International Journal on Document Analysis and Recognition (IJDAR) (2018) 21:161–175
https://doi.org/10.1007/s10032-018-0305-2

SPECIAL ISSUE PAPER

Learning to detect, localize and recognize many text objects in document images from few examples

Bastien Moysset[1,2] · Christopher Kermorvant[3] · Christian Wolf[2,4]

Received: 29 September 2017 / Revised: 21 May 2018 / Accepted: 23 May 2018 / Published online: 9 June 2018

Abstract
The current trend in object detection and localization is to learn predictions with high capacity deep neural networks trained on a very large amount of annotated data and using a high amount of processing power. In this work, we particularly target the detection of text in document images and we propose a new neural model which directly predicts object coordinates. The particularity of our contribution lies in the local computations of predictions with a new form of local parameter sharing which keeps the overall amount of trainable parameters low. Key components of the model are spatial 2D-LSTM recurrent layers which convey contextual information between the regions of the image. We show that this model is more powerful than the state of the art in applications where training data are not as abundant as in the classical configuration of natural images and Imagenet/Pascal-VOC tasks. The proposed model also facilitates the detection of many objects in a single image and can deal with inputs of variable sizes without resizing. To enhance the localization precision of the coordinate regressor, we limit the amount of information produced by the local model components and propose two different regression strategies: (i) separately predict lower-left and upper-right corners of each object bounding box, followed by combinatorial pairing; (ii) only predict the left side of the objects and estimate the right position jointly with text recognition. These strategies lead to good full-page text recognition results in heterogeneous documents. Experiments have been performed on a document analysis task, the localization of the text lines in the Maurdor dataset.

Keywords Text line detection · Neural network · Recurrent · Regression · Local · Document analysis

1 Introduction

Object detection and localization in images is currently dominated by approaches which first create proposals (hypothesis bounding boxes) followed by feature extraction and pooling on these boxes and classification, the latter steps being usually performed by deep networks [2,14,15,35,36]. Very recent methods also use deep networks for the proposal step [22,35,36], sometimes sharing features between localization and classification. Differences exist in the detailed architectures in the way calculations are shared over layers, scales, spatial regions, etc. (see Sect. 3 for a detailed analysis). Another criterion is the coupling between hypothesis creation and confirmation/classification. Earlier works create thousands of hypotheses per image, sometimes using low-level algorithms (e.g., R-CNN [15]), leaving the burden of validation to a subsequent classifier. Current work tends to create very few proposals per image, which satisfy a high degree of "objectness".

In this work, we focus on the localization step, targeting cases where the existing methods tend to give weak results (Fig. 1):

– the current trend is to design high capacity networks trained on large amounts of training data either directly or as a pre-training step. However, in some applications, the image content is only very weakly correlated with the data available in standard dataset like ImageNet/ILSVRC [37]. In the case of small and medium amounts of training data, fully automatic training of deep models remains a challenge.
– we allow for the detection and localization of a relatively high number of potentially small objects in an image,

✉ Bastien Moysset
bm@a2ia.com

1 A2iA SA, Paris, France
2 LIRIS, UMR 5205, INSA-Lyon, 69621 Villeurbanne, France
3 Teklia SAS, Paris, France
4 CNRS, Université de Lyon, Lyon, France

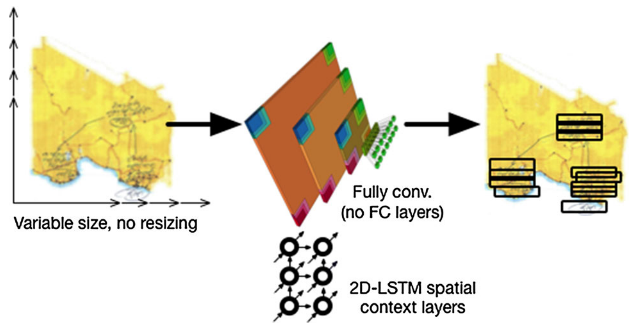
Variable size, no resizing
Fully conv. (no FC layers)
2D-LSTM spatial context layers

Fig. 1 A fully convolutional model with high spatial parameter sharing and fully trainable 2D-LSTM context layers learns to detect potentially many objects from few examples and inputs of variable sizes

which is especially hard for existing methods [2]. Our target application is the localization of text boxes.
– we focus on the precision of the localization, which is important when detecting small objects and, in our case, for the following text recognition step. Traditional regression-based object detection models tend to have issues with these small objects [35].

Similar to recent work, the proposed method localizes objects by direct regression of (relative) coordinates. The main contribution we claim is a new model which performs spatially local computations, efficiently sharing parameters spatially. The main challenge in this case is to allow the model to collect features from local regions as well as globally pooled features in order to be able to efficiently model the context.

Similar to models like YOLO [35] and single-shot detector [22], our outputs are assigned to local regions of the image. However, in contrast to these methods, each output is trained to be able to predict objects in its support region, or outside. Before each gradient update step, we *globally* match predictions and ground-truth objects. Each output of our model directly sees only a limited region of the input image, which keeps the overall number of trainable parameters low. However, outputs get additional information from outside regions through context, which is collected using spatial 2D recurrent (LSTM) units. This spatial context layer proved to be a key component of our model.

We propose the following contributions:

– A new fully convolutional model for object detection using spatial 2D-LSTM layers for handling spatial context with an objective of high spatial parameter sharing.
– The capability of predicting a large number of outputs, made possible by the combination of highly local output layers (1×1 convolutions) and preceding spatial LSTM layers.
– The possibility of predicting outputs from input images of variable size without resizing the input.
– An application to document analysis with experiments on the difficult and heterogeneous Maurdor dataset, which shows that the model significantly outperforms the state of the art in objects detection.
– The comparison of three regression strategies, namely (i) detecting a full bounding box; (ii) detecting two corners separately, followed by combinatorial pairing; and (iii) detecting the left side of the bounding box, followed by joint text recognition and detection of the right side (the end of the bounding box).

The paper is organized as follows: the next section briefly outlines related work. Section 3 discusses properties and trade-offs of deep models related to convolutions, pooling and subsampling, which will be related to our proposed model. Based on these conclusions, a new model, recurrent and local, is introduced in Sect. 4. Three object detection strategies are explained in Sect. 5 to increase the precision of the predictions. The training strategy of the networks is detailed in Sect. 6 and experiments and results are shown in Sect. 7.

2 Related work

Many text line detection techniques have been developed for document analysis [11,21]. Most of them are using traditional image processing techniques. Rules are used to group the connected components of the image [46], parts of these connected components [38] or directly the pixels [39]. Another approach is to successively split the image to segment the text lines. It can be done by projection profiles [27], by following a path between the lines [30] or by minimizing an energy function [40]. These techniques can work well on the task they were designed for but lack generalization on heterogeneous datasets. For these reasons, learning techniques based on neural networks have been used for the classification of the parts of an image as text or non-text [6,8] but still need post-processing.

Similarly, earlier (pre-deep learning) work on object recognition proceeded through matching of local features [25] or by decomposing objects into mixtures of parts and solving combinatorial problems [13]. Early work on deep learning first extended the sliding window approach to deep neural networks. To avoid testing a large number of positions and aspect ratios, R-CNN [15] introduced the concept of object proposals, created by separate methods, followed by convolutional networks to classify each proposal. The concept was improved as Fast R-CNN [14] and Faster R-CNN [36].

Erhan et al. proposed Multibox [10,42], which performs direct regression of bounding box locations instead of relying on object proposals. After each forward pass, network outputs are assigned to target ground-truth boxes through a matching algorithm. YOLO [35] and the single-shot detector [22] can be seen as variants of this concept, they will be discussed in more detail in Sect. 3.

R-FCN [7] extend the Faster R-CNN approach using fully convolutional networks and predicting fixed parts of the object and [28] enables these parts to be deformable.

Some recent work strives to detect and localize objects with pixel-wise precision, which somewhat blurs the boundaries between object detection and semantic segmentation [33]. Methods which learn to segment without pixel-wise ground truth have also been proposed [32]. Pixel-wise segmentation is not needed in our application, where the segmentation step is performed in a latter stage jointly with recognition (recognition results will be given in the experimental section).

Recurrent networks have been used for object detection in early works [1], and context through spatial 2D-LSTM recurrent networks has been proposed as early as in [17]. However, up to our knowledge, no method did use 2D-LSTM networks for object localization. Similarly to our method, inside-Outside-Nets [2] contain 2D spatial context layers collecting information from 4 different directions. However, the hidden transitions of recurrent layers are set to identity, whereas our model contains fully fledged trainable 2D-LSTM layers. Moreover, localization is performed as ROI proposals with selective search, the deep model being used only for classification and bounding box correction, whereas we do not require a region proposal step. Our model directly performs bounding box regression. Other recent work uses 2D recurrent networks for semantic segmentation [45].

CNNs have been used before for text detection, for instance in [48], a fully convolutional network (FCN) is used to classify each position of a salient map as text or non-text.

Some of these techniques have been used for the detection of text in natural scene images using adaptations of Edge-Boxes [20], Faster RCNN [26], SSD [23] or a YOLO-related method [18]. Some of them [23,26] enable to handle oriented text but only few objects are present in the images and, to our knowledge, such approaches have not been applied to document analysis tasks yet.

The problem of dataset sizes has been addressed before, with strategies reaching from external memories [43] and unsupervised learning, for instance by learning feature extraction from physics [41].

3 Delving again into convolutions, pooling, strides and spatial structure

Object detection and localization with convolutional deep neural networks are governed by a set of implicit properties and requirements, which we will try to lay out in the following lines. We will concentrate on the approach of direct prediction of object locations (as opposed to creating proposals from additional and not-tightly connected methods). The goal of this section is to discuss the effects and importance of each part and the trade-offs to consider in these architectures, which will lead us then to the formulation of the proposed model.

The input image is passed through a series of convolutional layers, each of which extracts features from the preceding layer. Although not absolutely required, reducing the spatial size of the features maps (often combined with pooling) is frequently done in order to increase the receptive fields, i.e., the relative size of the filters with respect to the inputs. Choosing when to pool and to reduce can be critical, and optimizations can lead to large decreases in the number of trainable parameters [19]. An alternative to in-between-layer pooling is changing the size of filters, especially as "*a trou*" computation in order to keep the number of parameters low [47].

At some point, a model needs to collect features from a spatial support region. The way this pooling is distributed over the different layers will decide important properties of the model:

– Classical networks stop the sequence of convolutions and reductions before the *spatial* size of the feature map shrinks to 1×1, keeping a spatial/geometrical structure in the feature representation. Subsequent fully connected layers then perform further feature extraction, implicit pooling and decision taking.
  The spatial structure of the feature map allows to perform controlled pooling from specific regions, but limits the shift invariance of the representation.
– More recently, fully convolutional networks (FCN) perform convolutions and reductions+pooling until the spatial size of the feature map is negligible, e.g., 1×1, with a high feature dimension ($1\times1\times4096$ in the network for semantic segmentation proposed in [24]). The goal here is to fully translate geometry and appearance into features and semantics.
  Training can in principle lead to a spatial structuring of the feature dimension, i.e., training can lead to a situation where different elements of the feature layer correspond to different regions in the input image. However, this is not a constraint in the model and each activation of the last feature layer can potentially contain features from the full image.

Object detection and localization require certain properties, like shift invariance, spatial precision and context collected from the global scene. Several state-of-the-art models, Multi-box [10], YOLO [35] and single-shot detector (SSD) [22]

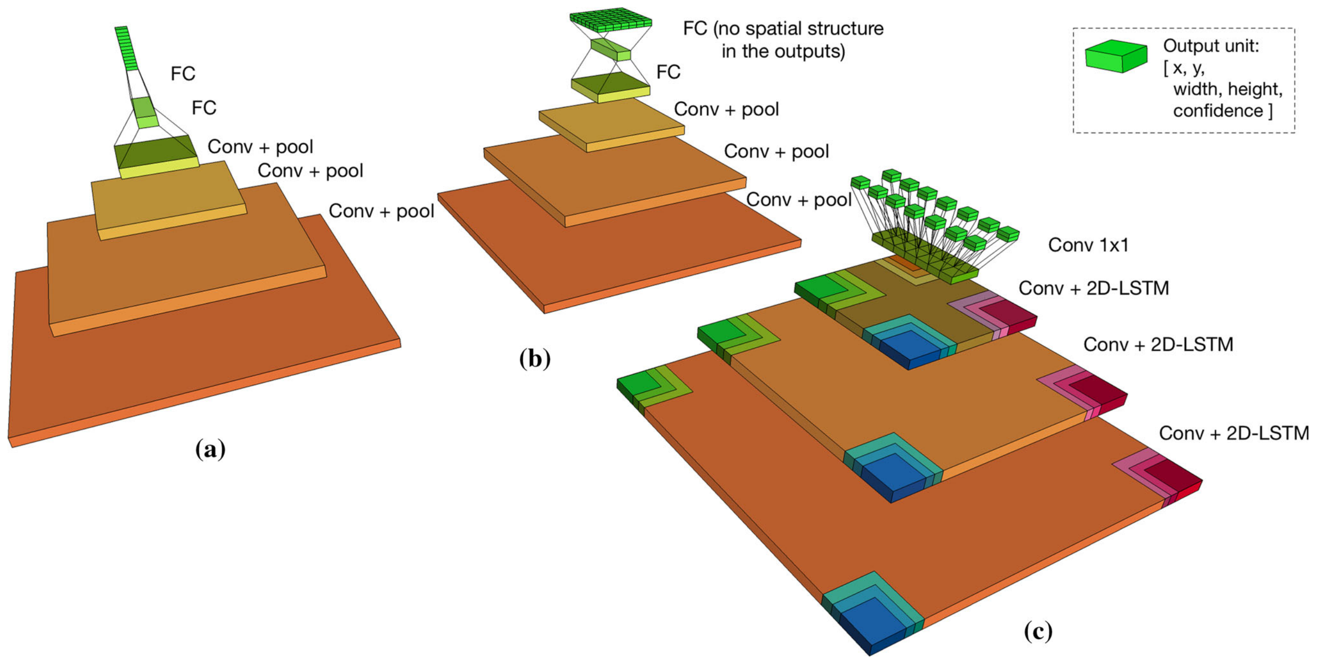

Fig. 2 Sketches of different ways to model bounding box regression spatially (we do not show the correct numbers of layers and units). **a** Multibox [10]; **b** Yolo [35] and SSD [22]; these three models pass through a fully connected layer, which connects to a set of bounding box outputs; architecture-wise, these three models are identical. The grid-like display of the outputs in **b** is due to training; **c** our proposed model is fully convolutional (no fully connected (= FC) layers) and keeps spatial structure in the feature map. Global context is handled through 2D-recurrent LSTM networks, indicated through colored squares starting in each corner of each feature map (see also Fig. 3)

tackle this through an architecture sketched in Fig. 2a, b.[1] A sequence of convolutions and reductions decrease the spatial size of feature maps down to a small grid (7×7 for [35], 9×9 for [22]). This map is then fully connected to a 1×1×4096 feature layer and again fully connected to a set of outputs, each output predicting bounding box positions and confidence scores (as well as class scores if required). This approach has several advantages. Each of the outputs is fully connected to previous layers and therefore potentially has access to information from the full image. The last feature layer mixes spatial structure and feature dimensions in a trainable way.

Although there is no principled difference in how the last fully connected layer is actually implemented in the three models, we display the output layer differently for Multibox [10] (Fig. 2a) and for YOLO [35] or SSD [22] (Fig. 2b). For the latter two, and also in accordance with the figures of the respective papers, the outputs are shown as a spatial grid (7×7 for [35], 9×9 for [22]). However, this structuring is an interpretation, as the spatial structure of the grid is not wired into the network architecture. It is justified through the way training is performed in these models, in particular on the way ground-truth outputs are matched (assigned) to the network outputs. In the case of [35], this assignment is purely spatial: outputs of a given cell are trained to provide predictions of a spatial region corresponding to this cell (see Sect. 6).

The main shortcoming of these models, which we will address in the next section, lies in the fully connected feature and output layers at the end. We argue that they limit invariance and contain too many parameters.

4 A local spatially recurrent model

Document analysis tasks, and especially the localization of text lines in heterogeneous documents, are different from the traditional scene text object detection tasks like Pascal-VOC [12] or ImageNet/ILSVRC [37]. The main differences are summarized in Table 1. Namely, the Maurdor dataset [5] is much smaller than the scene text object datasets and the number of object to detect is higher, both when considering the mean and the maximum number of objects per image. On the contrary, there is only one class of object, the text lines, which lead to a probably lower complexity of the task.

In order to deal with these specificities, we propose a new model designed to detect a large number of (potentially)

[1] The purpose of this figure is to show the strategy these models use to translate geometry and resolution into features. In particular, we do not show the actual numbers of layers and units. For SSD [22], we do not show the way how this model handles multiple scales.

Table 1 Comparison of statistics for Pascal-VOC [12] and ILSRVC [37] scene text object detection datasets and Maurdor [5] document text line object detection dataset

	Pascal-VOC 2012	ILSRVC 2014	Maurdor
Number of images	17,125	544,546	3995
Number of objects	40,138	615,299	135,834
Mean number of objects per image	2.34	1.13	34.00
Maximum number of object per image	56	17	567
Mean object area	0.1529	0.3833	0.0050
Mean horizontal object size	0.3406	0.6033	0.2327
Mean vertical object size	0.449	0.6353	0.0216
Number of classes	20	200	1

small objects from a low number of training examples, i.e., with a model with a small number of trainable parameters. We achieve this with two techniques:

A. Feature sharing—we predict different object locations from local features only. More precisely, the output layer of a single object bounding box is not fully connected to the previous layer, as illustrated in Fig. 2c. Outputs are connected through 1×1 convolutions, i.e., each element (i, j) of the last feature map is connected to its own set of K output modules, each module consisting of relative x and y positions, width, and height and a confidence score used to confirm the presence of an object at the predicted position. The objectives here are twofold:

- To drastically reduce the number of parameters in the output layer by avoiding parameter hungry fully connected layers.
- To share parameters between locations in the image, increasing shift invariance and significantly reducing the requirements for data augmentation.

Indeed, the number of input neurons of the fully connected layer is divided by the size of the last feature map as we use a vector of this feature map and not all of it. This is enabled by the fact that we only ask these features to carry the relevant information about a part of the image, and not the whole image. Moreover, the number of output neurons of this fully connected layer can also be divided because we now want to be able to detect the maximum number of objects that can be present in a small part of the image and not the maximum number of objects that can be present in the whole image. Most of our network parameters were in this last fully connected layer and reducing it that way enables our network to be trained with a lower amount of data.

Sharing the parameters between the locations means that the same parameters will be responsible to predict several objects, for example both a line at the top of the page and a line at the bottom of the page. It means that more object examples will be seen by a given parameter which will reduce the problem of data scarcity.

B. Spatial recurrent context layers—the drawback of local parameter sharing is twofold: (i) objects may be larger than the receptive field of each output, and (ii) we may lose valuable context information from the full input image. We address both these concerns through context layers consisting of Multi-Dimensional Long-Short term memory models [17], which are inserted between the convolutional layers. These MD-LSTM layers aim at recovering the context information from the area outside of the receptive field.

Figure 3 illustrates how the context layers are organized. Each convolutional layer is followed by 4 different parallel 2D-LSTM layers, which propagate information over the feature map elements in 4 different diagonal directions, starting at the 4 edges. For each of the directions, each element gets recurrent connections from 2 different neighboring sites. The outputs of the 4 directions are summed—concatenation would have been another possibility, albeit with a drastically higher amount of parameters. No pooling is performed between the convolutions. Spatial resolution is reduced through convolutions with strides between 2 and 4 (see Table 5).

The network outputs are computed from the last hidden layer as a regression of the *normalized relative* bounding box locations. In particular, the absolute location of each predicted object is calculated by multiplying the network output with a width parameter vector Λ and adding an offset vector Δ, whose values depend on the architecture of the network. Λ corresponds to the sizes of the associated convolutional receptive fields while Δ corresponds to the position of these receptive fields in the whole page image.

More formally, if $l_{i,j,k}$ is the location prediction for the k^{th} object prediction of element (i, j) of the last feature map, then $l_{i,j,k}$ is a vector of size P (where P is the number of coordinates predicted per object. For a box, $P = 4$).

$$l_{i,j,k} = \{l^p_{i,j,k}\},\ p = 1 \ldots P. \tag{1}$$

And it is computed as follows:

$$l_{i,j,k} = \Lambda^T \sigma \left(\mathbf{U}_k \mathbf{h}_{i,j} + \mathbf{c}_k\right) + [i-1 \quad j-1]^T \Delta \tag{2}$$

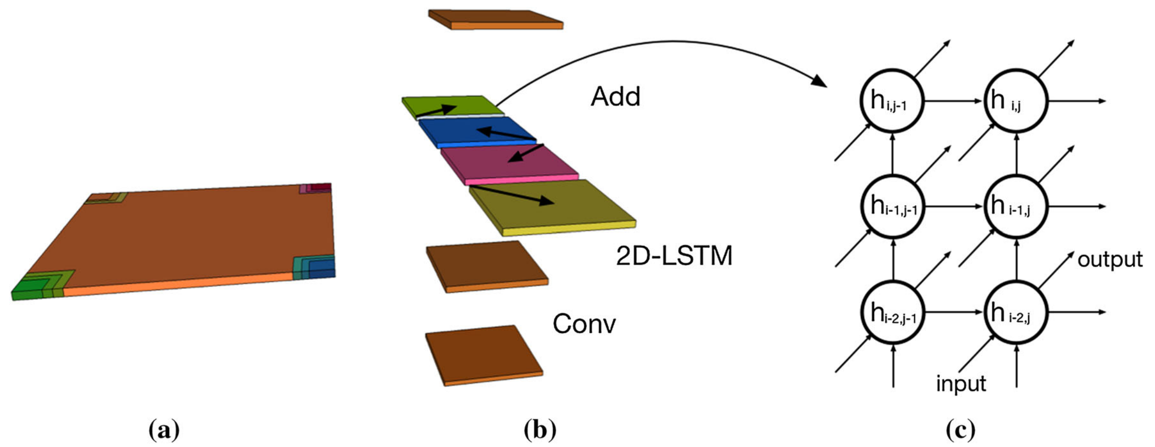

Fig. 3 **a** One of the conv+LSTM modules as shown in Fig. 2; **b** the module is composed of a convolutional layer and four-directional 2D-LSTM layers in parallel, whose output feature maps are then element-wise added (not concatenated); **c** a single-directional 2D-LSTM, shown for left-to-right/bottom-to-top direction. Each element gets recurrent connections from two different predecessors. Only a single unit is shown per site; RNN notation is used (memory cell/gates are not shown). In contrast to [2], we use real LSTM models with trainable transition matrices

where **h** is the last hidden layer, $\sigma(\cdot)$ is the element-wise sigmoid function and the weights $\mathbf{U}_k$ and biases c_k are trainable parameters. Note that, since the outputs are 1×1 convolutions, the parameters $\{\mathbf{U}_k, c_k\}$ are shared over locations (i, j). However, each object predictor k features its own set of parameters.

Flexibilty—another significant advantage of the proposed local method is that we can handle images of varying sizes without performing any resizing or cropping. Decreasing or increasing the size of the input image, or changing its aspect ratio, will change the size of the post-convolutional feature maps accordingly. This will change the number of network outputs, i.e., object predictions.

5 Optimizing precision: regression strategies

As illustrated in Table 1, the objects we want to detect in typical document analysis problems are smaller than objects in traditional scene object detection tasks, especially in the vertical direction. This means that the precision of the position predictions with respect to the image size is more important; a small error can cause the object to be missed. Moreover, our final objective is to recognize the text present in the text lines. Text recognition is very sensitive to localization errors, a horizontal error means that some letters will not be present in the image and thus will be lost for recognition. On the other hand, a vertical error means that the bottom or the top of the letters can be lost, which could make all the letters of the text line unrecognizable. For these reasons, localization precision is very important in the text line localization task our work is aimed at.

Table 2 Comparison of the F-measure scores for the detection of lower-left corners with respect to a given acceptance zone size for networks trained to detect lower-left corners, left sides or boxes

Acceptance zone size	0.003 (%)	0.01 (%)	0.03 (%)	0.1 (%)
Lower-left network, $P = 2$	10.7	57.4	85.7	91.7
Left-sides network, $P = 3$	11.2	58.4	87.0	92.6
Boxes network, $P = 4$	6.8	45.0	82.8	89.9

Acceptance zones are given as a proportion of page width

5.1 Strategy 1: bounding box regression

The first strategy is the obvious strategy employed by standard deep models in object detection and recognition: regression of the full bounding box coordinates (upper left corner and lower-right corner; or, alternatively, one corner plus width and height). This strategy is not optimal in the case of our local model with shared output regression. In particular, the ending of a text line is often outside of the convolutional receptive fields and predicting precisely this position is a difficult problem, which is entirely put as a burden on the recurrent layers. These context layers can decrease the problem but cannot completely solve it.

The problem can be confirmed experimentally. Table 2 shows that the lower-left corners of the text line bounding boxes are more precisely located when the network is trained to detect and localize only these lower-left corners (when $P = 2$ in Eq. 1) than when the network is trained to predict also the width and the height of the bounding boxes (when

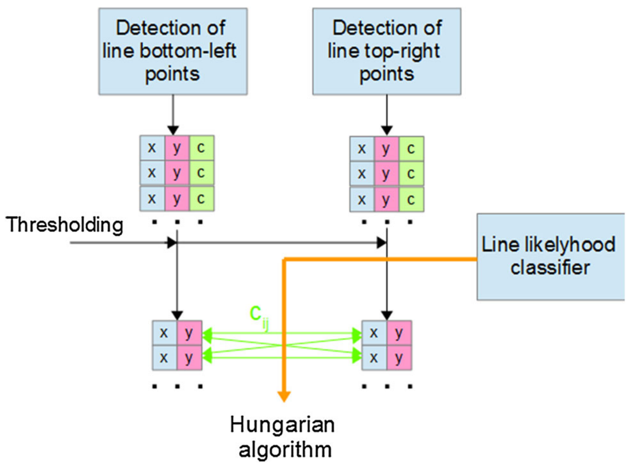
Detection of line bottom-left points
Detection of line top-right points
x y c
Thresholding
Line likelihood classifier
C_{ij}
x y
Hungarian algorithm

Fig. 4 Illustration of the point pairing process. Three different neural networks are used for, respectively, the detection of the lower-left corners of the text line bounding boxes, the detection of the upper-right corners, and for the computation of the pairing likelihoods

$P = 4$ in Eq. 1). In the following, we will propose two alternative strategies which address exactly this problem.

5.2 Strategy 2: detecting lower-left and upper-right points + pairing

The second strategy is illustrated in Fig. 4. Two different networks are responsible for, respectively, predicting the lower-left corners and the upper-right corners of the bounding boxes. These networks share the same architecture but not their trainable parameters. Each regressed entity/point fully falls into the receptive field of the network, easing the burden on the regressor.

The predicted lower-left corners and upper-right corners need to be paired to form the desired bounding box predictions. This pairing is performed by minimizing the following energy function with respect to discrete variable $z = \{z_{ij}\}$, which indicates that that lower-left corner $l_{ll}(i)$ is paired with upper-right corner $l_{ur}(j)$.

$$\hat{z} = \arg\min_{z} \sum_i \sum_j z_{ij} D(l_{ll}(i), l_{ur}(j))$$
$$\text{s.t.} \quad \forall j \sum_i z_{ij} \in \{0,1\}, \quad \forall i \sum_j z_{ij} \in \{0,1\} \qquad (3)$$

This constraint satisfaction problem can be solved using the Hungarian algorithm [29], which ensures that a given left corner candidate $l_{ll}(i)$ is not paired with several right corner candidates $l_{ur}(j)$ and vice versa.

The distance function $D(.,.)$ is responsible for evaluating if a left and a right corner can be paired, or in other words, to predict if the box formed by these two corners is likely to be a text line bounding box. We implement this distance as a third convolutional neural network made of 6 convolutional layers followed by a max-pooling and trained on a large number of images of pairs of left and right corners, some of which are part of the same text box, and some of which are not. The network takes as input the image inside the box formed by the two candidate points and it outputs a classification result of this being a line or not. The posterior probability of this decision (the soft-max output) corresponds to distance D.

Table 3 Comparison of the F-measure scores for the detection of the triplet of left-side positions with respect to a given acceptance zone size for networks trained to detect left sides or boxes

Acceptance zone size	0.003 (%)	0.01 (%)	0.03 (%)	0.1 (%)
Left-sides network, $P = 3$	4.2	47.2	84.8	92.4
Boxes network, $P = 4$	3.4	24.6	71.4	89.7

Acceptance zones are given as a proportion of page width

5.3 Strategy 3: detecting the left sides + text recognition

Using this corner detection and pairing technique can help to improve the precision of the results. But the system is more complicated because three different networks have to be trained only for the text line detection. Moreover, the processing time is higher because the detection is done twice and because the pairing distance network has to be run for each couple of lower-left and upper-right hypothesis corners. Finally, the pairing step itself is responsible for some of the failures.

Further experiments show that only the detection of the width of the text lines is damaging the precision of the prediction localizations. Detecting the height does not encounter the same issue. This is illustrated in Tables 2 and 3. This is probably due to the fact that the height of the text lines are often smaller and, consequently, lie inside the receptive fields.

A second alternative strategy, which avoids the combinatorial pairing process, involves detecting the lower-left corner of each text entity plus the height of the text box, but not its width. This is motivated by the fact, that the full-text height is often within the receptive field of each local predictor, but not the full width of the text box. Instead of detecting the text width separately, it is predicted jointly with the following *text recognition* step. In other words, we jointly recognize the text as well as where it ends.

The text recognizer is given as input an image with a left side defined by the localizer and a right side being the corresponding right border of the page image. It means that other text objects can be included in this right added part, the text recognizer is retrained in order to learn to ignore them. This process is illustrated in Fig. 5 and justified by the results

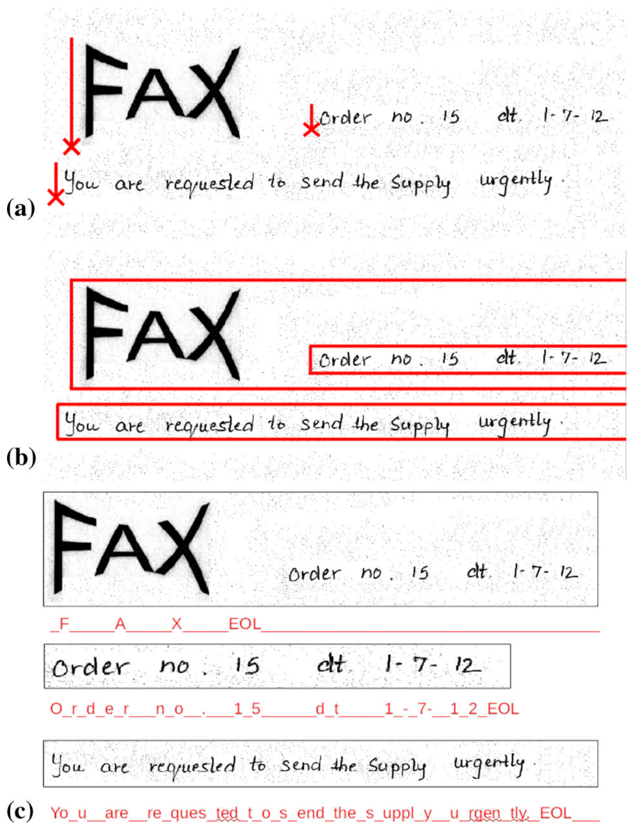
(a)
FAX
Order no. 15 dt. 1-7-12
You are requested to send the Supply urgently.
(b)
(c)
_F_____A_____X_____EOL
O_r_d_e_r___n_o__.___1_5______d_t______1_-_7-__1_2_EOL
Yo_u__are__re_ques_ted_t_o_s_end_the_s_uppl_y__u_rgen_tly._EOL___

Fig. 5 Description of the left-side localization technique. **a** Detection of the left-side position triplet objects. **b** Extraction of corresponding text lines. **c** Recognition of the text lines content with a recognizer trained to ignore text from other lines. The extra end-of-line characters (EOL) helps this training

Table 4 Text recognition Word Error Rates (WER) for networks trained/evaluated on reference boxes or box defined only by their left sides and extended to the right border of the page

	Evaluated on reference boxes (%)	Evaluated with left sides only (%)
Trained on reference boxes	9.0	46.7
Trained with left sides only	10.6	9.8

shown in Table 4 which show that a text recognizer network trained and evaluated on text boxes defined only by their left sides has a word error rate (9.8%) close to the error rate (9.0%) of the reference network trained and evaluated on the regular reference line bounding box images.

6 Training

The models are trained with stochastic gradient descent (SGD) using mini-batches of size 8 and dropout for regularization. During training, object predictors (network outputs) need to be matched to targets, i.e., to ground-truth object positions. Similar to the strategy in MultiBox [10], this is done globally over the entire image, which allows each object predictor to respond to any location in an image.

We denote by M the number of predicted objects, given as $M = I * J * K$, with I and J being the width and the height of the last feature map and K the number of predictors per feature map location; we denote by N the number of reference objects in the image. Matching is a combinatorial problem over the matching matrix X, where $X_{nm} = 1$ when hypothesis m is matched to target n, and 0 otherwise. For each forward-backward pass for each image, X is estimated minimizing the following cost function:

$$Cost = \sum_{n=0}^{N} \sum_{m=0}^{M} X_{nm} \left(\alpha \, \|l_m - t_n\|^2 - \log(c_m) \right) - (1 - X_{nm}) \log(1 - c_m) \tag{4}$$

where l_m is a vector of size P corresponding to a predicted location, c_m is the corresponding confidence, and t_n is a target object location which is of size P as well. The first term, euclidian, handles location alignment, while the second and third terms, logarithmic, favor high confidence objects. α is a weight between both.

Equation (4) is minimized subject to constraints, namely that each target object is matched to at most one hypothesis object and vice versa. This is a well known bi-partite graph matching problem, which can be solved with the Hungarian algorithm [29]. Please note, that this *detection/ground-truth matching* process is different from *left/right point pairing* process described in Sect. 4, although both are solved using the Hungarian algorithm.

Equation (4) gives the loss function used in the SGD parameter updates. However, we prefer to set different values of α for gradient updates and for matching. We found it important to increase α for the matching in order to help the network to use more diverse outputs.

As mentioned earlier, our matching strategy is similar to the one described in MultiBox [10] and has the same global property brought by the confidence term (albeit applied to local outputs, compared to the global outputs in [10]). On the other hand, in SSD [22] and YOLO [35], matching is done locally, i.e., predictors are matched to targets falling into spatial regions they are associated with. This is the reason for the spatial interpretation of the output grid shown in Fig. 2. Moreover, YOLO matches only one target location with each spatial cell, which leads to non-matched targets in the case of several objects with bounding box centers in the same cell. In our target application, where a large number of objects may be present, a large number of objects will not be matched to any predictor during training, as can be seen in the example in Fig. 6.

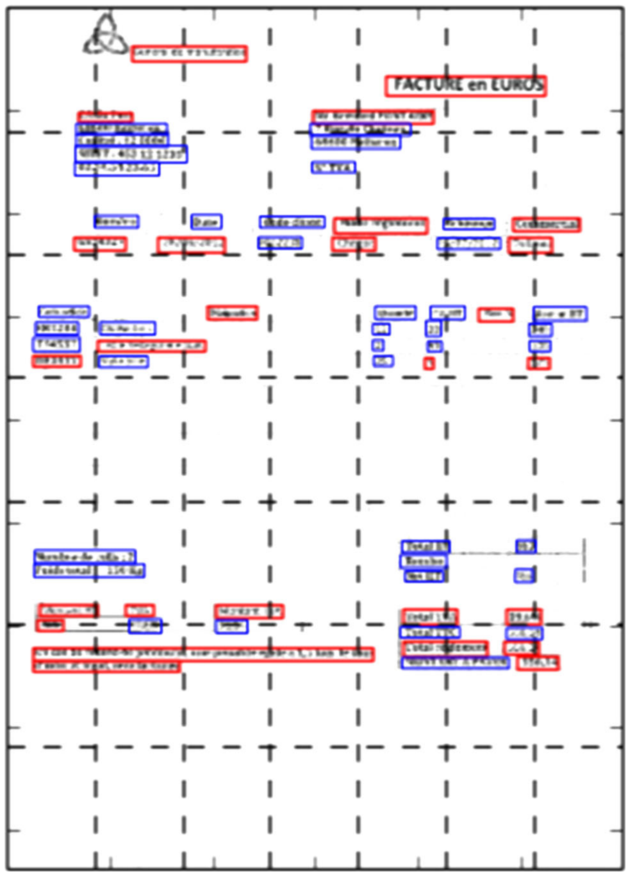

Fig. 6 YOLO matches target objects according to the spatial region in which they fall in the image, which leaves many targets unassigned (shown in blue). In applications with many objects, this makes the method non-applicable (color figure online)

In SSD and MultiBox, the matching process is restricted to a fixed dictionary of anchor locations obtained arbitrarily [22] or with clustering [10], which helps the network to create outputs specialized to regions in the image. This was proved unnecessary and even counter-productive in our case, where predictors share parameters spatially.

7 Experimental results

We tested the proposed model and the baselines on the publicly available Maurdor dataset [5]. This highly heterogeneous dataset is composed of 8773 document images (train:6592; valid:1110; test:1071) in mixed French, English and Arabic text, both handwritten and printed.

The dataset is annotated at paragraph level. For this reason, we use the technique detailed in [4] to get annotation at line level and we keep only the pages where we are confident that the automatic line position generation has worked well. We obtain a restricted dataset containing 3995 training pages, 697 validation pages and 616 test pages that are used for training, validation and test for the evaluation of intersection over union and DetEval metrics.

For the Bag of Word metric, we evaluate on the 265 pages fully in English and on the 507 pages fully in French of the full Maurdor test set in order to avoid the language classification task.

7.1 Metrics

We evaluate the performance of our method using three different metrics:

Intersection over union—IoU is a commonly used metric in object detection and image segmentation. It is given as the ratio of the intersection and the union of reference and hypothesis objects. Reference objects and hypothesis objects are matched by thresholding their IoU score. In most frequent versions, only one hypothesis can be associated with a reference box, the others are considered as error/insertions. Alternatively to reporting IoU directly, after thresholding IoU, an *F*-measure can be computed from Precision and Recall.

DetEval—DetEval [44] is the metric chosen for the ICDAR robust reading series of competitions. Its main advantage is that it allows many-to-many matchings between reference objects and hypothesis objects, which is important in applications where fragmentation of objects should be allowed (and eventually slightly punished), which is the case in text localization. Objects are assigned by thresholding overlap, and Precisions, Recall and *F*-measure are reported.

Bag-of-words recognition error—BoW is a goal-oriented method described in [34], which measures the performance of a subsequent text recognition module. The objective is to avoid the need of judging the geometrical precision of the result and to directly evaluate the performance of the goal of any localization method. In the case of the target application this is the subsequent recognizer.

We use the recognition model from [31], which is based on deep convolutional networks and spatial/sequence modeling with 2D-LSTM layers. Assigning character labels to network outputs is performed with the Connectionist Temporal Classification framework [16]. Recent follow-up work solves this problem with attention-based mechanisms [3], this will be investigated in future work. The recognizer is trained on both handwritten and printed text lines, separately on English and French text. We apply them on crops of localized bounding boxes. The Bag of Word metric, on the contrary to metrics based on the Levenshtein distance enables to avoid an alignment that can be ambiguous at page level. Word insertion and deletions are computed at page level and *F*-measure is reported.

7.2 Baselines

Traditional text segmentation methods—For comparison, we used two techniques based on image processing (w/o machine learning) for document text line segmentation. Shi et al. [39] use steerable directional filters to create an adaptive local connectivity map. Line locations are given by the positions of the connected components extracted from the binarization of this connectivity map. These positions are refined with heuristic-based post-processing. The method proposed by Nicolaou et al. [30] follows the whitest and blackest paths in a blurred image to find lines and interlines.

Machine Learning-based methods: Yolo and MultiBox—For YOLO, we used two classes for the object classification part of the model, handwritten text lines and printed text lines. It helped the model to learn better than with only one class.

Both systems were tested in two different configurations: the original architecture tuned for large-scale image recognition, and an architecture which we optimized for our task on the validation set. In particular, the size of the filters was adapted to the shape of the objects. Hyper-parameter tuning led in both cases to architectures with heavily reduced numbers of layers and less units per layer. We also optimized learning rates and minibatch sizes.

7.3 Architectures

The network architecture of the proposed model has been tuned to correspond to our task. The found hyper-parameters are detailed in Table 5. The width and height of the feature maps are given for illustration but it can of course vary. In particular, the aspect ratio of the image can vary. We would like to stress again that the number of parameters is independent of the actual size of the input image.

The inputs of our network are raw gray-scaled images with width normalization. The use of color images was not improving the results on our task.

Note that the number of weights in our last layer, the position prediction layer, is rather small : 3,700. To be able to predict the same number of objects, with the same number of input features, MultiBox [10] and Yolo [35] would have needed 15,688,200 parameters.

For training, we used a learning rate of 10^{-4} and mini-batches of size 8. Dropout with 0.5 probability is applied after each 2D-LSTM layer. The α parameter in Eq. (4) is set to 1000 for matching and to 100 for weight updates during SGD.

We experimentally found that resolution reduction between layers works better using strides > 1 of the convolutional layers instead of max-pooling. This can be explained by our need for precision, while max-pooling is known to lead to shift invariance.

Table 5 Network architecture/hyper-parameters

Layer	Filter size	Stride	Size of the feature maps	Number of parameters
Input	–	–	$1 \times (598 \times 838)$	
C1	4×4	3×3	$12 \times (199 \times 279)$	204
LSTM1	–	–	" "	8880
C2	4×3	3×2	$16 \times (66 \times 139)$	2320
LSTM2	–	–	" "	15,680
C3	6×3	4×2	$24 \times (16 \times 69)$	6936
LSTM3	–	–	" "	35,040
C4	4×3	3×2	$30 \times (5 \times 34)$	8670
LSTM4	–	–	" "	54,600
C5	3×2	2×1	$36 \times (2 \times 33)$	6516
Output	1×1	1×1	$5 \times 20 \times (2 \times 33)$	3700

The input and feature map sizes are an illustrative example. The number of parameters does *NOT* depend on the size of the input image

Table 6 Detection performance: *F*-measure with various thresholds(T) on IoU

Method	*F*-measure		
	$T = 0.3$ (%)	$T = 0.5$ (%)	$T = 0.7$ (%)
Shi et al. [39]	40.7	31.1	21.1
Nicolaou et al. [30]	36.1	26.3	15.9
Multibox [10]	11.3	2.1	0.2
Multibox [10] (optimized)	48.7	23.0	5.2
Boxes, no LSTMs	49.9	23.7	5.3
Boxes	73.8	43.6	14.1
Points and pairing	69.1	45.1	18.2

On the restricted Maurdor test set (616 pages)

7.4 Results and discussion

Localization results on the restricted Maurdor test set, for our box detection and our points detection plus pairing system alongside with baselines and a reference object detection method, are shown with the IoU metric in Table 6 and with the DetEval metric in Table 7. The left-side detection method is not shown in these tables because it does not give the geometric position of the boxes explicitly. **Text recognition results** (on text objects localized by our three methods), evaluated with the Bag of Word metric, are shown in Table 8, respectively, for pages fully in French and fully in English of the whole Maurdor test set.

For the IoU metric, results are given in Table 6. We report *F*-measure for different thresholds on IoU, i.e., for different localization quality requirements. The image-based techniques Shi et al. [39] and Nicolaou et al. [30] perform poorly when the threshold is low, i.e., when we are interested in the ability of the system to detect all the boxes regardless of the

Table 7 Detection performance with DetEval [44]

Method	Recall (%)	Precision (%)	*F*-Meas. (%)
Shi et al. [39]	35.1	38.4	36.7
Nicolaou et al. [30]	46.7	39.6	42.9
Multibox [10]	4.2	10.0	6.0
Multibox [10] (optimized)	28.8	52.3	31.1
Boxes, no LSTMs	28.6	52.4	31.1
Boxes	51.2	61.4	55.9
Points and pairing	54.2	58.6	56.3

On the restricted Maurdor test set (616 pages)

Table 8 Detection and recognition performance: word recognition *F*-measure in BOW mode on the full English or French Maurdor test set

Method	French (507 pages) (%)	English (265 pages) (%)
Shi et al. [39]	48.6	30.4
Nicolaou et al. [30]	65.3	50.0
Multibox [10]	27.2	14.8
Multibox [10] (optimized)	32.4	36.2
Boxes, no LSTMs	57.8	56.9
Boxes	71.2	71.1
Points and Pairing	71.7	72.3
Left sides + joint recognition	79.9	79.1

exact location. They suffer from low general recall. However, they are relatively precise. *F*-measure drops less than the learning-based methods when the precision requirements are increased by increasing the threshold on IoU. This can be explained by the nature of these algorithms, which proceed by binarization of the input images. In the normal operating range of these algorithm, when the segmentation steps work out well, precision is almost guaranteed to be high. However, once images don't fall into the situations the algorithms have been tuned for, performance breaks down.

On the other hand, methods based on direct regression as MultiBox [10] and our proposed method are more robust and achieve better general recall, an advantage which is bought with a slight drop in precision. Our proposed methods give the best results for realistic thresholds.

We can also see in Table 6 that the point detection and pairing of left and right corners method is better than the box detection method when we use a high threshold value. It confirms that this method is better when preciseness is needed. On the contrary, it is worse for a small IoU threshold of 0.3. That can be due to the errors in the pairing process.

The DetEval metric results shown in Table 7 confirm the results from the intersection over union metric.

From the application perspective, namely the full-page text recognition in documents, Table 8 shows that the proposed methods deliver good results with over 70% *F*-measure Bag of Word score for the box detection method and nearly 80% for the left-side detection system, on both French and English. It outperforms all the other methods. This can be explained by its high recall, while the similar precision of image-based methods is not an advantage since a recognizer can compensate for it, up to a certain limit.

In Tables 6, 7 and 8, results for the direct detection of boxes are given with and without recurrences in the model. It stressed the importance of adding 2D-LSTM layers to recover information as it significantly improves performances for all the metrics. The power of the 2D-LSTM layers can also be shown in Fig. 7, which gives some example detections of the box detection system. Figure 7a, c shows that the model is capable of detecting objects which are larger than the receptive fields of the individual bounding box predictors. This is made possible through the context information gathered by the LSTM layers. Figure 7c, f shows that the system is capable of detecting and locating a large number of small objects.

Figure 8 shows the results of the left-side detection model and the results can be compared to those of Fig. 7. We can see that, especially when small objects have to be detected as in Fig. 7c, f, the preciseness of the position predictions is improved with this left-side detection. Good results are shown on a very heterogeneous set of documents, including forms, letters, maps or newspaper pages. Both for French and English, both for Printed and Handwritten texts.

Multibox [10] is significantly outperformed by our method, even if we optimize its hyper-parameters (on the validation set). We attribute this to the fact, that the output layers are not shared. The model needs to express similar prediction behavior for each output, thus relearn the same strategies several times.

YOLO [35] proved to be impossible to apply to this kind of problem, at least in its current shape. As reported by the authors, the system gives excellent results on the tasks it has been designed for. However, despite extensive tuning of its hyper-parameters, we were not able to reach satisfying results, although we worked with two different implementations: the original implementation of the authors, as well as our own implementation. We did identify the problem, however. YOLO has been designed for a small number of objects, with a predictor/target matching algorithm adapted to these settings (see also Sect. 6). As mentioned, only one target can be associated with each spatial cell, which is a harmless restriction for traditional object detection tasks. However, this is a real problem in our case, as shown in the example image in Fig. 6. A large part of the ground-truth objects in most figures will not be assigned to any predictor, and not trained for. Not only are these boxes missing at training, net-

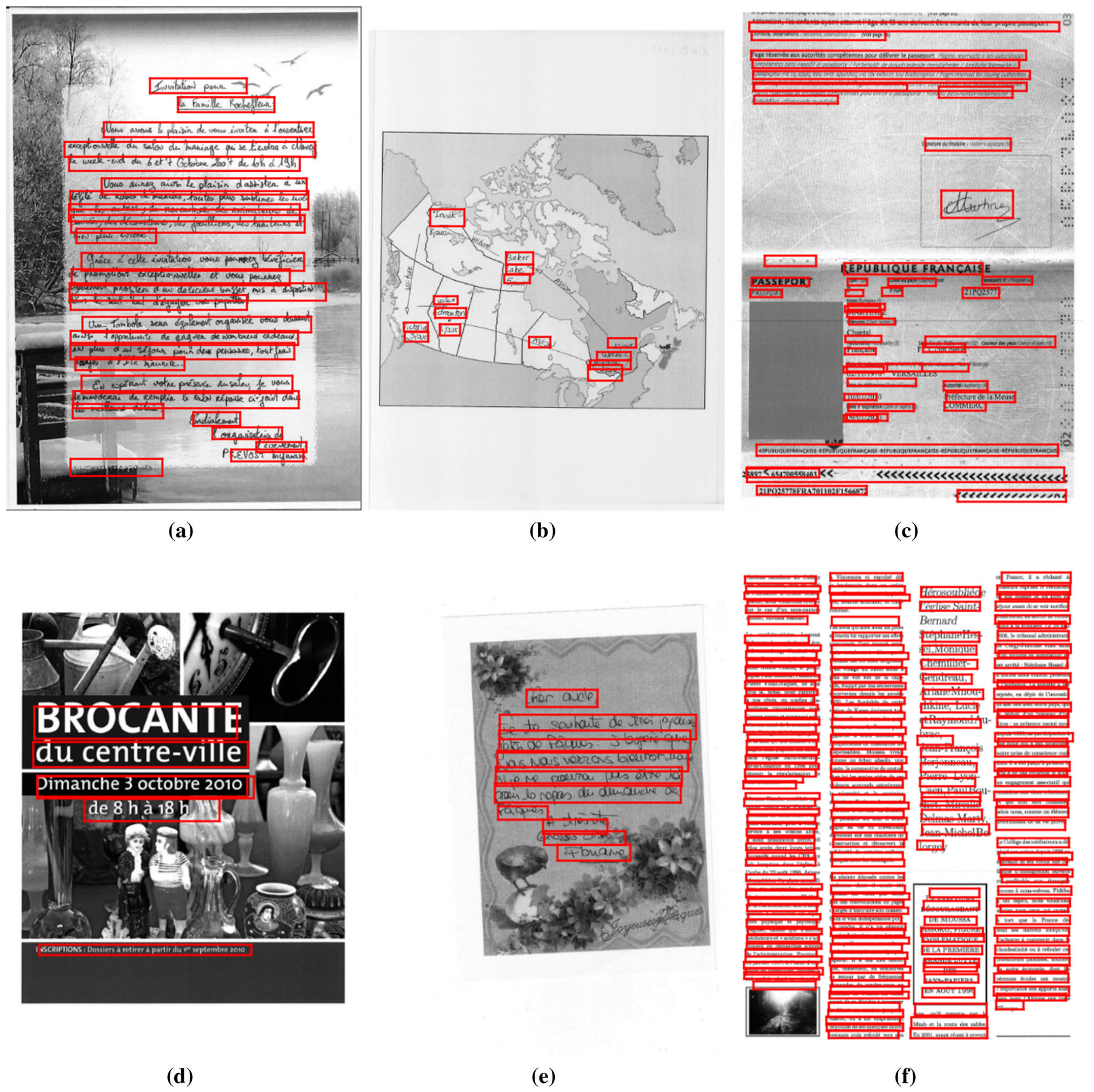

Fig. 7 Samples of results obtained with the box detection method on images of the Maurdor test set. The actual inputs are shown

work outputs predicting their locations will be punished at the next parameter update, further hurting performance and hindering the networks from converging properly.

7.5 Implementation

No deep learning framework was used for the implementation of the proposed method, since, until recently and the Theano version from Doetsch et al. [9], no 2D-LSTMs implementation was, up to our knowledge, yet existing in Tensorflow, Torch, Theano or Caffe. The system has been implemented using our in-house framework implemented in C++, including the SGD optimizer and dropout. For this reason also, the model has been trained on CPUs.

The three detection systems that we propose in this paper and that, respectively, detect points, left sides or boxes have approximately the same computing time because most of the time is spent on the first layers of the network, because our last layer is only locally connected and because the size of the last feature map is small.

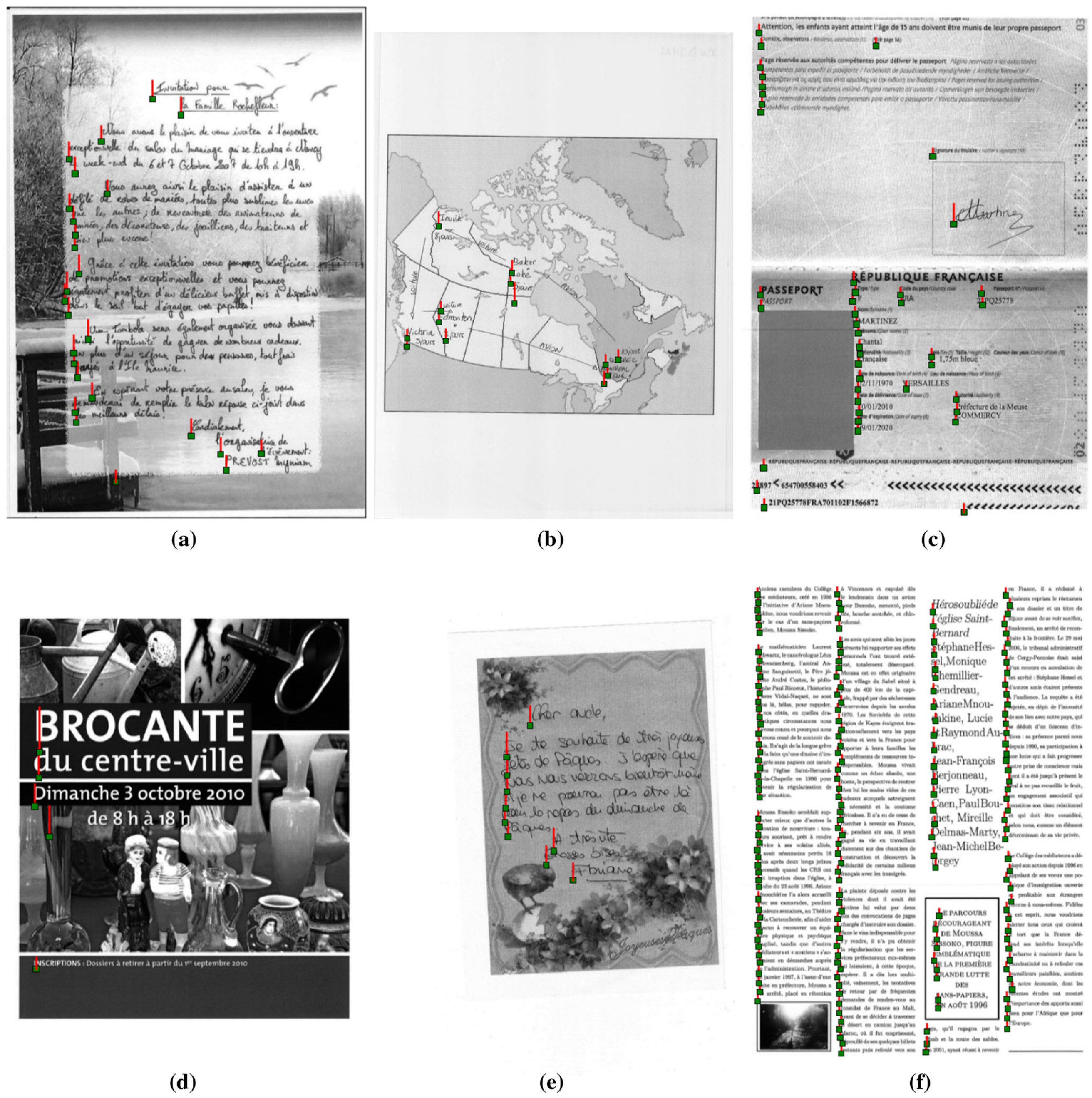

Fig. 8 Samples of results obtained with the left-side detection method on images of the Maurdor test set. The actual inputs are shown

For an image of size 598 × 838, our model took a mean decoding time of 245 ms (64 ms without recurrent layers) while the Multibox network (initial architecture) took 453 ms. All performances are CPU only on Intel Xeon E5-2640-v4 with 64 GB of RAM.

For YOLO we used two different implementations. We implemented and trained our own implementation in Tensorflow, and we also used the official source code published by the authors.[2]

[2] http://pjreddie.com/darknet/yolo.

8 Conclusion

We presented a new fully convolutional model for the detection and localization of a potentially large number of objects in images. To optimize invariance and in order to limit the number of trainable parameters, we shared parameters of the output layer over spatial blocks of the image, implementing the output layer as 1×1 convolution. To deal with objects which are larger than the receptive fields, and in order to allow the model to collect features from the global context, we added 2D-LSTM layers between the convolutional layers.

We proposed three different strategies based on the detection of boxes, corners or left sides in order to improve the precision of the detections. We compared the proposed models to the state of the art in object detection, in particular to Multibox [10] and YOLO [35] that does not converge on our data. We measured detection performance and word recognition performance of a subsequent text recognition system. Our experiments showed, that the proposed model significantly outperforms both methods, even if their hyper-parameters are optimized for the targeted configurations. Good results have been obtained on a broad range of documents.

References

1. Behnke, S.: Face localization and tracking in the neural abstraction pyramid. Neural Comput. Appl. **14**(2), 97–103 (2005)
2. Bell, S., Zitnick, L., Bala, K., Girshick, R.: Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas (2016)
3. Bluche, T.: Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In: Advances in Neural Information Processing System, Barcelona (2016)
4. Bluche, T., Moysset, B., Kermorvant, C.: Automatic line segmentation and ground-truth alignment of handwritten documents. In: International Conference on Frontiers in Handwriting Recognition, Crete (2014)
5. Brunessaux, S., Giroux, P., Grilheres, B., Manta, M., Bodin, M., Choukri, K., Galibert, O., Kahn, J.: The maurdor project—improving automatic processing of digital documents. In: Document Analysis Systems, Tours (2014)
6. Chen, K., Seuret, M., Hennebert, J., Ingold, R.: Convolutional neural networks for page segmentation of historical document images. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, Kyoto, vol. 1, pp. 965–970. IEEE (2017)
7. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: object detection via region-based fully convolutional networks. In: Advances in neural information processing systems, Barcelona, pp. 379–387 (2016)
8. Delakis, M., Garcia, C.: Text detection with convolutional neural networks. In: International Conference on Computer Vision Theory and Applications, Madeira, pp. 290–294 (2008)
9. Doetsch, P., Zeyer, A., Voigtlaender, P., Kulikov, I., Schlüter, R., Ney, H.: Returnn: The rwth extensible training framework for universal recurrent neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, New Orleans, pp. 5345–5349. IEEE (2017)
10. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition, Colombus (2014)
11. Eskenazi, S., Gomez-Krämer, P., Ogier, J.M.: A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognit. **64**, 1–14 (2017)
12. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
13. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
14. Girshick, R.: Fast R-CNN. In: International Conference on Computer Vision, Santiago (2015)
15. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, Colombus (2014)
16. Graves, A., Fernandez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: International Conference on Machine Learning, Pittsburgh (2006)
17. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing System, Vancouver (2008)
18. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas (2016)
19. Iandola, F., Hand, S., Moskewicz, M., Ashraf, K.: Squeezenet: Alexnet-level accuracy with 50× fewer parameters and <0.5MB model size. In: Openreview submission to ICLR 2017, Toulon (2016)
20. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. Int J Comput Vis **116**(1), 1–20 (2016)
21. Likforman-Sulem, L., Zahour, A., Taconet, B.: Text line segmentation of historical documents: a survey. Int J Doc Anal Recognit **9**(2–4), 123–138 (2007)
22. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.: Ssd: single shot multibox detector. In: European Conference on Computer Vision, Amsterdam (2016)
23. Liu, Y., Jin, L.: Deep matching prior network: toward tighter multi-oriented text detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, vol. 2, p. 8 (2017)
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, Boston (2015)
25. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
26. Ma, J., Shao, W., Ye, H., Wang, L., Wang, H., Zheng, Y., Xue, X.: Arbitrary-oriented scene text detection via rotation proposals. IEEE Transactions on Multimedia (2018)
27. Messelodi, S., Modena, C.M.: Automatic identification and skew estimation of text lines in real scene images. Pattern Recogn. **32**(5), 791–810 (1999)
28. Mordan, T., Thome, N., Cord, M., Henaff, G.: Deformable part-based fully convolutional network for object detection. In: British Machine Vision Conference (BMVC), London (2017)
29. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)
30. Nicolaou, A., Gatos, B.: Handwritten Text Line Segmentation by Shredding Text into its Lines. In: International Conference on Document Analysis and Recognition, Barcelona (2009)
31. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. In: International Conference on Frontiers in Handwriting Recognition, Crete (2014)
32. Pinheiro, P., Collobert, R.: From image-level to pixel-level labeling with convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, Boston (2015)
33. Pinheiro, P., Lin, T., Collobert, R., Dollar, P.: Learning to refine object segments. In: European Conference on Computer Vision, Amsterdam (2016)
34. Pletschacher, S., Clausner, C., Antonacopoulos, A.: Europeana newspapers ocr workflow evaluation. In: Workshop on Historical Document Imaging and Processing, Nancy (2015)
35. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas (2016)

36. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing System, Montreal (2015)
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
38. Ryu, J., Koo, H.I., Cho, N.I.: Language-independent text-line extraction algorithm for handwritten documents. Signal Process. Lett. **21**(9), 1115–1119 (2014)
39. Shi, Z., Setlur, S., Govindaraju, V.: A Steerable Directional Local Profile Technique for Extraction of Handwritten Arabic Text Lines. In: International Conference on Document Analysis and Recognition, Barcelona (2009)
40. Stafylakis, T., Papavassiliou, V., Katsouros, V., Carayannis, G.: Robust text-line and word segmentation for handwritten documents images. In: IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, pp. 3393–3396. IEEE (2008)
41. Stewart, R., Ermon, S.: Label-free supervision of neural networks with physics and domain knowledge. In: AAAI, San Francisco, pp. 2576–2582 (2017)
42. Szegedy, C., Reed, S., Erhan, D., Anguelov, D.: Scalable, high-quality object detection. arXiv:1412.1441 (2015)
43. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems, Barcelona (2016)
44. Wolf, C., Jolion, J.M.: Object count/area graphs for the evaluation of object detection and segmentation algorithms. Int. J. Doc. Anal. Recognit. **8**(4), 280–296 (2006)
45. Wonmin, W., Breuel, T., Raue, F., Liwicki, M.: Scene labeling with LSTM recurrent neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition, Boston (2015)
46. Yin, F., Liu, C.L.: Handwritten chinese text line segmentation by clustering with distance metric learning. Pattern Recogn. **42**(12), 3146–3157 (2009)
47. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: International Conference on Learning Representations, San Juan (2016)
48. Zhang, Z., Zhang, C., Shen, W., Yao, C., Liu, W., Bai, X.: Multi-oriented text detection with fully convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas (2016)