Inductive Visual Localisation: Factorised Training for Superior Generalisation

Ankush Gupta
Andrea Vedaldi
Andrew Zisserman
{ankush,vedaldi,az}@robots.ox.ac.uk

Visual Geometry Group
Department of Engineering Science
University of Oxford

Abstract

End-to-end training of Recurrent Neural Networks (RNNs) have been successfully applied to numerous problems that require processing sequences, such as image captioning, machine translation, and text recognition. However, RNNs often struggle to generalise to sequences longer than the ones encountered during training. In this work, we propose to optimize neural networks explicitly for *induction*. The idea is to first decompose the problem in a sequence of inductive steps and then to explicitly train the RNN to reproduce such steps. Generalization is achieved as the RNN is not allowed to learn an arbitrary internal state, but is rather tasked with mimicking the evolution of a valid state. Here, in particular, the state is a spatial memory that keeps tracks of which parts of the image have already been visited by the RNN in previous steps. The RNN is trained to update the memory in addition to producing the desired output. We evaluate our method on two different visual recognition problems involving sequences: (1) text spotting, *i.e.* joint localisation and reading of text, in images containing multiple lines (or a block) of text, and (2) sequential counting of objects in aerial images. We show that inductive training of recurrent models enhances their generalisation ability on challenging image datasets.

1 Introduction

A key issue in sequence and program learning is to model long-term structure in the data. For example, in language modelling one has two choices: The first is to consider simple models such as character and word n-grams, which generalise well but fail to capture long-term correlations in the data. The second is to switch to models such as Recurrent Neural Networks (RNNs) that, in principle, can capture arbitrarily long correlations. In practice, however, RNNs are trained using back-propagation through time on sequences of limited length and may fail to generalize to longer sequences [12, 17, 21, 32].

This is in contrast with the standard and very familiar notion of mathematical induction, which allows sequences to be analysed or generated ad infinitum. Many problems, such as counting objects or reading text, have an inherent inductive structure: all one needs to do is 1) count the current object or read the current character; and 2) move to the next object or character (or stop when finished). The two steps can then be iterated to process data of arbitrary length. However, as noticed by several authors [12, 17, 21, 32], and confirmed in our experiments, RNNs fail to correctly repeat these steps beyond the number of times considered during training.
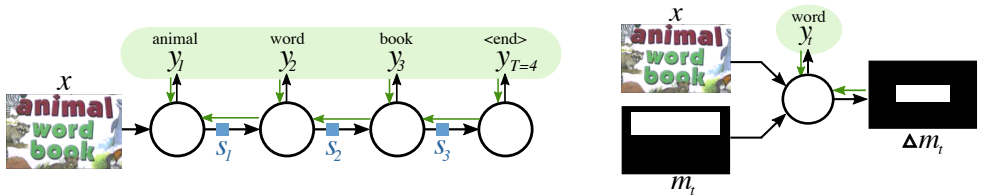
Figure 1: **Inductive Visual Localisation.** A recurrent neural network (RNN) for sequence recognition is trained end-to-end on sequences $(y_1, \ldots, y_T)$. The internal states $(\mathbf{s}_t)$ are learnt from gradients of losses summed over the entire sequence. Without structuring the state space, the recurrent unit may fail to learn the appropriate loop invariant, and thus may be unable to generalise to sequences longer than the length T used for training. We address this problem by decomposing the end-to-end training procedure (left) into one-step inductive updates (right). We achieve this by restricting the recurrent state to a spatial memory map $\mathbf{m}_t$, which keeps track of the progress in processing the sequence. The recurrent network learns to incrementally predict, in addition to the output sequence $\mathbf{y}_t$, updates $\Delta \mathbf{m}_t$ to the memory. Using this inductive decomposition, our network can generalize well to sequences of length far greater than those in the training set.

In this paper we marry the idea of induction to sequence processing using recurrent neural networks (RNN). The inductive approach in this paper can be viewed as an application to the spatial domain of recent approaches for learning programs using recurrent and compositional networks [7, 32].

Our contributions are threefold: 1) we propose to train recurrent networks with the explicit notion of *induction*, where the end-to-end training procedure is decomposed into inductive sub-steps. We show that RNNs trained on one-step inductive updates have superior generalisation ability. 2) We develop a recurrent module with inductive factorisation for recognising multiples lines of text in challenging scene text images, and outperform state-of-the-art methods by combining text localisation and recognition into a single architecture. 3) We apply our approach to sequential visual counting, and validate it on a challenging aerial image dataset, once again demonstrating improved generalisation capabilities.

The rest of the paper is organised as follows: in section 2 we first review related work; next, in section 3 we introduce our inductive decomposition approach; finally, in section 4 we present results on recognising multiple lines of text in images containing blocks of text, and on sequential visual object counting.

2   Related Work

Learning & Composing Programs.   Explicit decomposition into the repeated sub-tasks is related to the recent work on neural controllers, which interface with sub-programs or modules. Neural Programmer-Interpreters [32] present a general framework for dispatching functions using a program-stack, which Cai *et al*. [7] augment with the explicit notion of recursion, showing superior generalisation. Zaremba *et al*. [46], learn controllers for operators acting on coarse 2D grids of discrete symbols, while our method works on dense pixel-grids.

RNNs for Scene Text Recognition   Scene text recognition is a well-studied problem, with roots in optical and handwritten character recognition [24]. Traditional methods focused on single character recognition [8, 44] with explicit language models (e.g., n-grams or lexicons) to form words or sentences [1, 19, 26, 28, 30, 36, 40, 41]. More recently, the word-level text

recognition has been explored extensively [11, 13, 20, 31], primarily using Convolutional Neural Networks (CNNs) [24] to encode images, and RNNs as the decoders [15, 25, 34, 35, 37]. A key component of the RNN decoders is soft-attention [5] which iteratively pools image features at each step of the recurrence [25, 35]. [6] extend this attention to 2D feature-maps for recognising multiple-lines of handwritten text.

RNNs for Visual Object Counting. Counting objects in images [3] has numerous applications *e.g.*, histological analysis of micoscopy images [2], parsing medical scans, and population studies from aerial imagery [4]. Sequential counting has been shown to be the primary method of counting in humans [10], and was explored with convolutional-RNNs in [53]. More recently, [48] combine the object density based regression methods [27] with iterative counting using recurrent fully-convolutional networks. We factorise the end-to-end training of such iterative counting methods, and achieve superior generalisation.

3 Method

In order to generalise correctly to sequences of arbitrary lengths, an iterative algorithm must maintain a suitable invariant. For example, an algorithm that counts objects via enumeration maintains as invariant the list of objects visited so far, which must contain no repetition. In order to maintain this invariant, the algorithm must visit at each step a "new" object, or terminate if no more objects are available. However, RNNs are trained without any explicit constraints on the structure of their hidden state, and may not learn such an invariant correctly. For example, a list of objects has no a-priori limitation on its size. While for a human this is obvious, an RNN may be unable to understand it as it cannot experience unbounded lists during training.

In order to address this issue, we design an RNN to update a state that, by construction, has a universal step-independent validity. In particular, we restrict the recurrent state to a *spatial memory map* which keeps track of which part of the input image has already been explored. At each step, the model is conditioned on this memory map (in addition to the input image), and predicts as output a token for the sequence, as well as an update to the spatial memory. Such updates for the counting example above amount to adding one more object to the list of visited objects. This is analogous to taking the *inductive step* in mathematical induction.

The rest of the section describes our model in detail. In section 3.1 we discuss encoder-decoder RNNs [9, 38, 39] enhanced with soft-attention [5, 43] as these are suitable for modelling sequences in images. In section 3.2 we describe our inductive decomposition for these recurrent models. Finally, in section 3.3 we detail training and inference.

3.1 Encoder-Decoder Models

Let $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{H \times W \times C}$ be an image, where H, W and C are its height, width and number of color channels. Furthermore, let $\mathbf{y} = (y_1, \ldots, y_T) \in \mathcal{Y}^T$ be the corresponding sequence label, where T is the sequence length. Encoder-decoder methods model the conditional probability $p(\mathbf{y}|\mathbf{x})$ as a product of conditionals for the next token y_{t+1} given a context vector $\mathbf{c}_t$ and the previously-predicted tokens $y_{1:t}$:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=0}^{T-1} p(y_{t+1}|y_{1:t}, \mathbf{c}_t) \tag{1}$$

These conditional probabilities are modelled using an RNN Φ. We consider in particular an LSTM [17] with hidden state $\mathbf{s}_t$, and write:

$$\Big(p(y_{t+1}|y_{1:t}, \mathbf{c}_t), \ \mathbf{s}_{t+1} \Big) = \Phi(\mathbf{s}_t, \mathbf{c}_t, y_t).$$

The context vector $\mathbf{c}_t$ injects into the model information extracted from the input image. Context can be kept constant for all steps, or can be dynamically focused on different parts of the image using an attention mechanism. In the first case, exemplified by sequence-to-sequence models [39], the context is extracted[1] by a Convolutional Neural Network (CNN) [23] $\mathbf{c}_t = \mathbf{c} = \Psi(\mathbf{x}) \in \mathbb{R}^{H' \times W' \times C'}$. In the second case, exemplified by [5], the context vector is computed at each step via attention by reweighing the CNN output:

$$\mathbf{c}_t = \sum_{i \in H'} \sum_{j \in W'} \alpha_{ij} \Psi(\mathbf{x})_{ij}, \tag{2}$$

$$\alpha_{ij} = \frac{\exp(v_{ij})}{\sum_{i'} \sum_{j'} \exp(v_{i'j'})}, \tag{3}$$

$$v_{ij} = w^T \tanh(W\mathbf{s}_t + W'\Psi(\mathbf{x})_{ij} + b), \tag{4}$$

where, $v_{ij} \in \mathbb{R}$ is the unnormalised attention score and w, W, W', b are learnable parameters.

The model is trained end-to-end to maximize the log of the posterior probability (1) averaging over example (image $\mathbf{x}$, sequence $\mathbf{y}$) pairs.

3.2   Inductive Decomposition

The RNN models discussed in the previous section may fail to learn a correct inductive decomposition of the problem, and thus fail to generalize properly to sequences of arbitrary length. We propose to address this problem in a simple and yet effective manner: rather than allowing the RNN to learn its own state space, we specify a suitable state space a priori and train the RNN to make use of it. In more detail, we set the RNN state to be a spatial memory $\mathbf{s}_t = \mathbf{m}_t$ containing a mask covering all the visual objects that have been accounted for up to time t. In this manner, the content of the memory can be derived from the ground-truth data annotations and the step number t. Furthermore, the RNN does not need to learn a new state space from scratch, but only how to generate $\mathbf{m}_{t+1}$ from $_t$; for this, training can focus on learning single-step predictions, from t to $t + 1$, rather than whole-sequence predictions.

In more detail, the spatial memory $\mathbf{m} \in \mathbb{R}^{H \times W}$ is implemented as a single 2D map of the same dimensions as the image $\mathbf{x}$. At each step, the model predicts y_t, as well as an update $\Delta \mathbf{m}_t$ to the memory. In this work, we focus on sequence prediction tasks where each token in the sequence corresponds to a 2D location in the image. Hence, $\Delta \mathbf{m}_t$ is trained to encode the 2D location in the image associated with y_t.

Predictions at each step are conditioned on the context vector $\mathbf{c}_t$, as well as the memory $\mathbf{m}_t$:

$$\Big(p(y_{t+1}|y_{1:t}, \mathbf{c}_t), \ \Delta \mathbf{m}_{t+1} \Big) = \Phi(\mathbf{c}_t, \mathbf{m}_t). \tag{5}$$

In practice, at each step, the memory is concatenated with the image to obtain the encoded representation $\Psi([\mathbf{x}||\mathbf{m}])$ instead of being fed into Φ directly; $\mathbf{c}_t$ is obtained from Ψ as detailed in section 3.1. The memory is initialised to all zeros, i.e., $\mathbf{m}_{t=0} = 0^{H \times W}$, and is updated after each step as:

$$\mathbf{m}_{t+1} = \mathbf{m}_t + \Delta \mathbf{m}_t. \tag{6}$$

---

[1]Since in this case no attention is used, the CNN is usually configured so that $H' = W' = 1$.
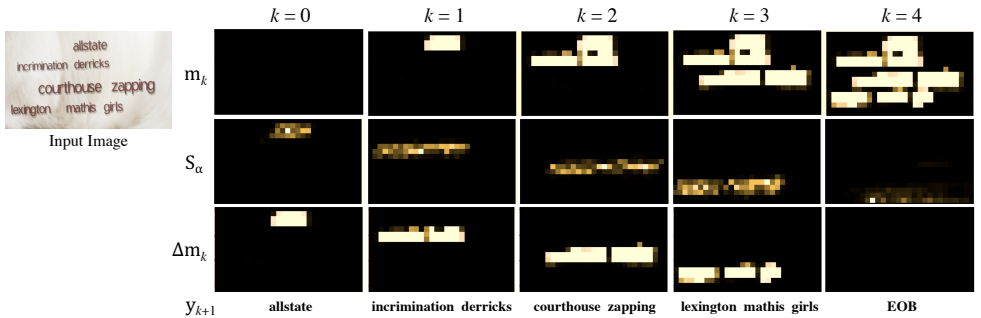
Figure 2: **Multi-line text recognition inference visualisation.** Decoding procedure for an example image containing four lines. Inference runs for five steps, predicting the line characters (y_k) in the first four steps, and indicating the end-of-block (EOB) in the fifth step. The memory-map is initialised (at $k=0$) to all zeros, and is iteratively updated by adding in the predicted update $\Delta\mathbf{m}_k$, regressed from the accumulated attention maps S_α.

The exact architecture of Φ and the location representation in $\mathbf{m}_t$ are application dependent, and examples are detailed in sections 4.1.1 and 4.2.1.

3.3 Training and Inference

Training. The model is trained for one-step predictions, where each training sample is a tuple — $(\mathbf{x}, y_t, \mathbf{m}_t, \mathbf{m}_{t+1})$: image, token at time t, ground-truth accumulated location maps $\mathbf{m}_t$ and $\mathbf{m}_{t+1}$. The model is optimised through stochastic gradient descent (SGD) to minimise the following loss:

$$-\log p(y_t|\mathbf{x}_t, \mathbf{m}_t) + \gamma||\mathbf{m}_t + \Delta\mathbf{m}_t - \mathbf{m}_{t+1}||_2^2 \qquad (7)$$

where, $\gamma > 0$ balances the terms. The first term maximises the probability of the correct token y_t, while the second is a pixel-wise reconstruction loss for the predicted spatial memory update $\Delta\mathbf{m}_t$. The sequence label for the final step (y_T) indicates the end-of-sequence, *e.g.* through an additional class in the output labels; it is used for terminating the inference loop.

Optimisation. All model parameters are initialised randomly (sampled from a gaussian with 0.01 standard deviation). The model is trained with SGD using the AdaDelta optimiser [47].

Inference. The memory is initialised to all zeros. At each step, the memory is concatenated with the image and fed through the image encoder to get the encoded-representation $\Psi([\mathbf{x}||\mathbf{m}])$. Then, the log-probabilities for y_t, and memory updates $\Delta\mathbf{m}_t$ are regressed from the recurrent module Φ. The memory is updated per eq. (6), and fed into the model iteratively, until the end-of-sequence is predicted. An example is shown in fig. 2.

4 Experiments

We evaluate our method on two different tasks – in section 4.1 we present results on recognising lines of text in images containing a block or multiple lines of text, and in section 4.2 we explore counting objects in images through enumeration. We demonstrate superior generalisation ability of recurrent modules when trained with inductive factorisation, and outperform state-of-the-art methods on a text-spotting task.

	# lines→	1	2	3	4	5	6	7	8	9	10
end-to-end	precision	66.69	63.97	59.23	53.70	-	-	-	-	-	-
	recall	69.27	65.50	56.52	39.14	-	-	-	-	-	-
	ED	15.91	17.81	25.08	43.78	-	-	-	-	-	-
inductive	precision	85.13	84.79	85.57	87.25	87.32	86.11	85.41	85.51	84.57	84.41
	recall	84.89	84.74	85.32	86.99	87.22	85.91	84.43	84.03	80.47	76.80
	ED	6.76	7.79	7.09	6.29	5.77	6.96	8.77	9.11	13.18	17.23

Figure 3: (Top) Samples with different number of text lines from the Synthetic Text Blocks test set. (Bottom) Word-level precision, recall, and character-level normalised edit-distance (ED) are reported (all in %).

4.1 Recognising Multiple Lines of Text

We apply our model to the task of spotting, *i.e.* joint localisation and recognition, of text in images containing multiple lines (or a block) of text. We proceed in two steps: first, using synthetically generated text-block images, we show superior generalisation ability of the model trained with inductive factorisation; second, we fine-tune this text-block spotting model trained on synthetic data, on real text data extracted from the ICDAR 2013 [22] benchmark, and outperform the state-of-the-art word-level text spotting method.
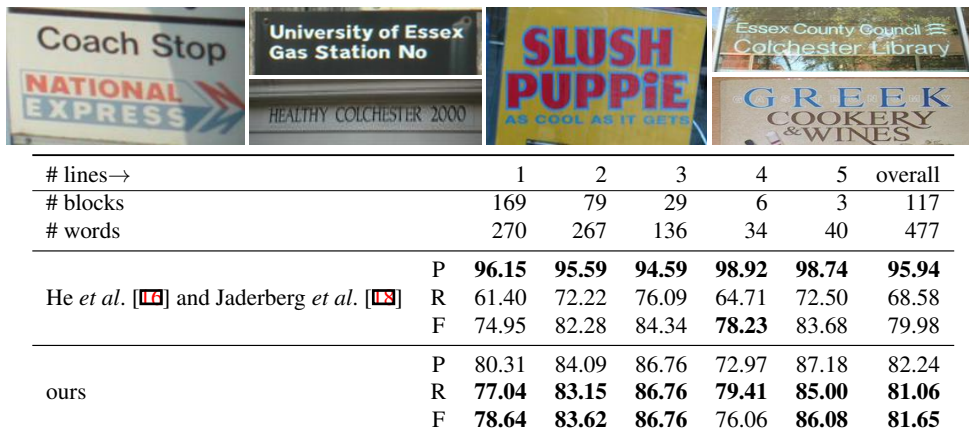
4.1.1 Model Details

Given an image **x** containing multiple lines of text (or a text-block; see examples in fig. 3), the corresponding sequence label **y** is a sequence of characters in lexicographic order (*i.e.* left-to-right, first-to-last line). We factorise the problem of spotting text in text-blocks at the level of lines, *i.e.* a "token" y_k (in section 3.2), corresponds to the kth line. Hence, y_k itself is a sequence of n characters $\{y_1^k, \ldots, y_n^k\}$ in the kth line. At each inductive step, one full-line is recognised.

Spatial Memory Representation. The spatial memory $\mathbf{m}_k$ at step k represents the location of first k lines which have been recognised so far, by setting the pixel-values inside the corresponding line-level bounding-boxes to 1 (the background is 0). The memory updates $\Delta\mathbf{m}_k$ correspond to the location of kth line (see fig. 2).

Image Encoder (Ψ). We employ the fully-convolutional DRN-C-26 Dilated Residual Network [45] as the image-encoder, which consists of six residual blocks; the encoder downsamples the image by a factor of 8, and has a stride of 32 (details in the reference, and supplementary material). Hence, an input image, concatenated with memory-map (**m**) of dimensions $H \times W \times 4$ is encoded as a feature-map of dimensions $\lceil \frac{H}{8} \rceil \times \lceil \frac{W}{8} \rceil \times 512$.

Recurrent Module (Φ). We use an LSTM-RNN with soft-attention over the convolutional features as the line-level character decoder. The state-size is set to 1024, while the attention-embedding dimension is set to 512. The attention weights (α in eq. (3)) corresponding to all the predicted characters in the current line are summed up; this produces an approximate localisation (S_α) of the line (second row in fig. 2). S_α is concatenated with the image-representation Ψ, and convolved with a stack of 2 convolutional+ReLU [29] layers (128 filters each), and a final 1×1 convolutional layer to produce the memory-update ($\Delta\mathbf{m}$).

# lines→		1	2	3	4	5	overall
# blocks		169	79	29	6	3	117
# words		270	267	136	34	40	477
	P	96.15	95.59	94.59	98.92	98.74	95.94
He *et al.* [16] and Jaderberg *et al.* [18]	R	61.40	72.22	76.09	64.71	72.50	68.58
	F	74.95	82.28	84.34	**78.23**	83.68	79.98
	P	80.31	84.09	86.76	72.97	87.18	82.24
ours	R	77.04	83.15	86.76	**79.41**	85.00	81.06
	F	78.64	83.62	86.76	76.06	86.08	81.65

Figure 4: (Image) Samples with different number of text lines from the ICDAR-2013 Text Blocks test set. (Table) Number of block-images with the given number of text lines (*# blocks*), the total number of words in these images (*# words*), and the word-level precision (*P*), recall (*R*), and F-score (*F*) (all in %) are reported.

4.1.2 Synthetic Text Blocks

Following the success of synthetic data in text-spotting [13, 18, 41], we test the generalisation to number of lines beyond those present in the training set, on synthetically generated text-block images (fig. 3). Using synthetic data enables this study, as it is difficult to collect real-world images of text with a large number of lines.

Dataset. The training set consists of text-block containing 3–5 lines, while the test contains 1–10 lines, with 500000 samples for each number of lines. To generate a synthetic text-block image for a given number of lines, the following procedure is followed: a random number of words (3–5 per line) are selected from a lexicon of approximately 90k words [18]. Then the text-lines are randomly aligned (left, centre, right), resized to potentially different heights (within the same block), separated by random amounts of line-spacing, transformed with a small perspective or affine transformation, and finally rendered against a randomly chosen background image, with a font chosen from over 1200 fonts. Figure 3 shows some samples generated through this procedure.

Evaluation Metrics. We report the word-level precision and recall, computed as the intersection of the predicted words and the ground-truth words, normalised by the total number of predicted or ground-truth words respectively, in addition to the normalised edit-distance.

Results. The results are summarised in fig. 3, and fig. 2 visualises the predicted updates to the mask on a test image. We note consistent levels of precision and recall, even when testing with twice the number of lines than in the training set. We also trained a RNN model (with identical Ψ, Φ) end-to-end without any inductive factorisation. It suffered from two difficulties: (1) block-level end-to-end training did not converge for more than three lines; and (2) the model did not generalise to more than three lines: note, the steep fall in the recall rates, and increase in the edit-distance for four lines.

4.1.3   Real Text Blocks – ICDAR-2013

We fine-tune the inductive block-parser trained on synthetic data on text-blocks extracted from the ICDAR 2013 Focussed Scene Text [22] dataset, and compare with state-of-the-art word-level methods

Dataset.   ICDAR 2013 is a dataset of 229 training, and 233 test scene images containing text, with word-level bounding boxes and text-string annotations. As our model is tailored for recognising text in blocks, we extract images of text-blocks, *i.e.* images containing multiple lines of text from the dataset: first, text lines are formed by linking together pairs of words with relative distance at most half, and three times the minimum of their heights, along the vertical and horizontal directions respectively; then, the bounding boxes of these lines are doubled along the height, and those with area of intersection at least a third of the maximum of their areas are merged into text-blocks. Some samples are visualised, and the number of blocks obtained and other statistics are given in fig. 4.

Baseline word level model.   We combine the word localisation model of He *et al.* [16] (88% F-score on ICDAR13 Focussed Scene localisation), with the strong lexicon based word-level recognition network of [18] (90.8% on ICDAR13 cropped word recognition).[2] This combination of state-of-the-art models for localisation and recognition, provides a strong baseline to compare our joint model against. To minimise the discrepancy between the test and training setting we run the detector on full scene images and then using the detected word locations, associate them with the text-blocks used in this experiment. Note, we also ran the baseline detector on block images but this lead to worse results: word-spotting F-score = 78.83% (block-images) vs. 79.98% (full-scene images).

Evaluation.   As is standard in the benchmark, we report the word-level recall, precision and the F-score. For fair comparison with the lexicon-based word-recognition model [18], we use the same lexicon of 90k words to constrain the predictions of our model.

Results.   Figure 4 summarises the results. We note that our inductive block parser, consistently achieves a higher recall and F-score (except for 4-lines) across all lines. Our method combines the stages of both localisation and recognition, and hence avoids downstream error propagation, achieving greater recall. The higher precision of the baseline is due to the detector only producing high-confidence detections.

4.2   Counting by Enumeration

We further test improvement in generalisation ability induced by inductive training in a different setting: counting objects in images through enumeration, where a recurrent module acts as an enumerator, counting one object in each step, and terminating when done. We test on two datasets: first, images containing randomly coloured shapes, and second: aerial images of airplanes extracted from the recently introduced DOTA dataset [42].

4.2.1   Model Details

In each inductive step one object is counted, and a corresponding label of '0' is produced; the enumeration terminates when all the objects have been accounted for, producing '1' as the output, as in [33]. Hence, for a given image **x** containing multiple ($= N \geq 0$) objects of interest, the corresponding sequence label is $(0, \ldots, 0, 1)$, *i.e.* a sequence containing N zeros, and one 1.
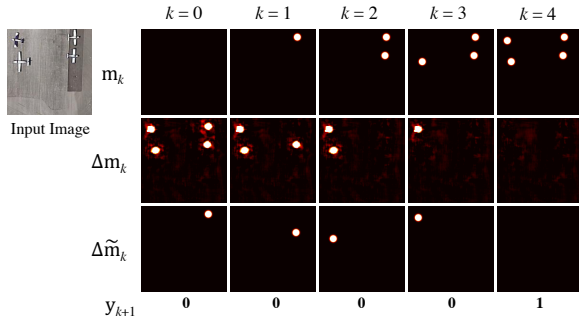

Figure 5: **Enumerative counting visualisation.** Decoding procedure for an example image containing four airplanes. Inference runs for five steps, indicating the end-of-sequence ($y_k = 1$) in the fifth step. At each step, the memory update (Δm_k) regresses peaks at the location of *all* the remaining objects, out of which one is randomly picked ($\Delta \tilde{m}_k$) to update the memory-map for the next step.

Spatial Memory Representation. The spatial memory $\mathbf{m}_k$ at step k represents the location of first k objects which have been enumerated, by placing a small gaussian peak at their centre. Due to lack of any natural order for counting, it is difficult to assign a particular object to a specific enumeration step. Hence, the memory update $\Delta \mathbf{m}_k$ at the kth step regresses gaussian peaks at the locations of *all* the remaining objects. The memory is updated with the location of one of the remaining objects (selected randomly); fig. 5 visualises the inference steps for a sample image.

Image Encoder (Ψ). A fully-convolutional Dilated Residual Network [45], consisting of three residual-blocks each with two pre-activation residual units [14], is used to encode the images (detailed architecture in supplementary material). The images are not downsampled, hence, the feature-maps retain the original dimensions of the input.

Recurrent Module (Φ). The memory updates $\Delta \mathbf{m}_k$ are regressed from the image-features, using a 1×1 convolutional layer. The binary valued token at each step y_k is predicted as: $p(y_k = 1) = \text{Sigmoid}\left(w \cdot \text{MaxPool}(\Delta \mathbf{m}_k) + b\right)$.

4.2.2 Datasets

We evaluate on two datasets described below. Images of size 128×128 were used for both the datasets. The training sets consisted of images containing $\{3, 4, 5\}$ objects; to test for generalisation beyond training sequence lengths, the test set included 3–10 objects. Figure 6 visualises some samples from the datasets.

Coloured Shapes. We start with a procedurally generated synthetic dataset, which consists of shapes with random colour, position and type (circle, triangle, or square) placed on the canvas. This dataset provides a simplified setting for analysis without confounding complexity. The training set consists of 10k images, while the test set consists of 200 images for each count.

DOTA Airplanes. DOTA [42] is a recently introduced image dataset of high-resolution aerial images, where 15 object categories have been labelled with oriented bounding-boxes. We extract image crops from the DOTA dataset which consist of airplanes. The training set consists of 20k crops extracted from 100 images from the dataset's training set, while the test set consists of 200 image crops for each object count (ranging from 3 to 10) extracted from 70 images from the dataset's validation set.

Dataset	Model	Number of Objects							
		3	4	5	6	7	8	9	10
Coloured Shapes	end-to-end	99.53	99.53	98.93	0	0	0	0	0
	inductive	100	99.89	99.52	98.93	97.18	98.47	95.48	95.45
DOTA	end-to-end	82.00	70.50	74.80	0	0	0	0	0
	inductive	82.50	79.00	75.50	72.50	69.00	43.81	32.21	29.20

Figure 6: (Top) Samples from the Coloured Shapes and DOTA Airplane datasets containing different number of objects. (Bottom) Accuracy of enumerative counting with and without inductive training (all in %).

4.2.3   Results

We compare our inductive model against a soft-attention LSTM-RNN trained end-to-end with the same image encoder Ψ. We report the mean accuracy of prediction, where a test image is evaluated as correct, if the predicted count matches the ground-truth count exactly. Figure 6 summarises the results on both the datasets for the two models. On both the datasets, end-to-end trained RNN fails to generalise to object counts beyond those in the training set (> 5), while the one with inductive training does not fail catastrophically at higher counts. Lower accuracy of the inductive model at higher counts can be attributed to crowding of objects, which is not seen in the training set (*e.g.* fig. 6 rightmost image).

5   Conclusions

While RNNs may seem a perfect match for problems with an inductive structure, these networks fail to learn appropriate invariants to allow recursion to extend beyond what is encountered in the training data. We have shown how to repurpose standard RNN models to restrict the recurrent state to a suitable state-representation – which in our application are the image locations corresponding to the predictions at each step – where the correct invariant can be enforced. The result is an iterative visual parsing architecture which generalises well-beyond the training sequence lengths. This idea can be extended to visual problems with a tail-recursive structure, from object tracking to boundary and line tracing.

References

[1] O. Alsharif and J. Pineau. End-to-end text recognition with hybrid HMM maxout models. In *Proc. ICLR*, 2014.

[2] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Learning to detect cells using non-overlapping extremal regions. In N. Ayache, editor, *International Conference on Medical Image Computing and Computer Assisted Intervention*, Lecture Notes in Computer Science, pages 348–356. MICCAI, Springer, 2012.

[3] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Interactive object counting. In *Proc. ECCV*, 2014.

[4] C. Arteta, V. Lempitsky, and A. Zisserman. Counting in the wild. In *Proc. ECCV*, 2016.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015.

[6] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *NIPS*, 2016.

[7] J. Cai, R. Shin, and D. Song. Making neural programming architectures generalize via recursion. In *Proc. ICLR*, 2017.

[8] de T. Campos, B. R. Babu, and M. Varma. Character recognition in natural images. *VISAPP*, 2009.

[9] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[10] Stanislas Dehaene and Laurent Cohen. Dissociable mechanisms of subitizing and counting: Neuropsychological evidence from simultanagnosic patients. *Journal of Experimental Psychology: Human Perception and Performance*, 20(5):958, 1994.

[11] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Proc. ICLR*, 2014.

[12] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[13] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proc. CVPR*, 2016.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, 2016.

[15] P He, W Huang, Y Qiao, CC Loy, and X Tang. Reading scene text in deep convolutional sequences, 2016. In *The 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, volume 1, 2016.

[16] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li. Single shot text detector with regional attention. In *Proc. ICCV*, 2017.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Workshop on Deep Learning, NIPS*, 2014.

[19] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Proc. ECCV*, 2014.

[20] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. In *International Conference on Learning Representations*, 2015.

[21] A. Joulin and T. Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *NIPS*, 2015.

[22] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, L. P. de las Heras, et al. ICDAR 2013 robust reading competition. In *Proc. ICDAR*, pages 1484–1493, 2013.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] C.Y. Lee and S. Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proc. CVPR*, 2016.

[26] C.Y. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu. Region-based discriminative feature pooling for scene text recognition. In *Proc. CVPR*, 2014.

[27] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NIPS*, 2010.

[28] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. *Proc. BMVC.*, 2012.

[29] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010.

[30] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *Proc. ECCV*, pages 752–765. Springer, 2012.

[31] A. Poznanski and L. Wolf. Cnn-n-gram for handwriting word recognition. In *Proc. CVPR*, 2016.

[32] S. Reed and N. De Freitas. Neural programmer-interpreters. In *Proc. ICLR*, 2015.

[33] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *Proc. ECCV*, 2016.

[34] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *ArXiv e-prints*, 2015.

[35] Baoguang Shi, Xinggang Wang, Pengyuan Lv, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proc. CVPR*, 2016.

[36] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *Proc. CVPR*, 2013.

[37] Bolan Su and Shijian Lu. Accurate scene text recognition based on recurrent neural network. In *Proc. ACCV*, 2014.

[38] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[39] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[40] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Proc. ICCV*, pages 1457–1464. IEEE, 2011.

[41] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proc. ICPR*, pages 3304–3308. IEEE, 2012.

[42] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proc. CVPR*, 2018.

[43] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. ICML*, 2015.

[44] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *Proc. CVPR*, 2014.

[45] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Proc. CVPR*, 2017.

[46] W. Zaremba, T. Mikolov, A. Joulin, and R. Fergus. Learning simple algorithms from examples. In *Proc. ICML*, 2016.

[47] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[48] S. Zhang, G. Wu, J. P. Costeira, and J. MF Moura. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *Proc. ICCV*, 2017.