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. We use purely data-driven, deep learning models to perform word recognition on the whole image at the same time, departing from the character based recognition systems of the past. These models 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 introduce three novel models, each one “reading” words in a complementary way: via large-scale dictionary encoding, character sequence encoding, and bag-of-N-gram encoding. In the scenarios of language/lexicon based and completely unconstrained text recognition we demonstrate 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 [28] or region grouping methods [5, 7, 20] 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.

We present three novel models that give state-of-the-art scene text recognition performance through the use of large, supervised convolutional neural networks (CNNs) [15]. These models have complementary representations of words, suitable for different purposes: large scale dictionary encoding (Sect. 3.1), character sequence encoding (Sect. 3.2), and bag-of-N-grams encoding (Sect. 3.3). Our recognition methods work by performing inference on the word image holistically, mimicking more closely how humans read [6], rather than by sequential character classification as is traditionally done. Due to the large training data requirements of these models, we also present a synthetic data generation method. This synthetic data (Fig. 1) is sufficiently realistic that we are
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 drawn in Sect. 5.

**Related work.** Traditional text recognition methods are based on sequential character classification by either sliding windows [28, 29] or connected components [19, 20], 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 [23] in Wang et al. [28], and CNNs in Wang & Wu [29]. Both [28] and [29] use a small fixed lexicon as a language model to constrain word recognition.

More recent works such as [1, 2, 21] make use of over-segmentation methods, guided by a supervised classifier to generate candidate proposals, which are subsequently classified as characters or false positives. Methods of generating candidate character regions include: oriented stroke detection [21], binarization and a sliding window classifier in the PhotoOCR system of [2] (which uses a dictionary of 100k words for re-ranking candidate words), and a CNN character classifier in Alsharif et al. [1] (which subsequently uses a 50k dictionary for accurate recognition). Words are recognised through a sequential beam search optimization over character candidates.

In contrast to these approaches based on character classification, the work by [8, 18, 22, 25] instead uses the notion of holistic word recognition. [18, 22] still rely on explicit character classifiers, but construct a graph to infer the word, pooling together the full word evidence. Rodriguez et al. [25] use aggregated Fisher Vectors [24] and a Structured SVM framework to create a joint word-image and text embedding. [8] use whole word-image features to recognize words by comparing to simple black-and-white font-renderings of lexicon words.

While not performing full text recognition, the system presented by Goodfellow et al. [9] recognises street numbers from loosely cropped images. They use a large CNN which outputs the sequence of numbers in the image almost directly. This is also applied to full text CAPTCHAs with great success, however, although randomly generated, CAPTCHAs are purely synthetic images. Their model is closely related to the character sequence encoding model in Sect. 3.2, with similarities discussed in that section.

In this work, we move further in the direction of holistic word classification for scene text, and build upon the success of advances in training large CNN architectures for object recognition [13, 26], as words can be viewed as a special class of objects. Our first model (Sect. 3.1) draws directly on this intuition, performing word classification in a standard CNN object recognition manner. The second model (Sect. 3.2), similar to [9], predicts directly the sequence of characters contained within the image. Our third method (Sect. 3.3) performs N-gram classification, creating a bag-of-N-grams embedding that can be easily used to infer the full word. All three models take the whole word image as input to perform recognition.

2 Synthetic Data Engine

This section describes our scene text rendering algorithm, used to generate data to learn the deep word classifiers. As our CNN models capture whole words instead of individual characters, it is
in fact 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 \[13,16,17,27]\, the Street View Text (SVT) dataset \[28]\, 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 \[2,9]\, but neither of these approaches are wholly accessible or scalable.

Here we follow the success of some synthetic character datasets \[4,29]\ 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. The key to our synthetic text generator is that all the components of the synthetic image are blended to resemble natural images, keeping the data distribution in line with that of the real world data.

Fig. 1 illustrates this 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 \(https://www.google.com/fonts\). 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 \[17]\ 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 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** – finally, Gaussian noise, blur, and JPEG compression artifacts are introduced to the image.

The synthetic data can be used in place of real world data and the labels generated from a corpus or dictionary as desired. By creating training datasets many orders of magnitudes larger than what has been seen before, we are able to use data-hungry deep learning algorithms and define new richer, word-based models.

3 Model Architectures

Text recognition is, at a fundamental level, the task of sequentially classifying each character in a text line. Character classification itself is not an easy task, with state-of-the-art systems achieving 92\% \[2]\ – compounding the error of each character classification of a text line means completely unconstrained recognition is generally unfeasible. Therefore, amongst existing methods there is a reliance on the assumption that all text is drawn from a language or fixed lexicon, allowing the use of lexical language models (e.g. an N-gram model \[3]\ or edit distance to a dictionary) which pull together character classifier results. However, global word inference is not performed at the visual recognition stage in previous work.

We address this modeling shortfall with three new models described in this section. All use the same framework of generating synthetic text data to train deep convolutional neural networks on whole word image samples, but with different objectives. The three architectures illustrate three different methods of reading. Sect. 3.1 describes a model performing pure word classification to a large dictionary, explicitly modeling 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 left-to-right 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.

All three models are trained purely on the synthetic data described in Sect. 2. The word samples
are generated with a fixed height of 32 pixels, but a variable width to preserve aspect ratio. Due
to the CNNs being constrained to a fixed width input, we simply linearly interpolate the width
of each image sample to 100 pixels, distorting the aspect ratio. We experimented with different
padding regimes to preserve input sample aspect ratio and potentially provide a word-length cue to
the network, but found that the results are not quite as good as when performing width resizing,
the results of which can be found in Sect. 4. Although aspect ratio is not preserved with width resizing,
the horizontal frequency distortion of image features when up-sampling or down-sampling most
likely provides any word-length cue needed.

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, 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.
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 \times 2 max-pooling follows the first, second and third
convolutional layers. Rectified linear units are used throughout as opposed to maxout \[10\] to reduce
the number of parameters required (as, for example, a layer with two maxout groups requires two
times as many parameters to achieve the same output dimensionality). 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 \( \mathcal{W} \) for a given input image \( x \) is given
by \( w = \arg \max_{w \in \mathcal{W}} P(w|x) \), where the per word output probability \( P(w|x) \) is modeled by softmax
scaling of the final fully connected layer. A schematic of the network is shown in Fig. 2(a).

\textbf{Training.} We train the network by back-propagating the standard multinomial logistic regression
loss with dropout \[11\], which improves generalization. Optimization uses stochastic gradient de-
scent (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 effec-
tively becomes large, slowing down training a lot. Therefore, for large dictionaries, we perform in-
cremental 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 be-
ing 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 dic-
tionary 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.

A word \( w \) of length \( N \) is modeled 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 a reasonable maximum number, which we set to 23, and introduce a null character class. Therefore a word is represented by a string $w = (C \cup \{\phi\})^{23}$. Then for a given image $x$, each character is predicted as $c^*_i = \arg\max_{c_i \in C \cup \{\phi\}} P(c_i | \Phi(x))$. $P(c_i | \Phi(x))$ 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.

This character sequence encoding model is a complete departure from the dictionary constrained model, as this allows entirely unconstrained recognition of words.

**Training.** 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 in spirit to the model used by Goodfellow et al. in [9]. Although the model of [9] 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. Our model achieves the same thing but without requiring any post processing – the word can be read directly from the CNN output by reading each character off the sequence classifier in order and stopping when a no-character class is predicted.

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 $s \in C$ and $w \in C^M$ 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 $G_N(w) = \{ s : s \subset w \land |s| \leq N \}$ the set of all grams of word $w$ of length up to $N$ and with $G_N = \cup_{w \in W} G_N(w)$ the set of all such grams in the language. For example, $G_3(\text{spires}) = \{ s, p, i, r, e, s, sp, pi, ir, es, sp, pir, ire, res \}$.

Even for small values of $N$, $G_N(w)$ encodes each word $w \in W$ 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 of our three text recognition models. Sect. 4.1 describes the benchmark data, Sect. 4.2 the implementation results, and Sect. 4.3 the results of our methods, that exceed state-of-the-art accuracy, significantly in some cases.

4.1 Datasets

A number of standard datasets are used for the evaluation of our systems – ICDAR 2003, ICDAR 2013, and Street View Text. ICDAR 2003 [17] 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 [1, 28, 29] 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 [28] and used in [1, 28, 29] 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 words as IC13. Street View Text [28] 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 [28] and used in [1, 2, 28, 29] are referred to as SVT-50.

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 (http://hunspell.sourceforge.net/), 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, Synth-50k, and 9 million images for the 90k word lexicon, Synth-90k.

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 $32 \times 100$ 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.
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 the IC03, SVT and IC13 datasets as well as our generated synthetic test dataset. The full set of results are shown in Table 1.

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.

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. Dictionary encoding 50-lexicon results are obtained by performing classification across the subset of classes, corresponding to the test image’s 50-word lexicon, rather than the full set of the dictionary classes. The Synth dataset performs very closely to that of the ICDAR 2003 dataset, confirming that the synthetic data is indeed 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 79.1% on SVT for full 90k-way classification, almost identical to a dictionary of 50k with the basic CNN architecture.

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 sequence of characters until a no-character classification
Table 2: Left: Comparison to previous methods. The ICDAR 2013 results given are case-insensitive. Bolded results outperform previous state-of-the-art methods. 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.

is encountered. 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 74.8% with the CHAR+2 model on ICDAR 2013 as opposed to 89.3% 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 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. The figure to the right of Table 2 shows the N-gram recognition results for the NGRAM+2 model, thresholded at 0.99 probability.

Comparison with the State of the Art & Discussion. Table 2 compares of 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 +6-10%, due to the fact our framework of synthetic data generation and classification allows explicit training on the lexicons, whereas with many previously published frameworks this is not possible.

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 [2], 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 [2].

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.
5 Conclusions

In this paper we have introduced a new framework for scalable, state-of-the-art word recognition – synthetic data generation followed by whole word input CNNs. We introduce three novel models using this framework, each with a unique method for recognising text, and demonstrate the vastly superior performance of these systems on standard datasets, with minimal tweaking or tricks. In addition we introduced a new synthetic word dataset, orders of magnitude larger than any released before. There are many extensions that are possible to these models taking further advantage of this plethora of data. For future work we hope to combine the three models of different reading styles together in a multi-task scenario, creating a unified recognition system.

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] O. Alsharif and J. Pineau. End-to-End Text Recognition with Hybrid HMM Maxout Models. In Proc. ICLR, 2014.
[2] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. PhotoOCR: Reading text in uncontrolled conditions. In Proc. ICCV, 2013.
[3] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In EMNLP-CoNLL, pages 858–867, 2007.
[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 Astilomar, 2011.
[6] M. Coltheart, K. Rastle, C. Perry, R. Langdon, and J. Ziegler. DRC: a dual route cascaded model of visual word recognition and reading aloud. Psychological review, 108(1):204–256, 2001.
[7] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In Proc. CVPR, pages 2963–2970. IEEE, 2010.
[8] 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.
[9] 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.
[10] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. arXiv:1302.4389, 2013.
[11] 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.
[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] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, pages 1106–1114, 2012.
[15] 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.
[16] S. M. Lucas. ICDAR 2005 text locating competition results. In Document Analysis and Recognition, pages 80–84, 2005.
[17] S. M. Lucas, L. Panaretos, L. Sosa, A. Tang, S. Young, and R. Young. ICDAR 2003 robust reading competitions. In ICDAR, pages 682–687, 2003.
[18] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. 2012.
[19] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In Proc. ACCV, pages 770–783. Springer, 2010.
[20] L. Neumann and J. Matas. Real-time scene text localization and recognition. In Proc. CVPR, 2012.
[21] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In Proc. ICCV, pages 97–104, December 2013.

[22] 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.

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

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

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

[26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In Proc. ICLR, 2014.

[27] 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.

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

[29] 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.