

**<https://tinyurl.com/mapreader-richmond>**

# Introduction to MapReader

Katie McDonough

w/materials developed by Rosie Wood,  
Daniel Wilson, Kaspar Beelen & Kalle  
Westerling

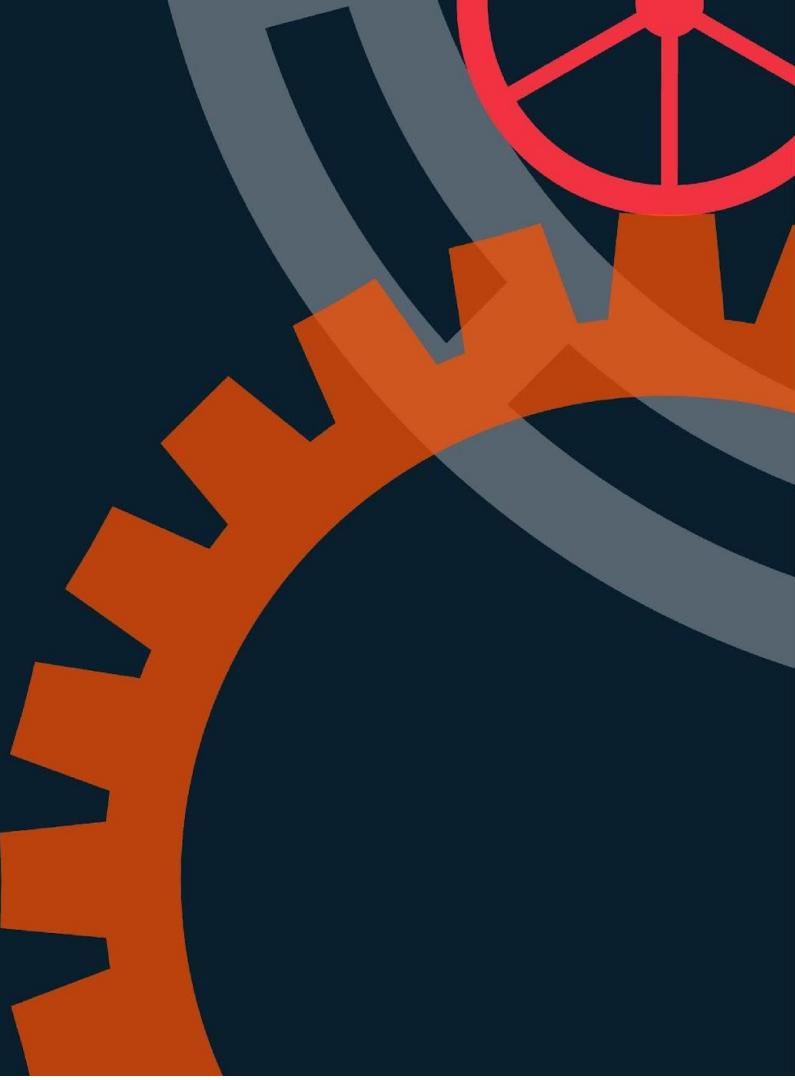
## Our Partners



## Our Funders



# Why MapReader?



1888

# How do researchers work with maps?



**Digitized maps can be more than sheets to browse in a virtual reading room. *But how?***

**Ordnance Survey**  
maps of England,  
Wales, and  
Scotland

6 inches to 1 mile  
1888-1913  
(2nd edition)

**~15K sheets**

# **Make trustworthy claims based on thousands of maps**

**case studies** → ‘high resolution’ local archival research or anecdotes from printed materials

**aggregated statistics** → ‘low resolution’ regional/national

**Ordnance Survey (series) maps as non-aggregated, high-resolution, national-coverage sources**

# You Cannot Ground Truth the Past

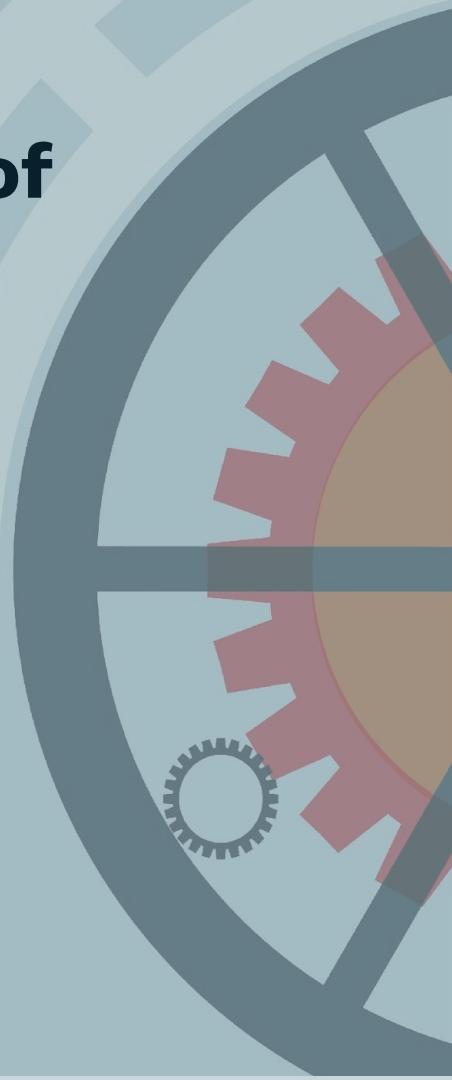
Historical OS maps tell us how Victorians represented the British landscape, and how that landscape was changing, but they are not a ‘ground truth’.

We want to ***use CV to advance interpretation, not define truth.***

# **Step away from the GIS paradigm of vectorizing everything on a map**

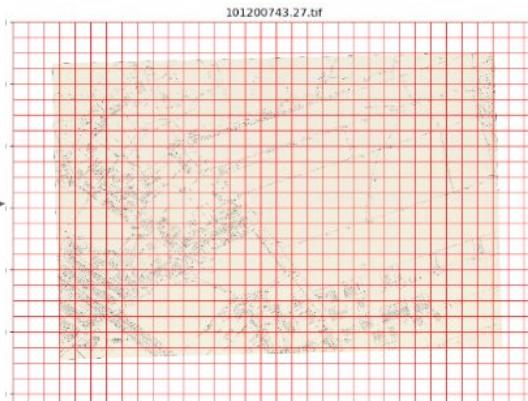
**Stop** historical **data “mining”/“extraction”**  
and insisting on overly-precise data

**Start** developing methods that permit  
**critical distance between scholars and sources** & question-appropriate data  
collection and exploration

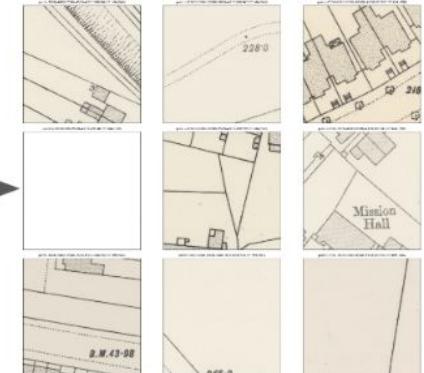


# Solution: ‘Patches’ as a new shape for historical research

Parent image



Patches



# MapReader in 2025

MapReader performs 2 tasks:

1. Classify map regions (patches) with concepts that have a **visual** signal
2. Find and transcribe **text**

[github.com/maps-as-data/  
MapReader](https://github.com/maps-as-data/MapReader)

The screenshot shows the GitHub repository page for 'maps-as-data / MapReader'. The main area displays a list of commits, showing frequent updates to files like '.github', 'conda', 'docs', 'mapreader', 'paper', 'tests', and 'worked\_examples'. The sidebar on the right provides details about the repository, including its purpose ('A computer vision pipeline for exploring and analyzing images at scale'), its tags ('machine-learning', 'computer-vision', 'deep-learning', 'article', 'maps', 'pytorch', 'digital-humanities', 'spatial-data', 'hut23', 'hut23-96'), and its activity metrics (97 stars, 7 watching, 13 forks). The bottom section shows the repository's status: 15 branches, 37 tags, and various badges for build status (e.g., Jupyter Notebook 99.3%, codecov 79%), and deployment information (github-pages 16 hours ago, + 324 deployments).

# Image Classification for Map Patches

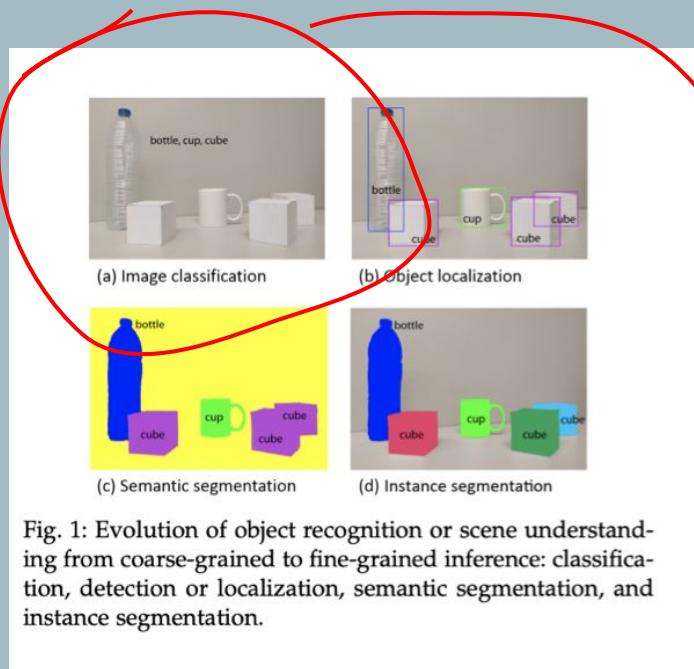
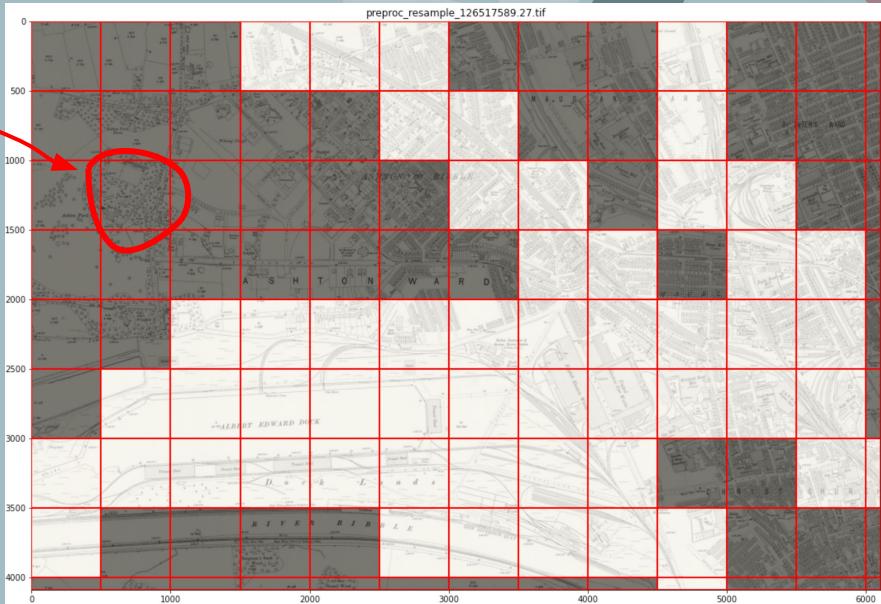
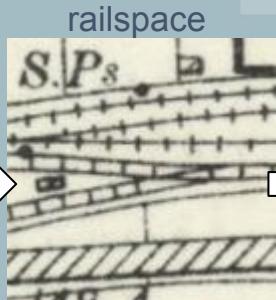
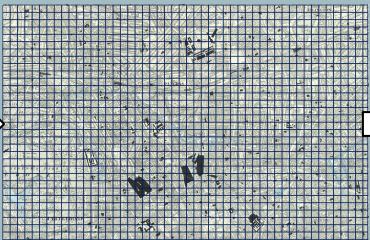
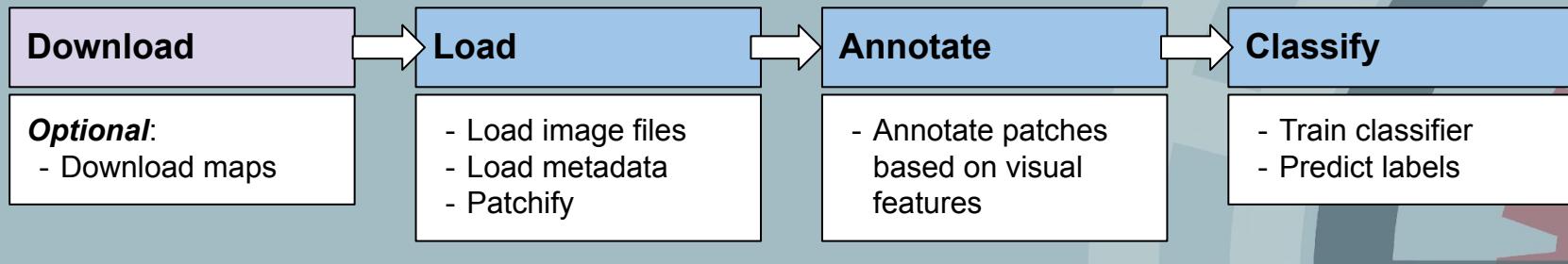


Fig. 1: Evolution of object recognition or scene understanding from coarse-grained to fine-grained inference: classification, detection or localization, semantic segmentation, and instance segmentation.

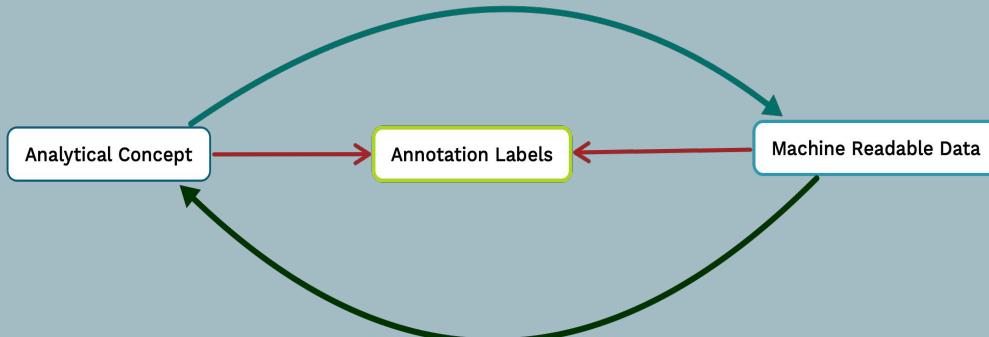


Map patches annotated as training data

# Classification pipeline



# Annotating patches: What is a good label?



Rail Space <sup>1</sup> No Rail Space <sup>2</sup> ← back <sup>j</sup> → next <sup>k</sup>

<Figure size 432x288 with 0 Axes>

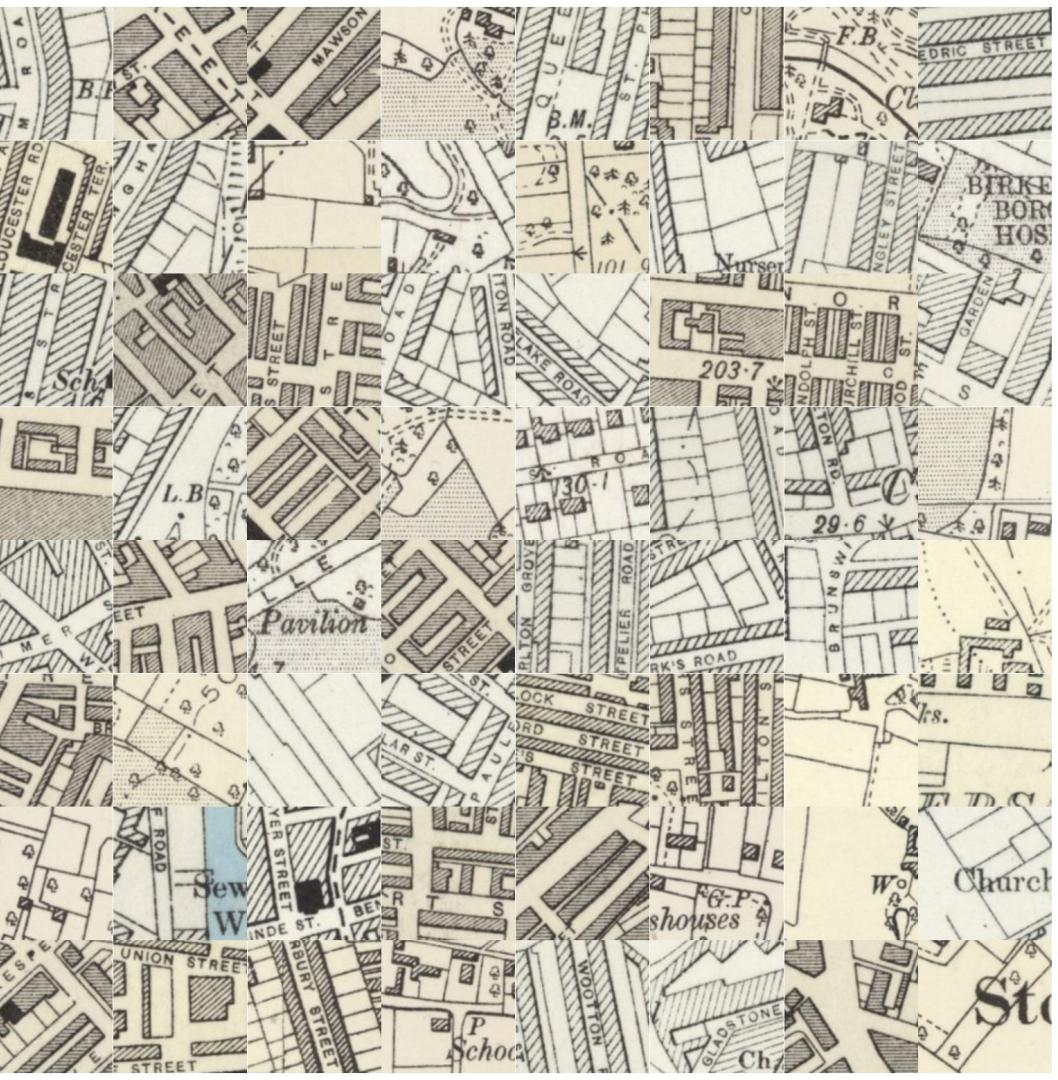
tile-4500-2000-5000-2500-#preproc\_resample\_126517589.27.tif#.PNG

The screenshot shows a digital map interface. At the top, there are two buttons: "Rail Space <sup>1</sup>" (green) and "No Rail Space <sup>2</sup>" (blue). Below the buttons are navigation links: "← back <sup>j</sup>" and "→ next <sup>k</sup>". A status bar at the bottom indicates "<Figure size 432x288 with 0 Axes>" and the file name "tile-4500-2000-5000-2500-#preproc\_resample\_126517589.27.tif#.PNG". The main area displays two versions of a map patch. The top version is labeled "Rail Space" and shows a map with several tracks and buildings, including "B.M. 881", "PRIORITY", "ABBREV.", and "INGOT ST.". The bottom version is labeled "No Rail Space" and shows the same map without the tracks. A red rectangular box highlights a specific area in the "No Rail Space" version. Below the maps, the text "Additional info:" is followed by a URL: "URL to the NLS map: <https://maps.nls.uk/view/126517589>".

# Railspace



# Buildings





## Living with Machines

*Computational Histories of the Age of Industry*

by Ruth Ahnert, Emma Griffin, Jon Lawrence,  
The Living with Machines Team

*Living With Machines* is a data-driven history of the coming of the machine age in Britain in the long nineteenth century. Featuring an innovative open access edition enhanced with interactive maps, datasets and visualisations, digital notebooks, video, audio and images, this book harnesses the combined power of massive digitised historical collections and computational analytical tools to examine the ways in which technology altered the very fabric of human existence on a hitherto unprecedented scale.

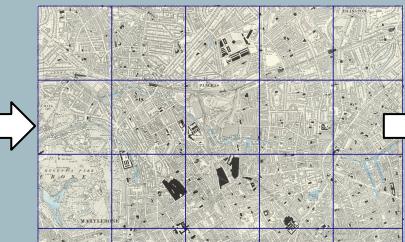
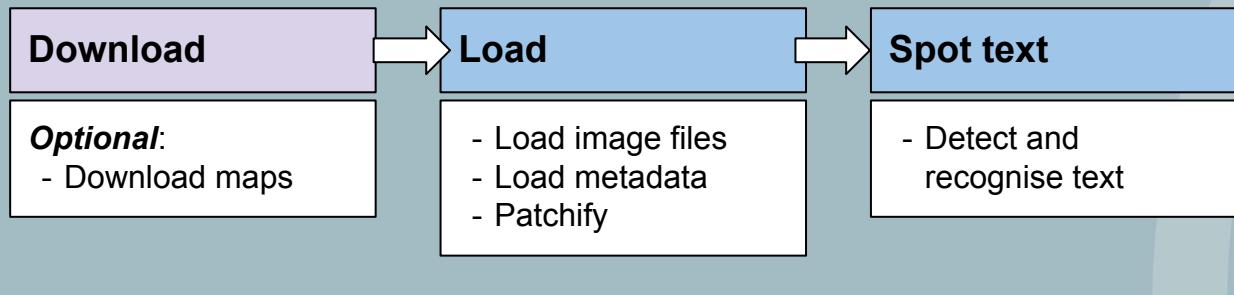


ABOUT DIGITAL EDITIONS

**Beyond the Tracks: re-connecting people, places and stations in the history of late-Victorian railways**

Joshua Rhodes, Jon Lawrence, Kaspar Beelen, Katherine McDonough, Daniel C.S. Wilson

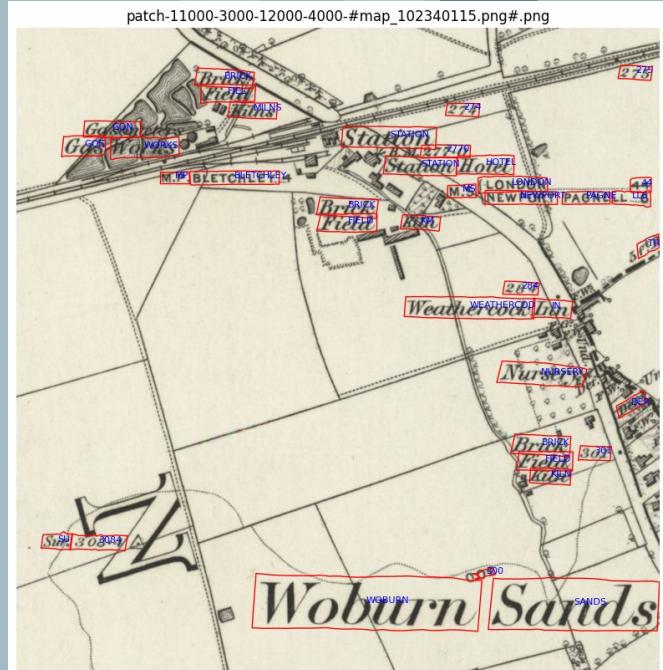
# Text spotting pipeline



# Text spotting frameworks

Three different options in MapReader:

1. **DPText-DETR** (detection only)
2. **DeepSolo** (detection + recognition)
3. **MapTextPipeline** (detection + recognition)
4. Coming soon - ICDAR MapText 2025 models!



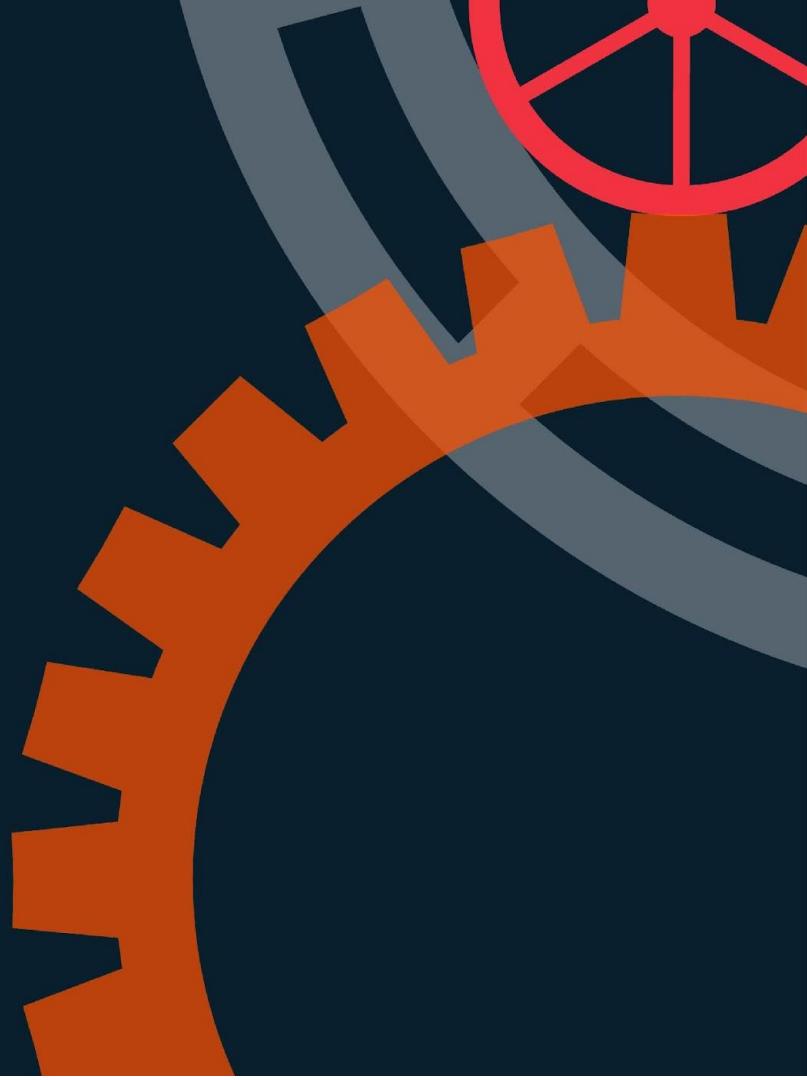
# MapReader

MapReader is a sandpit for thinking with map content and metadata to critically create structured data for onward linking and analysis.

We privilege software design and methods that:

- Illuminate understudied cartographic elements, like text
- Encourage interpretation over extraction, with patches
- Foster stronger connections with collections & curators

# Using MapReader



# <https://github.com/maps-as-data/mapreader-workshop>

maps-as-data / mapreader-workshop

Type  to search

+

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

 **mapreader-workshop** Public  
forked from [maps-as-data/workshop-dh2025](#)

[Edit Pins](#) [Watch 0](#) [Fork 0](#) [Star 0](#)

[main](#) [1 Branch](#) [0 Tags](#) [Go to file](#) [Add file](#) [Code](#)

This branch is 5 commits ahead of [maps-as-data/workshop-dh2025:main](#).

[Contribute](#) [Sync fork](#)

 **kmcdono2** Update README link for environment setup instructions [fb845a3 · 29 minutes ago](#) [93 Commits](#)

 **MapTextPipeline** combined commit [2 months ago](#)

 **images** update open notebook image [2 months ago](#)

 **maps\_nls** separate the maps directories [2 months ago](#)

**About**

Introductory MapReader Workshop

[Readme](#) [Activity](#) [Custom properties](#) [0 stars](#) [0 watching](#) [0 forks](#)

[Report repository](#)

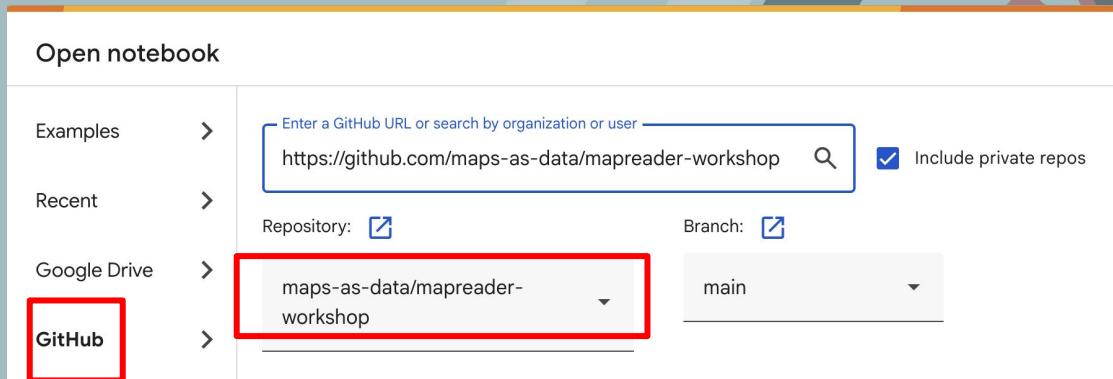
# Worked Examples for Today

1. **Patch Classification with georeferenced tile layers of Ordnance Survey maps** from the National Library of Scotland
2. *Patch Classification with maps available as IIIF resources*
3. **Text Spotting with IIIF resources** from Boston Public Library

# Setting Up

**Follow the  
instructions for  
Google Colab in the  
README**

- Step 1: select ‘Github’**
- Step 2: enter repo URL**
- Step 3: select repo**
- Step 4: open  
mapreader\_classification  
ipynb file**



# Jupyter Notebook Basics

- Two different cell types - **Code or Text**
- Running cells - Press “►” or  
“⌘/Ctrl/Shift+Enter”

**WARNING: Google Colab will time out after a while  
so make sure you save your work!**

# Open notebook & run first code cell

## ▼ MapReader Workshop @ ADHO DH 2025

### Image Classification Worked Example with National Library of Scotland Ordnance Survey Maps

Written by Rosie Wood and Katherine McDonough. Reviewed and tested by Kalle Westerling, Kaspar Beelen, and Daniel Wilson.

Learn more about the MapReader team at <https://github.com/maps-as-data/MapReader?tab=readme-overfile#contributors>.

This notebook demonstrates how to use the MapReader pipeline to

- load maps from a XYZ tileserver
- create 'patches' from selected maps
- annotate patches with user-defined labels
- fine-tune an image classification model
- infer the labels on unseen patches
- export the results

Let's get started!

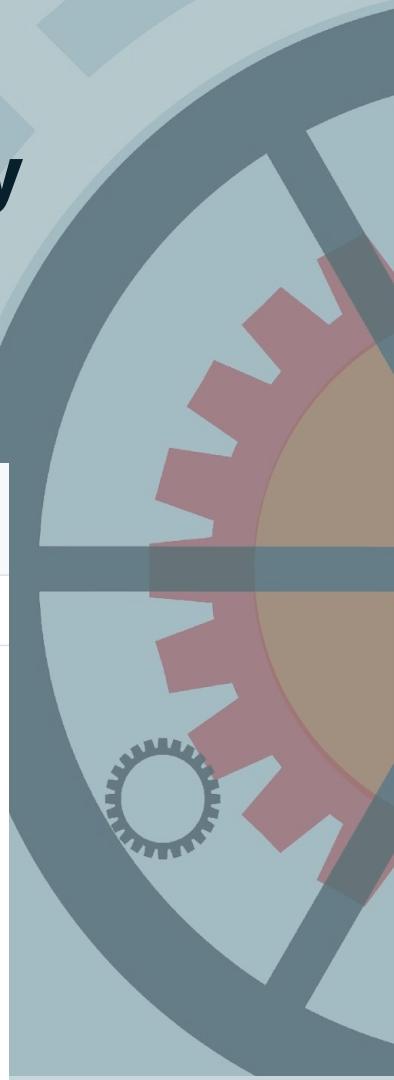
NB: documentation is available at <https://mapreader.readthedocs.io/en/latest/>

```
[ ] # set up for google colab - this cell will take a while to run!
!git clone https://github.com/maps-as-data/workshop-dh2025
!pip install mapreader[dev]
```



# MapReader (Main) Github Repository

**<https://github.com/maps-as-data/MapReader>**



Screenshot of the MapReader (Main) Github Repository page.

The page shows the following navigation and search bar:

- Code (selected)
- Issues 85
- Pull requests 5
- Discussions
- Actions
- Projects 3
- Wiki
- Security
- Insights
- Settings

The repository details are as follows:

- MapReader (Public)
- 15 Branches
- 37 Tags
- Go to file
- Edit Pins
- Unwatch 8
- Fork 15
- Starred 97

The commit history table includes the following entries:

Author	Commit Message	Date
kmcdono2	add modelcard template	4 hours ago
.github	Update publish-to-conda-forge.yml	last month
conda	v. minor tidy-up of conda specification	2 years ago
docs	Merge branch 'main' into text_dev	last month
mapreader	fix explore methods	last month
paper	Update paper.md	4 months ago
tests	Merge branch 'main' into text_dev	last month

The About section contains the following information:

- A computer vision pipeline for exploring and analyzing images at scale
- [mapreader.readthedocs.io/en/latest/](https://mapreader.readthedocs.io/en/latest/)
- Tags: machine-learning, computer-vision, deep-learning, article, maps, pytorch, digital-humanities, spatial-data, hut23, hut23-96

Links at the bottom include:

- Readme
- View license
- Code of conduct

# MapReader Documentation

<https://mapreader.readthedocs.io/en/latest/>

The screenshot shows the MapReader documentation website. At the top left is the MapReader logo and the word "latest". Below it is a search bar with the placeholder "Search docs". On the left side, there's a sidebar with a dark background containing links: "Introduction to MapReader", "Getting Started", "Using MapReader", "In-Depth Resources", and "Community and contributions". A small rectangular box in the sidebar contains an "ON-DEMAND WEBINAR" section with a play button icon, the text "How to reduce your organization's reliance on 'bad' open source packages.", and two buttons: "WATCH NOW" and "TIDELIFT". Below this is the text "On-demand webinar: Reduce your organization's reliance on 'bad' open source packages. WATCH NOW!". At the bottom of the sidebar, it says "Ad by EthicalAds" followed by a small icon. The main content area has a light background. It starts with a header "MapReader" and a sub-header "What is MapReader?". Below this is a paragraph: "MapReader is an end-to-end computer vision (CV) pipeline for exploring and analyzing images at scale." To the right of the text is a map of a city area with red and purple overlays. At the bottom, there's a section titled "Overview" with the text: "MapReader is a groundbreaking interdisciplinary tool that emerged from a specific set of geospatial". Above the main content area, there's a "Edit on GitHub" link.

# Patch Classification Notebook Sections

***No need to download maps, we have provided them!***

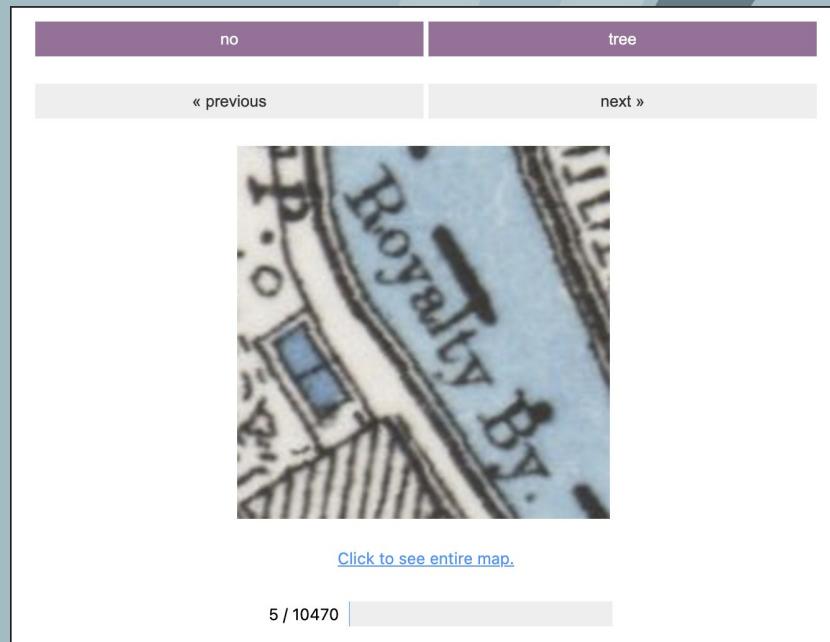
- **Install dependencies**
- **Load & Patchify maps**
- **Annotate patches with your label of choice**
- **Fine-tune a model**
- **Review model performance**
- **Infer labels on remaining patches**
- **Export labeled patch data**

# Annotations - before we begin

**Only one label per patch**

**Two approaches:**

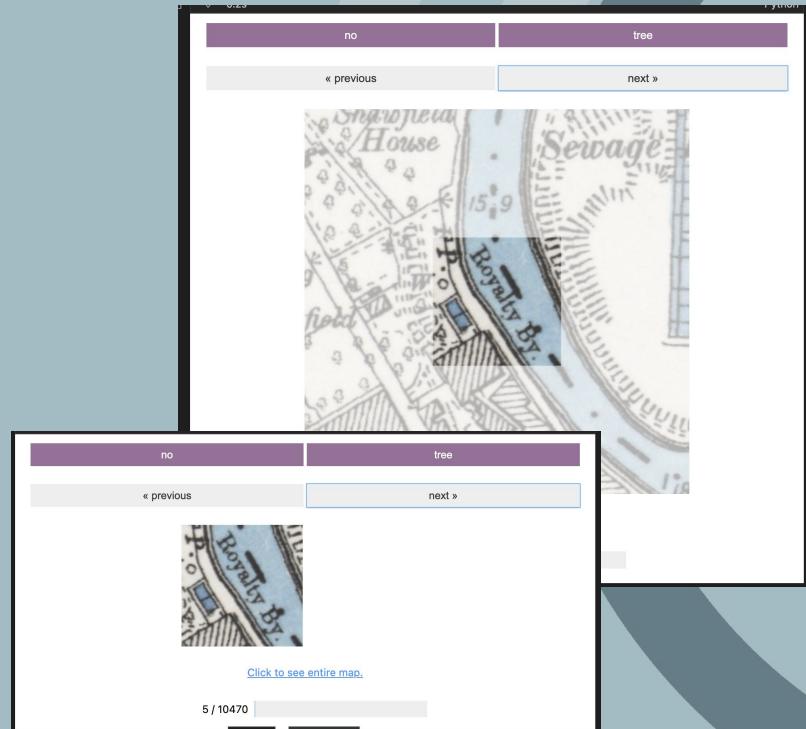
- 1. yes/no (for 1 concept)**
- 2. concept 1/concept n/neither concept**



# Annotations - before we begin

Context or no context?

- Model doesn't "see" context
- If we can't see our feature in the patch, how will the model see it?



# Labels suggestions for today

- Railways
- Water
- Footpaths (double dotted lines)
- Trees/forest
- Buildings
- Remember to have a “None” or “Not present” label
- **Anything you like...**

# Patch sizes and label choices

E.g.



1000 pixel  
Too big



200 pixel  
Woods

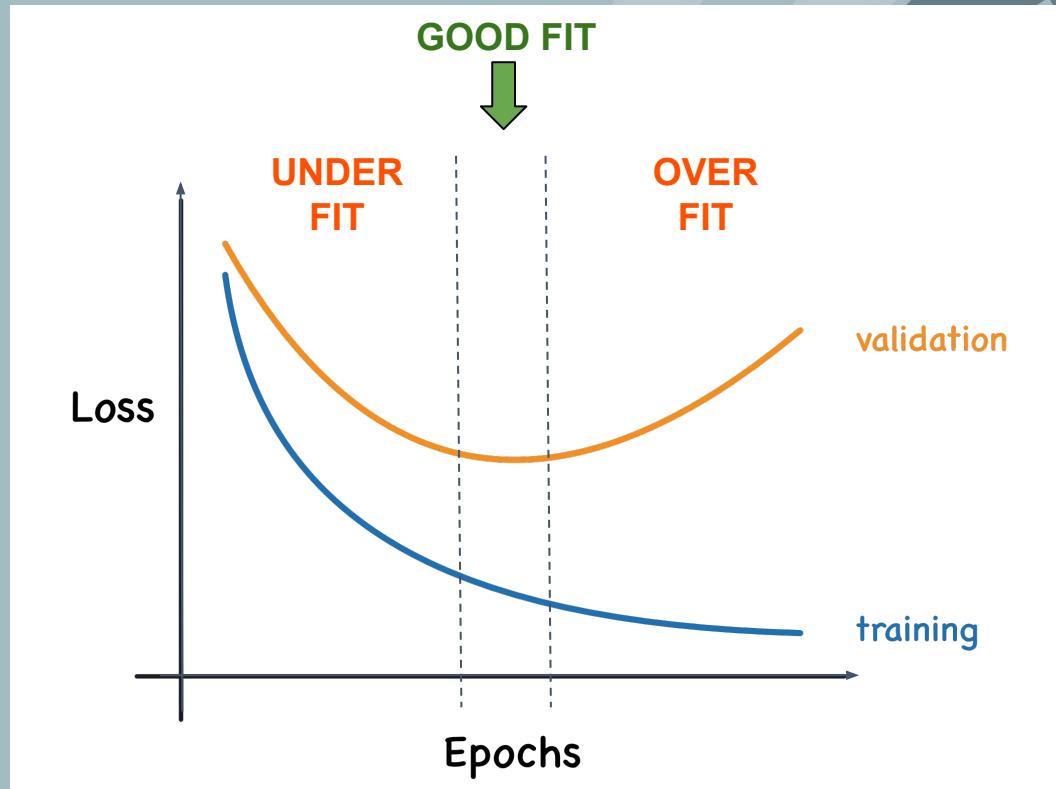


100 pixel  
Trees

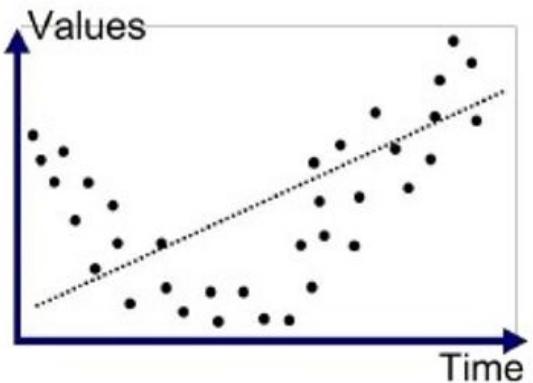
# Training

Annotations split into train, validation and test datasets

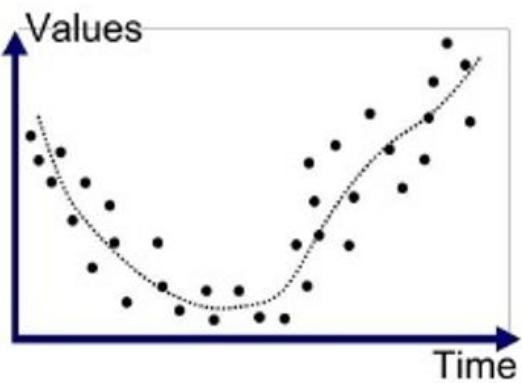
Goal is to minimise both training and validation losses



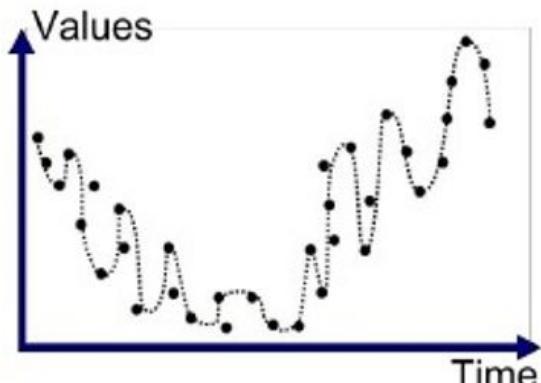
# Training



Underfitted



Good Fit/Robust



Overfitted

# More on training CV models

Try these *Programming Historian* Lessons:

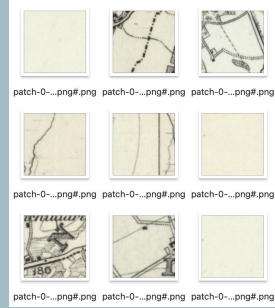
**<https://programminghistorian.org/en/lessons/computer-vision-deep-learning-pt1>**

**<https://programminghistorian.org/en/lessons/computer-vision-deep-learning-pt2>**

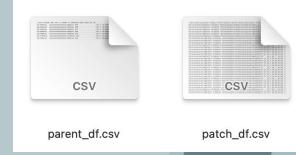
# Output

```
project
  |- your_notebook.ipynb
  |- patch_df.csv
  |- parent_df.csv
  |- patches.geojson
  `-- maps
      `-- ...
        `-- patches_100_pixel
            |-- patch-0-100-#map1.png#.png
            |-- patch-0-100-#map1.png#.tif
            |-- patch-100-200-#map1.png#.png
            |-- patch-0-100-#map1.png#.tif
            `-- ...
```

**Default:** Patches as .png images (no metadata)

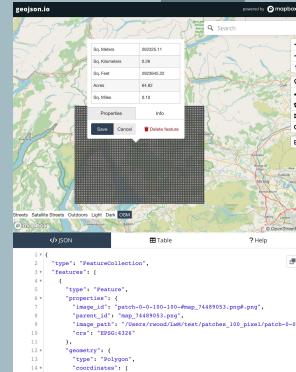
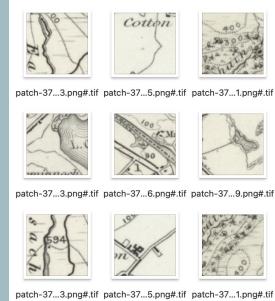


Metadata in .csv files



Patch metadata as .json (no images)

Georeferenced patches as .tiff images



# What do output files look like?



**PNG:** Portable Network Graphic  
(Raster image format)  
= image

**TIF(F):** Tag Image File Format  
(Raster image format)  
= image + coords



```
In [2]: import rasterio
In [5]: tif = rasterio.open("patches_100_pixel/patch-0-0-100-100-#map_74488553.png#.tif")
In [10]: tif.crs
Out[10]: CRS.from_epsg(4326)
In [14]: tif.bounds
Out[14]: BoundingBox(left=-5.42724609375, bottom=55.015425940562984, right=-5.418958992793642, top=55.02013891943969)
In [9]: tif.lnglat()
Out[9]: (-5.4231025432718205, 55.01778243000133)
```

# What do output files look like?

parent\_df.csv

- all metadata related to parent images

image_id	parent	image_path	shape	name	url	coordinates	crs	published_date	grid_bb
map_101105421.png		/Users/rwood/LwM.(5120, 716	map_1011054:	https://maps.nls.uk/view/	(-2.87567138671875, 54.257202811	EPSG:4326	1920	[(17, 64489, 41905)x(17, 645	
map_101428221.png		/Users/rwood/LwM.(4608, 665	map_1014282:	https://maps.nls.uk/view/	(0.57403564453125, 51.431751825	EPSG:4326	1909	[(17, 65745, 43611)x(17, 657	
map_101443444.png		/Users/rwood/LwM.(4608, 665	map_1014434:	https://maps.nls.uk/view/	(-3.8507080078125, 50.890906622	EPSG:4326	1888	[(17, 64134, 43925)x(17, 641	
map_101595433.png		/Users/rwood/LwM.(4608, 691	map_1015954:	https://maps.nls.uk/view/	(-2.76031494140625, 52.335339071	EPSG:4326	1884	[(17, 64531, 43078)x(17, 645	
map_101089504.png		/Users/rwood/LwM.(4864, 742	map_1010895:	https://maps.nls.uk/view/	(-3.197021484375, 54.93503226271	EPSG:4326	1926	[(17, 64372, 41480)x(17, 644	

# What do output files look like?

`predictions_patch_df.csv`

- predictions and metadata related to patches

image_id	parent_id	image_path	shape	pixel_bounds	coordinates	crs	polygon	predicted_label	pred	conf
patch-0-0-134-134-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 0, 134, 134)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	1.0000
patch-0-134-134-268-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 134, 134, 268)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	1.0000
patch-0-268-134-402-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 268, 134, 402)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	1.0000
patch-0-402-134-536-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 402, 134, 536)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	1.0000
patch-0-536-134-670-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 536, 134, 670)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	0.9998
patch-0-670-134-804-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 670, 134, 804)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	0.9998
patch-0-804-134-938-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 804, 134, 938)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	1.0000
patch-0-938-134-1072-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 938, 134, 1072)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	0.9999
patch-0-1072-134-1206-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 1072, 134, 1206)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 no		0	0.9993
patch-0-1206-134-1340-#map_99383316.png		/Users/rwood/LwM/m	(134, 134, 4)	(0, 1206, 134, 1340)	(-0.16204833984	EPSG:4326	POLYGON ((-0.160610 railspace		1	1.0000

# What do output files look like?

One “feature” = one patch!

Properties = “metadata”

Geometry = a polygon with a bounding box (of latitude and longitudes) describing the patch

# patches.geojson

```
{  
  "type": "FeatureCollection",  
  "crs": {  
    "type": "name"  
  },  
  "properties": {  
    "name": "urn:ogc:def:crs:OGC:1.3:CRS84"  
  },  
  "features": [  
    {  
      "type": "Feature",  
      "properties": {  
        "image_id": "patch-0-0-100-100-#map_74489053.png#.png",  
        "parent_id": "map_74489053.png",  
        "image_path": "/Users/rwood/LwM/test/patches_100_pixel/patch-0-0-100-100-#map_74489053.png#.png",  
        "crs": "EPSG:4326"  
      },  
      "geometry": {  
        "type": "Polygon",  
        "coordinates": [  
          [  
            [  
              [-5.462904305293642,  
               56.571589039597221  
              ],  
              [  
                [-5.462904305293642,  
                 56.57612776906884  
                ],  
                [  
                  [-5.47119140625,  
                   56.57612776906884  
                  ],  
                  [  
                    [-5.47119140625,  
                     56.571589039597221  
                    ],  
                    [  
                      [-5.462904305293642,  
                       56.571589039597221  
                      ]  
                    ]  
                  ]  
                ]  
              ]  
            ]  
          ]  
        ]  
      }  
    },  
    {  
      "type": "Feature",  
      "properties": {  
        "image_id": "patch-0-100-100-200-#map_74489053.png#.png",  
        "parent_id": "map_74489053.png",  
        "image_path": "/Users/rwood/LwM/test/patches_100_pixel/patch-0-100-100-200-#map_74489053.png#.png",  
        "crs": "EPSG:4326"  
      }  
    }  
  ]  
}
```

A GeoJSON  
“FeatureCollection”  
contains a list of Features

## Coordinate reference system

# Iterative question development and experiment design

A craft, not a science!

- Which maps?
- What metadata?
- Which labels?
- *What size patches?*

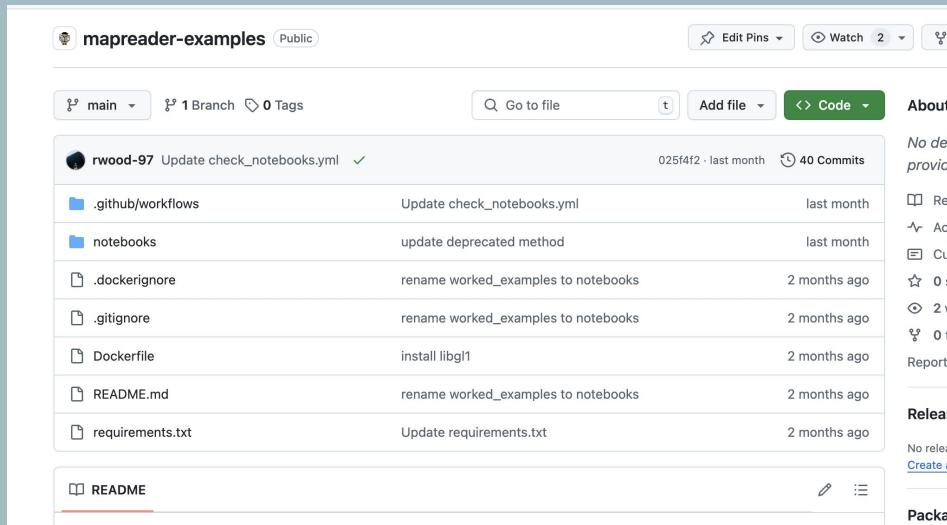
## Input guidance

### Table of Contents

- Input options
  - Option 1 - If you want to download map sheets from a TileServer
  - Option 2 - If your files are already saved locally
- Recommended directory structure
- Preparing your metadata
  - Option 1 - Using a `csv`, `xls` or `xlsx` file
  - Option 2 - Loading metadata from other file formats

# More MapReader example notebooks!

**<https://github.com/maps-as-data/MapReader-examples>**



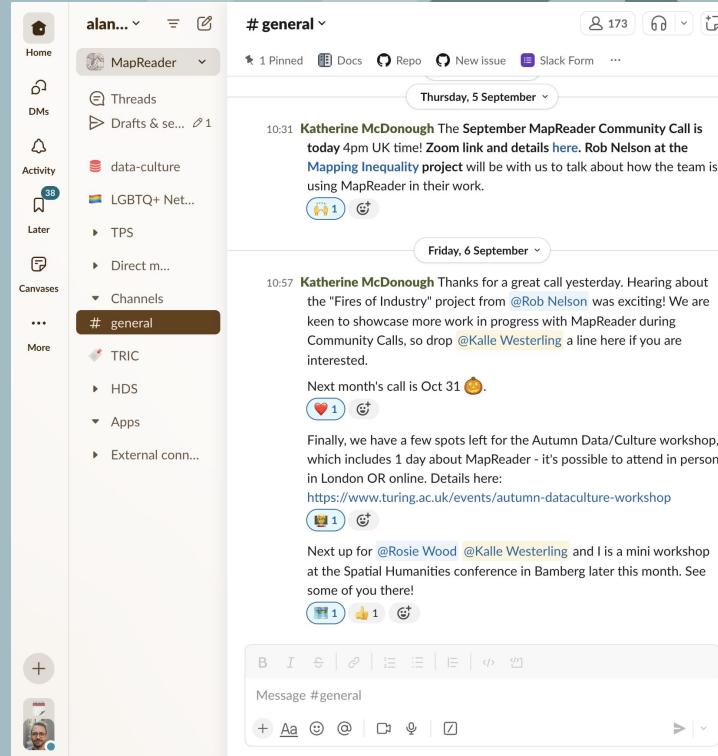
A screenshot of a GitHub repository page for 'mapreader-examples'. The repository is public and has 1 branch and 0 tags. The main commit is by 'rwood-97' titled 'Update check\_notebooks.yml' with a green checkmark, made last month. Below it is a list of other commits:

File	Commit Message	Date
.github/workflows	Update check_notebooks.yml	last month
notebooks	update deprecated method	last month
.dockerignore	rename worked_examples to notebooks	2 months ago
.gitignore	rename worked_examples to notebooks	2 months ago
Dockerfile	install libgl1	2 months ago
README.md	rename worked_examples to notebooks	2 months ago
requirements.txt	Update requirements.txt	2 months ago

The README file is also visible at the bottom.

# Stay in touch: Slack

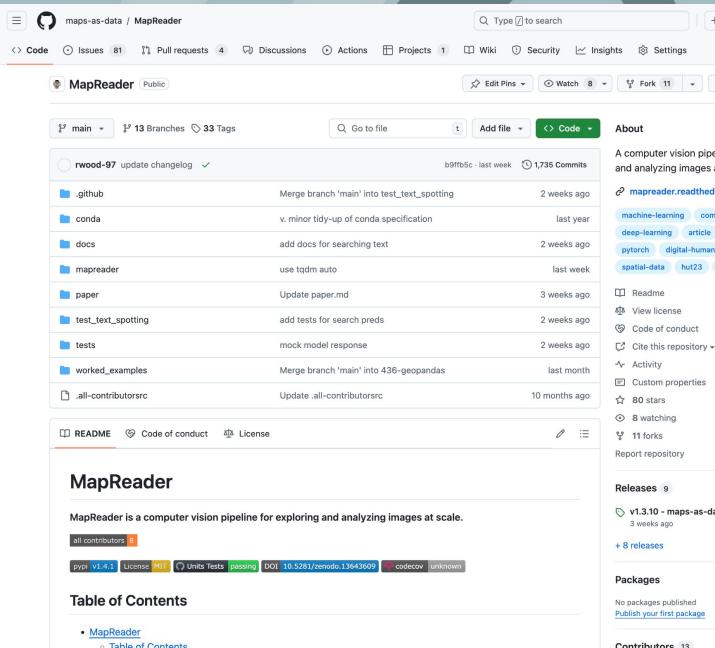
- **Get Help Fast:** Ask questions and get immediate support from the team and community.
- **Stay Informed:** Hear about new releases, features, and events first.
- **Share Ideas:** Discuss, learn, and exchange real-world use cases with others.
- **Shape the Future:** Contribute feedback, suggestions, or code to improve MapReader.
- **Connect with Experts:** Network with researchers and developers using MapReader.



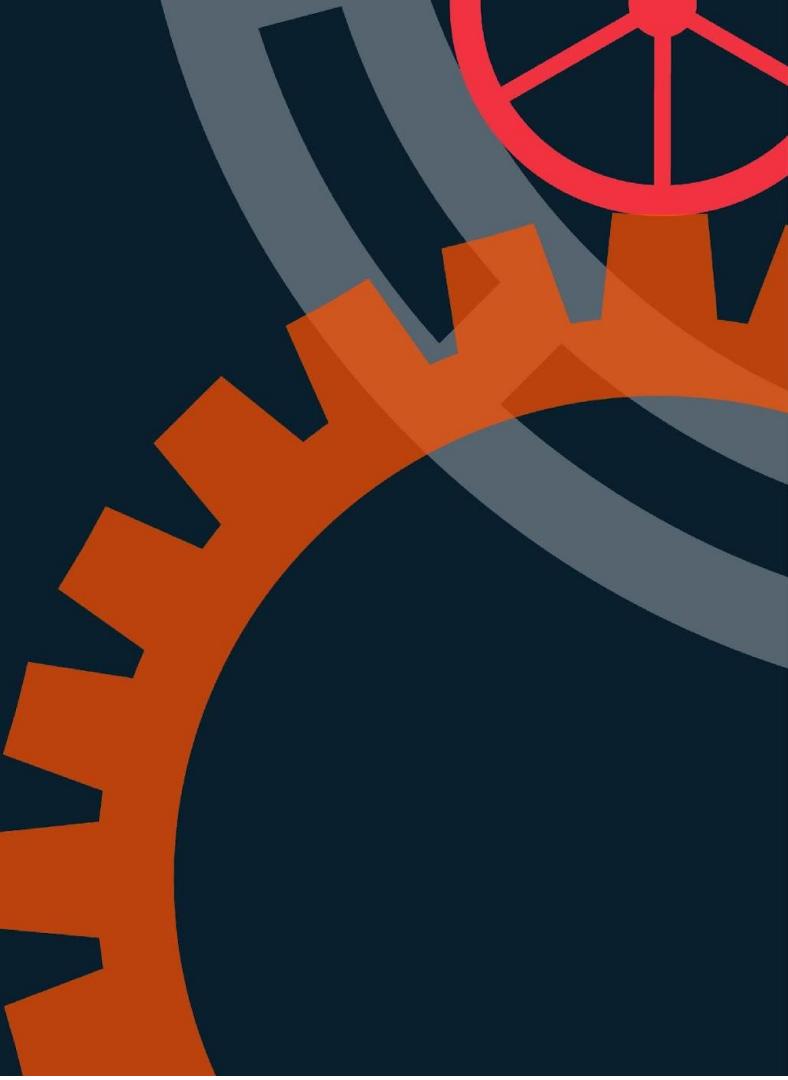
# Stay in touch: Github

<https://github.com/maps-as-data/MapReader>

- **Share Feedback:** Report bugs, suggest features, or give general input.
- **Improve Docs:** Help make guides and tutorials clearer by raising issues.
- **Track Progress:** Follow development updates in GitHub discussions.
- **Contribute Code:** Submit pull requests to add features or fix bugs. (For those who are more code-inclined!)
- **Collaborate:** Join discussions to shape the future of MapReader.



# Thank you!



## How to find all georeferenced layer URLs

1. To access any of our georeferenced maps and their associated XYZ or WMTS URLs (for adding the layer into QGIS, ArcGIS, geojson.io or OpenStreetMap - see [Guides](#)), go to our [Georeferenced Maps viewer](#) to browse our maps and select the dark blue **Overlay** tab in the footer for your chosen map (see image).

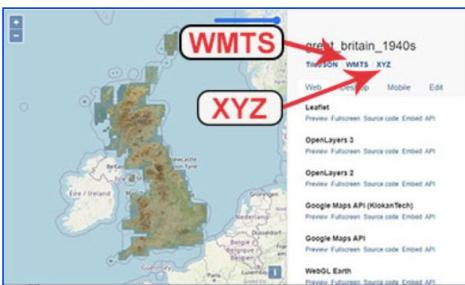
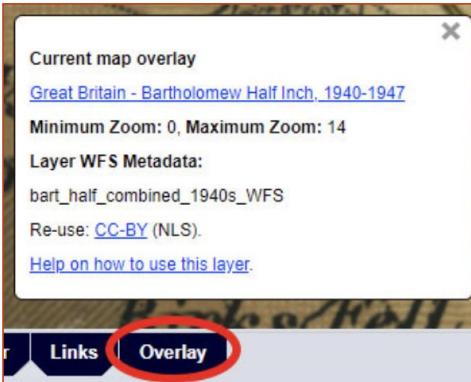
2. Once you select this dark blue **Overlay** tab, the **Current map overlay** link in the popup box will take you to a home page appearing as shown in the image (or see, for example, the [Bartholomew half-inch 1940s layer](#)). Many of our georeferenced layers have home pages in this style, with links to the WMTS and XYZ URLs to the upper right.

For using XYZ layers, note down too the **Maximum Zoom Level**.

3. The **Layer WFS Metadata** is the name of the TypeName or Layer in our Web Feature Service, allowing you to see the geographic metadata relating to the specific map within the georeferenced layer. Further details are available on how to use this in [QGIS](#) and [ArcGIS](#).

4. Please comply with the specific **Re-use** terms, as stated for each layer. NLS has a default [CC-BY](#) re-use terms, but there are layers which have other re-use terms ([further information](#)).

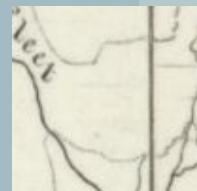
5. You can then follow the **Guides** below, to paste these URLs and re-use the layers in QGIS, ArcGIS, geojson.io, or OpenStreetMap.



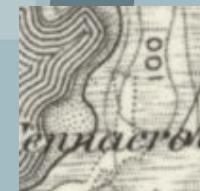
# Annotations - sorting patches

Sorting patches can be helpful to get the most “interesting” patches first.

- Can sort by any value in your patch dataframe
- Mean pixel values a good choice



[0.88, 0.87, 0.81]  
Higher = closer  
to white



[0.72, 0.71, 0.66]  
Lower = closer to  
black

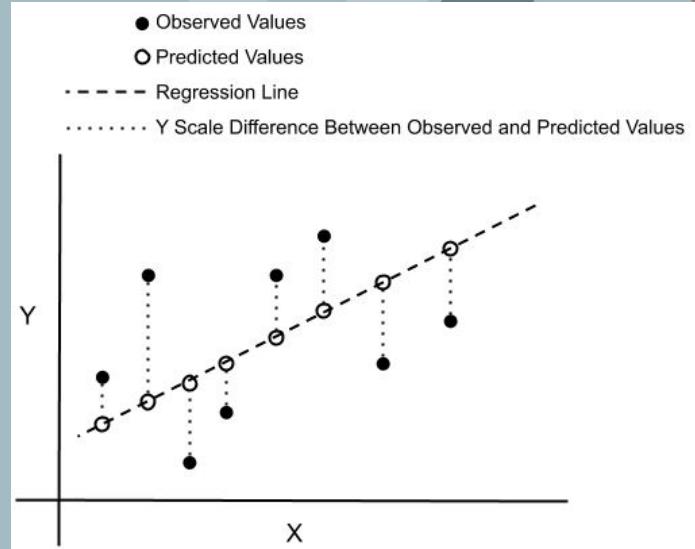
# How many patches should be annotated?

We cannot give a number, but here are some tips:

- Start by creating **representative validation & training sets**. The former is not used for training.
- **Progressively** increase the number of patches in the training set.
- In most experiments, patch-level classification is quick, so we can **iteratively check the performance** of our models.
  - e.g. in the railspace experiments, we visually inspected the results and identified some of the systematic errors made by our model. Then went back and annotated these problem areas.
- Use **contextual** information. Do you expect a certain pattern in neighboring patches? You can use that to identify and locate possible errors.
- Use external datasets (e.g., StopsGB and railway patches).

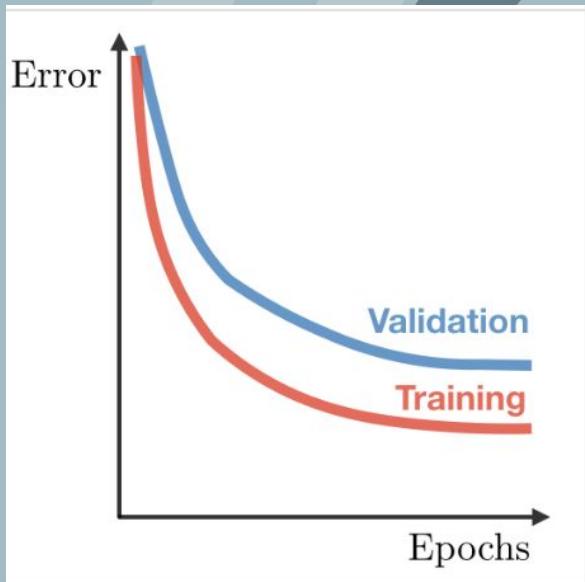
# Loss function, optimiser and scheduler

- **Loss function** calculates error
  - e.g. mean squared error (MSE)
- **Optimiser** works out how to adjust the neural network to reduce the error
- **Scheduler** controls how much the neural network changes throughout training



# Metrics

- Loss - **defined by loss function**



# Metrics

- Loss - defined by loss function
- Precision - **proportion of positive predictions that are correct**

$$\text{Precision} = \frac{TP}{TP + FP}$$



		PREDICTED VALUES	
		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE	TRUE POSITIVE	FALSE NEGATIVE
	NEGATIVE	FALSE POSITIVE	TRUE NEGATIVE

# Metrics

- Loss - defined by loss function
- Precision - proportion of positive predictions that are correct
- Recall - **proportion of actual positives correctly identified**

$$\text{Recall} = \frac{TP}{TP + FN}$$

		PREDICTED VALUES	
		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE	TRUE POSITIVE	FALSE NEGATIVE
	NEGATIVE	FALSE POSITIVE	TRUE NEGATIVE

# Metrics

- Loss - defined by loss function
- Precision - proportion of positive predictions that are true
- Recall - proportion of actual positives correctly identified
- F-scores - **combination of precision and recall**

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

		PREDICTED VALUES	
		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE	TRUE POSITIVE	FALSE NEGATIVE
	NEGATIVE	FALSE POSITIVE	TRUE NEGATIVE