Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition

Max Jaderberg **Karen Simonyan** **Andrea Vedaldi** **Andrew Zisserman**
Visual Geometry Group, University of Oxford
{max,karen,vedaldi,az}@robots.ox.ac.uk

Abstract

In this work we present a framework for the recognition of natural scene text. Our framework does not require any human-labelled data, and performs word recognition on the whole image holistically, departing from the character based recognition systems of the past. The deep neural network models at the centre of this framework are trained solely on data produced by a synthetic text generation engine – synthetic data that is highly realistic and sufficient to replace real data, giving us infinite amounts of training data. This excess of data exposes new possibilities for word recognition models, and here we consider three models, each one "reading" words in a different way: via 90k-way dictionary encoding, character sequence encoding, and bag-of-N-grams encoding. In the scenarios of language based and completely unconstrained text recognition we greatly improve upon state-of-the-art performance on standard datasets, using our fast, simple machinery and requiring zero data-acquisition costs.

1 Introduction

Text recognition in natural images, *scene text recognition*, is a challenging but wildly useful task. Text is one of the basic tools for preserving and communicating information, and a large part of the modern world is designed to be interpreted through the use of labels and other textual cues. This makes scene text recognition imperative for many areas in information retrieval, in addition to being crucial for human-machine interaction.

While the recognition of text within scanned documents is well studied and there are many document OCR systems that perform very well, these methods do not translate to the highly variable domain of scene text recognition. When applied to natural scene images, traditional OCR techniques fail as they are tuned to the largely black-and-white, line-based environment of printed documents, while text occurring in natural scene images suffers from inconsistent lighting conditions, variable fonts, orientations, background noise, and imaging distortions.

To effectively recognise scene text, there are generally two stages: word detection and word recognition. The detection stage generates a large set of word bounding box candidates, and is tuned for speed and high recall. Previous work uses sliding window methods [26] or region grouping methods [5, 6, 19] very successfully for this. Subsequently, these candidate detections are recognised, and this recognition process allows for filtering of false positive word detections. Recognition is therefore a far more challenging problem and it is the focus of this paper.

While most approaches recognize individual characters by pooling evidence locally, Goodfellow *et al.* [8] do so from the image of the whole character string using a convolutional neural network (CNN) [14]. They apply this to street numbers and synthetic CAPTCHA recognition obtaining excellent results. Inspired by this approach, we move further in the direction of holistic word classification for scene text, and make two important contributions. Firstly, we propose a state-of-the-art CNN text recogniser that also pools evidence from images of entire words. Crucially, however, we regress all the characters simultaneously, formulating this as a classification problem in a large lexicon of 90k possible words (Sect. 3.1). In order to do so, we show how CNNs can be efficiently

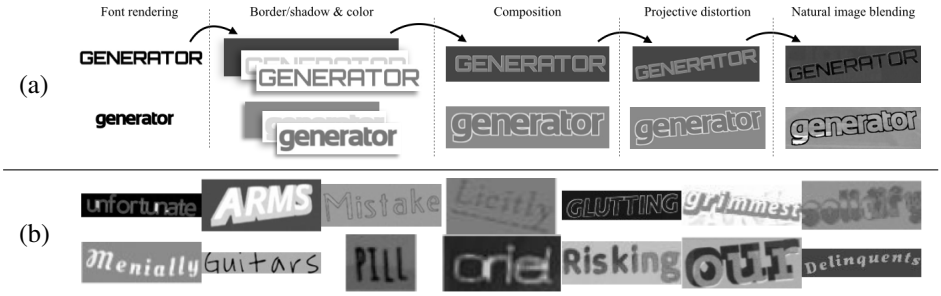

Figure 1: (a) The text generation process after font rendering, creating and coloring the image-layers, applying projective distortions, and after image blending. (b) Some randomly sampled data created by the synthetic text engine.

trained to recognise a very large number of words using *incremental training*. While our lexicon is restricted, it is so large that this hardly constitutes a practical limitation. Secondly, we show that this state-of-the-art recogniser can be trained *purely from synthetic data*. This result is highly non-trivial as, differently from CAPTCHA, the classifier is then applied to real images. While synthetic data was used previously for OCR, it is remarkable that this can be done for scene text, which is significantly less constrained. This allows our framework to be seamlessly extended to larger vocabularies and other languages without any human-labelling cost. In addition to these two key contributions, we study two alternative models – a character sequence encoding model with a modified formulation to that of [8] (Sect. 3.2), and a novel bag-of-N-grams encoding model which predicts the unordered set of N-grams contained in the word image (Sect. 3.3).

A discussion of related work follows immediately and our data generation system described after in Sect. 2. Our deep learning word recognition architectures are presented in Sect. 3, evaluated in Sect. 4, and conclusions are drawn in Sect. 5.

Related work. Traditional text recognition methods are based on sequential character classification by either sliding windows [11, 26, 27] or connected components [18, 19], after which a word prediction is made by grouping character classifier predictions in a left-to-right manner. The sliding window classifiers include random ferns [22] in Wang *et al.* [26], and CNNs in [11, 27]. Both [26] and [27] use a small fixed lexicon as a language model to constrain word recognition.

More recent works such as [2, 3, 20] make use of over-segmentation methods, guided by a supervised classifier, to generate candidate proposals which are subsequently classified as characters or false positives. For example, PhotoOCR [3] uses binarization and a sliding window classifier to generate candidate character regions, with words recognised through a beam search driven by classifier scores followed by a re-ranking using a dictionary of 100k words. [11] uses the convolutional nature of CNNs to generate response maps for characters and bigrams which are integrated to score lexicon words.

In contrast to these approaches based on character classification, the work by [7, 17, 21, 24] instead uses the notion of holistic word recognition. [17, 21] still rely on explicit character classifiers, but construct a graph to infer the word, pooling together the full word evidence. Rodriguez *et al.* [24] use aggregated Fisher Vectors [23] and a Structured SVM framework to create a joint word-image and text embedding. [7] use whole word-image features to recognize words by comparing to simple black-and-white font-renderings of lexicon words.

Goodfellow *et al.* [8] had great success using a CNN with multiple position-sensitive character classifier outputs (closely related to the character sequence model in Sect. 3.2) to perform street number recognition. This model was extended to CAPTCHA sequences (up to 8 characters long) where they demonstrated impressive performance using synthetic training data for a synthetic problem (where the generative model is known), but we show that synthetic training data can be used for a real-world data problem (where the generative model is unknown).

2 Synthetic Data Engine

This section describes our scene text rendering algorithm. As our CNN models take whole word images as input instead of individual character images, it is essential to have access to a training dataset of cropped word images that covers the whole language or at least a target lexicon. While

there are some publicly available datasets from ICDAR [13, 15, 16, 25], the Street View Text (SVT) dataset [26] and others, the number of full word image samples is only in the thousands, and the vocabulary is very limited. These limitations have been mitigated before by mining for data or having access to large proprietary datasets [3, 11], but neither of these approaches are wholly accessible or scalable.

Here we follow the success of some synthetic character datasets [4, 27] and create a synthetic word data generator, capable of emulating the distribution of scene text images. This is a reasonable goal, considering that much of the text found in natural scenes is computer-generated and only the physical rendering process (*e.g.* printing, painting) and the imaging process (*e.g.* camera, viewpoint, illumination, clutter) are not controlled by a computer algorithm.

Fig. 1 illustrates the generative process and some resulting synthetic data samples. These samples are composed of three separate image-layers – a background image-layer, foreground image-layer, and optional border/shadow image-layer – which are in the form of an image with an alpha channel. The synthetic data generation process is as follows:

1. *Font rendering* – a font is randomly selected from a catalogue of over 1400 fonts downloaded from Google Fonts. The kerning, weight, underline, and other properties are varied randomly from arbitrarily defined distributions. The word is rendered on to the foreground image-layer's alpha channel with either a horizontal bottom text line or following a random curve.
2. *Border/shadow rendering* – an inset border, outset border or shadow with a random width may be rendered from the foreground.
3. *Base coloring* – each of the three image-layers are filled with a different uniform color sampled from clusters over natural images. The clusters are formed by k-means clustering the three color components of each image of the training datasets of [16] into three clusters.
4. *Projective distortion* – the foreground and border/shadow image-layers are distorted with a random, full-projective transformation, simulating the 3D world.
5. *Natural data blending* – each of the image-layers are blended with a randomly-sampled crop of an image from the training datasets of ICDAR 2003 and SVT. The amount of blend and alpha blend mode (*e.g.* normal, add, multiply, burn, max, *etc.*) is dictated by a random process, and this creates an eclectic range of textures and compositions. The three image-layers are also blended together in a random manner, to give a single output image.
6. *Noise* – Gaussian noise, blur, and JPEG compression artefacts are introduced to the image.

The word samples are generated with a fixed height of 32 pixels, but with a variable width. Since the input to our CNNs is a fixed-size image, the generated word images are rescaled so that the width equals 100 pixels. Although this does not preserve the aspect ratio, the horizontal frequency distortion of image features most likely provides the word-length cues. We also experimented with different padding regimes to preserve the aspect ratio, but found that the results are not quite as good as with resizing.

The synthetic data is used in place of real-world data, and the labels are generated from a corpus or dictionary as desired. By creating training datasets much larger than what has been used before, we are able to use data-hungry deep learning algorithms to train richer, whole-word-based models.

3 Models

In this section we describe three models for visual recognition of scene text words. All use the same framework of generating synthetic text data (Sect. 2) to train deep convolutional networks on whole-word image samples, but with different objectives, which correspond to different methods of reading. Sect. 3.1 describes a model performing pure word classification to a large dictionary, explicitly modelling the entire known language. Sect. 3.2 describes a model that encodes the character at each position in the word, making no language assumptions to naively predict the sequence of characters in an image. Sect. 3.3 describes a model that encodes a word as a bag-of-N-grams, giving a compositional model of words as not only a collection of characters, but of 2-grams, 3-grams, and more generally, N-grams.

3.1 Encoding Words

This section describes our first model for word recognition, where words w are constrained to be selected in a pre-defined dictionary $\mathcal{W}$. We formulate this as multi-class classification problem,

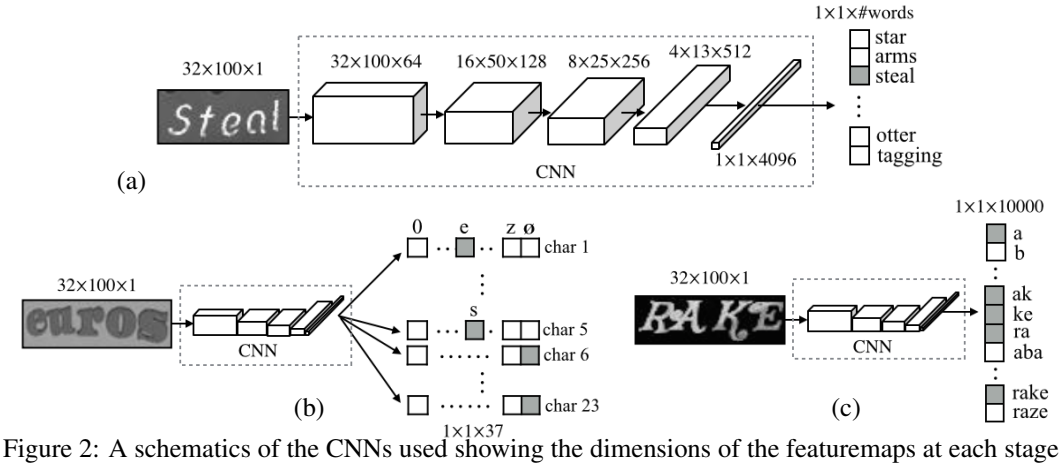

Figure 2: A schematics of the CNNs used showing the dimensions of the featuremaps at each stage for (a) dictionary encoding, (b) character sequence encoding, and (c) bag-of-N-gram encoding. The same five-layer, base CNN architecture is used for all three models.

with one class per word. While the dictionary $\mathcal{W}$ of a natural language may seem too large for this approach to be feasible, in practice an advanced English vocabulary, including different word forms, contains only around 90k words, which is large but manageable.

In detail, we propose to use a CNN classifier where each word $w \in \mathcal{W}$ in the lexicon corresponds to an output neuron. We use a CNN with four convolutional layers and two fully connected layers. Rectified linear units are used throughout after each weight layer except for the last one. In forward order, the convolutional layers have 64, 128, 256, and 512 square filters with an edge size of 5, 5, 3, and 3. Convolutions are performed with stride 1 and there is input feature map padding to preserve spatial dimensionality. 2×2 max-pooling follows the first, second and third convolutional layers. The fully connected layer has 4096 units, and feeds data to the final fully connected layer which performs classification, so has the same number of units as the size of the dictionary we wish to recognize. The predicted word recognition result w^* out of the set of all dictionary words $\mathcal{W}$ in a language $\mathcal{L}$ for a given input image x is given by $w^* = \arg\max_{w \in \mathcal{W}} P(w|x, \mathcal{L})$. Since $P(w|x, \mathcal{L}) = \frac{P(w|x)P(w|\mathcal{L})P(x)}{P(x|\mathcal{L})P(w)}$ and with the assumptions that x is independent of $\mathcal{L}$ and that prior to any knowledge of our language all words are equally probable, our scoring function reduces to $w^* = \arg\max_{w \in \mathcal{W}} P(w|x)P(w|\mathcal{L})$. The per-word output probability $P(w|x)$ is modelled by the softmax scaling of the final fully connected layer, and the language based word prior $P(w|\mathcal{L})$ can be modelled by a lexicon or frequency counts. A schematic of the network is shown in Fig. 2 (a).

Training. We train the network by back-propagating the standard multinomial logistic regression loss with dropout [10], which improves generalization. Optimization uses stochastic gradient descent (SGD), dynamically lowering the learning rate as training progresses. With uniform sampling of classes in training data, we found the SGD batch size must be at least a fifth of the total number of classes in order for the network to train.

For very large numbers of classes (*i.e.* over 5k classes), the SGD batch size required to train effectively becomes large, slowing down training a lot. Therefore, for large dictionaries, we perform *incremental training* to avoid requiring a prohibitively large batch size. This involves initially training the network with 5k classes until partial convergence, after which an extra 5k classes are added. The original weights are copied for the original 5k classes, with the new classification layer weights being randomly initialized. The network is then allowed to continue training, with the extra randomly initialized weights and classes causing a spike in training error, which is quickly trained away. This process of allowing partial convergence on a subset of the classes, before adding in more classes, is repeated until the full number of desired classes is reached. In practice for this network, the CNN trained well with initial increments of 5k classes, and after 20k classes is reached the number of classes added at each increment is increased to 10k.

3.2 Encoding Sequences of Characters

This section describes a different model for word recognition. Rather than having a single large dictionary classifier as in Sect. 3.1, this model uses a single CNN with multiple independent classifiers, each one predicting the character at each position in the word. This character sequence encoding

model is a complete departure from the dictionary-constrained model, as this allows entirely unconstrained recognition of words.

A word w of length N is modelled as a sequence of characters such that $w = (c_1, c_2, \ldots, c_N)$ where each $c_i \in \mathcal{C} = \{1, 2, \ldots, 36\}$ represents a character at position i in the word, from the set of 10 digits and 26 letters. Each c_i can be predicted with a single classifier, one for each character in the word. However, since words have variable length N which is unknown at test time, we fix the number of characters to 23, the maximum length of a word in the training set, and introduce a null character class. Therefore a word is represented by a string $w = (\mathcal{C} \cup \{\phi\})^{23}$. Then for a given image x, each character is predicted as $c_i^* = \arg\max_{c_i \in \mathcal{C} \cup \{\phi\}} P(c_i|\Phi(x))$. $P(c_i|\Phi(x))$ is given by the i-th classifier acting on a single set of shared CNN features $\Phi(x)$.

The base CNN has the same structure as the first five layers of Sect. 3.1: four convolutional layers followed by a fully connected layer, giving $\Phi(x)$. The output of the fully connected layer is then fed to 23 separate fully connected layers with 37 neurons each, one for each character class. These fully connected layers are independently softmax normalized and can be interpreted as the probabilities $P(c_i|\Phi(x))$ of the width-resized input image x. Fig. 2 (b) illustrates this model. The model is trained as in Sect. 3.1 on purely synthetic data by SGD with dropout regularisation, back-propagating gradients from each 23 softmax classifier to the base net.

Discussion. This sequential character encoding model is similar to the model used by Goodfellow *et al.* in [8]. Although the model of [8] is not applied to scene text (only street numbers and CAPTCHA puzzles), it uses a separate character classifier for each letter in the word, able to recognise numbers up to 5 digits long and CAPTCHAs up to 8 characters long. However, rather than incorporating a no-character class in each character positions's classifier, a further length classifier is trained to output the predicted length of the word. This requires a final post-processing stage to find the optimal word prediction given the character classifier outputs and the length classifier output. We achieve a similar effect but without requiring any post processing – the word can be read directly from the CNN output, stripping the no-character class predictions.

3.3 Encoding Bags of N-grams

This section describes our last word recognition model, which exploits compositionality to represent words. In contrast to the sequential character encoding of Sect. 3.2, words can be seen as a composition of an unordered set of character N-grams, a *bag-of-N-grams*. In the following, if $s \in \mathcal{C}^N$ and $w \in \mathcal{C}^M$ are two strings, the symbol $s \subset w$ indicates that s is a substring of w. An N-gram of word w is a substring $s \subset w$ of length $|w| = N$. We will denote with $G_N(w) = \{s : s \subset w \wedge |s| \leq N\}$ the set of all grams of word w of length up to N and with $G_N = \cup_{w \in \mathcal{W}} G_N(w)$ the set of all such grams in the language. For example, $G_3(\text{spires}) = \{\text{s}, \text{p}, \text{i}, \text{r}, \text{e}, \text{s}, \text{sp}, \text{pi}, \text{ir}, \text{re}, \text{es}, \text{spi}, \text{pir}, \text{ire}, \text{res}\}$. This method of encoding variable length sequences is similar to *Wickelphone* phoneme-encoding methods [28].

Even for small values of N, $G_N(w)$ encodes each word $w \in \mathcal{W}$ nearly uniquely. For example, with $N = 4$, this map has only 7 collisions out of a dictionary of 90k words. The encoding $G_N(w)$ can be represented as a $|G_N|$-dimensional binary vector of gram occurrences. This vector is very sparse, as on average $|G_N(w)| \approx 22$ whereas $|G_N| = 10k$. Given w, we predict this vector using the same base CNN as in Sect. 3.1 and Sect. 3.2, but now have a final fully connected layer with $|G_N|$ neurons to represent the encoding vector. The scores from the fully connected layer can be interpreted as probabilities of an N-gram being present in the image by applying the logistic function to each neuron. The CNN is therefore learning to recognise the presence of each N-gram somewhere within the input image.

Training. With a logistic function, the training problem becomes that of $|G_N|$ separate binary classification tasks, and so we back-propagate the logistic regression loss with respect to each N-gram class independently. To jointly train a whole range of N-grams, some of which occur very frequently and some barely at all, we have to scale the gradients for each N-gram class by the inverse frequency of their appearance in the training word corpus. We also experimented with hinge loss and simple regression to train but found frequency weighted binary logistic regression was superior. As with the other models, we use dropout and SGD.

4 Evaluation

This section evaluates our three text recognition models. Sect. 4.1 describes the benchmark data, Sect. 4.2 the implementation details, and Sect. 4.3 the results of our methods, that improve on the state of the art.

4.1 Datasets

A number of standard datasets are used for the evaluation of our systems – ICDAR 2003, ICDAR 2013, Street View Text, and IIIT5k. **ICDAR 2003** [16] is a scene text recognition dataset, with the test set containing 251 full scene images and 860 groundtruth cropped images of the words contained with the full images. We follow the standard evaluation protocol by [2, 26, 27] and perform recognition on only the words containing only alphanumeric characters and at least three characters. The test set of 860 cropped word images is referred to as IC03. The lexicon of all test words is IC03-Full, and the per-image 50 word lexicons defined by [26] and used in [2, 26, 27] are referred to as IC03-50. There is also the lexicon of all groundtruth test words – IC03-Full which contains 563 words. **ICDAR 2013** [13] test dataset contains 1015 groundtruth cropped word images from scene text. Much of the data is inherited from the ICDAR 2003 datasets. We refer to the 1015 groundtruth cropped words as IC13. **Street View Text** [26] is a more challenging scene text dataset than the ICDAR datasets. It contains 250 full scene test images downloaded from Google Street View. The test set of 647 groundtruth cropped word images is referred to as SVT. The lexicon of all test words is SVT-Full (4282 words), and the smaller per-image 50 word lexicons defined by [26] and used in [2, 3, 26, 27] are referred to as SVT-50. **IIIT 5k-word** [17] test dataset contains 3000 cropped word images of scene text downloaded from Google image search. Each image has an associated 50 word lexicon (IIIT5k-50) and 1k word lexicon (IIIT5k-1k).

For training, validation and large-lexicon testing we generate datasets using the synthetic text engine from Sect. 2. 4 million word samples are generated for the IC03-Full and SVT-Full lexicons each, referred to as Synth-IC03 and Synth-SVT respectively. In addition, we use the dictionary from Hunspell, a popular open source spell checking system, combined with the ICDAR and SVT test words as a 50k word lexicon. The 50k Hunspell dictionary can also be expanded to include different word endings and combinations to give a 90k lexicon. We generate 9 million images for the 50k word lexicon and 9 million images for the 90k word lexicon. The 9 million image synthetic dataset covering 90k words, Synth, is available for download at `http://www.robots.ox.ac.uk/~vgg/data/text/`.

4.2 Implementation Details

We perform experiments on all three encoding models described in Sect. 3. We will refer to the three models as DICT, CHAR, and NGRAM for the dictionary encoding model, character sequence encoding model, and N-gram encoding model respectively. The input images to the CNNs are greyscale and resized to 32×100 without aspect ratio preservation. The only preprocessing, performed on each sample individually, is the sample mean subtraction and standard deviation normalization (after resizing), as this was found to slightly improve performance. Learning uses a custom version of `Caffe` [12].

All CNN training is performed solely on the Synth training datasets, with model validation performed on a 10% held out portion. The number of character classifiers in the CHAR character sequence encoding models is set to 23 (the length of the largest word in our 90k dictionary). In the NGRAM models, the number of N-grams in the N-gram classification dictionary is set to 10k. The N-grams themselves are selected as the N-grams with at least 10 appearances in the 90k word corpus – this equates to 36 1-grams (the characters), 522 2-grams, 3965 3-grams, and 5477 4-grams, totalling 10k.

In addition to the CNN model defined in Sect. 3, we also define larger CNN, referred to as DICT+2, CHAR+2, and NGRAM+2. The larger CNN has an extra 3×3 convolutional layer with 512 filters before the final pooling layer, and an extra 4096 unit fully connected layer after the original 4096 unit fully connected layer. Both extra layers use rectified linear non-linearities. Therefore, the total structure for the DICT+2 model is conv-pool-conv-pool-conv-conv-pool-conv-fc-fc-fc, where conv is a convolutional layer, pool is a max-pooling layer and fc is a fully connected layer. We train these larger models to investigate the effect of additional model capacity, as the lack of over-fitting experienced on the basic models is suspected to indicate under-capacity of the models.

Model	Trained Lexicon	Synth	IC03-50	IC03	SVT-50	SVT	IC13
DICT-IC03-Full	IC03-Full	98.7	99.2	98.1	-	-	-
DICT-SVT-Full	SVT-Full	98.7	-	-	96.1	87.0	-
DICT-50k	50k	93.6	99.1	92.1	93.5	78.5	92.0
DICT-90k	90k	90.3	98.4	90.0	93.7	73.0	86.3
DICT+2-90k	90k	95.2	98.7	93.1	95.4	80.7	90.8
CHAR	90k	71.0	94.2	77.0	87.8	56.4	68.8
CHAR+2	90k	86.2	96.7	86.2	92.6	68.0	79.5
NGRAM-NN	90k	25.1	92.2	-	84.5	-	-
NGRAM+2-NN	90k	27.9	94.2	-	86.6	-	-

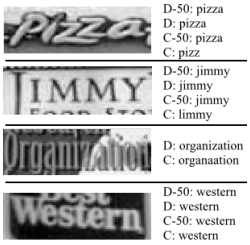

Table 1: *Left:* The word recognition accuracy for the different proposed models with different trained lexicons. Where a lexicon is not specified for a dataset, the only language constraints are those imposed by the model itself. The fixed lexicon CHAR model results (IC03-50 and SVT-50) are obtained by selecting the lexicon word with the minimum edit distance to the predicted character sequence. *Right:* Some random example results from the SVT and ICDAR 2013 dataset. D denotes DICT+2-90k with no lexicon, D-50 the DICT+2-90k model constrained to the image's 50 word lexicon, C denotes the CHAR+2 model with completely unconstrained recognition, and C-50 gives the result of the closest edit distance 50-lexicon word.

4.3 Experiments

We evaluate each of our three models on challenging text recognition benchmarks. First, we measure the accuracy on a large dataset, containing the images of words from the full lexicon (up to 90k words depending on the model). Due to the lack of human-annotated natural image datasets of such scale, we use the test split of our Synth dataset (Sect. 4.1). This allows us to assess how well our models can discriminate between a large number of words. Second, we consider the standard benchmarks IC03 [16], SVT [26], and IC13 [13], which contain natural scene images, but cover smaller word lexicons. The evaluation on these datasets allows for a fair comparison against the state of the art. The results are shown in Table 1 and Table 2.

Dictionary Encoding. For the DICT model, we train a model with only the words from the IC03-Full lexicon (DICT-IC03-Full), a model with only the words from the SVT-Full lexicon (DICT-SVT-Full), as well as models for the 50k and 90k lexicons – DICT-50k, DICT-90k, and DICT+2-90k. When a small lexicon is provided, we set the language prior $P(w|\mathcal{L})$ to be equal probability for lexicon words, otherwise zero. In the absence of a small lexicon, $P(w|\mathcal{L})$ is simply the frequency of word w in a corpus (we use the opensubtitles.org English corpus) normalized according to the power law.

The results in Table 1 show exceptional performance for the dictionary based models. When the model is trained purely for a dataset's corpus of words (DICT-IC03-Full and DICT-SVT-Full), the 50-lexicon recognition problem is largely solved for both ICDAR 2003 and SVT, achieving 99.2% and 96.1% word recognition accuracy respectively, that is 7 mistakes out of 860 in the ICDAR 2003 test set, of which most are completely illegible. The Synth dataset performs very closely to that of the ICDAR 2003 dataset, confirming that the synthetic data is close to the real world data.

Drastically increasing the size of the dictionary to 50k and 90k words gives very little degradation in 50-lexicon accuracy. However without the 50-lexicon constraint, as expected the 50k and 90k dictionary models perform significantly worse than when the dictionary is constrained to only the groundtruth words – on SVT, the word classification from only the 4282 groundtruth word set yields 87% accuracy, whereas increasing the dictionary to 50k reduces the accuracy to 78.5%, and the accuracy is further reduced to 73.0% with 90k word classes. Incorporating the extra layers in to the network with DICT+2-90k increases the accuracy a lot, giving 80.7% on SVT for full 90k-way classification, almost identical to a dictionary of 50k with the basic CNN architecture.

We also investigate the contribution that the various stages of the synthetic data generation engine make to real-world recognition accuracy. Figure 3 (*left*) shows DICT-IC03-Full and DICT-SVT-Full accuracy when trained identically but with different levels of sophistication of synthetic training data. As more sophisticated training data is used, the recognition accuracy increases – the addition of random image-layer colouring causing a significant increase in performance (+44% on IC03 and +40% on SVT), as does the addition of natural image blending (+1% on IC03 and +6% on SVT).

Character Sequence Encoding. The CHAR models are trained for character sequence encoding. The models are trained on image samples of words uniformly sampled from the 90k dictionary.

The output of the model are character predictions for a possible 23 characters of the test image's word. We take the predicted word as the MAP-optimal sequence of characters, stripping any no-

Model	IC03-50	IC03-Full	IC03-50k	SVT-50	SVT	IC13	IIIT5k-50	IIIT5k-1k
Baseline ABBYY [26]	56.0	55.0	-	35.0	-	-	24.3	-
Wang [26]	76.0	62.0	-	57.0	-	-	-	-
Mishra [17]	81.8	67.8	-	73.2	-	-	-	-
Novikova [21]	82.8	-	-	72.9	-	-	64.1	57.5
Wang & Wu [27]	90.0	84.0	-	70.0	-	-	-	-
Goel [7]	89.7	-	-	77.3	-	-	-	-
PhotoOCR [3]	-	-	-	90.4	78.0	87.6	-	-
Alsharif [2]	93.1	88.6	85.1	74.3	-	-	-	-
Almazan [1]	-	-	-	89.2	-	-	91.2	82.1
Yao [29]	88.5	80.3	-	75.9	-	-	80.2	69.3
Jaderberg [11]	96.2	91.5	-	86.1	-	-	-	-
Gordo [9]	-	-	-	90.7	-	-	93.3	86.6
DICT-IC03-Full	**99.2**	**98.1**	-	-	-	-	-	-
DICT-SVT-Full	-	-	-	**96.1**	**87.0**	-	-	-
DICT+2-90k	**98.7**	**98.6**	**93.3**	**95.4**	**80.7**	**90.8**	**97.1**	**92.7**
CHAR+2	**96.7**	**94.0**	**89.5**	**92.6**	68.0	79.5	**95.5**	85.4
NGRAM+2-SVM	**96.5**	**94.0**	-	-	-	-	-	-

Table 2: Comparison to previous methods. The ICDAR 2013 results given are case-insensitive. Bolded results outperform previous state-of-the-art methods. The baseline method is from a commercially available OCR system.

character classifications. The constrained lexicon results for IC03-50, IC03-Full, and SVT-50, are obtained by finding the lexicon word with the minimum edit distance to a raw predicted character sequence. Given this is a completely unconstrained recognition, with no language model at all, the results are surprisingly good. The 50-lexicon results are very competitive compared to the other encoding methods. However, we can see the lack of language constraints cause the out-of-lexicon results to be lacklustre, achieving an accuracy of only 79.5% with the CHAR+2 model on ICDAR 2013 as opposed to 90.8% with the DICT+2-90k model. As with the DICT models, increasing the number of layers in the network increases the word recognition accuracy by between 6-8%.

Some example word recognition results with dictionary and character sequence encodings are shown to the right of Table 1.

Bag-of-N-grams Encoding. The NGRAM model's output is thresholded to result in a binary activation vector of the presence of any of 10k N-grams in a test word. Decoding the N-gram activations into a word could take advantage of a statistical model of the language. Instead, we simply search for the word in the lexicon with the nearest (in terms of the Euclidean distance) N-gram encoding, denoted as NGRAM-NN and NGRAM+2-NN models. This extremely naive method still gives competitive performance, illustrating the discriminative nature of N-grams for word recognition. Instead, one could learn a linear SVM mapping from N-gram encoding to dictionary words, allowing for scalable word recognition through an inverted index of these mappings. We experimented briefly with this on the IC03-Full lexicon – training an SVM for each lexicon word from a training set of Synth data, denoted as NGRAM+2-SVM – and achieve 97% accuracy on IC03-50 and 94% accuracy on IC03-Full. Figure 3 (*right*) shows the N-gram recognition results for the NGRAM+2 model, thresholded at 0.99 probability.

Comparison & Discussion. Table 2 compares our models to previous work, showing that all three models achieve state-of-the-art results in different lexicon scenarios. With tightly constrained language models such as in DICT-IC03-Full and DICT-SVT-Full, we improve accuracy by +6%. However, even when the models are expanded to be mostly unconstrained, such as with DICT+2-90k, CHAR+2 and NGRAM+2-SVM, our models still outperform previous methods. Considering a complete absence of a language model, the no-lexicon recognition results for the CHAR+2 model on SVT and IC13 are competitive with the system of [3], and as soon as a language model is introduced in the form of a lexicon for SVT-50, the simple CHAR+2 model gives +2.2% accuracy over [3]. Performance could be further improved by techniques such as model averaging and test-sample augmentation, albeit at a significantly increased computational cost. Our largest model, the DICT+2-90k model comprised of over 490 million parameters, can process a word in 2.2ms on a single commodity GPU.

Our models set a new benchmark for scene text recognition. In a real-world system, the large DICT+2-90k model should be used for the majority of recognition scenarios unless completely unconstrained recognition is required where the CHAR+2 model can be used. However, when looking at the average edit distance of erroneous recognitions, the CHAR+2 model greatly outperforms the DICT+2-90k model, with an average error edit distance of 1.9 compared to 2.5 on IC13, suggesting

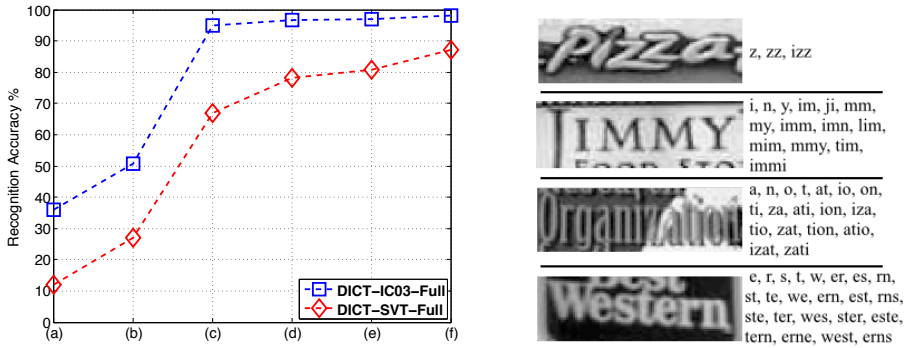

Figure 3: *Left:* The recognition accuracies of the DICT-IC03-Full and DICT-SVT-Full models evaluated on IC03 and SVT respectfully. The models (a-f) are trained on purely synthetic data with increasing levels of sophistication of the synthetic data. (a) Black text rendered on a white background with a single font, Droid Sans. (b) Incorporating all of Google fonts. (c) Adding background, foreground, and border colouring. (d) Adding perspective distortions. (e) Adding noise, blur and elastic distortions. (f) Adding natural image blending – this gives an additional 6.2% accuracy on SVT. *Right:* The N-gram recognition results with probability over 0.99 from the NGRAM+2 model on random test images from SVT and ICDAR 2013.

the CHAR+2 model may be more suitable for a retrieval style application in conjunction with a fuzzy search.

5 Conclusion

In this paper we introduced a new framework for scalable, state-of-the-art word recognition – synthetic data generation followed by whole word input CNNs. We considered three models within this framework, each with a different method for recognising text, and demonstrated the vastly superior performance of these systems on standard datasets. In addition, we introduced a new synthetic word dataset, orders of magnitude larger than any released before.

Acknowledgements

This work was supported by the EPSRC and ERC grant VisRec no. 228180. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. In *TPAMI*, 2014.

[2] O. Alsharif and J. Pineau. End-to-End Text Recognition with Hybrid HMM Maxout Models. In *Proc. ICLR*, 2014.

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

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

[5] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, H. Chen, R. Vedantham, R. Grzeszczuk, and B. Girod. Residual enhanced visual vectors for on-device image matching. In *Asilomar*, 2011.

[6] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Proc. CVPR*, pages 2963–2970. IEEE, 2010.

[7] V. Goel, A. Mishra, K. Alahari, and C. V. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *ICDAR*, pages 398–402, 2013.

[8] 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. *arXiv:1312.6082*, 2013.

[9] A. Gordo. Supervised mid-level features for word image representation. *ArXiv e-prints*, Oct 2014.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[11] M. Jaderberg, A Vedaldi, and A. Zisserman. Deep features for text spotting. In *European Conference on Computer Vision*, 2014.

[12] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[13] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, S. R. Mestre, L. G. i Bigorda, J. Mas, D. F. Mota, J. Almazan, and L. P. de las Heras. ICDAR 2013 robust reading competition. In *ICDAR*, pages 1484–1493, 2013.

[14] 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.

[15] S. M. Lucas. ICDAR 2005 text locating competition results. In *Document Analysis and Recognition*, pages 80–84, 2005.

[16] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. In *ICDAR*, pages 682–687, 2003.

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

[18] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. ACCV*, pages 770–783. Springer, 2010.

[19] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Proc. CVPR*, 2012.

[20] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In *Proc. ICCV*, pages 97–104, December 2013.

[21] 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.

[22] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. CVPR*, 2007.

[23] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proc. CVPR*, 2010.

[24] J. A. Rodriguez-Serrano, F. Perronnin, and F. Meylan. Label embedding for text recognition. In *Proc. BMVC.*, 2013.

[25] A. Shahab, F. Shafait, and A. Dengel. ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *ICDAR*, pages 1491–1496, 2011.

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

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

[28] Wayne A Wickelgran. Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review*, 76(1):1, 1969.

[29] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4042–4049. IEEE, 2014.