https://tinyurl.com/mapreader-richmond

Introduction to MapReader

Katie McDonough

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

Our Partners

The Alan Turing Institute

British Library

University of Cambridge

University of East Anglia

University of Exeter

Queen Mary University of London

Our Funders

Arts & Humanities Research Council

UK Research and Innovation

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

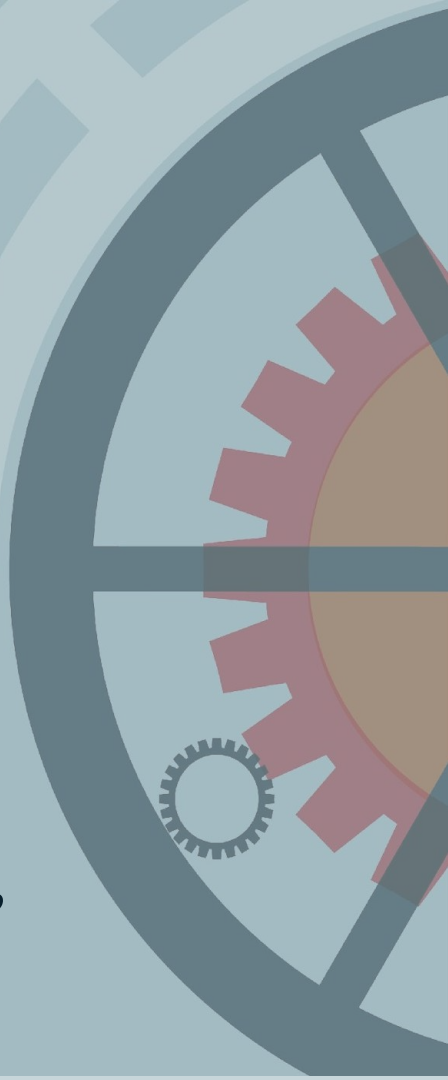
Credit: Olivia Vane for visualisation, National Library of Scotland for map images & metadata

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

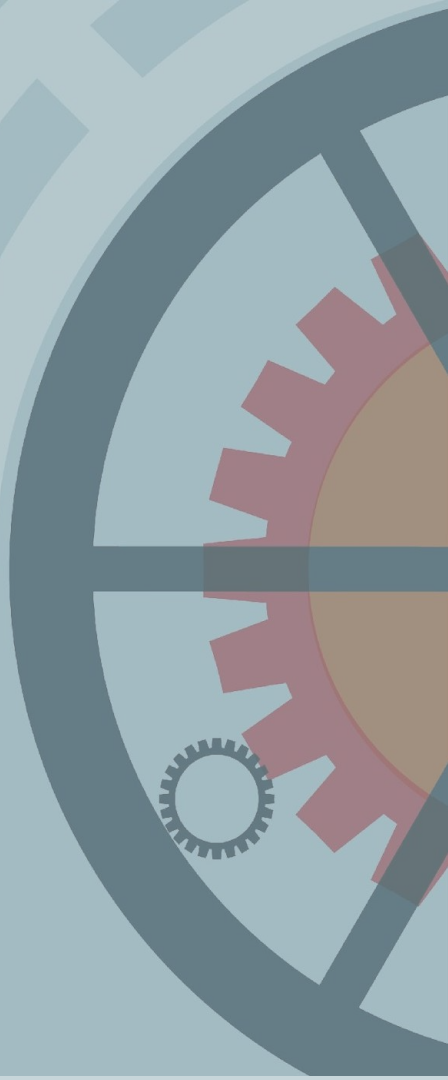
Ordnance Survey (series) maps as non-aggregated, high-resolution, national-coverage sources



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’.

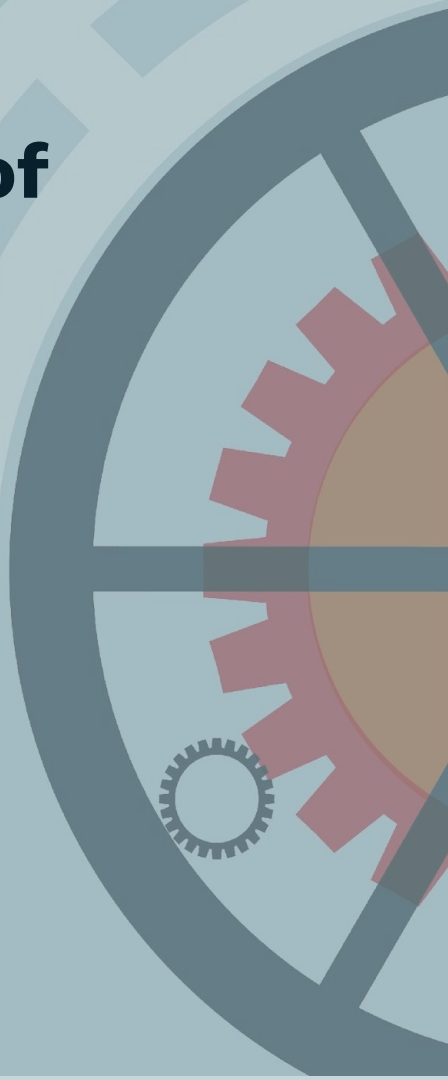
We want to ***use CV to advance interpretation, not define truth.***



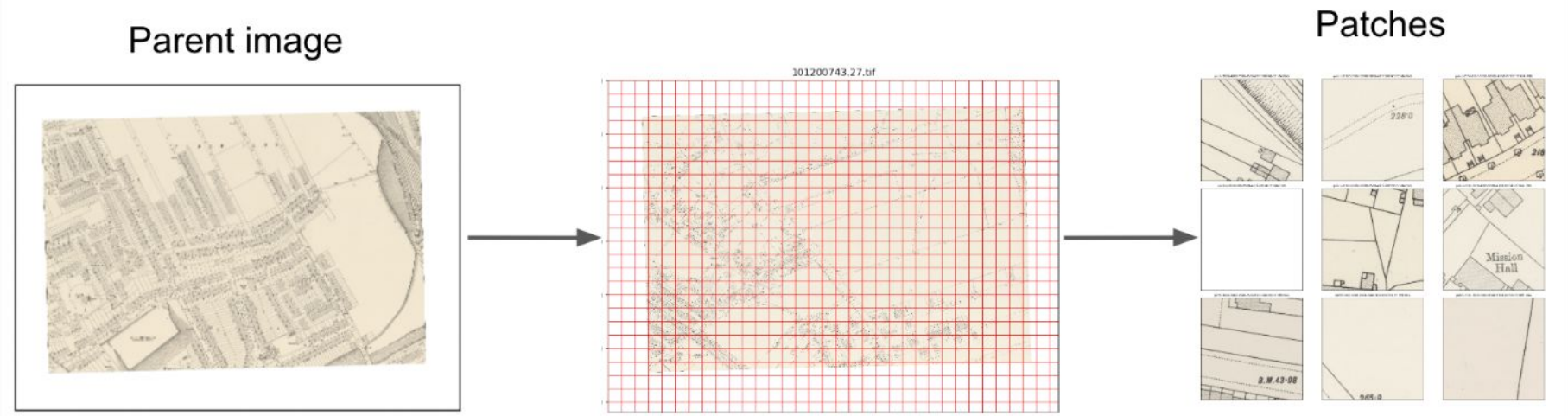
Step away from the GIS paradigm of vectorizing everything on a map

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



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

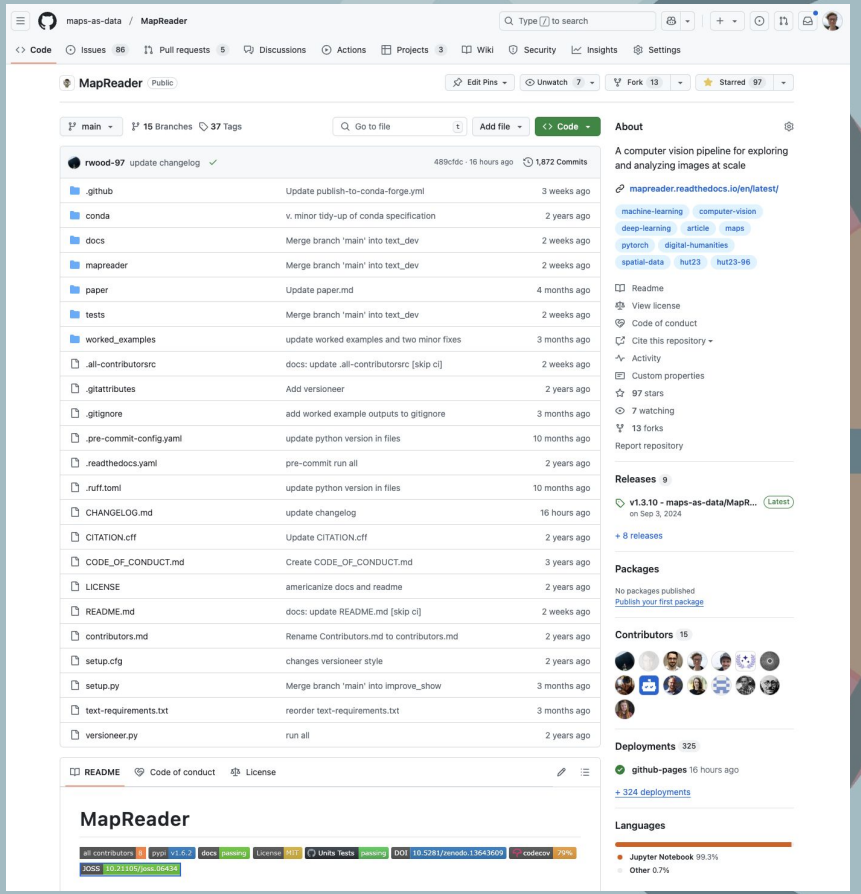


Image Classification for Map Patches

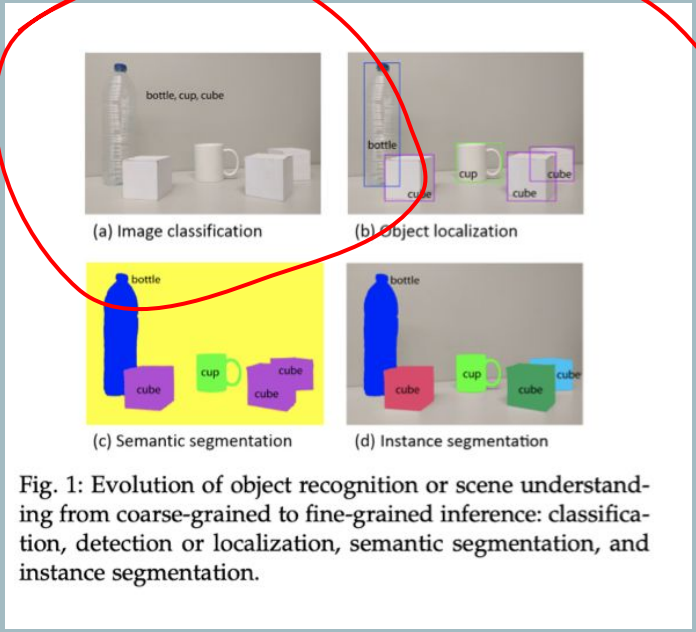
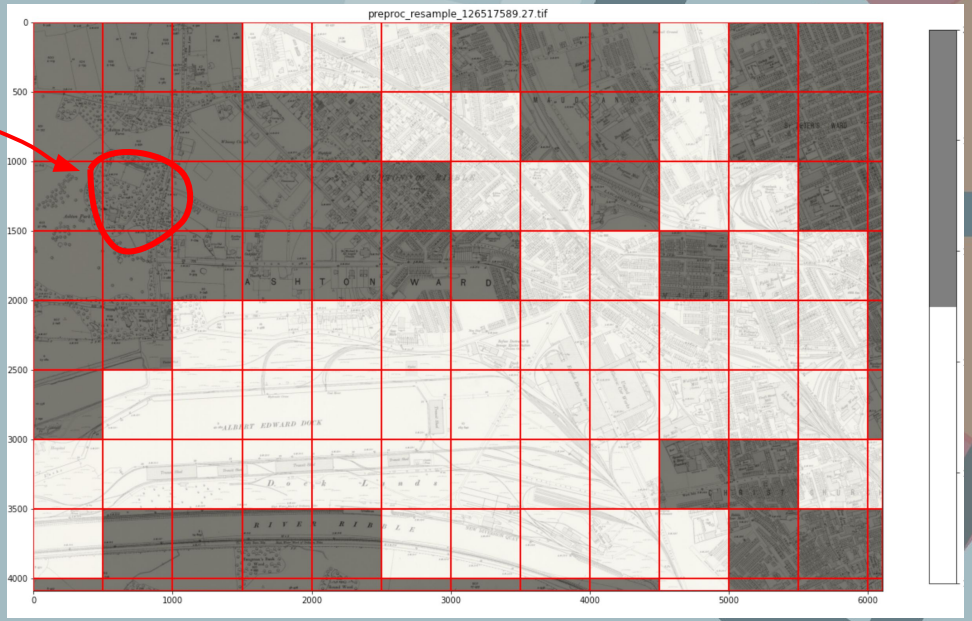


Fig. 1: Evolution of object recognition or scene understanding from coarse-grained to fine-grained inference: classification, detection or localization, semantic segmentation, and instance segmentation.

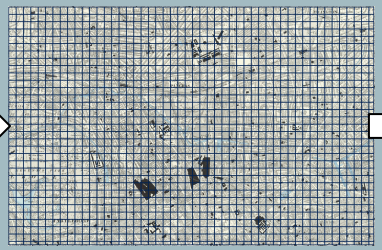
https://arxiv.org/pdf/1704.06857.pdf



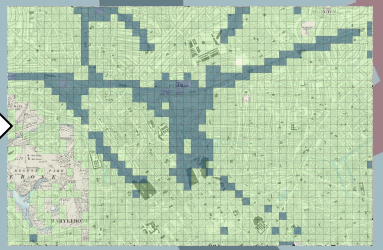
Map patches annotated as training data

Classification pipeline

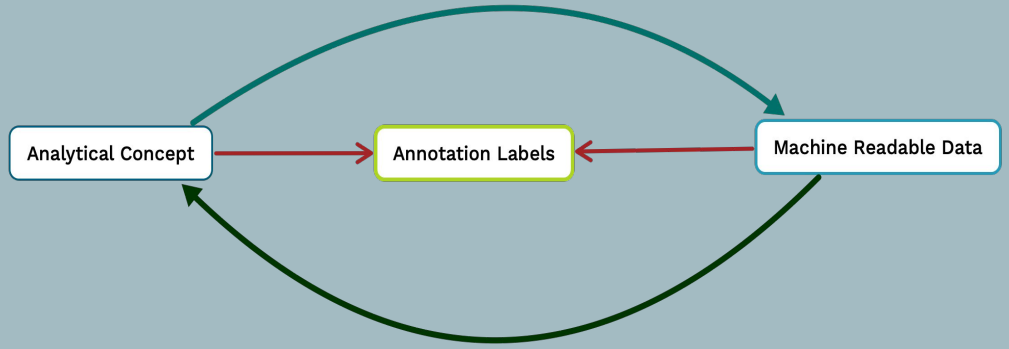
Download	Load	Annotate	Classify
Optional: - Download maps	- Load image files - Load metadata - Patchify	- Annotate patches based on visual features	- Train classifier - Predict labels

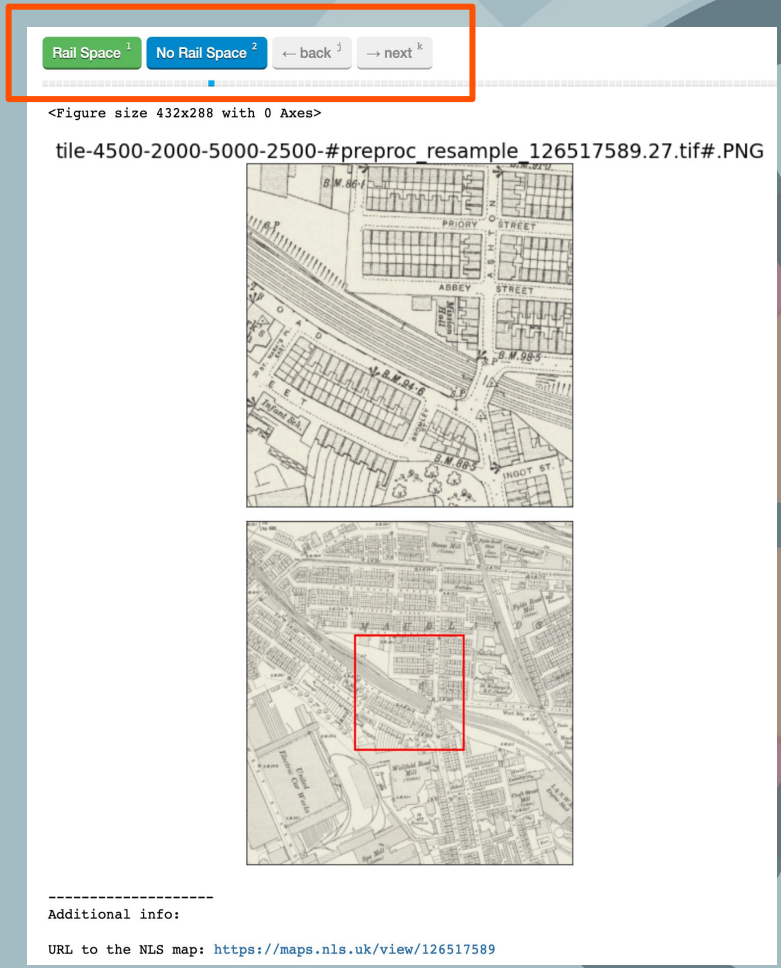


railspace



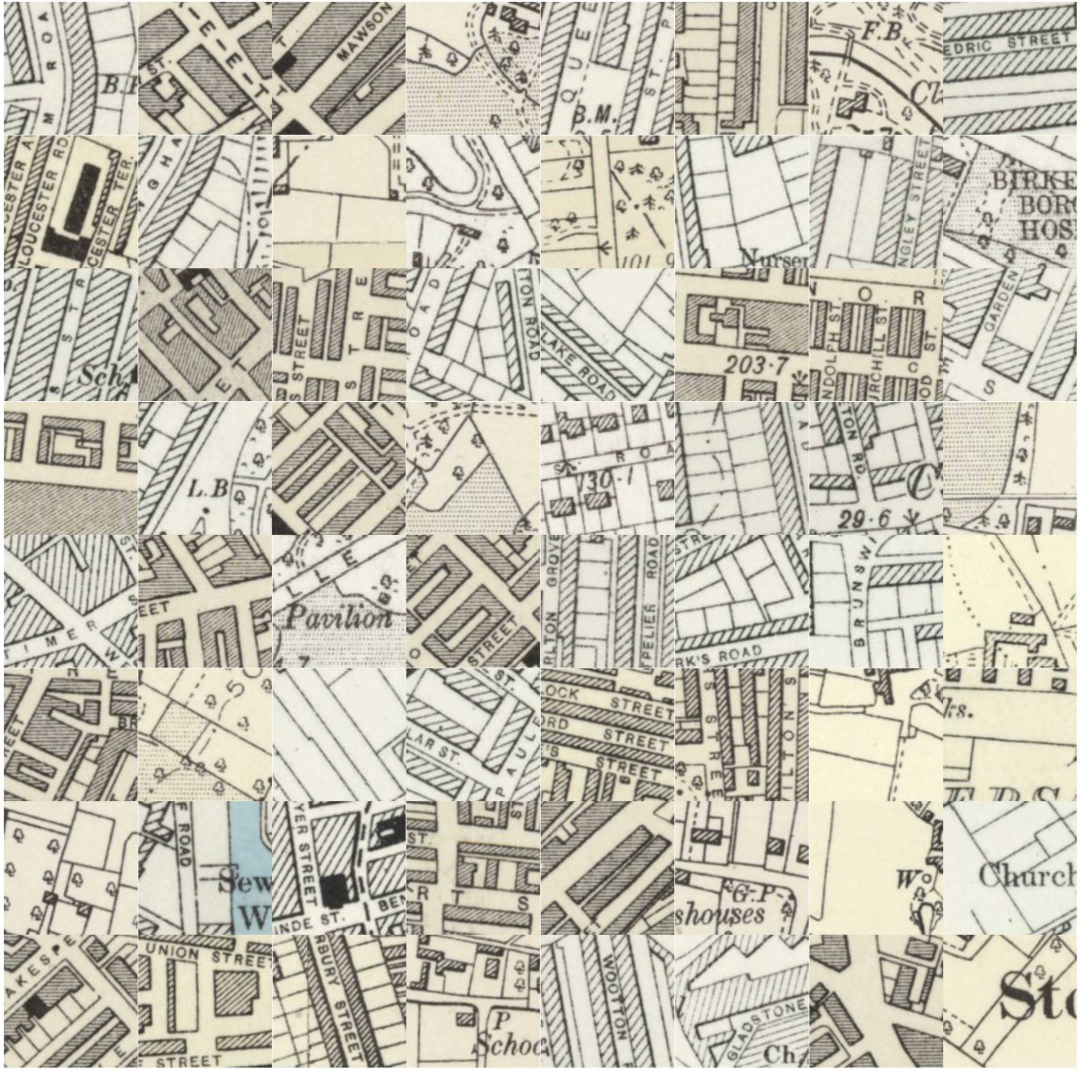
Annotating patches: What is a good label?





Railspace

Buildings


MAWSON
EDRIC STREET
LOUCESTER RD
Pavilion
LAKE ROAD
203·7
29·6
UNION STREET
WOOTTON
GLADSTONE
Church

Living with Machines

Computational Histories of the Age of Industry

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

ABOUT DIGITAL EDITIONS

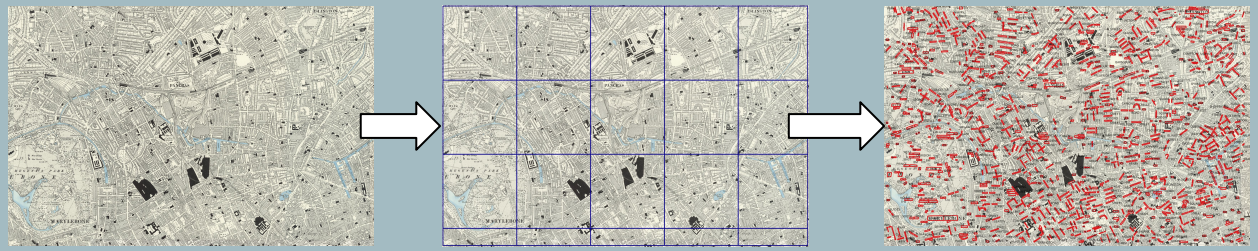
Living With Machines is a data-driven history of the coming of the machine age in Britain in the long nineteenth century. Featuring an innovative open access edition enhanced with interactive maps, datasets and visualisations, digital notebooks, video, audio and images, this book harnesses the combined 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.

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

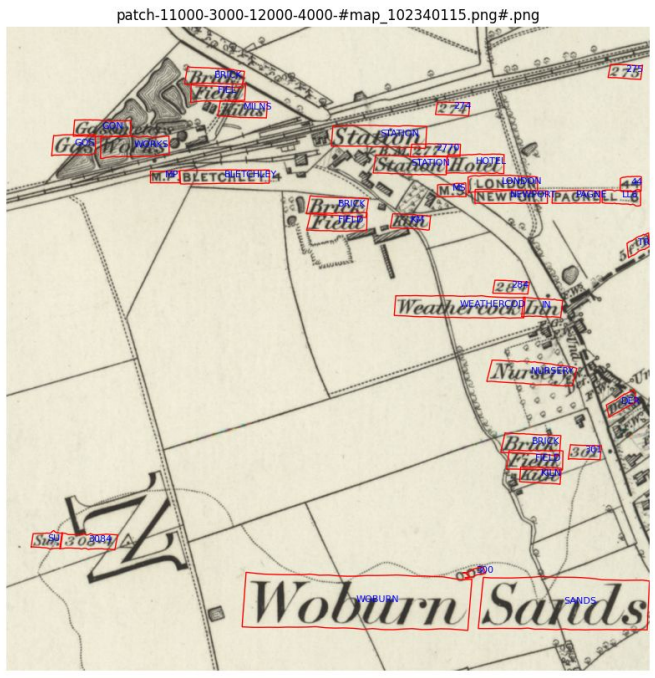
Download	Load	Spot text
Optional: - Download maps	- Load image files - Load metadata - Patchify	- Detect and recognise text



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!

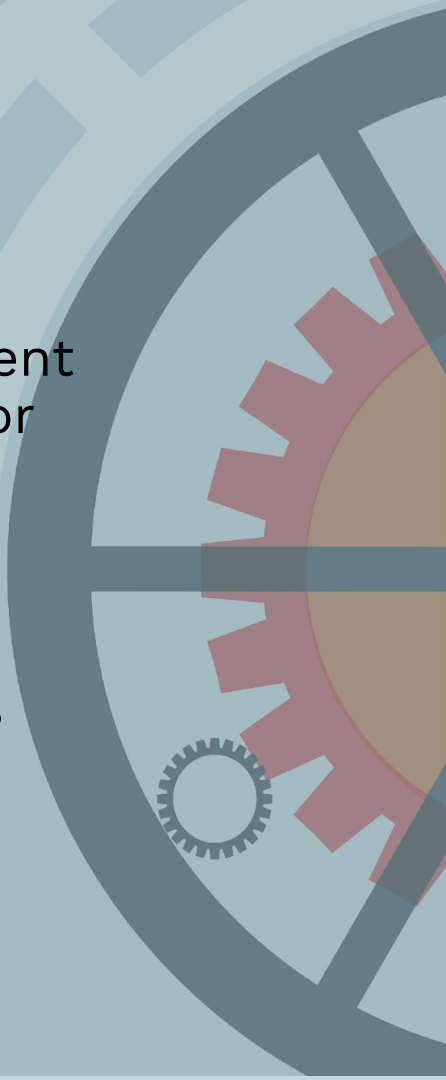


MapReader

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

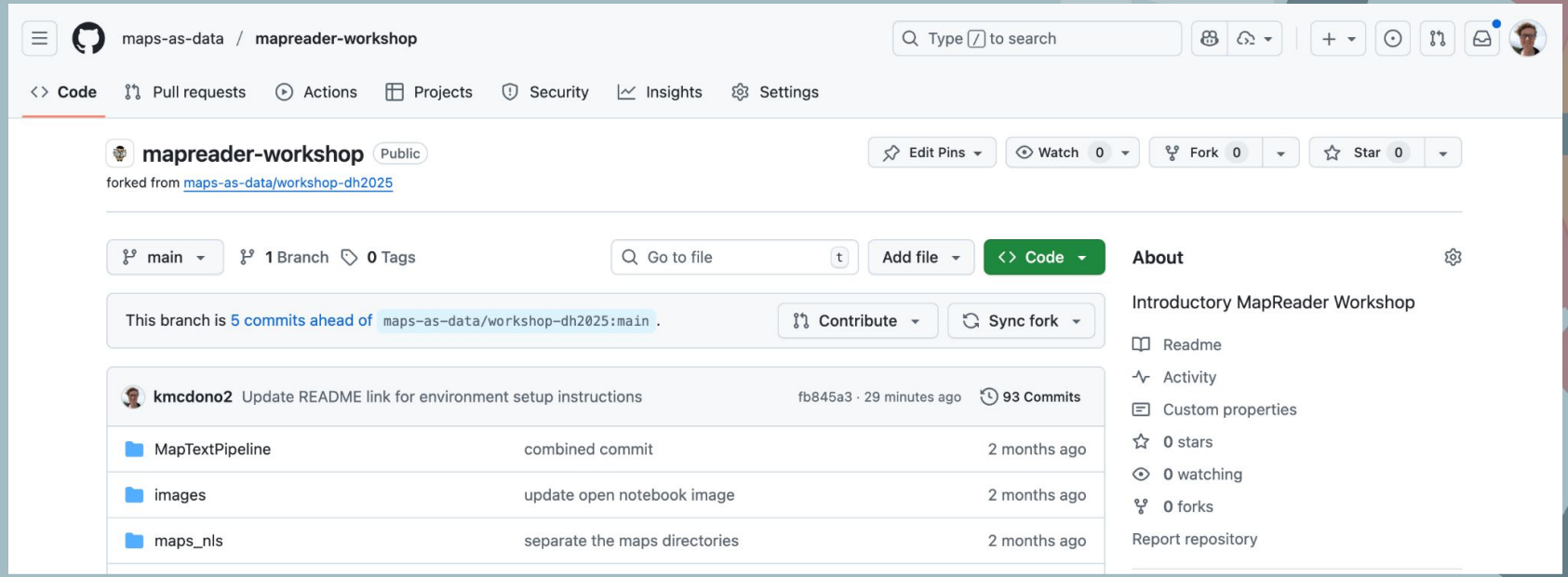
We privilege software design and methods that:

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



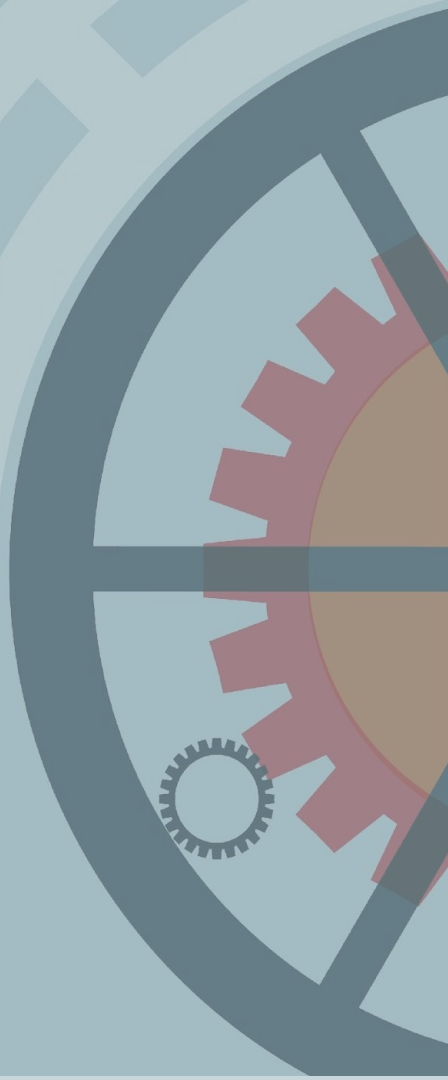
Using MapReader

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



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

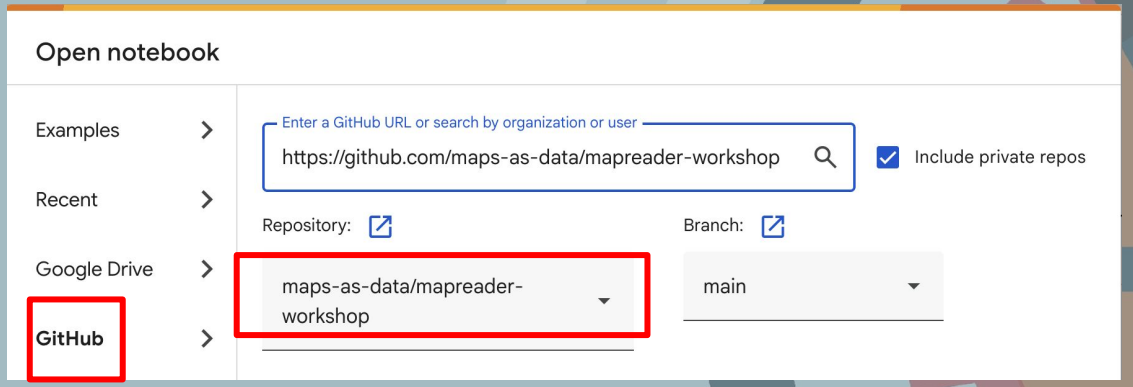
Follow the instructions for Google Colab in the README

Step 1: select 'Github'

Step 2: enter repo URL

Step 3: select repo

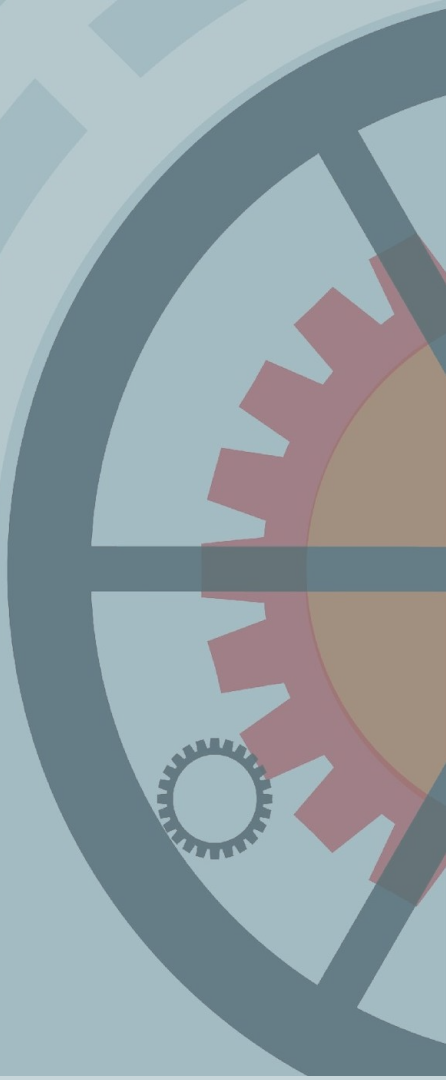
Step 4: open mapreader_classification **ipynb file**



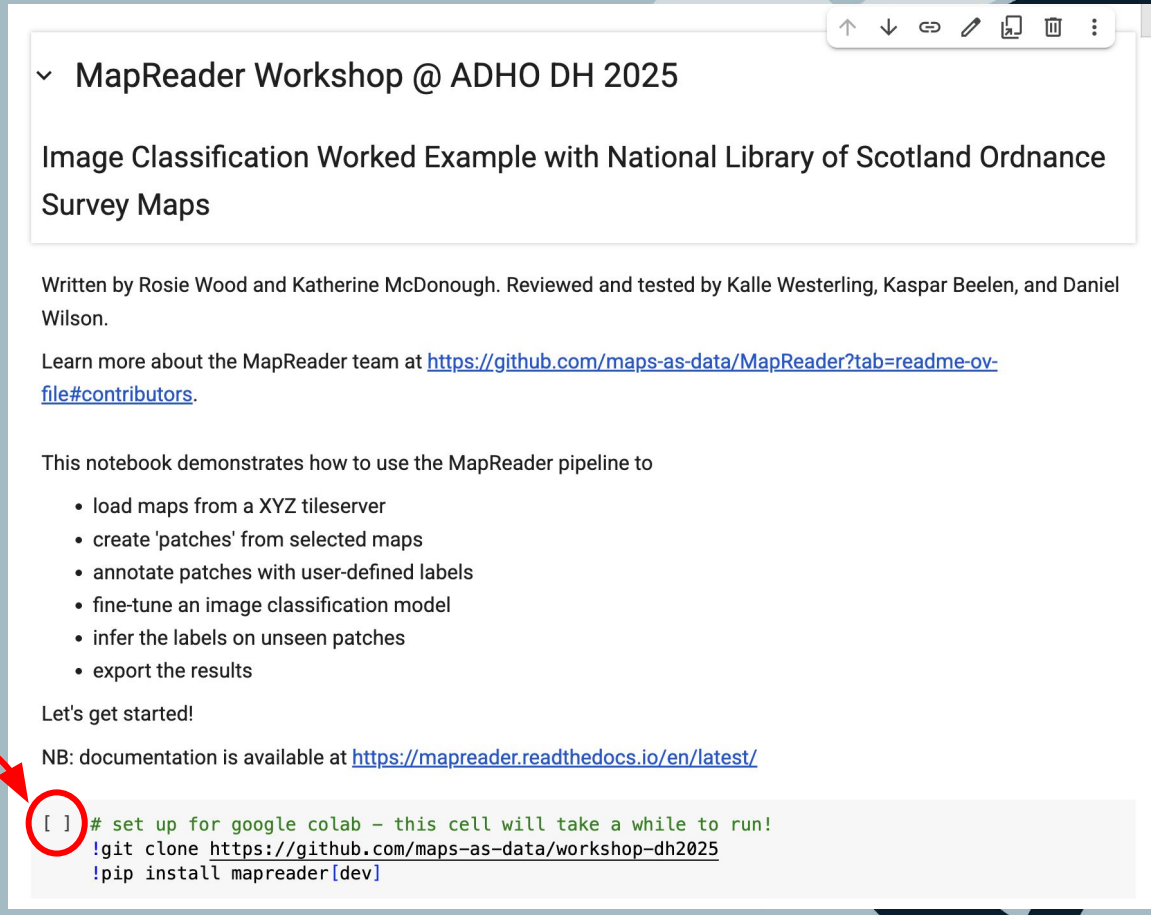
Jupyter Notebook Basics

- **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!

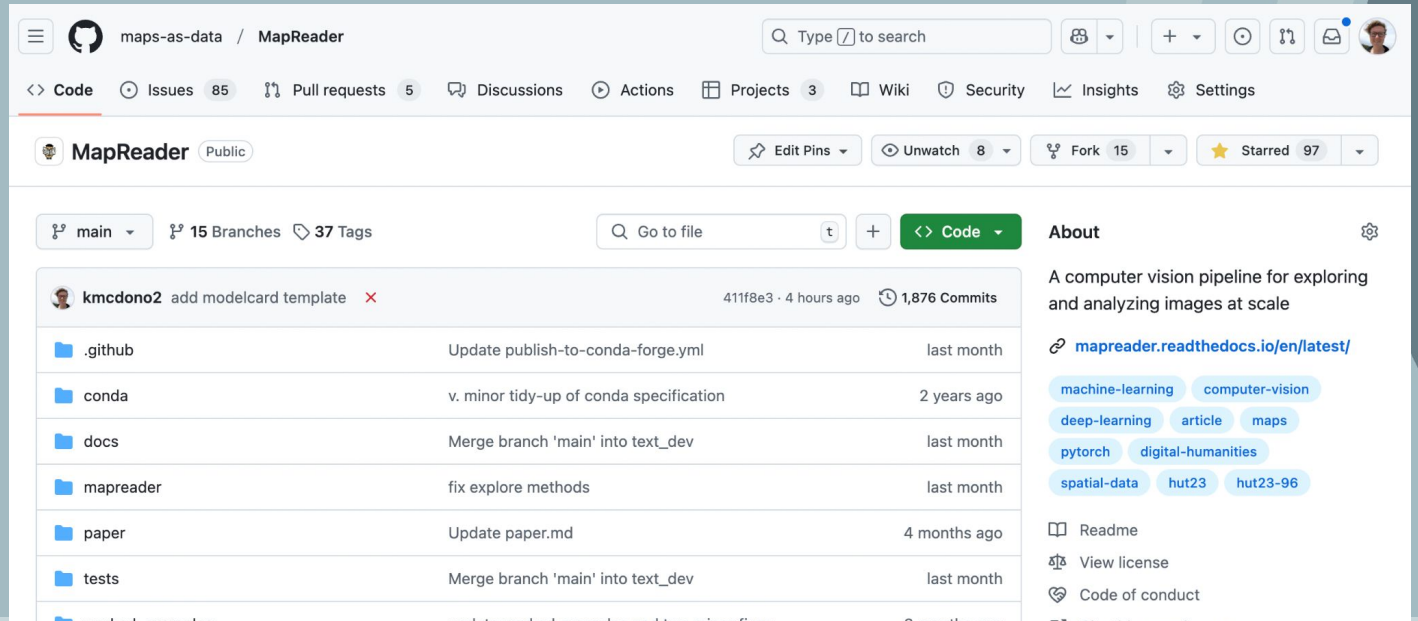


Open notebook & run first code cell



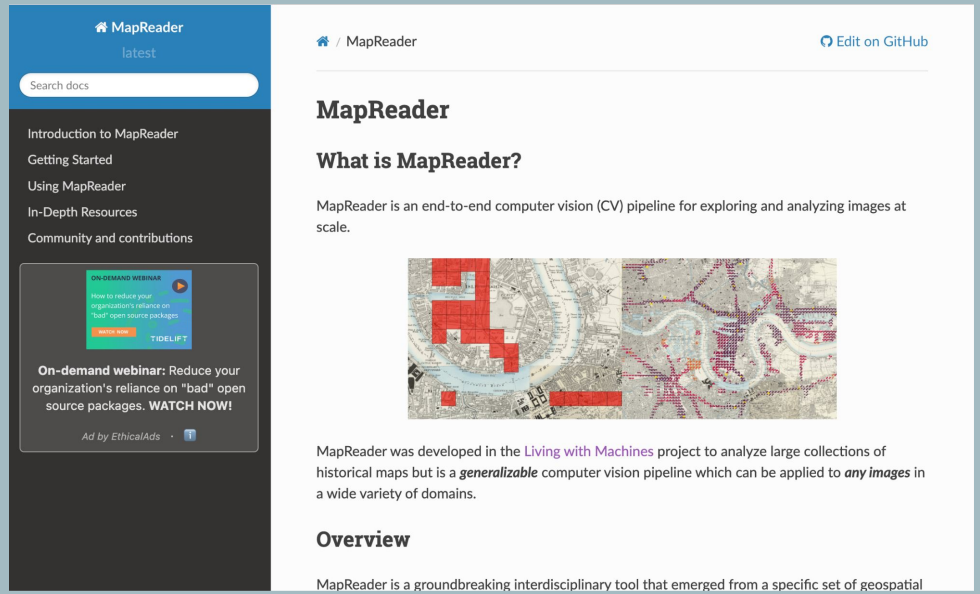
MapReader (Main) Github Repository

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



MapReader Documentation

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



Patch Classification Notebook Sections

No need to download maps, we have provided them!

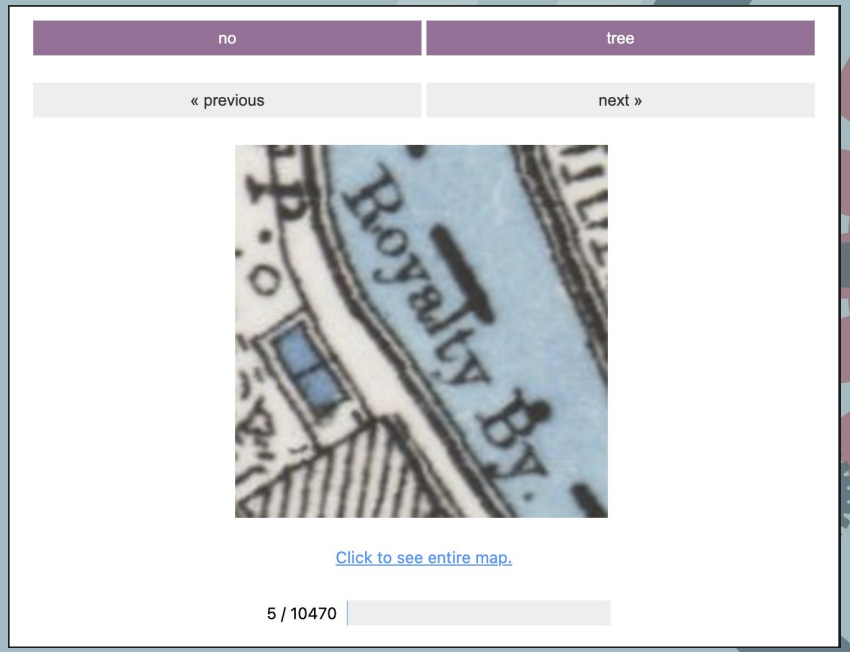
- **Install dependencies**
- **Load & Patchify maps**
- **Annotate patches with your label of choice**
- **Fine-tune a model**
- **Review model performance**
- **Infer labels on remaining patches**
- **Export labeled patch data**

Annotations - before we begin

Only one label per patch

Two approaches:

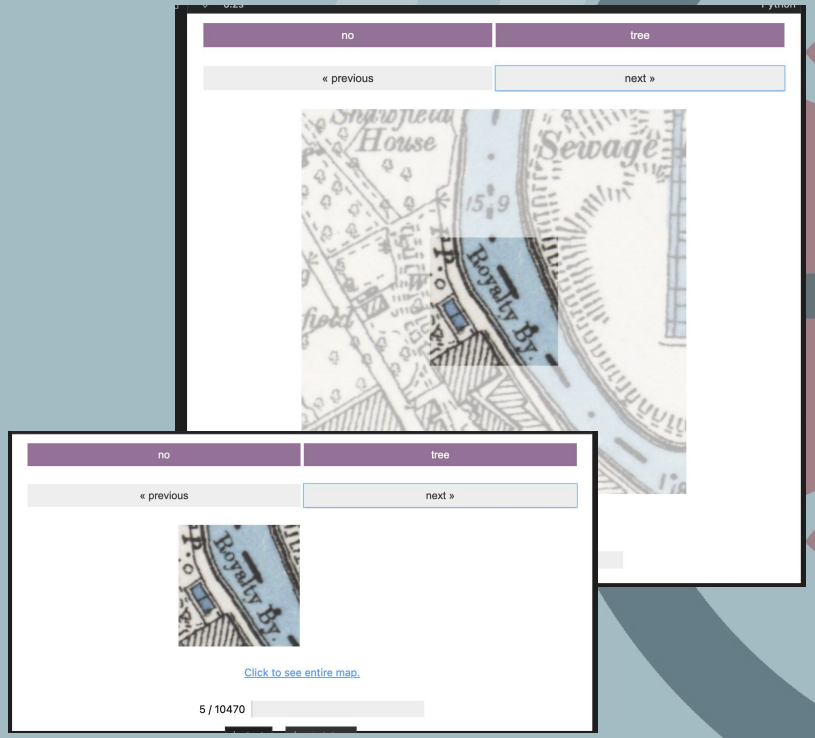
1. **yes/no (for 1 concept)**
2. **concept 1/concept n/neither concept**



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

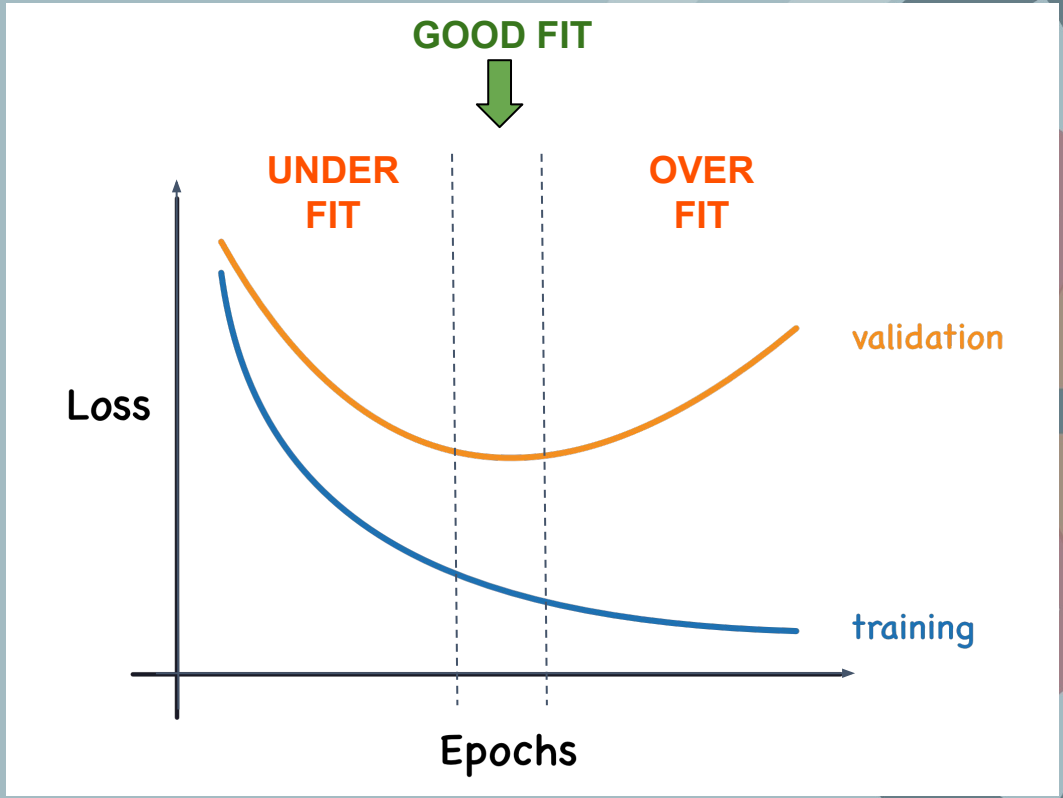
200 pixel
Woods

100 pixel
Trees

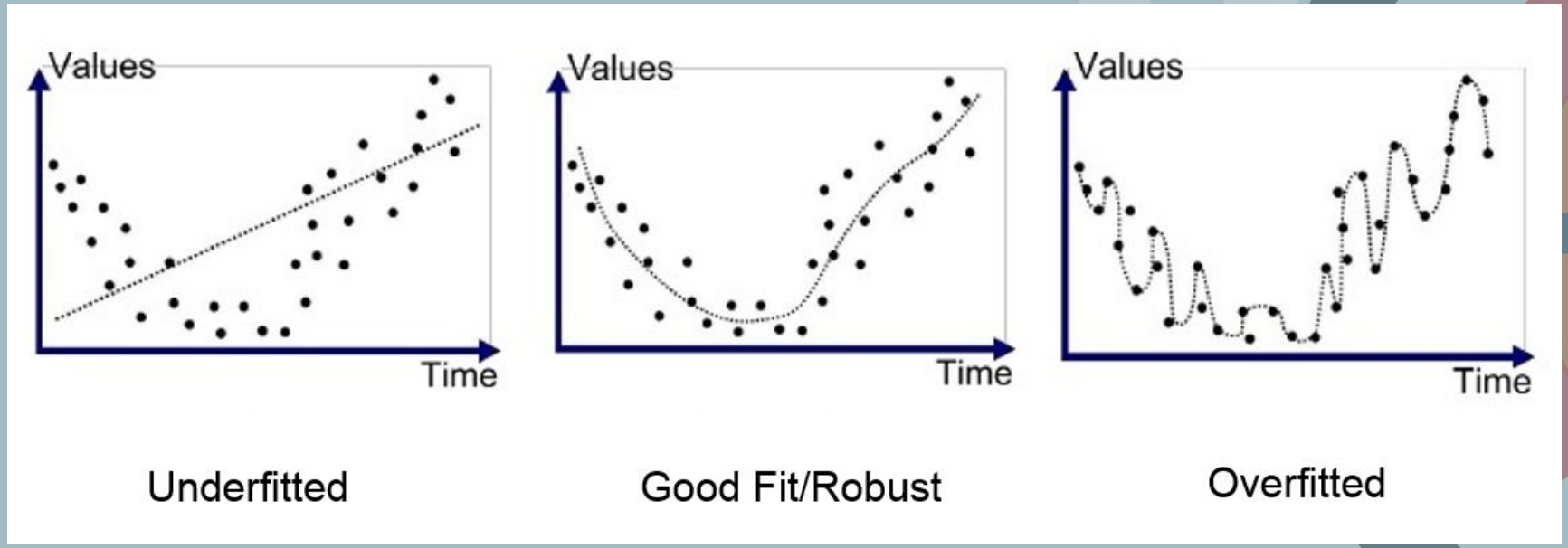
Training

Annotations split into train, validation and test datasets

Goal is to minimise both training and validation losses



Training

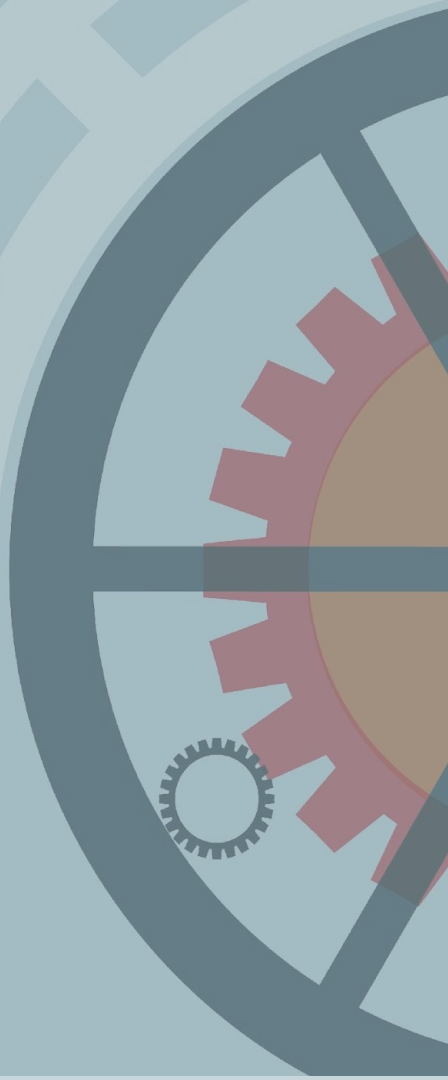

Values
Time
Underfitted
Values
Time
Good Fit/Robust
Values
Time
Overfitted

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



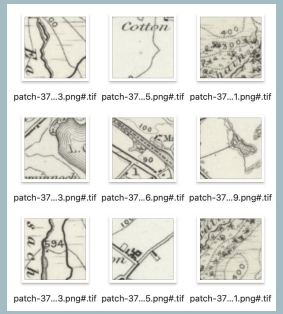
Output

```
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
     └── ...
```

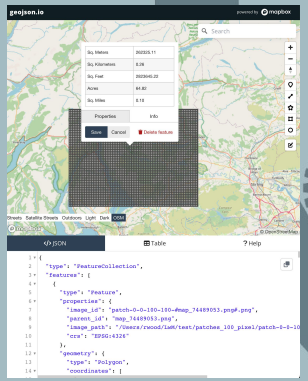
Default: Patches as .png images (no metadata)

Metadata in .csv files

Georeferenced patches as .tiff images



Patch metadata as .json (no 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_99	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_	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_	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_	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_	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_	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_	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	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-#ma	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-#ma	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

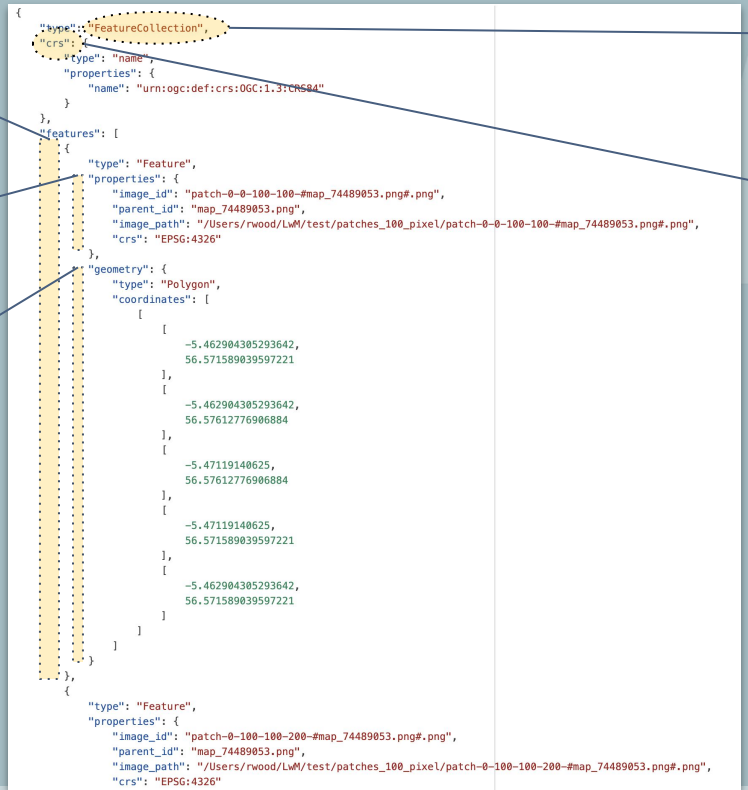
What do output files look like?

patches.geojson

One “feature” = one patch!

Properties = “metadata”

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



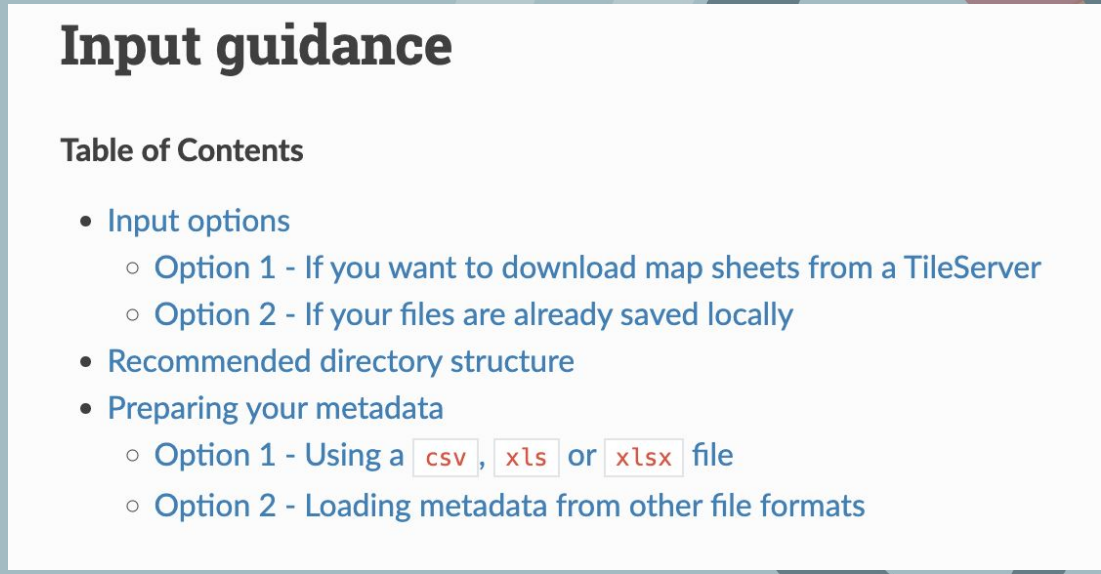
A GeoJSON “FeatureCollection” contains a list of Features

Coordinate reference system

Iterative question development and experiment design

A craft, not a science!

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



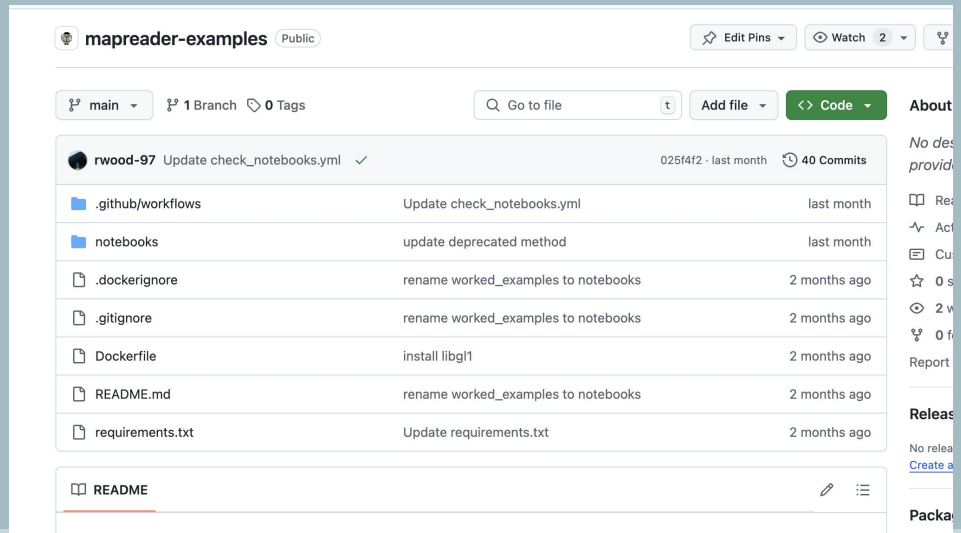
Input guidance

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

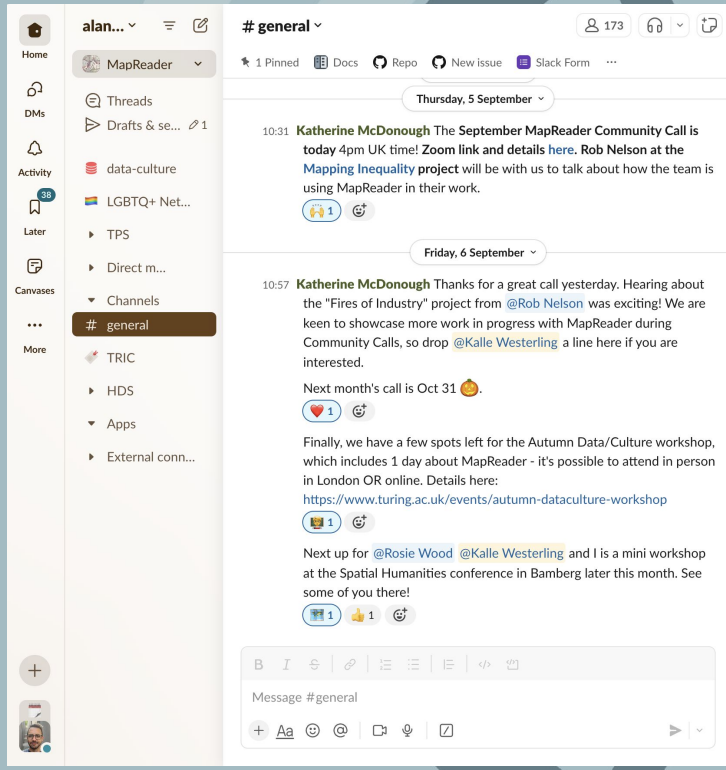
More MapReader example notebooks!

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



Stay in touch: Slack

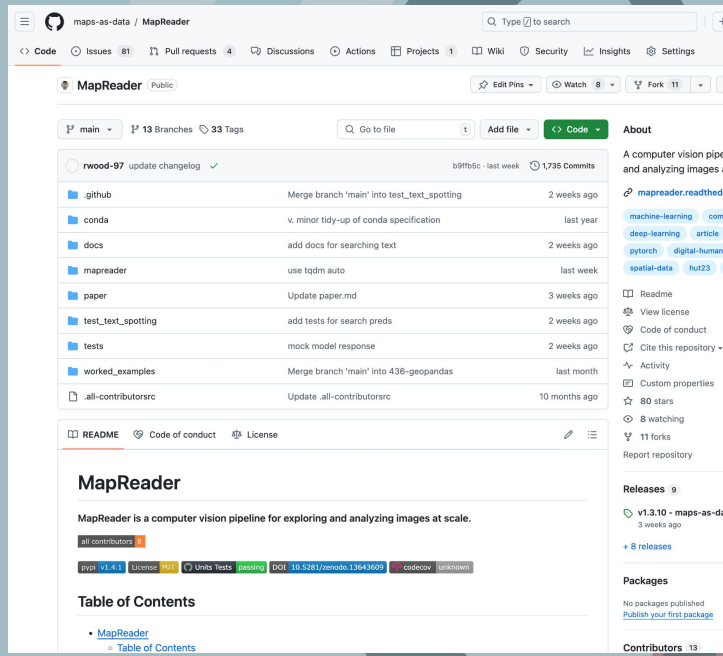
- **Get Help Fast:** Ask questions and get immediate support from the team and community.
- **Stay Informed:** Hear about new releases, features, and events first.
- **Share Ideas:** Discuss, learn, and exchange real-world use cases with others.
- **Shape the Future:** Contribute feedback, suggestions, 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

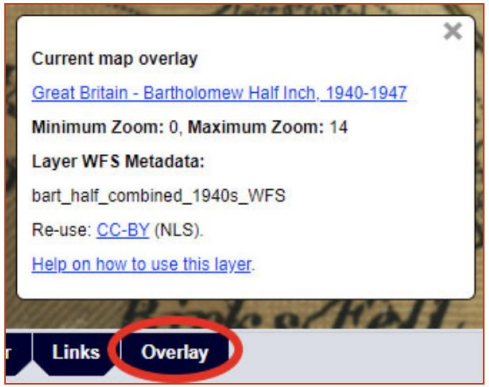
- **Share Feedback:** Report bugs, suggest features, or give general input.
- **Improve Docs:** Help make guides and tutorials clearer by raising issues.
- **Track Progress:** Follow development updates in GitHub discussions.
- **Contribute Code:** Submit pull requests to add features or fix bugs. (For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



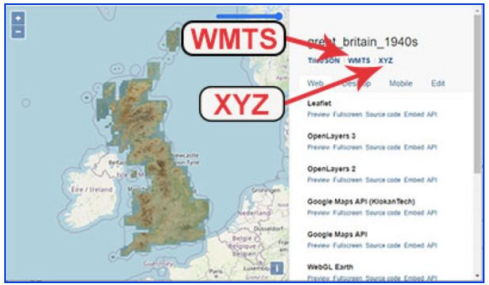
Thank you!

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

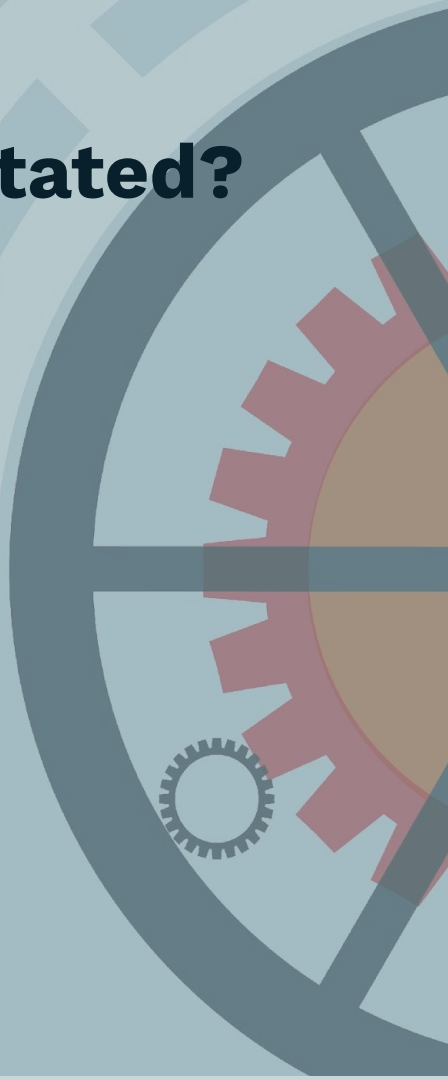
[0.88,0.87, 0.81]
Higher = closer to white

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

How many patches should be annotated?

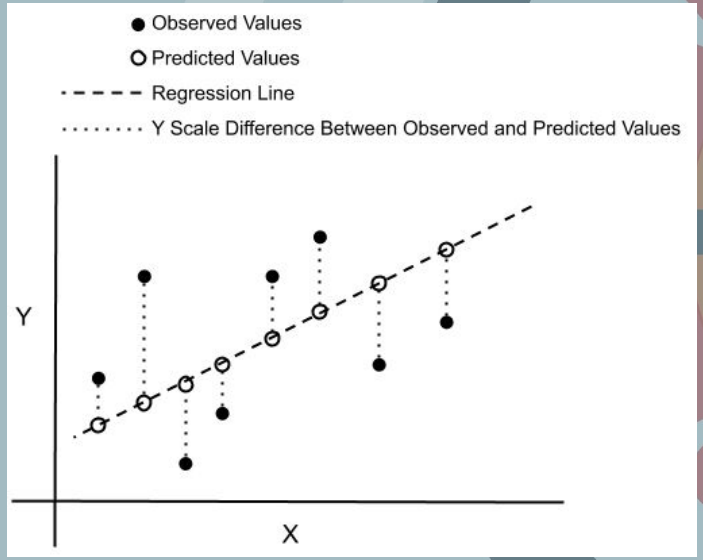
We cannot give a number, but here are some tips:

- 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 an 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).



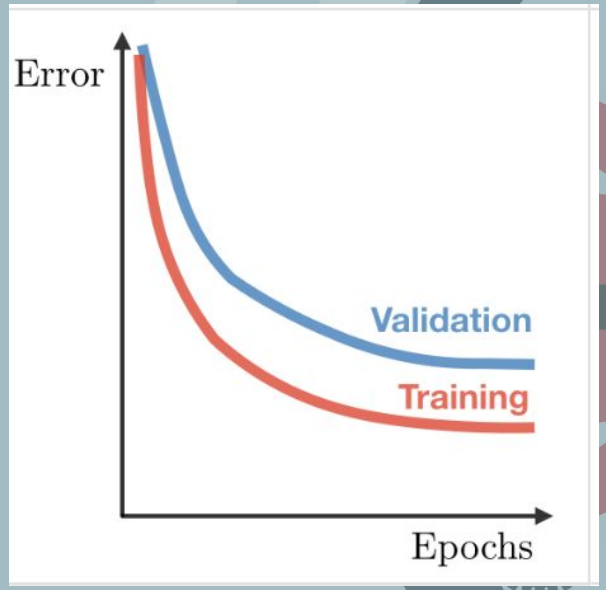
Loss function, optimiser and scheduler

- **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

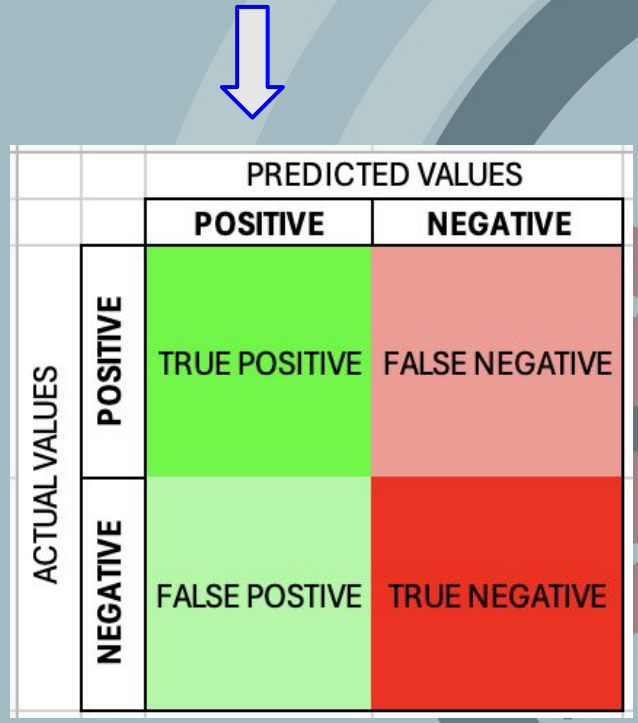
- Loss - **defined by loss function**



Metrics

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

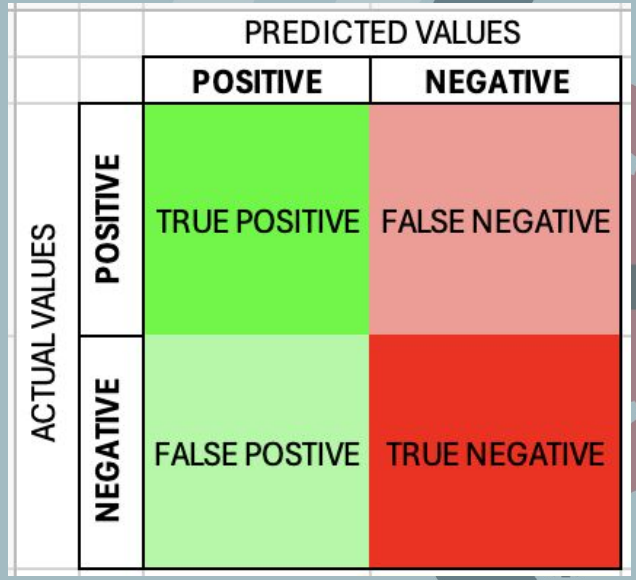
$$\text{Precision} = \frac{TP}{TP + FP}$$



Metrics

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

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



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**

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