CNN-N-Gram for Handwriting Word Recognition

Arik Poznanski and Lior Wolf
The Blavatnik School of Computer Science
Tel Aviv University
arik.com@gmail.com, wolf@cs.tau.ac.il

Abstract

Given an image of a handwritten word, a CNN is employed to estimate its n-gram frequency profile, which is the set of n-grams contained in the word. Frequencies for unigrams, bigrams and trigrams are estimated for the entire word and for parts of it. Canonical Correlation Analysis is then used to match the estimated profile to the true profiles of all words in a large dictionary. The CNN that is used employs several novelties such as the use of multiple fully connected branches. Applied to all commonly used handwriting recognition benchmarks, our method outperforms, by a very large margin, all existing methods.

1. Introduction

The most prominent method in the current application of deep learning to computer vision is the Convolutional Neural Network (CNN) [30]. Developed initially for the task of reading handwritten digits, this method is now utilized for almost every perceptual task relating to image and video data. In many such tasks, CNNs lead to state of the art performance. It is, therefore, somewhat surprising that in the field of handwriting recognition, CNNs are currently not leading the performance charts.

CNNs are trained in a supervised way. The first question when training it, is what type of supervision to use. Previous applications of CNNs in the field of printed word recognition [25] have demonstrated that a word based encoding is preferable to encoding using bag-of-n-grams. The results presented in this work show that, at least for handwriting recognition, a very effective system can be built using an attributes based encoding, in which the input image is described as having or lacking a set of n-grams in some spatial sections of the word.

The attributes used in our work are heavily based on the earlier work of Almazán et al. [3]. These binary attributes check whether the word contains a specific n-gram in some part of the word. For example, one such property may be: *Does the word contain the bigram "ou" in the second half of the word?* If the examined word is "ingenious" then the answer is positive, whereas if the checked word is "outstanding" the answer is negative.

While the original work of [3] uses SVMs over Fisher Vectors of SIFTs, we employ CNNs directly over raw pixel values. Moreover, in order to reach high levels of performance, multiple novelties had to be incorporated. These include the creation of specialized sub-networks that focus on subsets of the set of 600-1200 attributes used (this number depends on the size of the alphabet).

The obtained network is fairly large, and training it on the relatively small datasets available for handwriting recognition is challenging. We incorporate gradual training and other ideas in order to tackle this.

The resulting method is extremely potent. The same architecture applied to all significant handwriting benchmarks that we are aware of, achieves a very sizable improvement over state of the art. While the network was not designed for printed text, it also achieves state of the art results on the Street View Text (SVT) benchmark [54] and results comparable to state of the art on IIIT5K [37] benchmark.

2. Related work

Currently, the leading handwriting recognition benchmarks are IAM, RIMES and IFN/ENIT. The performance charts for these datasets are currently dominated by the use of Recurrent Neural Networks (RNNs) and its extensions such as Long-Short-Term-Memory (LSTM) networks, Hidden Markov Models (HMMs), and various combinations of these methods. CNNs are absent from current results.

The IAM dataset [33] contains images of handwritten English words. The current state of the art on the IAM benchmark is a system by Bluche et al. [10] which uses the ROVER voting scheme [15] for combining four models. Two of them are based on Bidirectional Long Short-Term Memory (BDLSTM) RNNs, and the other two are based on deep Multi-Layer Perceptrons (MLPs).

A close second to the state of the art method on IAM is a system by Doetsch et al. [11], which uses an LSTM-RNN with an additional parameter that controls the shape of the

squashing functions in the gating units.

The RIMES dataset [5] is composed of images of handwritten French words. The current state of the art on RIMES is a system by Menasri et al. [34] that uses a single RNN. To further improve their state of the art results, they also present a system that uses an optimized combination of seven recognizers, including one grapheme based MLP-HMM, two variants of sliding window based GMM-HMMs, and four variants of their proposed RNNs.

IFN/ENIT [41] is a dataset of Arabic handwritten word images. This dataset has four different test scenarios, named "abc-d", "abcd-e", "abcde-f" and "abcde-s". The current state of the art method depends on the test scenario. In scenario "abcde-f", the current state of the art is a system by Graves and Schmidhuber [13], which uses a hierarchy of multidimensional LSTM-RNNs. In all other IFN/ENIT scenarios, the leading method is the one by Stahlberg and Vogel [49], which uses fully connected deep neural networks for optical modeling with features extracted from raw pixel gray-scale intensity values of foreground segments. A comparable method, with regards to overall performance, is an HMM system by Azeem and Ahmed [6].

The work by Almazán et al. [3] is closely related to our work. Their method encodes the input word image as Fisher Vectors (FV) [43], i.e., as an aggregation of the gradients of a Gaussian Mixture Model (GMM) over some low-level descriptors, SIFT in this case. It then trains a set of linear SVM classifiers, one per each binary attribute contained in a set of word properties. Canonical Correlation Analysis (CCA) is used to link the vector of predicted attributes and the binary attributes vector generated from the actual word. The CCA method finds a common vector subspace where the predicted attributes vector and binary attributes vector are naturally comparable. To find the transcription, a simple nearest neighbor search is made in the transformed lexicon space, which was projected to the common subspace.

Almazán et al.'s method is currently not among the state of the art methods in handwriting recognition, on any of the datasets. In our work, using almost the same binary attributes as in [3], we replace the FV based classifiers with a specific type of CNN. Unlike [3], we train over raw pixel values and, in addition, we benefit from using a single classifier that predicts all the binary attributes, instead of using one classifier per attribute. Instead of relying on the predictions, we apply CCA to the representation vector obtained from one layer below the prediction layers. Using our method, we obtain a sizable improvement over all commonly used handwriting recognition benchmarks, halving, in almost all cases, the best reported error rate.

Attributes based methods are now commonplace throughout computer vision. As an example, Movshovitz-Attias et al. [38] predict a category from a given hierarchical set of categories. Instead of learning to predict the category directly, they have flattened the hierarchy and learned to predict all the categories to which the object belongs. This conversion from single label to multiple binary attributes enables them to better exploit the data since every low level category is also an example of its ancestors categories. This is similar in spirit to the attributes based model we employ.

Another recent use of attributes prediction using CNNs is the work by Zhang et al. [57]. In their work they propose a method for inferring human attributes such as gender, hair style, cloth style, and more from images of people under large variation of viewpoint, pose, appearance, articulation and occlusion. They propose a method which combines part-based models and deep learning by training a pose-normalized CNNs. They use a network which shares the layers including the first fully connected layer, and only then they branch out to a fully connected layer per attribute. In our network we only share the conventional layers, and use a three layers deep fully connected layers per each group of attributes.

Another related work is the work by Jaderberg et al. [25], which uses CNNs trained on synthetic data for Scene Text Recognition. Although they get state of the art results on the SVT [54] dataset, their method is not evaluated on handwriting recognition since, unlike for scene text, synthetic fonts lack the full variability which handwritten text poses. In their work, three different CNNs are presented, one of which is trained on words encoded as bag-of-n-grams. That n-gram based CNN achieves inferior results, compared to their other CNNs, on both SVT and SVT-50. Our n-gram based method, when applied outside our main scope of handwriting recognition, achieves results better than their best network on SVT. This is done, when training on the same synthetic dataset used for training the system of [25].

We identify four differences that we believe make our method work better than Jaderberg at al.'s. The first is our use of CCA to factor out dependencies between attributes. The second difference is that we take into consideration the spatial location of the n-gram inside the word. The third difference is our network structure, which is deeper and uses multiple parallel fully connected layers, each handling a different set of attributes. Lastly, our method uses considerably less n-grams than the list of 10,000 n-grams they have used.

The architecture we propose employs multiple fully connected "arms" diverging from the CNN body. This is done in order to create spatial and type of n-gram specialization. Spanning multiple branches from a neural network in order to support more effective learning is done in [50], where intermediate networks were added in order to support the training of a very deep network. Unlike our usage, these sub-networks were not used during test time.

The success of the training process relies on gradual training of the network, adding one such arm at each training phase. Other forms of gradual learning have been pro-

posed in the literature. For example, the work of [25] mentioned above, learns the lexicon gradually.

3. Method

In offline handwriting recognition, we are given two disjoint sets, which we name train and test. Each of the sets contains pairs (I, t) such that I is an image and t is its textual transcription. The goal is to build a system which, given an image, produces a prediction of the image transcription. The construction of the system is done using information from the train set only.

In order to evaluate the method's performance, we apply it to the test images and for each image compare the predicted transcription with the actual image transcription. The result of such an evaluation can be reported by one of several related measures. These include Word Error Rate (WER), Character Error Rate (CER), and Accuracy (1-WER). WER is the ratio of the reading mistakes, at the word level, among all test words. CER measures the Levenshtein distance normalized by the length of the true word.

3.1. From a text word to attributes vector

Our method leverages the inherent structure of a textual label by considering common attributes that are shared between different words. The attributes that we use were named Pyramidal Histogram of Characters (PHOC) in [3].

The simplest attributes are based on unigrams and pertain to the entire word. An example of a binary attribute in English is "does the word contain the unigram 'A'?" There are as many such attributes as the size of the character set of the benchmark that we tackle. The character set may contain lower and upper case Latin alphabet, digits, accented letters (e.g. é, è, ê, ë), Arabic alphabet, etc. We call the attributes that check whether a word contains a specific unigram, Level-1 unigram attributes.

We define Level-2 unigram attributes as attributes that observe whether a word contains a specific unigram in the first or second half of the word, e.g., - "does the word contain the unigram 'A' in the first half of the word?". For example, the word "BABY" contains the letter 'A' in the first half of the word ("BA"), but doesn't contain the letter 'A' in the second half of the word ("BY").

To address words with an odd number of characters and other fractional cases, we need to properly define the decision rule used to determine whether a letter appears in a specific section of the word. The decision is made purely using the word text, without any image involved. The process does not require any information about the location of the letters in the image.

The decision rule is that a word's letter is part of a specific section of the word if that section contains at least 50% of the letter location. For example, in the word "KID", the first half of the word contains 1.5 letters, including the letter 'K' and 50% of the letter 'I'. Therefore, according to our definition, the first half of the word "KID" contains the letters 'K' and 'I'. Similarly, The second half of the word "KID" contains the letter 'D' in full, and exactly 50% of the letter 'I'. Therefore, the second half of the word "KID" contains both 'D' and 'I'. While it may seem as if we are assuming that every letter has an equal width, we do not explicitly cut the images. As long as we are consistent with our definition of the attributes, the supervision given to the network is consistent.

Attributes group	Relevant part of word	Positive attributes
Level 1 Unigrams	optimization	$\{a,i,m,n,o,p,t,z\}$
Level 2 Unigrams	optimi	$\{i,m,o,p,t\}$
	zation	$\{a,i,n,o,t,z\}$
Level 3 Unigrams	opti	$\{i,o,p,t\}$
	miza	$\{a,i,m,z\}$
	tion	$\{i,n,o,t\}$
Level 4 Unigrams	opt	$\{o,p\}$
	imi	$\{i,m,t\}$
	zat	$\{a,t,z\}$
	ion	$\{i,n,o\}$
Level 5 Unigrams	op	$\{o,p\}$
	tim	$\{i,m,t\}$
	iz	$\{i,z\}$
	ati	$\{a,i,t\}$
	on	$\{n,o\}$
Level 2 Bigrams	optimi	$\{ti\}$
	zation	$\{at,ti,io,on\}$
Level 2 Trigrams	optimi	$\emptyset$
	zation	$\{ion,tio,ati\}$

Figure 1. An example of the attributes which are set for the word "optimization". Since we use only common bigrams and trigrams, not every bigram and trigram is an attribute.

Similarly, Level-3 unigram attributes are also defined, breaking the word into three equal parts. In addition, Level-2 bigram attributes are defined as binary attributes that indicate whether the word contains a specific bigram. We also include extensions to trigrams. Fig. 1 demonstrates the attributes associated with a word.

Other attributes are possible, but we did not find these to be helpful. For example, we tried attributes pertaining to the first letters or to the end of the word, e.g., "does the word end with an 'ing'?"

Ideally, enough binary features would be defined such that every word in our lexicon has a unique attributes vec-

tor. We will later use this bijective mapping to map a given attributes vector to its respective generating word.

The set of attributes used throughout our experiments, unless otherwise noted, contains the unigram attributes based on the entire list of each benchmark's character set, inspected in levels 1 to 5; the 50 most common bigrams in level 2, and the 20 most common trigrams in level 2. Given that the character set of a given benchmark contains k symbols, the total number of binary attributes would be: $k(1+2+3+4+5)+50\times 2+20\times 2 = 15k+140$.

3.2. Learning attributes vectors for images

While the transformation from words to attributes is technical, the transformation from an image to an estimated vector of attributes is learned, from examples, using a CNN. One of the strengths of CNNs, compared to the per-attribute classifiers used by Almazán et al. [3], is the sharing of intermediate computations. Many of the attributes are similar in nature. For example, "does the word contain the unigram 'A' in its first half?" and "does the word contain the unigram 'A' in the second half of the word?" are similar classification problems; "does the word contain the bigram 'AB'?" is also related to both. A shared set of filters can tackle these problems successfully, and the CNN benefits from solving multiple classification problems at once.

Compared to the approach of [25], which enjoyed a practically unlimited training set, handwriting recognition is based on smaller datasets. The advantage of attributes in such cases is that the training set is utilized much more efficiently. For example, consider the case of a training set of size 1,000. The word "SLEEP" may appear only twice, but attributes such as "does the word contain the unigram 'S' in the first half of the word?" will probably have many more instances. Therefore, learning to detect the attribute will be easier to train than learning to detect the word. Since CNNs benefit substantially from a larger training set, the advantage of the attributes based method for handwriting recognition is significant.

Another advantage of learning attributes rather than the words themselves, is that similar words may confuse the network. For example, consider the words "KIDS" and "BIDS". A "KIDS" word image is a negative sample for the "BIDS" category, although a large part of their appearance is shared. This similarity between some categories makes a category based classifier harder to learn, whereas an attributes based classifier uses this for its advantage.

3.3. Network architecture

The basic layout of our CNN is a VGG [47] style network consisting of small (3×3) convolution filters. Starting with a small 100×32 input image, a relatively deep network structure of 12 layers is being used.

The employed CNN has nine convolutional layers and three fully connected layers. In forward order, the convolutional layers have 64, 64, 64, 128, 128, 256, 256, 512 and 512 filters of size 3×3. Convolutions are performed with a stride of 1 and there is input feature map padding by 1 pixel to preserve the spatial dimension. The layout of the fully connected layers is not conventional, and is detailed below.

We use maxout [18] activation for each layer, including, what seems to be less conventional, the convolutional layers. We apply batch normalization [23] after each convolution, and before each maxout. The network also contains 2×2 max-pooling layers, with a stride of 2, following the 3rd, 5th and 7th convolutional layers.

An important design novelty that we introduce is the usage of multiple separate and parallel fully connected layers. Each layer leads to a separate group of attributes predictions. We divide the attributes to groups of unigrams, bigrams, or trigrams, and for levels and spatial locations. For example, one group of attributes contains only level 2, 2nd word-half, bigram attributes.

Thus, instead of using a single fully connected layer, as is customary, to generate the entire attributes vector, we employ one fully connected layer per each group of attributes. In our configuration, we have a total of 19 groups of attributes, $1+2+3+4+5$ for unigram based attributes at levels one to five, 2 for bigram based attributes at level two, and 2 for trigram based attributes at the same level.

The layers leading up to this set of fully connected layers are all convolutional and are shared. The motivation for this network structure is that the convolutional layers learn to recognize the letters' appearance, regardless of their position in the word, and the fully connected layers learn the spatial information, i.e., the approximate position of the n-gram in the word. Hence, splitting the one fully connected layer into several parts, one per spatial section, allows the fully connected layers to specialize, leading to an improvement in accuracy. This is verified in Sec. 5, where we demonstrate that there is a clear advantage for using multiple, separated, fully connected layers.

Fig. 2 illustrates the structure of the network. The output of the last convolutional layer is fed into 19 branches. Each such branch contains three fully connected layers. These layers have 128 units, 2048 units, and as many units as the relevant group of binary attributes. In our configuration, for unigram based groups that number is equal to the size of the character set: 52 for IAM, 78 for RIMES, 44 for IFN/ENIT, and 36 for SVT and IIIT5K. For bigram based groups the size is 50, and for trigram based groups the size is 20. The activations of the last layer are transformed into probabilities using a sigmoid function.

3.4. Training and implementation details

The network is trained using the aggregated sigmoid cross-entropy (logistic) loss. Stochastic Gradient Descent

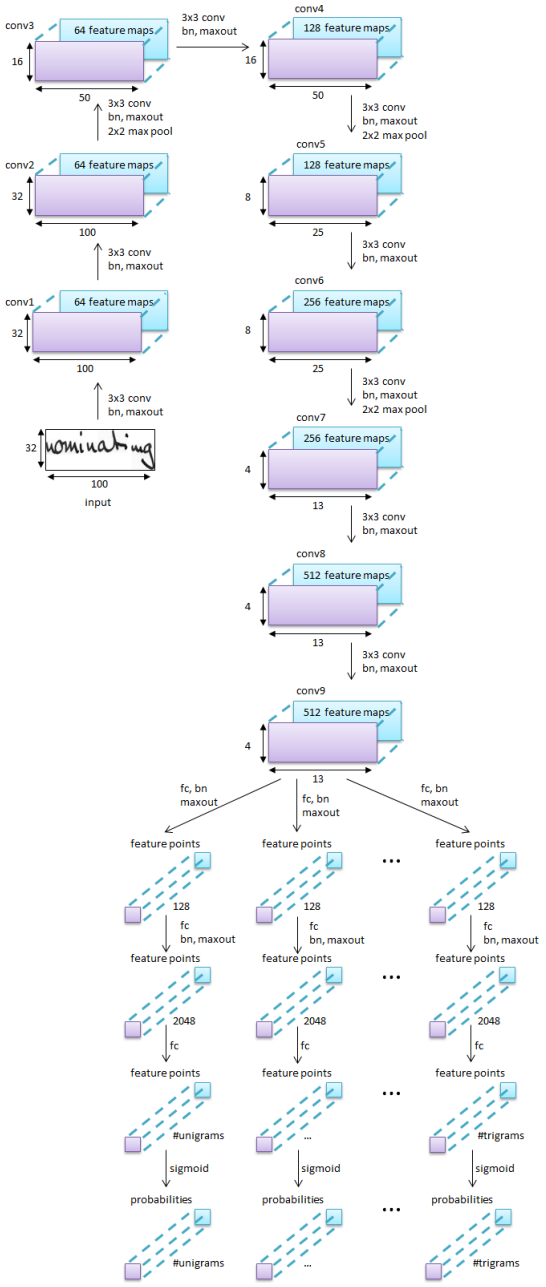

Figure 2. The network architecture used. Nine shared convolutional layers, with intermediate batch normalization layers, maxout activations, and max pooling layers are employed, in sequence, to an input of size 100×32 pixels. The network then splits into 19 parallel branches. In each branch, there are two hidden layers and an output layer that is converted to probabilities.

(SGD) is employed as the optimization method, with a momentum set to 0.9, and dropout [48] after the two fully connected hidden layers with a parameter set to 0.5.

An initial learning rate of 0.01 is used, and is lowered, as is customary, when the validation set performance stops improving. Each time the learning rate is divided by 10 with this process repeated three times. The batch size is set in the range of 10 to 100, depending on the dataset that we train on and the memory load. When enlarging the network and adding more FC layers, the GPU memory becomes congested and the batch size is lowered. The network is initialized using Glurot and Bengio's [16] initialization scheme.

Training is done in stages, by gradually adding more attributes groups as training progresses. We start by training on a single group of attributes, the Level-1-Unigrams, using a single fully connected arm. When the loss stabilizes, another group of attributes is added and the training continues. Groups of attributes are being added in the following order: Level-1-Unigrams, Level-2-Unigrams, ..., Level-5-Unigrams, Level-2-Bigrams, and Level-2-Trigrams. When adding groups, the initial learning rate is used. Only when all 19 groups have been added, do we begin to lower the learning rate. We have found this gradual way of training to generate considerably superior results over the alternative of directly training on all the attributes groups at once.

In the SVT and IIIT5K experiments, when training the network on the huge synthetic dataset of [25], training was initialized on a partial subset of the training set. In detail, we train the network using 10k images out of the 7M images until partial convergence, after which we continue training using 100k images until partial convergence. We repeat this process with 200k and 1M, and finally we train on the entire train set. When trying to train directly on the entire train set the network did not converge. Our solution is similar to the incremental training used by [25] where a multiclass classifier was trained on 90k different words. They first train on 5k words, and once converged, increased the training set to 10k words. This was repeated until all classes were covered.

We used a custom version of Caffe framework [27] to implement our network.

3.5. Regularization and training data augmentation

In order to avoid overfitting, dropout, as mentioned, is applied after the first and second fully connected layers of each branch. In addition, a weight decay of 0.0025 is applied to learned weights.

The inputs to the network are grayscale 100×32 images. Where needed, we stretch the input image to this size, without preserving the aspect ratio. Since the handwriting datasets are rather small and we are training a deep neural network with tens of millions of parameters, data augmentation is warranted.

The data augmentation is performed as follows. For each input image, we apply rotation around the images center with each of the following angles (degrees): $\{-5, -3, -1, +1, +3, +5\}$. In addition, we apply shear using the following angles $\{-0.5, -0.3, -0.1, 0.1, 0.3, 0.5\}$. This way, 36 additional images are generated per each input

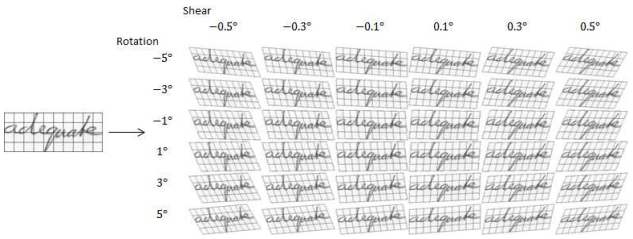

Figure 3. The data augmentation process. Each input image is transformed using both rotation and shear to generate additional similar, but different, samples for a given label.

image, dramatically increasing the amount of training data. This image augmentation process is described in Fig. 3.

This data augmentation technique employed is rather basic. We conjecture that the system can benefit from an even more sophisticated data augmentation technique. For example, elastic distortion [46] can be used.

Note that our method augments the test data as well. This is described below in Sec. 4.

4. Recognition

Each word in our lexicon is represented by its ground truth attributes vector. This is an efficient process that is only done once.

Given an image, it is run through the network. One can then directly compare the set of predicted attributes to the attributes of the lexicon words. However, this is suboptimal for several related reasons. First, the network was not trained for the purpose of matching lexical words. Rather, it was trained for per feature success. Second, such a method ignores the correlations that exist between the various coordinates due to the nature of the attributes. For example, a word which contains the letter 'A' in the first third of the word will always contain the letter 'A' in the first half of the word. Third, the relative importance and discriminative power of attributes or subsets of attributes is not taken into account in a direct comparison. In addition to these reasons, a direct comparison also requires a careful calibration of the network's output probabilities.

To solve these issues, we use Canonical Correlation Analysis (CCA) in order to learn a common linear subspace to which both the binary attributes of the lexicon words and the network representations are projected. This shared subspace is learned such that images and matching words would be projected as close as possible.

For added accuracy, the regularized CCA method [52] is used. The regularization parameter is fixed to be the largest eigenvalue of the cross correlation matrix between the image representations and the matching attributes vectors.

Since CCA does not require that the matching input vectors of the two domains are of the same type or the same size, we are able to use "neural codes" in lieu of the attributes probability estimations provided by the network. The neural codes are the activations of the network in the layer just below the classification, i.e., in our case, the concatenation of the second fully connected layers from all branches of the network.

When doing so, we obtain a rather long representation. There are 19 groups of classifiers and each of them has $2,048$ units in the second locally connected layer (post maxout). The total representation size is, therefore, $38,912$. In order to avoid memory explosions, we subsample $12,000$ vector elements out of the $38,912$ and use the subsampled vectors as input to the CCA. We observe a very small change when resampling the subset: the change is less than 0.1 percent in all experiments we performed.

Similar to Almazán et al. [3] work, and other applications of CCA, we L2-normalize the input of the CCA algorithm, and use the cosine distance in order to efficiently find the nearest neighbor in the shared space.

Test side data augmentation In order to further improve results, test-side data augmentation is being employed. Given a test image, we generate 36 additional variants using the same method described in Sec. 3.5. The 37 images are all encoded using the network. The final representation is taken to be the mean vector of all 37 representations.

5. Experiments and results

We present results on the commonly used handwriting recognition benchmarks. The datasets used are: IAM, RIMES and IFN/ENIT, which contain images of handwritten English, French and Arabic, respectively. The same network was run in all cases, using the same parameters. Hence, no language specific information was needed except for the character set of the benchmark. In addition, while the network was not developed with printed text in mind, we report results on two scene text recognition benchmarks: SVT and IIIT5K.

The IAM Handwriting Database [33] is a known offline handwriting recognition database of mostly cursive English word images. The database contains 115,320 words written by 500 authors. The database comes with a standard split into train, validation and test sets, such that every author contributes to only one set. In other words, it is not possible that the same author would contribute handwriting samples to both the train set and the test set.

RIMES [5] contains more than 60,000 words written in French by over 1000 authors. The RIMES database has several versions with each one a super-set of the previous one. In our experiments, we employ the latest version presented in an ICDAR 2011 contest.

IFN/ENIT [41] is a cursive Arabic handwriting benchmark which contains several sets and has several scenarios that can be tested and compared to other works. The

Database	IAM		RIMES	
Model	WER	CER	WER	CER
Bertolami and Bunke [7]	32.83	-	-	-
Dreuw et al. [12]	28.80	10.10	-	-
Boquera et al. [14]	15.50	6.90	-	-
Telecom ParisTech [21]	-	-	24.88	-
IRISA [21]	-	-	21.41	-
Jouve [21]	-	-	12.53	-
Kozielski et al. [28]	13.30	5.10	13.70	4.60
Almazan et al. [3]	20.01	11.27	-	-
Messina and Kermorvant [36]	19.40	-	13.30	-
Pham et al. [44]	13.60	5.10	12.30	3.30
Bluche et al. [9]	20.50	-	9.2	-
Doetsch et al. [11]	12.20	4.70	12.90	4.30
Bluche et al. [10]	11.90	4.90	11.80	3.70
Menasri et al. (single) [34]	-	-	8.90	-
Menasri et al. (7 combined) [34]	-	-	4.75	-
This work	**6.45**	**3.44**	**3.90**	**1.90**

Table 1. Comparison to previous methods on IAM and RIMES (ICDAR2011) datasets. Our results are based on the full CNN-n-gram system. Slightly better results appear in table 4. all numbers in percent.

Database	IFN/ENIT			
Scenario	abc-d	abcd-e	abcde-f	abcde-s
Pechwitz & Maergner [42]	89.74	-	-	-
Alabodi & Li [2]	93.30	-	-	-
Lawgali et al. [29]	-	90.73	-	-
SIEMENS [40]	-	-	82.22	73.94
Dreuw et al. [12]	96.50	92.70	90.90	81.10
Graves&Schmidhuber [20]	-	-	91.43	78.83
UPV PRHLT [32]	-	-	92.20	84.62
Graves&Schmidhuber [13]	-	-	93.37	81.06
RWTH-OCR [31]	-	-	92.20	84.55
Azeem & Ahmed [6]	97.70	93.44	93.10	84.80
Ahmad et al. [1]	97.22	93.52	92.15	85.12
Stahlberg & Vogel [49]	97.60	93.90	93.20	88.50
This work	**99.29**	**97.07**	**96.76**	**94.09**

Table 2. Comparison to previous methods on IFN/ENIT dataset. Shown is accuracy in percent. Our results are based on the full CNN-n-gram system. Slightly better results appear in table 4.

most common scenarios are: "abcde-f", "abcde-s", "abcd-e" (older) and "abc-d" (oldest). The naming convention specifies the train and the test sets. For example, the "abcde-f" scenario refers to a train set comprised of the sets a, b, c, d, and e; testing in this scenario is done on set f.

In addition, while the network was not developed with printed text in mind, we also report results on SVT [54] and IIIT5K [37] datasets. Two benchmarks exist for SVT: using a general lexicon and using the SVT-50 subset, in which the task is to recognize among 50 words. Similarly, IIIT5K contains two benchmarks: IIIT5K-50 and IIIT5K-1000.

Protocol In Offline Handwriting Recognition, the goal is to find the transcription given a test image. The transcription is limited to a lexicon associated with the tested dataset. On IAM and RIMES, we use the lexicon of all the dataset

Database	SVT		IIIT5K	
Scenario	50	Full	50	1000
ABBYY [35] [53]	35.0	-	24.30	-
Wang et al. [53]	57.0	-	-	-
Mishra et al. [37]	73.57	-	64.10	57.50
Novikova et al. [39]	72.9	-	-	-
Wang et al. [55]	70.0	-	-	-
Goel et al. [17]	77.28	-	-	-
PhotoOCR [8]	90.39	77.98	-	-
Alsharif and Pineau [4]	74.3	-	-	-
Almazán et al. [3]	89.18	-	91.20	82.10
Yao et al. [56]	75.89	-	80.20	69.30
Jaderberg et al. [26]	86.1	-	-	-
Gordo [19]	91.81	-	93.30	86.60
Jaderberg et al. [24]	95.4	80.7	97.1	92.7
Baoguang et al. [45]	96.40	80.8	97.6	**94.4**
This work	**96.60**	**83.62**	**97.93**	94.15
unigrams, probs CCA, no test	95.67	81.76	96.38	91.82
unigrams, probs CCA, with test	**96.60**	81.61	97.04	93.10
unigrams, FC CCA, no test	95.05	**83.62**	97.23	93.41
unigrams, FC CCA, with test	95.36	81.92	97.62	94.15
trigrams, probs CCA, no test	94.59	81.76	96.41	92.24
trigrams, probs CCA, with test	95.83	81.92	97.16	93.26
trigrams, FC CCA, no test	94.28	81.61	97.43	93.30
trigrams, FC CCA, with test	94.90	80.99	**97.93**	93.80

Table 3. Comparison to previous work on SVT and IIIT5K datasets. Shown is accuracy in percents. The term "unigrams" refers to training on unigrams only, without bigrams and trigrams. The term "trigrams" refers to training on unigrams, bigrams and trigrams. The term "probs CCA" means the input to CCA was the final probabilities layer, whereas the term "FC CCA" means the input is the last fully connected layer. The term "no/with test" refers to whether test-side data augmentation was employed.

words, both train and test sets, as done in all the lexicon based methods in the literature, e.g. [3, 9, 14]. On IFN/ENIT, we use the official lexicon attached to the benchmark. On SVT, we use the general purpose, 90k words lexicon used in [25, 24], whereas for SVT-50 we use the 50 words lexicon associated with this reduced benchmark. On IIIT5K-50 and IIIT5K-1000 we use the 50 words lexicon and 1000 words lexicon respectively associated with the benchmarks.

The method's prediction is compared with the actual image transcription. The different benchmarks use several different measures as detailed in Sec. 3. To ease comparison to other methods, we report using the same measure commonly used in the respected benchmarks. Specifically, on IAM and RIMES, we show our results using WER and CER measures, whereas on IFN/ENIT, SVT and IIIT5K, the results are shown using the accuracy measure.

Since the benchmarks are in different languages, different character sets were used. Specifically, for IAM, the character set contains the lower and upper case Latin alphabet. Digits were not included since they are rarely used in this dataset. However, when they appear we did not ig-

Database	IAM		RIMES		abcde-s
Model	WER	CER	WER	CER	WER
Full CNN-n-gram system	6.45	3.44	3.90	1.90	5.91
v1: CCA on probabilities	6.56	3.46	3.85	**1.73**	6.42
v2: no trigrams during test	6.33	3.34	3.95	1.86	6.10
v3: no bi- and tri-grams during test	**6.29**	3.37	**3.78**	1.89	6.10
v4: no trigrams during train	6.32	**3.33**	4.15	1.91	5.91
v5: no bi- and tri-grams during train	6.36	3.36	3.85	1.82	**5.85**
v6: 7 FCs instead of 19 FCs	7.16	3.95	4.93	2.34	6.48
v7: 1 FC instead of 19 FCs	7.81	4.33	4.93	2.31	7.63
v8: no test-side augmentation	6.94	3.71	4.27	2.02	6.42
v9: without using CCA	8.83	5.93	5.17	3.43	6.68

Table 4. Comparison to different variants of the CCN-n-gram system. The full system contains 19 FCs, with bigrams and trigrams, with test-side data augmentation, CCA gets the FC layer as input. For reasons of table consistency, IFN/ENIT (abcde-s) results are given in terms of WER instead of Accuracy (1-WER).

nore them. Therefore, if a prediction is different from the ground truth label only by a digit, it is still considered a mistake. In RIMES, the character set used contains the lower and upper case Latin alphabet, digits and accented letters. In IFN/ENIT, the character set was built out of the set of all unigrams in the dataset. This includes the Arabic alphabet, digits and symbols. In SVT the character set used contains the Latin alphabet, disregarding case, and digits.

The network used for SVT and IIIT5K was slightly different from the networks used for handwriting recognition. Since the synthetic dataset used to train for the SVT and IIIT5K benchmarks has many training images, we doubled the number of neurons in the first fully connected layer.

Comparison with previous work We compare to the state of the art on IAM and RIMES in Table 1 and on IFN/ENIT in Table 2. Our method achieves state of the art results on all benchmarks, including all versions of the IFN/ENIT benchmark. The improvement over the state of the art, in these competitive datasets, is such that the error rates are cut in half throughout the datasets: IAM (6.45% vs. 11.9%), RIMES (3.9% vs. 8.9% for a single recognizer), IFN/ENIT set-f (3.24% vs. 6.63%) and set-s (5.91% vs. 11.5%),

In Table 3, our results are compared to other state of the art methods on the popular SVT and IIIT5K datasets. As can be seen, we achieve state of the art results when using the same global 90k dictionary used in [25]. State of the art results are also achieved on the more common SVT-50 variant. In addition, we get state of the art results on IIIT5K-50 and results comparable to state of the art on IIIT5K-1000. Interestingly, we have also compared the accuracy on the test set of the synthetic data: we obtain 96.88% compared to 95.2% obtained by the best network of [25].

In order to study the influence of each of the method's elements to its success, we have included several variants in Table 4. Somewhat reassuringly, the method is robust to various design choices. For example, using CCA on the aggregated probability vectors provide a compatible level of performance. Similarly, bigrams and trigrams do not seem to consistently affect performance. We tried removing these only from the test stage, or from both train and test stages, without any clear indications.

What does seem to matter is the separation of the fully connected layers into multiple branches. Reducing the number of branches from 19 to 7 by merging related attributes groups (e.g., using a single branch for all level 5 unigram attributes instead of 5 branches), or to one branch of fully connected hidden layers, hurts performance. Increasing the number of hidden units in order to make the total number of hidden units the same (not shown in the table), hinders convergence during training. Lastly, test-side data augmentation seems to consistently improve performance.

For completeness we also present the results of using the full system without CCA. There are several ways to obtain results without CCA. Out of a few such alternatives, the best results seem to be obtained by thresholding the attribute scores and then using the Jaccard similarity between the binary prediction and binary ground truth. As expected, removing CCA hurts performance.

6. Discussion

Handwriting recognition is the birthplace of deep learning. CNNs were created in order to read handwritten postal codes, and RNNs success in multiple handwriting benchmarks preceded, by a few years, the success of CNNs on imagenet. It is therefore somewhat disappointing that the recent wave of leaps in performance achieved using deep learning throughout computer vision has so far skipped handwriting recognition.

We believe that there is nothing inherently challenging in handwriting recognition that prevents machines from achieving a human level of performance in the same way that such performance is achieved in face recognition [51] and even in some object recognition metrics [22].

The success of our method suggests that a successful handwriting recognition system can be built without investing much effort in atomic tasks such as image binarization and letter segmentation. This follows the end-to-end trend of deep learning. It could be the case that future systems would benefit from multiple sub-systems. Similarly, we employ only one network and do not use ensembles or voting schemes in order to obtain a slight improvement in performance. However, future systems might benefit from employing multiple systems in order to utilize multiple tactics depending on the nature of the script.

Looking forward, the ultimate goal of handwriting recognition is the construction of a universal reading system, which is adaptable not only for different scribes (as is done here) but also for different script types. In such a system, learning to read the next script would be easier than training from scratch.

References

[1] I. Ahmad, G. Fink, S. Mahmoud, et al. Improvements in sub-character hmm model based arabic text recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 537–542. IEEE, 2014.

[2] J. Alabodi and X. Li. An effective approach to offline arabic handwriting recognition. *International Journal of Artificial Intelligence & Applications*, 4(6):1, 2013.

[3] J. Almazan, A. Gordo, A. Fornes, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2552–2566, 2014.

[4] O. Alsharif and J. Pineau. End-to-end text recognition with hybrid hmm maxout models. *arXiv preprint arXiv:1310.1811*, 2013.

[5] E. Augustin, M. Carré, E. Grosicki, J.-M. Brodin, E. Geoffrois, and F. Prêteux. Rimes evaluation campaign for handwritten mail processing. In *Proceedings of the Workshop on Frontiers in Handwriting Recognition*, number 1, 2006.

[6] S. A. Azeem and H. Ahmed. Effective technique for the recognition of offline arabic handwritten words using hidden markov models. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(4):399–412, 2013.

[7] R. Bertolami and H. Bunke. Hidden markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognition*, 41(11):3452–3460, 2008.

[8] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 785–792. IEEE, 2013.

[9] T. Bluche, H. Ney, and C. Kermorvant. Tandem hmm with convolutional neural network for handwritten word recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2390–2394. IEEE, 2013.

[10] T. Bluche, H. Ney, and C. Kermorvant. A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In *Statistical Language and Speech Processing*, pages 199–210. Springer, 2014.

[11] P. Doetsch, M. Kozielski, and H. Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 279–284. IEEE, 2014.

[12] P. Dreuw, P. Doetsch, C. Plahl, and H. Ney. Hierarchical hybrid mlp/hmm or rather mlp features for a discriminatively trained gaussian hmm: a comparison for offline handwriting recognition. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 3541–3544. IEEE, 2011.

[13] H. El Abed and V. Märgner. Icdar 2009-arabic handwriting recognition competition. *International Journal on Document Analysis and Recognition (IJDAR)*, 14(1):3–13, 2011.

[14] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):767–779, 2011.

[15] J. G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 347–354. IEEE, 1997.

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

[17] V. Goel, A. Mishra, K. Alahari, and C. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 398–402. IEEE, 2013.

[18] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[19] A. Gordo. Supervised mid-level features for word image representation. *arXiv preprint arXiv:1410.5224*, 2014.

[20] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552, 2009.

[21] E. Grosicki and H. El-Abed. ICDAR 2011: French handwriting recognition competition. In *Proc. of the Int. Conf. on Document Analysis and Recognition*, pages 1459–1463, 2011.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[23] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[24] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, pages 1–20, 2014.

[25] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[26] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Computer Vision–ECCV 2014*, pages 512–528. Springer, 2014.

[27] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[28] M. Kozielski, P. Doetsch, and H. Ney. Improvements in rwth's system for off-line handwriting recognition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 935–939. IEEE, 2013.

[29] A. Lawgali, M. Angelova, and A. Bouridane. A framework for arabic handwritten recognition based on segmentation. *International Journal of Hybrid Information Technology*, 7(5):413–428, 2014.

[30] 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.

[31] V. Margner and H. Abed. Icdar 2011-arabic handwriting recognition competition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1444–1448.

[32] V. Märgner and H. E. Abed. Icfhr 2010-arabic handwriting recognition competition. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 709–714. IEEE, 2010.

[33] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[34] F. Menasri, J. Louradour, A. Bianne-Bernard, and C. Kermorvant. The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition. In *Proceedings of SPIE*, volume 8297, 2012.

[35] E. Mendelson. Abbyy finereader professional 9.0. *PC Magazine*, 2008.

[36] R. Messina and C. Kermorvant. Over-generative finite state transducer n-gram for out-of-vocabulary word recognition. In *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*, pages 212–216. IEEE, 2014.

[37] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. In *BMVC 2012-23rd British Machine Vision Conference*. BMVA, 2012.

[38] Y. Movshovitz-Attias, Q. Yu, M. C. Stumpe, V. Shet, S. Arnoud, and L. Yatziv. Ontological supervision for fine grained classification of street view storefronts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1693–1702, 2015.

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

[40] M. Pechwitz, S. Maddouri, V. Mägner, N. Ellouze, and H. Amiri. Icdar 2007 arabic handwriting recognition competition. In *Colloque International Francophone sur l'Ecrit et le Document (CIFED), Hammamet, Tunis*, 2002.

[41] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, H. Amiri, et al. Ifn/enit-database of handwritten arabic words. Citeseer.

[42] M. Pechwitz and V. Maergner. Hmm based approach for handwritten arabic word recognition using the ifn/enit-database. In *null*, page 890. IEEE, 2003.

[43] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer, 2010.

[44] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE, 2014.

[45] 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 preprint arXiv:1507.05717*, 2015.

[46] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE, 2003.

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[49] F. Stahlberg and S. Vogel. The qcri recognition system for handwritten arabic. In *Image Analysis and Processing—ICIAP 2015*, pages 276–286. Springer, 2015.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[51] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. June 2014.

[52] H. Vinod. Canonical ridge and econometrics of joint production. *Journal of Econometrics*, 4(2):147 – 166, 1976.

[53] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011.

[54] K. Wang and S. Belongie. *Word spotting in the wild*. Springer, 2010.

[55] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[56] C. Yao, X. Bai, B. Shi, and W. 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.

[57] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1637–1644. IEEE, 2014.