Learning to Read by Spelling

Towards Unsupervised Text Recognition

Ankush Gupta
Visual Geometry Group
University of Oxford
ankush@robots.ox.ac.uk

Andrea Vedaldi
Visual Geometry Group
University of Oxford
vedaldi@robots.ox.ac.uk

Andrew Zisserman
Visual Geometry Group
University of Oxford
az@robots.ox.ac.uk

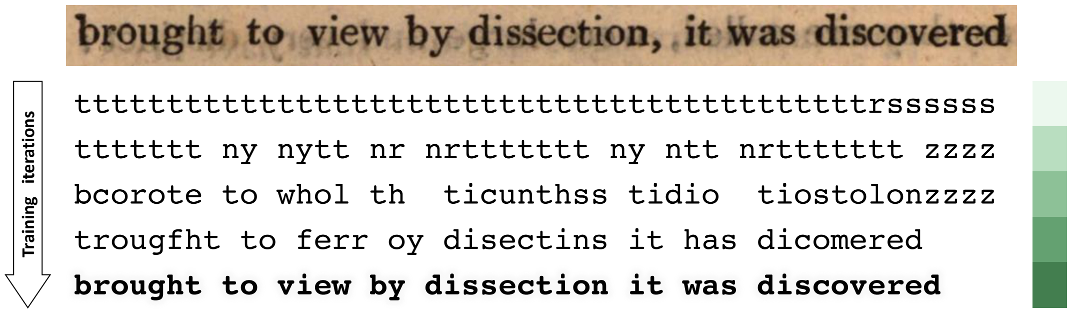

brought to view by dissection, it was discovered
Training iterations
tttttttttttttttttttttttttttttttttttttttttrssssss
ttttttt ny nytt nr nrttttttt ny ntt nrttttttt zzzz
bcorote to whol th ticunthss tidio tiostolonzzzz
trougfht to ferr oy disectins it has dicomered
brought to view by dissection it was discovered

Figure 1: Text recognition from unaligned data. We present a method for recognising text in images without using any labelled data. This is achieved by learning to align the statistics of the predicted text strings, against the statistics of valid text strings sampled from a corpus. The figure above visualises the transcriptions as various characters are learnt through the training iterations. The model first learns the concept of `{space}`, and hence, learns to segment the string into words; followed by common words like `{to, it}`, and only later learns to correctly map the less frequent characters like `{v, w}`. The last transcription also corresponds to the ground-truth (punctuations are not modelled). The colour bar on the right indicates the accuracy (darker means higher accuracy).

ABSTRACT

This work presents a method for visual text recognition without using any paired supervisory data. We formulate the text recognition task as one of aligning the conditional distribution of strings predicted from given text images, with lexically valid strings sampled from target corpora. This enables fully automated, and unsupervised learning from just line-level text-images, and unpaired text-string samples, obviating the need for large aligned datasets. We present detailed analysis for various aspects of the proposed method, namely — (1) impact of the length of training sequences on convergence, (2) relation between character frequencies and the order in which they are learnt, (3) generalisation ability of our recognition network to inputs of arbitrary lengths, and (4) impact of varying the text corpus on recognition accuracy. Finally, we demonstrate excellent text recognition accuracy on both synthetically generated text images, and scanned images of real printed books, using no labelled training examples.

CCS CONCEPTS

• **Computing methodologies → Unsupervised learning**; *Image representations*; *Object recognition*; • **Applied computing → Optical character recognition**;

KEYWORDS

unsupervised learning, text recognition, adversarial training

ACM Reference Format:
Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. 2018. Learning to Read by Spelling: Towards Unsupervised Text Recognition. In *11th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP 2018), December 18–22, 2018, Hyderabad, India,* Anoop M. Namboodiri, Vineeth Balasubramanian, Amit Roy-Chowdhury, and Guido Gerig (Eds.). ACM, New York, NY, USA, Article 33, 10 pages. https://doi.org/10.1145/3293353.3293386

1 INTRODUCTION

read (riːd) *verb* • Look at and comprehend the meaning of (written or printed matter) by interpreting the characters or symbols of which it is composed.
spell (spɛl) *verb* • Write or name the letters that form (a word) in correct sequence.
— *Oxford Dictionary of English*

Text recognition, namely the problem of reading text in images, is a classic problem in pattern recognition and computer vision that has enjoyed continued interest over the years, owing to its many practical applications, such as recognising printed [69, 75] or handwritten [12, 43] documents, or more recently, text in natural

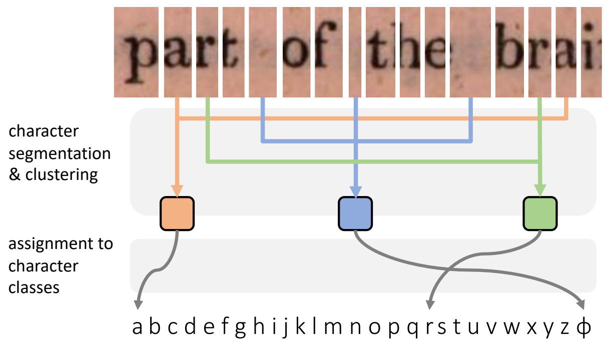

part of the brain
character segmentation & clustering
assignment to character classes
a b c d e f g h i j k l m n o p q r s t u v w x y z ϕ

Figure 2: Unsupervised text recognition can be factored into two sub-problems: (1) *visual*: segmentation at the character-level, followed by clustering (or recognition) into a known number of classes, and (2) *linguistic*: determining the character identity of these clusters based on language constraints. Three character classes corresponding to {a, ϕ, r} are visualised above (ϕ stands for {space}). A given text image is mapped to a sequence of characters using a fully-convolutional network; the predicted sequences are compared against linguistically valid text-strings using an *adversarial discriminator*, which guides the mapping of the characters to the correct identity. The two networks trained end-to-end jointly, enable text recognition without any labelled training data.

images [35, 54, 58]. Consequently, many different and increasingly accurate methods have been developed. Yet, all such methods adopt the same *supervised learning* approach that requires example images of text annotated with the corresponding strings.

Annotations are expensive because they must be *aligned* to individual training images. For example, for a text-image of *cats*, the corresponding annotation is the string {c, a, t, s}. A straightforward but tedious approach is to collect such annotations manually [37, 57, 77]; however, since datasets often comprise several million examples [32, 41], this scales poorly. Another, perhaps more pragmatic, approach is to engineer highly-sophisticated synthetic data generators to mimic real images [24, 32, 79]. However, this requires developing new generators for each new textual domain, and could be problematic for special cases such as text in ancient manuscripts.

We propose instead to develop learning algorithms that can work with *unaligned* annotations. In this paradigm, images containing text can be extracted *e.g.* from scanned documents or by mining online image collections [33]. Independently, strings containing the same *type* of text (but not exactly the same text) can be readily harvested from machine readable text corpora (*e.g.* WMT datasets [1]). Both steps can be implemented economically in a fully-automated manner, making such an approach highly desirable.

More specifically, we demonstrate visual text recognition by only providing examples of valid textual strings, but without requiring them to be aligned to the example images. In this manner, the method is almost unsupervised, as by only knowing how to **spell** correctly, it learns to **read**. The method works by learning a predictor that converts images into strings that *statistically* match the target corpora, implicitly reproducing quantities such as letter and word frequencies, and n-grams. We show empirically that this seemingly weak principle is in fact sufficient to drive learning successfully (section 5).

Text recognition can be factored into two sub-problems (see fig. 2): (1) *visual*: segmenting the text-image into characters and clustering the different characters into a known number of distinct classes, and (2) *linguistic*: assigning these clusters to the correct character identity. Indeed, earlier attempts at unsupervised text recognition proposed two-stage solutions corresponding to the two sub-problems [3, 28, 36, 39]. We address the first problem by exploiting the properties of standard fully-convolutional networks [51] — namely locality and translation invariance of the network's filters. The second problem is equivalent to solving for the correct permutation, or breaking a 1:1–substitution cipher [62]. The latter problem is NP-hard under a bi-gram language model [60]. While several solutions like aligning uni-gram (*i.e.* frequency matching) or n-gram statistics [50, 74] have been proposed traditionally for breaking ciphers [16], we instead adopt an adversarial approach [20]. The result is a compact fully-convolutional sequence (*i.e.* multiple words/text-string) recognition network which is trained against a discriminator in an end-to-end fashion. The discriminator uses as input only unaligned examples of valid text strings.

We study various factors which affect training convergence, and use synthetically-generated data for these controlled experiments. We also show excellent recognition performance on real text images from the Google1000 dataset [21], given *no aligned labelled data*.

The rest of the paper is structured as follows. Section 2 reviews related work, section 3 describes our technical approach, section 4 gives the implementation details, section 5 evaluates the method on the aforementioned data, and section 6 summarises our findings.

2 RELATED WORK

Supervised Text Recognition. Distinct paradigms have emerged and evolved in text recognition. Traditional character-level methods adopt either sliding-window classifiers [33, 54, 78–80], or over / under segment into parts [5, 11, 58], followed by grouping through classification. Words or sentences are then inferred using language models [5, 33, 45, 54, 55, 59, 68, 77–79]. Another set of methods process a whole word image, modelling it either as retrieval in a collection of word images from a fixed lexicon [4, 18, 22, 65] or as learning multiple position dependent classifiers [32, 34, 63]. Our recognition model is similar to these character-sequence classifiers in that we train with a fixed number of output characters; but there is an important difference: we discard their fully-connected (hence, position sensitive) classifier layers and replace them with fully-convolutional layers. This drastically reduces the number of model parameters, and lends generalisation ability to inputs of arbitrary length during inference. More recent methods treat the text-recognition problem as one of sequence prediction in an encoder-decoder framework [14, 73]. [72] adopted this framework first, using HOG features with Connectionist Temporal Classification (CTC) [23] to align the predicted characters with the image features. [26, 66] replaced HOG features with stronger CNN features, while [44, 67] have adopted the soft-attention [9] based recurrent decoders. Note, all these methods learn from labelled training examples.

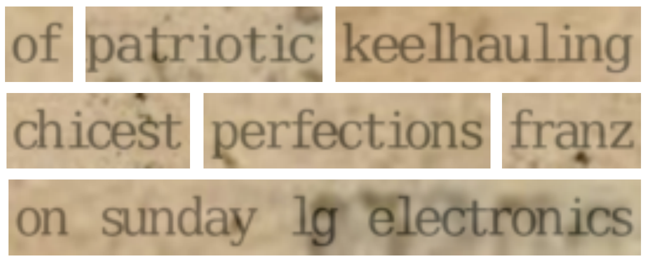
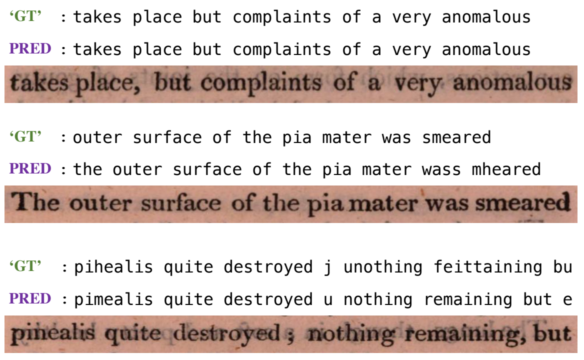


Figure 3: Synthetic text-image samples. A few synthetically generated samples of different lengths used in the controlled experiments (see section 5). Our model attains ≈99% *character* and ≈95% *word* accuracy respectively (section 5.4), after training on only *unaligned* image and text examples.

Unsupervised Text Recognition. Unsupervised methods for text recognition can be classified into two categories. First, category includes generative models for document images. A prime example is the *Ocular* system [10], which jointly models the text content, as well as the noisy rendering process for historical documents, and infers the parameters through the EM-algorithm [15], aided by an n-gram language model. The second category includes methods for automatic decipherment. Decipherment, is the process of mapping unintelligible symbols (ciphertext) to known alphabet/language (plaintext). When the input is visual symbols, it becomes equivalent to text recognition. Some early works [13, 56] for optical character recognition (OCR), indeed model it as such. [27] cluster connected components in binarised document images and assign them to characters by maximising overlap with a fixed lexicon of words based on character frequencies and co-occurrence; [28, 36] also follow the same general approach. [3] break the Borg cipher, a 17th century 408-pages manuscript, by also first clustering symbols but decipher using the noisy-channel framework of [38] through finite-state-machines. [46] learn mappings from hidden-states of an HMM with their transition probabilities initialised with conditional bi-gram distributions. [40] propose an iterative scheme for bootstrapping predictions for learning HMMs models, and recognise handwritten text. However, their approach is limited to (1) word images, (2) fixed lexicon (≈44K words) to facilitate exhaustive tree search, whereas, our method is applicable to full *text strings*, does not require a pre-defined lexicon of words.

Unsupervised Learning by Matching Distributions. Output Distribution Matching (ODM) which aligns the *distributions* of predictions with *distributions* of labels was proposed in [74] for "principled" unsupervised learning; although similar ideas for learning by matching statistics have been explored earlier, *e.g.* for decipherment (see above), and also for machine translation [64, 70]. [50] extend ODM to sequences, and apply it to OCR with known character segmentations and pre-trained image features. In essence, ODM [74], or Empirical-ODM [50] minimises the KL-divergence cost between the empirical predicted and ground-truth n-gram distributions. Our learning principle is the same, however, we do not explicitly formulate the matching cost, instead learn it online using an adversary [20]. Recent works [7, 42] have demonstrated unsupervised machine translation using such adversarial losses, however they closely follow the *CycleGAN* framework [81] which



'GT' : takes place but complaints of a very anomalous
PRED : takes place but complaints of a very anomalous

takes place, but complaints of a very anomalous

'GT' : outer surface of the pia mater was smeared
PRED : the outer surface of the pia mater wass mheared

The outer surface of the pia mater was smeared

'GT' : pihealis quite destroyed j unothing feittaining bu
PRED : pimealis quite destroyed u nothing remaining but e

pinealis quite destroyed ; nothing remaining, but

Figure 4: Real text-image samples. Randomly selected samples from a *real* scanned book's test set along with the "ground-truth" ('GT') and the predicted strings (PRED); punctuations are not modelled. Our model achieves excellent recognition performance — 96.2% *character*, and 84.8% *word* accuracy (see section 5.6) without using any aligned/labelled training examples. Note, the "ground-truth" ('GT') comes from Google's OCR engine output, hence is not perfect (*e.g.* second and third image above).

learns a bidirectional mapping between the input and target domains to enforce bijection. This framework has also been applied recently in *CipherGAN* to break ciphers [19]. The *CycleGAN* framework learns a bi-directional mapping to enforce strong correlation between the input and the generated output to avoid collapsing to the same output instance regardless of the input. We, however, dispense with back-translation/reconstruction, and instead enforce correlation directly in the structure of the recogniser by limiting the receptive-field of convolutional layers. Hence, our method is an instantiation of the original (single) generator–discriminator framework of *GANs* [20]. However, our method is perhaps the first to decode sequences of discrete symbols from images using an adversarial framework; these two domains have only been explored independently in *CycleGAN* and *CipherGAN* respectively.

3 METHOD

The aim of text recognition is to predict a sequence of characters given an image of text. Let the image be a tensor $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{H \times W \times C}$, where H, W, C are its height, width, and number of colour channel(s) respectively. Furthermore, let $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathcal{Y}$ denote the corresponding character string where each y_i is a character from an alphabet $\mathcal{A}$ containing K symbols, *i.e.* $|\mathcal{A}| = K$. For later convenience, a character y_i is represented as a K-dimensional one-hot vector. Since such vectors are elements of the K-dimensional simplex Δ^K, we set $\mathcal{Y} = (\Delta^K)^n \subset \mathbb{R}^{K \times n}$. Without loss of generality, we consider strings of a fixed length $n \in \mathbb{N}$. The objective of *unsupervised* text recognition, then, is to learn the mapping $\Phi(\mathbf{x}) = \mathbf{y}$, given only *unpaired* examples from the two domains $\{\mathbf{x}_i\}_{i=1}^N$ where $\mathbf{x}_i \in \mathcal{X}$ and $\{\mathbf{y}_j\}_{j=1}^M$ where $\mathbf{y}_j \in \mathcal{Y}$.

We cast this in an adversarial learning framework based on Goodfellow *et al.* [20]. We view the *text recogniser* $\Phi : \mathcal{X} \to \mathcal{Y}$ as a *conditional generator* of strings $\mathbf{y}$. The recogniser competes against an adversarial discriminator $D_\mathcal{Y}$, which aims to distinguish between *real* strings $\{\mathbf{y}\}$ and *generated* strings $\{\Phi(\mathbf{x})\}$. In other words, Φ and $D_\mathcal{Y}$ are optimised simultaneously to play the following two-player minimax game [20] $\min_\Phi \max_{D_\mathcal{Y}} \mathcal{L}(\Phi, D_\mathcal{Y})$ where the value function is given by:

$$\mathcal{L}(\Phi, D_{\mathcal{Y}}) = \mathop{\mathbb{E}}_{\mathbf{y}\sim\mathcal{Y}}[\log D_{\mathcal{Y}}(\mathbf{y})] + \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{X}}[\log(1 - D_{\mathcal{Y}}(\Phi(\mathbf{x}))].$$

The recogniser learns the visual problem of segmenting characters in images, and organising them into distinct categories; while the discriminator, by checking the predicted sequence of characters against linguistically valid strings, guides the assignment of these categories into the respective correct character classes (see fig. 2).

Grounding. A potential pitfall is that the string generator network (or recogniser Φ) may learn to use the input image as a mere source of noise, using it to generate the correct distribution of strings, without learning to recognise the string represented in the image. A useful mapping, instead, must be *grounded*, *i.e.* the generated string $\mathbf{y} = \Phi(\mathbf{x})$ should correspond to the text represented in the input image $\mathbf{x}$.

A possible way to encourage grounding is to ensure that the image $\mathbf{x}$ can be recovered back from the string $\mathbf{y}$. Both *CycleGAN* [81] and *CipherGAN* [19] achieve this by learning a second inverse mapping $\Psi : \mathcal{Y} \to \mathcal{X}$ from the target domain back to the input and complete the cycle $\Psi(\Phi(\mathbf{x})) \triangleq \mathbf{x}$. However, learning a mapping from character strings to images is highly ambiguous: rendering a given string requires sampling the background image, font style, font colour, geometry of the glyphs, shadows, noise etc. This ambiguity arises because text recognition requires translating between two very different *modalities*, *viz.* text and images, which is much harder than translating within the same modality, *e.g.* between images in *CycleGAN* [81] where only local texture is modified, or between character strings in *CipherGAN* [19], where the characters are permuted.

Instead of enforcing cycle-consistency, we encourage grounding via the following two key architectural modifications in the recogniser Φ (architectural details are given in section 4):

(1) **Prediction Locality.** The character predictor is local, with a receptive field large enough to contain at most two or three characters in the image. While this may sound simple, it embodies a powerful constraint. Namely, such local predictors can generate a string which is globally consistent only if they correctly transduce the structure of the underlying image. Otherwise, local predictors may be able to match local text statistics such as n-grams, but would not be able to match global text statistics, such as forming proper words and sentences (see also section 6.1 of [74] for similar ideas). Global consistency is enforced by the adversarial discrminiator which has a large receptive field over the predicted characters (see section 4).
(2) **Reduced Stochasticity.** We also make the generated strings a deterministic function of the input. We achieve this by removing the noise input from Φ which is normally used in generator networks. Furthermore, we do not use dropout regularization [71].

Training Objective. The discriminator $D_{\mathcal{Y}}$ operates in the domain of *discrete* symbols. While the real symbols are represented as one-hot vectors or *vertices* $\mathrm{Vert}(\Delta^K)$ of the standard simplex, the generated symbols are output of a SoftMax operator over predicted logits, and hence typically belong to the *interior* of the simplex Δ^K. This was identified, as the cause for *uninformative discrimination* in *CipherGAN* [19], where the discriminator distinguishes using this unimportant difference, rather than soundness of the generated strings. To mitigate this, we adopt their proposed solution and learn a d-dimensional embedding for each of the K symbols in the alphabet, collectively represented by a matrix $W \in \mathbb{R}^{K\times d}$. Furthermore, we replace the log-likelihood loss with a squared difference loss, as proposed by [53]. Hence, we optimise the following revised training objective:

$$\mathcal{L}(\Phi, D_{\mathcal{Y}}, W) = \mathop{\mathbb{E}}_{\mathbf{y}\sim\mathcal{Y}}[D_{\mathcal{Y}}(W^T\mathbf{y})^2] + \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{X}}[(1 - D_{\mathcal{Y}}(W^T\Phi(\mathbf{x})))^2].$$

The embeddings W are trained to aid discrimination among symbols by solving $\min_\Phi \max_{D_{\mathcal{Y}},W} \mathcal{L}(\Phi, D_{\mathcal{Y}}, W)$. Learning such embeddings improved the speed of convergence and final accuracy, as also noted in [19], while using square differences improved numerical stability.

Discussion: Why is this a feasible learning problem? While learning to recognise visual symbols without any paired data seems unattainable, the tight structure of natural language provides sufficient constraints to enable learning. First, lexically valid text strings form a tiny sub-space of all possible permutations of symbols, *e.g.* there are only $\approx 13k$ valid English words of length 7, as opposed to almost 8 billion permutations of the 26 English letters. Second, the relative frequencies of the characters and their co-occurrence patterns impose further constraints (see section 5.3 for correlation between character-frequency and learning). These constraints combined with strong correlation between the input image and the predicted characters are sufficient to drive learning successfully.

4 IMPLEMENTATION

Both, the recogniser (Φ) and the discriminator ($D_{\mathcal{Y}}$) are implemented as fully-convolutional networks [51]. The recogniser ingests an image of text and produces a sequence of character logits. The discriminator operates instead on character strings represented as sequence of character vectors, and produces a scalar discrimination score as output. The discriminator acts as a *spell-checker*, pointing out the errors in the generated strings. We describe their architecture and optimisation details below.

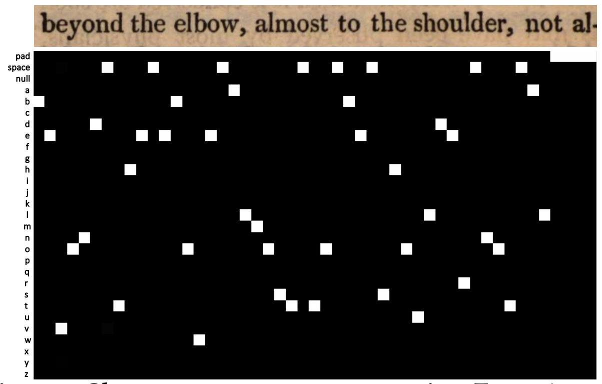


Figure 5: Character sequence representation. Text strings are represented as sequences of n one-hot (for *real* strings) or SoftMax normalised logits (for *predictions*) over $|\mathcal{A}| = K$ character classes. A sample image and the model's prediction are visualised above (one-hot *real* strings look similar); here $K = 29$ and $n = 50$.

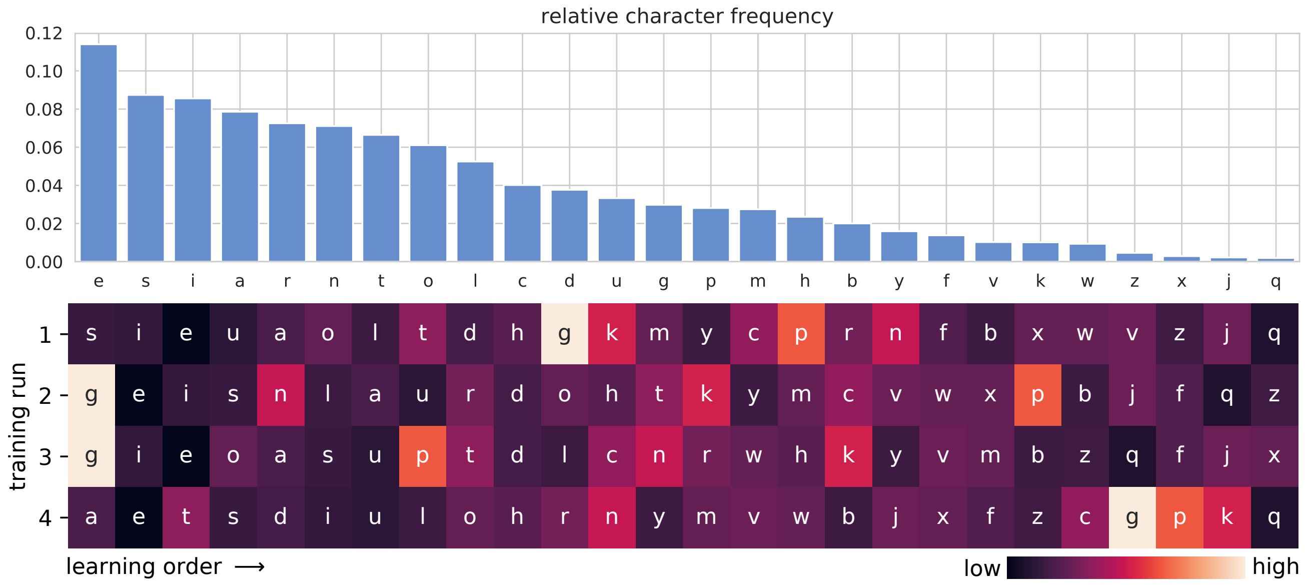


Figure 6: Learning order for different characters. The order in which the various characters are learnt is strongly correlated (Spearman's rank correlation coefficient $\rho = 0.80$, p-value $< 1e{-}5$) to their frequency in the English language **[top]**. Ranking for the learning order is based on the training iteration number at which the model achieves 50% accuracy for a given character. **[bottom]** Rankings from four different training runs are presented to show the variance — bright colours signify high variance in rank across runs, while dark colours correspond to low variance. The character {g} is a curious exception to the trend, as it is sometimes learnt first (runs 2, 3); see section 5.3 for the reason and further discussion.

Recogniser Φ. We train our models for strings of a maximum fixed number of characters = n. To this end, the input image dimensions are held fixed at $32 \times (n \cdot 2^4)$ pixels (= height×width). Hence, an image of size $H \times W$ is scaled to $H' \times W' = 32 \times \min(\lceil W \cdot \frac{32}{H} \rceil, n \cdot 2^4)$; if the $W' < n \cdot 2^4$, it is padded on the right with the mean channel intensity. The recogniser employs four blocks, each consisting of two convolution layers, followed by a 2×2 max-pooling layer. Each convolutional layer comprises of 32 filters of 3×3 dimensions, and is followed by batch-normalisation [30] and leaky-ReLU activation (slope= 0.2) [52]. Since max-pooling in each block downsamples the input by a factor of two, final output dimensions are $2 \times n \times D$ (where, $D = 32$ is the number of features). The height is collapsed using average-pooling, and each of the n D-dimensional feature vectors are mapped to $|\mathcal{A}| = K$ dimensional logits through linear projection, yielding a $K \times n$ dimensional tensor. Note the receptive field of the final (prediction) layer is small to encourage *locality*; specifically, it is 76 pixels wide which corresponds to ≈2.5 characters in the image. Although we train our recogniser on fixed-length strings, yet it generalises to different other lengths due to its fully-convolutional architecture (see section 5.4).

Discriminator (or spell-checker) $D_{\mathcal{Y}}$. The input $\mathbf{y}$ to the discriminator are $K \times n$ dimensional tensors of predicted and real strings containing n characters, represented as logits and one-hot vectors, respectively (see fig. 5). The predicted logits are first normalised through SoftMax to a valid probability distribution over the K characters for each of the n positions. Next, embeddings $\mathbf{y}_e \in \mathbb{R}^{d \times n}$ for both, the real and predicted strings are obtained: $\mathbf{y}_e = W^T \mathbf{y}$, where $W \in \mathbb{R}^{K \times d}$ are the character embeddings ($d = 256$). We adopt the fully-convolutional *PatchGAN* discriminator architecture [31, 48, 49], where patches correspond to substrings here. The embedded input $\mathbf{y}_e$ is fed to a stack of five 1D-convolutional layers, each with 512 filters of size 5. This amounts to a final receptive field of 21 characters which helps to enforce long-range structure. Each layer is followed by layer-normalisation [8] and leaky-ReLU (slope = 0.2); zero padding is used to maintain the size. The resulting $d \times n$ dimensional output is linearly projected to $1 \times n$, and average-pooled to obtain the final scalar score $D_{\mathcal{Y}}(\mathbf{y})$.

Optimization details. Recogniser, discriminator and character embeddings are trained jointly end-to-end. The parameters are initialised with Xavier initialization [17]. We use the RMSProp optimizer [76] with a constant learning rate of 0.001. The two-part discriminator loss objective is multiplied with $\frac{1}{2}$ as in [31]. The models are implemented in TensorFlow [2].

5 EXPERIMENTS

Our experiments have two primary goals. First, is an extensive analysis of various factors which affect the training: we study — (1) the impact of the length of training sequences on convergence (section 5.2); (2) the order in which various characters are learnt and its correlation with their frequencies (section 5.3), (3) generalisation ability of the fully-convolutional recogniser to different sequence lengths (section 5.4), and (4) impact of varying the text corpus on recognition accuracy (section 5.5). For these experiments we use synthetically generated text data as it provides fine control over various nuisance factors. The second objective is to show applicability of the proposed method to *real* document images

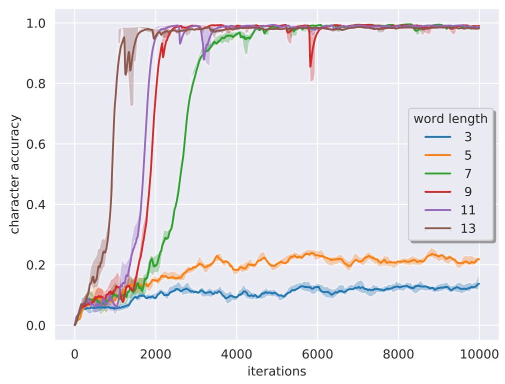


Figure 7: Effect of text length on convergence. Training with longer words leads to faster convergence: the order of convergence {13,11,9,7} mirrors the word lengths (see section 5.2). No convergence is seen for models trained on shorter words of length 3 and 5. For each word-length, the run with largest area-under-curve (AUC) from eight trials is plotted.

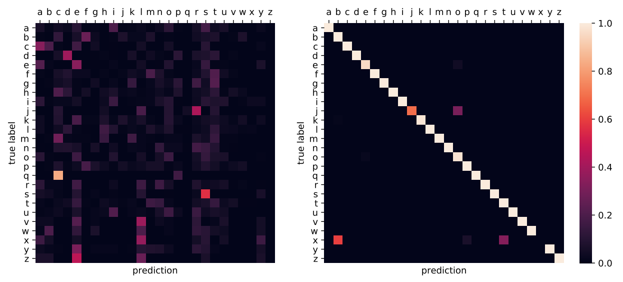


Figure 8: Confusion matrices for models trained on words of length 5 & 7. The model trained on length-5 words does not converge to a high accuracy, while the one trained on length-7 words does (see fig. 7 and section 5.2). Further, the accuracy for a character depends on its frequency: the length-5 model **[left]** confuses most characters, yet it is quite accurate for the common character {s}; while, the length-7 model **[right]** recognises most characters with high accuracy, yet it confuses the two least common characters {j,x} (see section 5.3).

(section 5.6). We first describe the datasets used in our experiments in section 5.1, and then present the results.

5.1 Datasets

Synthetic data. We generate synthetic text data to simulate old printed documents. Synthetic data aids the controlled ablation studies, as it provides tight control over the various factors, *e.g.* text content, font style and glyph geometry, background, colours, and other noise parameters. We sample the text content from two different sources depending on the experimental setting — (1) *words*: individual English words are sourced from a lexicon of 90K words used in the Hunspell spell-checker [29], and (2) *lines*: these are full valid English language text strings extracted from the 2011 news-crawl corpus provided by WMT [1]. Note, these text sources are used for rendering images, as well as for providing examples of valid strings to the discriminator. To limit the variance in position of characters, we use the VerilySerifMono fixed-width font. The background image data is sampled from the margins of historical books [6] to simulate various noise effects. The font colour is sampled from a k-means colour model learnt from the same dataset. The character set consists of the 26 English letters, one space character, and one additional null class for padding smaller strings, *i.e.* $|\mathcal{A}| = K = 28$. Punctuations, and other symbols in the text are ignored; lower and upper case letters are mapped to the same class. Different synthetic datasets are generated as required by the experiments; the training sets consist of 100k image samples, while the tests set contain 1k samples. Figure 3 visualises some synthetically generated samples.

Real data. For testing the validity of our method on real text images, we use a scanned historical printed book from the Google1000 dataset [21]. Specifically, we use the book titled *Observations on the Nature and Cure of Gout* by James Parkinson [61]. For simplicity, we discard cover, title and start-of-chapter pages, and pages with significant number of footnotes; we only work with the remaining 140 pages (total 200 pages) which contain text in a relatively uniform font. Nevertheless, this data is still challenging due to: (1) non-fixed-width font which makes character segmentation difficult, (2) varying spacing between words due to fully justified alignment, (3) varying case (lower/upper) and italics, (4) different background colours and textures, (5) show-through from the back of the page, (6) fading and other noise elements, and (7) presence of various punctuations and other symbols. We use the localisation output of the provided OCR engine output to segment the pages into lines; first 300 lines are assigned to the test set, while the remaining 3000 form the training set (no page is shared between the splits). We use the provided OCR text output for lines in the training split, as examples of valid text strings for the discriminator. Note, these strings are sampled uniformly at random during training, and hence, do not have any direct correspondence to images in the training batch. The text lines typically consist of ≈50 characters. The character-set consists of 26 English letters, one space character, one unknown <UNK> character, and one null class for padding, for a total of $|\mathcal{A}| = K = 29$ characters. We do not distinguish between upper and lower cases; the following symbols and punctuations: , . ? ! ' " * ( ) are suppressed (ignored), and any other character is mapped to <UNK>. Figure 4 visualises some sample text-lines.

Metrics. We measure accuracy at the *character* and *word* levels:

- **character accuracy**: this is computed as $1 - \frac{1}{N}\sum_{i=1}^{N}\frac{\text{EditDist}\left(\mathbf{y}_{\text{gt}}^{(i)}, \mathbf{y}_{\text{pred}}^{(i)}\right)}{\text{Length}\left(\mathbf{y}_{\text{gt}}^{(i)}\right)}$, where $\mathbf{y}_{\text{gt}}^{(i)}$ and $\mathbf{y}_{\text{pred}}^{(i)}$ are the i^{th} ground-truth and predicted strings respectively in a dataset containing N strings; EditDist is the *character-level Levenshtein* distance [47]; Length $\left(\mathbf{y}_{\text{gt}}\right)$ is the number of characters in $\mathbf{y}_{\text{gt}}$.
- **word accuracy**: computed as *character accuracy* above, but here the *Levenshtein* distance uses *words* (contiguous strings demarcated by space) as tokens, and is normalized by number of ground-truth words in $\mathbf{y}_{\text{gt}}$.

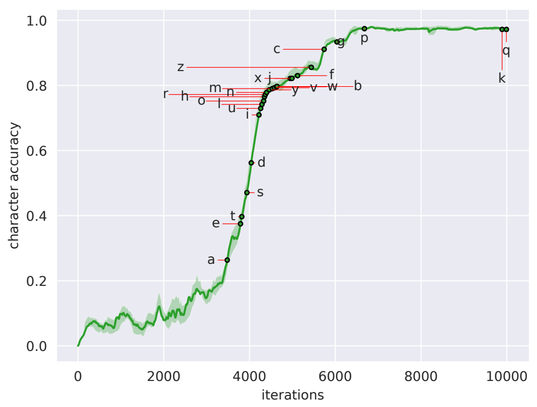


Figure 9: Temporal learning order. Training iterations for a model trained on length-7 words, annotated at the steps when it becomes at least 50% accurate for each character. This curve corresponds to `run-#4` in fig. 6. The characters are learnt in the order of their frequencies (see section 5.3).

5.2 Effect of text length on convergence

Although earlier works use low-order, namely uni/bi-gram statistics for alignment [40, 46], higher-order *n*-grams could be more informative. In this experiment we examine the impact of the *length* of the training text-sequences on convergence. We train separate models on synthetic datasets containing *one* word of a given length, namely — {3,5,7,9,11,13}. Figure 7 tracks *character accuracy* as the training progresses; due to instabilities in training GANs, we train on each word-length eight times, and plot the run with the maximum area-under-curve (AUC) (earliest "take-off"). Note, models trained on longer words converge faster, achieving ≈99% *character accuracy.* In detail, the model trained on length-13 words converges the fastest, followed by those trained on 11, 9, and 7 (in order). No convergence is seen for shorter lengths 3 and 5 (although the accuracy is higher for 5). This confirms that longer text-sequences impose stronger structural constraints on the possible outputs, leading to faster convergence. Figure 8 visualises the confusion matrices for models trained on lengths 5 and 7; the length-5 model confuses most characters, whereas the length-7 model recognises most characters almost perfectly. Note, convergence is independent of the *number of distinct word-instances* for a given length: there are more length-5 words (≈7000) than length-13 (≈2500) in the lexicon, yet training with length-5 words does not converge. Further, words of length 7 are the most in number (≈13000), yet it converges last.

5.3 Which character is learnt first?

We examine the dynamics of learning, more specifically, we probe the order in which the model learns about different symbols — is there a pattern? Figure 6 visualises the order in which models (trained on synthetic word images of length 7) achieve an accuracy of at least 50% for each character. We note that this ranking is highly correlated with the frequency of the characters in the English language — Spearman's rank correlation coefficient $\rho = 0.80$, p-value $< 1e{-}5$. It further visualises the variance in the ranking of the characters across multiple runs. The characters at the extremities of the frequency distribution have low variance — common characters (*e.g.* `e,s,i,a`) are almost always learnt first, and the least common

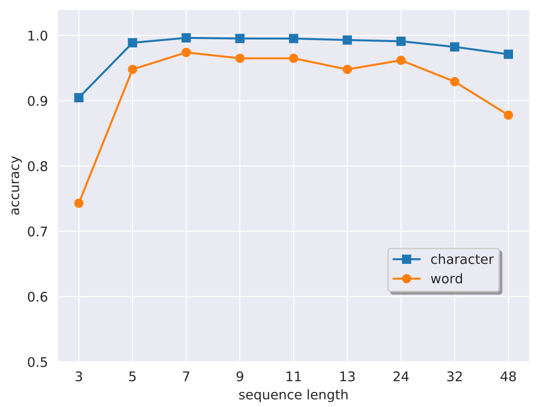


Figure 10: Generalisation to different sequence lengths. A model trained on text-strings of length 24, is evaluated on images containing both shorter and longer strings of lengths — $\{3, 5, 7, 9, 11, 13, 32, 48\}$. Word and character accuracies are plotted. The fully-convolutional architecture of the recognition network enables significant generalisation to lengths not in the training set, with small variance in performance (see section 5.4).

characters (*e.g.* `z,x,j,q`) are learnt last; while characters in the middle, *viz.* {`g,p`} show the highest variance. The character {`g`} is a curious exception as it is sometimes learnt first. This is because of 8.54% of the training (length-7) words end in the suffix '`-ing`'. Hence, {`g`} appears at the last position quite frequently, and becomes relatively easy to learn. Figure 9 annotates the training steps at which the accuracy for a character first reaches 50%. After the model becomes confident about the first symbol {`a`}, it quickly learns the other most commons ones; then it slowly learns the less frequent symbols in the order of their frequencies. Further, fig. 8 visualises the confusion-matrices for models trained on word-lengths 5 and 7. Again, we can note the dependence on character frequencies — even though model for length-5 words does not converge (see section 5.2), it is somewhat accurate about the frequent character {`s`}, while the length-7 model is almost perfect at recognising most characters, yet it confuses two of the least common characters {`j,x`}.

5.4 Generalisation to different lengths

The fully-convolutional architecture of our recognition network generalises to images of lengths significantly different from those it was trained on. To demonstrate this, we train a model on synthetic *text-strings* of length 24 (containing multiple words), and evaluate on synthetic images of different lengths: (1) *shorter* single-word images of lengths — {3, 5, 7, 9, 11, 13}, and (2) *longer* text-string (multiple words) images of lengths — {32, 48}. Figure 10 plots the recognition accuracy against the word lengths. We note excellent and consistent *character* (≈99%) and *word* accuracies (≈95%) for both, shorter and longer lengths (5 — 32). Note, this demonstrates significant generalisation ability, as the model is never trained on such images. There is a drop in the character accuracy (≈95%) for length-48 text-strings, as the model does not learn a long-range language model. Performance suffers for words of length 3 due to image-edges being close in short images, which is not encountered during training with images of long words.

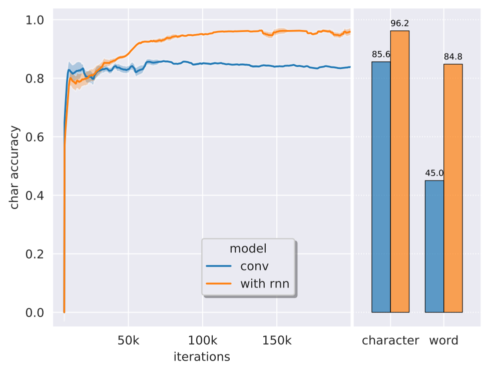


Figure 11: Recognising historical printed books. [left] *Character & word accuracy* on the test split of the *real* dataset (see fig. 4) for both the fully-convolutional and the *skip-RNN* recognition models. *Skip-RNN* dramatically improves: *character accuracy* from 85.6% to 96.2%, and *word accuracy* from 45.0% to 84.8%. *Character accuracy* is also visualised against training iterations (see section 5.6).

5.5 Varying the text corpus

We examine the impact of varying the text corpus from which samples of text strings are obtained, on the recognition accuracy. We examine the following three different sources for sampling the strings. The synthetic text-images are held constant across the three settings, and contain up to 24 characters with the text content in them sampled from WMT newscrawl (as before). (1) same strings as in text-images but randomly sampled for each batch, (2) strings from the same corpus (WMT newscrawl) but with no overlap with text-image strings, and (3) strings sampled from a very different corpus, namely, Tolstoy's *War and Peace*. Table 1 summarizes the *character* and *word* accuracies. Training with the completely unrelated lexicon (#3) does have a small adverse effect (*word* accuracy drops to 92.53% from ≈ 95%), while using a related lexicon does (#2) not have such an effect.

5.6 Recognising a historical printed book

Finally, we apply our model to *real* text-line images extracted from a historical printed book (see section 5.1 for dataset details). As noted in section 5.1, non-fixed width fonts and fully-justified text alignment introduce non-uniform spacing between characters and words. This poses a significant challenge to the fully-convolutional recogniser, making segmentation of the text-image into individual characters difficult (see fig. 4 for example images). Hence, we augment the penultimate layer of the fully-convolutional recogniser with a *skip-RNN* — a *uni*-directional (left to right) RNN (256-dimensional LSTM) with a residual skip-connection [25] (see fig. 12). The RNN lends pliability to the convolutional features, thereby aids character segmentation. All other model parameters are as those used for the synthetic data experiments (see section 4), except: (1) the discriminator filter size is increased from 5 to 11, and (2) number of layers is doubled to 8 to exploit the long-term structure in the much longer text strings (≈50 characters each). Figure 11 visualises the *character* and *word* accuracies for both the fully-convolutional and *skip-RNN* recognition models. *Skip-RNN* dramatically improves the recognition performance: *word accuracy* improves from 45.0% to 84.8%, while *character accuracy* improves from 85.6% to 96.2%. Figure 4 visualises randomly selected examples from the test set and shows the model's predictions; the predictions are comparable to the "ground-truth" annotations obtained from Google's OCR engine. Full page read-outs from our model are visualised in the supplementary material.

corpus →	(1) WMT	(2) WMT no overlap	(3) War & Peace
char	98.98	99.13	98.43
word	95.33	96.08	92.53

Table 1: Effect of varying the text corpus on recognition accuracy. Text-strings are sampled from three increasingly distant text corpora. This has a small adverse effect on recognition *word* and *character* accuracies (in %) (see section 5.5).

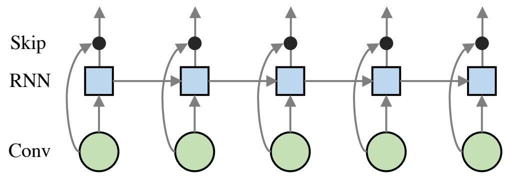


Figure 12: Skip-RNN architecture for real text images. Non-uniform spacing and non-fixed width fonts pose a significant challenge to the fully-convolutional recogniser. We augment the recognition network with a *skip-RNN*, which acts on the convolutional features, and predicts residual updates to the inputs (the residual predictions are *added* to the inputs). This improves the *word accuracy* from ≈45% to ≈85% (see fig. 11).

6 CONCLUSION

We have developed a method for training a text recognition network using only *unaligned* examples of text-images and valid text strings. We have presented detailed analysis for various aspects of the proposed method. We have established — (1) positive correlation between the length of the input text and convergence rates; (2) the order in which the characters are learnt is strongly dependent on their relative frequencies in the text; (3) the generalisation ability of our method to input images of different lengths, specifically our recognition model trained on strings of length 24 generalises to both much shorter and longer strings (3 – 48) without drastic degradation in performance; (4) the effect of varying the text corpus used as the source of valid sentences on the recognition accuracy. Finally, we have shown successful recognition on real text images, without using any labelled supervisory data. These results open up a new and promising direction for training sequence recognition models for structured domains (*e.g.* language) given no labelled training data. The proposed method is applicable not just to text images, but other modalities as well, *e.g.* speech and gestures.

ACKNOWLEDGMENTS

We thank Iasonas Kokkinos, Triantafyllos Afouras, and Weidi Xie for insightful discussions, and anonymous reviewers for their detailed feedback. Financial support was provided by UK EPSRC AIMS CDT EP/L015987/2, EPSRC Seebibyte Grant EP/M013774/1, and Clarendon Fund scholarship.

REFERENCES

[1] EMNLP conference on machine translation, 2018.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] N. Aldarrab. Decipherment of historical manuscripts. Master's thesis, University of Southern California, 2017.

[4] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE PAMI*, 36:2552–2566, 2014.

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

[6] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher. ICDAR 2013 competition on historical book recognition (hbr 2013). pages 1459–1463. IEEE, 2013.

[7] M. Artetxe, G. Labaka, E. Agirre, and K. Cho. Unsupervised neural machine translation. In *Proc. ICLR*, 2017.

[8] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[9] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015.

[10] T. Berg-Kirkpatrick, G. Durrett, and D. Klein. Unsupervised transcription of historical documents. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 207–217, 2013.

[11] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. PhotoOCR: Reading text in uncontrolled conditions. In *Proc. ICCV*, 2013.

[12] H. Bunke, S. Bengio, and A. Vinciarelli. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *PAMI*, 26(6):709–720, 2004.

[13] R. G. Casey. Text OCR by solving a cryptogram. 1986.

[14] 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.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. 39 B:1–38, 1977.

[16] J. F. Dooley. *A brief history of cryptology and cryptographic algorithms*. Springer, 2013.

[17] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[18] V. Goel, A. Mishra, K. Alahari, and C. V. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *International Conf. on Document Analysis and Recognition (ICDAR)*, pages 398–402, 2013.

[19] A. N. Gomez, S. Huang, I. Zhang, B. M. Li, M. Osama, and L. Kaiser. Unsupervised cipher cracking using discrete GANs. In *Proc. ICLR*, 2018.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, 2014.

[21] Google Inc. Book search dataset, Aug 2018. Version V.

[22] A. Gordo. Supervised mid-level features for word image representation. In *Proc. CVPR*, 2015.

[23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

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

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[26] P. He, W. Huang, Y. Qiao, C. 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.

[27] T. K. Ho and G. Nagy. OCR with no shape training. In *Proc. ICPR*, 2000.

[28] G. Huang, E. Learned-Miller, and A. McCallum. Cryptogram decoding for optical character recognition. 2007.

[29] Hunspell. https://hunspell.github.io.

[30] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[31] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, 2017.

[32] 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.

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

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

[35] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 116(1):1–20, Jan. 2016.

[36] A. Kae and E. Learned-Miller. Learning on the fly: font-free approaches to difficult OCR problems. 2009.

[37] 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.

[38] K. Knight, A. Nair, N. Rathod, and K. Yamada. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL*, pages 499–506. Association for Computational Linguistics, 2006.

[39] K. Knight, B. Megyesi, and C. Schaefer. The Copiale cipher. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*. Association for Computational Linguistics, 2011.

[40] M. Kozielski, M. Nuhn, P. Doetsch, and H. Ney. Towards unsupervised learning for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 549–554. IEEE, 2014.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1106–1114, 2012.

[42] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. In *Proc. ICLR*, 2017.

[43] 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.

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

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

[46] D.-S. Lee. Substitution deciphering based on HMMs with applications to compressed document processing. *PAMI*, (12):1661–1666, 2002.

[47] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707, 1966.

[48] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proc. ECCV*, pages 702–716. Springer, 2016.

[49] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Proc. NIPS*, pages 700–708, 2017.

[50] Y. Liu, J. Chen, and L. Deng. Unsupervised sequence classification using sequential output statistics. In *Proc. NIPS*, pages 3550–3559, 2017.

[51] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015.

[52] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3, 2013.

[53] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *Proc. ICCV*, pages 2813–2821. IEEE, 2017.

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

[55] A. Mishra, K. Alahari, and C. Jawahar. Top-down and bottom-up cues for scene text recognition. In *Proc. CVPR*, 2012.

[56] G. Nagy. Efficient algorithms to decode substitution ciphers with applications to OCR. In *Proc. ICPR*, pages 352–355, 1986.

[57] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS DLW*, volume 2011, 2011.

[58] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Proc. CVPR*, volume 3, pages 1187–1190. IEEE, 2012.

[59] 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.

[60] M. Nuhn and H. Ney. Decipherment complexity in 1: 1 substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 615–621, 2013.

[61] J. Parkinson. *Observations on the Nature and Cure of Gout: On Nodes of the Joints; and on the Influence of Certain Articles of Diet, in Gout, Rheumatism, and Gravel.* Symonds, 1805.

[62] S. Peleg and A. Rosenfeld. Breaking substitution ciphers using a relaxation algorithm. *Communications of the ACM*, 22(11):598–605, 1979.

[63] A. Poznanski and L. Wolf. CNN-N-Gram for handwriting word recognition. In *Proc. CVPR*, 2016.

[64] S. Ravi and K. Knight. Attacking decipherment problems optimally with low-order n-gram models. In *proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 812–819. Association for Computational Linguistics, 2008.

[65] J. A. Rodriguez-Serrano, A. Gordo, and F. Perronnin. Label embedding: A frugal baseline for text recognition. *International Journal of Computer Vision*, 113(3):193–207, 2015.

[66] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *ArXiv e-prints*, 2015.

[67] B. Shi, X. Wang, P. Lv, C. Yao, and X. Bai. Robust scene text recognition with automatic rectification. In *Proc. CVPR*, 2016.

[68] 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.

[69] R. Smith. An overview of the Tesseract OCR engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.

[70] B. Snyder, R. Barzilay, and K. Knight. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1048–1057. Association for Computational Linguistics, 2010.

[71] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *Proc. ICML*, 2015.

[72] B. Su and S. Lu. Accurate scene text recognition based on recurrent neural network. In *Proc. ACCV*, 2014.

[73] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112, 2014.

[74] I. Sutskever, R. Jozefowicz, K. Gregor, D. Rezende, T. Lillicrap, and O. Vinyals. Towards principled unsupervised learning. In *ICLR workshop*, 2016.

[75] Tesseract OCR. https://github.com/tesseract-ocr/, 1985 – 2018.

[76] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[77] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. ECCV*. Springer, 2010.

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

[79] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Proc. ICPR*, pages 3304–3308. IEEE, 2012.

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

[81] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, 2017.