MapReader: A Computer Vision Pipeline for the Semantic Exploration of Maps at Scale
Kasra Hosseini Daniel C.S. Wilson Kaspar Beelen Katherine McDonough
{khosseini,dwilson,kbeelen,kmcdonough}@turing.ac.uk
The Alan Turing Institute
UK

Figure 1: British railspace and buildings as predicted by a MapReader computer vision model. ≈30.5M patches from ≈16K nineteenth-century OS map sheets were used. (a) Predicted railspace; (b) predicted buildings; (c) and (d) predicted railspace (red) and buildings (black) in and around Middlesbrough and London, respectively. MapReader classifies information from large images or a set of images at a patch-level, as depicted in the figure insets. Map images courtesy of the National Library of Scotland (NLS).

ABSTRACT
We present MapReader, a free, open-source software library written in Python for analyzing large map collections. MapReader allows users with little computer vision expertise to i) retrieve maps via web-servers; ii) preprocess and divide them into patches; iii) annotate patches; iv) train, fine-tune, and evaluate deep neural network models; and v) create structured data about map content. We demonstrate how MapReader enables historians to interpret a collection of ≈16K nineteenth-century maps of Britain (≈30.5M patches), foregrounding the challenge of translating visual markers into machine-readable data. We present a case study focusing on rail and buildings. We also show how the outputs from the MapReader pipeline can be linked to other, external datasets. We release ≈62K manually annotated patches used here for training and evaluating the models.

CCS CONCEPTS
• Computing methodologies → Supervised learning by classification; Object recognition; Neural networks; • Software and its engineering → Software libraries and repositories; • Applied computing → Digital libraries and archives.
KEYWORDS
Computer vision, Deep learning, Supervised learning, Classification, Historical maps, Digital libraries and archives.

ACM Reference Format:
Kasra Hosseini Daniel C.S. Wilson Kaspar Beelen Katherine Mc-Donough . 2022. MapReader: A Computer Vision Pipeline for the Semantic Exploration of Maps at Scale. In The 6th ACM SIGSPATIAL International Workshop on Geospatial Humanities (GeoHumanities ’22) (GeoHumanities ’22), November 1, 2022, Seattle, WA, USA. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3557919.3565812

1 INTRODUCTION
Cultural heritage series map collections contain vast amounts of immensely detailed historical and geographical information, but the contents of these maps have until recently been inaccessible for computational research methods. Initiatives for scanning and georeferencing map collections have brought such material into the digital realm, allowing researchers to browse them easily on screen; however, it remains difficult for historians and geographers to search within these maps systematically in order to make claims about the past. This results both from the size of these collections and also because their contents are not yet machine-readable, and so not amenable to digital exploration and analysis. Our focus in this paper is on searching the contents of Britain’s Ordnance Survey (OS) maps. During the nineteenth century, the OS produced thousands of definitive map sheets for the nation at three different scales. A single series of maps printed at a scale of six inches to the mile, for example, contains ≈16K sheets. Newly available in digital form, these maps have the potential to address open historical research questions. One such example relates to the arrival of the railways in nineteenth-century Britain, which we explore using the method presented in this paper.

To investigate qualitative geographical phenomena computationally requires researchers to translate, in this case, a visual map feature into a machine-readable text label. This procedure is at the heart of MapReader, which applies what we term a “patchwork method” for dividing our source image into constituent cells or “patches” which are then annotated according to the presence of a target feature. This has certain affordances well suited to spatial analysis involving relational, additive or abstract target concepts, such as those typically of interest to humanists. We apply MapReader to OS maps, but it can be used to search within any series, or large homogeneous set of, maps. It could also be used with remote sensing imagery.

Annotation of scanned maps is a form of translation which inevitably involves a loss of descriptive information. The present method helps address this problem by allowing flexible experimentation with decisions around the labeling and annotation of data. This method, in combination with these digitized cartographic sources in particular, allows a wholly new approach to the fields of history and historical geography, framing questions it has not previously been possible to ask, for example, by thinking about visual typologies and suggesting abstract spatial relationships that do not yet exist in the literature.

In this study, we select annotation labels, or categories of historical information, and identify them at the patch, rather than pixel, level (see Fig. 1). In contrast to vector geometries for feature types that must be fixed at the outset of a project, patches are meaningful semantic units that are highly flexible both in nature (their size can vary) and as part of a workflow (one label can remain stable while others change or are tested iteratively). Patches can be annotated from one or more maps without the need to annotate a whole sheet. This paves the way for researchers to design annotation tasks which can allow inference on very large datasets, as compared, for example, to image segmentation methods. While well-known in computer vision (CV) research, we present patches as a novel structure for visual signals found on maps that can be linked to other visual or structured text data.

We demonstrate how MapReader (a) enables researchers to create and refine annotation schema and provides procedures for (b) digital annotation and (c) model training that ensure reliable out-of-sample generalization. The power of MapReader lies in its ability to analyze at high volumes the products of major mapping initiatives that are now available digitally around the world. Its interactive annotation tool can be used by researchers to label a large number of patches. MapReader provides functionalities to train or fine-tune various CV models using the annotated datasets. The main contributions of this paper are as follows:

- An open-source, end-to-end CV pipeline for the semantic exploration of high-volume map collections using the patch method (see Fig. 2).1
- New, expert-labeled training and evaluation datasets for the CV classification task, consisting of 62,020 human-annotated patches. These datasets and the CV outputs used in this study are described on MapReader’s GitHub repository2 and can be downloaded from Zenodo [24]3.
- First nation-wide analysis of historical maps of Britain. This task consists of classifying ≈30.5M patches from ≈16K maps.
- A rigorous analysis of visual annotation and model training for the semantic exploration of maps at high volumes. We focus on an initial interpretation of the results for a specific case study: the relationship between rail and the built environment in late nineteenth-century Britain.

2 RELATED WORK
Historical map processing is a growing subfield in GIScience (geographic information science) that has thus far combined the methods of image processing with the concerns of geographers. Using rule-based and now machine-learning methods, significant advances have been made in automating the otherwise tedious processes of generating vector and raster data from images [9, 10]. Researchers are working to improve methods for creating datasets of, for example, building footprints [20] and road and rail networks [12, 11, 35] using series maps comparable to OS maps.

Such research often builds on the lessons of working with remote sensing imagery or aerial photography to digitize map content, relying on the traditional spatial geometries of GIS to translate

1 The MapReader library is released under MIT License. Its source codes are on GitHub (https://github.com/Living-with-machines/MapReader). We have also provided Jupyter Notebooks on GitHub to reproduce the main results presented in this paper.
2 https://github.com/Living-with-machines/MapReader/wiki/GeoHumanities-workshop-in-SIGSPATIAL-2022
3 https://doi.org/10.5281/zenodo.7147906
MapReader is an end-to-end CV pipeline with two main components: preprocessing/annotation (top) and training/inference (bottom).

Figure 2: MapReader is an end-to-end CV pipeline with two main components: preprocessing/annotation (top) and training/inference (bottom).

Cartographic landscapes as machine-readable content. “Mining” maps in this way can produce useful data for studying, for example, historical settlement patterns and land uses [34, 36, 27]. New work in the Digital Humanities exploring the semantic segmentation of maps [32] contributes to general (e.g., cross-collection) solutions for automating the creation of vector data. Our method complements these ways of working with historical maps that are now available as scanned images, but it is also distinct in its approach to how we translate maps into data. MapReader is the first end-to-end pipeline for quickly asking questions of large map collections that is designed to be re-used by other researchers.

While historians and others have written extensively about the spatial character of British industrialization, many questions remain unanswered [8]. This is in part because the heterogeneity of historical sources makes it difficult to compile evidence at a high enough resolution to assess national trends in the built environment. OS maps have the potential to overcome this shortcoming and have been used by scholars to (manually) identify chimneys [19] and (automatically) extract land use type [6]. More often, however, maps have been used as figures to illustrate a specific point in a historian’s argument, rather than as sources of new information in themselves. Our approach differs from previous map processing work because we integrate the insights of scholars working with historical maps in a non-digital setting, including ideas of source criticism and data provenance, in this case in relation to the contested nature of map production itself [39, 17, 15, 2]. It is also our explicit ambition to bring historical maps into dialogue with other datasets, for example, texts and other sources of information about the past. Scholars working in the Digital Humanities have only recently embraced a “visual digital turn” [38], a welcome effort to reflect on the implications of working with visual data at scale. Arnold and Tilton introduced “distant viewing” as a methodology which critically interrogates the “interpretive nature of extracting semantic metadata from images” [5]. Our work builds on these observations, in particular, in the way that our interface enables scholars to fine-tune annotations and models iteratively—in dialogue with historical materials—so that users remain alert to the dangers of the semantic gap between the visual source and its textual label.

3 MAPREADER PIPELINE

Fig. 2 shows the main components of the MapReader pipeline. The first component, “preprocessing/annotation”, has functionalities to load locally stored images or to retrieve maps via webservers. Here, we use the National Library of Scotland (NLS) Historical Maps API to download ≈16K nineteenth-century OS map sheets and their metadata. The same interface can be used to download maps from other webservers (e.g., tiles based on the OpenStreetMap data). MapReader can then be used to preprocess the retrieved maps, such as resampling the images, removing borders outside the neatline or reprojecting the map.

In the next step, preprocessed map sheets are sliced into patches. The size of these patches can be specified by the number of pixels or by length in meters. For the latter, the geographic information (i.e., latitudes and longitudes) of the bounding box corners must be available in the map file or metadata. Determining the patch size is an important step in the MapReader pipeline: it should be large enough that the target label is presented in one patch with enough visual features to be identifiable by both the human annotator and the CV classifier; but it should be small enough to offer good spatial resolution.

MapReader provides a flexible and interactive annotation tool to label patches (Fig. 3). The tool’s flexibility lies in the simple options for controlling the interface. Its interface shows a patch to be annotated and, if selected by the user, a visual context around that patch. The user has control of the labels, the size of the context image and the selection criteria (by default, the patches are selected randomly, but they can be ordered by the mean or standard deviations of pixel intensities). We developed this interface based on feedback from researchers with different backgrounds, including historians, linguists, data scientists and software engineers. We have tested the usability of the annotation tool and its user experience in various tasks, all of which informed its design to support humanists asking questions of maps. MapReader’s annotation tool also supports reviewing model predictions and adding (expert) revised labels to the gold standard (see Section 5.4 for details on this review process). We used this approach to annotate 62,020 patches for our gold standard.

The second component, “training/inference” (Fig. 2, bottom), has functionalities to read in the annotations and to split them into three sets: train, validation and test. We use a stratified method for splitting the annotations, that is, each set contains approximately the same percentage of samples of each target label as the original set [31]. Next, the user defines transformations which will be used for data augmentation during training.

The “training/inference” component of MapReader is built upon torchvision and PyTorch [30] and has been tested on both CPU and GPU. A custom-defined CV model or a pretrained model can be used in this module. MapReader can also be used to build model ensembles. We discuss this functionality, “context-aware patchwork method”, in Section 5.4. The user can define (or use existing) optimizers, schedulers and optimization criteria. MapReader allows the user to “freeze” any layers in the model architecture when training or fine-tuning a model. One learning rate can be assigned to all layers, or MapReader can assign different learning rates to

---

Various pretrained models can be downloaded from torchvision or PyTorch Image Models [40].
We report on experiments to capture two kinds of physical structures; and (4) railspace and [non railspace] building. These metrics include training and validation losses, F1-scores (for each label separately as well as micro- and macro-averages), accuracy, precision and recall. MapReader provides several plotting tools to visualize and analyze the results of model inference. MapReader is available as a Python library. We provide extensive documentation – including examples for the main functionalities of MapReader in Jupyter Notebooks – to enable the smooth adoption of its components. All its functionalities, such as retrieving or loading maps, slicing them into patches, reading annotations, training, and fine-tuning and evaluating models, have easy-to-use interfaces. Consult Appendix A and MapReader’s GitHub page for additional information and examples.

4 EXPERIMENTS AND RESULTS

We report on experiments to capture two kinds of physical structures that are both important visual signals on these OS maps and key elements of the nineteenth-century British landscape: buildings and railway infrastructure. Any patch containing any building or any railway infrastructure is labeled as such. Selecting “building” and “railspace” as our two basic labels resulted in the following annotation possibilities: (1) no [building or railspace]; (2) railspace; (3) building; and (4) railspace and [non railspace] building. These two basic measures of development are an early demonstration of MapReader’s utility for identifying, collating, analyzing (quantitatively) and viewing (qualitatively) the content on thousands of maps. Combining these outputs, we piece together a “visual census” [22] of Britain both different from and also complementary to existing historical maps.

4.1 Dataset

To create the building and railspace datasets discussed below, we used the second edition of the OS six-inch-to-a-mile maps (a scale of 1:10,560). These sheets were surveyed between the late 1880s (for Scotland) and early 1890s (for England and Wales) and the beginning of World War I (first at six inches, and after 1893, at twenty-five inches to a mile) and printed (and re-printed, as revisions were made), at the scale of six inches between 1888 and 1913.5

The collection of six-inch 2nd edition sheets digitized by the NLS is missing only 15 sheets across Britain (about 0.1%): it is nearly a complete national set. The scanned sheets have been made available as georeferenced images with content outside the neatline masked. The NLS maintains a seamless layer for this edition, which we access via their tiles server.7

We downloaded 16,439 map sheets (∼600GB of storage) from the NLS tiles server, and sheet-level metadata was shared by the NLS (and can be accessed via MapReader’s GitHub repository). This metadata is essential to understanding the complex survey and print chronologies of this edition. In addition to these dates, metadata includes other identifying information for each sheet, whether a sheet is held by the NLS, whether it has been scanned, and more. After downloading the maps, we sliced each sheet into patches of 100m by 100m.8 This step generated ≈30.5M patches from 16,439 map sheets, and it took ≈32 hours on 6 cores.

4.2 Annotations

Railspace. We introduce the spatial concept of railspace as an element of the industrializing landscape related to understanding the impact of rail, not simply as a means of transport between two stations, but as a type of space. Railspace contains anything represented on the map related to the use of rail – including single and double tracks, stations, depts, and embankments (but excluding urban trams). This concept goes beyond existing datasets of railway tracks which are linear representations, or station datasets, which are points, to offer a more qualitative view of the spatial footprint of rail across Britain at its height in the late nineteenth century.

Buildings. Like railspace, the building label captures another basic piece of information about the historical British landscape to be found on maps. The label allows us to investigate the rapid increase of industrial, residential, civic, and commercial building stock. In the absence of open data about historical building locations, we developed this label as a coarse measure of the extent of buildings on the map.

Gold standard. The gold standard consists of 62,020 manually annotated patches from thirty maps across England, Wales, and Scotland and early 1890s (for England and Wales) and the beginning of World War I (first at six inches, and after 1893, at twenty-five inches to a mile) and printed (and re-printed, as revisions were made), at the scale of six inches between 1888 and 1913.5

The collection of six-inch 2nd edition sheets digitized by the NLS is missing only 15 sheets across Britain (about 0.1%): it is nearly a complete national set. The scanned sheets have been made available as georeferenced images with content outside the neatline masked. The NLS maintains a seamless layer for this edition, which we access via their tiles server.7

We downloaded 16,439 map sheets (∼600GB of storage) from the NLS tiles server, and sheet-level metadata was shared by the NLS (and can be accessed via MapReader’s GitHub repository). This metadata is essential to understanding the complex survey and print chronologies of this edition. In addition to these dates, metadata includes other identifying information for each sheet, whether a sheet is held by the NLS, whether it has been scanned, and more. After downloading the maps, we sliced each sheet into patches of 100m by 100m.8 This step generated ≈30.5M patches from 16,439 map sheets, and it took ≈32 hours on 6 cores.

4.2 Annotations

Railspace. We introduce the spatial concept of railspace as an element of the industrializing landscape related to understanding the impact of rail, not simply as a means of transport between two stations, but as a type of space. Railspace contains anything represented on the map related to the use of rail – including single and double tracks, stations, depts, and embankments (but excluding urban trams). This concept goes beyond existing datasets of railway tracks which are linear representations, or station datasets, which are points, to offer a more qualitative view of the spatial footprint of rail across Britain at its height in the late nineteenth century.

Buildings. Like railspace, the building label captures another basic piece of information about the historical British landscape to be found on maps. The label allows us to investigate the rapid increase of industrial, residential, civic, and commercial building stock. In the absence of open data about historical building locations, we developed this label as a coarse measure of the extent of buildings on the map.

Gold standard. The gold standard consists of 62,020 manually annotated patches from thirty maps across England, Wales, and Scotland and early 1890s (for England and Wales) and the beginning of World War I (first at six inches, and after 1893, at twenty-five inches to a mile) and printed (and re-printed, as revisions were made), at the scale of six inches between 1888 and 1913.5

The collection of six-inch 2nd edition sheets digitized by the NLS is missing only 15 sheets across Britain (about 0.1%): it is nearly a complete national set. The scanned sheets have been made available as georeferenced images with content outside the neatline masked. The NLS maintains a seamless layer for this edition, which we access via their tiles server.7

We downloaded 16,439 map sheets (∼600GB of storage) from the NLS tiles server, and sheet-level metadata was shared by the NLS (and can be accessed via MapReader’s GitHub repository). This metadata is essential to understanding the complex survey and print chronologies of this edition. In addition to these dates, metadata includes other identifying information for each sheet, whether a sheet is held by the NLS, whether it has been scanned, and more. After downloading the maps, we sliced each sheet into patches of 100m by 100m.8 This step generated ≈30.5M patches from 16,439 map sheets, and it took ≈32 hours on 6 cores.

4.2 Annotations

Railspace. We introduce the spatial concept of railspace as an element of the industrializing landscape related to understanding the impact of rail, not simply as a means of transport between two stations, but as a type of space. Railspace contains anything represented on the map related to the use of rail – including single and double tracks, stations, depts, and embankments (but excluding urban trams). This concept goes beyond existing datasets of railway tracks which are linear representations, or station datasets, which are points, to offer a more qualitative view of the spatial footprint of rail across Britain at its height in the late nineteenth century.

Buildings. Like railspace, the building label captures another basic piece of information about the historical British landscape to be found on maps. The label allows us to investigate the rapid increase of industrial, residential, civic, and commercial building stock. In the absence of open data about historical building locations, we developed this label as a coarse measure of the extent of buildings on the map.

Gold standard. The gold standard consists of 62,020 manually annotated patches from thirty maps across England, Wales, and

---

1The layer-wise learning rate was inspired by the implementation in fastai [25].

2Distinguishing between the 1st and 2nd edition maps can be artificial, but the vast majority of sheets known as 2nd edition sheets were prepared at the beginning of the 1880s.

3Details about this layer are available at https://maps.nls.uk/projects/subscription-api/#gb6inch.

4MapReader estimates the width and height of each pixel by using coordinates of the borders.
Scotland. These map sheets were selected to include different types of landscapes in the annotation set. We used an iterative approach to annotate these maps and their patches as described in Section 5.1 and “speeding up annotation” of Section 5.4.

In our experiments, we used 37,212 patches for training, 12,404 for validation and 12,404 for testing. The dataset is imbalanced with the following distribution: 56,372, 1,041, 3,634 and 973 patches for the four labels stated above, respectively. We used a stratified method for splitting the annotations into training, validation and test sets; that is, each set contains approximately the same percentage of samples of each target label as the original set. These datasets are described in detail on MapReader’s GitHub repository and can be downloaded from Zenodo [24].

4.3 Deep neural network classifiers

We fine-tuned various CV models on the training set of our gold standard with 37,212 labeled patches (see Section 4.2). Our multi-class classification task has four classes as follows: no building or railspace; railspace; building; and railspace and non railspace building. 11

We followed the same training procedure for all the model architectures listed in Table 1. Each patch has a size of 100 m². (The number of pixels varies depending on the size and zoom-level of a map sheet). Before feeding the patches to a model, several data transformations (i.e., augmentations) were applied, including horizontal and vertical flips, Gaussian blur and resizing to lower resolution images with 50 × 50 pixels. All of these transformations were applied randomly with a probability of 0.25. The patches were then normalized using the mean and standard deviations of pixel intensities in all the patches in our dataset, and they were resized based on the CV model and its required input size (e.g., 224×224 pixels in resnet or resnest models).

Due to the class imbalance in our gold standard, we employed a weighted sampler when generating training batches. This weighted sampler increases the number of minority classes and tries to form a balanced dataset. We used the Cross-Entropy criterion and the AdamW [29] optimization method with a linear, layer-wise learning rate ranging from 0.0001 to 0.001 from the first to last layers in a neural network architecture, respectively. In AdamW, we set β1 and β2 (i.e., the coefficients used for computing running averages of gradient and its square) to 0.9 and 0.999, respectively. We also used a scheduler to decay the learning rate of each parameter group by 0.1 (i.e., multiplicative factor of learning rate decay) every 5 epochs. We set the batch size to 32 and fine-tuned for 30 epochs over the training set.

For each neural network architecture, the model with the least validation loss was selected. We then compared the F1-macro scores of these selected models on the validation set with 12,404 labeled patches. (Refer to Appendix B and Table 2 for details.) Resnest101e had the highest F1-macro score of 96.74% on the validation set (and we used this model for all the results presented in this paper).

Finally, we measured the performance of the selected models on the test set. This step was done only once, and the results are reported in Table 1.

Some of our experiments were not used in the final training as they did not improve the performance significantly.12 These included changing the weights of the sampler from 10/(class_count) to 1/(class_count); switching off the scheduler; assigning learning rates using a geometric progression (i.e., spaced evenly on a log scale) from 0.00001 for the first layer to 0.001 for the last layer in the neural network architecture; and “freezing” all the layers except for the last during fine-tuning.

All the models in Table 1 were pretrained and further fine-tuned on our training set except resnest101e_no_pretrain. Comparing the two versions of resnest101e shows that the predictive performance was substantially improved by fine-tuning the pretrained model rather than training the model from scratch [21, 26].

Our selected model, resnest101e, was then used for model inference on all 30,490,411 patches from the ≈16k map sheets covering England, Wales and Scotland. In practice, this step is trivial to parallel. We distributed the patches on four NVIDIA Tesla K80 GPUs, and it took ≈172 GPU hours in total. This time includes model inference and some pixel-level calculations, such as mean and standard deviation pixel intensities at patch level. For model inference, we applied both normalization and resizing transformations as described above.

After performing model inference on all patches, one issue we discovered among our predictions was the presence of disconnected areas of railspace. Given the structure of rail networks, we concluded that such disconnected “islands” of rail are likely (though not always) false positives. To improve the quality of our predictions, we introduced a post-processing step that takes advantage of the structure inherent to other label predictions nearby.

To do this, we first computed the distance of a patch to its closest neighboring patch with the same predicted label. We used k-d tree to find neighbors and calculate distances efficiently [7]. For both railspace and buildings, we then removed those patches which had no other neighboring patches with the same label within a radius of 250 meters (see Fig. 1). This step removed 4,082 isolated patches (~0.8%) out of 487,360 total patches for railspace.13

5 DISCUSSION

This section addresses the challenges in designing a CV pipeline for the humanistic analysis of large map collections using computational methods, as well as the research opportunities that flow from the experiments reported in this study. We also discuss the limitations of our current method and avenues to address them.

5.1 Systematic errors

A major issue we observed in our experiments was that high test scores achieved on gold standard data masked poor performance outside of the annotated sample, due to unobserved biases in the data. We explain how overcoming this issue requires a combination

---

6https://github.com/Living-with-machines/MapReader/wiki/GeoHumanities-workshop-in-SIGSPATIAL-2022
7https://doi.org/10.5281/zenodo.7147906
8An alternative approach is to train two binary classifiers, one for railspace and one for buildings. We leave the assessment and comparison between these two formulations for the future.
9https://doi.org/10.5281/zenodo.7147906
10An alternative approach is to train two binary classifiers, one for railspace and one for buildings. We leave the assessment and comparison between these two formulations for the future.
11The raw and post-processed outputs can be downloaded from https://doi.org/10.5281/zenodo.7147906.
Table 1: Performance of CV classifiers on the test set with 12,404 labeled patches (20% of all manually annotated patches). We used pretrained models from PyTorch Image Models [40] and PyTorch [30]. The models are ordered according to F1-macro. The F1-score for each label is also listed. F1-0: no [building or railspace]; F1-1: railspace; F1-2: building; and F1-3: railspace and [non railspace] building. Time is for model inference on all 12,404 patches. Refer to Appendix B and Table 2 for the results on the validation set.

| Model                        | F1-macro | F1-micro | F1-0 | F1-1 | F1-2 | F1-3 | Time (m) | #params (M) |
|------------------------------|----------|----------|------|------|------|------|----------|-------------|
| resnest50d_4s2x40d [42]     | 97.35    | 99.55    | 99.81| 96.40| 97.41| 95.79| 11       | 28.4        |
| resnest101e [42]            | 97.17    | 99.55    | 99.83| 97.36| 97.20| 94.27| 8        | 46.2        |
| swsl_resnext101_32x8d [41]  | 96.28    | 99.38    | 99.76| 96.19| 96.13| 93.05| 19       | 86.8        |
| resnet152 [18]              | 96.19    | 99.25    | 99.67| 95.71| 95.17| 94.21| 10       | 58.2        |
| resnest101e_no_pretrain [42]| 91.12    | 98.12    | 99.13| 87.87| 88.65| 88.83| 12       | 46.2        |
| tf_efficientnet_b3_ns [33]  | 90.00    | 98.33    | 99.29| 88.29| 89.23| 83.19| 11       | 10.7        |
| swin_base_patch14 [28]      | 76.92    | 93.32    | 96.54| 64.52| 67.19| 79.43| 16       | 86.7        |
| vit_base_patch16_224 [14]   | 62.50    | 91.03    | 95.77| 37.93| 62.93| 53.37| 15       | 85.8        |

Figure 4: Systematic errors in predicting railspace (red dots). The Cuillin Hills (Isle of Skye, Scotland) contain an example of features drawn on the map using hatched rectangles. This is similar to how rail tracks are represented on the map. The initial, weak model predicted these as railspace (false positive). Map images courtesy of the NLS.

5.2 Combining MapReader outputs

MapReader is designed to make digitized map collections accessible as research data for the humanities. To demonstrate this, we are developing a case study on the distribution of rail infrastructure and buildings across Britain. Using the geographic coordinates of a map sheet provided in its metadata, we compute the center latitude and longitude of each patch. These can be used to plot results on a map (Fig. 1). However, the utility of the tool lies not only in locating objects: its ability to make use of multiple labels in combination enables scholars to construct entirely novel analytical concepts. For example, by combining predicted patches labeled as “buildings” and “railspace”, we explore built-up areas dominated by rail using the relation between patches to enhance the spatial semantics of our

buildings and sometimes railspace because of hachures that mimic both terraced housing and railway tracks. Lines like this were used prolifically by OS draughtsmen in the Highlands and along the cliffs of Britain’s undeveloped coasts. Other common errors after the first round of training involved coastal walls, reservoirs, drainage ditches, river or lake shores, and rocky areas predicted as railspace patches.

False negatives (e.g., missed railspace patches) included railways passing through forests and railways on certain Scottish sheets. Further systematic errors include false negatives for isolated farms and other buildings. We increased the dataset to include more sheets with a high percentage of rural space to improve accuracy. There are well-founded concerns among humanists (though few from historians so far) for making arguments based on inferred data, and, just as we know that there are many biases shaping archival collections, we know that we cannot account for all machine-introduced errors. However, given the high quality of our results, we find this limitation to be acceptable when working with such a large number of maps. Nonetheless, we will continue to assess the effect of errors on interpretation as we conduct new experiments.

Hosseini et al.

14 Other performance metrics of this model are F1-micro = 99.19%; F1-0 = 99.64%; F1-1 = 92.31%; F1-2 = 98.16%; F1-3 = 93.75%.

15 We used kepler.gl (https://github.com/keplergl/kepler.gl) for visualization and qualitative model inspection.

16 Less common full-size sheets in Scotland threw up more systematic errors than the more common quarter-size sheets in England and Wales. This motivated switching our definition of patch size from 100x100 pixels to ≈100m×100m: each patch now represented the same size area and this did indeed improve prediction of railspace and buildings across the areas of Scotland printed on full sheets.
Figure 5: Building patches for which at least 20% of neighboring patches are classified as railspace. Neighbors are defined as all patches within 500m of a building patch. Colour determined by quantiles (an equal number of patches in each color interval). To make the map legible, some have been merged in the visible colorbar. (The interval bounds, before being merged, were 20.0, 21.4, 23.2, 25.0, 27.5, 30.9, 37.0, 87.0%). Refer to Appendix D for other examples. Map images courtesy of the NLS.

Figure 6: Visual comparison between railspace as predicted by MapReader (red) and the location of railway stations from StopsGB (black) [3]. Map images courtesy of the NLS.

5.4 Patchwork method: advantages and limitations

Speeding up annotation. In addition to its basic functionality for annotation, MapReader provides a convenient way to extend the labeled dataset without requiring us to annotate all patches on a given sheet. First, we train a weak classifier with ≈7.8K manually annotated patches and use it for model inference on all map sheets (with ≈30.5M patches). As discussed in Section 5.1, this helps to identify systematic errors and biases. It is possible to inspect these results visually to extend the labeled dataset by sampling from regions where the weak model made the most mistakes. Moreover, instead of annotating all gold standard patches from scratch, annotators can review predictions of the initial, weak classifier in batches and add them to the gold standard. Because many areas of maps are blank or have simple visual features, these can be correctly identified even by a weak classifier.

Fig. 7 shows the results of fine-tuning resnest101e models progressively on more training instances. We followed the same procedure described in Section 4.3 to fine-tune pre-trained resnest101e models. A model fine-tuned on only 1000 patches has a F1-score of around (or even more than) 80%. This model can be used as the initial, “weak” classifier (see above). It should be noted that the performance of such a model varies depending on the class (see Fig. 7 bottom), for example, the F1-score of class 1 (i.e., railspace) is ≈60% while all the other F1-scores are more than 80%, with class 0 (i.e., “no”) being distinct. This label represents many different kinds of visual signals on the maps, from forests to farmland. Because of this
Figure 7: Performance of resnest101e models as a function of training instances used in fine-tuning. These multiclass classifiers have four classes (see Table 1 for the list of classes). Here, we used the same number of training instances in each class (e.g., in a model with 1000 training patches, each class has 250 patches). The F1-macros of resnest101e models fine-tuned progressively on more training instances are compared in the box-and-whisker plot (top). The sample median and the first and third quartiles are computed from 25 independent runs. The horizontal, dashed blue line indicates the F1-macro of our selected resnest101e model used in this paper (fine-tuned on 37,212 patches). In the bottom figure, the F1-score of each class is compared. We only show the sample median of 25 independent runs. See Table 1 for more details on the class names and the F1-score of each class in our selected resnest101e model.

heterogeneity, it is more difficult to assess whether a small number of labeled sample patches will actually be representative of the class. Such a label is likely to be common in similar experiments seeking to divide map content into classes reflecting a given concept and its absence: understanding variety within labels can inform how many annotations we collect and how we interpret the final output.

Context-aware patchwork method. A limitation of the patchwork method is that it ignores neighboring patches when training or performing model inference. To capture the strong, spatial dependencies between neighboring patches (which will be common in maps for many types of labels [16]), MapReader supports building model ensembles as shown in Fig. 8. Model-1 and Model-2 are two CV models with neural network architectures defined by the user. These models can have different or similar architectures. For example, in one of our experiments, Model-2 was a pre-trained model on our patch dataset while Model-1 was a pre-trained model from PyTorch Image Models [40] with a different architecture. As shown in Fig. 8, a patch and its context image are fed into Model-2 and Model-1, respectively. In practice, the user only specifies the size of the context image, and MapReader extracts and preprocesses the context image from the dataset. Model-1 and Model-2 generate vector representations for the input images, $V_1$ and $V_2$. The size of these vectors are defined by the user and a combination of the two (e.g., by concatenation) is then fed into Model-3 for prediction. Such model ensembles can be an efficient approach to achieving high-performing CV models [37]. As described above, MapReader has the functionality to build model ensembles, but we leave assessment of their usability to the future.

Image segmentation. We implemented the patchwork method instead of image segmentation for transforming visual features on maps into usable data for two reasons. First, it is fast and easy to create large training and evaluation datasets of patches, one of the pre-requisites for high-quality predictive CV models. The resulting trained models can be applied to large map datasets within reasonable computation time (e.g., it took $\approx$172 GPU hours to do model inference and some pixel-level calculations on all the $\approx$30.5M patches used here). For image segmentation, hand-labeling of each pixel is tedious compared to patch-level labeling. Second, patches are meaningful semantic units on maps. Highly flexible in size, patches can be sized according to both the size of the visual feature of interest and the accuracy of the map. Patches prevent the researcher from unintentionally producing overly-precise information about historical landscapes documented in poorly surveyed or unprojected maps. Nevertheless, pixel-level segmentation does
have advantages: the results are not bounded to a specific patch size, and some patch-level outputs could be reconstructed from the pixel-level information. However, it is not clear how many map sheets would need to be annotated for an image segmentation method to perform similarly to our current models.

**Patch sizing.** We considered two factors in selecting patch size for our experiments (described in Section 3). In practice, patch sizing is an important step that can significantly affect the model’s performance. In future research, we plan to automate the selection of an optimal-sized patch for a given map type label. Moreover, as the information density on maps is highly variable, we plan to assess the usability of adaptive patch sizing. In this case, the local patch size would be determined according to the expected information content (in contrast with the global or static patch size used in this study).

6 CONCLUSIONS

We presented MapReader, a free, open-source software library written in Python. MapReader allows users with little CV expertise to work with large collections of maps. We presented a case study focusing on British rail infrastructure and buildings as depicted in a collection of ~16K nineteenth-century British OS maps (~30.5M patches), demonstrating how patches on their own and in combination with other patches or external datasets can offer new insights into the spatial patterns of industrial development in modern Britain. Using an image classification task at patch level transforms a common, indeed unsophisticated, CV method into a radically new way for historians and others to interact with maps. Our analysis of the national landscape in terms of spatial attributes like railspace or building density, sets the stage for new forms of CV-driven research. Such future work could make use of extensive new historical datasets to identify spatial patterns, as well as discovering changes to the built and natural environments, for example, by relating information drawn from maps to that being generated by modern sources, such as remote sensing data. To this end, we open-source ~62K expertly annotated patches to foster further collaboration between the fields of CV, Machine Learning and History, as well as with libraries and archives.

ACKNOWLEDGMENTS

The map images and metadata used in this research are provided by the National Library of Scotland. The data is available for non-commercial re-use under CC BY-NC-SA 4.0 licence (see https://maps.nls.uk/copyright.html#exceptions-os). We thank Ruth Ahnert, Mariona Coll Ardanuy, Jon Lawrence and Joshua Rhodes for discussions and feedback. This work was supported by Living with Machines (AHRC grant AH/S01179X/1) and The Alan Turing Institute (EPSRC grant EP/N510129/1). Living with Machines, funded by the UK Research and Innovation (UKRI) Strategic Priority Fund, is a multidisciplinary collaboration delivered by the Arts and Humanities Research Council (AHRC), with The Alan Turing Institute, the British Library and the Universities of Cambridge, East Anglia, Exeter, and Queen Mary University of London.

REFERENCES

[1] Martin Abadi et al. 2016. Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1605.08467.
[2] James R. Akerman. 2017. Decolonizing the map: Cartography from colony to nation. University of Chicago Press.
[3] Mariona Coll Ardanuy, Kaspar Beelen, Jon Lawrence, Katherine McDonough, Federico Nanni, Joshua Rhodes, Giorgia Tofo, and Daniel CS Wilson. 2021. Station to station: linking and enriching historical british railway data. Proceedings http://ceur-ws.org, ISSN: 1613, 0073.
[4] Mariona Coll Ardanuy, Kasra Hosseini, Katherine McDonough, Amrey Krause, Daniel van Strien, and Federico Nanni. 2020. A deep learning approach to geographical candidate selection through toponym matching. In Proceedings of the 28th International Conference on Advances in Geographic Information Systems, 385–388.
[5] Taylor Arnold and Lauren Tilton. 2019. Distant viewing: analyzing large visual corpora. Digital Scholarship in the Humanities, 34, Supplement_1, 0–116.
[6] Brian Baily. 2007. The extraction of digital vector data from historic land use maps of great britain using image processing techniques. E-perimetro, 2, 4, 209–223.
[7] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM, 18, 9, 509–517.
[8] Dan Bogart. 2014. The transport revolution in industrialising britain. The Cambridge economic history of modern Britain, 1, 386–391.
[9] Benedikt Budig. 2018. Exploring spatial information from historical maps: algorithms and interaction. BoD–Books on Demand.
[10] Yao-Yi Chiang, Wewei Duan, Stefan Leyk, Johannes H Uhl, and Craig A Knoblock. 2020. Historical map applications and processing technologies. In Using Historical Maps in Scientific Studies. Springer, 9–36.
[11] Yao-Yi Chiang, Wewei Duan, Stefan Leyk, Johannes H Uhl, and Craig A Knoblock. 2020. Training deep learning models for geographic feature recognition from historical maps. In Using historical maps in scientific studies. Springer, 65–98.
[12] Yao-Yi Chiang and Craig A Knoblock. 2010. Strabo: a system for extracting road vector data from raster maps. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, 544–545.
[13] Nan Z Da. 2019. The computational case against computational literary studies. Critical inquiry, 45, 3, 601–639.
[14] Alexey Donovskiy et al. 2020. An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2001.12988.
[15] Matthew H Edney. 2019. Cartography: The ideal and its history. University of Chicago Press.
[16] Manfred M. Fischer and Arthur Getis. (Eds.) 2010. Spatial autocorrelation. Handbook of Applied Spatial Analysis: Software Tools, Methods and Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, 255–278. issn: 978-3-642-03647-7. doi: 10.1007/978-3-642-03647-7_14.
[17] J Brian Harley. 1988. Silences and secrecy: the hidden agenda of cartography in early modern europe. Image mundi, 40, 1, 57–76.
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778.
[19] Stephan Hoblich, Alex Trew, and Yanos Zydidberg. 2021. East-side story: historical pollution and persistent neighborhood sorting. Journal of Political Economy, 129, 5, 1508–1552.
[20] Robert Hecht, Hendrik Herold, Martin Behnisch, and Mathias Jehling. 2018. Mapping long-term dynamics of population and dwellings based on a multi-temporal analysis of urban morphologies. ISPRS International Journal of Geo-Information, 8, 1, 2.
[21] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. Science, 313, 5786, 504–507.
[22] Kasra Hosseini, Katherine McDonough, Daniel van Strien, Olivia Vane, and Daniel CS Wilson. 2021. Maps of a nation? the digitized ordnance survey for new historical research. Journal of Victorian Culture, 26, 2, 284–299.
[23] Kasra Hosseini, Federico Nanni, and Mariona Coll Ardanuy. 2020. Deeyonmatch: a flexible deep learning approach to fuzzy string matching. In Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations, 62–69.
[24] Kasra Hosseini, Daniel C.S. Wilson, Kaspar Beelen, and Katherine McDonough. Mapreader_data_sigspatial_2022. Version v0.3.3. Zenodo, (Oct. 2022). doi: 10.5281/zenodo.7147906.
[25] Jeremy Howard and Sylvain Gugger. 2020. Fastai: a layered api for deep learning. Information, 11, 2, 108.
[26] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. Journal of machine learning research, 10, 1.
[27] Stefan Leyk, Johannes H Uhl, Dylan S Connor, Anna E Braswell, Nathan Mietkiewicz, Jennifer K Balch, and Myron Gutmann. 2020. Two centuries of
settlement and urban development in the United States. *Science Advances*, 6, 23, eaba2937.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zhong Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

[29] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[30] Adam Paszke et al. 2019. Pytorch: An imperative style, high-performance deep learning library. (2019). *arXiv: 1912.01703 [cs.LG]*.

[31] F. Pedregosa et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[32] Rémi Petitpierre, Frédéric Kaplan, and Isabella di Lenardo. 2021. Generic semantic segmentation of historical maps. *Proceedings* http://ceur-ws.org ISSN, 1613, 0073.

[33] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, PMLR, 6105–6114.

[34] Johannes H Uhl, Dylan S Connor, Stefan Leyk, and Anna E Braswell. 2020. Urban spatial development in the United States from 1910 to 2010: A novel data-driven perspective. *Available at SSRN 3537768*.

[35] Johannes H Uhl, Stefan Leyk, Yao-Yi Chiang, and Craig A Knoblock. 2022. Towards the automated large-scale reconstruction of post road networks from historical maps. *Computers, Environment and Urban Systems*, 94, 101794.

[36] Johannes H Uhl, Stefan Leyk, Caitlin M McShane, Anna E Braswell, Dylan S Connor, and Deborah Balk. 2021. Fine-grained, spatiotemporal datasets measuring 200 years of land development in the United States. *Earth system science data*, 13, 1, 119–153.

[37] Xiaofang Wang, Dan Kondratyuk, Eric Christiansen, Kris M Kitani, Yair Alon, and Elad Eban. 2020. Wisdom of committees: An overlooked approach to faster and more accurate models. *arXiv preprint arXiv:2012.01988*.

[38] Melvin Wevers and Thomas Smits. 2020. The visual digital turn: Using neural networks to study historical images. *Digital Scholarship in the Humanities*, 35, 1, 194–207.

[39] Karen Wigen and Caroline Winterer. 2020. *Time in maps: from the Age of Discovery to our digital era*. University of Chicago Press.

[40] Ross Wightman. 2019. Pytorch image models. https://github.com/rwightman/pytorch-image-models. (2019). doi: 10.5281/zenodo.4414861.

[41] I Zeki Valinz, Hervé Jegou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. 2019. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*.

[42] Hang Zhang et al. 2022. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2736–2746.
A MAPREADER INTERFACE

MapReader is an end-to-end CV pipeline, and it is available as a Python library. As an example, loading maps, slicing them into patches of size $50 \times 50$ pixels and plotting four sample results can be executed by:

```python
from mapreader import loader
mymaps = loader(path_images)
# "method" can be pixel or meters
mymaps.sliceAll(path_save=path_save_patches,
                 slice_size=50,
                 method="pixel")
mymaps.show_sample(4)
```

and initializing a CV classifier object using a pretrained resnet152 model and showing the details of each layer (i.e., the number of trainable weights and biases, the name of each layer as specified in a model and the total number of trainable and "frozen" parameters) can be initiated with:

```python
from mapreader import classifier
myclassifier = classifier()
myclassifier.initialize_model("resnet152",
                         pretrained=True)
myclassifier.model_summary(only_trainable=False)
```

Other functionalities, such as retrieving maps from webservers, reading annotations, training, fine-tuning and model evaluation have similar easy-to-use interfaces. Consult MapReader’s GitHub page for additional information and examples.

B MODEL PERFORMANCE ON VALIDATION SET

We used 37,212 patches for training (60% of the total number of labeled patches), 12,404 (20%) for validation and 12,404 (20%) for testing the model performance. For each neural network architecture, the model with the least validation loss was selected. Table 2 lists the performance of these models as measured by F1 scores (macro, micro and for each label separately). Based on these results, we selected resnest101e model for the results of this paper. See Table 1 for the performance scores of these models on the test set.

C LINKING AND ENRICHING RAILSPACE DATA

StopsGB [3] is a structured dataset of over 12,000 railway stations based on Michael Quick’s reference work “Railway Passenger Stations in Great Britain: a Chronology”. In [3], the authors used traditional parsing techniques to convert the original document into a structured dataset of stations, with attributes containing information such as operating companies and opening and closing dates. Moreover, for each station, a set of potential Wikidata candidates was identified using DeezyMatch [23, 4]. The best matching entity was then determined by using a supervised classification approach. Because most stations are linked to the best Wikidata candidate, this is a high-quality source for location data for historical British railway stations, an integral element within the class of railspace.

Fig. 6 shows a visual comparison between StopsGB and railspace as predicted by MapReader. 89.3% of StopsGB stations were within 150 meters of the center of a railspace patch (here, each patch has a size of 100 m²).

D CLASSIFICATION OF BUILDINGS BY THEIR NEIGHBORING PATCHES

MapReader produces data at patch level. However, the context around each patch can be used to enhance the spatial semantics of our analysis. We do this by identifying space in terms of the relation between (and not just the presence of) our labels (as well as other external datasets, like StopsGB). In Figs 5, 9 and 10, we show how building and railspace patches relate to each other. Such a perspective (shown here for different regions and zoom levels) enables historians to search for places and compare them along new axes; for instance in this example, by identifying (first visually and then quantitatively) those areas that share certain features of London’s railspace, expressed as the relation between given classes (such as “building density” and “distance from a station”). In this way, MapReader can help researchers go beyond the existing literature in their fields: on the one hand, by exploring the extent to which well-known case studies are in fact representative of wider datasets; and on the other hand, to help them identify entirely new areas of interest not previously considered, using visual semantic patterns as their guide.
Table 2: Performance of CV classifiers on the validation set with 12,404 labeled patches (20% of all manually annotated patches). We used pretrained models from *PyTorch Image Models* [40] and *PyTorch* [30]. The models are ordered according to F1-macro. The F1-score for each label is also listed. F1-0: no [building or railspace]; F1-1: railspace; F1-2: building; and F1-3: railspace and [non railspace] building. Time is for both training and validation.

| Model                    | F1-macro | F1-micro | F1-0       | F1-1       | F1-2       | F1-3       | Time (m) / epoch | #params (M) |
|--------------------------|----------|----------|------------|------------|------------|------------|------------------|-------------|
| resnest101e [42]         | 96.74    | 99.59    | 99.88      | 95.59      | 97.75      | 93.75      | 98.2             | 46.2        |
| tf_efficientnet_b3_ns [33]| 96.50    | 99.52    | 99.84      | 95.67      | 97.12      | 93.37      | 89.7             | 10.7        |
| resnest50d_4s2x40d       | 96.47    | 99.42    | 99.77      | 95.17      | 96.48      | 94.46      | 86.6             | 28.4        |
| swsl_resnext101_32x8d [41]| 96.33    | 99.42    | 99.79      | 95.67      | 96.39      | 93.47      | 130.8            | 86.8        |
| resnet152 [18]           | 95.80    | 99.29    | 99.73      | 95.67      | 95.30      | 92.51      | 94.9             | 58.2        |
| resnest101e_no_pretrain  | 90.67    | 98.23    | 99.25      | 85.97      | 89.47      | 88.00      | 98.3             | 46.2        |
| swin_base_patch4 [28]    | 77.05    | 93.39    | 96.59      | 64.54      | 68.02      | 79.04      | 109.9            | 86.7        |
| vit_base_patch16_224 [14]| 64.78    | 91.43    | 95.93      | 43.38      | 64.61      | 55.21      | 99.6             | 85.8        |

Figure 10: Similar to Fig. 5 except that it shows results for greater Leeds. Map images courtesy of the NLS.