You Only Look for a Symbol Once: An Object Detector for Symbols and Regions in Documents

William A. P. Smith and Toby Pillatt

University of York, York, UK
{william.smith,toby.pillatt}@york.ac.uk

Abstract. We present YOLSO, a single stage object detector specialised for the detection of fixed size, non-uniform (e.g. hand-drawn or stamped) symbols in maps and other historical documents. Like YOLO, a single convolutional neural network predicts class probabilities and bounding boxes over a grid that exploits context surrounding an object of interest. However, our specialised approach differs from YOLO in several ways. We can assume symbols of a fixed scale and so need only predict bounding box centres, not dimensions. We can design the grid size and receptive field of a grid cell to be appropriate to the known scale of the symbols. Since maps have no meaningful boundary, we use a fully convolutional architecture applicable to any resolution and avoid introducing unwanted boundary dependency by using no padding. We extend the method to also perform coarse segmentation of regions indicated by symbols using the same single architecture. We evaluate our approach on the task of detecting symbols denoting free-standing trees and wooded regions in first edition Ordnance Survey maps and make the corresponding dataset as well as our implementation publicly available.

Keywords: non-uniform symbol detection · object detection · map digitisation.

1 Introduction

Historical maps and documents often contain an array of symbols and markings. Recognising and localising these symbols is useful for many reasons, including digitisation of historic documents and georeferencing of map contents. However, while the symbols may be (part-)standardised, to modern eyes (and computer vision) they are non-uniform, being either hand-drawn or reproduced inconsistently. For example, although draftsmen and printers often used stamps to speed up reproduction and adhere to drawing conventions, it was not uncommon for different workers to use different stamps for the same symbol, introducing both subtle and stark variations between maps and documents. Even for the same draftsman using a single stamp, small disparities in the way the stamp was applied or in the other processes of printmaking—during etching, engraving or inking, for example—could result in subtle differences in symbol appearance.

This could be the case even within a single map sheet or page. A more contemporary problem is that the digitisation of an original printed map or document can introduce distortions and corruptions that result in further variability, increasing difficulty of automated detection and all but ruling out simple template matching. Furthermore, unlike handwriting recognition, where the symbols are limited to alphanumeric characters and where words and sentences provide strong context to help disambiguate characters, more general symbols can be much more varied, may have far fewer examples per class, have less constraining local context and can partially overlap with each other.

Object detection is a widely researched topic with many high performing methods based on CNNs [9,26] or, very recently, transformer-based architectures [34]. However, we argue that the task of non-uniform symbol detection is subtly different to general object recognition in photos and that significant performance gains can be achieved by specifically tuning an architecture to exploit these differences. Concretely, the main differences are:

1. **Scale**: Symbols are drawn at a fixed scale so we do not need to predict bounding box dimensions. Instead, it is enough to predict the box centre and symbol class. We can then use a fixed size bounding box for each class.
2. **Regions**: Some symbols indicate that a bounded area forms a region of a particular class (e.g. woodland or orchard), rather than indicating a precise spatial location with particular meaning. This means that the symbol detection problem also encompasses a partial semantic segmentation problem.
3. **Context**: Local context is important. For example, in the tree detection task in our experiments, useful local context includes knowing that trees cannot appear on a road or inside a building or that trees commonly grow at the boundaries of fields. For region segmentation, sometimes a segmentation decision must be based only on local context when a grid cell contains no symbols indicating it is inside a region. However, beyond a certain locality, context becomes unimportant so we do not want to consider global context.
4. **Boundaries and absolute position**: The absolute position of a grid cell conveys no information about whether that cell contains a particular symbol. Therefore we do not want our network to learn spatial reasoning, for example from distance to boundary.
5. **Large scale inference**: Map and document datasets can easily scale to extreme size. For example, at the resolution we work at in our experiments, a national scale map would be of terrapixel size. Therefore, efficient methods are required that can process very large volumes of data.
6. **Symbol sparsity**: Since symbol occurrence is relatively sparse within a map, hard negative mining and weighting is essential for good performance.

In this paper, we propose a method that we call 'You Only Look for a Symbol Once' (YOLSO) that takes inspiration from the single-shot detector methods that build upon YOLO [26]. It is specifically adapted to exploit the properties and tackle the challenges of the symbol detection problem given above. We apply our method to a new dataset of first edition Ordnance Survey maps for the task of detecting symbols denoting free-standing trees and wooded regions. We show

that the specialisation of YOLSO to this task leads to significant performance improvements compared to applying generic object detection pipelines to the task of map symbol detection.

2 Related work

Object detection Object detection has benefited significantly from developments in deep learning and we only mention learning-based methods here. Early attempts to apply deep neural networks to the problem of object detection separated the task into two phases: object proposal and object recognition. Region-based CNN (R-CNN) [9] used Selective Search [31] to propose regions of interest (ROIs). Each ROI is then cropped and nonuniformly resized to a fixed size for input to a feature extraction CNN. These features are then passed to a support vector machine for classification as either background or one of a set of possible object classes. Since there may be thousands of object proposals per image, this method is extremely slow. Fast R-CNN [8] partially addressed this by passing the input image through the feature extraction network only once and then cropping and resizing the ROI features from the feature volume. Faster R-CNN [29] further improved speed and unified the object proposal and recognition stages by using the same network for both ROI generation and feature extraction. Since all of these methods pass cropped input to the object classifier, they cannot exploit wider context within the image. Classifying each region independently also remains a performance bottleneck.

"You Only Look Once" (YOLO) [26] was the first of what are now called *single shot detectors*. With a single forward pass through a CNN, these methods tackle the whole object detection problem in one go. The underlying idea is to discretise the output space of possible bounding boxes to one or more grids, each cell of which can make a fixed number of bounding box predictions. This eliminates the need for object proposals entirely, shares extraction of overlapping features between grid cells for efficiency and can exploit context according to the receptive field of the output grid cell. YOLO used a network that included fully connected layers. This enables it to exploit global context but this is unhelpful for symbol detection where useful context is relatively local to a symbol. The Single Shot Detector (SSD) [21] used multiscale grids for detection of objects at different sizes, used anchor boxes so that each scale has a notion of default box shapes and sizes, and replaced the fully connected layers with a fully convolutional architecture. YOLOv2 [27] and YOLOv3 [28] also switched to a fully convolutional architecture and incorporated many minor improvements that increased performance while retaining the same underlying approach. All of these approaches use architectures such that the receptive field of each grid cell is very large (potentially larger than the image) and therefore includes a lot of content outside the image which must be filled with padding. This introduces a boundary dependency such that the networks can exploit positional context (for example, learning that a face often occurs near the centre of the image). This is unhelpful for symbol detection in documents without meaningful boundaries. In

addition, the very large receptive fields make recognition and precise localisation of small symbols very challenging. Our approach is inspired by these single shot techniques but is specialised to the problem of symbol detection.

Non-uniform symbol detection Some of the earliest and best known work in CNN-based computer vision, e.g. LeCun et al. [19] tackled the problem of handwritten character *recognition* (i.e. where the symbol has already been located and cropped from the document). There have been relatively few attempts to tackle the specific problem of non-uniform symbol detection within documents and the majority of these applied existing methods without adaptation. Early work was based on classical template matching combined with multiple classifiers and explicit handling of rotation invariance [16]. Julca-Aguilar and Hirata [15] address the symbol detection problem in the context of online handwriting recognition using Faster R-CNN. Elyan et al. [6] apply YOLO to the problem of symbol detection in engineering drawings. In both cases, they use fixed sized small crops from the overall document such that the symbols are relatively large within the crop. Adorno et al. [1] take a different approach, treating the problem as one of dense semantic segmentation and post-processing to extract symbol bounding boxes. Recently, attention-based architectures have been applied to symbol detection, for example for recognising mathematical expressions [20].

Historic maps Historic maps are important sources for reconstructing social and landscape history. As collections have been digitised access has been significantly improved, but the conversion of scanned map rasters to labelled vector data remains a laborious task when performed manually. Early attempts at map vectorisation used image processing techniques [2,4] but more recently deep learning has been applied to both object detection and segmentation tasks. For example, Maxwell et al. [23] use a UNet architecture for semantic segmentation to identify a single feature class - historic mine extents - from a standardised series of polychrome topographic maps, while Petitpierre [25] experiments with using UNet architecture with a ResNet encoder for multiclass segmentation across whole corpuses of cadastral maps produced to different styles and conventions, and Garcia-Molsosa et al. [7] use a variety of detectors built on the UNet-based Picterra[1] platform to identify and segment the different, non-uniform symbols used to denote archaeological sites on multiple early twentieth-century colonial map series. Perhaps ostensibly most similar to the approach we describe is Groom et al.'s [10] use of CNN symbol detection to classify segments produced using colour-based image object analyses. However, unlike our method, this is predicated on polychrome maps and is not reliant on particularly high object detection accuracy.

Across the above examples, it is notable that while computer vision has developed markedly in recent years, the richness of map data, overlapping features, and imprecise georeferencing have continued to complicate analyses. Moreover, a particular limitation of deep learning models is the significant time and resource

[1] https://picterra.ch/

investment required to train them. Consequently, an ongoing strand of research, pioneered by the 'Linked Maps' project[2] [5], is on ways of creating training data from more recent mapping, sometimes by directly applying modern labelling to corresponding regions on old maps [30], or sometimes with intermediary steps, such as by using Generative Adversarial Networks to produce synthetic maps images that imitate historical map styles but preserve spatial patterning and labelling from modern maps [33]. While it is conceivable that some applications of YOLSO would benefit from such an approach, for our example the Ordnance Survey stopped mapping individual trees in the early twentieth century meaning there is no direct analogue that can be used to generate training data.

The need to invest time and resources in training means that deep learning tasks tend to be targeted towards big datasets, where any efficiencies to gained from such an approach are realised. When working with maps, this generally means working across multiple mapsheets encompassing large areas. The Living With Machines[3] project recently developed a series of useful tools for working with very large collections of historical Ordnance Survey maps [12,11]. Instead of a pixel-level segmentation model, they train a binary classification model using labelled 'patches' that can be of any size, enabling them to explore urbanisation and railway expansion across the whole of nineteenth-century Great Britain. However, the resolution of the analysis is determined by the patch size, and multiple iterations of labelling, training and analysis are required if one needs to drill down to examine groups of features at different resolutions. Conversely, an advantage of our method is that you only look for a symbol once.

Tree mapping Detailed and accurate data concerning the location of trees are an important prerequisite of tree risk management, and form the basis of economic and ecosystem service-based valuations [22]. A significant area of research is in using airborne remote sensing, including through LiDAR, photogrammetry and multispectral image analysis, to map urban and managed forest environments. These can be combined with computer vision analyses of street-level imagery, such as from Google Street View, to produce detailed tree maps [18,3]. The fact that many of these types of dataset are now available spanning multiple epochs opens up the prospect of studies of treescape change over time. However, where trees were in the more distant past is understood only in very broad terms. This is in spite of the fact that trees were often routinely plotted on old maps and plans, reflecting their value as sources of food, fuel and materials [32]. Identifying and geolocating trees on historical OS maps will present a baseline from which to explore changes to Britain's treed environment throughout the 20th century.

3 Method

The YOLSO architecture is a CNN that outputs a grid, in which each cell contains the symbol detection result for a corresponding region in the input image

[2] https://usc-isi-i2.github.io/linked-maps/
[3] https://livingwithmachines.ac.uk/

(see Figure 1). The network is fully convolutional, meaning that we do not need to choose and fix the dimensions or aspect ratio of the input image and the output grid dimension is determined by the input image dimensions. The entire symbol detection result is computed with a single forward pass through a small CNN, meaning it is computationally efficient and therefore applicable to very large datasets such as national-scale maps.

3.1 Architecture

The network is formed of blocks comprising: a convolution layer with no padding, ReLU activation, batch normalisation [13] and max pooling. The input image size, grid cell resolution, layer hyperparameters and number of blocks must be carefully chosen to ensure appropriately sized output.

For an input of spatial dimension $H \times W$, the output is a grid of spatial dimension $S_H \times S_W$ where $S_H = (H - 2P)/R$ and $S_W = (W - 2P)/R$. Each output grid cell contains the symbol detection result for the corresponding $R \times R$ pixel region in the input image. This means that R must be a factor of both $H - 2P$ and $W - 2P$. The P pixel boundary is required to ensure all grid cells can benefit from local context and to avoid using padding in the convolutional layers which would introduce unwanted boundary dependency (see below).

We use max pooling layers with 2×2 kernel for downsampling. Hence, to reduce an $R \times R$ region to a 1×1 grid cell, our network must contain $p = \log_2 R$ pooling layers and hence the grid cell resolution must be a power of 2, $R = 2^p$.

3.2 Padding and receptive field

We use no padding in our convolutional layers for two reasons. First, padding provides a network with a means to learn boundary dependence [17,14]. Since we do not expect symbols to be any more likely to appear in any particular location within a random map crop, positional dependence is undesirable. Second, padding introduces incorrect local context. Zero or replication padding introduces featureless regions. Reflection padding introduces reversed symbols that will not occur in real data.

Using no padding means that each convolution layer reduces the spatial size of the feature map. To compensate for this, we include a P pixel boundary in the input image. Each output grid cell has a receptive field in the original image of size $2P + R \times 2P + R$ (see Figure 1, red dashed square) and this boundary region ensures that the receptive field of all output grid cells lies within the image.

The appropriate value for P depends on the parameters of the convolutional layers and the number of pooling layers. Suppose that the parameters of the convolution layers are stored in the set $\mathcal{L}$. Each layer comprises the pair (k, d) where $d \in \{0, \ldots, p\}$ is the depth, i.e. the number of pooling layers that precede it, and k is the kernel size. The required padding can be computed as:

$$P = \sum_{(k,d) \in \mathcal{L}} \frac{k-1}{2} \cdot 2^d. \tag{1}$$

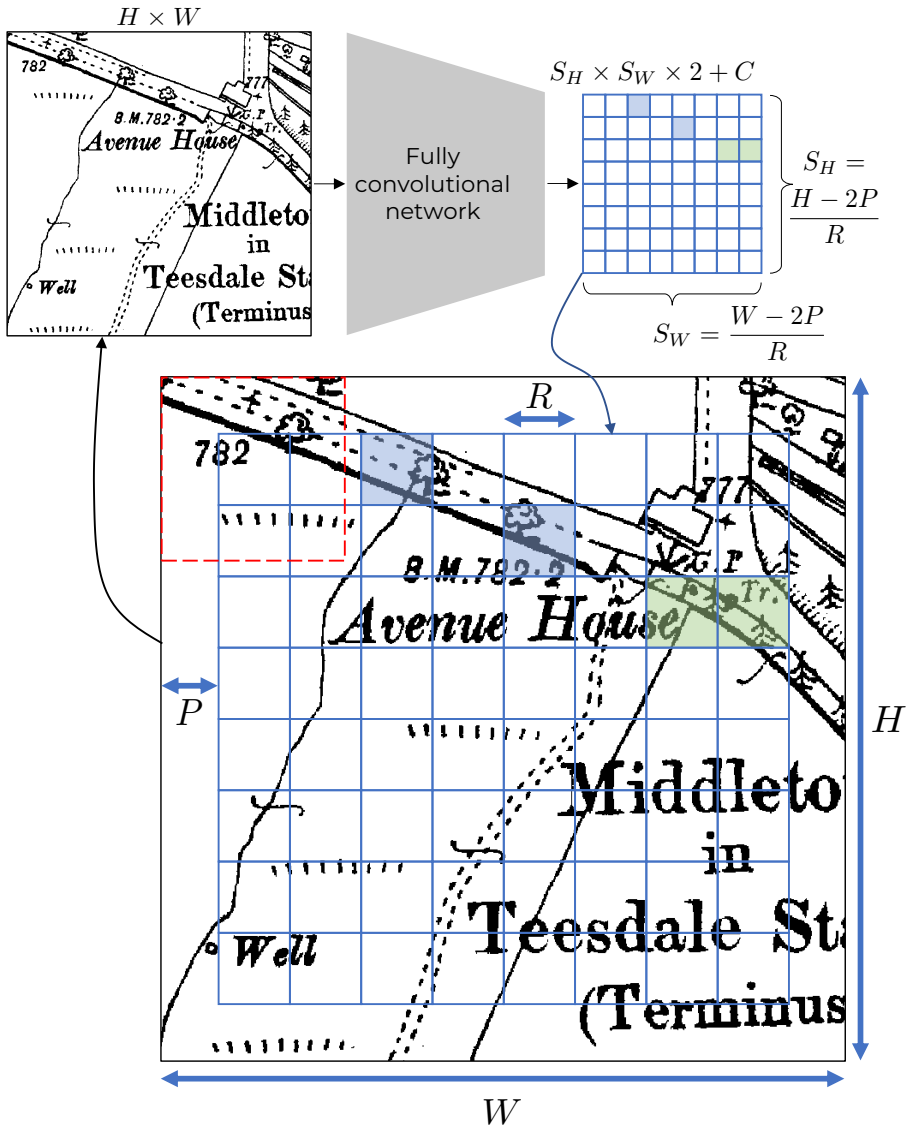

Fig. 1. YOLSO is a fully convolutional network that takes as input an $H \times W$ image and outputs a $(H-2P)/R \times (W-2P)/R$ grid. Each cell in the output grid indicates whether the corresponding $R \times R$ region in the input image contains a symbol (class indicated by coloured fill) and, if so, the offset of the bounding box centre. The convolution layers use no padding and, hence, the receptive field of each grid cell (shown in red for the top left cell) is contained entirely within the original input image. To enable this, the input image must include a border of width P which ensures all output grid cells benefit from local context and no boundary dependency is introduced.

At training time, it is important that the boundary region contains typical content, such that the local context around a grid cell is meaningful. For this reason, we use $(H-2P)/R \times (W-2P)/R$ crops from the training images. At inference time, in order to detect symbols for the whole of any input image of arbitrary size, we pad images by P on the left and top, $P+T_W$ on the right and $P+T_H$ on the bottom where $T_W = R(1-W/R+\lfloor W/R \rfloor)$ and $T_H = R(1-H/R+\lfloor H/R \rfloor)$ ensure that the grid covers all of the input image.

For our task of symbol detection, it is important that the local context (and hence receptive field of the output grid cells) is not too large. While local context is helpful, beyond some modest distance, map content becomes unrelated to the symbol within the grid cell and unhelpful to the detection process. This means that the depth of the network must be kept relatively modest. We find that a very small network is adequate for the symbol detection task on our dataset.

3.3 Bounding box regression

We can choose the grid cell resolution R appropriately for our data to ensure that only one symbol will lie within any grid cell. This means that we only need to detect a single bounding box per grid cell, simplifying our output and loss functions (contrast this with YOLO [26] which uses relatively large grid cells and must allow multiple detections per cell). In order to regress bounding box coordinates and class labels, the output tensor (including channel dimension) is of size $S_H \times S_W \times 2 + C$, where C is the number of classes (including one for background) and hence logits that we output per cell. We supervise the classification output with cross entropy loss (see below).

The additional two outputs predict bounding box centre offsets from the centre of the grid cell. We apply sigmoid activation to these spatial outputs such that they represent normalised coordinates within a grid cell. We compute an L1 loss over predicted bounding box centre offsets for those cells that contain a symbol:

$$\mathcal{L}_{\text{coord}} = \frac{1}{\sum_{i=1}^{S_W}\sum_{j=1}^{S_H} \mathbb{1}_{ij}^{\text{sym}}} \sum_{i=1}^{S_W}\sum_{j=1}^{S_H} \mathbb{1}_{ij}^{\text{sym}} |x_{i,j} - \hat{x}_{i,j}|, \tag{2}$$

where $\mathbb{1}_{ij}^{\text{sym}} \in \{0,1\}$ indicates whether grid cell (i,j) contains a symbol, $x_{i,j}$ is the ground truth bounding box centre offset and $\hat{x}_{i,j}$ the estimated one.

3.4 Region segmentation

Our approach and network can naturally be extended to perform coarse semantic segmentation of regions simultaneously with detection of individual symbols. In the context of maps, regions are usually represented by a number of symbols (for example indicating woodland of a particular type) enclosed by symbols indicating boundaries such as walls, hedges and fences. Here, context is crucially important as the input region corresponding to a grid cell might be empty but the cell belongs to a region because it is surrounded by symbols indicating that

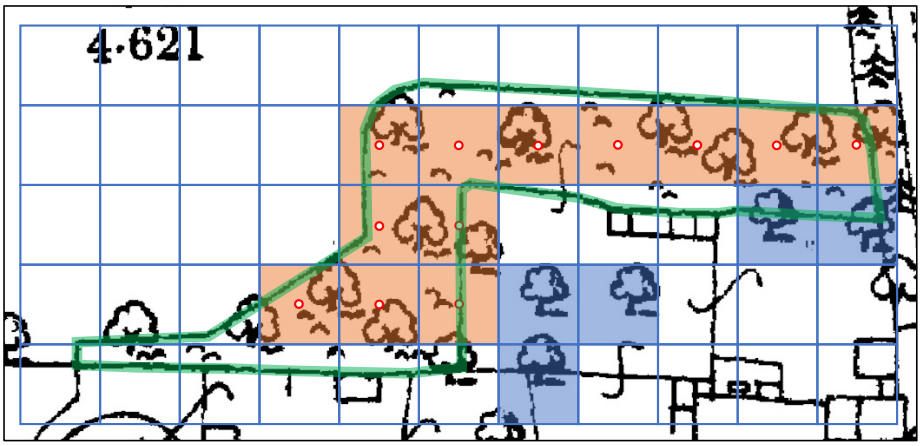

Fig. 2. Coarse semantic segmentation with the YOLSO architecture. A region annotation (shown in green) is converted to a semantic segmentation at grid resolution by applying a point-in-polygon test to the centre of each grid cell (shown as red dots for cells with centres inside the region polygon). Cells labelled as inside the region are shaded orange while cells containing single objects can still be detected as normal (shaded blue here).

region. In addition, the network must be able to reason about which side of a boundary symbol a cell is.

We label segments by drawing arbitrarily shaped polygons. Then, for each grid cell, we perform a point-in-polygon test for the coordinate of the centre of the cell. For any cells with centres lying inside the polygon, we label that cell with the class of the segment. This requires no modification to training for object detection. The number of output classes C is simply increased to include both the symbol and region classes and we define $\mathbb{1}_{ij}^{\text{sym}} = 0$ if grid cell (i, j) contains a region class (i.e. we do not compute a coordinate loss for region cells).

3.5 Hard negative mining

In many datasets, including our map data, symbols and regions are relatively sparse. This means that the vast majority of grid cells are labelled as background. Naively training directly on this data leads to a very high false negative rate. For this reason, hard negative mining is essential for good performance. We do not ignore easy negatives entirely since this leads to misclassifying many background cells but instead downweight them as follows:

$$\mathcal{L}_{\text{class}} = \frac{1}{S_W S_H} \sum_{i=1}^{S_W} \sum_{j=1}^{S_H} -w_{ij} \log \frac{\exp z_{ijc_{ij}}}{\sum_{k=1}^{C} \exp z_{ijk}}, \tag{3}$$

where z_{ijk} is the kth logit for grid cell (i, j), c_{ij} is the ground truth class for grid cell (i, j) and w_{ij} downweights easy background cells as follows:

$$w_{ij} = \begin{cases} W & \text{if } c_{ij} = 1 \wedge z_{ij1} = \max_k z_{ijk} \\ 1 & \text{otherwise} \end{cases} . \tag{4}$$

An easy background cell is defined as one which has ground truth background label ($c_{ij} = 1$) and the logit for the background class is maximal over all logits for that cell ($z_{ij1} = \max_k z_{ijk}$), i.e. the network currently correctly labels it as background. We use a weight of $W = 0.1$ in all our experiments. In general, this weight should be set similarly to the proportion of non background symbols in the dataset.

3.6 Implementation

The symbols in our dataset originally had bounding boxes of size 48×48. We choose the next smallest power of two, $R = 32$, as our grid cell resolution. Since no bounding box centres occur closer then 32 pixels, this ensures that our assumption of only one object detection per grid cell is valid.

We use five blocks of convolution/batchnorm/ReLU/max pooling. The first convolution layer has kernel size 7×7, the following four have size 3×3 and the number of output channels per convolution layer are: 64, 64, 128, 256, 512. The final part of the network reasons at grid cell scale in order to compute final detection output. We apply a convolution/batchnorm/ReLU with kernel size 3×3 and 512 output channels in order to share information between adjacent grid cells before a final 1×1 convolution layer with $C + 2$ output channels. This architecture requires a boundary of width $P = 65$ pixels on the input images. This means that the receptive field of each output grid cell (corresponding to a 32×32 region in the input images) is 162×162. We find that this provides sufficient context for both symbol detection and region segmentation.

4 Experiments

4.1 Dataset

Our work focuses on the Ordnance Survey First Edition 1:2500 County Series, produced in the latter half of the nineteenth century and covering most of Great Britain. Only uncultivated areas in highest uplands and lowest lowlands were excluded [24]. These historic maps are unusual in that they attempt to show all freestanding trees. More modern maps tend to dispense with the gargantuan task of surveying so many trees, limiting themselves to only showing groups of trees and woodlands. Our objective is to develop a method of quickly detecting and georeferencing both freestanding trees and areas of trees on the historic maps, thereby creating new digital 'National Historic Tree Map', which can be used a baseline for researching landscape-scale environmental change throughout the

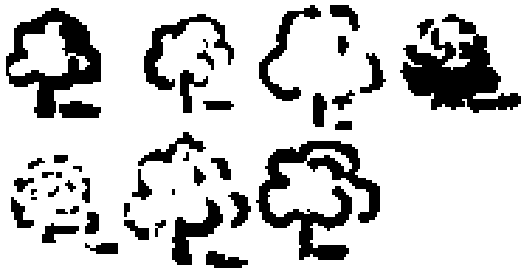

Fig. 3. A range of different symbols are used to denote freestanding broadleaved trees.

twentieth century and beyond. However, this is not a straightforward task. The symbols denoting trees are not consistent across the whole First Edition County Series (Figure 3), they can vary as a result of the mapmaking and digitisation processes, and they can be associated with other symbols and features that make them hard to detect or segment (Figure 4).

Historic Ordnance Survey maps are available from a number of third party data providers, libraries and archives. We elected to use digitised versions of the historic maps produced by the Landmark Information Group and made available through the online Edina Digimap platform[4]. One advantage of using this dataset is that the original mapsheets, which were surveyed using the Cassini Projection, have been re-projected to the modern National Grid and stitched together. The maps are thus served as individual $1km^2$ tiles in GeoTIFF format, scanned at two resolutions such that they are either 4724 or 6000px square. For our dataset, we randomly selected 135 individual map tiles from the within the historic county of Yorkshire (encompassing the North, West and East Ridings). This comprises approximately 1% of the total land area of the historic county. Each tile was manually labelled using GIS software, and point coordinates denoting individual trees and polygons marking out groups of trees were stored in shapefiles relating to each map tile.

782 regions (wooded areas) were labelled manually by defining a polygon and assigning one of four classes (deciduous woodland, coniferous woodland, mixed woodland, orchard). Point locations for individual tree symbols were labelled semi-automatically. 23,807 approximate locations were labelled manually. These were refined or discarded and one of two classes (broadleaf and conifer) chosen by applying template matching within a 96px square centred on the manually plotted point. The templates were of size 48px and, within each target area, the location of the highest template match was used to derive a pixel coordinate centre point for that tree symbol. Poor matches were excluded, such as those not reaching a certain confidence threshold and those that would purport to

[4] 1:2500 County Series 1st Edition [TIFF geospatial data], Scale 1:2500, Updated: 30 November 2010, Historic, Using: EDINA Historic Digimap Service, https://digimap.edina.ac.uk. Downloaded: 2015-2022.

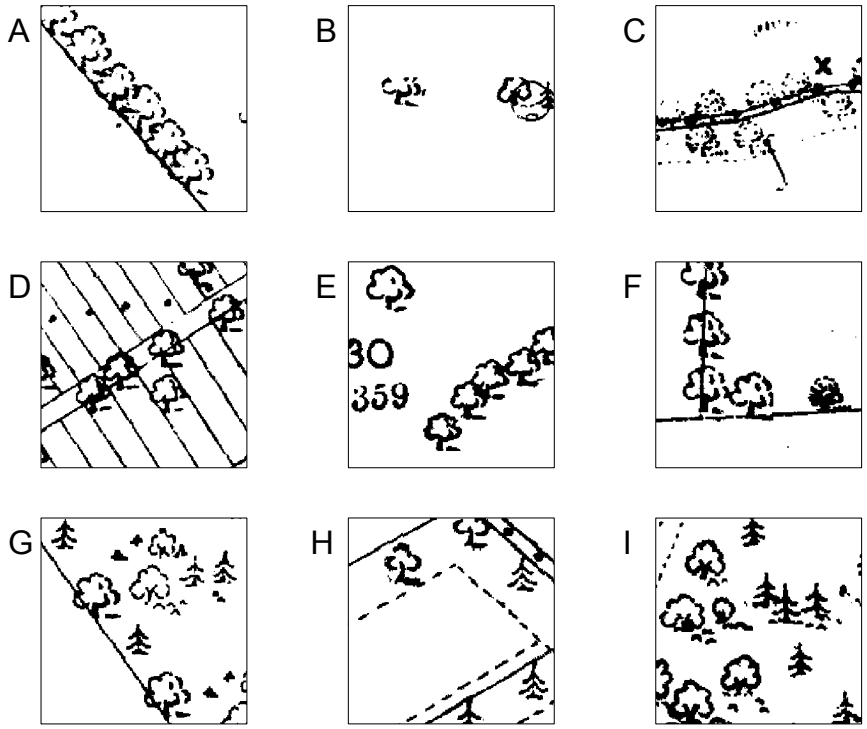

Fig. 4. Characteristics of the map images that make tree symbols difficult to detect and segment: **(A)** Touching or overlapping tree symbols. **(B)** Symbols partially truncated at the edges of map tiles. **(C)** Poorly reproduced symbols. **(D)** Symbols partially occluded by other map features. **(E)** Ostensibly similar symbols of different sizes. **(F)** Different symbols for the same class of tree within a single map tile - this is a result of different map sheets from different areas, sometimes produced at different times, being stitched together to produce a single contiguous map. **(G)** Individual tree symbols shown in close proximity to or overlapping with areas of woodland. **(H)** Areas where it is unclear whether individual trees or groups of trees are being depicted. **(I)** Areas of woodland that change abruptly from one class to another (here from deciduous broadleaved to coniferous) with no intervening boundary.

match with symbols that exceed the bounds of the 96px square target area. As a result, the final labelled dataset comprised 22,882 point features (96.1% of those initially manually labelled).

4.2 Results

In Figure 5 we show some qualitative results from our method. We choose two test regions which illustrate performance mainly on point symbols (top row) and

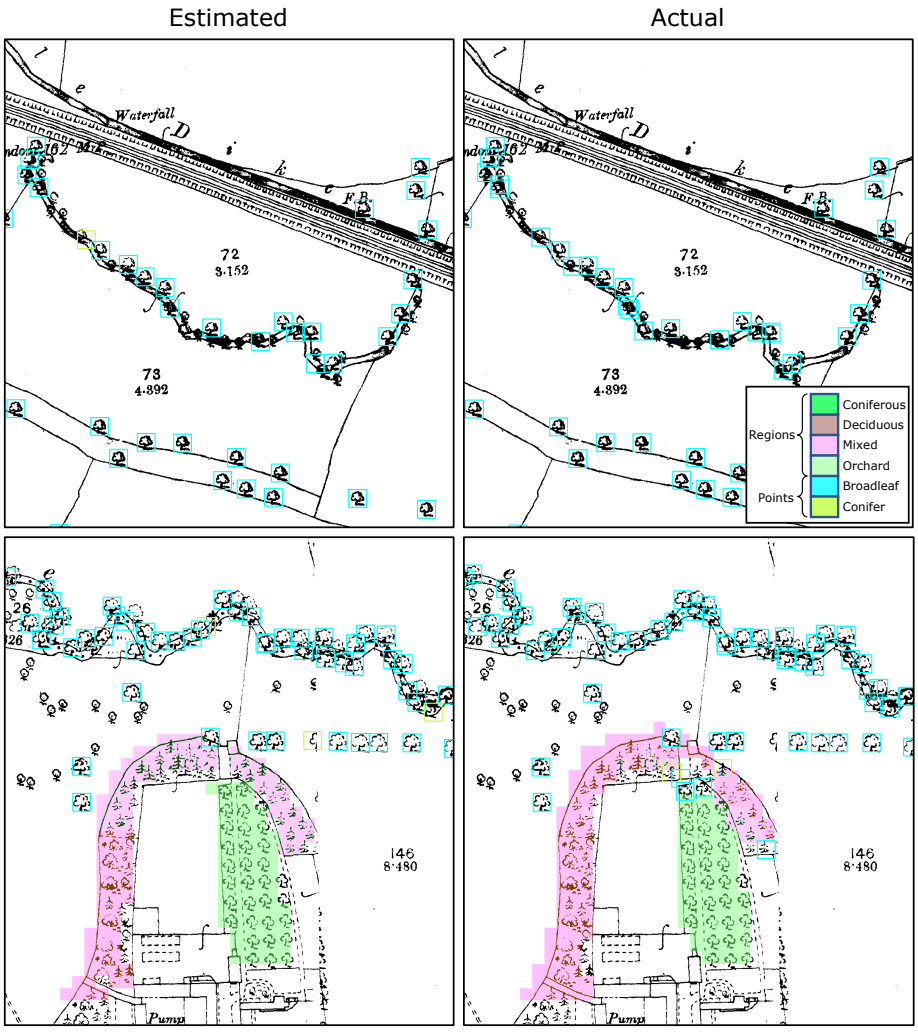

Fig. 5. Qualitative object detection and region segmentation results. On the first row we show a crop in which many point labelled individual trees are close together and obscured by boundary markings. In the bottom row we show a mix of point and region segmentations in which two different region types are adjacent.

on region segmentation (bottom row). In the top row, our approach is able to accurately label point symbols even with heavy distractors from overlaid boundary symbols. In the bottom row, our approach accurately recovers the shape and changing class of adjacent regions and correctly ignores symbols that do not indicate freestanding trees outside of the regions.

We compare our approach against YOLOv3 [28] and Faster R-CNN [29]. During training, we use different strategies for the different methods for cropping

training images from the 4k×4k images. For YOLSO, we train on a 16×16 grid, which, including the necessary boundary, means that we crop 642×642 images. The crops are random which has the same effect as translation augmentation - i.e. every symbol may be observed at every possible position within a cell during training. While our approach can handle training crops with very few positive samples (due to our easy negative down-weighting strategy and the fact that regions provide additional positive samples) for YOLOv3 and Faster R-CNN we found that performance was improved by excluding crops that contained too few positive samples and also shifting the crop window such that partial symbols at the boundary were avoided. We use 512×512 crops for Faster R-CNN and 416×416 for YOLOv3. At test time, our method can process an entire tile in one go (using the padding strategy described in Section 3.2). Whereas for YOLOv3 and Faster R-CNN, we retain the image size used for training and process a tile by sliding a window over the full image.

We provide quantitative results in Table 1. For point symbols, we define a correct prediction as one with correct class and IOU>0.5. Our approach significantly outperforms the two standard methods. Note that both comparison methods often estimate bounding boxes of completely the wrong aspect ratio or scale since they do not incorporate the constraint of known size that our approach does. Both comparison methods suffer more from false negatives than false positives, likely due to the extreme symbol sparseness and small size of the symbols compared to the objects these methods are usually used to detect.

In Figure 6 we illustrate the most common failure cases of our method. In A, symbols are interpreted as indicating a region due to the context provided by the boundary. However, despite the similarity, these symbols actually indicate freestanding trees and should be detected as points. In B, a mistake is made between the mixed and deciduous classes. This is challenging since the deciduous symbols are used in mixed regions with the context of other symbols (potentially quite distance) providing the disambiguation. In C, parts of a region are labelled as background due to lack of any symbol context in the surrounding region. Note also that a region symbol is misinterpreted as a freestanding conifer - again, the symbols are very similar.

5 Conclusions

We have proposed a variant of the YOLO object detection framework that is specialised for the detection of non-uniform symbols and regions depicted by

Method	Point mAP@0.5	Region mAP
YOLSO	97.9%	94.6%
YOLOv3 [28]	43.6%	n/a
Faster R-CNN [29]	70.2%	n/a

Table 1. Quantitative object detection (middle column) and region segmentation (right) results. YOLOv3 [28] and Faster R-CNN [29] do not perform segmentation.

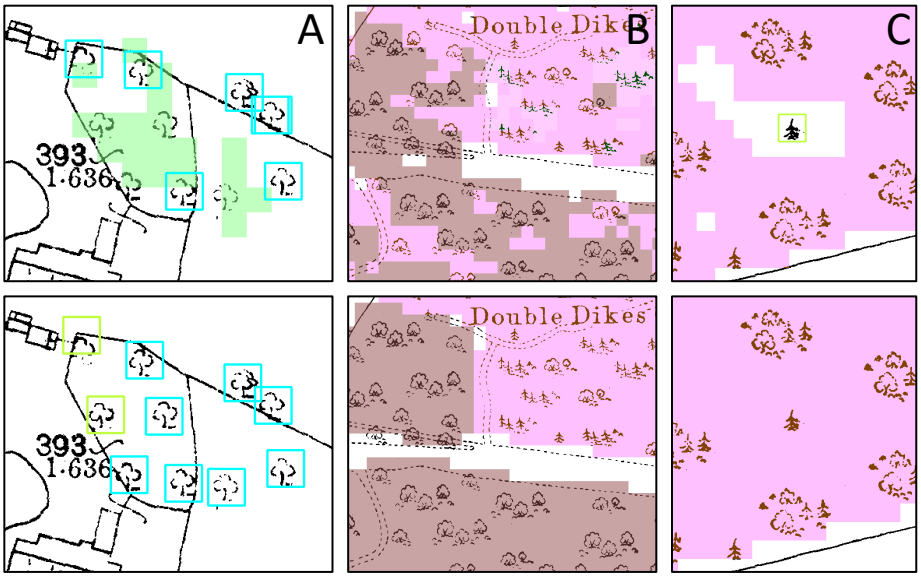

Fig. 6. Illustrative failure cases (top row: estimated, bottom row: ground truth). A: misinterpreting point symbols as regions. B: confusion between region classes. C: false negatives in empty regions where context is too distant.

symbols in documents such as historic maps. By accounting for the specific properties of the task, namely that symbol scale is fixed, local context is important but global context and spatial dependence should be ignored, we arrive at a model that is lightweight and efficient but significantly outperforms the application of existing generic object detection methods to the problem. In addition, our approach additionally computes coarse semantic segmentation using the same single network. We have shown that a relatively small network performs well on this well constrained task and perhaps even aids performance by avoiding the receptive field becoming too large. We believe that performance can be further improved by additional tuning of this trade off, particularly for featureless areas inside regions which require larger context. We did nothing to handle the imbalance between classes nor between region and point symbols and believe additional performance gains could be achieved here.

Acknowledgments

This research was conducted as part of the Future Of UK Treescapes project 'Branching Out: New routes to valuing urban treescapes', funded by UK Research and Innovation [Grant Number: NE/V020846/1]

References

1. Adorno, W., Yi, A., Durieux, M., Brown, D.: Hand-drawn symbol recognition of surgical flowsheet graphs with deep image segmentation. In: 2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE). pp. 295–302. IEEE (2020)
2. Baily, B.: The extraction of digital vector data from historic land use maps of great britain using image processing techniques. E-perimetron **2**(4), 209–223 (2007)
3. Branson, S., Wegner, J.D., Hall, D., Lang, N., Schindler, K., Perona, P.: From google maps to a fine-grained catalog of street trees. ISPRS Journal of Photogrammetry and Remote Sensing **135**, 13–30 (2018)
4. Budig, B.: Extracting spatial information from historical maps: algorithms and interaction. Würzburg University Press (2018)
5. Chiang, Y.Y., Duan, W., Leyk, S., Uhl, J.H., Knoblock, C.A.: Using historical maps in scientific studies: Applications, challenges, and best practices. Springer (2020)
6. Elyan, E., Jamieson, L., Ali-Gombe, A.: Deep learning for symbols detection and classification in engineering drawings. Neural networks **129**, 91–102 (2020)
7. Garcia-Molsosa, A., Orengo, H.A., Lawrence, D., Philip, G., Hopper, K., Petrie, C.A.: Potential of deep learning segmentation for the extraction of archaeological features from historical map series. Archaeological Prospection **28**(2), 187–199 (2021)
8. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
10. Groom, G.B., Levin, G., Svenningsen, S.R., Perner, M.L.: Historical maps machine learning helps us over the map vectorisation crux. In: Automatic Vectorisation of Historical Maps: International workshop organized by the ICA Commission on Cartographic Heritage into the Digital. pp. 89–98. Department of Cartography and Geoinformatics, ELTE Eötvös Loránd University (2020)
11. Hosseini, K., McDonough, K., van Strien, D., Vane, O., Wilson, D.C.: Maps of a nation? the digitized ordnance survey for new historical research. Journal of Victorian Culture **26**(2), 284–299 (2021)
12. Hosseini, K., Wilson, D.C., Beelen, K., McDonough, K.: Mapreader: a computer vision pipeline for the semantic exploration of maps at scale. In: Proceedings of the 6th ACM SIGSPATIAL International Workshop on Geospatial Humanities. pp. 8–19 (2022)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015)
14. Islam, M.A., Jia, S., Bruce, N.D.: How much position information do convolutional neural networks encode? In: International Conference on Learning Representations (2019)
15. Julca-Aguilar, F.D., Hirata, N.S.: Symbol detection in online handwritten graphics using faster r-cnn. In: 2018 13th IAPR international workshop on document analysis systems (DAS). pp. 151–156. IEEE (2018)
16. Kara, L.B., Stahovich, T.F.: An image-based, trainable symbol recognizer for hand-drawn sketches. Computers & Graphics **29**(4), 501–517 (2005)

17. Kayhan, O.S., van Gemert, J.C.: On translation invariance in CNNs: Convolutional layers can exploit absolute spatial location. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14274–14285 (2020)
18. Laumer, D., Lang, N., van Doorn, N., Mac Aodha, O., Perona, P., Wegner, J.D.: Geocoding of trees from street addresses and street-level images. ISPRS Journal of Photogrammetry and Remote Sensing **162**, 125–136 (2020)
19. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
20. Li, Z., Jin, L., Lai, S., Zhu, Y.: Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 175–180. IEEE (2020)
21. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
22. Ltd, B.I.: National tree map (Nov 2022), `https://bluesky-world.com/ntm/`
23. Maxwell, A.E., Bester, M.S., Guillen, L.A., Ramezan, C.A., Carpinello, D.J., Fan, Y., Hartley, F.M., Maynard, S.M., Pyron, J.L.: Semantic segmentation deep learning for extracting surface mine extents from historic topographic maps. Remote Sensing **12**(24), 4145 (2020)
24. Oliver, R.: Ordnance Survey Maps: a concise guide for historians. Charles Close Society (1993)
25. Petitpierre, R.: Neural networks for semantic segmentation of historical city maps: Cross-cultural performance and the impact of figurative diversity. ArXiv **abs/2101.12478** (2021)
26. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proc. IEEE conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (2016)
27. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7263–7271 (2017)
28. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
29. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
30. Uhl, J.H., Leyk, S., Chiang, Y.Y., Knoblock, C.A.: Towards the automated large-scale reconstruction of past road networks from historical maps. Computers, Environment and Urban Systems **94**, 101794 (2022)
31. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2), 154–171 (2013)
32. Williamson, T., Barnes, G., Pillatt, T.: Trees in England: management and disease since 1600. University of Hertfordshire Press (2017)
33. Wong, C.S., Liao, H.M., Tsai, R.T.H., Chang, M.C.: Semi-supervised learning for topographic map analysis over time: a study of bridge segmentation. Scientific Reports **12**(1), 18997 (2022)
34. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: International Conference on Learning Representations (2020)