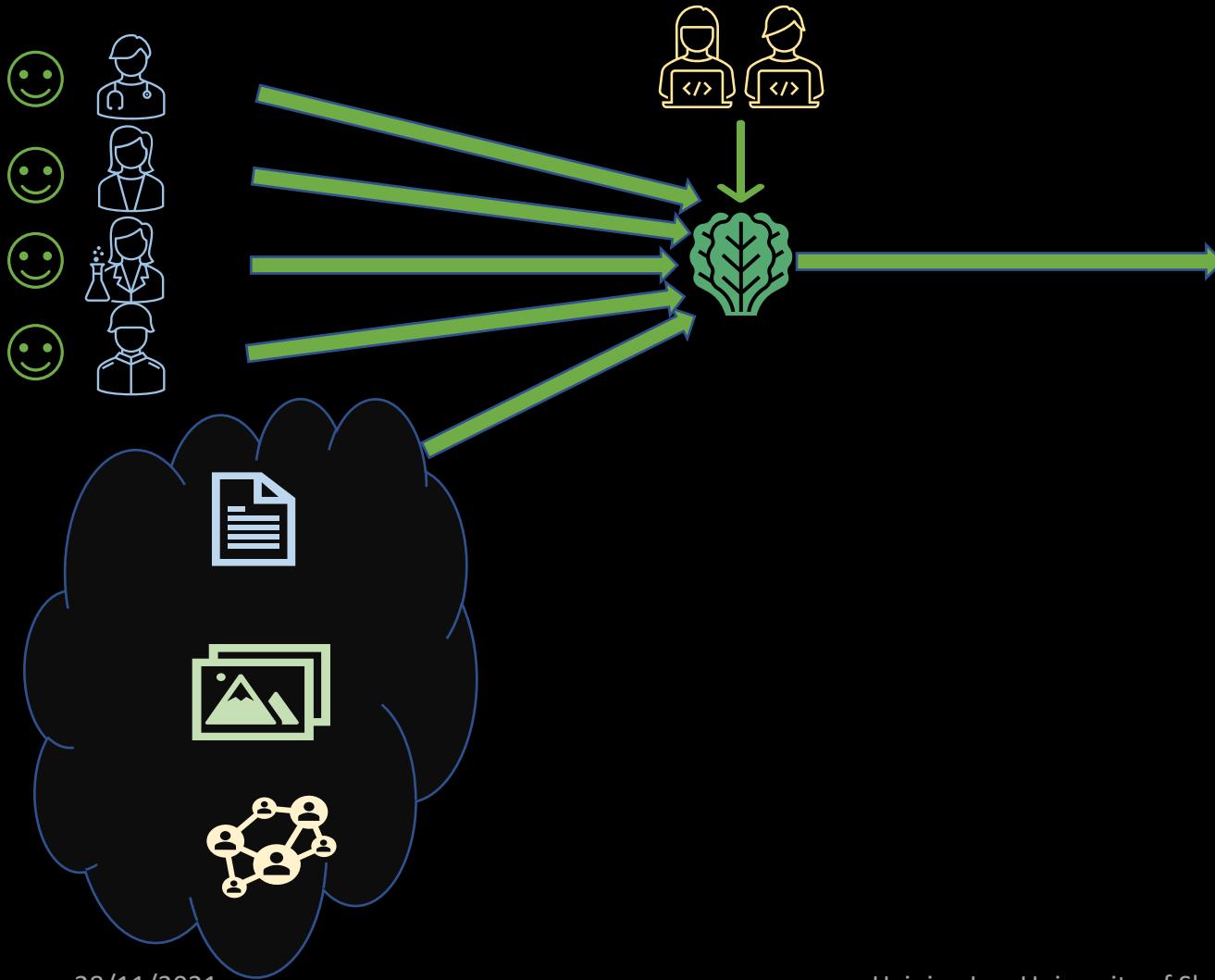
Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- What PyKale Looks Like
- Summary and Review

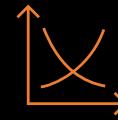
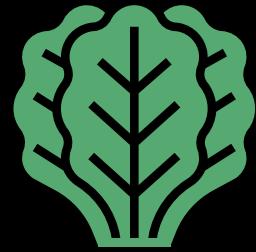
Accessibility issues in interdisciplinary research



Accessibility solution



Why build PyKale?



- Accessibility challenge in interdisciplinary research
- Abundant ML software vs unmet user demands



- Low-level ML software → focus on basics & fundamentals
- Single-domain ML software → tailor for single domains, varying APIs



- Our software → high-level, multi-source, unified API, domain-traversing
- Bridge gaps between data, software and end users for accessible, scalable, and sustainable research in machine learning

Week 10 Contents / Objectives

- Why PyKale
- **How PyKale was Built**
- What PyKale Looks Like
- Summary and Review

Is it FAIR?



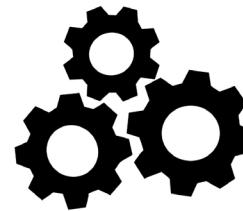
F
indable



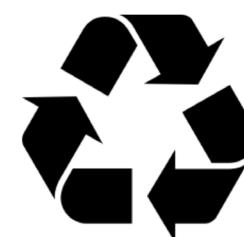
A
ccessible



I
nteroperable



R
Reusable

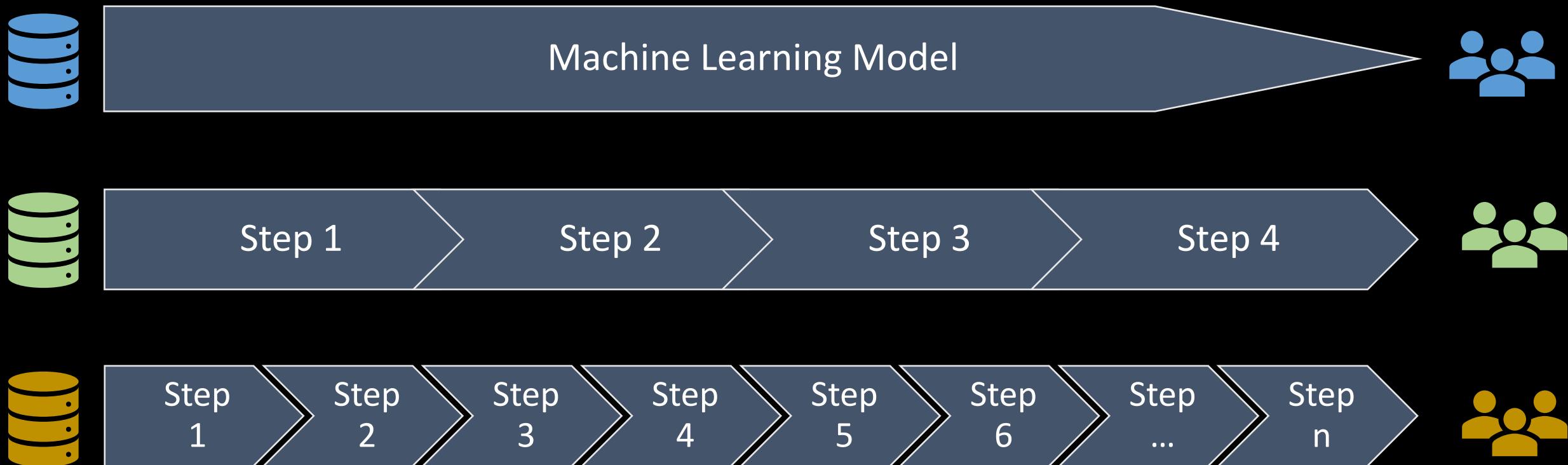


New principles: green machine learning

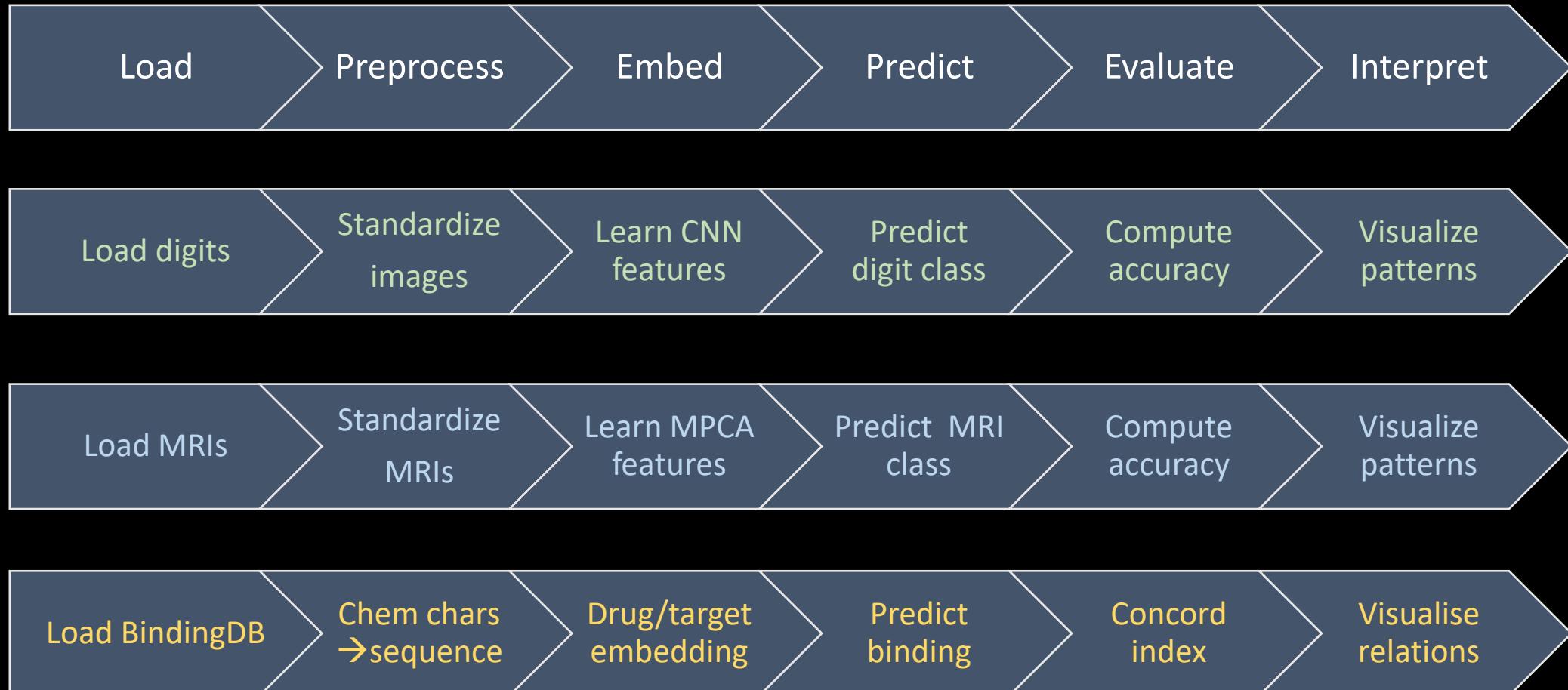
- **Reduce** repetition and redundancy
 - Refactor to standardize workflow and enforce styles
 - Identify and remove duplicated functionalities
- **Reuse** existing resources
 - Reuse the same machine learning pipeline for different data
 - Reuse existing software for available functionalities
- **Recycle** learning models across areas
 - Identify commonalities between applications
 - Recycle models for App A to App B



API without standardization 😞



Pipeline-based API



Accessible machine learning



- Bridge the gap: abundant ML software   unmet user demands



- Separate code and configuration: pykale/examples

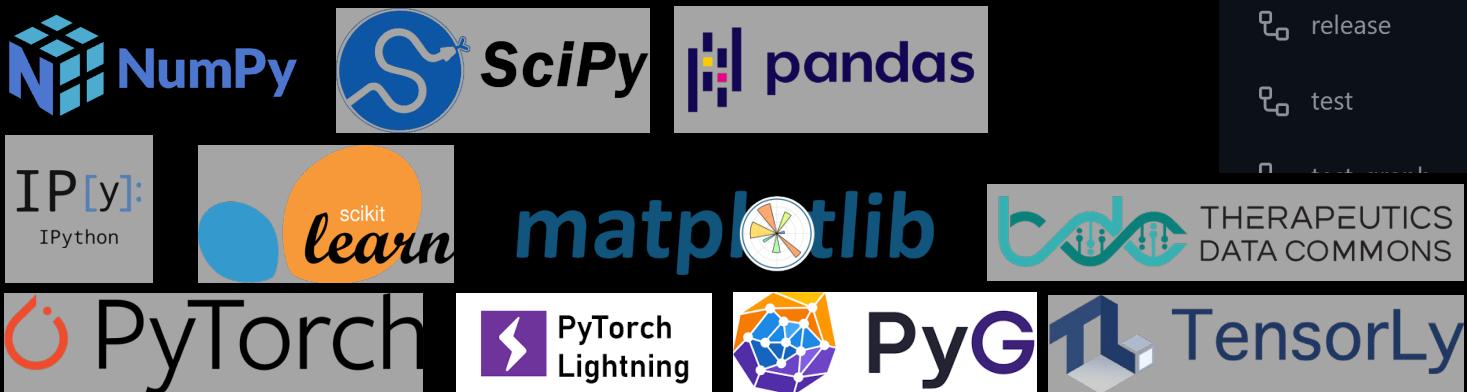


- End users (non-experts in ML)
 - Configure via YAML, no need to write Python/ML → Lower entry barriers
- ML experts
 - Manage config with min coding → Improve reproducibility/efficiency
 - Deliver accessible ML software → **Scalable** collaboration
→ Broad applications



Sustainable software development

- GitHub ecosystem → industrial standard
 - Automation & continuous integration
 - Read the Docs, PyPI release
 - Linting, Pre-commit, PyTest,Codecov
- Python/PyTorch ecosystem



A screenshot of a GitHub repository's Actions page. The top navigation bar shows Code, Issues (5), Pull requests (3), Discussions, and Actions (selected). Below the navigation is a search bar and a 'New workflow' button. A sidebar on the left lists workflows: assign project, changelog, lint, release, and test. The main area shows 'All workflows' with 2,617 runs. A specific run for 'test' is highlighted with a green checkmark and labeled 'test #840: Scheduled'. Another run for 'Multi source example' is shown with a red X and labeled 'test #839: Pull request #222 synchronize by sz144'.

Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- **What PyKale Looks Like**
- Summary and Review

Kale API snapshot: pipeline-based

The screenshot shows a terminal window with four tabs, each displaying a list of files and a context menu. The tabs are:

- `pykale / kale /` (highlighted)
- `pykale / kale / loaddata /`
- `pykale / kale / utils /`
- `pykale / kale / pipeline /`

The context menu (visible for the first tab) contains the following items:

- Load Data
- Preprocess Data
- Embed
- Predict
- Evaluate
- Interpret
- Pipeline
- Utilities

The files listed in each tab are:

- `pykale / kale /`:
 - `..`
 - `embed`
 - `evaluate`
 - `interpret`
 - `loaddata`
 - `pipeline`
 - `predict`
 - `prepdata`
 - `utils`
- `pykale / kale / loaddata /`:
 - `..`
 - `sz144 fix digits_access`
 - `..`
 - `README.md`
 - `__init__.py`
 - `cifar_access.py`
 - `dataset_access.py`
 - `digits_access.py`
 - `get_dicom.py`
 - `mnistm.py`
 - `multi_domain.py`
 - `office_access.py`
- `pykale / kale / utils /`:
 - `..`
 - `XianyuanLiu Delete csv_logger.py`
 - `..`
 - `README.md`
 - `__init__.py`
 - `download.py`
 - `logger.py`
 - `print.py`
 - `seed.py`
- `pykale / kale / pipeline /`:
 - `..`
 - `haipinglu define acronyms`
 - `..`
 - `README.md`
 - `__init__.py`
 - `deep_dti.py`
 - `domain_adapter.py`
 - `mpca_trainer.py`
 - `multi_domain_adapter.py`
 - `video_domain_adapter.py`

At the bottom center of the terminal window, it says "u - University of Sheffield".

Standardized examples & code config separation

The image displays three GitHub commit screenshots illustrating the standardization of examples and code configuration separation.

Commit 1: [haipinglu add help on how to run](#)

- ..
- configs
- README.md
- __init__.py
- config.py
- main.py
- tutorial.ipynb

Commit 2: [haipinglu add help on how to run](#)

- ..
- configs
- figures
- README.md
- __init__.py
- config.py
- main.py
- model.py
- tutorial.ipynb

Commit 3: [haipinglu improve yaml doc](#)

- ..
- configs
- README.md
- __init__.py
- config.py
- main.py
- model.py
- tutorial.ipynb

YAML configuration

```
5 lines (4 sloc) | 49 Bytes

1 DAN:
2   METHOD: "DANN"
3
4 DATASET:
5   SOURCE: "svhn"
```

```
14 lines (11 sloc) | 155 Bytes

1 DAN:
2   METHOD: "CDAN"
3
4 DATASET:
5   NUM_REPEAT: 1
6   SOURCE: "svhn"
7   VAL_SPLIT_RATIO: 0.5
8
9 SOLVER:
10  MIN_EPOCHS: 0
11  MAX_EPOCHS: 3
12
13 OUTPUT:
14  PB_FRESH: None
```

```
# -----
# Dataset
# -----
_C.DATASET = CN()
_C.DATASET.ROOT = "../data" # Path to store data and results
_C.DATASET.NAME = "digits" # Name of datasets
_C.DATASET.SOURCE = "mnist" # The source dataset name
_C.DATASET.TARGET = "usps" # The target dataset name
_C.DATASET.NUM_CLASSES = 10
_C.DATASET.NUM_REPEAT = 10
_C.DATASET.DIMENSION = 784
_C.DATASET.WEIGHT_TYPE = "natural"
_C.DATASET.SIZE_TYPE = "source"
_C.DATASET.VAL_SPLIT_RATIO = 0.1
# -----
# Solver
# -----
_C.SOLVER = CN()
_C.SOLVER.SEED = 2020
_C.SOLVER.BASE_LR = 0.001 # Initial learning rate
_C.SOLVER.MOMENTUM = 0.9
_C.SOLVER.WEIGHT_DECAY = 0.0005 # 1e-4
_C.SOLVER.NESTEROV = True

_C.SOLVER.TYPE = "SGD"
_C.SOLVER.MAX_EPOCHS = 120 # "nb_adapt_epochs": 100,
# _C.SOLVER.WARMUP = True
_C.SOLVER.MIN_EPOCHS = 20 # "nb_init_epochs": 20,
_C.SOLVER.TRAIN_BATCH_SIZE = 150 # 150
_C.SOLVER.TEST_BATCH_SIZE = 200 # No difference in ADA
```

Community engagement

A screenshot of the PyKale GitHub repository page. The top bar shows the repository name 'PyKale' and the branch 'latest'. Below the header is a search bar labeled 'Search docs'. The main content area displays the 'GETTING STARTED' section, which includes links to 'Introduction', 'Installation', and 'Tutorial'. At the bottom of this section is a video player showing a YouTube video titled 'PyKale' with a thumbnail of a person speaking.

A screenshot of the PyKale Documentation website. The header shows the home icon and the text '» PyKale Documentation'. The main title is 'PyKale Documentation' and the sub-section is 'Getting Started'. Below the sub-section title is a bulleted list: '• Introduction' and '• Installation'.

A screenshot of the PyKale arXiv preprint page. The title is 'arXiv.org > cs > arXiv:2106.09756'. The main title of the paper is 'Computer Science > Machine Learning'. The abstract starts with 'PyKale: Knowledge-Aware Machine Learning from Multiple Sources in Python'. The authors listed are Haiping Lu, Xianyuan Liu, Robert Turner, Peizhen Bai, Raivo E Koot, Shuo Zhou, Mustafa Chasmai, Lawrence Schobs.

A screenshot of the PyKale GitHub repository. The top bar shows the URL 'https://github.com/pykale/pykale/blob/main/.github/CONTRIBUTING.md'. The page content includes sections for 'Light involvements (viewers/users)', 'Medium involvements (contributors)', and 'Heavy involvements (maintainers)'. It lists various ways to contribute: Ask questions, Report bugs, Suggest improvements, Branch, fork & pull, Coding style, Test, Review & merge, and Release & management. Below this is a GitHub project board for version v0.1.0. The board has three columns: 'Overview & backlog', 'To do', and 'In progress'. It contains cards for tasks like 'Added by haipinglu', 'v0.2.0', 'Roadmap: 0.1.0a1-->0.1.0b1-->0.1.0b3-->0.1.0rc1-->0.1.0rc3-->v0.1.0 to build', 'Remove dependency on csv logger and maybe remove csv logger.', 'Make pykale.github.io following https://github.com/Project-MONAI/PyKale', and 'DA for action recognition TA3N'. The bottom of the page shows a list of discussions, with the first few being 'Unique Selling Points', 'Should we have tests for tutorial notebooks?', and 'Where should we publish on PyKale?'. The GitHub interface includes standard navigation bars for code, issues, pull requests, discussions, actions, projects, security, and insights.

The screenshot shows a browser window with the URL <https://pykale.readthedocs.io/en/latest/notebooks.html>. The left sidebar is titled "PyKale latest" and includes a "Search docs" bar and sections for "GETTING STARTED" (Introduction, Installation, Tutorial) and "Jupyter Notebook Tutorials". The main content area displays "Jupyter Notebook Tutorials" with three bullet points: "Image classification: Digits domain adaptation", "Drug discovery: BindingDB drug-target interaction prediction", and "Medical image analysis: Cardiac MRI for MPCA-based diagnosis". Below this, a Colab notebook titled "tutorial.ipynb" is shown, featuring a toolbar with File, Edit, View, Insert, Runtime, Tools, Help, and buttons for + Code, + Text, and Copy to Drive. The notebook content starts with a section titled "PyKale Tutorial: Domain Adaptation on Digits with Lightning" with links to Open in Colab and Launch Binder, and a note about using Google Colab for GPU support.

PyKale latest

Search docs

GETTING STARTED

Introduction
Installation
Tutorial

Jupyter Notebook Tutorials

» Jupyter Notebook Tutorials

Jupyter Notebook Tutorials

- Image classification: Digits domain adaptation
- Drug discovery: BindingDB drug-target interaction prediction
- Medical image analysis: Cardiac MRI for MPCA-based diagnosis

tutorial.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

PyKale Tutorial: Domain Adaptation on Digits with Lightning

| [Open in Colab](#) (click Runtime → Run all (Ctrl+F9)) | [Launch Binder](#) (click Run → Run All Cells) |

If using [Google Colab](#), a free GPU can be enabled to save time via setting Runtime → Change runtime type

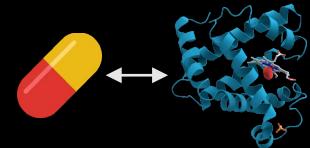
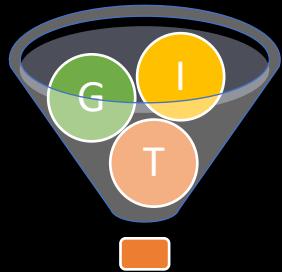
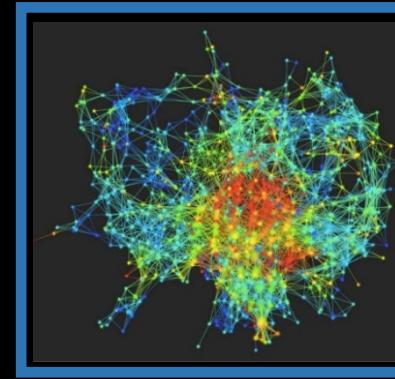
The screenshot shows a Jupyter Notebook interface. The URL in the browser is https://hub.gke2.mybinder.org/user/pykale-pykale-77qiut67/doc/tree/examples/digits_dann_lightn/tutorial.ipynb. The title of the notebook is "tutorial.ipynb". The menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Print, as well as Download, GitHub, Binder, and Markdown. On the right, it says "Python 3 (ipykernel)". The main content area displays the title "PyKale Tutorial: Domain Adaptation on Digits with Lightning" in large bold text. Below the title are links to "Open in Colab" and "Launch Binder". A note at the bottom states: "If using Google Colab, a free GPU can be enabled to save time via setting Runtime → Change runtime type → Hardware accelerator: GPU".



[livedemo-compressor.jpg \(1450×960\) \(posterno.com\)](#)

Data, model, & app

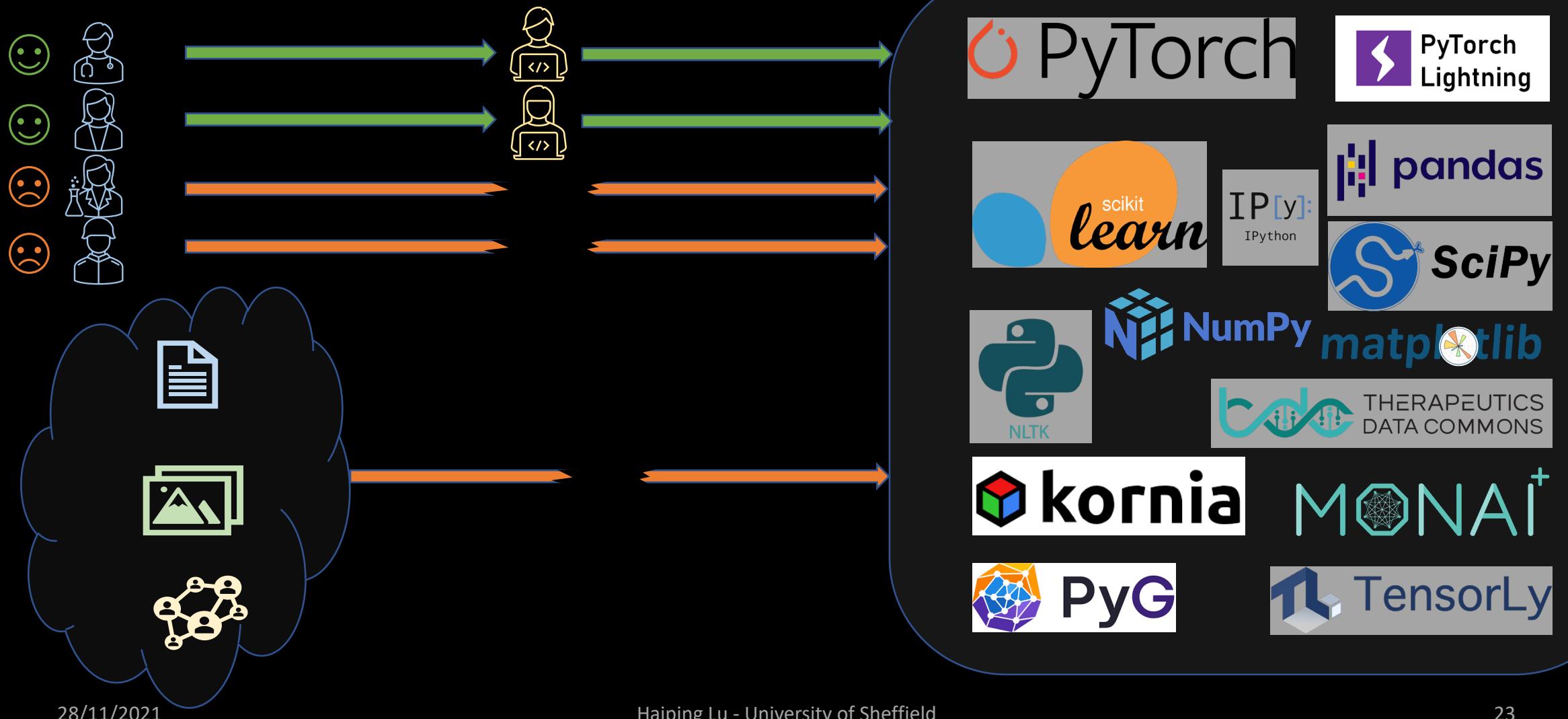
- Data: graph, image, text, video
- Machine learning models
 - Transfer learning: domain adaptation
 - Multimodal learning: integration of heterogeneous data
 - Deep learning: CNN, GNN/GCN, transformers
 - Dimensionality reduction: tensor-based
- Example applications
 - Image/video recognition: CIFAR, digits, office, action videos
 - Bioinformatics/graph analysis: BindingDB, knowledge graphs
 - Medical imaging: cardiac MRI, brain fMRI



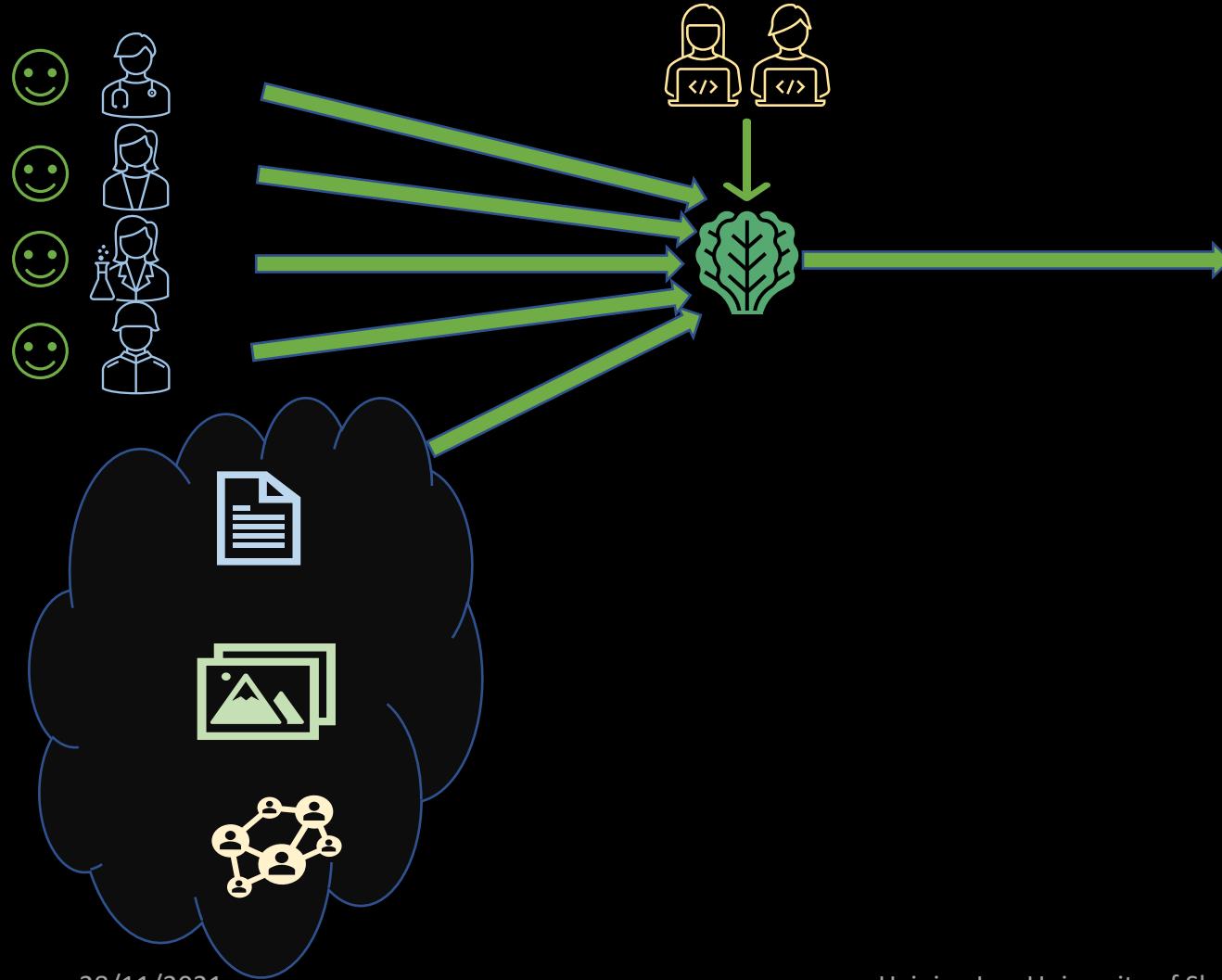
Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- What PyKale Looks Like
- **Summary and Review**

Accessibility issues in interdisciplinary research



PyKale fills the gaps



PyKale: accessible, scalable, sustainable → →

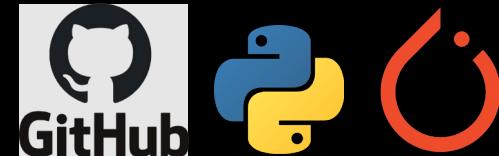
- **Green ML principles:** deliver FAIR ML software



- Pipeline-based API to unify workflow



- Industrial software engineering practices for **sustainability**

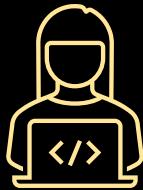


- **Accessible** and **scalable** with code-configuration separation

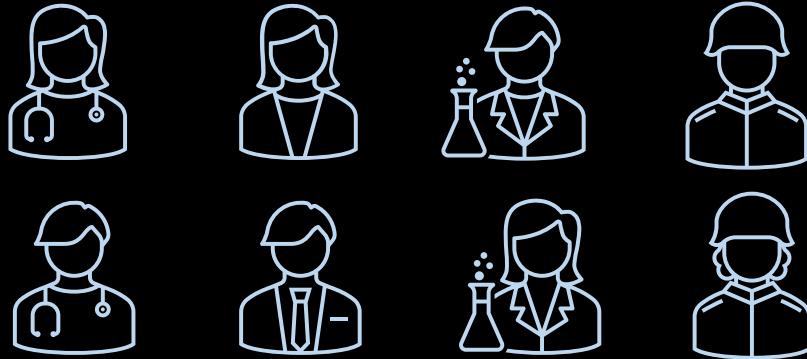


- Multimodal learning & transfer learning for graphs, images, videos, ...

Targeted audience



ML experts



End users

- Reproducible research
- Scalable collaboration
- Cross-boundary models

- Off-the-shelf ML pipelines
- Friendly configuration editing
- Multi modalities under one roof

Open with strong community engagement

Open & welcome

- Open source on GitHub (MIT License)
 - <https://github.com/pykale/pykale>
- Welcome feedback/contribution!

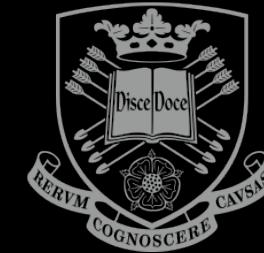
A screenshot of a web browser window. The address bar shows the URL <https://pytorch.org/ecosystem/>. The page content includes the PyTorch logo and the PyKale logo. A descriptive text block states: "PyKale is a PyTorch library for multimodal learning and transfer learning with deep learning dimensionality reduction on graphs, images, texts, and videos."



Acknowledgement



- Shef groups + community



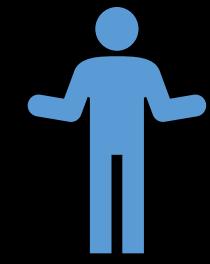
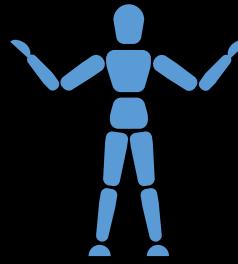
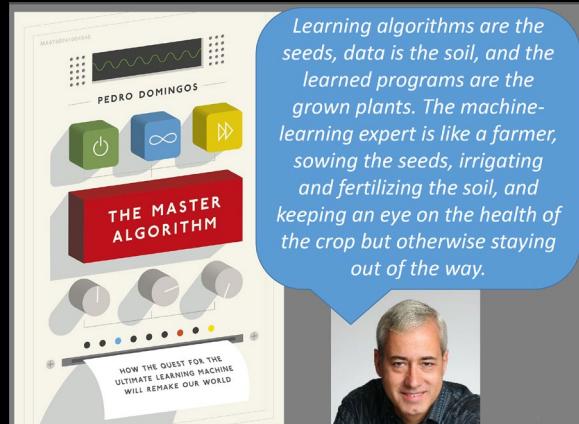
The
University
Of
Sheffield.



- Partially funded by Wellcome Trust via an Innovator Award

Review and Reflect on Machine Learning

- Training: model as well as people
 - You are a **model**
- Reflection and analogy
 - What you want to learn (objective function / goal)
 - How we teach you to learn (training / optimisation)
 - What you have learned (decision model)
- Key challenge: overfitting vs **generalisation**
- Post your questions
- Have fun



[kisspng-fashion-show-computer-icons-man-icon-5acd9cfaa3d9f5.3374529915234245066712.jpg
\(900x900\) \(cleanpng.com\)](https://kisspng.com/fashion-show-computer-icons-man-icon-5acd9cfaa3d9f5.3374529915234245066712.jpg)

Data, Model, Metric, Optimisation

- | | |
|--|--|
| 1. Given training data:
$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ | 3. Define goal:
– Objective function
$\theta^* = \arg \min_{\theta} \sum_{i=1}^N \ell(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$ |
| 2. Choose each of these:
– Decision function/model
$\hat{\mathbf{y}} = f_{\theta}(\mathbf{x}_i)$ | 4. Train/optimize with SGD: (take small steps opposite the gradient)
$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$ |