Recognizing Handwritten Characters

Lisa Yan
Stanford University
yanlisa@stanford.edu

Abstract

Traditional CNN-based text recognition problems have either been single-character classification like MNIST or complex, stateful full-word classifications like the LSTM approach in Graves et al. I explore a stateless approach to word recognition by using Faster R-CNN's region proposal network in conjunction with a heuristic to produce word labels from region proposals. While the results are promising for word images that have characters with very little overlap, they fall short for cursive word images, where individual characters require surrounding context to be identified.

1. Introduction

Optical Character Recognition (OCR), or the process of identifying letters and words for images of handwritten or typed characters, is a heavily researched area. While OCR is widely used, there is not a generally accepted, contemporary method for performing OCR. Moreover, commercial OCR platforms often take in a large amount of training data on an individual before testing the individual's handwritten text. It is uncommon for OCR platforms to perform well on handwriting from an individual for which training data is nonexistent.

Hidden Markov Models[17] have traditionally been the underlying platform for older OCR software, due to the way HMMs exploit spatial-temporal information and expert information. Yet since HMMs are generative models, they require some prior modeling of the hidden variables in the word classification problem, whereas discriminative models like neural networks need only retain information pertinent to the word classification variables themselves. Furthermore, while HMMs are stateful, they depend only on the current state, and do not incorporate information from characters earlier or later in the word image.

Convolutional neural networks (CNNs), on the other hand, also require training data to classify images but often perform well on new images due to their complex model weights. Single-character recognitions use relatively simple CNNs [2, 11] to achieve high accuracy. Yet word (multi-character) recognition is much more complex, due to both the tasks of per-character detection and the spatial information on character order. In order to achieve word recognition, neural networks must also perform object detection in conjunction with object classification.

In 2009, Long Short-Term Memory (LSTM) systems such as that from Graves *et al*. have been found to achieve high accuracy rates as well [5, 16]. These systems incorporate bidirectional image information to predict the next character in the image. On the Unipen [6] word dataset, an HMM model had a 64.5% word accuracy classification, whereas the LSTM model had approximately a 74.1% accuracy.

While LSTMs have been relatively effective in the word classification problem, object localization has greatly improved since Graves *et al*. [4, 14]. In this project, I explore the feasibility of a stateless CNN architecture in word detection. In particular, I want to see whether an object segmentation followed by character classification approach is sufficient for word recognition.

I adapt the object detection and classification model, Faster R-CNN [14], to train on words and characters. Then, I make the assumption that English words are read from left-to-right to decide the ordering of characters from the Fast R-CNN output. As a result, my main contribution in this project will not be designing a complex CNN, but rather seeing the feasibility of an existing architecture given the correct data.

In this paper, I demonstrate that an object detection approach is sufficient for identifying printed, non-cursive words but falls short of handwritten cursive words. In Section 2 I describe how I process the dataset to feed into Faster R-CNN, the Faster R-CNN architecture itself, and the heuristic to create word labels on the Faster R-CNN output. I then discuss experiment results in detail in Section 3 and conclude in Section 4.

2. Approach

The main obstacles in this research were finding a dataset that had the right labels for any region proposal neural net-

work. Most region proposal networks require bounding box information, which is absent in all of the existing word databases. Since hand-labeling bounding boxes is time-consuming, I also describe how I incorporated computer-generated word images to train the neural network. I then discuss the Faster R-CNN architecture and the word generation heuristic, followed by a short discourse on appropriate metrics for word accuracy.

2.1. Dataset

There are several datasets that provide single-character images or multi-character (word) images [2, 11, 12]. The MNIST dataset is comprised of printed numerical digits [11], while the Chars74K dataset has mostly printed characters and words, as well as a small set of written digits and characters [2]. The Unipen [6] and IAM [12] datasets have a plethora of multi-character word images, though they do not come in the bounding box format that is required of region proposal networks. However, the Unipen dataset is intended for online word recognition—that is, real-time recognition per character over time. Table 1 summarizes these datasets.

Computer-generated words. For this project, I identify the bounding boxes of the handwritten word images from the IAM dataset. Yet due to the time constraints of this project, I also use computer-generated word images to train and test my model. Some word images from this dataset are shown in 1a and 1b.

I generate word images using the Chars74K handwritten character images. I first find the bounding boxes per character, then I concatenate the character images with some amount of overlap and generate words obtained from the IAM word dictionary. The initial set of computer-generated words (so-called *clean* computer-generated characters) have minimal inter-character overlap, and the size of the characters are not resized to equal scale, nor are letters that extend above and below the handwriting line (e.g., 'h', 'l', 'g', 'p', etc.) are not scaled correctly (Figures 1c and 1d).

To correct for the unnaturalness of the clean computer-generated word set, I generate an additional set of words (*noisy* characters) that look closer to a handwritten word by shifting and scaling indidivual characters and increasing the inter-character overlap. I also add noise to the image to imitate the noise levels of the IAM dataset words (Figure 1e and 1f).

Table 2 summarizes the data used in this project. Figure 1 shows sample word images from these project.

2.2. Framework

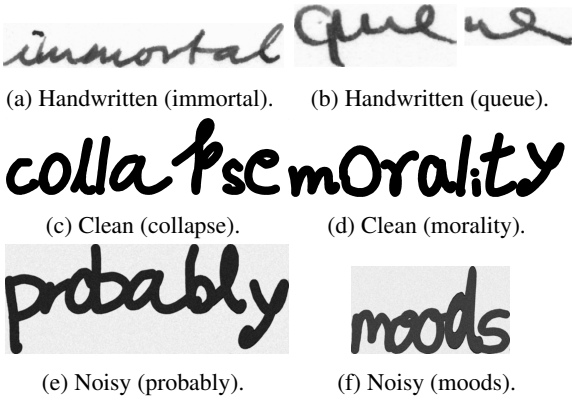
The Faster R-CNN model [14] for object detection and classification is the main model used in this project. This model incorporates a region proposal network (RPN) to suggest bounding boxes for objects within an image [14],

Dataset	Handwritten		Typed	
	Words	Characters	Words	Characters
Chars74K	×	3410	1000	70697
IAM	115320	–	×	×
Unipen	13119	–	×	×

Table 1: Existing datasets. Characters are alphanumeric and case-sensitive. IAM and Unipen consist only of words.

Word Type	Train	Test
Handwritten	266	29
Computer (clean)	2000	500
Computer (noisy)	2000	500
Total	4266	1029

Table 2: Dataset used in this project.



(a) Handwritten (immortal). (b) Handwritten (queue).

(c) Clean (collapse). (d) Clean (morality).

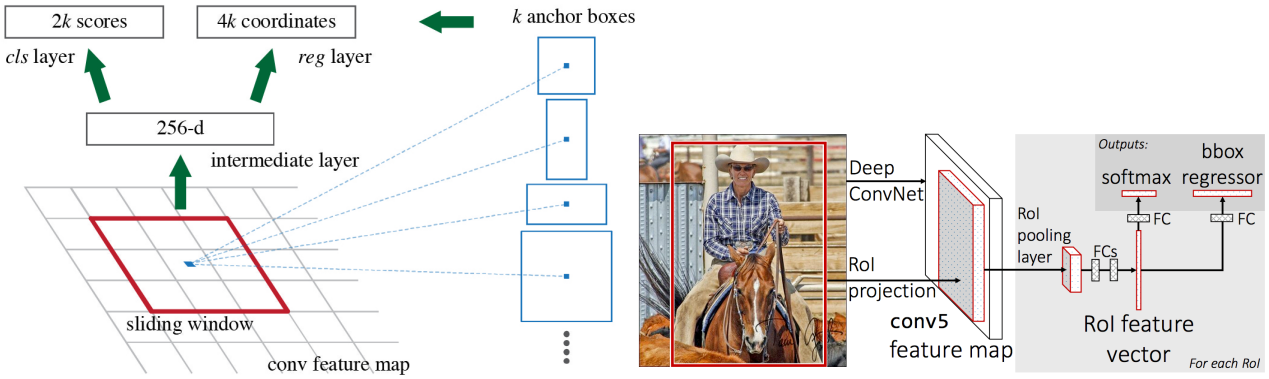
(e) Noisy (probably). (f) Noisy (moods).

Figure 1: Sample images (image type and word label).

followed by Fast R-CNN, which takes the bounding box input to classify each box into a set of object classes [4].

Faster R-CNN. The RPN in Faster R-CNN (Figure 2a) uses the output from the last convolutional layer and slides a small $n \times n$ window across to produce m *sliding windows*. Then, k *anchors* are assigned per sliding window, where each anchor specifies a different scale and aspect ratio for region proposals centered at that sliding window. In the Faster R-CNN architecture, $n = 3$ and $k = 9$. Then, the sliding window is passed through a layer to produce a lower resolution intermediate layer, which finally outputs region proposal defined by two layers, cls and reg. The cls layer gives a score of whether or not an object exists within each of the k anchors, while the reg produces the *bounding box* for the region proposal, defined by 4 coordinates (x_{min}, y_{min}, x_{max}, y_{max}).

The Faster R-CNN passes the mk region proposal outputs from the RPN into the Fast R-CNN network (Figure



(a) Region Proposal Network (Figure from Faster R-CNN [14]). (b) Fast R-CNN network (Figure from Fast R-CNN [4])

Figure 2: **The Faster R-CNN network.** (a) **Region Proposal Network (RPN).** The sliding windows output by the last convolutional layer are fed into the RPN to produce $k = 9$ anchor boxes (3 scales, 3 aspect ratios). Each of these anchor boxes is fed into a box-regression layer (reg) outputting the coordinates of k boxes ($4k$ outputs) and a box-classification layer (cls) outputting whether or not the k boxes contain objects ($2k$ outputs). (b) **Fast R-CNN.** Each Region of Interest (RoI) proposal is pooled then fed into the fully connected layers and classified independently.

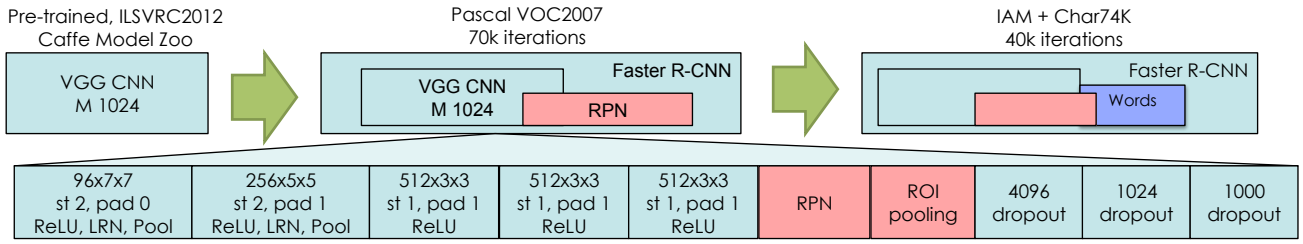


Figure 3: **Overview of CNN training procedure and Faster R-CNN architecture**. Local response normalization (LRN) [10] is performed on the first two convolutional layers, and the output of the RPN feeds into RoI pooling to produce equally-sized output prior to the fully connected layers.

2b) to produce mk softmax classifications per image. The RPN output is first passed through an RoI pooling layer [7], which scales each region proposal into equally sized inputs to feed into the fully connected layers for classification.

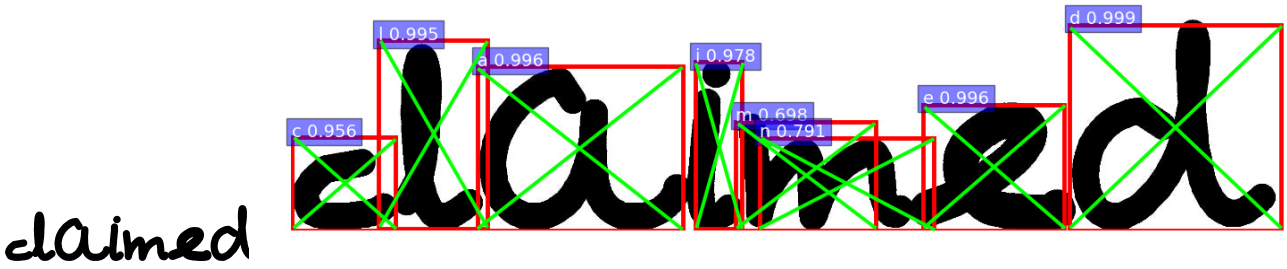
I utilize the approximate joint training procedure to simultaneously train RPN and Fast R-CNN. The RPN and Fast R-CNN share convolutional features, whose backward passes are calculated as the combination of the RPN and Fast R-CNN loss. For each forward pass, the region proposals are treated as fixed and pre-computed. The RoI pooling layer, whose input is both the convolutional feature and the bounding box proposals, thus avoids computing the non-trivial derivative with regards to the bounding box proposals.

Training procedure and architecture. Figure 3 shows the training procedure and final Faster R-CNN architecture. I begin with the VGG_CNN_M_1024, the VGG medium architecture with last fully-connected layer dimension 1024 from the Caffe Model Zoo [1, 9]. This pre-trained CNN is then appended with the RPN to create a Faster R-CNN and trained for 70,000 iterations on the Pascal VOC 2007 dataset [3], a standard image dataset for detecting 20 classes of objects in various images.

Finally, I use transfer-learning and train the network on my word image dataset, which has 62 classes todal (10 numeric digits and 26 upper- and lower-case English characters). The output of Faster R-CNN for word image is a set of region proposals each specified by 4 bounding box coordinates, class label (alphanumeric: 0-9, a-z, A-Z), and *confidence*, where confidence is the probability of the top class label.

2.3. Word generation

I want to generate words with associated confidences from the set of region proposals output by Faster R-CNN for each image. The process of generating words is described as follows but are also illustrated in Figure 4 for convenience.



(a) (b)

x-coordinate	Class	Confidence	$(x_{min}, y_{min}, x_{max}, y_{max})$
139	c	0.995	(21, 339, 257, 547)
343	l	0.994	(225, 77, 462, 547)
713	a	0.992	(459, 146, 967, 547)
1046	i	0.990	(986, 141, 1107, 524)
1313	m	0.853	(1114, 310, 1513, 547)
1325	h	0.444	(1069, 232, 1582, 546)
1373	n	0.846	(1197, 316, 1550, 547)
1703	e	0.994	(1485, 250, 1921, 547)
2145	d	0.996	(1893, 33, 2398, 547)

(c)

Word	Confidence	Edit distance
claimed	0.967	0
clained	0.966	1
claihed	0.909	1

(d)

Figure 4: **Word generation heuristic.** (a) **Initial word image.** Ground truth word is "claimed." (b) **Image with bounding boxes.** The green X's mark the center of masses. Note that the characters 'm' and 'n' overlap. (c) **Character data.** After performing NMS and confidence-thresholding (threshold 0.4) the characters are sorted by the x dimension of the bounding box's center of mass. (d) **Word proposals.** The words are sorted by average character confidence; edit distance from the ground-truth is also shown.

I first remove some overlapping bounding boxes via a non-maximum suppression (NMS) algorithm [13]. Then I further eliminate the bounding boxes by removing those with low confidence. I order the remaining bounding boxes by center-of-mass with increasing horizontal dimension (centered x coordinates left-to-right). I group together bounding boxes whose centers of mass are contained within each other (Figure 4b).

I then string together the characters from left-to-right, permuting if the next character can be selected from the groups created in the previous step (Figure 4c). These words are then assigned the average confidence of the characters' region proposals (Figure 4d). The top word and the top five words are proposed as possible word choices for this word image.

Edit distance. Since the characters in a word image are classified independently, my metric for accuracy should not simply be the word with top confidence. Instead, I should look for a more nuanced measure based on the number of accurate characters in a proposed word, as compared to the original ground truth character. I thus also measure the Levenshtein *edit distance* of each proposed word [8]. The edit distance is the minimum number of operations required to transfer one character string into another by adding, removing, or changing characters. The edit distance between two words a and b is given by $lev_{a,b}(|a|, |b|)$, where $|a|$ and $|b|$ are the respective word lengths, and

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+\mathbb{1}(a_i \neq b_j) \end{cases} & \text{otherwise.} \end{cases}$$

For example, in Figure 4d the edit distance of the third proposal, "claihed," to the ground-truth word is one character—by changing the 'h' into an 'm', we can recover the original word "claimed."

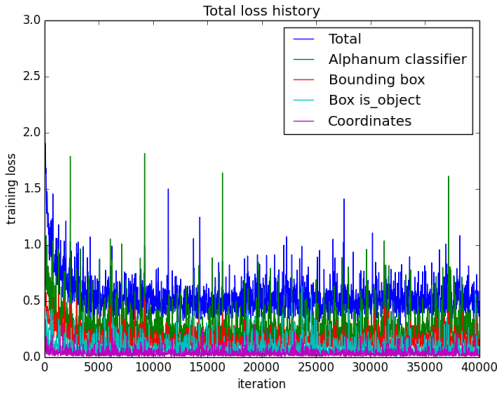
3. Experiment

I test the performance of the Faster R-CNN network itself and also the accuracy of the word generation based on Section 2.3's heuristic.

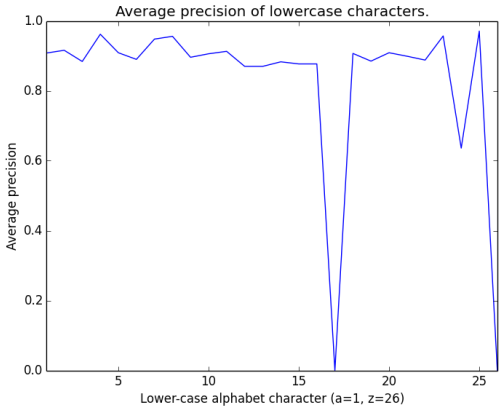
3.1. Experiment setup and results

To evaluate my word recognition system (Figure 3), I first obtained a Faster R-CNN architecture based on VGG_CNN_M_1024 trained on Pascal VOC 2007 for 70,000 iterations, then trained the model on the hybrid IAM and Chars74K training dataset (Table 2) for 40,000 iterations. Finally, the trained model was tested on the hybrid

IAM and Chars74K test dataset. All methods were implemented with Caffe [9] and were trained on a NVIDIA GRID K520 GPU. Figure 5a shows the training loss on the word image dataset; the final network took approximately 12 hours to train.



(a)



(b)

Figure 5: **Model performance.** (a) **Loss function over 40k iterations**. The total loss (Total) is the aggregation of the alphanumeric classifier (Alphanum classifier), the RPN loss (Bounding box), cls for whether the region contains an object (Box is_object), and the reg for the bounding box coordinates (Coordinates). (b) **Average precision (AP) on test set.** Only lowercase characters (a-z) are shown; 'q' and 'z' were not in the test set.

The average precision (AP) metric is used primarily in the VOC 2007 competition [3, 15] as a metric for object detection. AP relates the order and confidence of the classification proposals per bounding box to the classes themselves across all images. The APs of each class for the test dataset is shown in Figure 5b. The mean average precision over all alphanumeric classes is 0.8965, with the lowest AP being 'x' (0.636) and the highest APs being 'y' (0.9715) and 'd' (0.9621).

Dataset	Image Type	Top-1 Acc.	Top-5 Acc.	Avg. Confidence	Avg. edit distance
Train	All	0.617	0.660	0.952	0.553
	Hand	0.223	0.312	0.775	1.61
	Clean	0.778	0.823	0.967	0.271
	Noisy	0.509	0.544	0.960	0.694
Test	All	0.603	0.655	0.957	0.575
	Hand	0	0.138	0.776	2.48
	Clean	0.780	0.834	0.968	0.260
	Noisy	0.462	0.506	0.956	0.780

Table 3: **Word accuracy results.** The average confidence and edit distance are for the top-1 word.

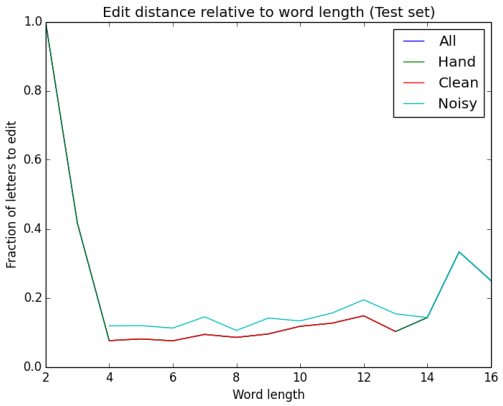
The output of the Fast R-CNN network was then fed into a series of Python scripts to produce final word proposals as described in Section 2.3. Table 3 shows the top-1 and top-4 accuracies, as well as the edit distance accuracies, for both the training set and the validation set. The results are listed by the three types of images: handwritten, clean computer-generated, and noisy computer-generated.

3.2. Analysis and comments

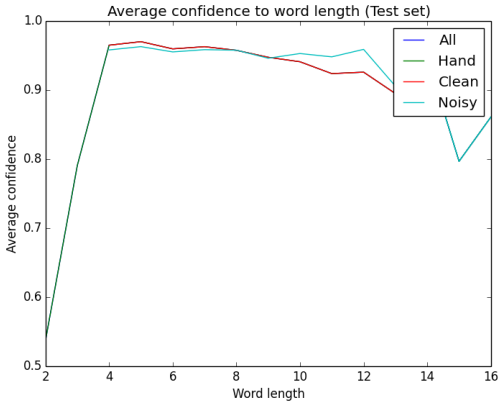
As shown in Table 3, the model performs very suboptimally when the top confidence word proposal is considered. Including the top five word proposals gives a better result. Evidently, the classifier performs best on the clean computer-generated word images, which have little inter-character overlap and no noise. Noisy computer-generated word images gave considerably lower accuracies, whereas handwritten word images performed the worst.

The edit distance metric gives a better notion of how accurate the model is for as the word length varies. Figure 6a shows the ratio of the edit distance to the word length. Evidently, while the two-character words (e.g., "to", "of", etc.) are almost completely incorrect, for the most part the edit distance for the average word length is small. More notably, the average edit distance is less than one character on average for the clean computer-generated images, suggesting that the model performs very well on the clean computer-generated images. Furthermore, the edit distance metric seems follow very closely to the overall confidence of the top word proposal per word length, as shown in Figure 6b; higher edit distances correspond to lower confidence levels.

Some sample output images are shown in Figure 7. Evidently, the clean images perform very well (Figure 7d). While the noisy images have decent region proposals, it is worthwhile to point out Figure 7f, which has bounding boxes around 'l' and 'a' as expected but also has a curious box around 'l' *and* 'a' and labels it 'b' with high confidence (0.751). A human would not expect this as he/she expects cursive b's to have a different connection to the following letter, and would identify the 'a' before the 'l' due to the tail of the 'a'. On a similar note, in Figure 7e the network



(a)



(b)

Figure 6: Edit distance (a) and confidence (b) by word length in the test set.

identifies a 'c' within the 'a'; this is perhaps due to the RPN also allowing vertical classification of characters, whereas human English readers would only read left to right and would ignore the hidden 'c' in favor of the larger 'a'.

The handwritten images tend to have more variability per-character. Overall, Faster R-CNN performs relatively well on printed handwritten words (Figure 7a) but fails miserably on cursive, connected words. In Figures 7b and 7c, the 'u', 'm', and 'n' all look very similar when cropped from the image as a whole, and the network does not even propose them as characters. However, a human reader can identify the word as a whole due to the surrounding characters.

These observations suggest that the neighboring characters are crucial to the identification of any single character. In fact, many times a human English reader will be able to identify words based on merely selecting a few characters within the word and noting their left-to-right order. However, an RPN will discard this neighboring information early on in its architecture and therefore can only classify words independently.

4. Conclusion

The main takeaway of this project is that while region proposal networks are sufficient in recognizing words when the characters have high separation (e.g., printed, non-cursive words), they fall short in cursive handwriting recognition, where large amounts of context are required. Furthermore, a major caveat of region proposal-based networks is labeling bounding boxes.

Even considering these two issues, I still found this project useful for finding out exactly why context-free word classification is insufficient. Furthermore, validating the performance of Faster R-CNN is beneficial in a general sense to make sure that the region proposal architecture can be applied to different types of object detection problems.

References

[1] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[2] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[3] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.

[4] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[5] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, May 2009.

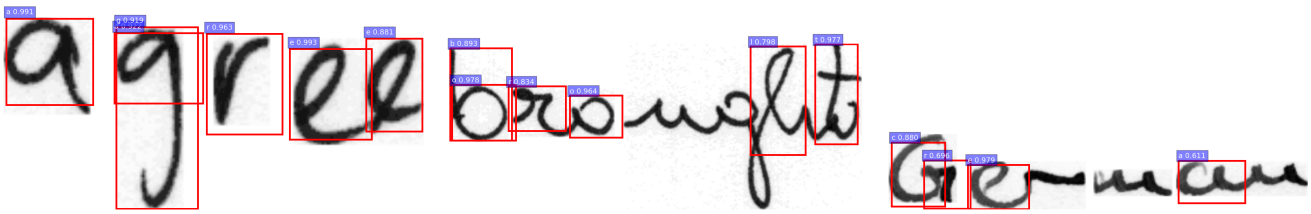
[6] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proc. 12th Intl Conf. Pattern Recognition*, pages 29–33, 1994.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

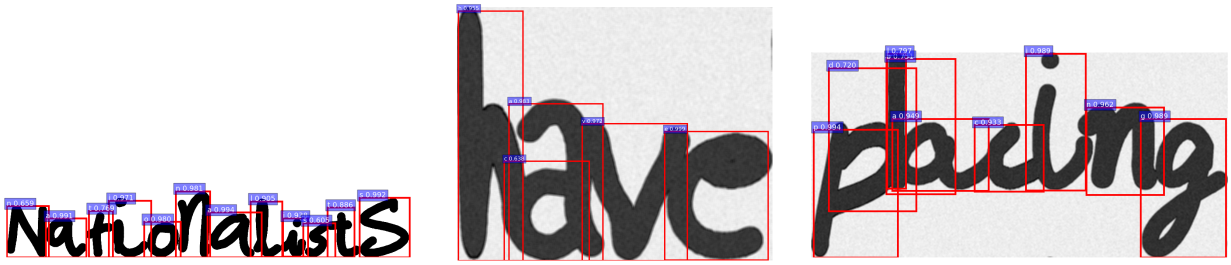
[8] H. Hyyr. Explaining and extending the bit-parallel approximate string matching algorithm of myers. Technical report, 2001.

[9] 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.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.



(a) Handwritten (agree). (b) Handwritten (brought). (c) Handwritten (German, from training set).



(d) Clean (Nationalists). (e) Noisy (have). (f) Noisy (placing, from training set).

Figure 7: **Sample outputs**. Bounding boxes are shown around each image; all images are taken from the test set except for images (c) and (f). Ground truths for each word image are in parentheses.

[11] 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.

[12] U. Marti and H. Bunke. The IAM-database: An english sentence database for off-line handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 2002.

[13] A. Neubeck and L. V. Gool. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855, 2006.

[14] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[15] G. Salton and M. J. McGill. In *Introduction to modern information retrieval*, New York, NY, USA, 1986. McGraw-Hill.

[16] A. Ul-Hasan and T. M. Breuel. Can we build language-independent OCR using LSTM networks? In *Proceedings of the 4th International Workshop on Multilingual OCR*, MOCR '13, pages 9:1–9:5, New York, NY, USA, 2013. ACM.

[17] M. Zimmermann and H. Bunke. Automatic segmentation of the IAM off-line database for handwritten english text. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 35–39, 2001.